

Dear student

There are significant efforts behind the completion of this book, in the process of its preparation, review, and development, and there are funds spent to print it and deliver it to you, to support your learning and your scientific and moral advancement, so, try to be faithful to this effort, appreciating it by preserving your book.



وزارة التعليم
Ministry of Education
2026 - 1448

Deposit number: **1446/19529**
ISBN: **978-603-514-078-2**

Software Engineering

Third year - Secondary stage - Pathways System

1448 - 2026 Edition

المملكة العربية السعودية

رؤية
2030
المملكة العربية السعودية
KINGDOM OF SAUDI ARABIA



وزارة التعليم
Ministry of Education



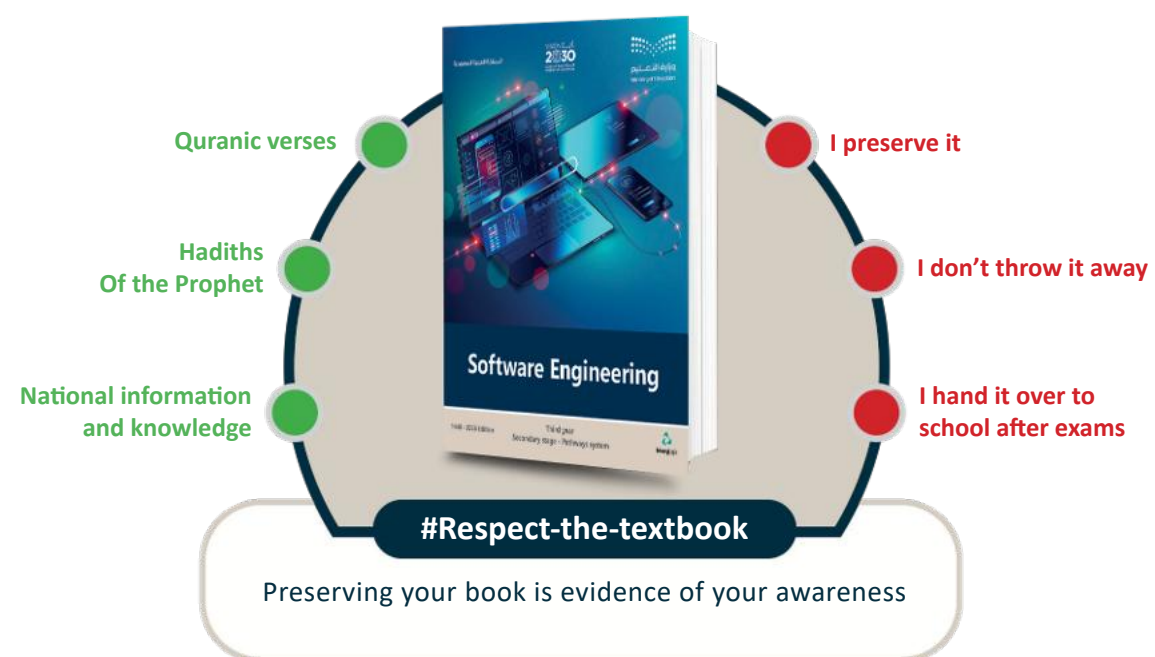
Software Engineering

1448 - 2026 Edition

Third year
Secondary stage - Pathways system



School:



قررت وزارة التعليم تدريس
هذا الكتاب وطبعه على نفقتها



المملكة العربية السعودية

Software Engineering

Secondary stage - Pathways system

Third year



وزارة التعليم
Ministry of Education
2026 - 1448

The book is distributed freely and cannot be sold.

1448 - 2026 Edition

Publisher: Tatweer Company for Educational Services

Published under a special agreement between Binary Logic SA and Tatweer Education Services Company (Contract No. 0003/2022) for use only in the Kingdom of Saudi Arabia

Copyright © 2026 Binary Logic SA

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without permission in writing from the publishers.

Please note: This book contains links to websites that are not maintained by Binary Logic. Although we make every effort to ensure these links are accurate, up-to-date and appropriate, Binary Logic cannot take responsibility for the content of any external websites.

Trademark notice: Product or corporate names mentioned herein may be trademarks or registered trademarks and are used only for identification and explanation without intent to infringe. Binary Logic disclaims any affiliation, sponsorship, or endorsement by the respective trademark owners. Tinkercad is a registered trademark of Autodesk Inc. "Python" and the Python logos are registered trademarks of Python Software Foundation. Jupyter is a registered trademark of Project Jupyter. CupCarbon is a registered trademark of CupCarbon. Arduino is a registered trademark of Arduino SA. "MIT App Inventor", "MIT", "Massachusetts Institute of Technology", and its logos and seal are trademarks of the Massachusetts Institute of Technology. Pencil Project is a trademark of Evolus.

The above companies or organizations do not sponsor, authorize, or endorse this book.

The publisher has made every effort to trace all copyright holders, but if they have inadvertently overlooked any they will be pleased to make the necessary arrangements at the first opportunity.



©National Curriculum Center, 2025

National Curriculum Center

Software Engineering. / National Curriculum Center

Riyadh, 2025

224p.; 210*25.5cm

L.D. no.: 1446/19529

ISBN: 978-603-514-078-2

Copyright reserved to the Ministry of Education

www.moe.gov.sa

Enrichment and support materials on the “iEN Ethraia Platform”



ien.edu.sa

Dear students, parents and anyone interested in education, we welcome your communication to improve our textbooks. Your suggestions are our top priorities.



fb.ien.edu.sa



Introduction:

The progress and development of countries is measured by the ability to invest in education, and the extent to which their educational system responds to the requirements and changes of the generations. In the interest of the Ministry of Education sustaining the development of its educational systems and in response to the vision of the Kingdom of Saudi Arabia 2030, the Ministry of Education has taken the initiative to adopt the “Secondary Education Pathways” system to bring about an effective and comprehensive change in high school education.

The secondary education pathways system provides a distinguished and modern educational model for high school in the Kingdom of Saudi Arabia, which efficiently contributes to:

- Strengthening the values of belonging to our homeland “the Kingdom of Saudi Arabia” and loyalty to its wise leadership “may God protect him” based on a pure belief supported by the tolerant teachings of Islam.
- Strengthening the values of citizenship by focusing on them in school subjects and activities, in line with the demands of sustainable development, and the development plans in the Kingdom of Saudi Arabia that emphasize the consolidation of both values and identity, based on the teachings of Islam and its moderation.
- Qualifying students in line with future specializations in universities or the required jobs; ensuring the consistency of education outputs with the labor market requirements.
- Enabling students to pursue education in their preferred path at early stages, according to their interests and abilities.
- Enabling students to join specific scientific and administrative disciplines related to the labor market and future jobs.
- Participation of students in an enjoyable and encouraging learning environment in school based on a constructive philosophy and applied practices within an active learning environment.
- Delivering students through an integrated educational journey from the primary level to the end of the high school level and facilitating their transition process to post-general education.
- Providing students with technical and personal skills that help them deal with life and respond to the requirements of their level.
- Expanding opportunities for graduate students through various options in addition to universities, such as: obtaining professional certificates, joining applied faculties, and earning job diplomas.

The pathways system consists of nine semesters that are taught over three years, including a common first year in which students receive lessons in various scientific and humanities fields, followed by two specialized years, in which students study a general path and four specialized paths consistent with their interests and abilities, which are: the Rightful path, Business Administration path, Computer Science and Engineering path, Health and Life path, which makes this system the best for students in terms of:

- The existence of new study subjects that match the requirements of the Fourth Industrial Revolution and development plans, and the Kingdom’s Vision 2030, which aims to develop higher-order thinking, problem-solving, and research skills.
- Elective field programs that are consistent with the needs of the labor market and students’ interests, as they enable students to join a specific elective field according to a specific job skill.
- Scale as it ensures the achievement of students’ efficiency and effectiveness, and helps them identify their tendencies and interests, and reveals their strengths, which enhances their chances of success in the future.

- Volunteer work designed specifically for students in line with the philosophy of activities in schools, and is one of the graduation requirements; which helps to promote human values, and build society (its development and cohesion).

- Bridging which enables students to move from one path to another according to specific mechanisms.
- Proficiency classes through which skills are developed and the achievement level improved, by providing enrichment and remedial mastery classes.
- The options of integrated learning and distance learning, which are built in the paths system based on flexibility, convenience, interaction and effectiveness.
- The graduation project that helps students integrate theoretical experiences with applied practices.
- Professional and skill certificates granted to students after completing specific tasks, and certain tests compatible with specialized organizations.

Accordingly, the Computer Science and Engineering Pathways, as one of the updated paths at the secondary level, contributes to achieving best practices by investing in human capital and transforming the student into a participating and productive individual for science and knowledge while providing him with the skills and experience necessary to complete his studies in fields that meet his interests and abilities or to join the labor market.

Software Engineering is one of the main subjects in the Computer Science and Engineering Pathways course. It helps students learn the basics of software engineering by engaging and participating in discovering a wide variety of topics in the field. This book provides an overview of the Software Development Life Cycle and discusses the main concepts of Human-Computer Interaction and prototyping. Additionally, the student learns to design and develop a mobile application with accessibility in mind. There are also realistic exercises for the student to solve that stimulate his cognitive levels under the guidance and supervision of the teacher.

The Software Engineering book is characterized by modern engagement methods which make students able to learn from and interact with it through the various exercises and projects it provides. This book also emphasizes important aspects of Software Engineering education and learning which are:

- The connection between the content and real-life problems.
- Diversity of ways to display engaging content.
- Highlighting the role of the learner in the teaching and learning processes.
- Attention to the content structure and coherence.
- The skill of employing appropriate techniques in different situations.
- The ability to employ various methods in evaluating students in proportion to their individual differences.

To be on pace with global developments in this field, the Software Engineering book will provide the teacher with an integrated set of diverse educational materials that take into account the individual differences between students, in addition to educational software and websites which provide students with the opportunity to employ modern technologies and practice-based communication; This solidifies their role in the teaching and learning process. As we present this book to our dear students, we hope they will capture their interest, meet their requirements, and make learning this material more enjoyable and useful.

God grants success



Contents

1. Software Engineering 8

Lesson 1	Principles of Software Engineering	9
	Exercises	21
Lesson 2	Programming Languages and Languages Processors	28
	Exercises	39
Lesson 3	Software Development Tools	42
	Exercises	55
Project		60

2. Prototyping 62

Lesson 1	Analysis	63
	Exercises	79
Lesson 2	Interaction Between the User and the Computer	84
	Exercises	91
Lesson 3	Creating a Prototype	95
	Exercises	109
Project		112

3. Developing Applications with App Inventor .. 114

Lesson 1	Introduction to MIT App Inventor	115
	Exercises	133
Lesson 2	Adding More Elements to the App	135
	Exercises	151
Lesson 3	Programming the Mobile App	153
	Exercises	180
Project		182

4. Software Accessibility and Digital Inclusion 184

Lesson 1	Testing and Deploying Applications	185
	Exercises	190
Lesson 2	Digital Inclusion	192
	Exercises	200
Lesson 3	Accessibility Features in an Application	203
	Exercises	219
Project		220



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



1. Software Engineering

In this unit, students will be introduced to the most common software development methodologies. They will learn about the life cycle of a system and explain its different phases. Students will also understand the importance of converting high-level programming languages into executable instructions in machine language, and they will learn about the programs used to do this.

Learning Objectives

In this unit, you will learn to:

- > Distinguish the most popular software development methodologies.
- > Recognize the different stages of the software development lifecycle.
- > Create a software development life cycle for an application.
- > Describe the analysis phase of the software development lifecycle.
- > Recognize different requirement collection methods.
- > Classify programming languages and their features.
- > Outline the function of the compiler and the interpreter.
- > Classify different software development tools.
- > Explain what a code editor is and what the advantages and challenges of using one are.
- > Recognize an integrated development environment and explain the advantages and challenges of using one.
- > Recognize how different software development tools can be used to provide different software solutions.





Software Engineering is a branch of computer science that deals with developing and maintaining software systems. It involves the application of engineering principles and practices to software product design, development, testing, and maintenance. Software engineering aims to produce reliable, efficient, and high-quality software that meets the requirements of its stakeholders. This is achieved through systematic and repeatable processes, tools, and techniques. Software engineering also involves the management of the software development process, including project planning, estimation, risk management, and quality assurance.

The Software Development Life Cycle (SDLC)

The Software Development Life Cycle (SDLC) describes the organization of the production processes for systems in various fields. The aim of the SDLC is not limited to improving the final product, but also extends to the management of the production and development processes and the optimal use of resources during these processes. In this lesson, we will discuss the stages of the SDLC in the context of developing systems for Information and Communication Technology (ICT). The SDLC consists of a series of phases, which are illustrated in the figure. You will discover all the phases of the SDLC through an example of building software for a banking system.

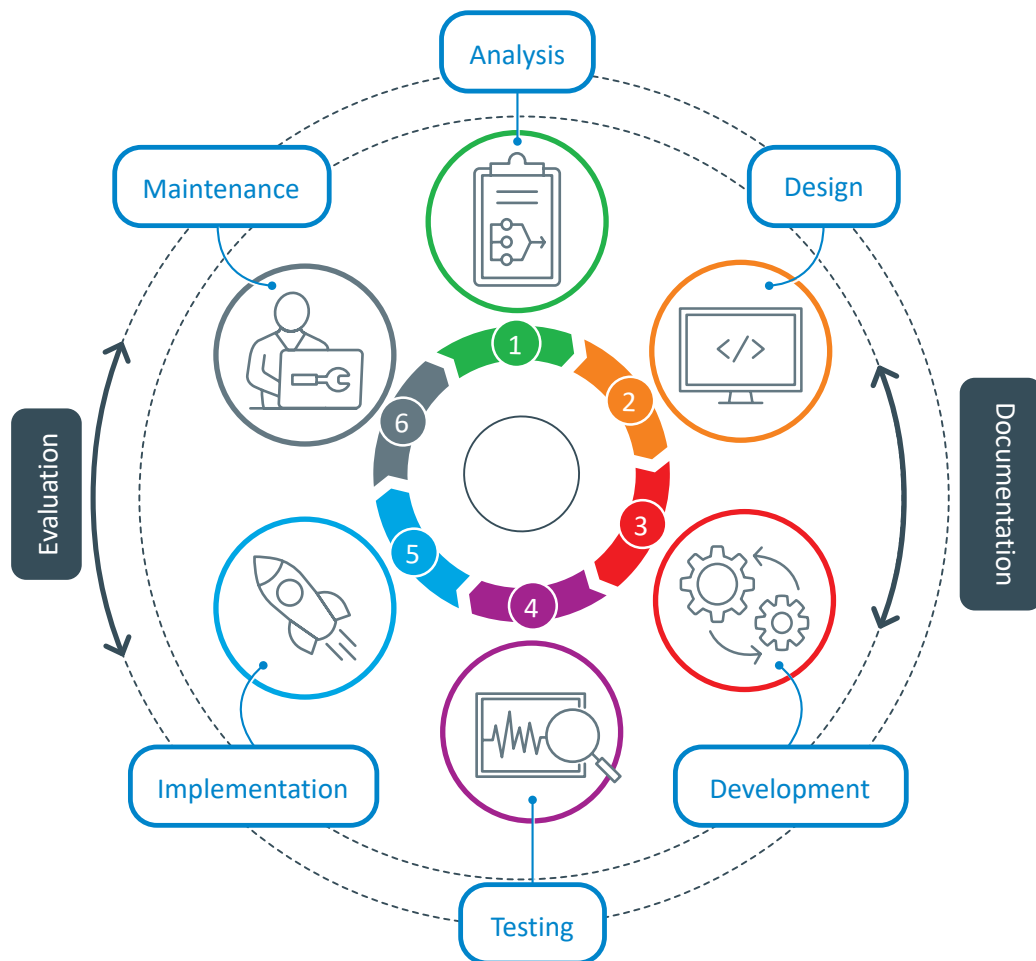


Figure 1.1: The Software Development Life Cycle

Analysis

The first step of the SDLC is to identify the problem that needs to be solved, followed by defining the requirements for its solution as accurately as possible through requirements engineering. Requirements engineering involves analyzing, specifying, validating, and managing the needs and expectations of stakeholders for a software system. It involves understanding the problem domain and determining the functional and non-functional requirements for the software. The difference between functional and non-functional requirements is the following:



- **Functional Requirements:** These are the specific capabilities or features that the software must have to meet its stakeholders' needs. For example, a common functional requirement for an online shopping website is the ability for users to search for products, add them to a shopping cart, and complete a secure checkout.
- **Non-functional Requirements:** These are the constraints and quality attributes the software must meet to be acceptable to its stakeholders. Examples of non-functional requirements include performance requirements (e.g., response time, throughput), security requirements (e.g., data privacy, authentication), and usability requirements (e.g., user-friendly interface).

During the analysis process, all resources (human, material, costs, budget, time available and everything else related to the project) must be taken into consideration, and all functions required for the new system must be defined in detail with reference to any limitations that exist.

The analysis process involves identifying users, and their needs and requirements. The following tools are most commonly used to collect the required data:

- Questionnaires
- Interviews
- Observation

Consider a bank. Its problem is how to establish an electronic system to provide banking services via the Internet. The analysis phase of this project will include collecting data from management and customers about their requirements in order to understand which banking services should be automated, the design required for the user interface, security requirements, digital permissions assigned to bank employees and customers, etc.

Design

The design phase is the second stage of the SDLC, in which the systems analyst participates in providing expertise and skills for building the structure and designing the solution to the given problem.

The design phase defines the different interfaces and data types that are used in the ICT system. More specifically, the following steps are taken:



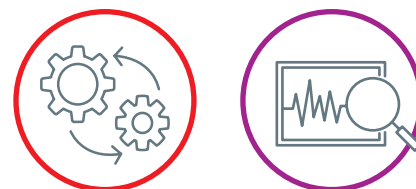
1. Define the flow of data and information in all aspects of the new system.
2. Determine the main data to be processed. This will define the data structures used by the system.

3. Determine where and how data is stored so that it is accessible and secure.
4. Design reports and other data and information outputs.
5. Design the user interface and define the functions of all the elements in it.
6. Design integration interfaces for data exchange with other ICT systems.
7. Determine the method for testing the system, the data used for testing and how to use these in quality assurance.

In the design of an electronic banking system, data flow pathways should be defined between the system and the user and various databases and integrated systems, all of which will depend on the different data types that need to be stored, secured and transferred. The system requirements for data input and output should be defined, and user interfaces for staff and clients should be designed as well as interfaces for data and monetary exchanges with other organizations. Finally, it must be decided what tests must be carried out to ensure the system operates as expected.

Development and Testing

Next come the development and testing phases. After carrying out detailed analysis and design processes, programmers and system testers must now convert the requirements and specifications into code segments using appropriate programming languages or computer programs. The development and testing phases cannot be separated as the system must be comprehensively tested during and after development to ensure all issues are addressed and that the system reaches its end users as per their requirements. Elements of the system that require independent testing include the following:



1. Testing the validity of the entered data:

The entry of invalid data may cause problems within the banking system and it is therefore important to test the validity of the entered data. In the example of electronic banking services, this will require the development of security rules for receiving passwords from users and the number of incorrect attempts allowed when entering passwords, rules for verifying the numbers entered into the system, such as phone numbers and personal numbers, and rules specifying the maximum amounts that can be withdrawn or transferred through the system.

2. Testing system functionality and usability:

This includes testing the user interface and user experience. For example, for the electronic banking system, this will require the formation of a group of stakeholders to test whether the system and its various functions (such as checking balances or making a transaction) are working as they should.



3. Operation Error Testing:

This includes testing for logical errors in the code. For example, in the electronic banking system, it may be necessary to test whether the steps to complete a transaction are done logically, and whether the appropriate messages (such as error and confirmation messages) appear at the appropriate steps.

4. Test communication with other systems:

This depends on the extent to which the system is linked with other systems. For the electronic banking system, this will require testing that the new system integrates well with other information technology systems in the bank, such as customer databases, currency conversion systems, and automated teller systems.

Implementation

After obtaining user approval for the new system that has been developed and tested, the implementation phase begins. This is the phase where the theory is converted into practice as the product is put into service. The system is prepared for deployment and installation at the target site in order to be operational and ready for productivity.

Implementation may involve training end users to ensure they know how to use and familiarize themselves with the system. The implementation phase may take a long time depending on the complexity of the system. Implementation sometimes requires transferring data from the previous system to the new system. It is often preferable to introduce the new system gradually.



If a bank implements a new electronic banking system, the transition may require the deployment of a "beta" version of the system, so that the public can test it and give feedback on the experience, before the final version of the system is implemented.



Figure 1.2: Designing and implementing a software product

Maintenance

Maintenance is necessary to eliminate errors in the system during its working life and tune the system to any variations in its working environments. It must meet the scope of any future enhancement, future functionality and any other added functional features to cope with the latest needs. Through user feedback and the evaluation of the IT team, the system is continuously assessed to ensure that it does not become obsolete. Working with the new system means that some small fixes or adjustments will be required. Needs and requirements change regularly, and during this stage the IT team has to keep everything working as expected.



In the example of an electronic banking system, once the system is implemented, ongoing maintenance will be required to ensure the system remains functional, secure and up-to-date. Systems require many major and minor software and hardware updates to protect against new security threats, fix unforeseen bugs, and implement new functionality. Some system maintenance can be done automatically, such as automated security updates, but other tasks, such as carrying out hardware updates, may require engineers to be present on site.

Documentation

The documentation process includes describing all the details of the analysis, design, development, testing, implementation, and maintenance of the system, and is later used to build a knowledge base of how the system works. System documentation is referred to if any change, repair or adjustment is required, and the documentation itself may then need to be updated. Documentation is important in software development because it promotes communication, transparency, maintenance, compliance, training, and legacy. Documentation helps to ensure that the software system is developed consistently and effectively and that it will continue to meet the needs of its intended users over time.



Evaluation

Each stage of the SDLC must be evaluated. This may involve making some difficult decisions, as a design problem may lead to larger problems later during development or when implementing and using the system.

Among the areas that need continuous evaluation are:

- System efficiency
- Ease of use and learning
- System suitability for the required tasks

Evaluation may be carried out by the following groups to ensure that the system meets the requirements:

- IT team
- Users
- Management

SDLC for a Smartphone Application

Let's say we want to create a mobile application that provides information about different tourist attractions in KSA. More specifically, the application aims to help elderly people with vision problems or trembling hands to browse on the screen for information on tourist sites to visit in KSA.

The application will allow people with vision problems to adjust the font size of the text to suit them so that they can read information easily, and it will also allow those who suffer from shaky hands the opportunity to adjust the size of the buttons to prevent them from incorrectly pressing a button by mistake, and finally, users will be able to change the colors in the application to black and white to facilitate reading and reduce eyestrain.

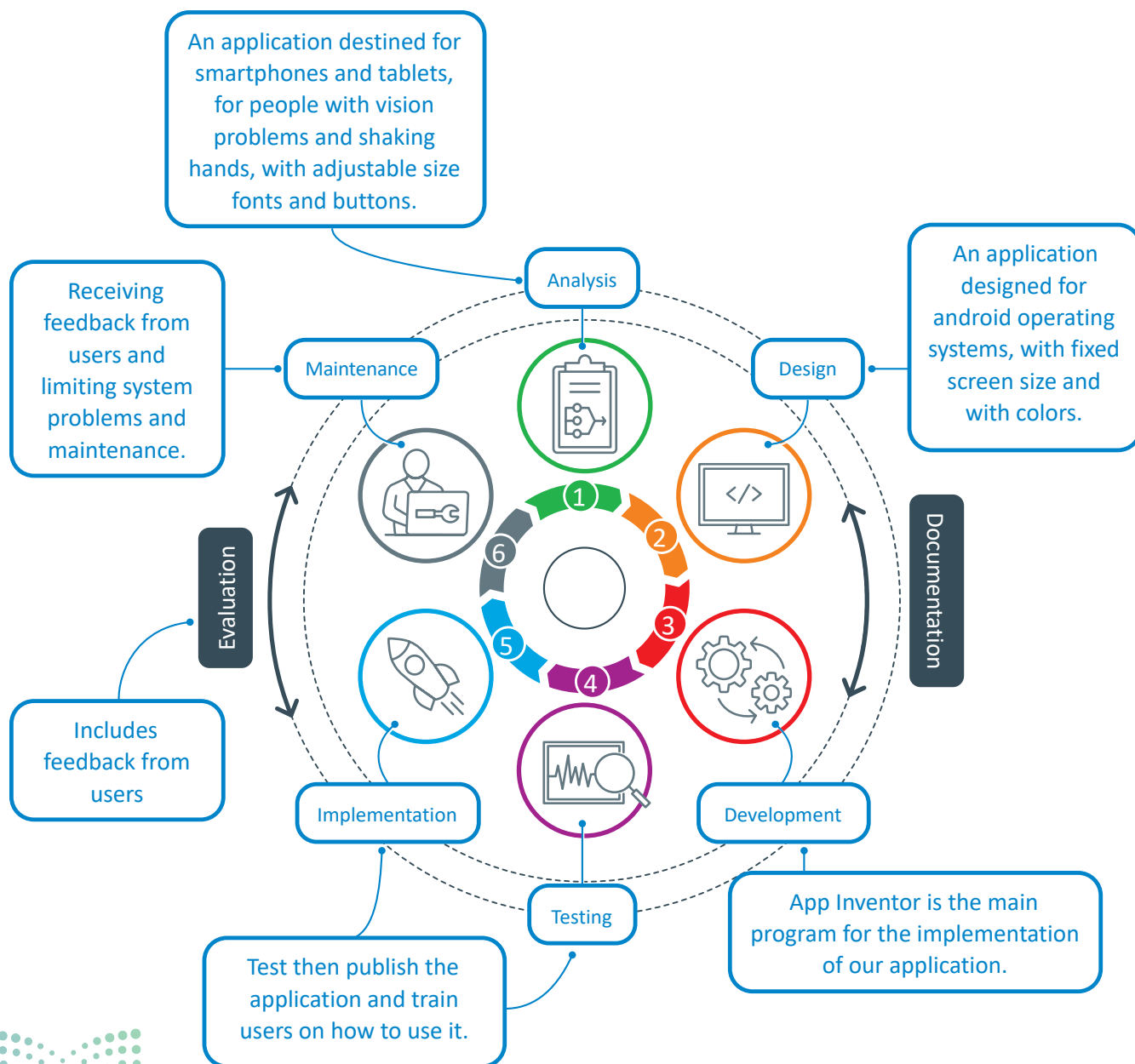


Figure 1.3: Summary of the SDLC of a smartphone application



Based on what you have learned so far in this lesson, the SDLC of this application will include the following phases of **Analysis, Design, Development and Testing, Implementation, Maintenance** and ongoing **Documentation and Evaluation**.

Analysis:

In the analysis phase, we identify the problem that needs solving. The application will be:

- Destined for smartphones and tablets.
- Designed for people with vision problems.
- Designed for people with shaky hands.

Based on these requirements, the applications must have adjustable font and button sizes. Also, for people with shaky hands, the buttons must be very big so that they can be easy to press.

Design:

The design phase will include the determination of all the technical details of our application. More specifically, the technical requirements include:

- The application must be designed for Android operating systems.
- The size of the screen must be fixed.
- It must not have many colors because that would be confusing for the users.

Development, Testing, Implementation:

In the development and testing phase, software engineers and testers will take the requirements and specifications described in the previous steps and implement them in the working code. For this purpose, we will use App Inventor as our main program for the development of our application. The app will then need to be thoroughly tested before being published on an app store such as Google Play. It may be preferable to begin by releasing a beta version, available to a limited number of users, in order to further test the app before its full release.

Maintenance:

The maintenance phase will include the procedure of getting feedback from users, in order to use it to improve our application. Through user feedback our application will be continuously assessed to ensure that it does not become obsolete. Small fixes or adjustments will be required.

Documentation and Evaluation:

Documentation for a mobile application is a set of written materials that provide information about the application, including its design, development, and maintenance. It helps developers, stakeholders, and users understand the application's purpose, functionality, and behavior. Regarding the evaluation, we can gather information from Google Play's ratings and reviews of our application. To document the app, you will need to perform the following steps:

- Write a clear document explaining the design of the system.
- Add annotations within code sections during the development process.
- Document system testing processes.
- Prepare a user guide.



Job Opportunities in Software Engineering

Software engineering offers various job opportunities in various domains and industries. The following are some of the most common job opportunities in software engineering:

- **Web Developer:** This role is responsible for developing websites and web applications.
- **Mobile Developer:** This role is responsible for developing mobile applications for iOS or Android platforms.
- **DevOps Engineer:** This role is responsible for automating the deployment and operation of software systems.
- **Cloud Engineer:** This role is responsible for building and maintaining cloud-based software systems.
- **Database Administrator:** This role is responsible for managing and maintaining databases.
- **Quality Assurance Engineer:** This role is responsible for testing software systems to ensure they meet quality standards.
- **System Administrator:** This role is responsible for maintaining and managing computer systems and networks.

Software Development Methodologies

The process of developing information systems differs from that of writing small programs. Developing large programs such as the systems of government institutions or commercial companies requires great effort and may take months or even years. Understanding customer requirements and the nature of the required system or software functionality is also a challenge for the development team. Software engineering aims to develop workflows, methods and protocols to overcome these difficulties.

There are many software development methodologies and each one is used for different purposes. The most common ones are the following.

Waterfall Methodology

This method is considered one of the oldest methods in software development. It gets its name because its development stages are sequential from one stage to another in a unidirectional manner. When a certain stage in development is completed, the transition to the next stage is carried out without returning to the previous stages. The outputs of each stage represent the inputs for the next stage.

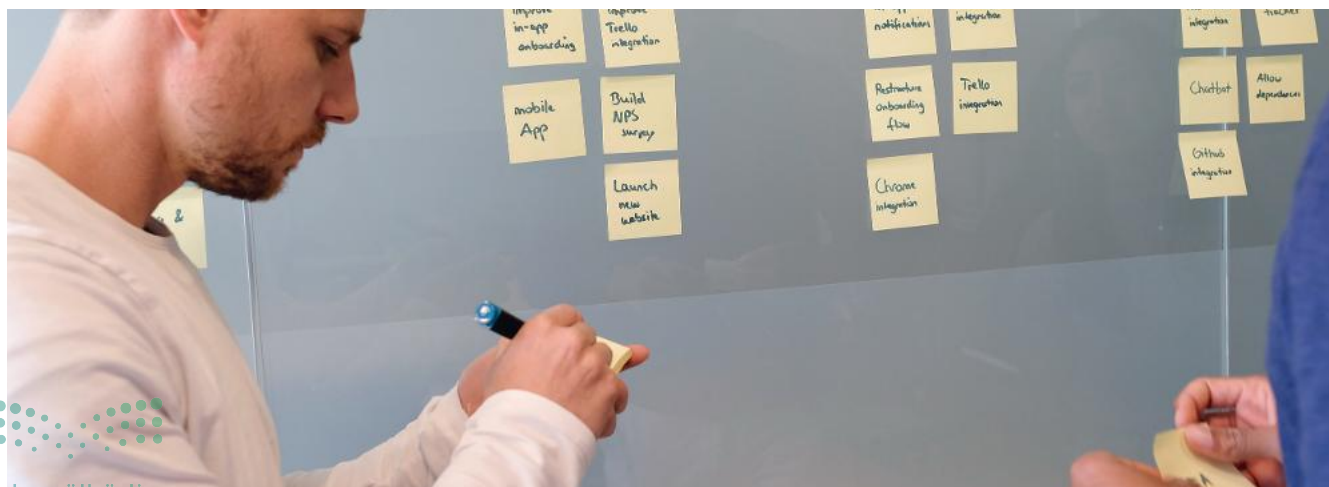


Figure 1.4: Planning the development of a software product

The waterfall methodology goes through all stages of developing the system sequentially, and each stage depends on the outputs of the stages that precede it, as follows:

1. Analysis stage

At this stage, requirements are collected in the ways that were explained earlier, and after completion, they are documented accurately and in detail. They are verified and approved by the customer before the design stage is begun. Changes to the requirements later in the process can cause serious problems in the system.

2. Design stage

The requirements documented from the previous stage are translated into a design that clarifies the structure of the system and identifies its resource needs. The system design reflects how the requirements are implemented from a technical point of view and the logical sequence of the operations that take place in it.

3. Development stage

At this stage, the system is built and programs are written based on the outputs of the design.

4. Testing stage

At this stage, testers verify that the system has met the requirements documented in the previous stages, and investigate whether there are any errors to be fixed. The customer sees the real product for the first time at this stage.

5. Implementation stage

This is the stage in which the system is implemented and delivered to the client. Users are trained or deployed, and the system's performance is monitored to ensure that there are no errors during the implementation.

6. Maintenance stage

This stage includes fixing errors that appear during everyday use of the system as well as making some developments and improvements to the system.

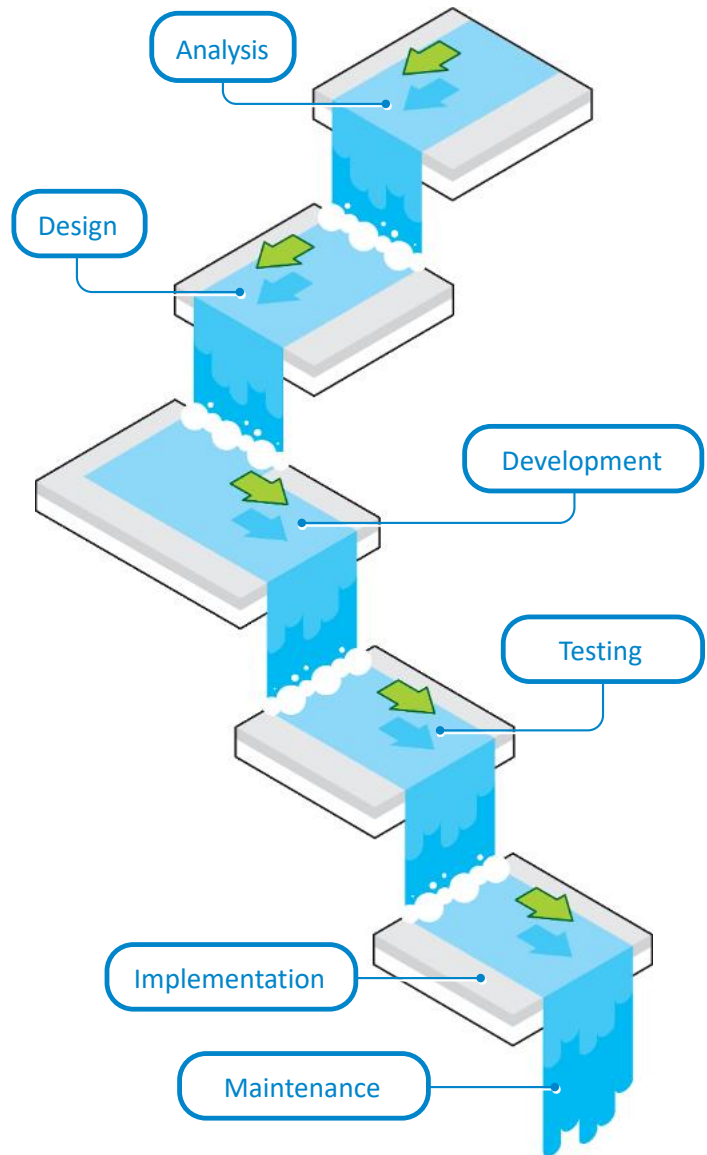


Figure 1.5: The stages of waterfall methodology

Advantages of the Waterfall Methodology in Development:

- The stages are clear and specific and do not overlap with each other.
- Project planning, management and follow-up is easy due to the clarity of the stages.
- It is suitable for small projects whose requirements are clear and stable.

Challenges of the Waterfall Methodology:

- It is difficult to go back from one stage to the previous stages for modification because each stage depends on the previous one. Going back and making changes to earlier stages greatly affects the next stages and increases the cost of development.
- It is not suitable for large and complex systems and software.
- It is not suitable for software and systems whose requirements are subject to change.
- It is not possible to begin a new stage before the completion of the previous stage. This delays the discovery of any misunderstanding of the client's requirements that may appear in late stages, and makes late-stage modification after that a difficult and costly process and even increases the risk of project failure.

Rapid Application Development Methodology (RAD)

In contrast to the waterfall methodology, in which the development process takes place through separate phases, the rapid application development methodology relies on development through iterative cycles. The key characteristic of this methodology is the development of prototypes of the system in order to obtain feedback and suggestions from the client in the early stages of development. This helps avoid misunderstanding of requirements and thus avoids the significant cost of returning to modify the system after development has been completed. It is worth noting that these prototypes are modified to become part of the final product.

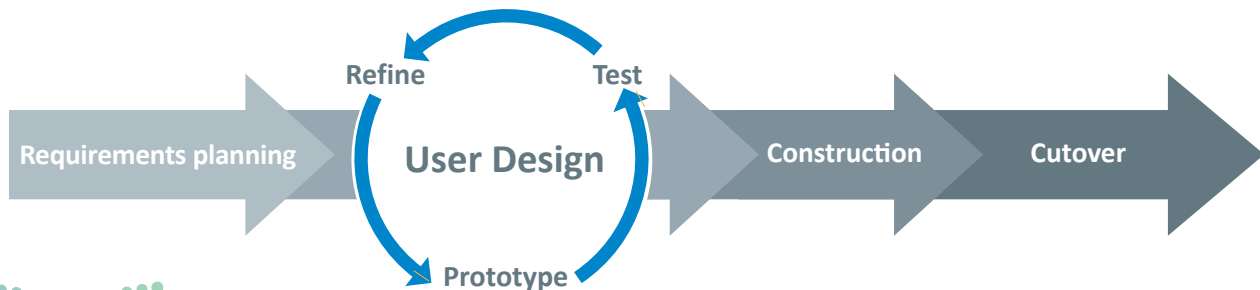


Figure 1.6: The stages of RAD methodology



This methodology includes the following stages:

1. Requirements Planning

The requirements planning phase in Rapid Application Development (RAD) is the stage in the software development process where the scope and objectives of the software project are defined. The requirements planning phase focuses on ensuring that the software project is well defined and understood and that the resources and schedule needed to complete the project are identified.

2. User Design

The user design phase in Rapid Application Development (RAD) is the stage in the software development process where the software requirements and design are created in close collaboration with end users. The user design phase focuses on ensuring that the software application meets the needs and expectations of its intended users.

3. Construction

The construction phase in RAD is the stage in the software development process where the software application is developed and built. This phase involves writing code, integrating the various components of the software application, and testing it to ensure it meets the requirements and quality standards. This phase also involves fixing any bugs or issues discovered during testing. The construction phase focuses on efficiently delivering a working software application that meets the requirements defined in earlier stages of the RAD process.

4. Cutover

The cutover phase in RAD is the final phase of the software development process, in which the new software application is transitioned into the live production environment. The cutover phase involves a series of activities that must be completed in order to successfully transition the new software application into the production phase and training the users on how to use the application.

Advantages of the Rapid Application Development Methodology

- Developers receive end-user feedback continuously from the beginning of the project, which reduces the risk of failure and the cost of modification.
- User participation, through their feedback, leads to more efficient software production and reduces the error rate.
- The use of iterative cycles and prototyping contributes to reducing the time required for system development.

Challenges of the Rapid Application Development Methodology

- The development cycle is more complex and must be carefully managed.
- Stakeholder interaction in providing insufficient feedback may result in a product that does not meet the desired requirements.
- It requires highly qualified programmers and designers who are able to get things done quickly.



Agile Methodology

The agile development methodology uses a project delivery method with successive releases called sprints. Each release adds new features to the previous one, and each release goes through all stages of system development from planning to testing and approval by the user.

The agile development methodology can be distinguished from the rapid application development methodology because in the agile development methodology a functioning product is presented to the user at each stage. When needed, the product is worked on and modified or new functions are added. This is in contrast to the rapid application development methodology, in which the user is presented with an incomplete prototype in order to give feedback and only then, when all the requirements are fully established is the final product developed.

Sprint

A sprint is a time-boxed iteration of work in Agile software development that typically lasts one to four weeks. The purpose of a sprint is to deliver a potentially shippable increment of the software product, focusing on meeting the goals and objectives set at the beginning of the sprint.

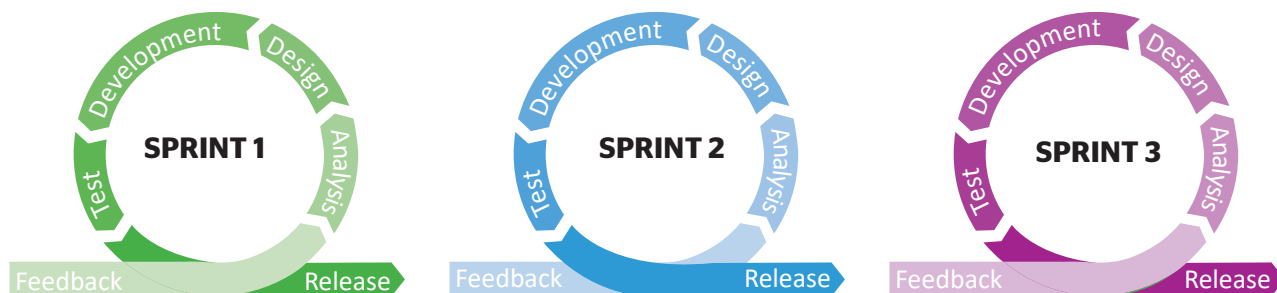


Figure 1.7: The stages of Agile methodology

Advantages of the Agile Development Methodology:

- Less time needed to achieve the first product release.
- Project risks are identified easily through user feedback.
- Stakeholder participation in the development of the system gives them more confidence in the developed software or system.

Challenges of the Agile Development Methodology:

- This methodology focuses more on development and less on documentation. It is therefore difficult to integrate new team members into a project after it has begun.
- The response and performance of users affects the speed of production and quality of the product.
- A change in project requirements can disrupt the entire project, especially if the user is frequently changing their mind.

INFORMATION

The agile methodology requires effective communication and continuous collaboration between all teams involved in planning, design, development, and user testing.

Exercises

1 Choose the appropriate method that corresponds to each of the following statements:

Waterfall
methodology

1

Rapid application
development

2

Agile methodology

3

☐

Product evolution in the form
of successive versions.

☐

You cannot move to the next
stage without completing the
previous one.

☐

It is based on the method of
designing and improving
prototypes.

☐

Not suitable for large
and complex systems.

☐

Repeated cycles are used
to reduce production time.

☐

The fastest way to get a preview
of the real product.



2 Compare the three methodologies described in the lesson by their design and implementation phases.

	Waterfall	RAD	Agile
Designing Phase			
Implementation Phase			

3 Clarify the role of the user in the three methodologies with reference to the stage in which this role appears clearly.

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and extend across the width of the page. There are no margins, text, or other markings on the paper.

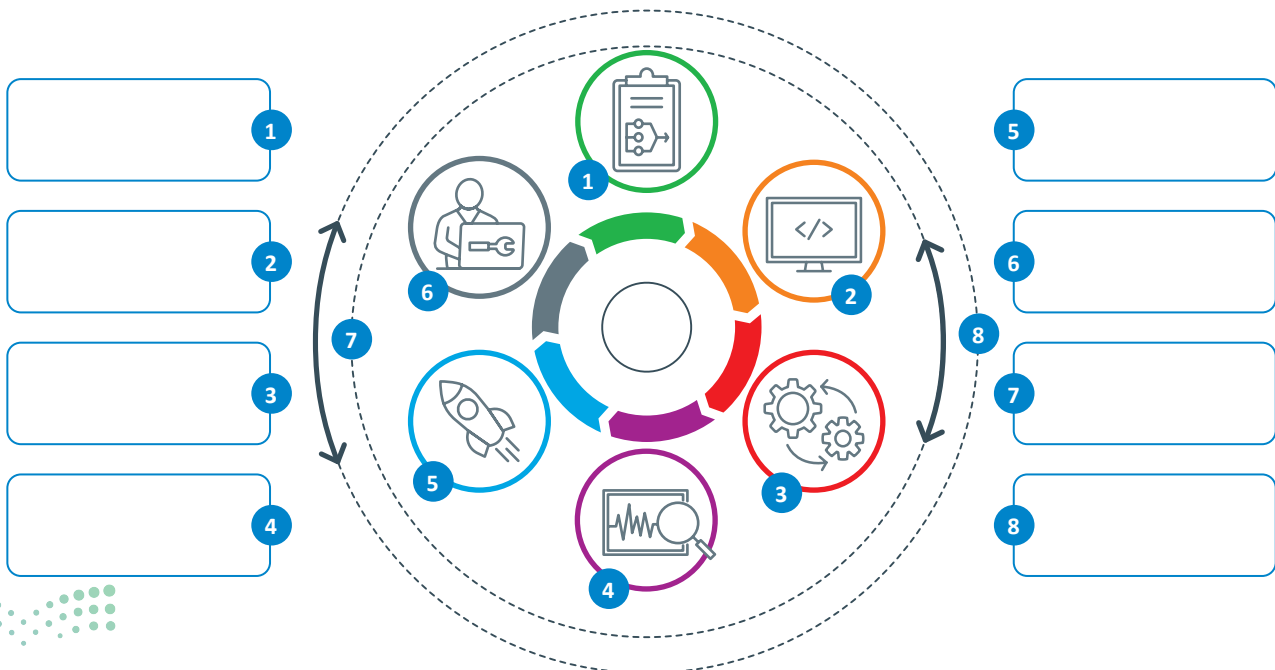
4

Read the sentences and tick ✓ True or False.

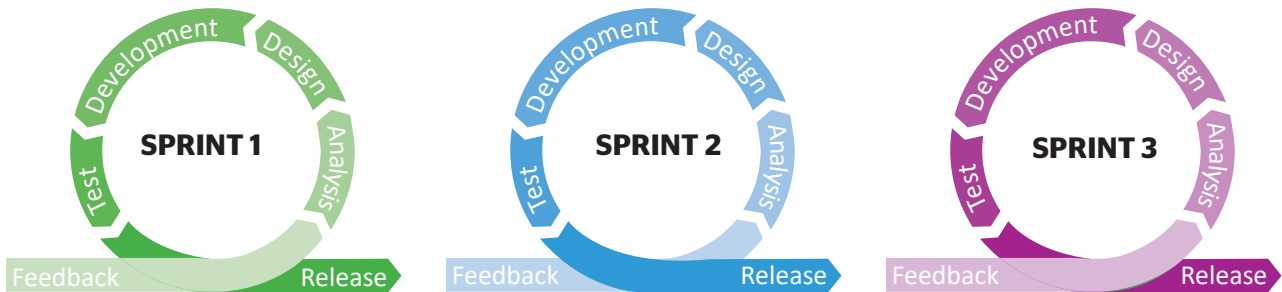
Read the sentences and tick ✓ True or False.	True	False
1. Software development refers to the process of planning.	☐	☐
2. The methodology of software development is a framework that is used for many procedures.	☐	☐
3. Software development involves the creation, testing and development of an information system.	☐	☐
4. The procedure of the development of an information system is controlled by the software development methodology that is used.	☐	☐
5. Software development is the process that divides the whole procedure into distinct phases and is also called the software development life cycle (SDLC).	☐	☐

5

Fill in the blanks for the phases of the SDLC in the following diagram.



6 Study the following figure and then answer the questions that follow.



1. Which software development methodology is represented by the figure?

2. What is meant by the term sprint?

3. What are two advantages of this methodology?

4. What are two challenges of this methodology?



7 Match each of the following stages of system development with the appropriate processes in each of the following sentences:

Analysis	1	☐	The theory is turned into practice.
Design	2	☐	This eliminates errors in the system during the working life.
Development and testing	3	☐	The requirements and the specifications are converted into effective code.
Implementation	4	☐	All the details of the new system are defined here.
Maintenance	5	☐	This has to do with the knowledge base which is required for anyone involved to understand how the system works.
Documentation	6	☐	The problem that needs to be solved is identified.
Evaluation	7	☐	This can be performed not only by the IT team but also by the users and management.



8 Graphically illustrate how each of the three software development methodologies works.

Waterfall methodology

Rapid Application Development methodology

Agile development methodology



Choose the correct answer:

1. The stage in which data collection tools are used is:	Analysis	☐
	Design	☐
	Implementation	☐
	Maintenance	☐
2. The stage in which a programming language or computer program is used to prepare the system is:	Design	☐
	Evaluation	☐
	Development	☐
	Documentation	☐
3. The stage in which the user guide for the system is prepared is:	Analysis	☐
	Documentation	☐
	Evaluation	☐
	Testing	☐
4. In the evaluation phase of the smartphone application:	The application is created using the App Inventor program.	☐
	The user needs are determined.	☐
	Feedback is received from users.	☐
	The application is designed to work on the Android platform.	☐



Lesson 2

Programming Languages and Languages Processors

Link to digital lesson



www.ien.edu.sa

A Brief History of the Development Programming Languages

Many things have changed since the creation of the first computer to the present day. Computer components and technologies have evolved greatly, as have advanced processing capabilities. Despite this, the concepts of computer operation formulated by von Neumann in 1945 still apply.

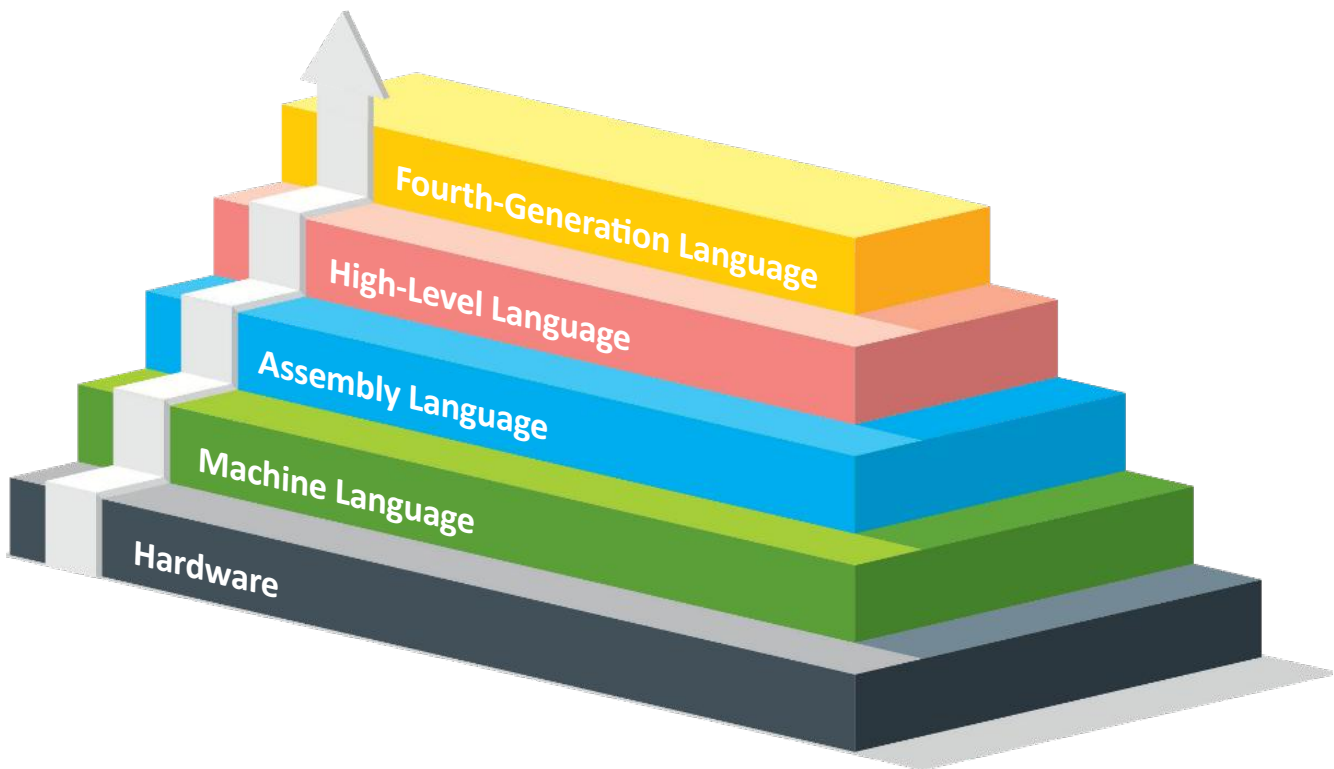


Figure 1.8: Evolution of programming

Programming languages were invented for the purpose of human-machine communication.



Machine Language

In order for a computer to perform any function required from it, it must be given the appropriate sequences in the form of binary numbers, consisting of 0s and 1s. This "machine language" will not be understood by humans. Machine languages are very difficult for a programmer to apply and implement due to the need for a deep knowledge of computer components and their structure, and especially since the machine language of each Central Processing Unit (CPU) is different.

A machine language program is a sequence of binary bits, which are the instructions issued to the processor to carry out basic operations.

Assembly Language

Between machine language and high-level programming languages there is an intermediate language called assembly language, also called symbolic programming language.

Assembly language is similar to machine language but is somewhat easier to program as it allows the programmer to replace numbers (0, 1) with symbols.

Human comprehensible assembly language commands are converted into corresponding sequences of 0's and 1's for the computer to understand and execute.

For example, in assembly language, the word ADD, followed by two numbers, is used for addition. This is easy to understand and memorize for humans, but must be translated into a series of bits in the computer to carry out the required operation. This translation process is carried out by a special program called the assembler.

Assembly language commands are made up of symbolic segments that correspond to machine language commands.

Challenges of Assembly Language

- The use of assembly language makes it easier to program simple operations of unintelligible binary sequences, but it is nevertheless considered a low-level language.
- The assembly language used varies depending on the architecture of each computer.
- Assembly language does not provide commands to perform more complex functions than simple additions, multiplications and comparisons, forcing the programmer to write long and complex programs that are difficult to understand and debug.
- A program cannot be transferred from one computer to another of different architecture.



The following table shows a program with an addition written in a high-level programming language and its equivalent in assembly and machine language for a computer with a 6502 8-bit CPU. The high-level language program can be used on most computers, while the assembly and machine translation will work only on a computer with the same CPU architecture.

Table 1.1: Calculating an addition		
High-level language	Assembly language	Machine language
sum = 0	LDA #0	10101001 00000000
	STA sum	10000101 00000000
	LDA sum	10100101 00000000
	CLC	00011000 00000000
sum = sum + 5	ADC #5	01101001 00000101
	STA sum	10000101 00000000
	LDA sum	10100101 00000000
print (sum)	JSR print	00100000 11100001

High-Level Programming Languages

The shortcomings of machine language and assembly language led to a concerted effort to achieve better human-machine communication, which resulted in the emergence of the first high-level programming language in the 1950s.

High-level programming languages use programming commands which resemble human language. The resulting programs must be translated into machine language by the computer itself, using a special program called the translator. Compilers and interpreters are types of translators used for different types of programming languages.

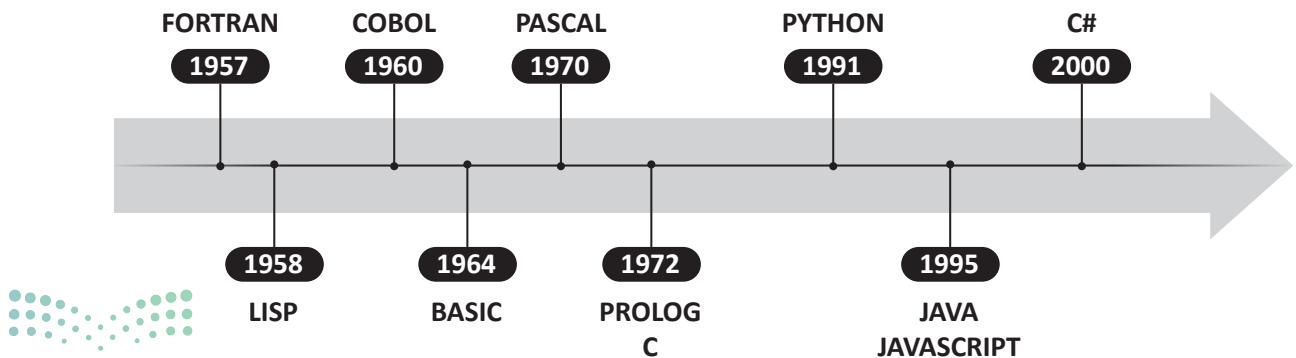


Figure 1.9: The evolution of programming languages

The Evolution of High-Level Programming Languages

The developer chooses the programming language that allows the application to be easily developed in a given environment to implement a software solution, but at the same time the developer also chooses the language based on their personal knowledge, skills and preferences.

Each programming language has a unique set of reserved words (words that the language understands) and a syntax that the programmer uses to write instructions.

Table 1.2: Basic information of programming languages

Programming language	Developer	Etymology	Properties
FORTRAN	IBM	FORMula TRANslation	Suitable for solving mathematical and scientific problems, but not suitable for managing data files, for example.
LISP	MIT	LISt Processor	A language for artificial intelligence.
COBOL	CODASYL	Common Business Oriented Language	Suitable for developing commercial applications and general management applications.
BASIC	Dartmouth College	Beginner's All Purpose Symbolic Instruction Code	A multi-domain programming language.
PASCAL	Professor Nicholas Wirth	Named after the mathematician Blaise Pascal	It is famous for introducing structured programming techniques. It adopts program design in a systematic and accurate manner.
C	Dennis Ritchie and Bell Labs	The C language is named after a prior language named B	It is used for UNIX operating system development, and is suitable for different operating systems.
JAVA	Sun Microsystems	Named after a type of coffee (Java)	It is an Object Oriented programming language used to develop applications that can run on a wide range of computers or different operating systems.



Features of High-Level Programming Languages:

High-level programming languages have several advantages over assembly language:

- They use logic and programming formulas that are understandable and close to human language.
- They are independent of the type of computer, and can be used on any device with or without minor modifications.
- Developers can learn high-level programming languages more easily and quickly.
- Software debugging and maintenance is much easier.

In general, high-level programming languages reduce the time and cost of software development significantly compared to low-level programming languages.

Fourth-Generation Programming Languages

Among high-level programming languages, we note that there are so-called fourth-generation programming languages, which are usually abbreviated as 4GL. Fourth-generation programming languages are closer to human language than other high-level languages and are accessible to people without formal training as programmers because they require less coding.

Fourth-generation languages are more programmer-friendly and enhance programming efficiency by using English-like words and phrases, as well as icons, symbolic representations, and graphical interfaces when needed. The key to achieving efficiency with 4GL is compatibility between the tool and the field of application.

Computer users in fourth-generation languages can make changes to the program in order to meet a new need and have the ability to solve small problems by themselves. Multiple joint operations can be performed using a single command entered by the programmer.

Scripting languages are a type of programming language typically interpreted rather than compiled. They are used to automate repetitive tasks, simplify complex operations, and enable rapid prototyping of software systems. Some common examples of scripting languages include JavaScript, Ruby, PHP and Perl. They often feature rich libraries, and a focus on productivity, making them well-suited for tasks that require rapid prototyping and iteration. However, they may not be as efficient or scalable as compiled languages and may not be suitable for performance-critical or resource-intensive applications.

For data operations, a user can create queries and reports using SQL for statistics and scientific projects, and a mathematician or researcher can use software such as SPSS, MATLAB, and LabVIEW to analyze this data.



Classifications of Programming Languages

There are many classifications of programming languages. Languages can be classified in terms of the type of commands used, such as procedural programming languages and object-oriented programming languages.

Procedural programming uses a set of instructions to tell the computer what to do step by step. Examples of procedural programming languages are COBOL, Fortran, and C.

In object-oriented programming, the program is divided into units called objects. Examples of object-oriented programming languages are C#, C++, Java and Python.

Programming languages can also be classified according to what they are used for:

1. General programming languages: In theory, any general programming language can be used to solve any problem, but in practice, each language is designed to solve a specific type of problem. These languages are divided as follows:

- Science-oriented languages such as Fortran.
- Business-oriented languages such as COBOL.
- Multi-domain languages such as BASIC and Pascal.
- Operating system programming languages such as C.
- Artificial intelligence languages such as PROLOG.
- Specialized database management languages such as SQL.

2. Specialized languages such as LISP are used for a specific type of application such as robotics or integrated circuits.

How Computers Understand Programming Languages

Any program written in any programming language is converted into machine language that can be understood and executed by a computer, through the use of special translation programs.

There are two ways to run programs written in a high-level language. The most common is to compile the program with a compiler but in some programming languages an interpreter is used instead.

Let's go over how to implement these two different methods.

Compiler

A compiler is a computer program that converts an entire block of code written in a high-level programming language into machine language, which is understood by the computer's processor.

Interpreter

An interpreter is a computer program that converts each line of code from a block of code written in a high-level language into machine language and sends it for execution directly, before moving on to the next line of code.



Program translation and linking process:

- The compiler accepts a program written in a high-level language as the input file (the source code), and produces an equivalent machine language program called object code.
- The compiler cannot compile statements that refer to standard libraries or resources outside of the source code, so the process will require an additional step of linking and converting those statements.
- Another program called the Linker or Loader handles the linking process and links the object code file with the standard library files, and produces the executable code, which is the final program that the computer executes.

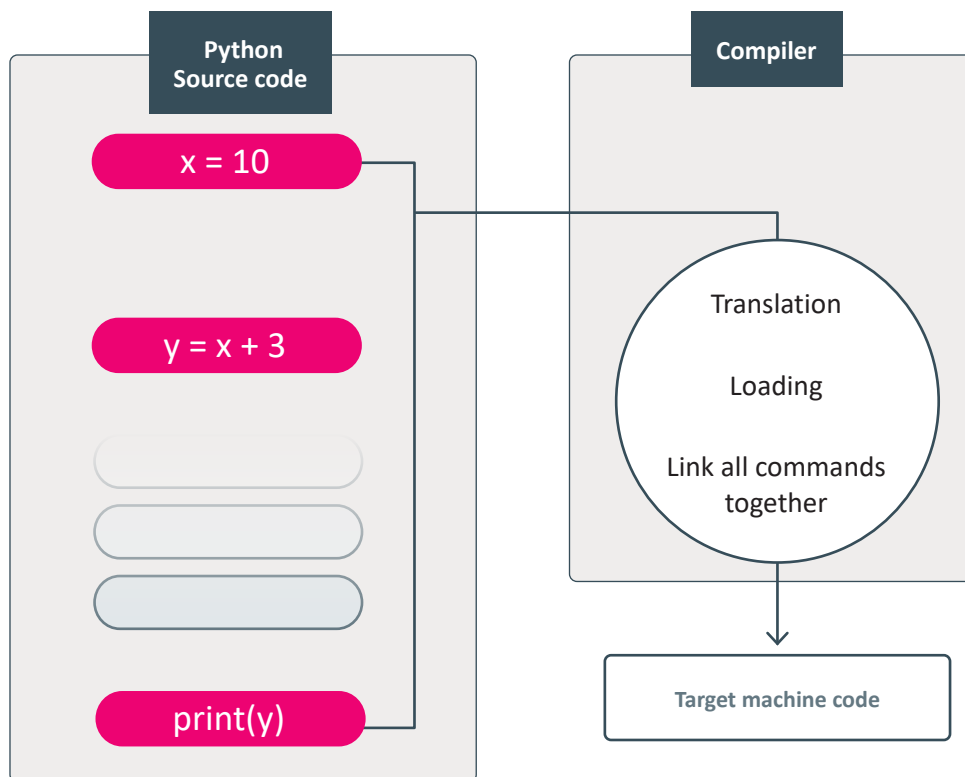


Figure 1.10: The process of compiling and executing a program using a compiler

The source code is the program written in a high level programming language.



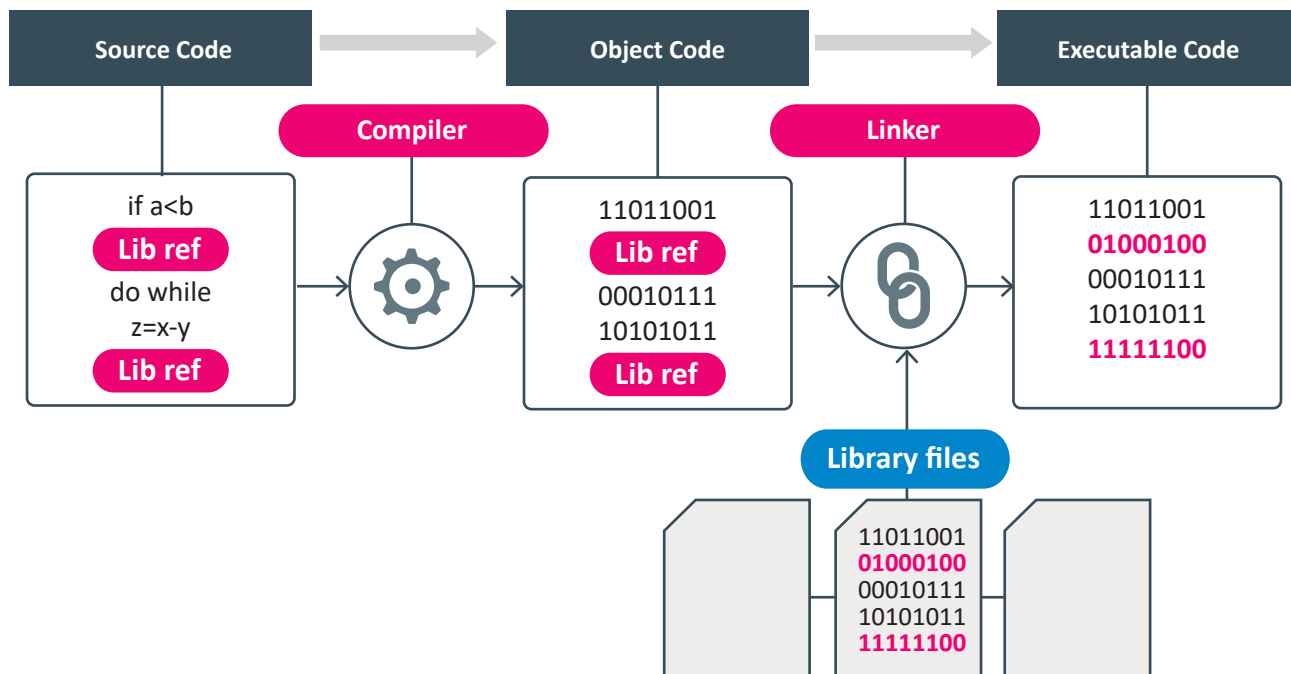


Figure 1.11: Stages of translating and linking the program

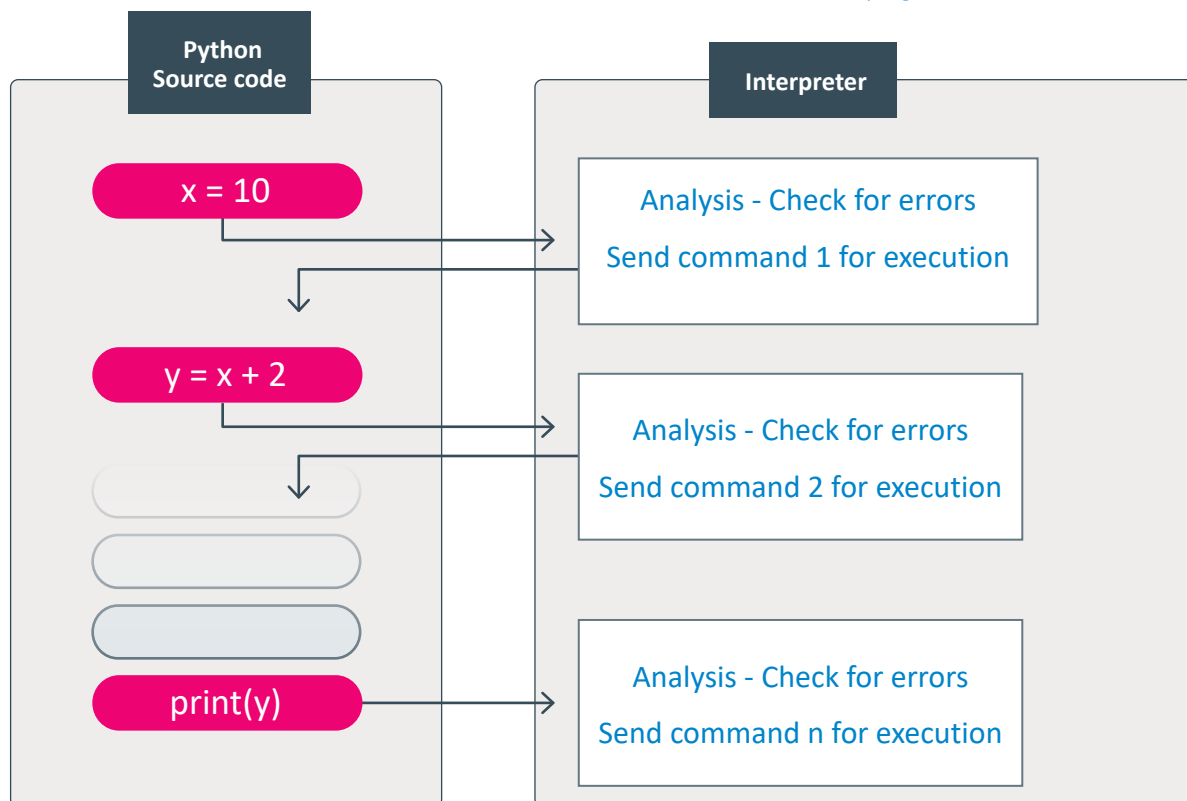


Figure 1.12: The process of compiling and executing the program using an interpreter

Compilers and interpreters perform the same task, which is to convert the program written in the high-level programming language into machine language, but in two different ways.

Interpreted and Compiled Programming Languages

Most modern programming languages use compilers to produce optimized code quickly, but there are some languages that still use interpreters when there is a need to create a simple program for which speed is not the primary concern.

Compiled Languages

The C, C++, C#, and Java programming languages use a compiler to build fast, reliable programs. The executable code is created for each type of computer hardware, which means that developers have to know what the end user's computer hardware is.

Interpreted Languages

Legacy JavaScript, LISP, and BASIC use interpreters, which means that programs run slowly but their source code can run on any computer that has a particular programming language's interpreter. For example, a web application written in JavaScript can run on a Windows computer or an Android tablet with a web browser that integrates the JavaScript interpreter.

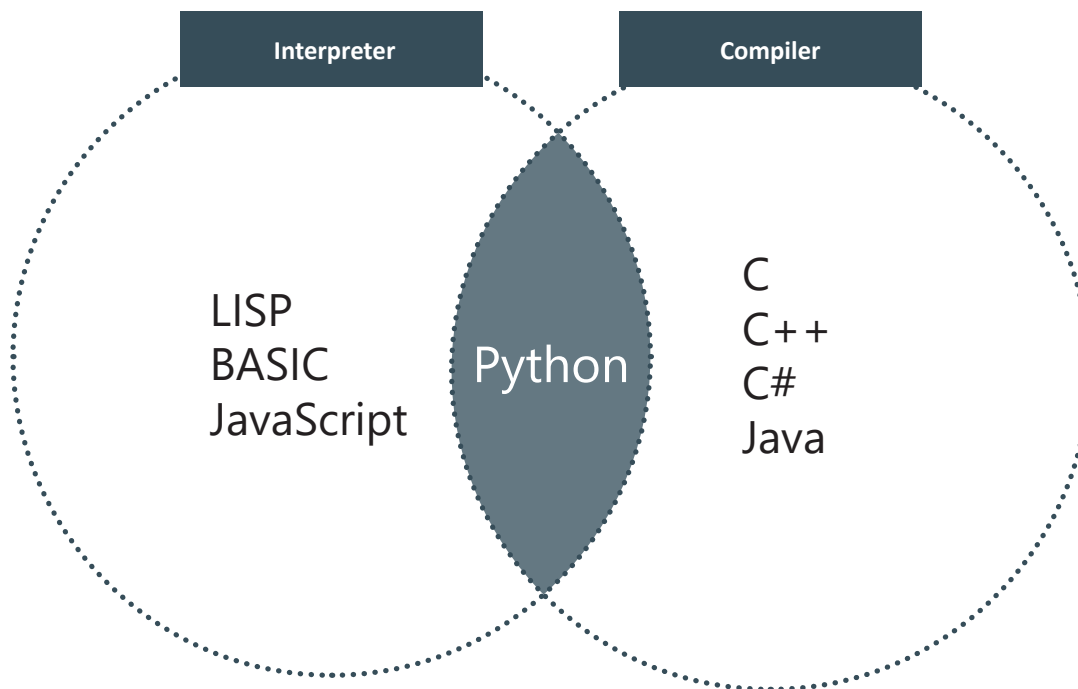


Figure 1.13: Python is an interpreted and a compiled language

Python is both an interpreted and a compiled language. The Python application compiles each line of code so that it can be read by the interpreter on the hardware in use. The syntax used by the programmer doesn't change because the Python application converts it into the correct form for the interpreter used on that hardware.



Table 1.3: Compiler vs Interpreter comparison

	Compiler	Interpreter
Main Function	Converts the entire source code written in a high-level programming language into machine language, and produces an executable program.	Converts the block of code into machine language so that it translates and then executes the block of code, moving to the next block while the program is running.
Input	The compiler takes the entire source code as input.	The interpreter takes one of the source code instructions as input each time.
Output	The compiler generates and stores an object code file as output.	The interpreter does not generate an object code file.
Memory	Requires more memory due to object code generation.	Less memory is required.
Implementation process	The compilation process takes place for the entire source code before execution begins.	The interpretation process for each code statement takes place in parallel with the execution process.
Error checking	The compiler displays all language errors and warnings when compiling the program. You cannot run the program until all errors are corrected.	The interpreter reads one line of code and displays any errors in it. This error must be corrected before moving on to the next line.
Link files	Needs a program to associate the object file with standard library files to create the executable.	Does not need the link process, and does not create an executable file.
Speed	Availability of .exe file makes execution faster.	The execution process is slower because the executable file is not available. The program is interpreted again on each execution.
Dependence on hardware and operating systems	The executable file generated by the compiler depends on the target hardware. It cannot run on different CPU architectures or different operating systems.	The interpreter varies depending on the operating system, but the same source code can run on multiple systems as long as a suitable interpreter is available for each system.



Dealing with Software Errors

Compilers and interpreters operate differently when they face errors and bugs in the source code.

Compiler:

1. Program creation.
2. The compiler will analyze and process all lines of code and make sure that they are correct.
3. If there is an error, an error message will appear.
4. If there is no error, the compiler will convert the source code into machine language. Multiple code files will be associated with a single executable program (known as an EXE file).

Interpreter:

1. Program creation.
2. The interpreter reads one line of code and displays any syntax error. This error must be corrected before moving on to the next line.
3. All the lines of the source code are executed line by line during program execution by the interpreter.

Debugging Process

The source code in its first version may often contain many errors, which are divided into three types:

- **Logical errors:** Errors in the logic of the program.
- **Runtime errors:** Errors that occur during the execution of the program.
- **Syntax errors:** Errors in the syntax of the code.

Logical errors and runtime errors only occur when the program is executed, while syntax errors occur during compilation. The program is executed only if the source program contains no syntax errors.

Debugging Syntax Errors:

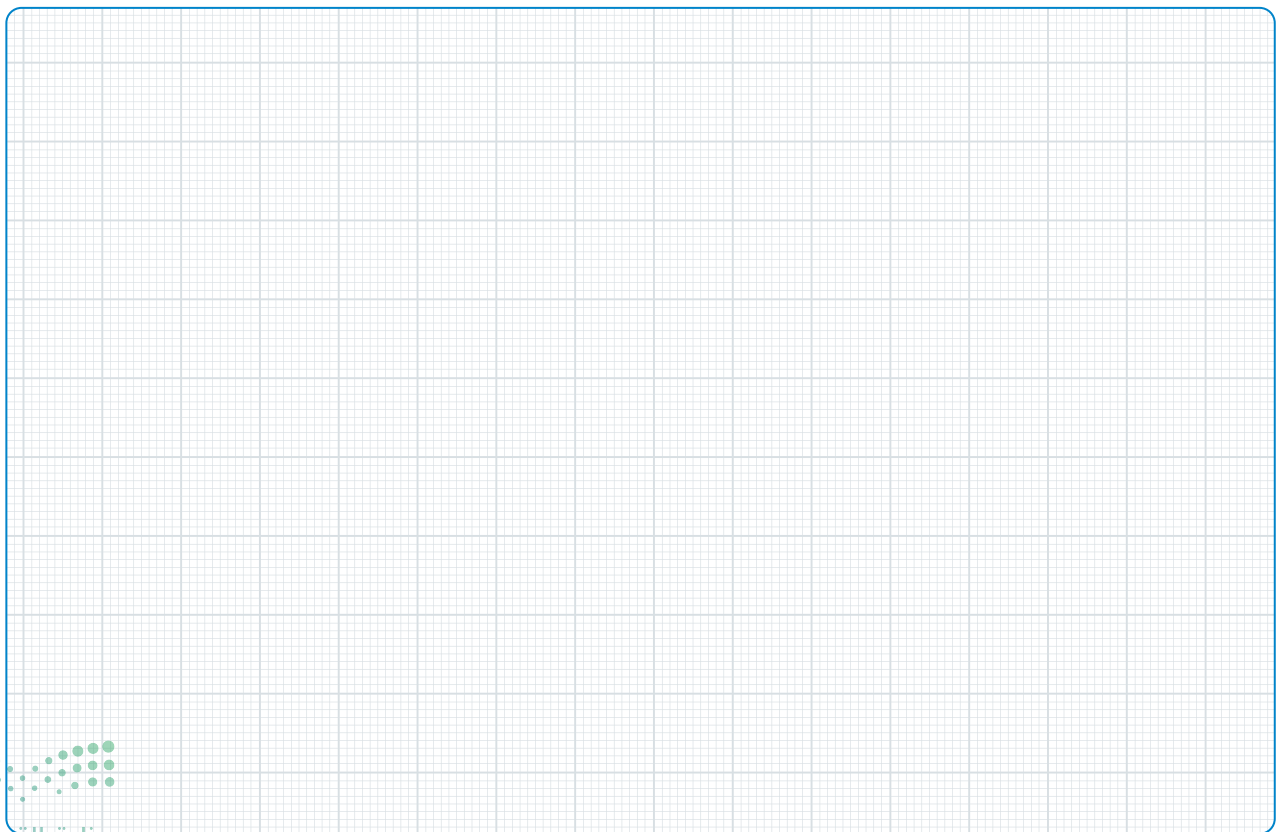
- In the first step, the compiler or interpreter detects syntax errors and presents messages indicating the error and its location. Some of them can specify the cause of the error.
- The next step is to correct errors in the program.
- Finally, the corrected program compiles correctly, without any error messages.



Exercises

1 What are the shortcomings of assembly language?

2 Draw a diagram to show the difference between the process of translating and implementing code in a compiler and in an interpreter.



3 Write three advantages of high-level programming languages.

4 Relate each programming language to the classification it belongs to.

Python

1

☐

Object-oriented languages

BASIC

2

☐

Artificial intelligence languages

PROLOG

3

☐

Multi-domain languages

C

4

☐

Systems programming languages



- 5 Choose the appropriate word or phrase to complete the sentences (Not all options apply to a blank line):

link

memory

syntax errors

interpreted
languages

standard libraries

object code

runtime errors

the compiler

the source code

1. _____ accepts the source program as input, and produces an equivalent machine language program called _____ .
2. The _____ used by the interpreter is less than that used by the compiler.
3. Using _____ is an advantage in terms of real-time debugging, but program execution is slower.
4. The compiler cannot convert statements that refer to _____ , so it needs to concatenate and convert those statements.
5. The executable can be created if there are no _____ in the source program.
6. Errors that occur during program execution are called _____ .





Software Development Tools and Programs

Developers use a wide range of tools to develop software applications, each of which has its advantages and disadvantages. The programming process requires developers to be flexible and creative to take full advantage of the capabilities of different software development tools to deliver high-quality work for their clients.

Software development tools and programs are used to assist the software development team in various tasks, including creating, modifying and maintaining programs, as well as debugging and implementing software tasks and development processes. There are also many specialized programs that provide or support specific tasks in the stages of the software development cycle.

Table 1.4: Classification of software development tools

Software development tools	Description
Code Editors	Used to write and make changes to code.
Compilers and Linkers	Translate programs into executable machine language.
Debuggers	Help us correct errors in the software.
Project Builders	Make sure that all the necessary files will be compiled and linked to one final program.
Code Management Tools	Ensure that program files are not accidentally replaced when multiple programmers are working on the same program concurrently.
Integrated Development Environment (IDE)	Provides programmers with an integrated software environment that includes a text editor, compiler, linker, and debugger.
Profilers	Usually give us a good idea of the program's needs and handling of processor time and memory resources while running.
Network Analyzers	Necessary when writing software for networking applications in particular.
Database Explorer and Analyzer	Allows dealing with databases and analyzing the performance of specific database queries.

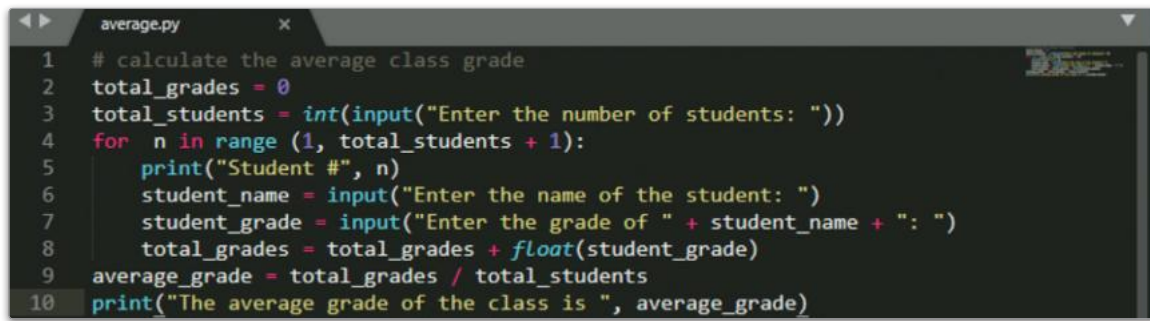


Code Editors

A code editor allows us to create and edit several connected programming language files, and usually it can handle many different languages like HTML, CSS, JavaScript, PHP, Ruby, Python, C, etc. Code editors use indents and different colors to format the code into code sections. This makes them much more suitable for writing code than ordinary word processors and text editors like Microsoft Word or Notepad.

Features of Code Editors

- Error-checking
- Auto-completing and code suggestions
- Code snippets
- Syntax highlighting
- Facilitate navigation of code files and resources
- Adding more functionality via extensions

A screenshot of a code editor window titled 'average.py'. The code is written in Python and calculates the average grade of a class. It uses syntax highlighting: comments are green, keywords are blue, strings are red, and variables/numbers are white. The code is as follows:

```
1 # calculate the average class grade
2 total_grades = 0
3 total_students = int(input("Enter the number of students: "))
4 for n in range(1, total_students + 1):
5     print("Student #", n)
6     student_name = input("Enter the name of the student: ")
7     student_grade = input("Enter the grade of " + student_name + ": ")
8     total_grades = total_grades + float(student_grade)
9 average_grade = total_grades / total_students
10 print("The average grade of the class is ", average_grade)
```

Figure 1.14: Python script in a code editor

There are many code editors that can be chosen by the programmer according to their preferences. The only criterion for choosing an editor is the efficiency of that editor for the required task. Some examples of code editors are:

- | | | |
|----------------------|---------------|-------------|
| • Sublime Text | • Espresso | • Notepad++ |
| • Zed | • Python IDLE | • Vim |
| • Visual Studio Code | • Nova | • BBedit |
| | | • Ultraedit |

Advantages and Challenges of Using Code Editors

Advantages

- They can rival the Integrated Development Environment (IDE) Editor for standard programming tasks when appropriate extensions to support different programming languages are used.
- They are smaller and faster to load than IDEs.
- Their streamlined interfaces make it easy to focus on our code.

Challenges

- They lack a lot of editing features which only IDEs provide, such as smart editing.
- Users may need to configure the code editor with the appropriate extensions before use.

Integrated Development Environments (IDEs)

Integrated development environments are usually presented with their own built-in applications, which include a number of software development tools such as an interpreter for use during the program creation phase, and a compiler for finalizing and publishing the program.

Modern integrated development environments are not limited to providing a compiler for the programming language only, but rather contain all the necessary programs and tools to help write and implement code, and most importantly, to diagnose and correct programs. Among the most important tools included in integrated programming environments are:

- File Explorer
- Code Editor
- Interpreter
- Compiler
- Linker
- Debugger
- Output Viewer

IDEs must include an editor dedicated to facilitating the creation of graphical objects such as forms, menus, and dialog boxes, in order to provide the developer with the appropriate tools to create the code blocks related to these objects.

Features of IDEs

- Smart completion of code in the code editor.
- Integration with code management tools for version control.
- Advanced testing tools
- Automatic linking of source code libraries.
- Tools for automating code creation and deployment.

All these services can be accessed through a unified user interface.

Examples of IDEs

In the past, most IDEs supported only one programming language and were usually created by the software companies or organizations that created that specific language.

Nowadays, most software development projects integrate different technologies and programming languages, which requires IDE development environments that can support a wide range of languages.

For example, Microsoft Visual Studio supports C, C++, C#, VB.Net, Python, Ruby, Node.js, JavaScript, HTML / CSS, etc.

Other popular IDE tools include NetBeans, Eclipse, Atom-IDE, Xcode, Android Studio, IntelliJ IDEA and PyCharm.

Xcode is used to develop mobile application software for iOS devices. For Android devices, Android Studio is used.

Advantages and Challenges of Using integrated Development Environments (IDE)

Advantages:

- Provide intelligent code completion and analysis tools for faster programming with fewer errors.
- They provide powerful code browsing and discovery tools, and make it easy to access any part of the program, no matter how large the project.
- They offer multiple ways to debug and test code without leaving the editor.
- They support many programming languages natively and provide many code navigation and code analysis tools to facilitate work and productivity on large projects.

Challenges:

- The interfaces are packed with a lot of features which can make them complicated and difficult to use.
- They require a certain amount of training to use them correctly.
- Excessive functions often lead to slow performance.

Cloud Software Environments

Besides traditional software development environments, there are web-based cloud development environments, such as GitHub Codespaces.

Cloud software environments provide the ability to work on our project from any computer, anywhere in the world, as our software development project data resides in the cloud.

One of the main drawbacks of these environments is the need to connect to the Internet to access data and do work.

Advantages of Using Cloud Software Development Environments

- Access to software development tools from anywhere in the world.
- Possibility of using any device with a web browser.
- There are no requirements to download and install the software environment.
- Can facilitate collaboration between remote developers.



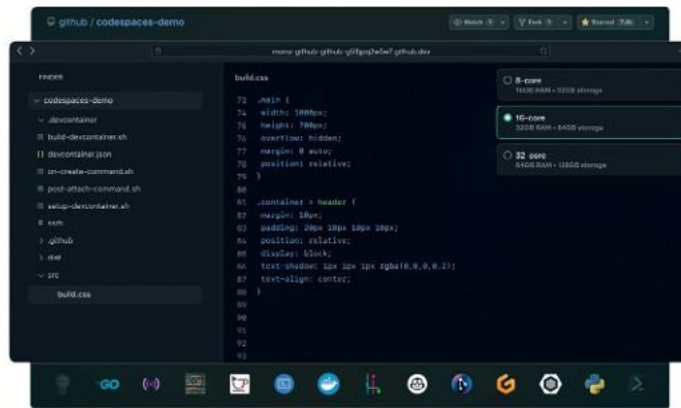


Figure 1.15: GitHub Codespaces environment

Programmers spend most of their programming time in testing and debugging, so integration of the code editor with the compiler and debugger is very convenient. This is the main feature of the IDE.

Specialized Tools for Specific Stages of Software Development

Creating professional software solutions requires working in a team and using a variety of tools that are not limited to the programming stage only, but extend to the process as a whole.

There are many tools that can be used during the SDLC of a software product, and it can be difficult to list all the software and other essentials needed to develop business software, but a selection of these tools are described below.

Prototype Creation

A software prototype is usually an organization chart, an image, or a set of images that show the functional elements of an application, or it may be a website used to map out applications or the structure and functionality of the website.

Examples of tools used:

- Pencil
- Balsamiq Mockups
- Figma

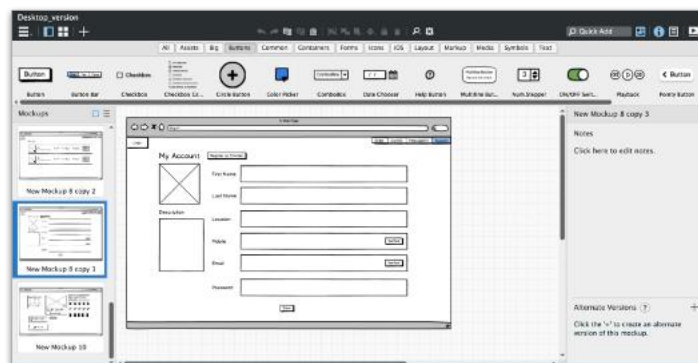


Figure 1.16: Balsamiq Mockups tool

Version Control Management - Source Code

The source code is subject to many modifications during the development process, and it may be necessary to undo certain steps in the program or reuse code that has been changed or deleted. When working in a team of programmers, two or more may need to work on the same files at the same time and make changes to the same code.

The tool we can use to control this process is called "version control management" or "code management". This tool enables the following:

1. Different team members can access the source code simultaneously without creating work conflicts.
2. Previous versions of code files can be kept for reference when some problems occur.

Version control uses a repository to record all changes made and creates a working copy of the project's code files, sometimes called a checkout copy, when a programmer wants to work on the code. All changes to the code are approved by the version control management software when changes to the code are saved to the repository.

Examples of tools used:

- Git
- Subversion
- Mercurial
- Azure DevOps
- DiffMerge

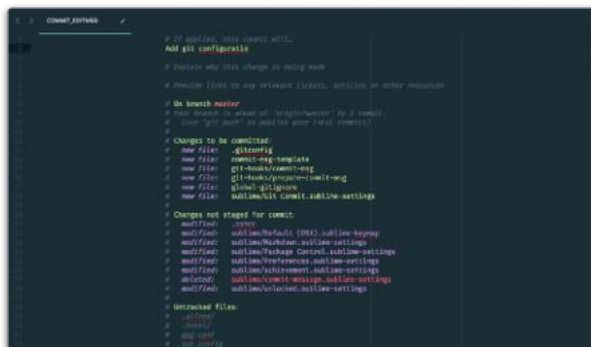


Figure 1.17: Version control management

Code Deployment

Until a few years ago, it was easy to deploy an application since the compiled output of the program was placed on a disk ready to use.

With the advent of the Internet, it became necessary to "publish" applications via the Internet, as installable software through application stores or directly as web applications, and accordingly special programs and tools appeared for publishing code on the web.

Examples of tools used:

- TeamCity
- Google Cloud Infrastructure Manager
- GitLab
- Jenkins
- AWS CodeDeploy
- Azure DevOps

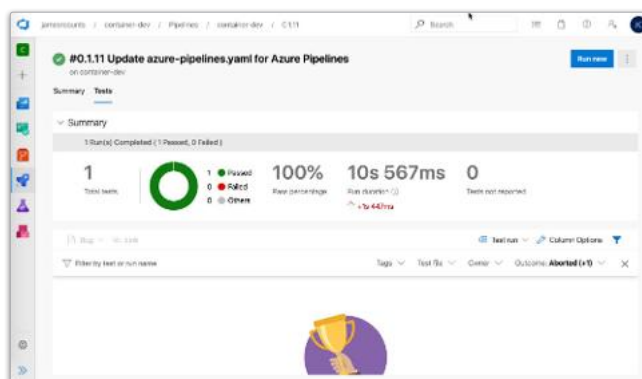


Figure 1.18: Code deployment

INFORMATION

"Branching" is a very useful feature of version control. It is the ability to copy all project code as a new parallel project to allow testing or making changes to create an updated or new version of the application. Parts of the new code can later be ported over to the original project to be used in the original application as well.

Testing

Testing is not just debugging the code, but also includes testing the operation of the program and the effectiveness of its use by a large number of users, as well as performing security and other tests.

Examples of tools used:

- Apache JMeter
- Selenium
- Azure DevOps
- Zed Attack Proxy
- Ghostlab
- Telerik Test Studio
- IronWASP
- Wapiti

Project Management, Collaboration and Issue Tracking

As we have already learned, having a successful product requires keeping track of the entire process and sharing knowledge with the entire team, especially when the team is expanding. This is where the project management process becomes especially important.

Examples of tools used:

- Microsoft Teams for collaboration and communication.
- Scrum Trello for Agile Planning and Tracking.
- Jira to track specific issues and manage projects.
- MeisterTask for task management.
- Slack for collaboration and communication.
- Basecamp for managing projects and communicating with clients.
- Azure DevOps for Application Lifecycle Management (ALM)

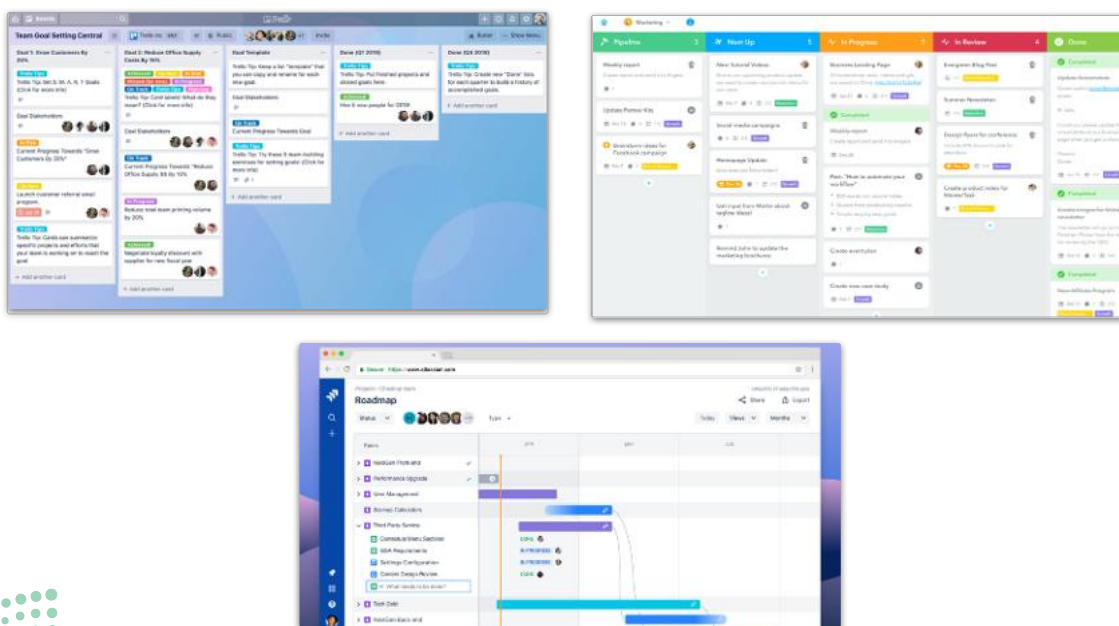


Figure 1.19: Examples of project management, collaboration and issue tracking tools

Using Development Tools to Provide Different Solutions

Development teams rely on the tools we described earlier to produce a wide range of IT solutions, many of which we use today to build applications of various kinds, such as:

- Web applications
- Smartphone Applications
- General applications
- Embedded systems

Building a Web Application

A web application is an interactive program that is built using HTML, CSS, and JavaScript web technologies, and which stores data on database servers. This application is used by users who perform tasks over the Internet.

Stages of Building a Web Application

1. The Ideation Stage:

Before creating a web application, we must set the goals and main idea of the application.

2. Market Research:

We must do what is called market analysis to find out:

- Whether the target consumer has a need for this product or service.
- Whether a similar product or service exists.

3. Define Web Application Functionality:

We must identify functions that provide solutions to the problems of the target market.

4. Wireframing or Prototyping:

Wireframing is about designing the layout of our web application, and prototyping takes the organization chart a step further by adding interactivity to test the functionality of the application.

5. Seek Validation:

At this stage, constructive opinions and feedback are collected from relevant parties and potential users regarding the design.

6. Architect and Build Database:

At this stage, the data needed by programmers and users is determined as well as the tool used to build the required database for the web application.



There are many database design tools that are used for different purposes, but the nature of the program and how it is proposed to deploy our software solution will determine the choice of a specific tool. Examples of tools used in designing and building databases are:

- MySQL
- Amazon DynamoDB
- MongoDB
- SQL Server
- Azure SQL
- Firebase

7. Building the Frontend

The front end is the visual element of our web application, and it represents the interface between the user and the system. This interface represents what the user sees and interacts with. Examples of tools used to build an optimized user interface for the web include:

- jQuery
- Django
- Angular
- Reactjs
- Vue.js

8. Building the Backend

The backend is used to manage the data in the program. It refers to the databases and servers as well as all other parts that are not visible to the user within the web application.

Building the backend includes writing the core code that provides the application's functionality, as well as preparing the database, the networks, and verifying the integration between the different subsystems. Security and performance considerations become of particular importance. Examples of tools used in building the backend are:

- Express JS
- Ruby on Rails
- Laravel
- ASP.NET
- Flask
- Spring Boot

9. Hosting our Web Application

To run our web application on a specific server, we need a web hosting provider. The hosting service may be simple and cheap, or it may be a large cloud computing service that allows our cloud infrastructure to grow as the number of application users grows and our needs grow.



Table 1.5: Web hosting providers

Types	Examples
Hosting Providers	• Bluehost • HostGator • GoDaddy • Rackspace
Cloud Service Providers	• IBM Cloud • Microsoft Azure • Amazon Web Services • Google Cloud Platform • Alibaba Cloud

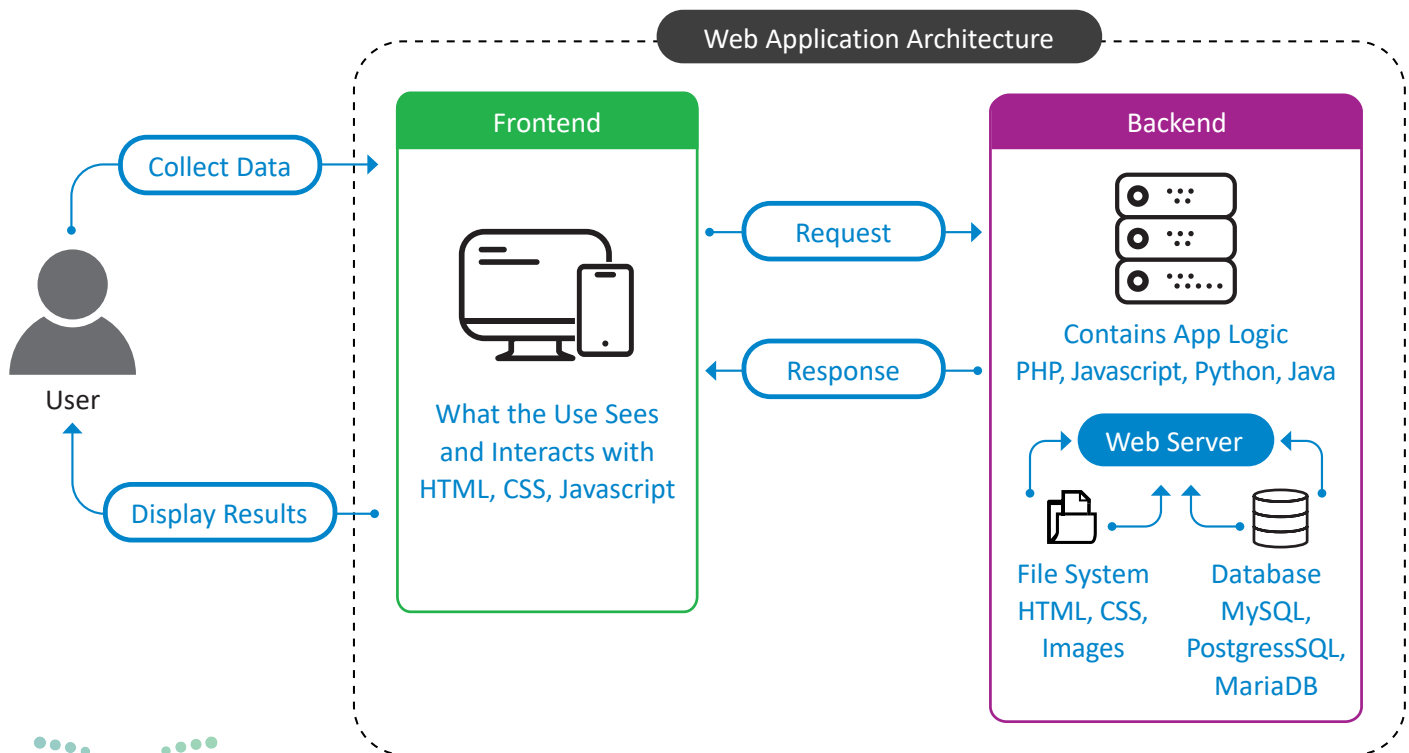


Figure 1.20: Basic web application architecture

The Cloud-Ready Application Architecture

It is preferable to develop and deploy cloud-based web applications as a set of cloud services. This process involves building data structures and then creating services, which are combined to form an integrated system.

The following diagram shows how to build a scalable and high-performance web application using Microsoft Azure services. The same concept applies to all cloud computing providers.

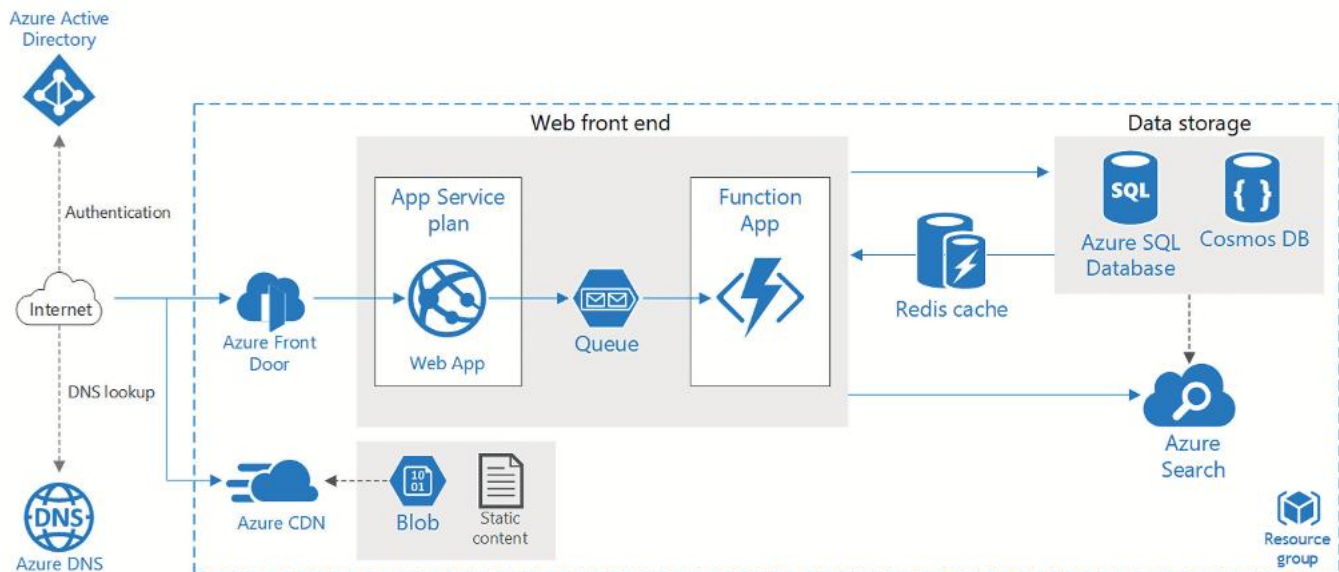


Figure 1.22: Building a web application with Microsoft Azure

The most important points to consider when using cloud application architecture are:

- The design of the application as a set of services.
- The separation of data, security and performance standards.
- The requirements for communication through networks between application components.
- The scalability of the design.
- System security must be a core part of the application and not something to be planned for later.
- The physical distance from users is the most important consideration when choosing data centers.



Building an Application for Smartphones

The steps for creating a mobile application are similar to those for a web application but with some special considerations. The mobile application is used on a phone which usually has a small screen, and, as the name suggests, the user will use the application "on the go", which means it is important to consider the convenience of the interface. The user should be able to adjust the screen size and access important information in a clear and simple way. It is also important to note that the difference in devices leads to the need for responsive applications.

The two major mobile platforms, iOS and Android, each support a different but similar set of technologies. For example, iOS recommends Xcode and Swift for software development, while Android recommends Android Studio and Java.

These environments only allow building a final app that is ready to be published to the specific app store in that environment. However, there are environments that try to solve this problem by supporting app deployment to multiple stores.

With the following tools, a single application can be developed in a way that runs in different software environments:

- NET MAUI
- React Native
- Ionic
- Kotlin

Testing mobile applications is a big challenge and it is difficult for a programmer or even a software development company to have all the mobile devices available in the market to do the testing. This is why there are online services that offer simulations for a wide range of mobile devices where we can simulate testing our application for compatibility on the different devices.

Examples of tools that enable application testing are:

- Appium
- BrowserStack
- Firebase Test Lab

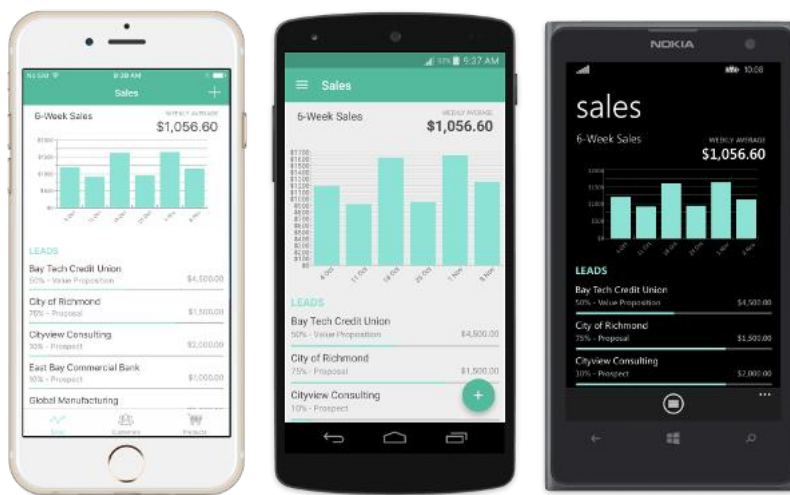


Figure 1.23: Same mobile application on different devices



Building a General-Purpose Application

General purpose software is a type of application that can be used to perform many tasks, as is the case with traditional office software such as word processors, graphic design software, Enterprise Resource Planning (ERP) business applications, or Customer Relationship Management (CRM) software.

Despite the focus of new software development technologies on the web and mobile applications, these traditional applications still retain their importance. The development of such applications relies on ready-made and reusable code libraries, especially user interface components and reporting tools.

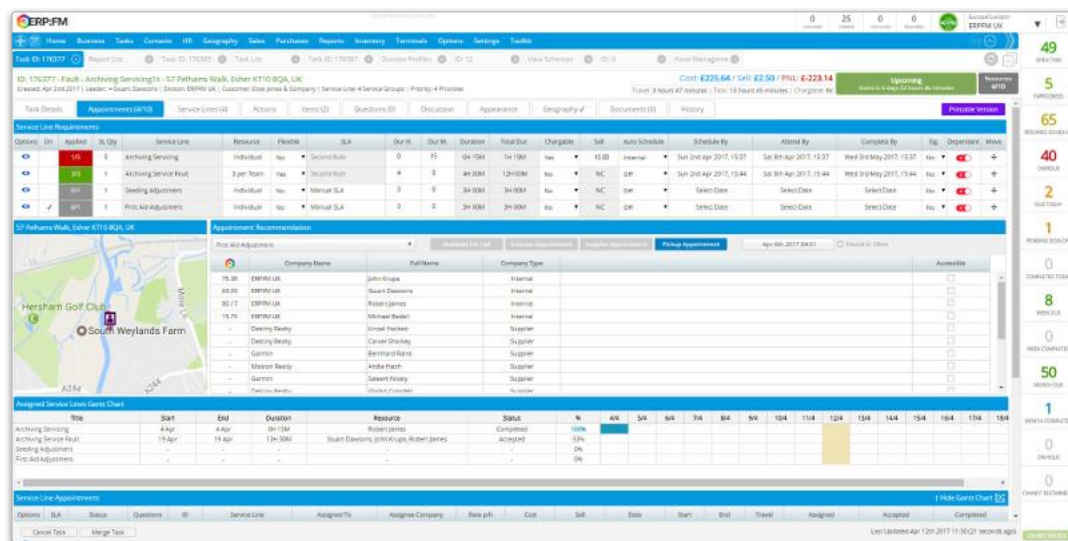


Figure 1.24: General-purpose application

Building an Embedded Application

An embedded system is a special computer with a real-time operating system, often without a user interface. Software on the embedded system handles sensors, actuators, and mechanisms for wired or wireless data exchange. These programs must be reliable, secure and fast. These applications require real-time operating systems such as RTLinux, Windows 10 IoT, and QNX as well as programming languages that are optimized for data processing, and network connectivity.

Examples of embedded systems are traffic lights, fire alarms and home security systems.

Embedded systems can be programmed using the following programming languages:

- Assembly language (difficult and unsuitable for practical use).
- C, Embedded C, nesC, Rust.
- Object Oriented languages such as C#, C++, and Java.



Figure 1.25: Embedded system

Exercises

1

Choose the correct answer:		
1. Project Builders:	Make sure that all the files you select will be compiled and linked into one final program.	☐
	Translate your program into executable code on the device.	☐
	Are necessary in the case of creating specialized programs related to networks.	☐
2. Code management tools:	Help you debug the program.	☐
	Work with databases and analyze the performance of queries on some databases.	☐
	Ensure that program files are not accidentally replaced when multiple programmers are working on the program concurrently.	☐
3. Profilers:	Provide or support a specific task in any state of the development or programming cycle.	☐
	Usually give us a good idea of the handling and needs of the program in terms of processor time and memory resources during operation.	☐
	Are special computers with real time operating systems and usually without a user interface.	☐



2 Choose the appropriate word to complete the sentences:

web application

version control

program prototype

word processors

general purpose

code editors

IDE

1. _____ help you write and make changes to the code.
2. _____ are not suitable for programming as they do not allow the easy formatting of code into code blocks.
3. A(n) _____ includes a code editor, compiler, linker and debugger.
4. A(n) _____ is an organization chart and is used for planning applications, website architecture, and functions.
5. _____ tools ensure that work is synchronously integrated by the various team members.
6. A(n) _____ is an interactive program that stores data on database servers and is used by a number of users who perform tasks over the Internet.
7. _____ programs are used to perform a wide range of tasks.



3 Match the following:

Text editor

1

☐ The visual elements of a web application; the interface between the user and the system.

Version control software

2

☐ Enables previous versions of code files to be preserved for reference when problems occur.

IDE

3

☐ A type of software used to modify text files.

Front end

4

☐ Manages your data, databases, servers and all components that the user can't see inside the web application.

Backend

5

☐ Contains all the software and tools needed to write and implement programs and to diagnose and fix problems.



4 What are the most important points to consider when using cloud application architecture?

5 What is meant by general purpose software? Give some examples.

6 Give four types of software development tools.



7 Give three examples of code editors.

8 List the basic steps for building a web application.

9 Give three basic features of an IDE.



Project

1

The Kingdom of Saudi Arabia has developed a vision for the future based on three primary themes: a vibrant society, a thriving economy and an ambitious nation. The Vision 2030 plan is the first step towards achieving Saudi Arabia's economic aspirations and transforming the lives of citizens.

2

Suppose you want to create a mobile application that provides information about tourism projects in Vision 2030. More specifically, the application aims to help elderly people with vision problems or trembling hands to browse for information on Vision 2030 megacity projects: Amaala, Neom and Qiddiya.

3

Search and find information and images about these projects that you will use when creating the application.

4

Then, create a summary of the SDLC of the application, where you will present what you will do in each phase of the SDLC.
Finally, create a presentation to illustrate this project.



Wrap up

Now you have learned to:

- > Differentiate between stages of the SDLC.
- > Classify the advantages and challenges of Waterfall, RAD and Agile methodologies.
- > Describe different programming languages, their history, classifications and areas of use.
- > Explain how a computer understands programming languages and deals with their errors through a compiler or an interpreter.
- > Identify the different software development tools and their uses in the different stages of software development and the production of different software solutions.

KEY TERMS

Agile Methodology	High-level Programming Language	Software Development Life Cycle (SDLC)
Assembly language	Integrated Development Environment (IDE)	Software Development Tool
Code Editor	Interpreter	Software Development Methodologies
Compiler	Lifecycle	Source Code
Development	Linker	Testing
Embedded System	Machine language	Version Control Management
Evaluation	Maintenance	Waterfall Methodology
Executable Program	Mobile Application	Web Application
Fourth-generation Language	Rapid Application Development (RAD)	
General-purpose Application		



2. Prototyping

In this unit, students will be able to compare the various ways of collecting the user requirements for a new system. Students will explain what a workflow diagram is, and they will learn to design a diagram with workflow processes. Finally, they will also learn how to prototype a mobile app using Pencil Project.

Learning Objectives

In this unit, you will learn to:

- > Recognize the requirements collection methods.
- > Recognize the types of charts used in the analysis phase.
- > Explain what analysis is.
- > Classify the functional and non-functional requirements.
- > List the data collection methods.
- > Describe workflow diagrams.
- > Use Pencil Project to design a workflow diagram.
- > Explain the human-computer interaction (HCI).
- > Illustrate the difference between user interface (UI) and user experience (UX) design.
- > Describe the basic functions/uses of mobile devices and desktop computers.
- > Identify the advantages and disadvantages of mobile devices and desktop computers.
- > Prototype a mobile app.

Tools

- > Pencil Project



Lesson 1

Analysis

Link to digital lesson



www.ien.edu.sa

Analysis Phase of the SDLC

As we have already mentioned in the previous lesson, SDLC can be broken down into five stages, the first of which is the Analysis phase. During the analysis stage, the system analyst meets with the users to determine exactly what the user wants and undertakes feasibility studies.

In the analysis stage, the details of the required system or any requirements raised by the customer are searched for, which are divided into two parts:

1. Functional requirements
2. Non-functional requirements

Functional Requirements

The official definition of a functional requirement is that it essentially specifies something the system should do. Some of the more typical functional requirements include:

- Business Rules and Administrative functions
- Transaction corrections, adjustments and cancellations
- Authentication and Authorization levels for the system
- External Interfaces for the system
- Certification Requirements for the system
- Reporting Requirements

Examples of functional requirements are:

1. The system must send a confirmation email whenever an order is placed.
2. The system must allow users to verify their accounts using their phone number.
3. The system must allow blog visitors to sign up for the newsletter by leaving their email.



Non-Functional Requirements

Non-functional software requirements are a set of constraints or criteria that define how a software system should behave, perform, or operate beyond its basic functional requirements. Some of the more typical non-functional requirements include:

- **Performance:** Requirements related to the speed, responsiveness, and scalability of the software system, such as response time, throughput, and resource usage.
- **Security:** Requirements for protecting sensitive data, such as user authentication, encryption, and access control.
- **Usability:** Requirements for ease of use and user experience, such as navigation, user interface design, and accessibility.
- **Reliability:** Requirements related to the software system's availability and stability, such as error handling, failover, and recovery.
- **Compatibility:** Requirements related to the compatibility of the software system with other systems, platforms, or devices, such as browser compatibility, mobile compatibility, and interoperability.

Examples of non-functional requirements are:

1. The ability of the system to recover unsaved data when a sudden power outage occurs.
2. The system works effectively when used by up to ten thousand users at the same time.

Now that the requirements have been defined, we can see how the gathering of these requirements can be realized and by which methods.

Requirements Gathering

An important point of analysis involves finding out what people want from the new proposed system or looking at the existing system to find out how it works and might be improved. There are several methods of data collection:

Questionnaires



Observation



Interviews



**Examination
of existing
documentation**



Questionnaires

A questionnaire could be given to each user and left with them for completion. Questions should be about how the job is done now and not about the overall running of the system. It could also be about the information the new system needs to give them.

Characteristics of Using Questionnaires:

- It is usually collected without identifying the user to get more credible answers.
- It takes less time compared to interviews.
- It can be analyzed automatically through the use of electronic forms and specialized software.

Challenges of Using Questionnaires:

- Incorrect answers are more likely due to unclear questions or the respondent's lack of interest.
- The questionnaires do not serve to collect descriptive data.

Interviews

Interviews take longer than questionnaires, so this method is appropriate when there are only a few users of the system. People at different levels in the organization, who will use the new system, should be interviewed. At these interviews you can find out how the existing system works and what is required from the new system.

Characteristics of Using interviews:

- An immediate explanation of the questions can be given by the interviewer when needed.
- Questions can be modified or changed to suit the team members being interviewed.
- People usually take an interview more seriously than a questionnaire.

Challenges of Using Interviews:

- Interested persons may get nervous during the interview, which affects the accuracy of the information provided.
- Interviews are expensive due to the need to visit people's whereabouts and disrupt their regular work.
- Arranging and conducting interviews takes a lot of time, especially when many people need to be interviewed.



Observation

In this method, an observer is with a user who is actually doing the job the new system is designed to do. Then the observer sees the problems encountered with the old system as well as talks to the user about what the new system must be able to do.

Characteristics of Using Observation:

- We can instantly identify the processes involved within the system.
- The analyst gets acquainted with accurate details in the current system that are difficult to obtain through questionnaires and interviews.
- It is less expensive than interviews as it does not require users to be interrupted while performing tasks.

Challenges of Using Observation:

- Using this method requires knowledge of the current system and the new system.
- The person being observed may act differently than they normally would during the observation.

Examination of Existing Documentation

This method of data collection involves examining any paperwork involved with the current system. This would include documents such as order forms, lists of stock, and so on. We can also look at the records that are kept in filing cabinets

Characteristics of Using Examination:

- It saves a lot of time, especially in the case of previous system analysis documents.
- The documents provide a clear picture of the process of data flowing through the system.
- The documents allow the person doing the analysis to determine the size of the system needed by looking at order volume, invoices, etc.
- The documents provide a clear picture of the current input and output designs of the system.

Challenges of Using Examination:

- It depends greatly on the quality of the documents in the organization and the accuracy of the data it provides.
- The process of collecting and analyzing documents is expensive and requires a lot of effort from those who carry out the collection and analysis.



Here is a comparison of the advantages and the disadvantages of the different ways to collect the requirements of the system in the Analysis phase.

Table 2.1: Comparison of the ways of collecting user requirements

Methods	Advantages	Disadvantages
Questionnaires	Users can be honest when they answer anonymous questionnaires.	Questionnaires may not be clear or well understood.
	It takes less time to collect information from a lot of people.	You cannot collect all the information you want via a questionnaire.
Interviews	The questions can be adjusted for specific users depending on their position or other criteria.	Users may not give honest answers since there is no anonymity.
	The participants (users) will take the interview or the focus group seriously.	Interviews are time consuming for the analysts but also for the users that will need to be taken away from their job. You cannot interview a lot of people, owing to it is an expensive process.
Observation	The analyst can get a real understanding of the existing system. The users can continue working.	Users may not work in a normal way as observation may be intimidating.

It is important to note that the criteria for choosing the method of data collection may differ according to the nature of the organization's work, the number of people targeted in the data collection process, etc. Usually, more than one method is used to collect data to obtain accurate and realistic outputs.

Using Diagrams in the Analysis Phase

Diagrams and charts are useful tools that can help us in the Analysis phase, especially workflow diagrams. But, before we start working with workflow diagrams, let's think about what exactly a diagram is. A diagram is a visualized representation of information, using shapes and arrows to show arrangements and relations.

Why Do we Use Diagrams

Through diagrams, we can better explain statistical data, system functions, organization charts and other processes. The visual representation of such information is more effective. For example, using shapes and different colors in a diagram, makes it easier for the reader to compare data and differentiate each result.

Diagrams are used in a wide range of applications. We can use a diagram to show the organization chart of a company, the flow of the processes for a task to be completed or how network components are connected and related.

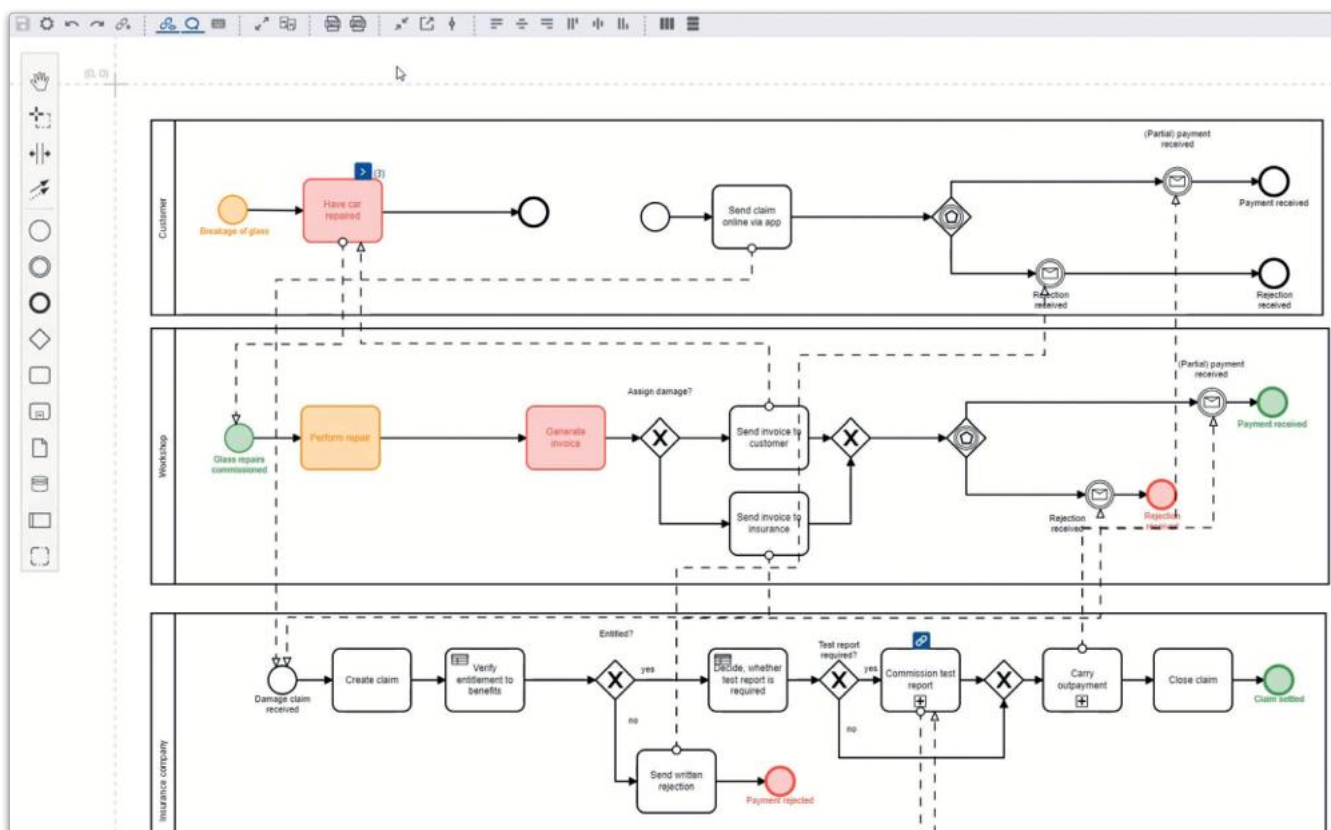


Figure 2.1: Diagramming a process



Different Diagram Examples

There are many different kinds of diagrams we can use during the different phases of a Software development life cycle. Some of them are:

Workflow Diagram

We can say that the Workflow diagram is similar to the flowchart diagram you learned to design in order to describe the algorithm of a program. Typically, it consists of a set of symbols representing actions and processes connected by arrows indicating the flow from one to another.

We can use workflow diagrams to show the flow of tasks during each phase of a SDLC.

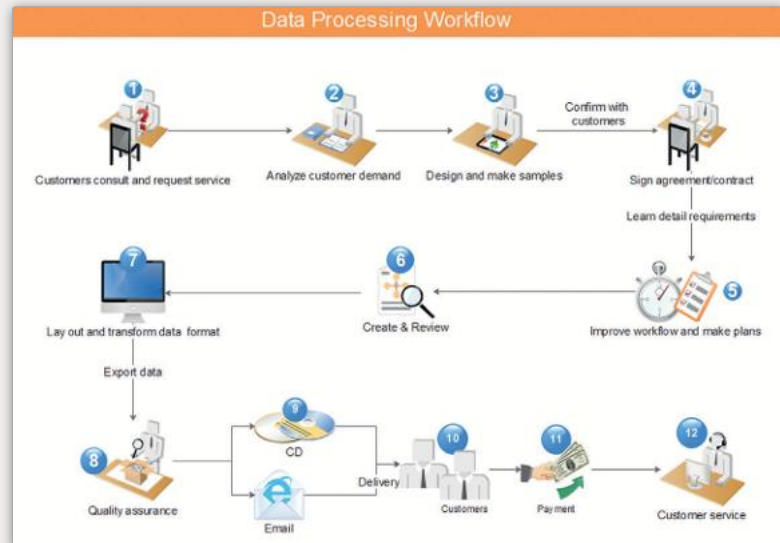


Figure 2.2: Workflow diagram

Tree Diagram

A Tree diagram represents the hierarchical nature or organization of a structure in a graphical form. The root is usually at the top and the tree elements, called nodes, at the bottom.

It is widely used to show how a company or the tasks of a project are organized. It can also be used to represent conditional probabilities in mathematics.

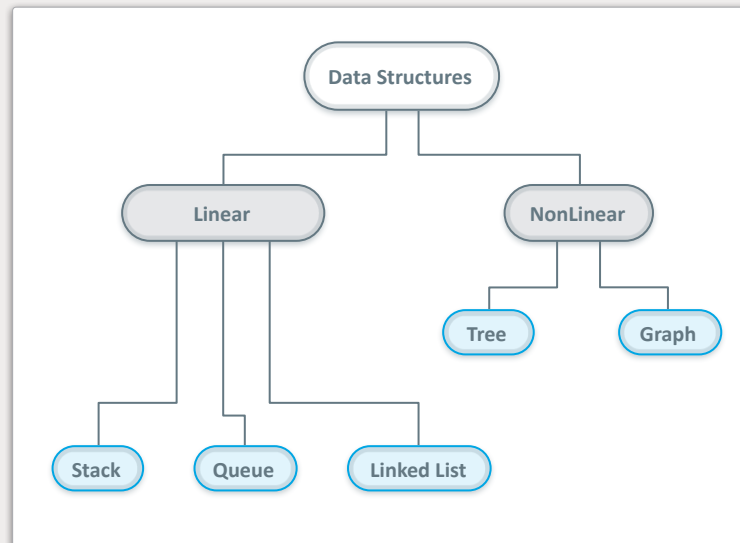


Figure 2.3: Tree diagram



Wireframe Diagram

A Wireframe diagram is a visual representation of the framework of a website or an online App. This chart usually lacks typographic style or graphics, since its purpose is to focus on the content's structure and functionality. It is widely used in website and applications development.

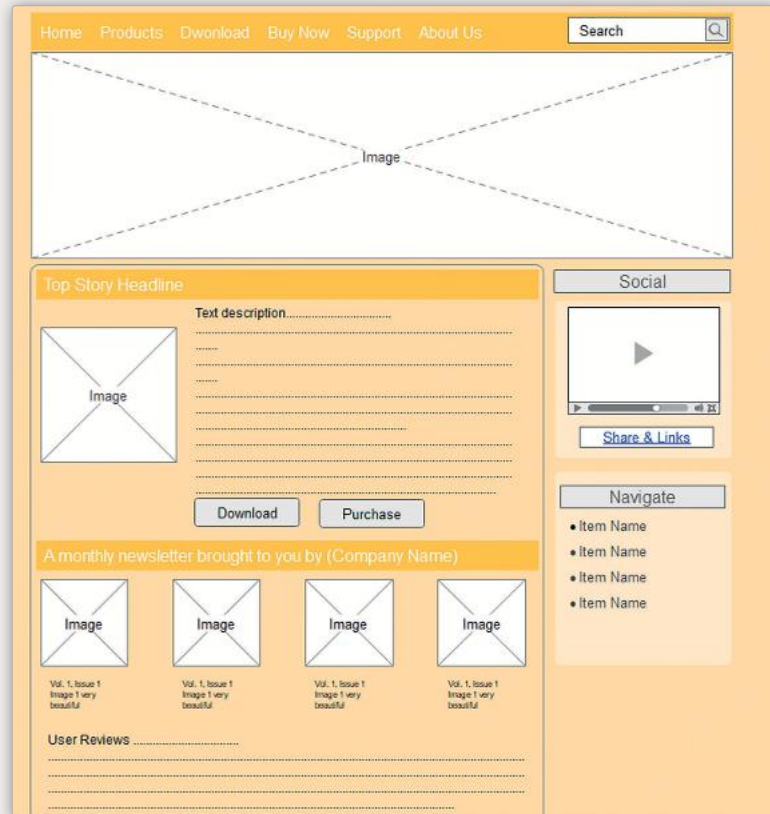


Figure 2.4: Wireframe diagram

Use Case Diagram

A Use Case diagram is a type of diagram that represents the different ways a user might interact with a system. Use case diagrams are very helpful to represent the gathered requirements of a system during the Analysis phase of a SDLC.

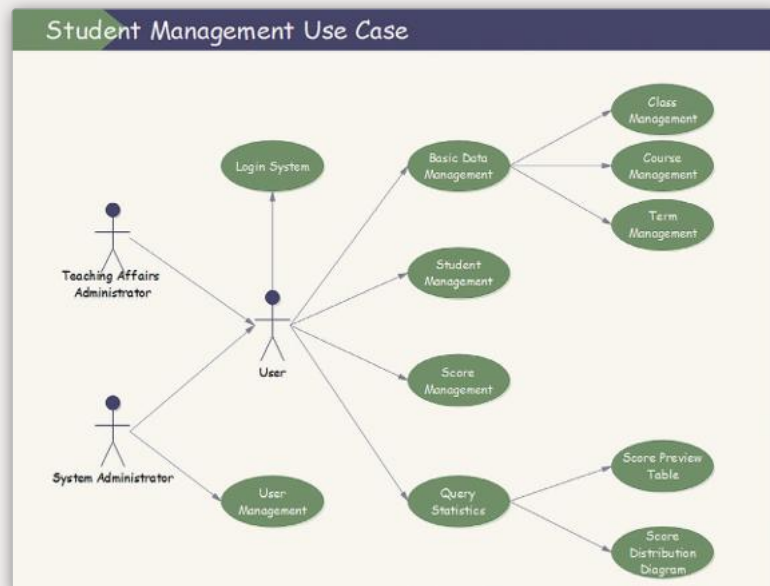


Figure 2.5: Use case diagram





Using Pencil Project to Design a Workflow Diagram

Pencil Project is a free and open-source GUI (Graphical User Interface) prototyping tool for making diagrams, using built-in drawing features and shape collections. We can use Pencil Project to create almost every type of diagram, such as flowcharts, workflows and wireframes.

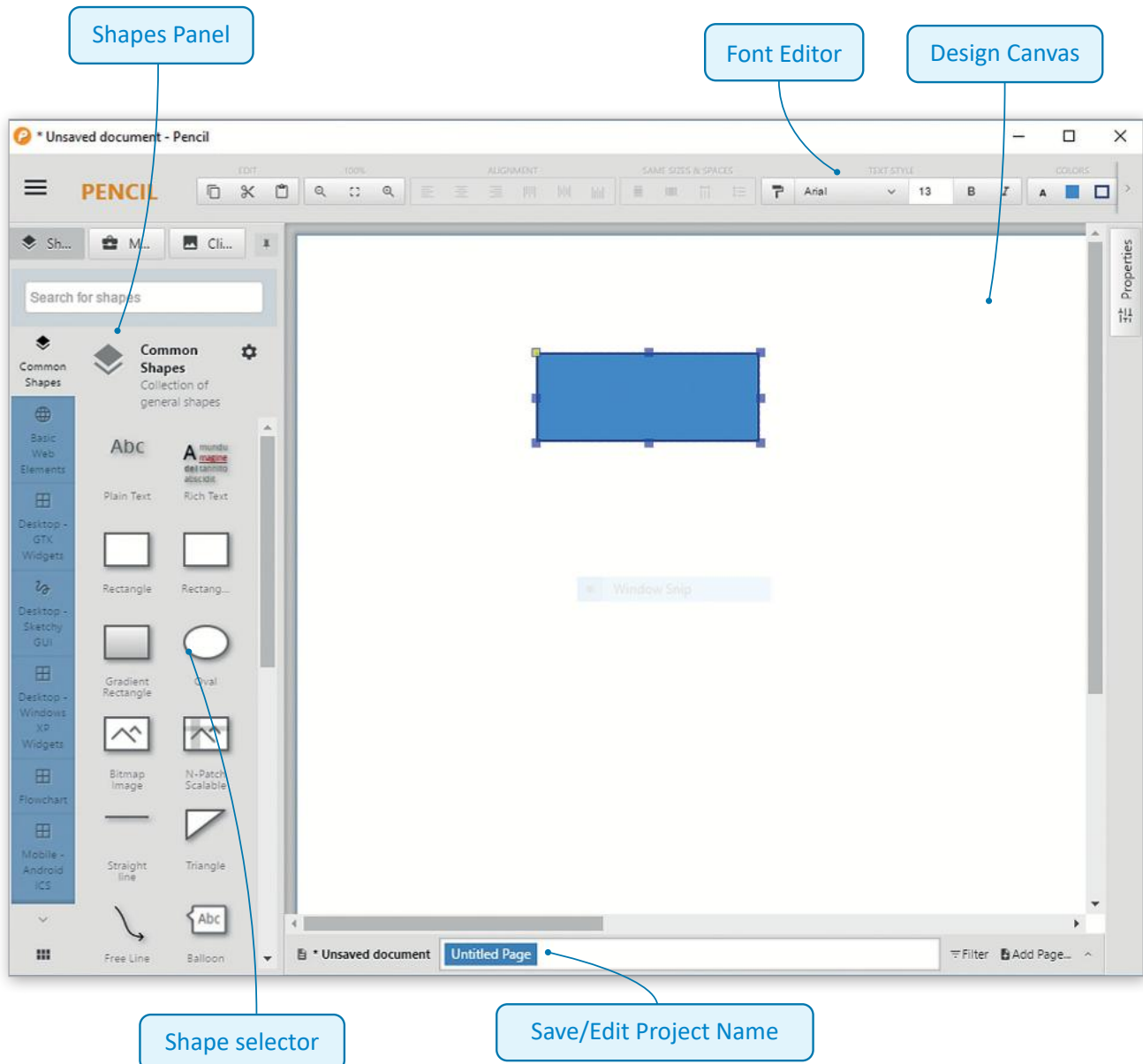


Figure 2.6: The Pencil interface

INFORMATION

In addition to the built-in shapes included in the program, more shapes and cliparts can be imported from the Internet to enrich the existing library shapes.



Basic Shapes of a Workflow Diagram

There are different symbols we can use to represent different aspects of a workflow diagram. For example, a process is represented by a rectangle while a diamond is used to represent a decision. The following table shows some basic shapes which are usually used in a workflow diagram.

Table 2.2: Basic shapes of workflow diagrams

Symbol	Name	Description
	Start / End	Represents a start or an end point for the workflow.
	Process	Represents a repeatable set of steps.
	Decision	Represents a decision needed to be made, leading to a process or another decision.
	Document	Represents a document such as error reports or other types of reports and outcome documents.
	Input / Output	Represents the input or output of Data.
	Arrows	A connector that shows relationships between the processes.

Creating a New Diagram

In this lesson, we will use Pencil Project to create a workflow diagram; the diagram will outline the maintenance of the application about the KSA tourist guide for elderly people with vision problems, which we will create later.

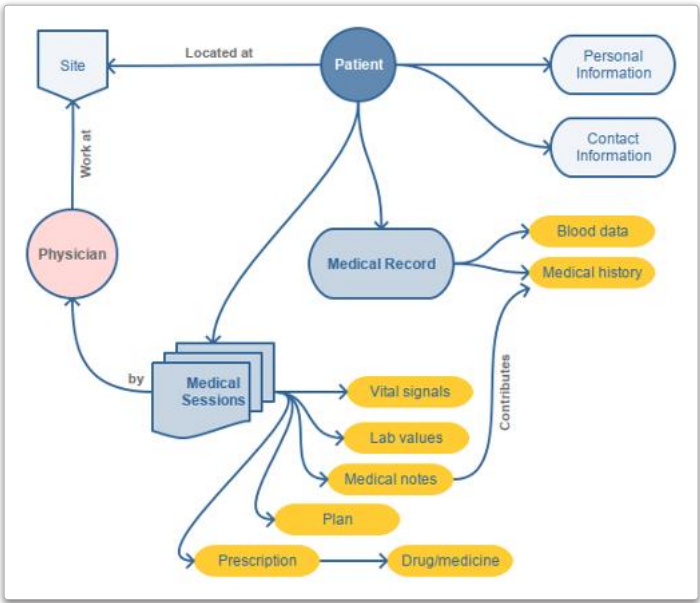


Figure 2.7: Application workflow diagram



To create a workflow diagram:

- > Open **Pencil** program and click **Create a New Document**. ①
- > On the **Shapes** panel click the **Flowchart** section to add a shape. ②
- > Drag and drop the **Terminator** shape into the canvas to set the starting point of the diagram. ③
- > The starting point of the diagram has been created. ④

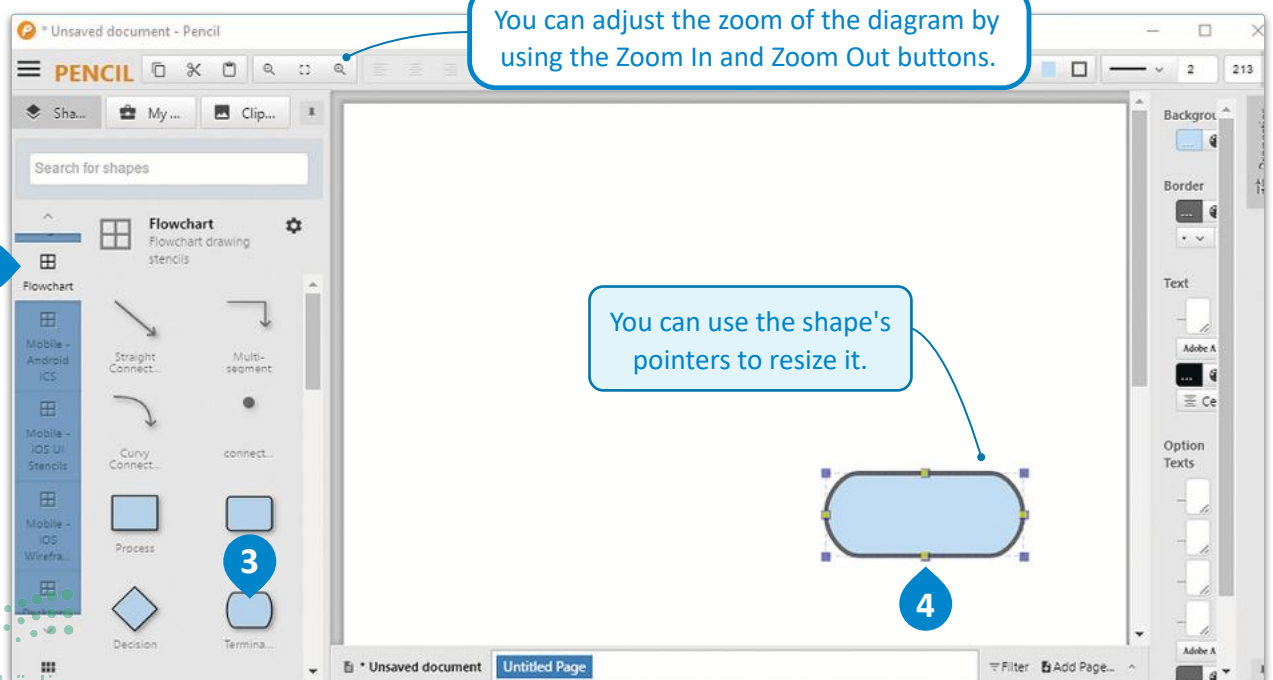
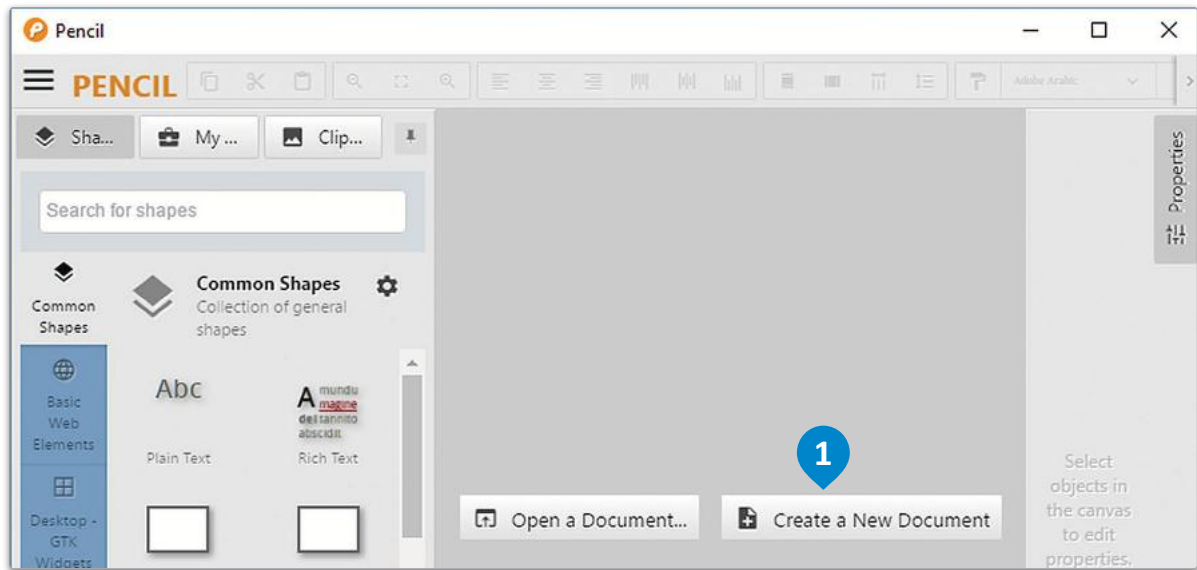


Figure 2.8: Creating a new workflow diagram

To add text to the shape:

- > Double click the shape you want to add text to, and then type the text you want. **1**

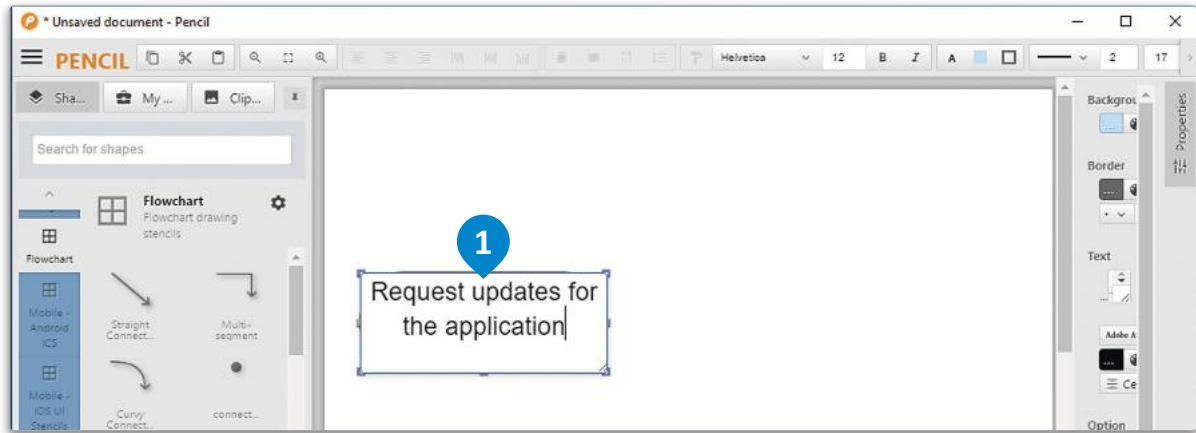


Figure 2.9: Adding text to a shape

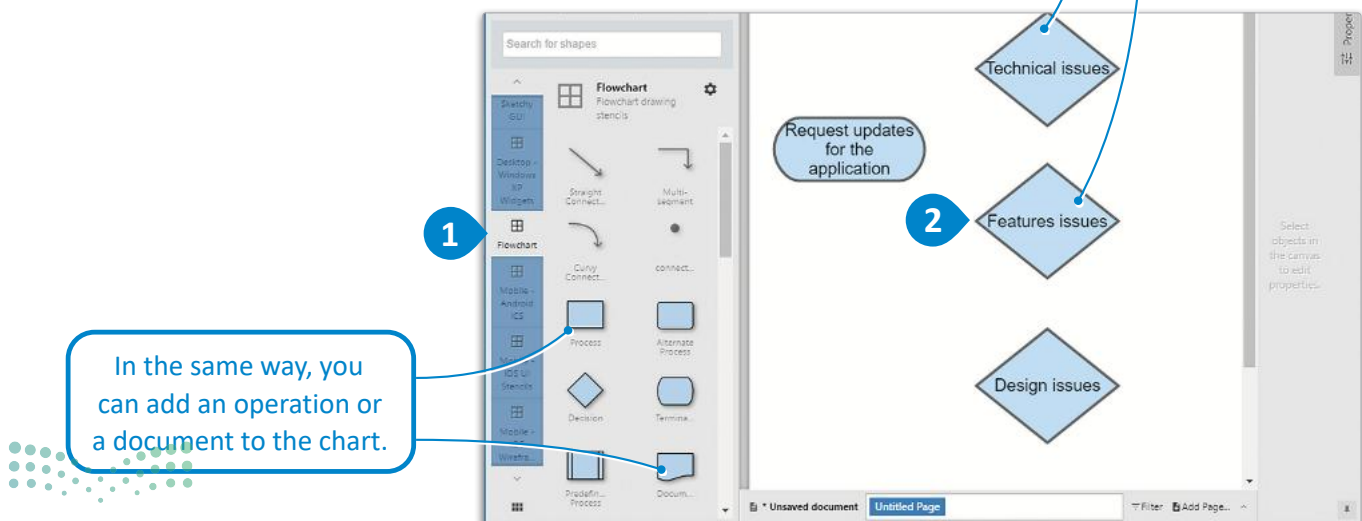
Adding New Shapes to the Chart

We can add new shapes that represent decisions, processes, documents, or any other information we want to add to the workflow diagram.

To add new shapes to a diagram:

- > To add a process, go to the **Shapes** panel, click the **Flowchart** section **1** and then drag and drop the **Decision** shape into the canvas.
- > The shape has been added to the diagram. **2**

You can copy any shape or text field.



In the same way, you can add an operation or a document to the chart.

Figure 2.10: Adding new shapes to a diagram

Adding Links and Texts

It is necessary to add links to represent the connection and relationships between the different shapes of the diagram, and we can add simple text when needed to explain or analyze the different outputs of a decision, process, or any other related form within the diagram.

To connect two shapes:

- > On the **Shapes** panel, click the **Flowchart** section and drag and drop a **Multi-segment Connector** into the canvas. **1**
- > Use the connector pointers to connect the starting point of the diagram with the next three decisions. **2**
- > Continue connecting all the shapes of the diagram with the appropriate connectors. **3**

To make the diagram fit on the canvas, you just have to resize the canvas by right-clicking on it and choose one of the three options:

Fit Content
Fit Content with Padding...
Fit Screen

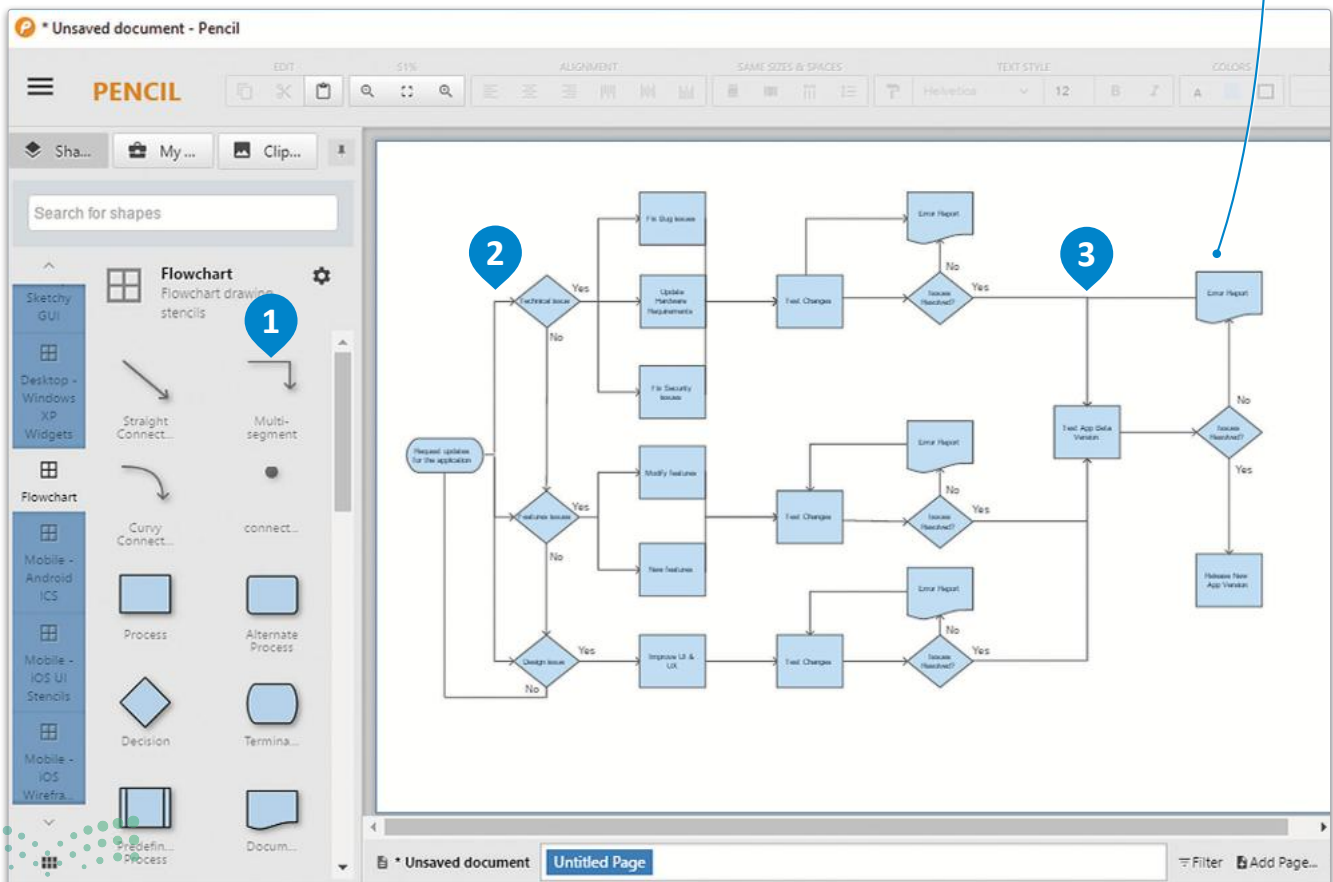


Figure 2.11: Connecting shapes in the diagram

To add a text block to the diagram:

- > On the **Shapes** panel click the **Common Shapes** section and drag and drop a **Rich Text** field into the canvas. **1**
- > Place the text field in the desired position inside the diagram and type the text you want. **2**
- > The text block has been added to the diagram. **3**
- > From the **Shapes** panel, click the **Common Shapes** section, drag and drop the **Plain Text** field onto the canvas, **4** to insert **Yes/No** options at the graph's **Decisions**. **5**

You can change the size of the text to make it more clear from the text style section.

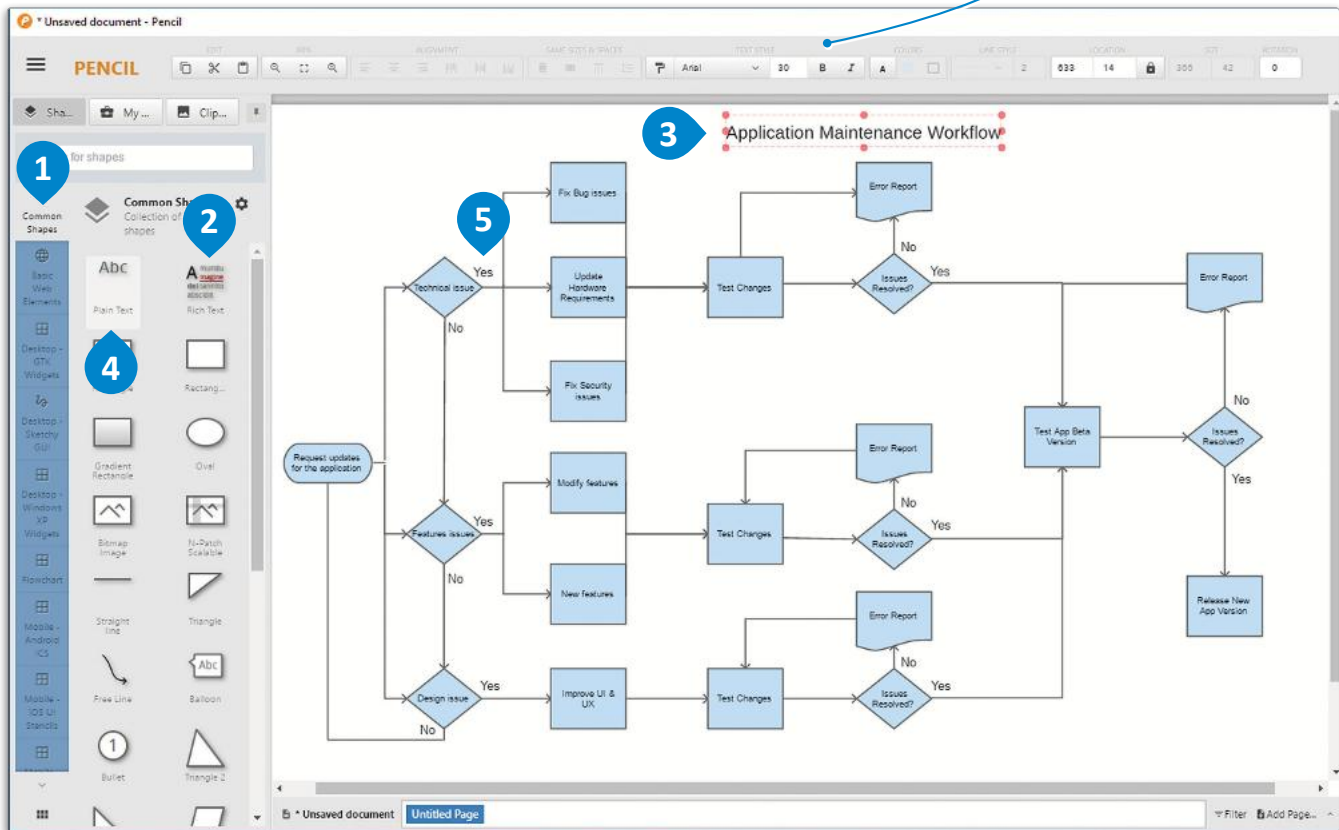


Figure 2.12: Adding a text block to the diagram



Saving the Diagram and Export Options

When the final diagram is finished, we can save the file and export it in various formats such as Image (PNG), PDF, Document or a Web Page.

To save a diagram project:

- > Click the main menu **1** and click **Save as**. **2**
- > In the appearing window type a name **3** for the file to save and click **Save**. **4**
- > The diagram has been saved.

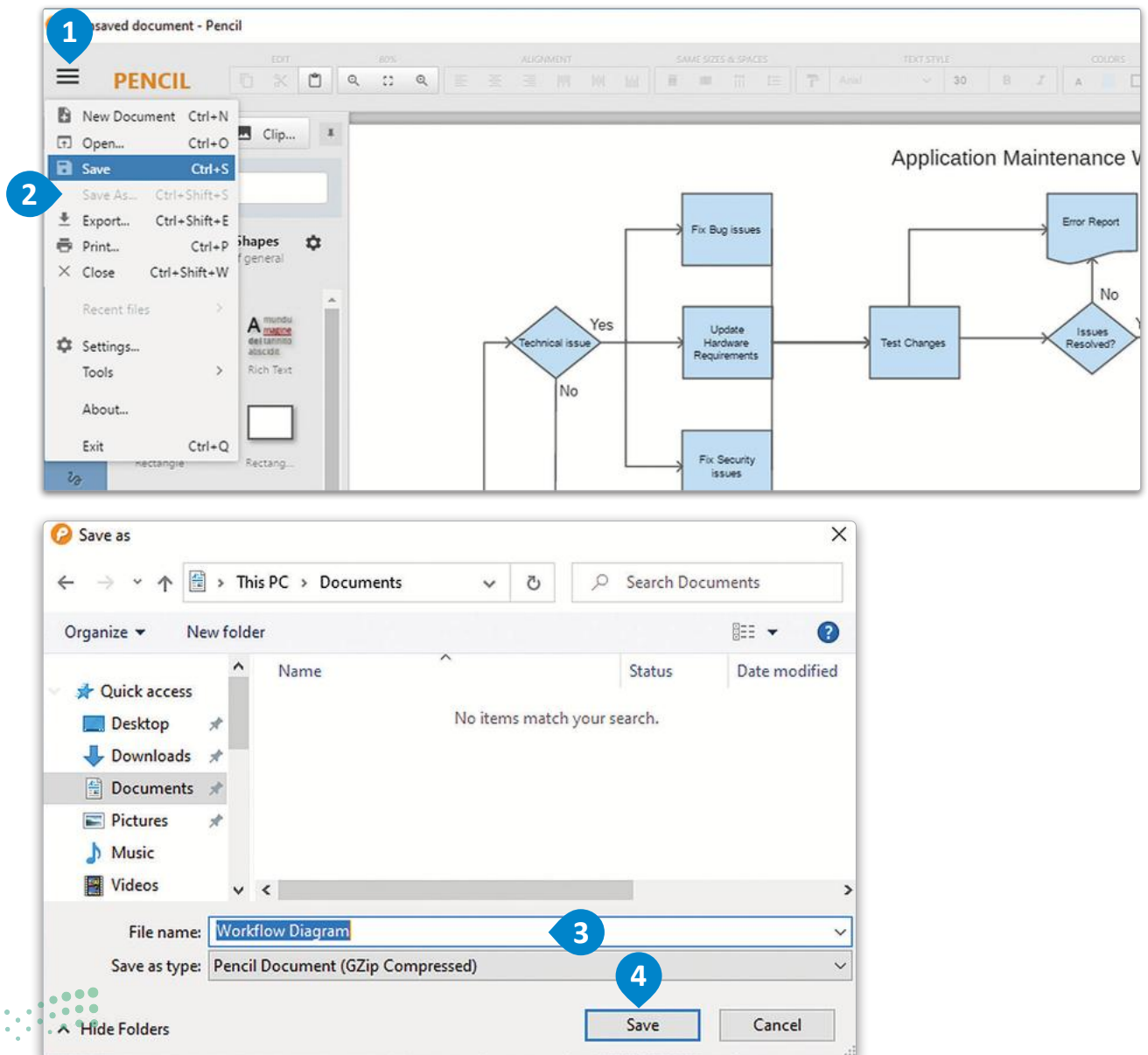


Figure 2.13: Saving a diagram project

To export the project:

- > Click the main menu **1** and click **Export**. **2**
- > In the **Export Document** window, click the **Output Type** to select the type of the exported diagram. **3**
- > Select the type you want (e.g. PDF) **4** and click Export. **5**

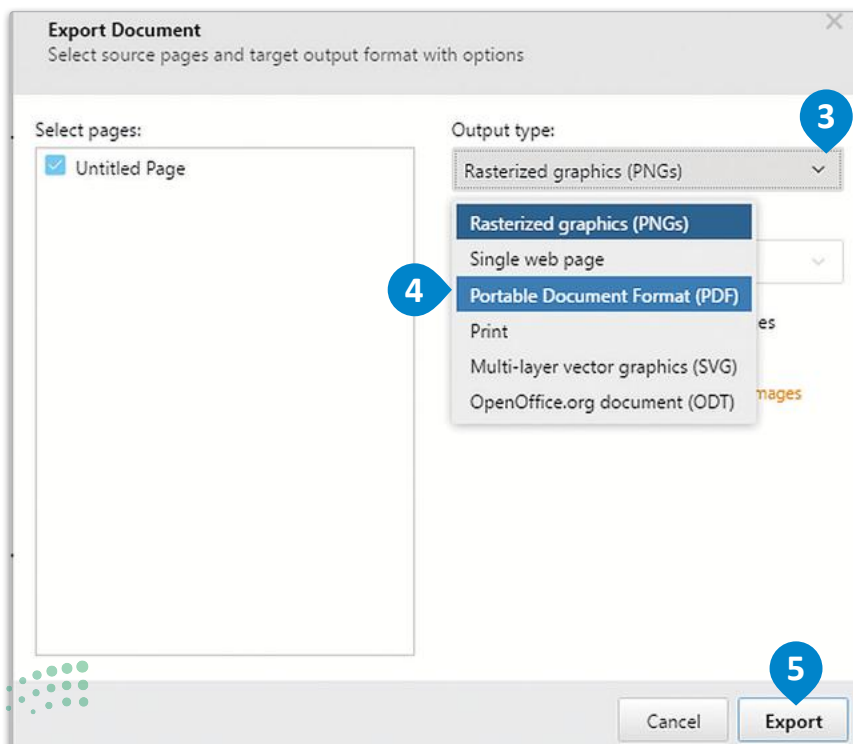
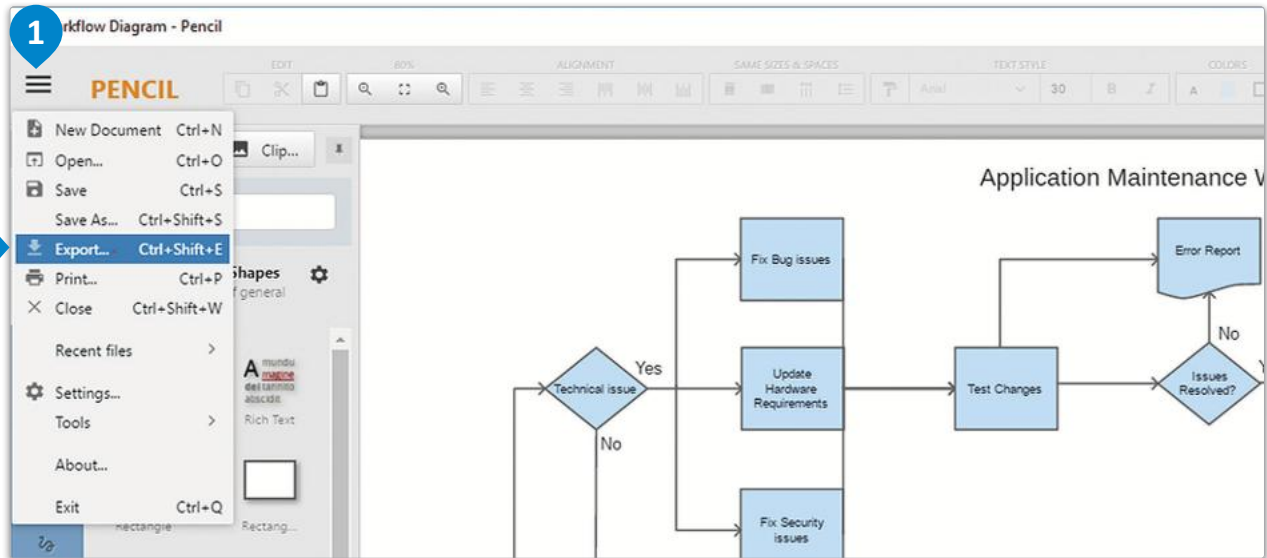


Figure 2.14: Exporting a diagram project

Exercises

1 Open Pencil Project and see what the following shapes represent:

Start / End Point

1



Document

2



Process

3



Input/Output Data

4



Decision

5



Connector

6



2 Match each of the following requirements with their examples:

Functional requirement	1	☐	Security
		☐	Administrative functions
		☐	External Interfaces for the system
		☐	Reporting Requirements
Non-Functional requirement	2	☐	Performance
		☐	Certification Requirements for the system
		☐	Reliability

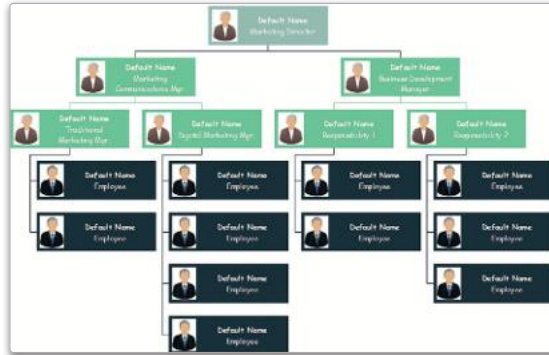


3

Read the sentences and tick ✓ True or False.	True	False
1. The identity of the interviewee can be kept anonymous.	☐	☐
2. The observation process must take place while users are using the system.	☐	☐
3. Examination of existing system documentation can show current output and input designs.	☐	☐
4. Inadequate answers regarding system functions can be obtained by examining the existing documentation.	☐	☐
5. The answers provided through questionnaires are more realistic.	☐	☐
6. An additional explanation of the questions can be provided during the questionnaires if the person has difficulty understanding the meaning of a question.	☐	☐
7. The person to be observed may act differently from his nature during the observation.	☐	☐



4 Match each of the following requirements with their examples:



Wireframe diagram

Tree diagram

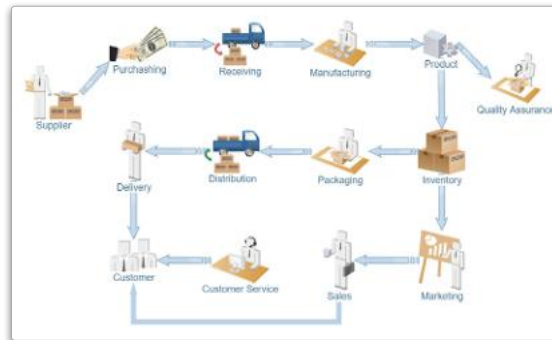
Workflow diagram



Wireframe diagram

Workflow diagram

Tree diagram



Use case diagram

Workflow diagram

Tree diagram

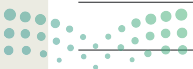
5 List one use for each of the following diagrams:

1. Workflow diagram:

2. Use case diagram:

3. Tree diagram:

4. Wireframe diagram:





Lesson 2

Interaction Between the User and the Computer

What is Human-Computer Interaction (HCI)?

Human-Computer Interaction (HCI) refers to the study of interaction between humans and computers, and is concerned with designing and adapting systems for human use, with a focus on designing interfaces used by people (users) and computers.

Researchers in this field note the ways in which humans interact with computers and the various design techniques that allow humans to interact with computers in new ways.

As mentioned earlier, the principle of HCI consists of three components: the user, the computer, and the interaction loop, which is defined as the flow of information between the human and the computer.

Human-Computer Interaction Majors

The study of human-computer interaction extends to draw data from the fields of human factors engineering and cognitive science as well as computer science.

HCI is concerned with the cognitive and academic aspects of user behavior, whose outputs are essential inputs to the applied field upon which User Experience (UX) and User Interface (UI) designs are based for various applications such as smartphone applications and websites.

Thus, the collaboration between researchers in the field of HCI and designers on the user experience and interface ultimately leads to perfect designs that meet the needs of users.

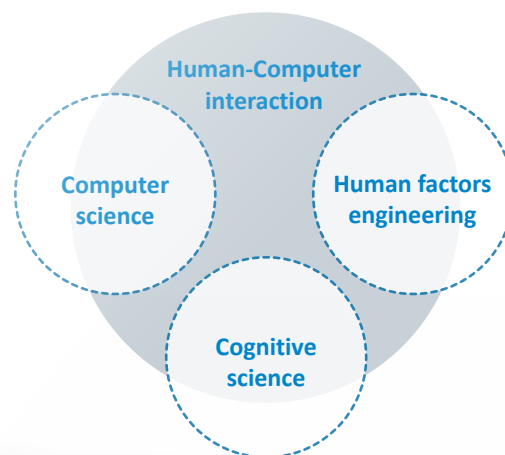


Figure 2.15: Human-Computer Interaction



User Experience Design (UX)

User experience (UX) refers to a person's impressions and attitudes about using a particular product, system, or service. This includes the practical and emotional aspects of human-computer interaction. User experience also includes a user's perception of various aspects of the system such as utilities, ease of use, and efficiency, and this concept can be applied to any system such as ATMs, cars, phones, etc.

Key factors affecting user experience:

The concept of user experience has expanded to include many aspects in addition to usability, and it has become important to pay attention to all aspects of user experience in order to deliver successful products to the market.

To improve user experience, the design, contents and functionality of the system should be:

- 1. Useful:** Meets users' needs.
- 2. Usable:** The system can be used easily and intuitively.
- 3. Attractive in appearance:** The design elements are used in a unique way that attracts the user and gives the system its own identity.
- 4. Easily Findable:** Its contents can be easily browsed and accessed from within or outside the system.
- 5. Accessible:** The design should include users with special needs in its characteristics.
- 6. Credible:** The system derives its content from reliable and approved sources.
- 7. Valuable:** The product must deliver value to the business which creates it and to the user who buys or uses it.

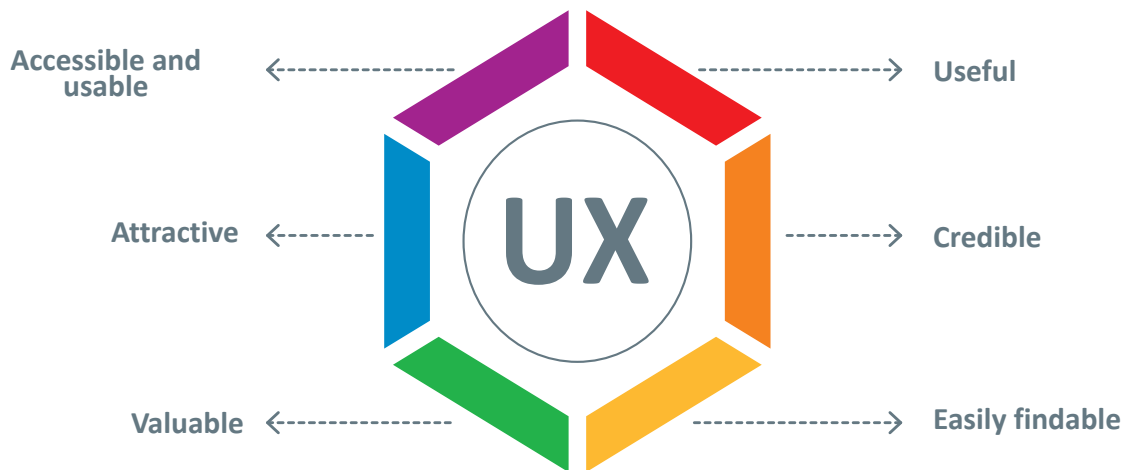


Figure 2.16: Key factors affecting UX



User Interface (UI)

The user interface (UI) is the point of interaction and communication between the human and the computer inside the device. It can be said that the user interface is the graphic form of the application, and it contains buttons, readable texts, images, scroll bars, and text entry fields, as well as many other elements that the user interacts with, including screen layout, transitions, GUI animations, all the interaction subtleties and any animation effects that need to be designed.



Figure 2.17: UI for tablet devices

The task of user interface (UI) designers is to define the look and of both the application and the interface. They must choose color schemes, button shapes, font widths, and text font types.

Considerations for creating a good user interface:

User interface design considerations are related to many other sciences and disciplines such as psychology and the visual arts, including the following:

1. The shape of buttons and other elements should be indicative of their function and the design should allow the user to easily explore these functions.



2. Interfaces must be designed and laid out properly and ergonomically for the user, so that controls are adjacent to their related objects.



3. Interface elements must take into account the visual capabilities of the user in terms of font size, text adjustment, and color brightness and contrast.

clear text

4. The interface should provide its users with messages and signals showing the system's response to the commands to be executed and providing feedback. Below are some examples:

Product added. 

Warning: Product description cannot be empty. 

The product could not be added. Ensure that the product name is valid. 

5. Consider providing as many default settings as possible to reduce the burden on the user (for example, providing pre-populated forms). Below are some examples:

What do you want to see next?

Do you have any product suggestions or ideas? We are all ears.

Name

Khaled Abdullah

Autofill

E-mail address

khaled@email.com

Autofill

Tell us why you like this idea?

Send

Idea details



How Do UX and UI Work Together?

The concepts of UX and UI are often confused in web design and smartphone applications. The difference is that user interface is concerned with the graphical layout of an application or website, while user experience focuses on how easy or difficult it is to interact with user interface elements. Therefore, user experience usually dictates user interface specifications.

Desktop Computer and Smartphone

The use of mobile devices has now overtaken the use of computers as the way most people browse, shop, use social media and perform other online tasks. Therefore, it is important to consider the device that the user is using when developing websites and smart applications, which we will address in the coming lessons.

When designing applications and websites, it should be taken into account how they will work on all devices (mobile and desktop), and how the user experience differs when using the application on the mobile phone from using the desktop computer, and to understand this, you must realize the important factors that make the mobile phone different. Once you understand this difference, you can take these factors into consideration when designing your mobile application or making your decision to design the website.

The Main Characteristics of Desktop Computers and Mobile Phones

In the following table, the characteristics of mobile devices (mobile phones and tablets) and computers (desktops and laptops) are illustrated.

Table 2.3: Main device properties

Property	Desktop and laptop computers	Mobile phones/tablets
Screen size	Desktop computers can connect to multiple screens, which enables you to choose what suits you best. Desktop or laptop screens are usually between 15-30 inches.	Mobile phone screens vary by manufacturer and model. However, they are always smaller than desktop or laptop computers, and the screen size usually ranges between four and seven inches.
Screen resolution	The smallest laptop screen contains 2304 x 1440 pixels.	Most mobile devices have fewer pixels than desktop computers, for example, a high-quality smartphone screen has 1334 x 750 pixels.
Transport	Although laptops are usually light and portable by design, they cannot compete with smartphones in this regard.	Mobile phones are lightweight and can fit in a purse or pocket, they are designed to go everywhere with you. A tablet may not fit in your pocket, but it is still light and can be carried in one hand.

Property	Desktop and laptop computers	Mobile phones/tablets
Input methods	The keyboard or mouse is used for input, and it is smooth and easy to use for the majority of users, and it comes in different sizes.	Mobile phones may have an on-screen keyboard or touch screen that is much smaller than a computer, and users with large fingers or vision problems can have difficulty typing.
Software	Because of their less restrictive size, components, and power requirements, desktop and laptop computers can run more powerful software than a smartphone or tablet.	Despite the huge development in the capabilities of mobile phones, they still fall short of running huge programs compared to the performance of traditional desktop or laptop computers.
Operating system	Desktop and laptop operating systems are designed to take advantage of fast CPUs, large disk space, large amounts of RAM, and use new chipset features that most mobile devices don't have. Microsoft's operating system is closed source.	Mobile operating systems (Android) and (iOS) are designed to run on a specific set of devices without giving you full access to your system hardware. They also have a strict system of hardware requirements because the mobile app ecosystem is closely tied to specific hardware features. In other words, you can't run the latest apps on an older mobile OS, and vice versa. The Android operating system is open source.
Internet Connection	Desktop computers are characterized by the ability to connect to the wired Internet network Ethernet, and most of them require the use of a wireless network card USB Wi-Fi Adapter to connect to the wireless network, most laptops contain a wireless and wired network card.	Mobile phones and tablets can connect to Wi-Fi networks to access the Internet. Smartphones and most tablets can also connect to a mobile data network, which allows access to the Internet from almost anywhere, but it can be more expensive.

Functional Differences between Mobile Phones and Desktop Computers

Mobile phones and computers have different functions and both are important in their own way. Mobile phones provide the flexibility necessary for the user to search online or use e-mail anywhere, while the computer is used for more complex tasks. Using both at the same time can facilitate your work and help you accomplish your tasks.

Android User Interface and Windows Operating System

The increasing reliance of many companies on web applications and mobile applications has led companies to focus on improving the user interface in order to improve the overall user experience, so there is a wide variety of types of user interfaces.

Both Microsoft Windows and Google Android OS support Graphical User Interface (GUI), which means that instead of typing commands, different graphical objects such as icons are handled using a pointing device. The basic principle of different GUIs is very similar. Using your knowledge of how to use a Windows user interface we'll walk you through how to use Android and some other GUIs.



Figure 2.18: GUI considerations in application development

The following are some of the main characteristics of the UI and UX of Microsoft Windows and Google Android, which are some of the most used operating systems.

Microsoft Windows User interface

Windows uses dialog boxes that contain various visual elements that quickly show the user as much relevant information as possible, and with the use of the mouse and minimal keyboard typing, the user can select appropriately and launch the required applications/commands.

Google Android User interface

User interface design requirements for mobile devices differ significantly from those for desktop computers, as the small screen size and touch screen controls force special considerations when designing the user interface to ensure usability, readability and consistency.

In the mobile interface, icons could be used more extensively and controls may be automatically hidden when not used. The icons themselves need to be smaller, and there isn't enough room to display a label for everything on the screen which can cause some confusion to the user. Users must be able to understand each command's icon and its meaning either through readable text or an understandable graphical representation.

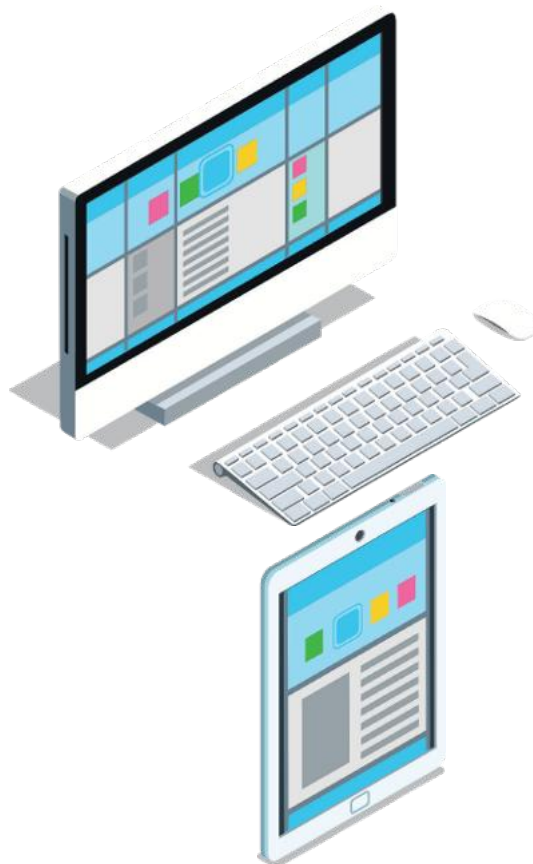


Figure 2.19: User interface on different device types

Exercises

1 Answer the following questions based on what you learned in this lesson.

1. What is meant by human-computer interaction (HCI)? Mention its components.

2. What is User Interface Design (UI)?

2 Briefly explain the difference between user experience and user interface.

UX	UI



3

Put a tick ✓ in front of the appropriate device type for each of the following descriptions:

Desktop
Computers

Portable
Devices

1. Low cost devices with high specifications.



2. Its screen size can be up to 30 inches.



3. Screen resolution is usually higher.



4. These devices are light in weight and fit in the pocket.



5. It is usually attached to a mouse and keyboard.



4

Compare the means by which desktop and mobile devices connect to the Internet.



5

Read the sentences and tick ✓ True or False.	True	False
1. HCI exclusively studies the business logic development of applications.	☐	☐
2. One of the most important similarities between the mobile and desktop experience is that people use them in the same way and for the same types of tasks.	☐	☐
3. HCI includes the scientific field of cognitive science.	☐	☐
4. The operating system has no effect on the speed of a mobile phone or desktop computer.	☐	☐
5. Mobile devices give you full access to all the hardware resources.	☐	☐
6. Mobile data networks are the cheapest means to connect to the Internet.	☐	☐
7. Microsoft Windows and Google Android use the same GUI components.	☐	☐
8. Mobile device users have the opportunity to search online while commuting or using public transportation.	☐	☐
9. Mobile devices are more commonly used by people in office environments than computers.	☐	☐
10. The difference in smartphone and desktop computer usage affects the types of websites and apps that work well on each device.	☐	☐

- 6 List some of the main UI and UX features of the Google Android and Microsoft Windows operating systems.

Google Android operating system	Microsoft Windows operating system

- 7 Explain the difference between the operating systems of desktop computers and mobile phones.



Lesson 3

Creating a Prototype

Link to digital lesson



www.ien.edu.sa

System Design

The system design stage comes directly after the analysis stage, during which the system elements, components, and system interfaces are identified, and this in turn includes planning for several things such as system architecture, hardware components, operating systems, programming, integration with other systems, and system security issues.

The Main Operations of the Design Phase

The processes in the design phase revolve around what the system (interfaces) and works (functions) will look like. Some parts of this phase focus on the technical features of the system while other parts focus on how the system responds and interacts with the user.

Other Operations Involved During the Design Phase

- Designing screen-based inputs, which includes designing how data enters the system, such as through text boxes, drop-down lists, forms, and so on.
- Designing User-Interface layouts, which includes how system menus, web pages, or applications will look. It is often useful to use organization charts to design this part.
- Designing system reports, which includes the process of designing system outputs such as usage reports, summaries, statistical data, invoices, or any type of printed report.
- Designing screen-based outputs, including screen outputs and system reports such as search results, error messages, or any type of report that appears only on the screen.
- Designing data structures to store data, including designing how data is stored in databases and tables.
- Designing rules for validating inputs and verifying data includes how to prevent incorrect or corrupt data from being entered into the system and how to validate it.



Now that you have gotten to know the concept of UI and UX and gotten acquainted with the operating systems of smartphones, you are ready to create an application for users with special needs. First you need to design the prototype of the application, and in this tutorial, you will create this model using the Pencil Project tool.

Prototype

A prototype is a model that simulates the product we want to create, and since designers want to fully define and understand how users will interact with the product to test it, they build a prototype, as it is not feasible to produce a final product and then have it tested by users.

Prototyping is designed to enable designers to think about their solutions differently, to reduce the cost of failure and to avoid investing time and money in an idea that may not be suitable.

The Importance of the Prototype

Prototyping helps focus on the basic functions of the application and gives an idea about the look and feel of the product to benefit the customer in making a right decision about the appropriateness of these elements.

Table 2.4: Why is the prototype important?	
Better understanding of design content.	Not only does prototyping provide a solid visualization of the design to understand the look and feel of the final product, but it also helps the team better understand why they are creating their design, what they are designing and for whom.
Facilitates the process of obtaining feedback.	Using prototypes, you can gather feedback from stakeholders at every stage of product development, whether adding new features or redesigning parts of the product, testing what works for them and what doesn't according to the specific objectives of the application in progress.
Validate modifications before development.	Prototyping allows for multiple discussions regarding changes to work before entering the final development phase. This process facilitates the adoption of appropriate changes and ensures that realistic requirements are built that meet the application objective.
Early changes save time and cost.	Early changes help you achieve your goals faster. Making adjustments in the final stages is expensive and may require radical restructuring and more thought and reformulation. Having a ready-made prototype enables us to make needed changes early before investing a lot of time and effort into creating the final product.

Prototyping Categories

There are different approaches to modeling, and you should always select the right one that works with your product and the resources available for the work.

Prototyping methods are generally classified based on their accuracy into the low-fidelity category, the medium-fidelity category, and the high-fidelity category.

Low-Fidelity Prototype

- > This model is usually created using paper in the initial stages of design and is constantly refined throughout the process.
- > It helps to make changes easily and quickly, as it focuses more on how the system is used rather than what it looks like.
- > As the product becomes more complex it is difficult to keep the low fidelity model in the development cycle, making paper prototypes ineffective in keeping up with the required depth of design.



Figure 2.20: Low-Fidelity prototypes

Medium-Fidelity Prototype

- > A model that is created to simulate and represent system functions, even limited ones, based on specific usage scenarios.
- > This model is best for the intermediate stages of product development when moving from a low fidelity prototype to a medium fidelity prototype.



Figure 2.21: Medium-Fidelity prototypes

High-Fidelity Prototype

- > This model is often confused with the final product because of the similarity between them in appearance, and the effectiveness of some system functions in the model. High-resolution models are the best in giving a realistic experience similar to the product with actual functions.
- > It is characterized by accuracy in estimating the required cost and time.
- > Supports the analysis of more complex parts of the product in advanced stages, as showing this model in the initial stages of modeling may confuse the stakeholders and not provide the necessary preliminary knowledge.



Figure 2.22: High-fidelity prototypes

Modeling Instructions

Proper prototyping is important to validate design solutions for our project, so let's go over some tips to keep in mind when working with prototyping:

- Invest time in creating the template and don't go into too much detail.
- Always remember product goals while working
- Consider the user first.

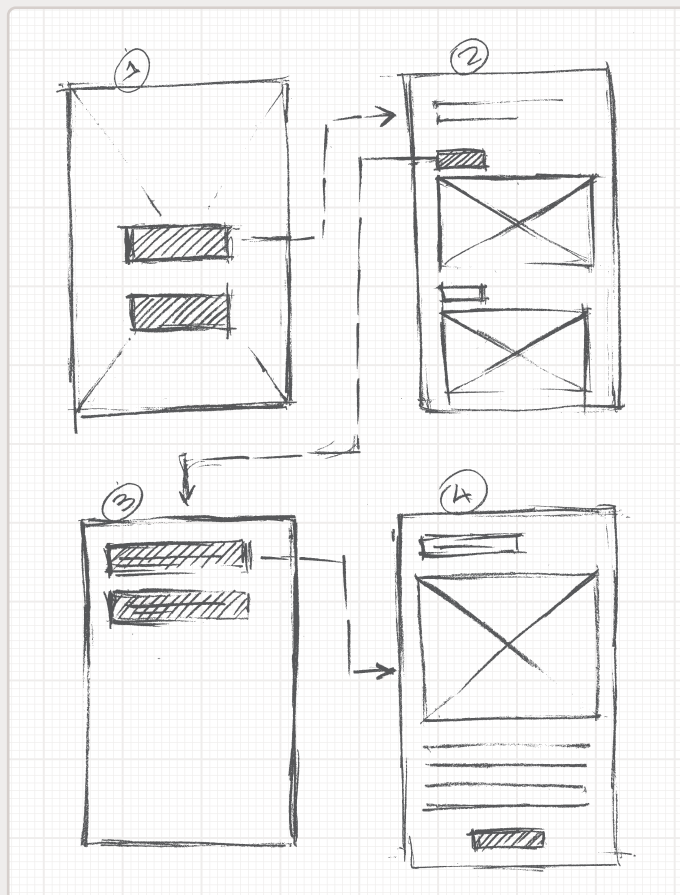
Application Scenario

Not all users have the same needs, that's why applications must take into account these differences and modify their user interface and functionality.

We are going to create an application to help tourists navigate through the screen so that they can read information about the different tourist spots that they can visit in the cities of Riyadh and Jeddah.

The Low-fidelity Prototype for our accessible tourism application would look like this:

- 1** The first screen of the application consists of an image and two buttons so that the user can press the first button that connects the user to the next screen and the second button to change the language from English to Arabic.
- 2** The second screen of the app consists of two images so that the user can choose the city he wants. The images also work as a button that connects the user with the next screen.
- 3** The third screen displays a list with two highlights of each city, each one works as a button to move on to the next screen.
- 4** The last screen displays an image and a simple description about the highlight.

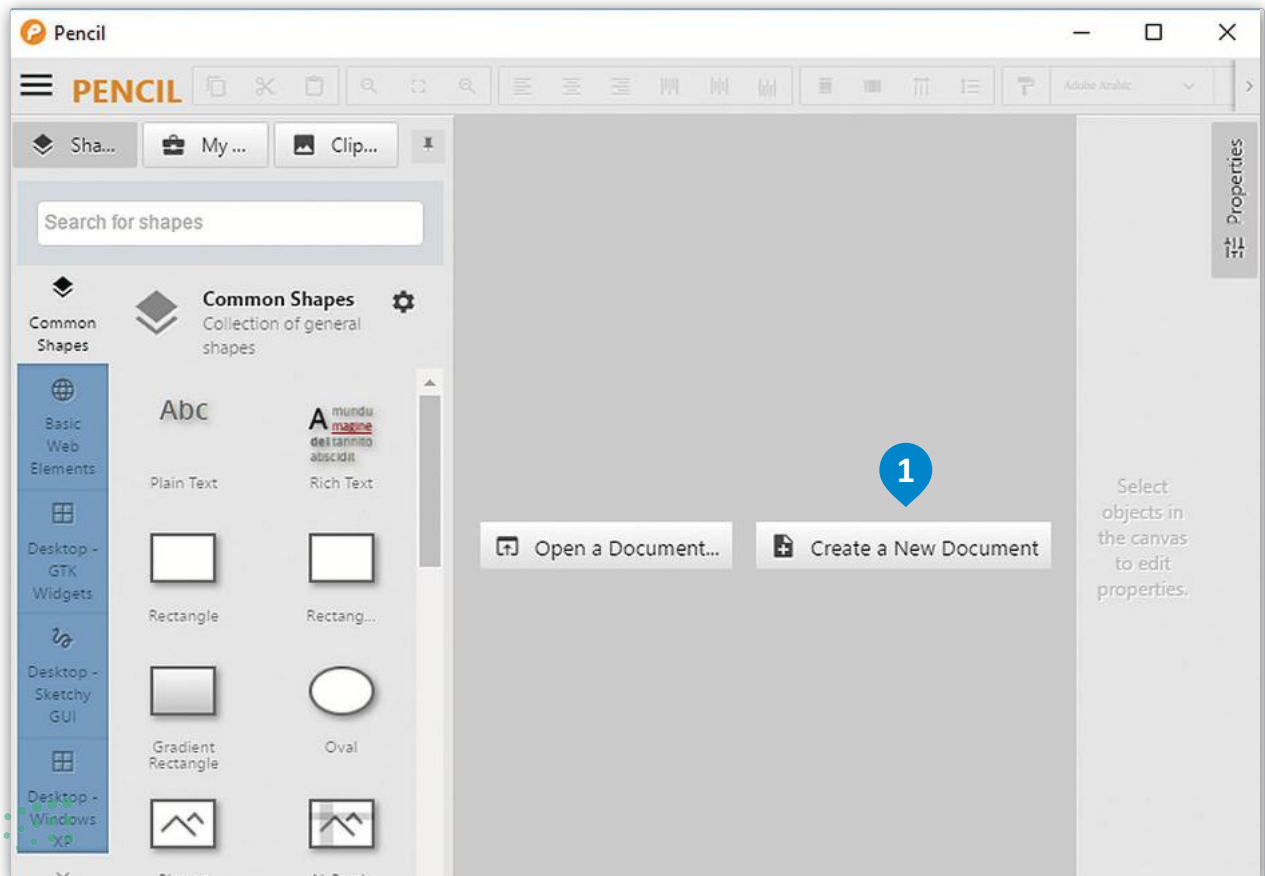


Creating the Prototype with the Pencil Project Software

Pencil Project provides an open source graphical user interface (GUI) for prototyping for all platforms. In the previous lesson, we used Pencil Project to create a flowchart. In this lesson, we will create a Medium fidelity Prototype for a mobile application.

To create a new prototype:

- > Open **Pencil Project** and click **Create a New Document**. **1**
- > From the **Shapes** panel, click **Mobile - Android ICS** to add a shape. **2**
- > Drag and drop the **Phone** shape to the canvas. **3**
- > Drag and drop the **Status Bar** shape to the top of the phone screen as it looks in a real phone. **4**
- > From the **Shapes** panel, click the **Common Shapes** section to add the shape. **5**
- > Drag and drop a **Bitmap Image** onto the canvas to load an image. **6**



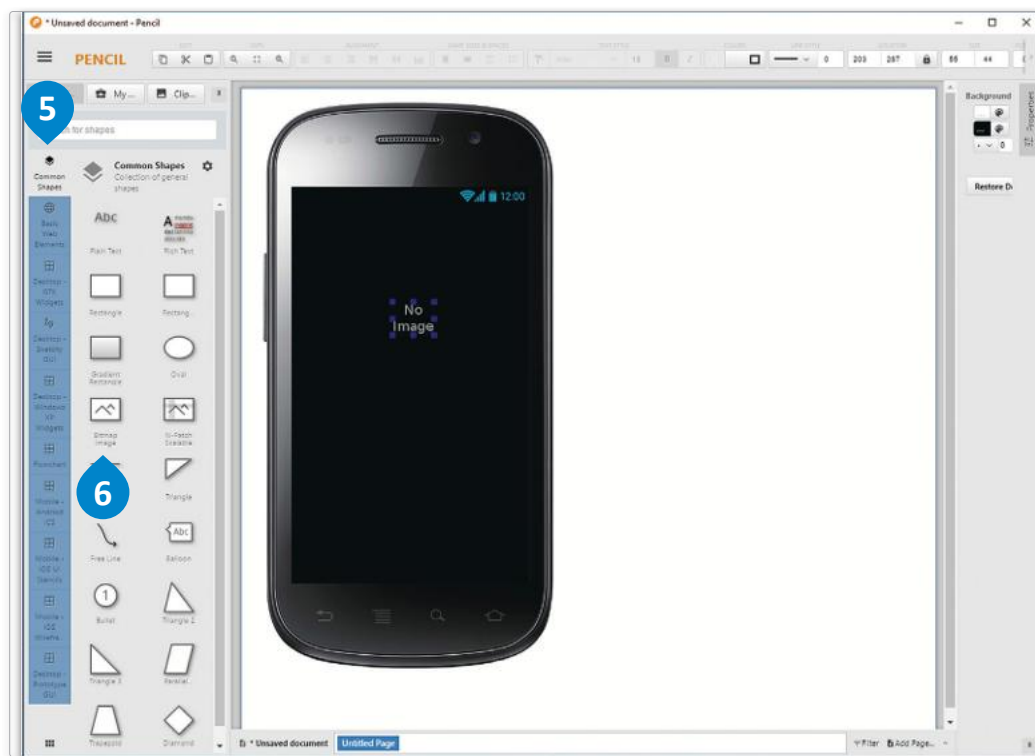
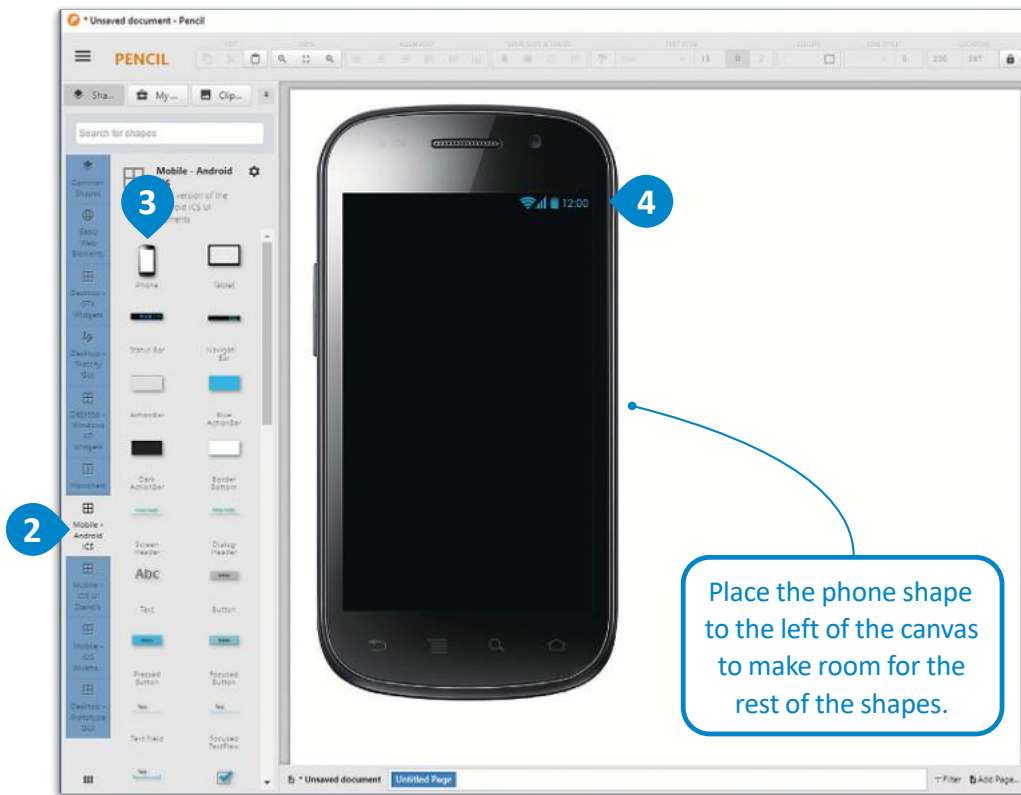


Figure 2.23: Creating a new prototype with Pencil



To insert an image:

- > Right click on the bitmap shape and select **Action** then **Load Embedded Image**. ①
- > Choose the image file. ②
- > Click **Open** to upload the image from your device. ③
- > Drag and drop the image to the middle of the image outline to fit the phone screen. ④

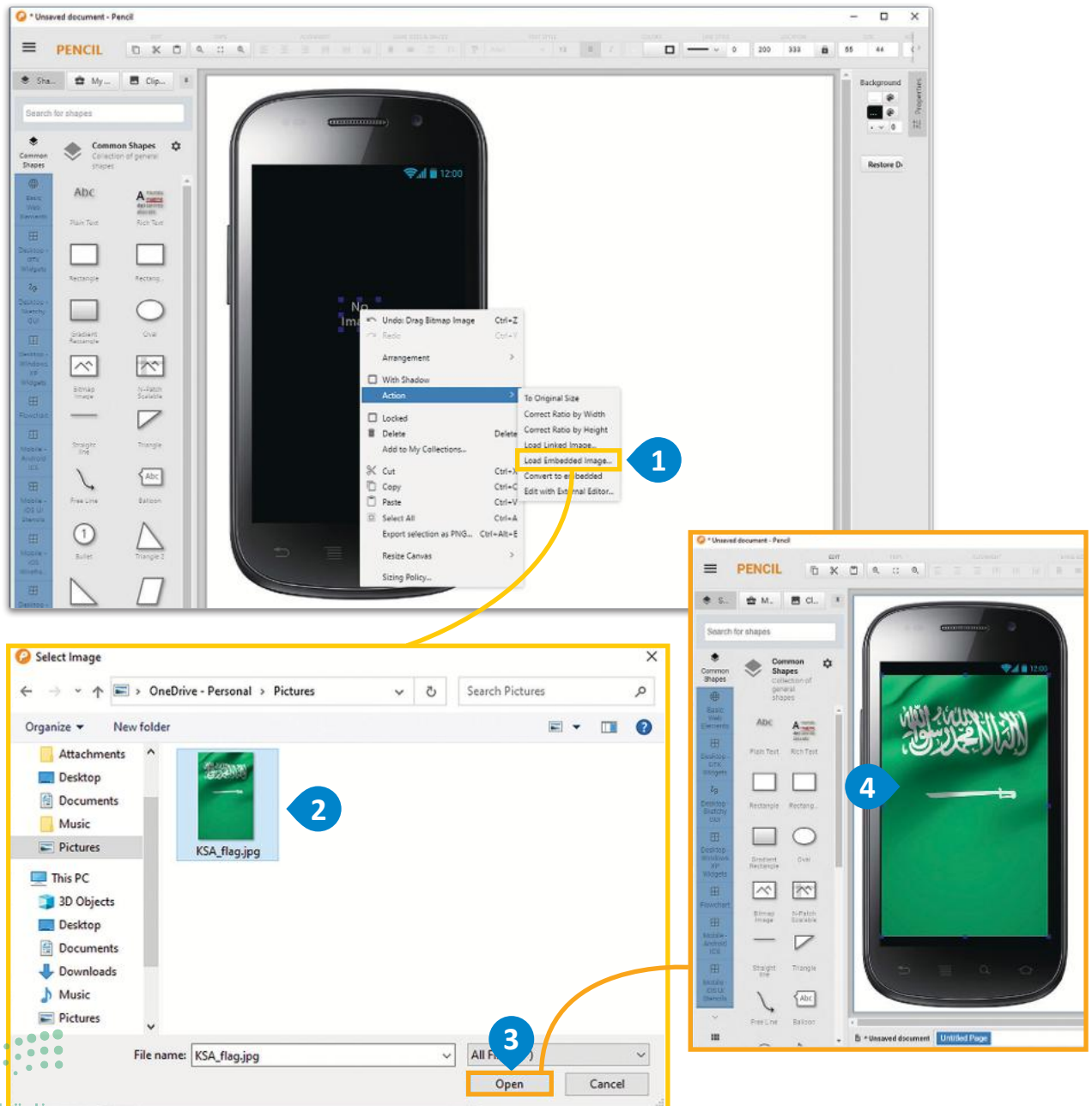


Figure 2.24: Inserting an image in the prototype

To add a button:

- > From the **Mobile - Android ICS** section, drag and drop the **Focused Button** shape to the middle of the screen. ①
- > Double tap and label the button **"Discover"**. ②

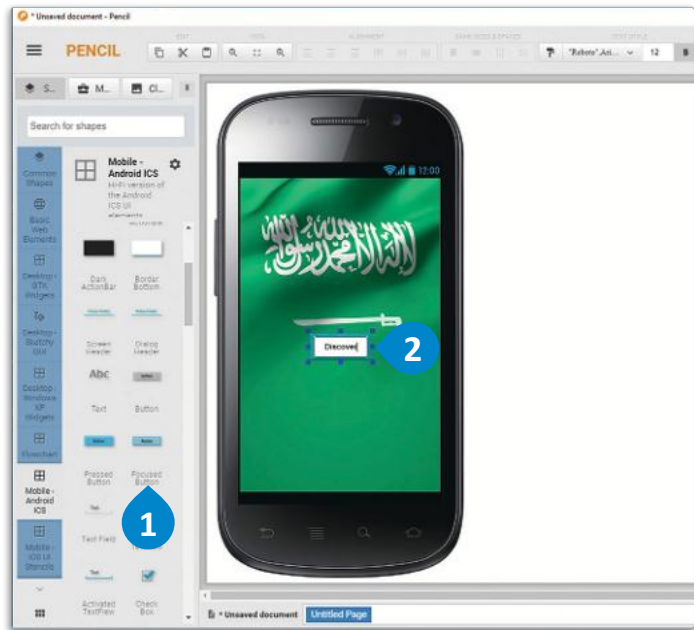


Figure 2.25: Adding an English button

Repeat the steps you followed when adding the button to add the second button with the label "Arabic".

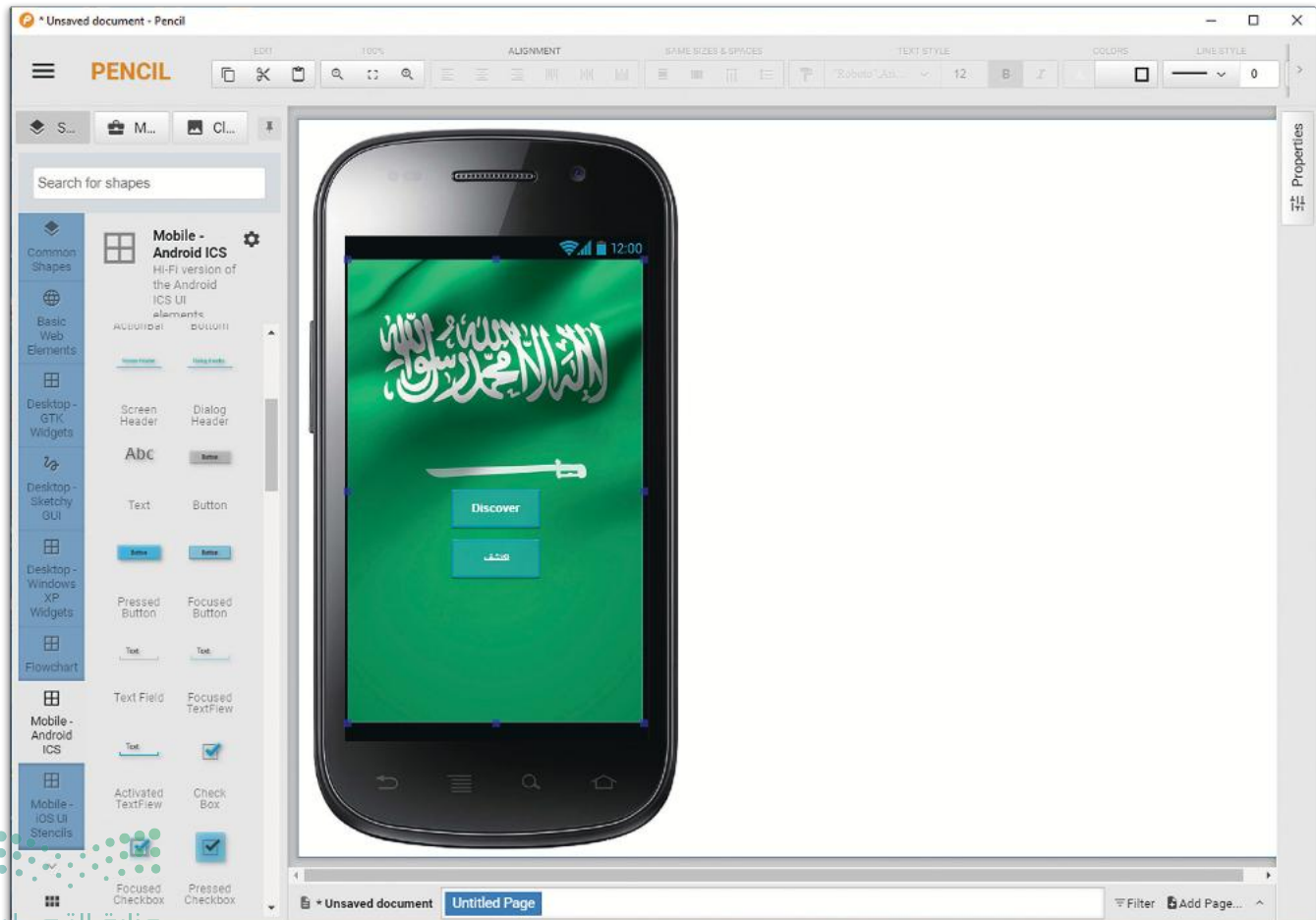


Figure 2.26: Adding an Arabic button

Multi-Page Application

The user cannot interact with the Pencil Project prototype, so the multiple app screens must appear side by side in the same order they would appear while using the actual app.

Create a Multi-Page Application

A Multi-Page Application (MPA) consists of several pages with static information (images, text, etc.) and links (text, button, image, etc.) as well as other pages.

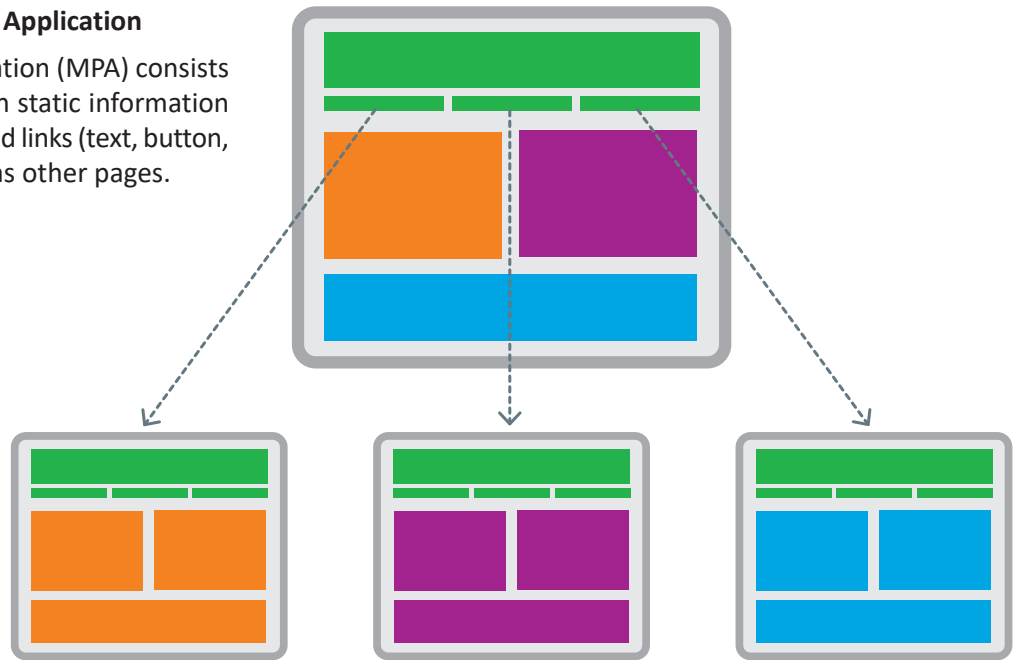


Figure 2.27: Multi-page application

To create the app's second screen:

- > From the **Mobile – Android ICS** section, drag and drop the **Phone** shape to the right of the first **Phone** shape. ①
- > Drag and drop the **Status Bar** shape to the top of the phone screen as it looks in a real phone. ②
- > Below the status bar, add the **Screen Header** shape, ③ then double-click and type in the title "**Discover Saudi Arabia**". ④
- > Change the text size to **11** from the **Font Editor** bar. ⑤

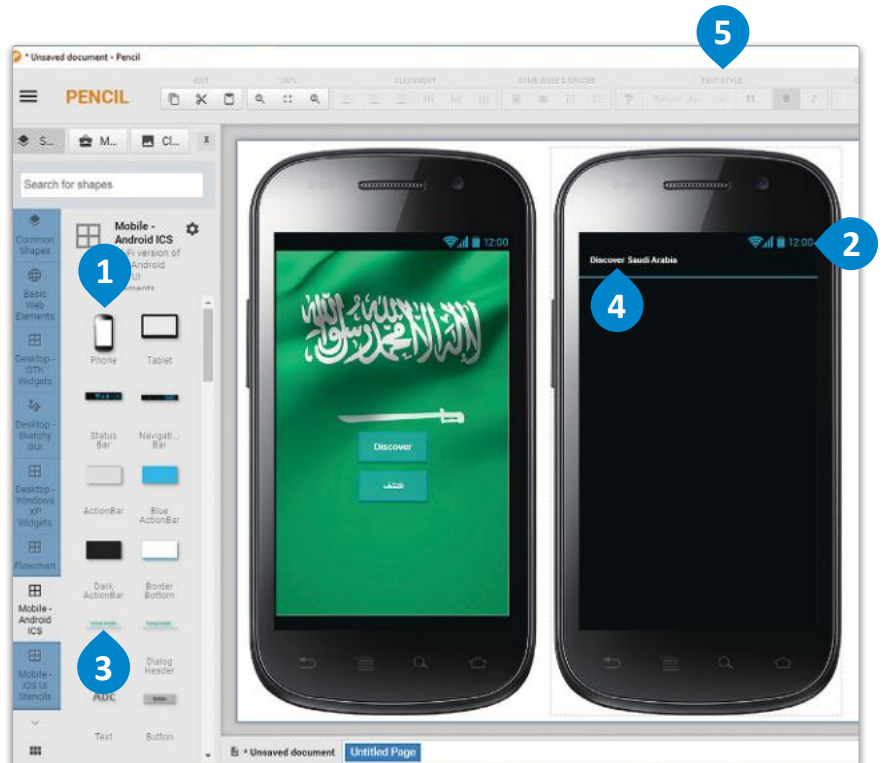


Figure 2.28: Creating a second screen

Repeat the steps you followed when adding the image and the Screen Header to make the screen look as shown on the right.

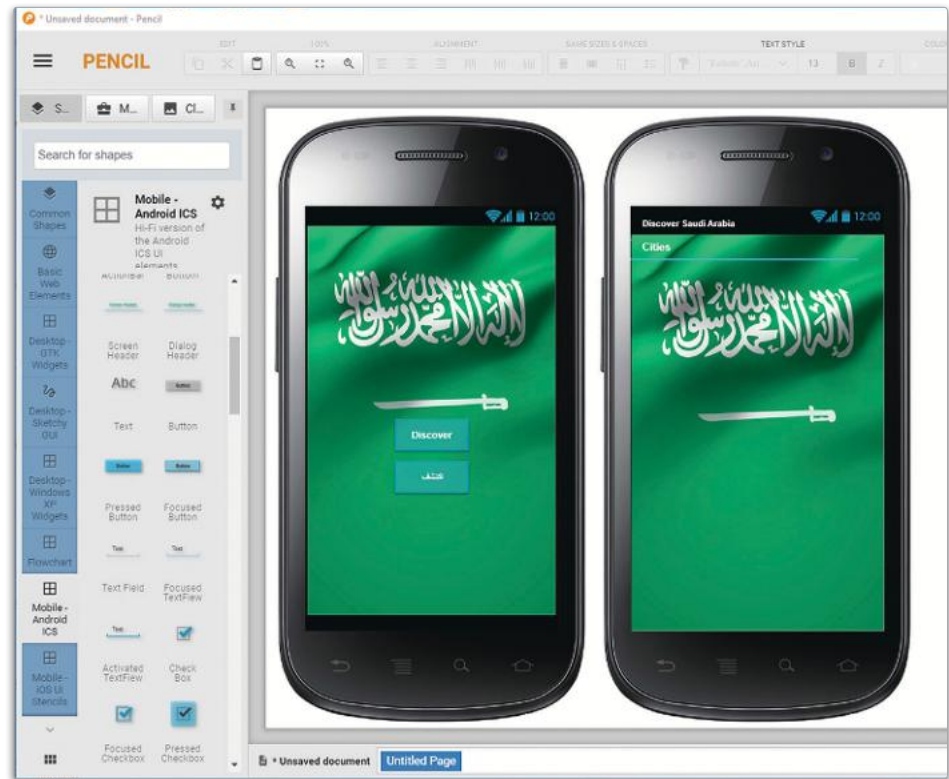


Figure 2.29: Setting up the second screen

Then you have to add two images displaying the cities of Riyadh and Jeddah as shown.

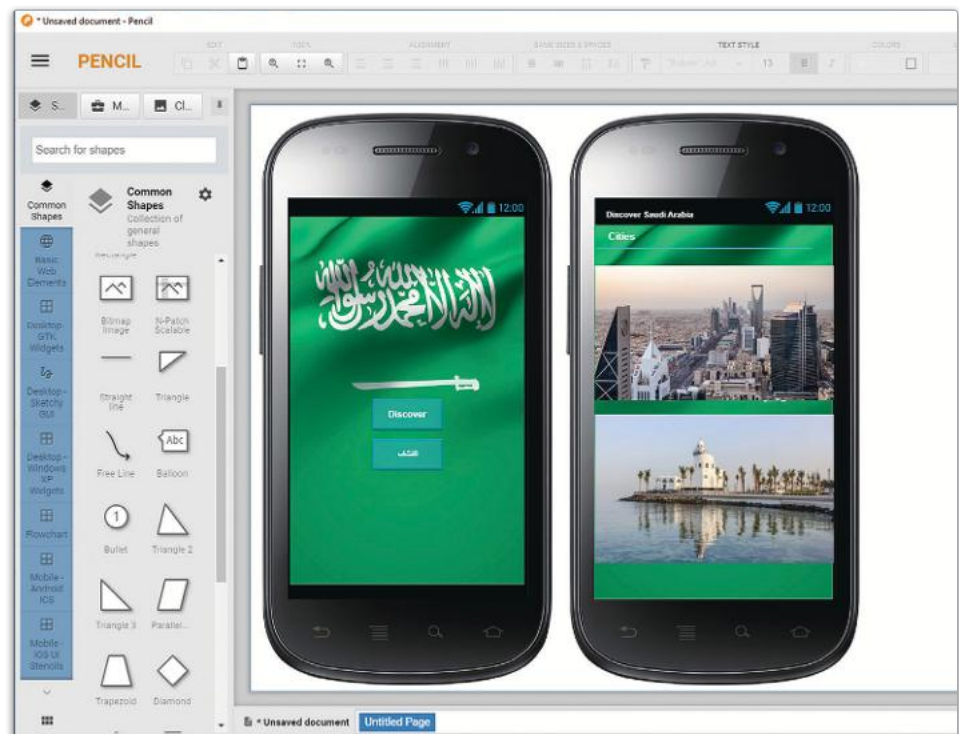
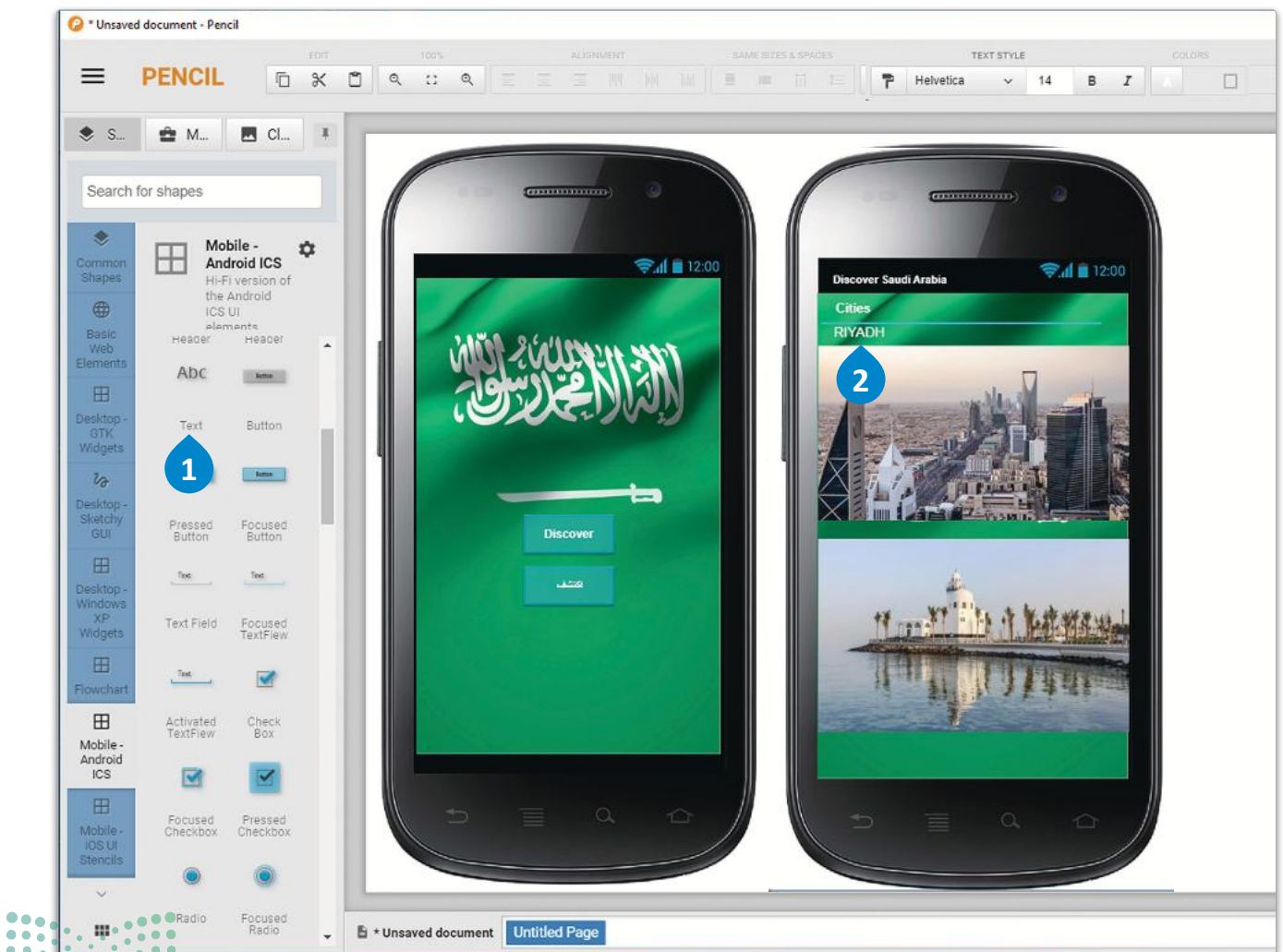


Figure 2.30: Adding images of cities



To insert a text shape:

- > From the **Mobile – Android ICS** section, drag and drop the **Text** shape. **1**
- > Then double-click and type " **RIYADH**". **2**
- > Change the text size to **22** from the **Font Editor** bar. **3**
- > Click on **Color Palette** **4** and choose the color with code: **#FFFFFF**. **5**
- > Repeat the steps to add a label for the second image. **6**



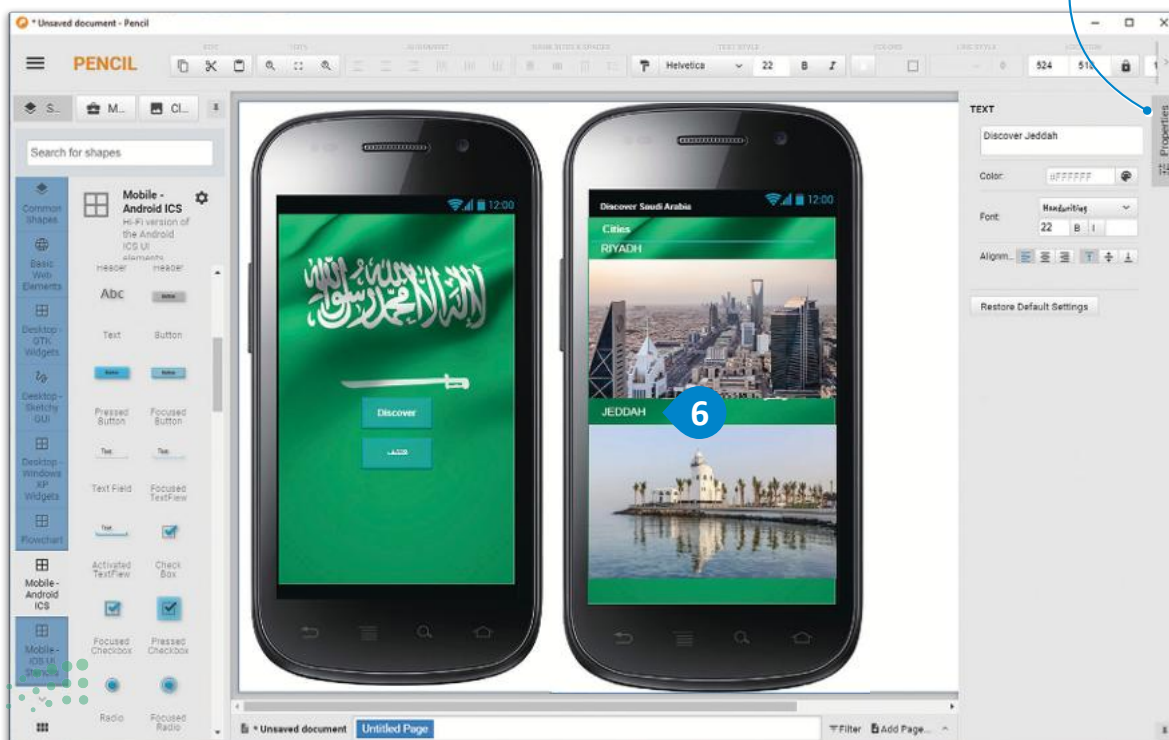
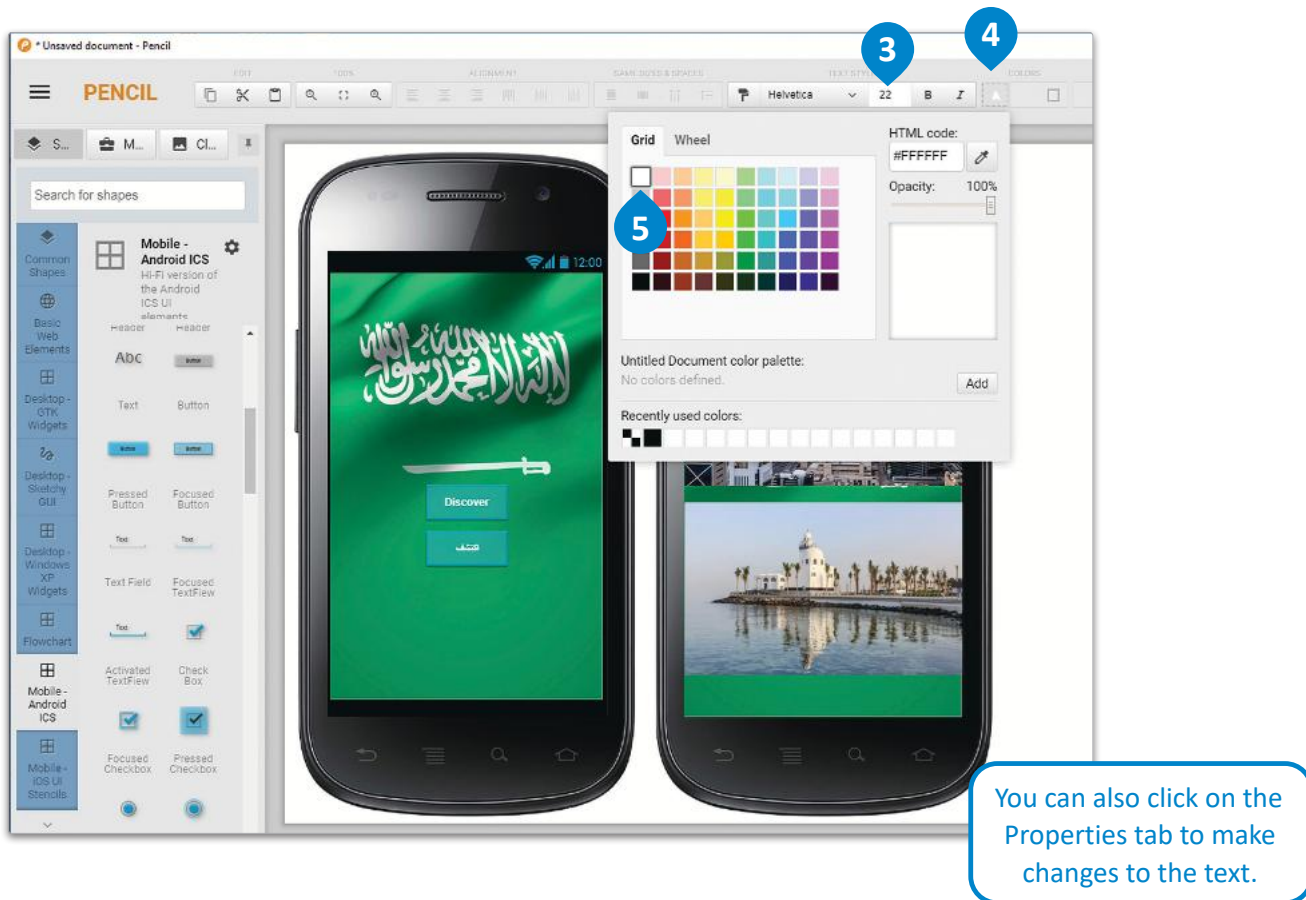


Figure 2.31: Setting up text label for the second image

To create the app's third screen:

- > From the **Mobile-Android ICS** section, drag and drop the **Phone** shape to the right of the second **Phone** shape. **1**
- > Drag and drop the **Status Bar** shape to the top of the phone screen as it looks in a real phone. **2**
- > Then drag and drop the **Text** shape **3** and then from the **Properties** tab change the color of the text and write "**Al Masmak**". **4**
- > Repeat the two last steps to write "**Boulevard of Riyadh City**". **5**

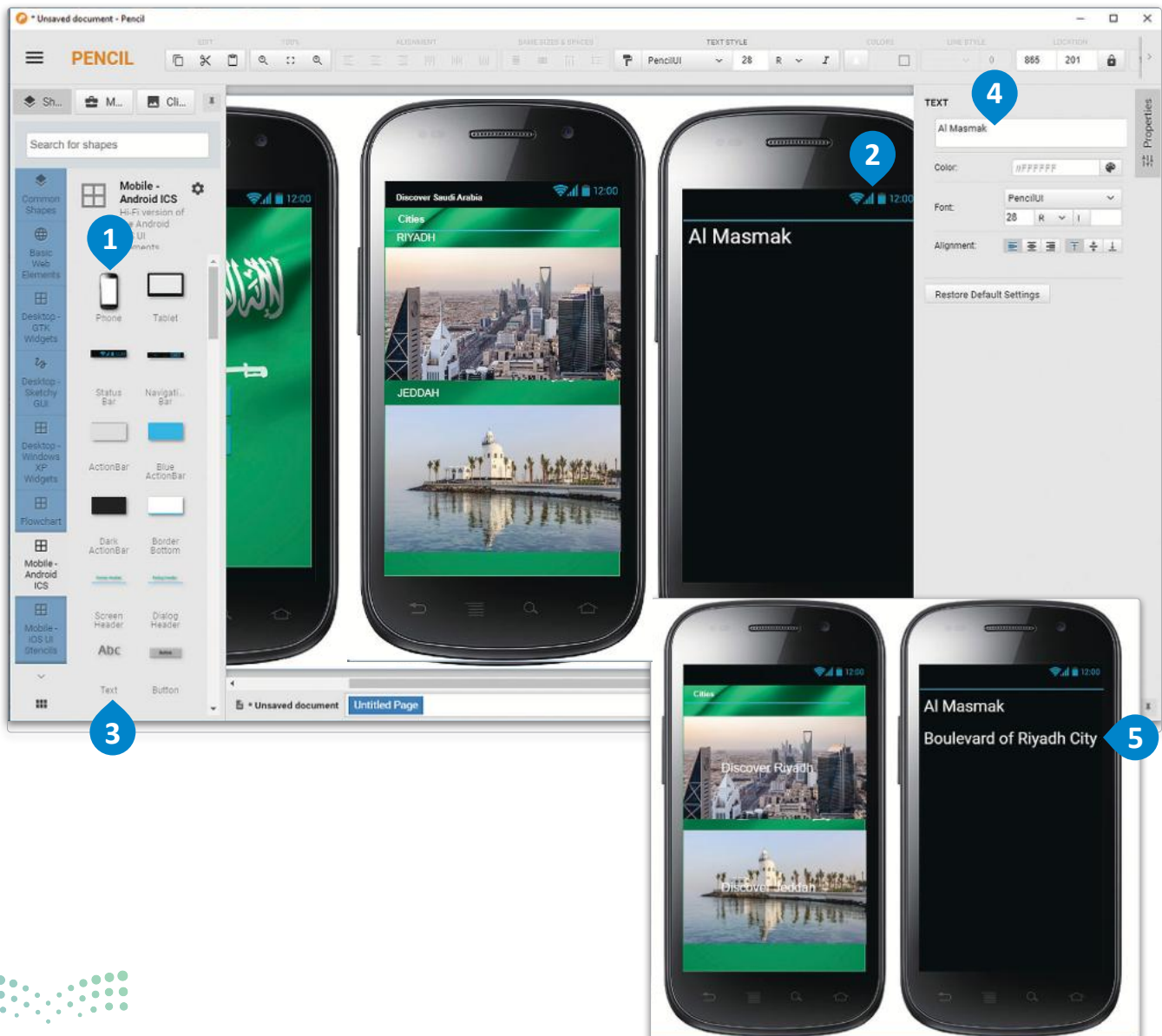


Figure 2.32: Creating a third screen

Finally, create the last screen, which displays an image and a brief description of Al Masmak.

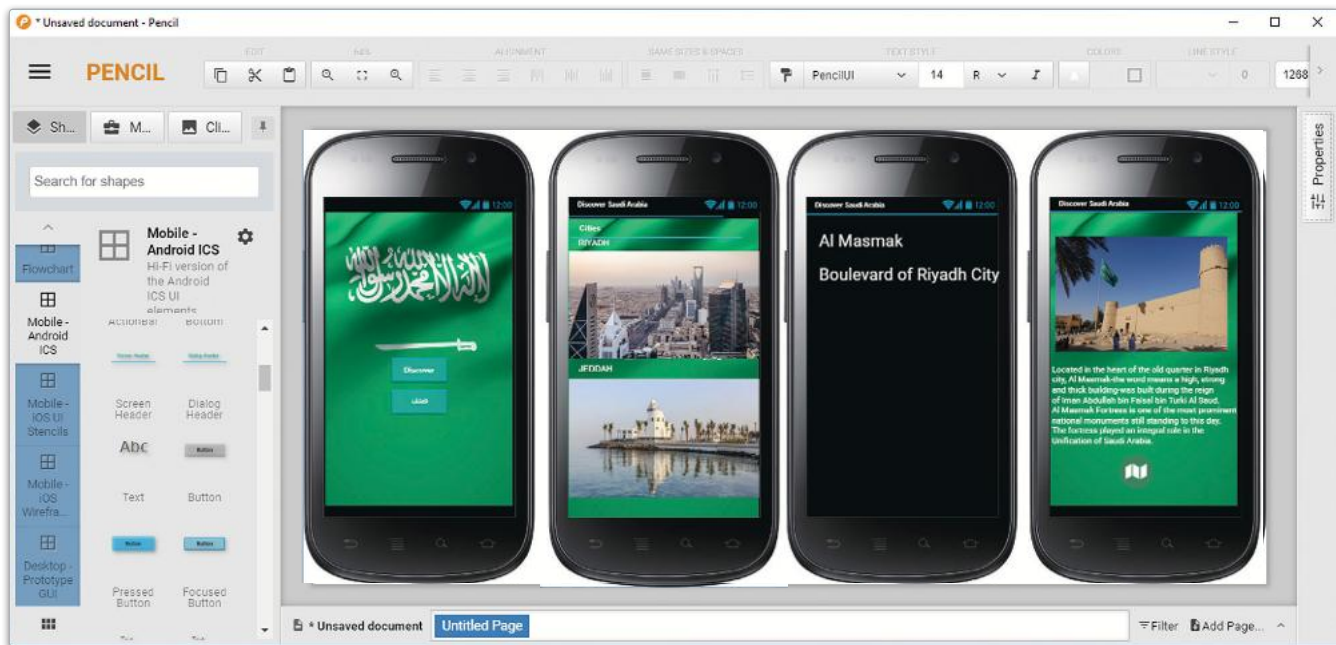


Figure 2.33: Creating the final screen

Do not forget to save your project when finished.

The Role of Users in Prototyping

After the prototyping creation process is complete, it is important that it be viewed by users. To facilitate the prototyping process, the system analyst must clearly communicate the purpose of the prototype to users, making it clear that prototypes are only valuable when meaningfully engaged with users.

Best Strategies for Getting Feedback on Prototyping:

- Find multiple, alternative ways to get feedback from users such as interviewing them to get them to talk about their thought process while using the prototype.
- Test your prototypes on the right people. If you're in the early stages of your design project and want to get some simple or advanced feedback, testing your prototypes on your teammates should suffice.
- Be sure of what you are going to test and ask the appropriate questions.
- Be neutral when presenting your ideas, avoid being biased towards your idea, and try to recognize the error when there are negative reactions.
- You can improvise from the original test scenario in order to adapt to the test environment, so as to get the best feedback from the users.
- Let the user contribute ideas based on your prototype and provide helpful criticism that will improve the application.

After obtaining feedback from users, the system analyst should modify the home screen designs according to users' feedback on the prototype.

Exercises

1 Match the types of prototypes with the appropriate descriptions.

High-Fidelity
Prototype

1

Low-Fidelity
Prototype

2

Medium-Fidelity
Prototype

3



It is used in the intermediate stages of product development.



It is designed to represent the functionality of the system and focuses more on it than appearance.



Closest prototype to what the actual final product looks like.



It can be expensive and time consuming.



Changes can be made to it easily and quickly.



It can be created with paper.



2 What are the benefits of prototyping?

[illegible]

3 What are the tips to follow when preparing a prototype?



4

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

5

- 



Project

1

You will continue the tourism mobile application that provides information about Saudi Vision 2030 that you started in the previous unit.

2

Use the Pencil Project program to create a tree diagram to represent how the components of the application you will build are organized, the pages it will contain and the components of each page.

3

Then, you have to create a Low-fidelity Prototype for your application. Use paper and pencil to draw the screens of your app.

4

In the next stage, use the Pencil Project to create a Medium-fidelity Prototype for the mobile application.

Finally, create a presentation to illustrate this project.



Wrap up

Now you have learned to:

- > Distinguish the diagrams in the Analysis stage.
- > Create a workflow diagram using Pencil Project.
- > Design a prototype with Pencil Project.

KEY TERMS

Diagram	Medium-Fidelity Prototype	Tree diagram
Flowchart		Use case diagram
Functional requirement	Multi-Page Application (MPA)	User Experience (UX) Design
High-Fidelity Prototype	Non-Functional requirement	User Interface (UI) Design
Human-Computer Interaction (HCI)	Process	Wireframe Diagram
Low-Fidelity Prototype	Prototyping	Workflow Diagram



3. Developing Applications with App Inventor

In this unit, you will use MIT App Inventor to develop a functional and interactive mobile application for a tourism campaign. You will use the prototype you created in the previous unit to design the UI and then you will program it.

Learning Objectives

In this unit, you will learn to:

- > Utilize a wireframe prototype as a guide to build a UI.
- > Design a functional UI.
- > Utilize prototype feedback to improve an application.
- > Develop an application from specifications.
- > Enrich a mobile application with content.
- > Enhance mobile application with interactive UX components.

Tools

> MIT App Inventor



Lesson 1

Introduction to MIT App Inventor

Link to digital lesson



www.iem.edu.sa

Developing Mobile Applications

The process of designing and developing a mobile application is similar to the process of developing a web or desktop application.

Table 3.1: Examples of smartphone applications

Email applications
Social media applications
Communication apps such as instant messaging applications
Map applications
Administrative applications for governmental entities such as ministries, hospitals or schools
Mobile games

Mobile Application

A mobile application is a type of application software that is designed to run on mobile devices such as smartphones and tablets.

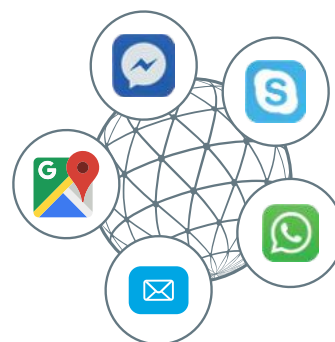


Figure 3.1: Types of mobile applications

Stages of Creating Smartphone Applications



Analysis and Design:

Select the idea, the goal of the application, and the target group. Create a manual app diagram of the different user interfaces and how they relate to one another.



Development:

Use a smartphone application development program to implement the design that you devised in the previous stage.



Testing:

Test the application and address any programming or design problems that may arise, then add the final touches to your work.



Publishing and Marketing:

Upload the application, so it can get approved and published to the application store.



MIT App Inventor

MIT App Inventor is a development tool for smartphone apps. It allows apps to be created without having to write code by using a building block environment similar to the Scratch application. Traditional mobile development uses coding with mobile-native programming languages like **Java**, **Kotlin** or **Swift**. App Inventor can also package your application in a form ready for distribution.

Advantages of Using MIT App Inventor



Shorter development time, as we can develop an application in less than one hour.



Helps develop creativity skills through the use of software building blocks and reduces the chances of making programming mistakes.



Ease of sharing the applications that are created in this program.



Access to many basic functions in the phone, such as phone calls, SMS messages, location sensors, audio and video, among others.



Ability to save data via cloud storage platforms.

Developing a Tourism Application

You will develop an application for tourists visiting the KSA that allows them to look for the most popular destinations. When they select a destination, they will be shown a list of highlights for that destination. They will then choose a highlight and be presented with a photo and a description of that highlight. This is the application that you developed a wireframe prototype for in the previous unit. In this lesson you will design all the screens for the application, and in the next lesson you will program them.

Differences between Prototyping UI and Developing UI

When designing wireframes with a prototyping tool, the appearance of the elements and the components on a screen is arranged with declarative tools. This means that the arrangement of components such as buttons, labels and images can be explicitly specified. When developing the actual application, components are arranged dynamically according to the tool used. MIT App Inventor, like most development platforms, uses container components that are used to arrange and align other components that are placed inside them. Keep in mind that the methods of creating a UI are different when prototyping and when developing an application.

INFORMATION

App Inventor was developed by Google in 2010 and is now maintained by MIT (Massachusetts Institute of Technology).



Transitioning from Prototype to Application

As you have already created a wireframe prototype, you know how to design the UI of the application. This means that the development time will be shorter because the UI and UX decisions have been made already.

You only need to use the tools provided by App Inventor to replicate the wireframe's appearance as closely as possible. However, the original wireframe prototype does not represent the final view of the application. UI changes and new features are implemented during the development of the application, because feedback is given by testing users.

While you are preparing to develop the application, users testing give feedback on the wireframe prototype that you created in the previous unit. Their comments can be used to iterate the prototype again, or they can be implemented during the development phase. In this case, you will implement them directly when developing the application in App Inventor.

The main points from the prototype feedback are the following:

- The components of the Cities and Highlight screens need to have a container with a slightly different background color to the image of the flag.
- The Highlight screen needs to have a look consistent with the Cities page.
- It would be helpful if there was a way to view the location of each highlight presented.

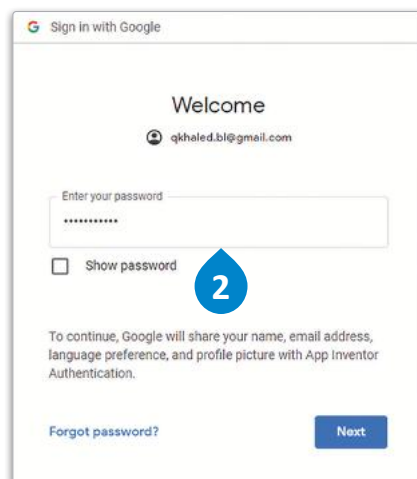
You will now begin to develop the application UI with the prototype as a guide, and you will also implement the comments from the given feedback.

Starting Building Apps with App Inventor

To start building apps with MIT App Inventor, you need to log into the App Inventor website with your Google account.

To start MIT App Inventor:

- > Go to **appinventor.mit.edu** and click **Create Apps!**. **1**
- > Sign in with your **Google** account. **2**
- > You are now viewing the **MIT App Inventor** workspace. **3**



3

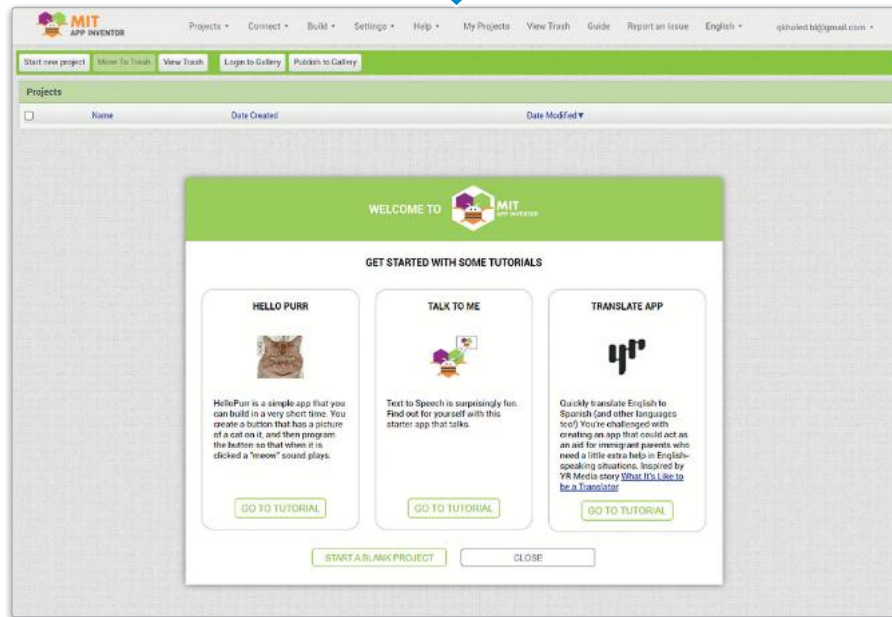


Figure 3.2: Signing in to App Inventor

To start a new project in MIT App Inventor:

- > Click on **Start new project.** ①
- > Type a name for your project ② and click **OK.** ③

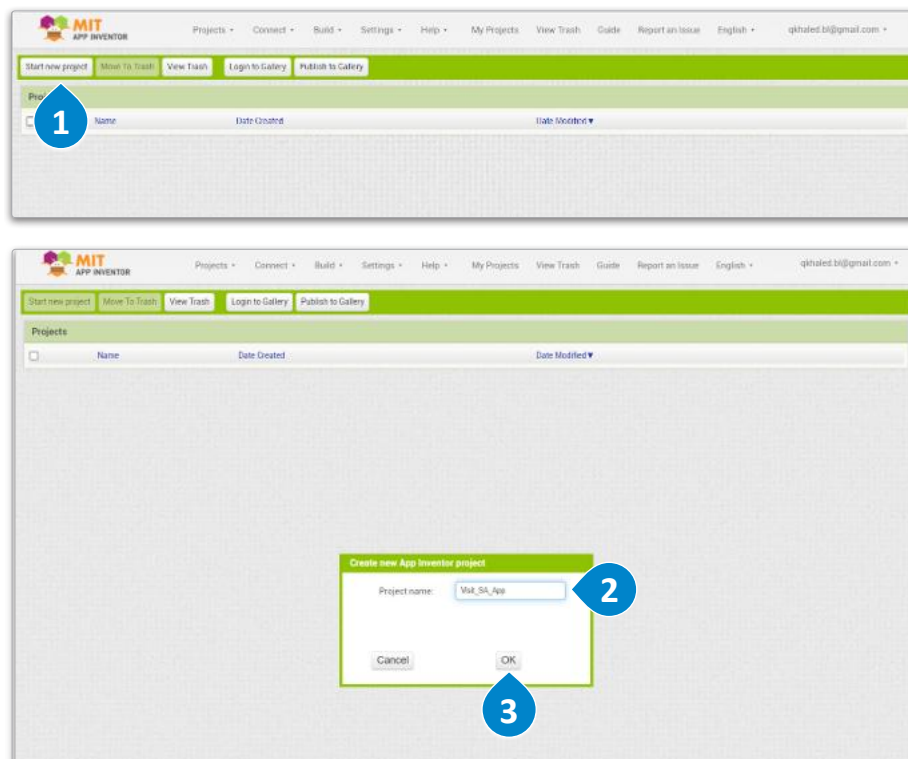


Figure 3.3: Start a new project in App Inventor



The App Inventor Interface

The App Inventor interface is split into two pages. These are the **Designer** and **Blocks** pages which you can access through the two buttons (Design & Blocks) at the top right of the screen. The **Designer** page is where you insert components into the screen and change their basic properties. The **Blocks** page is where you program those components.

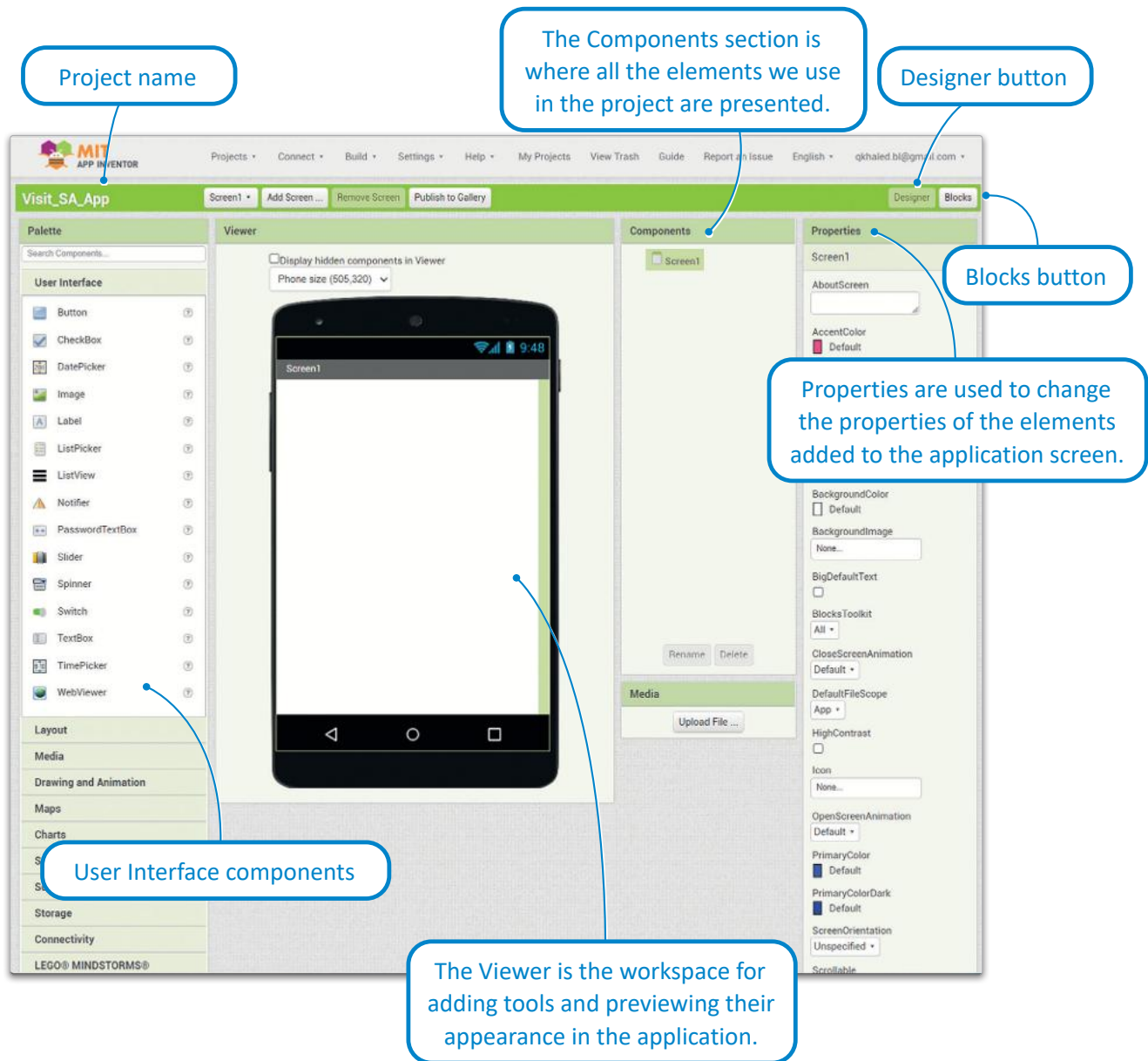


Figure 3.4: The App Inventor Designer interface



Table 3.2: App Inventor - most commonly used components

Component	Icon	Description
Button	 Button	A command button to perform a specific task when pressed.
Image	 Image	A special component that displays images.
Label	 Label	Displays text to be customized in the Text field in the Properties panel.
ListPicker	 ListPicker	It is a button that when pressed, displays several text options to choose from.

Changing the Properties of a Component

You will make the name **Home** the title of the first screen, **Screen1**, which will be the main screen of the application. In the viewer, you will change the **Title** of the screen to **Home**, as illustrated in the figure below.

To change the screen title:

- > Select **Screen1**, from the **Components** section. ¹
- > Scroll the sidebar down in the **Properties** section ² and in the **Title** field, write the word **Home**. ³

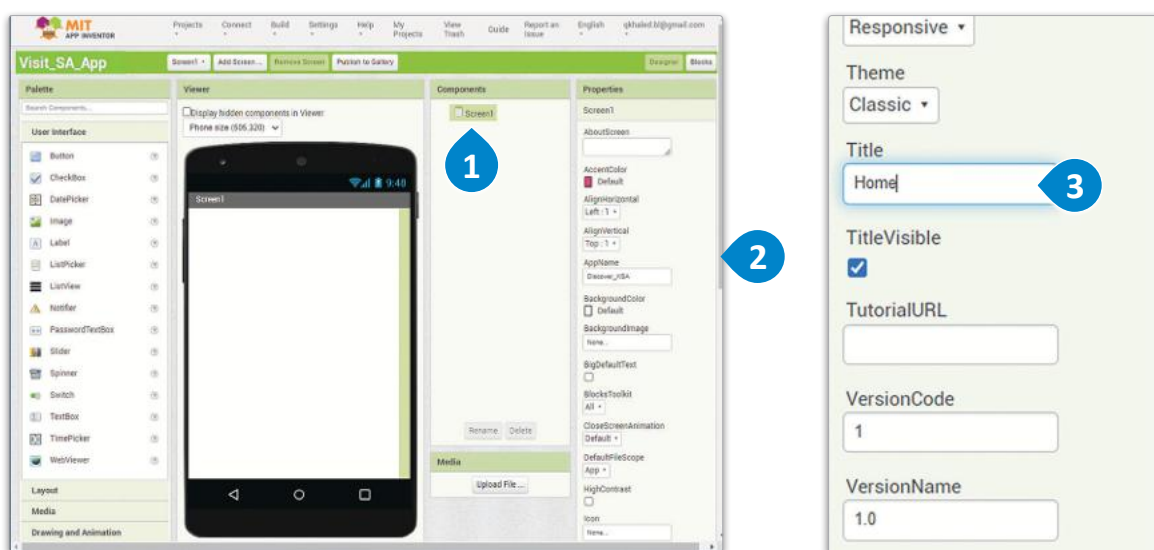


Figure 3.5: Changing the screen title



Adding a Button to the Screen

You will now create a command button called **Visit KSA**. When you click on this button, the app will open a new screen and offer destinations to visit in the KSA.

To add a button component:

- > Drag and drop the **Button** from the **User Interface** panel to the display. **1**
- > Click **Rename**. **2**
- > Type **Visit_KSA** **3** and press **OK**. **4**
- > Scroll the sidebar down in the **Properties** section, click **Text** and type "**Visit Saudi Arabia**". **5**

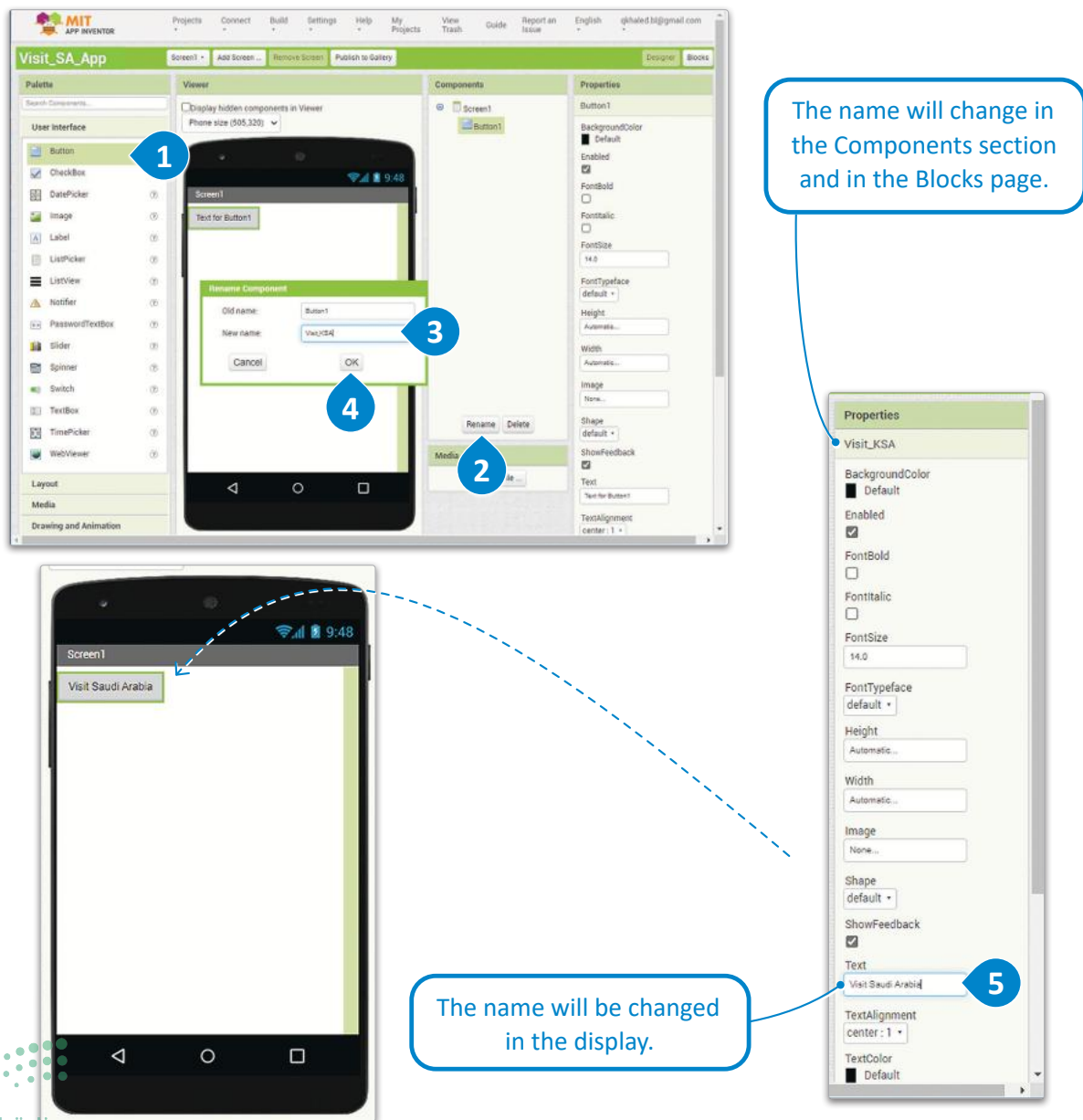


Figure 3.6: Adding a button component

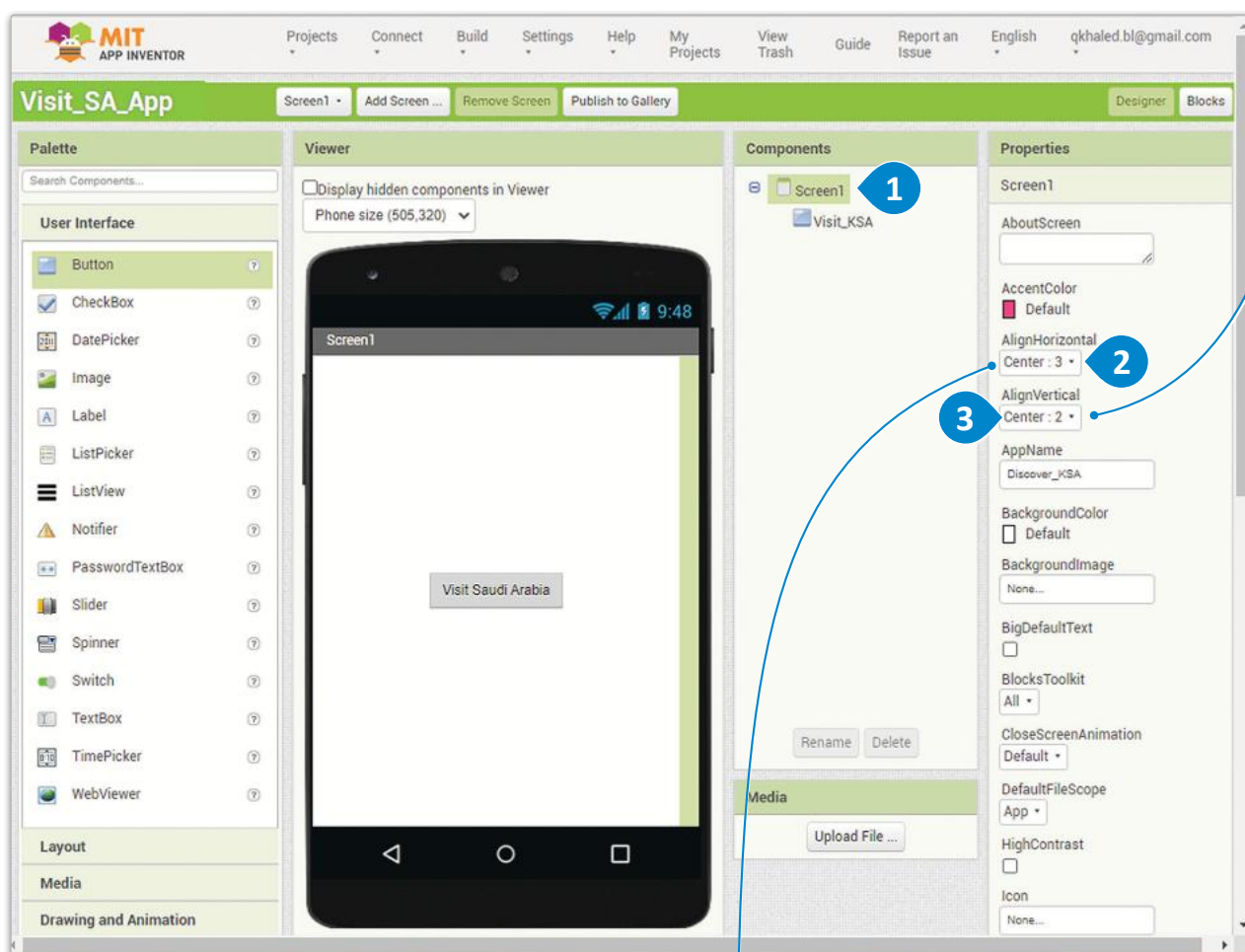
Aligning the Components of the Screen

Your application buttons should be located in the middle of the main screen.

To put the button in the center of the screen:

- > Select **Screen1**, from the **Components** section. **1**
- > From the **Properties** section, select **AlignHorizontal** to **Center : 3**. **2**
- > Then select **AlignVertical** to **Center : 2**. **3**

The number 2 is the number assigned to this setting of the vertical alignment tool.



The number 3 is the number assigned to this setting of the horizontal alignment tool.



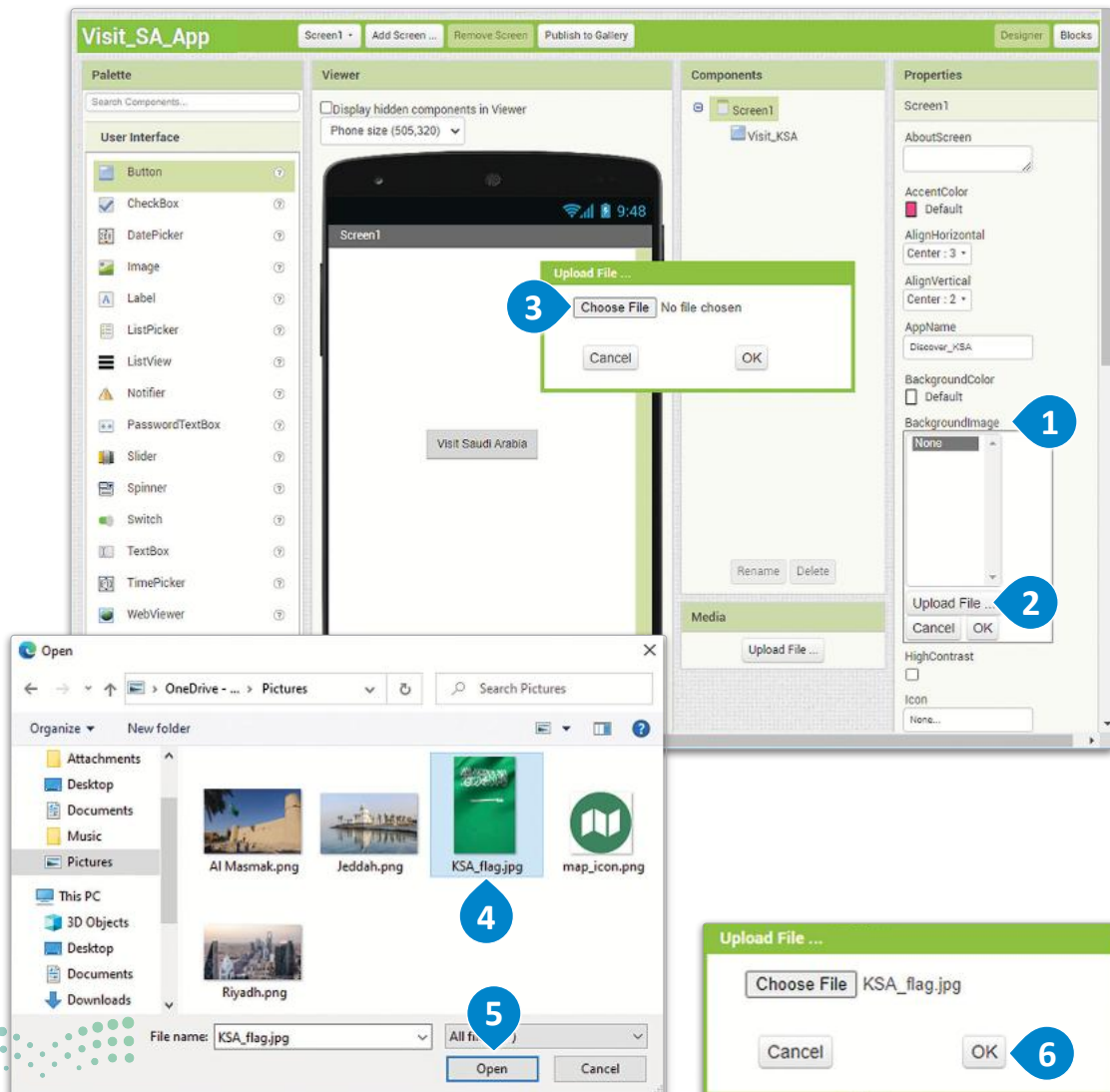
Figure 3.7: Putting the button in the center of the screen

Adding a Background Image to the Screen

You will make some improvements to the app by adding a background image.

To add a background image:

- > From the **Properties** section of **Screen1**, select the **BackgroundImage** property. **1**
- > Click **Upload File** to upload the image from your computer. **2**
- > Click **Choose File** to choose an image from your computer. **3**
- > An **Open** window will appear. Choose the image you want to add from your computer, **4** then click **Open**. **5**
- > Then click **OK**. **6**
- > Scroll the sidebar down in the **Properties** section of **Screen1**, **7** untick the **TitleVisible** property. **8**



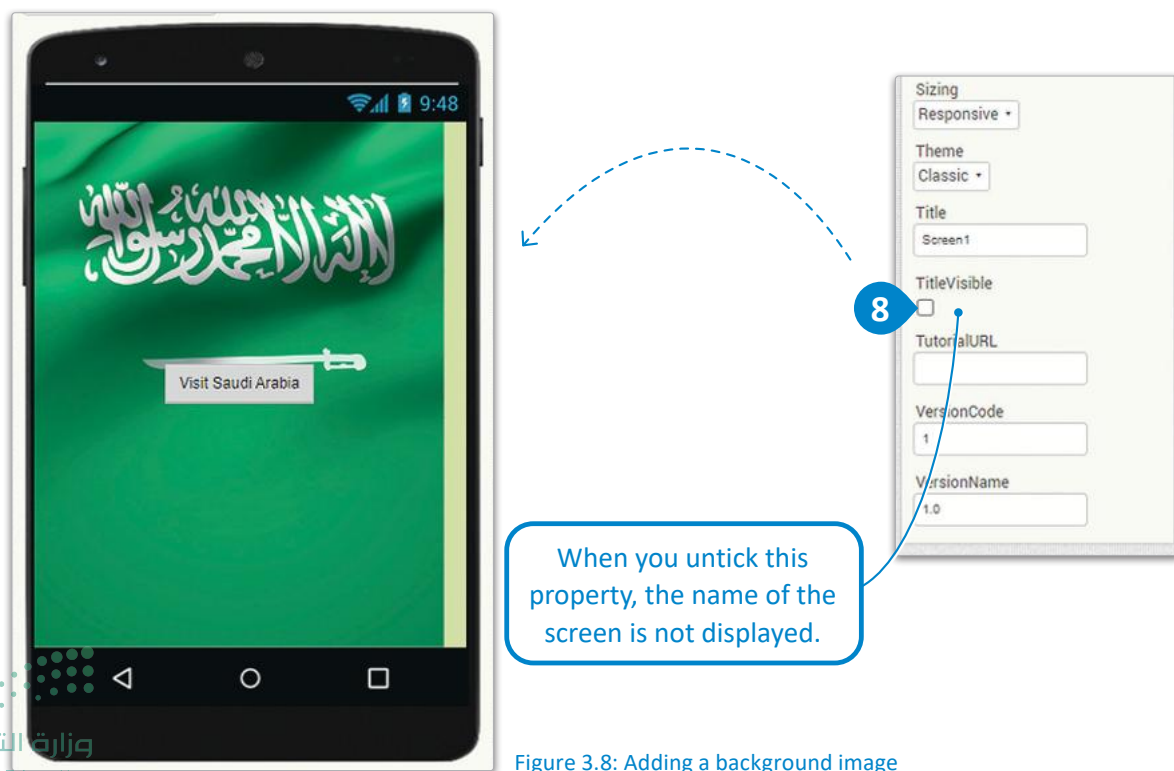
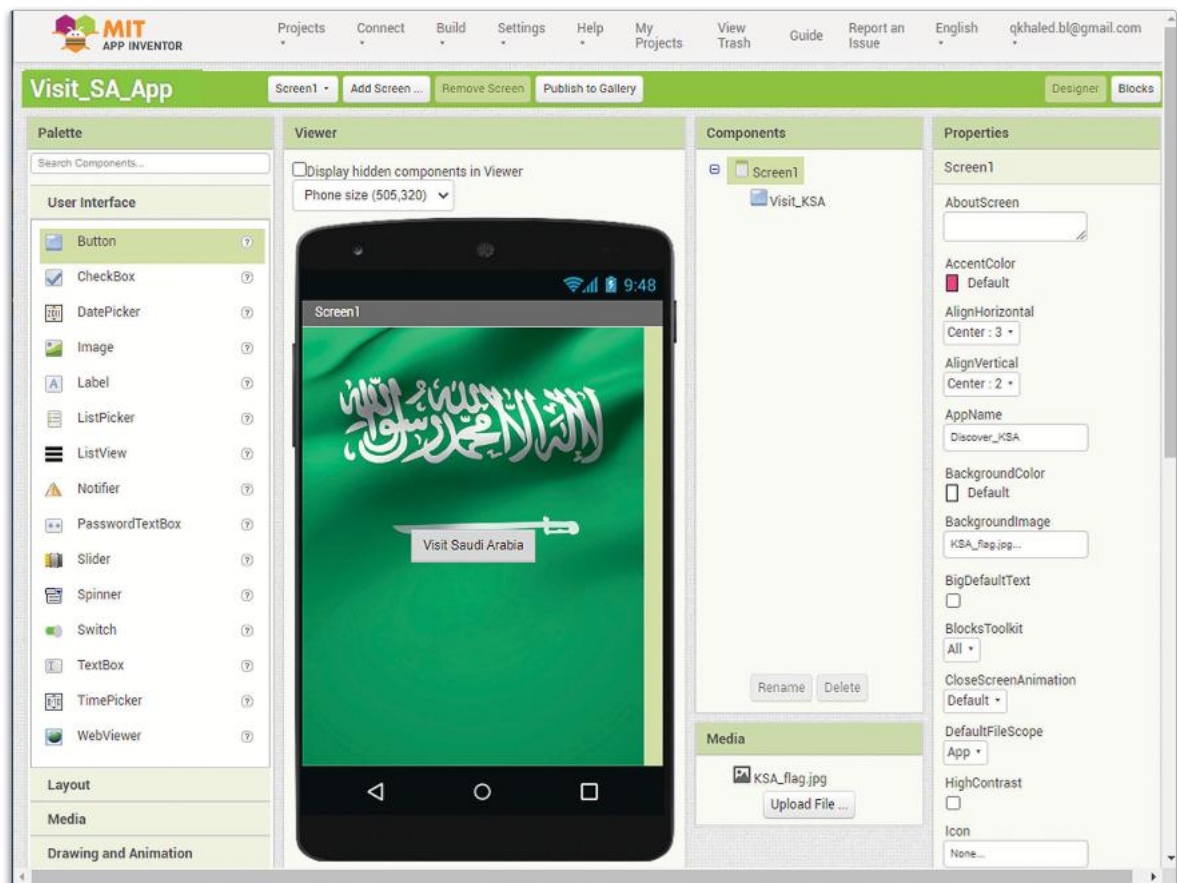


Figure 3.8: Adding a background image

To add a Label component:

- > From the **User Interface** group, add a **Label** component to the screen by dragging and dropping it below the button **1** and rename it **welcome_label** by selecting **Rename** from the **Components** panel. **2**
- > In the **welcome_label** component, clear the **Text** property **3** and set the **TextColor** property to **White**. **4**

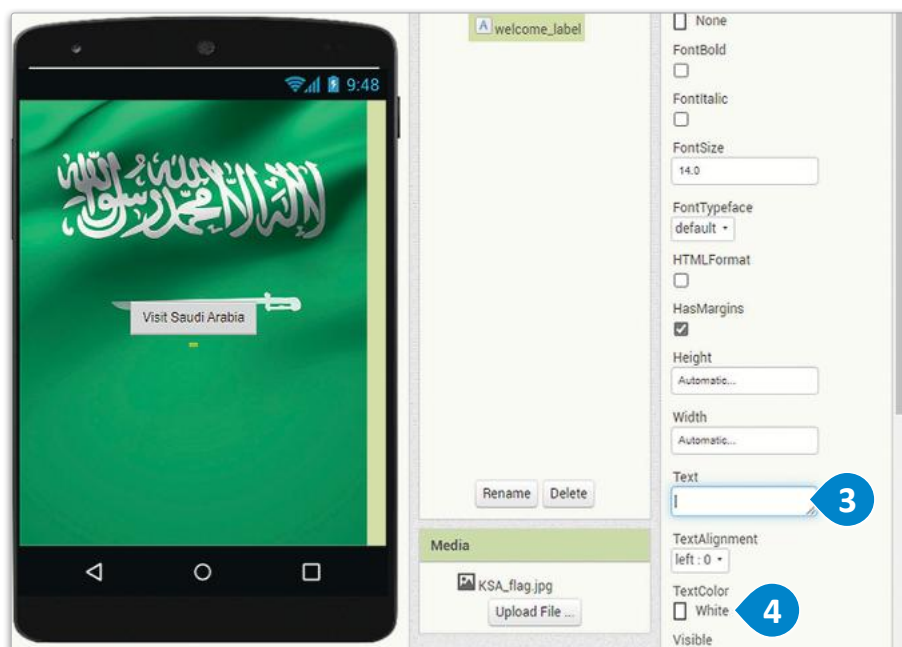
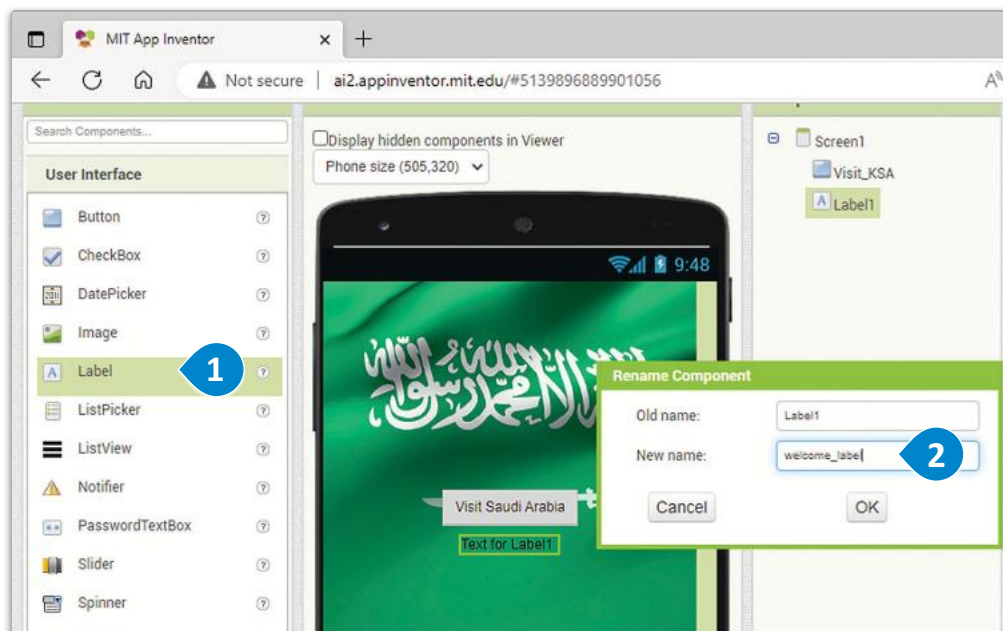


Figure 3.9: Adding a label component

Programming the Interactive Button

Now that you've added the command button, you'll add some code that makes the button display the sentence "Welcome to Saudi Arabia!" when pressed. To do this, you have to change the view from "Designer" to "Blocks".

The App Inventor Blocks Page

This is the **Blocks** page of the App Inventor interface. All the components that you add from the **Designer** page will be shown here and they can be programmed with a block-based visual programming language. There are block elements for program logic, event handling, variable manipulation and component alteration.

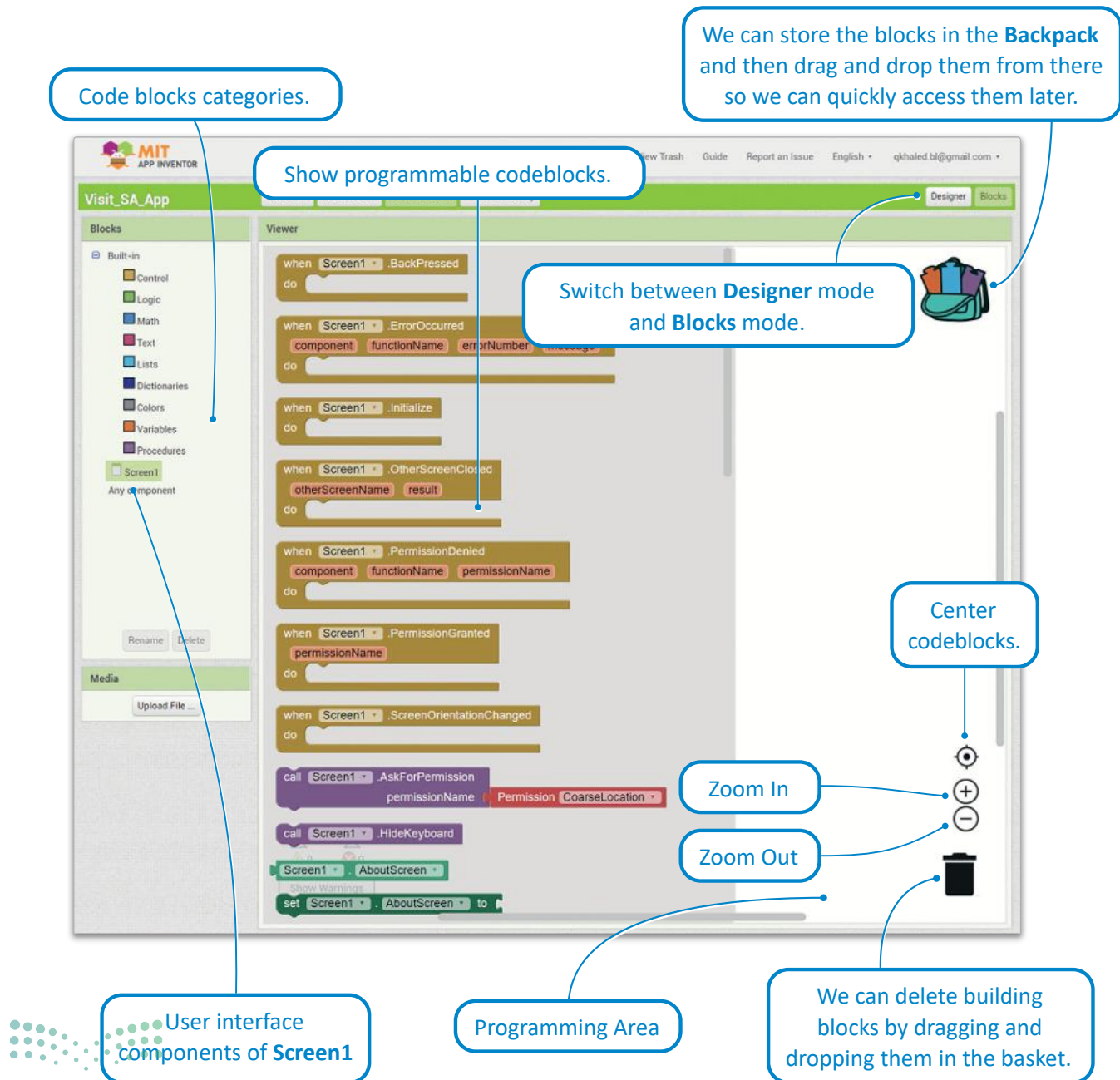


Figure 3.10: The App Inventor Blocks interface

Understanding the groups of programmable commands

	Control	Controlling the program flow.
	Logic	Performing logical operations.
	Math	Performing mathematical operations.
	Text	Performing operations on text and strings.
	Lists	Initializing and interacting with list data structures.
	Dictionaries	Initializing and interacting with dictionary data structures.
	Colors	Adding colors to components.
	Variables	Initializing and manipulating variables.
	Procedures	Performing custom procedures.

Each component that you select will have its own event handlers and operations to alter their properties.

To select the Click event for the button:

- > Select the **Visit_KSA** component. **1**
- > Select the **when Visit_KSA.Click** block. **2**
- > Drag and drop it into the programming area. **3**

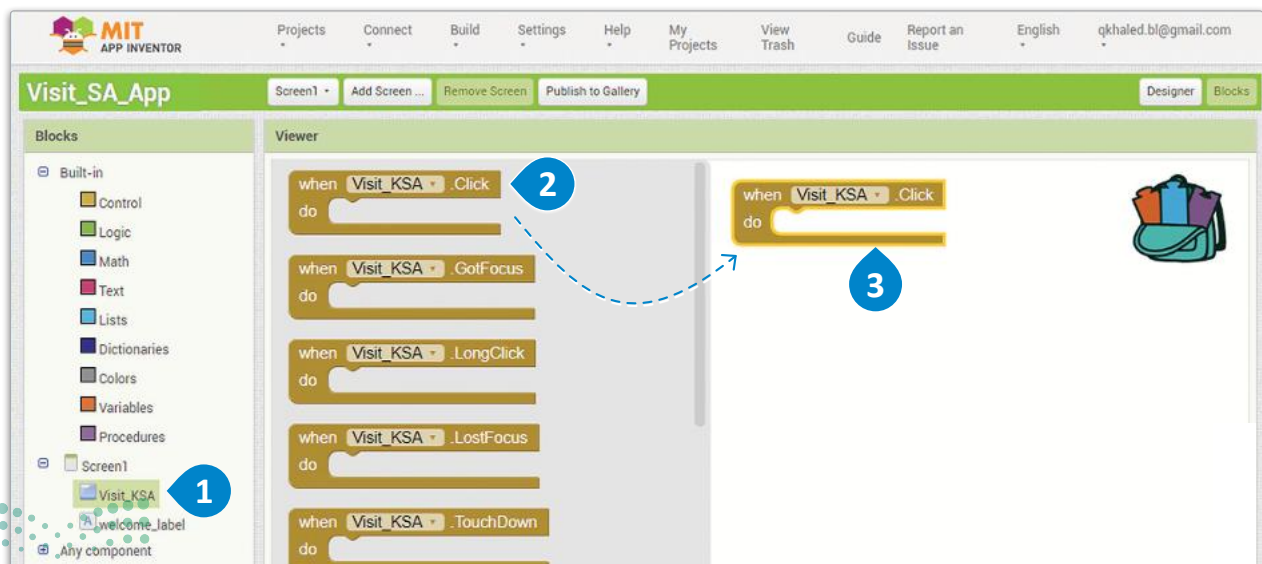


Figure 3.11: Selecting the Click event for the button

To access the Text property of the label:

- > Select the **welcome_label** component. ①
- > Drag and drop the **set welcome_label.Text** block. ②
- > Place it inside the **do** section of the **when Visit_KSA.Click** block. ③

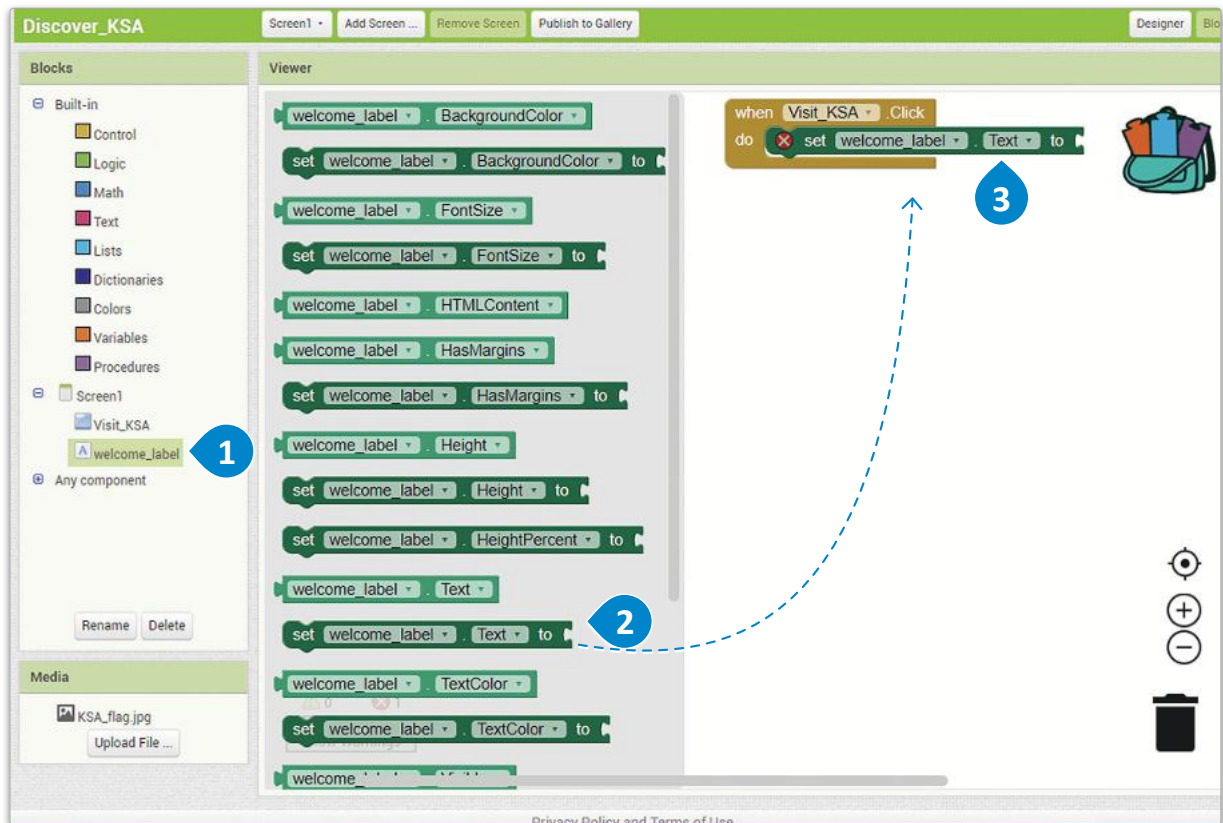


Figure 3.12: Accessing the Text property of the label



To modify the Text property of the label:

- > Select the **Text** group **1**
- > Select the **empty string** block. **2**
- > Place an **empty string** block in the **set Text to** block. **3**
- > Type **"Welcome to Saudi Arabia!"** in the **empty string** block. **4**

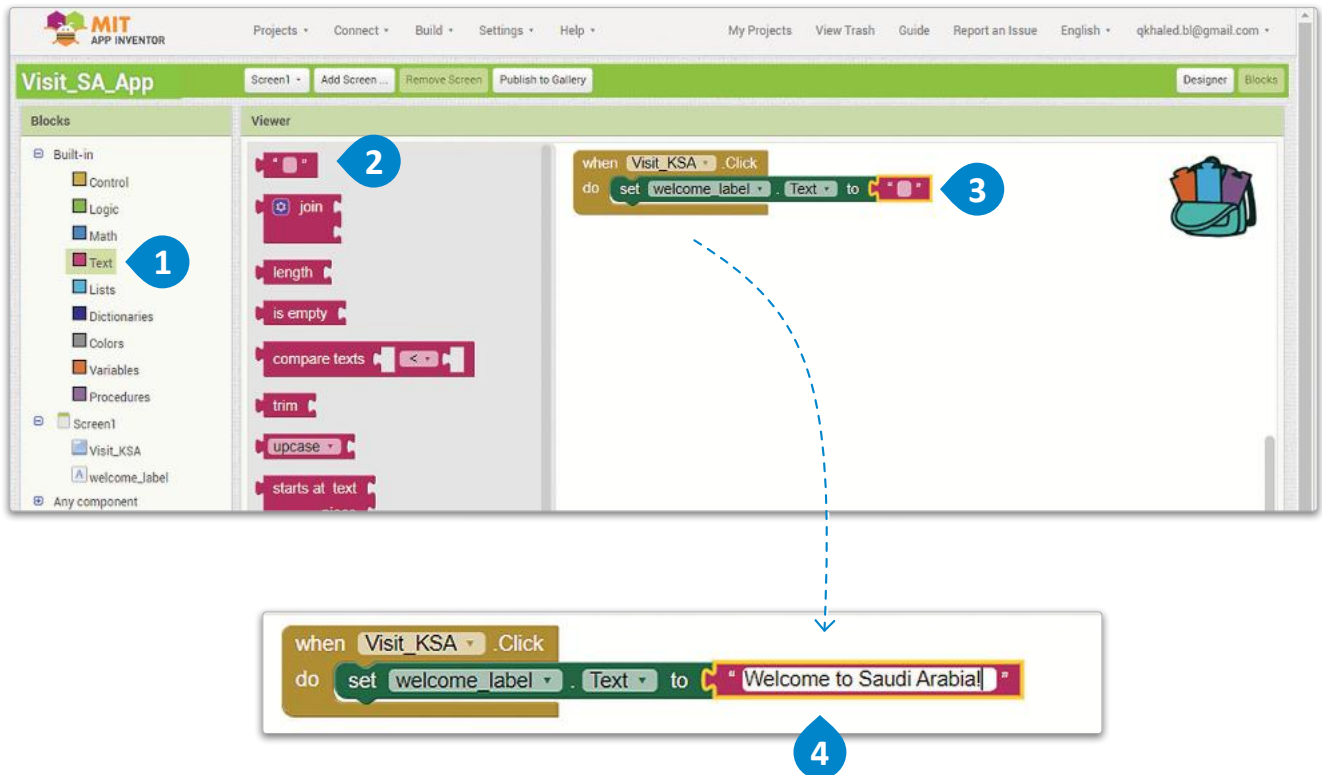


Figure 3.13: Modifying the Text property of the label



Testing the Application

At various stages in the development of an application, you need to test the application in order to make sure all the features are working. Frequent testing during development helps you discover potential bugs in the programming which should be corrected before publishing and delivering your application.

In MIT App Inventor there are two methods of testing your application. The first method is the **Emulator** which is a program you install on your computer that emulates a mobile device.

The second method is the MIT **AI2 Companion**, which is an application you install on your physical mobile phone. The App Inventor website provides you with a QR code that you scan with the MIT **AI2 Companion** application which loads the application that you have created on the browser on your physical phone. You can install the MIT **AI2 Companion** app at the following link:

<https://play.google.com/store/apps/details?id=edu.mit.appinventor.aicompanion3>

Setting up the Android Emulator

You will now install the Android Emulator to run the mobile application on your computer.

To setup the Android Emulator application:

- > Go to the following website: <https://appinventor.mit.edu/explore/ai2/windows>. ¹
- > Click on the **Download the installer** link to download the **.exe** installer file. ²
- > After the installer is downloaded, follow the steps as shown on the webpage. ³

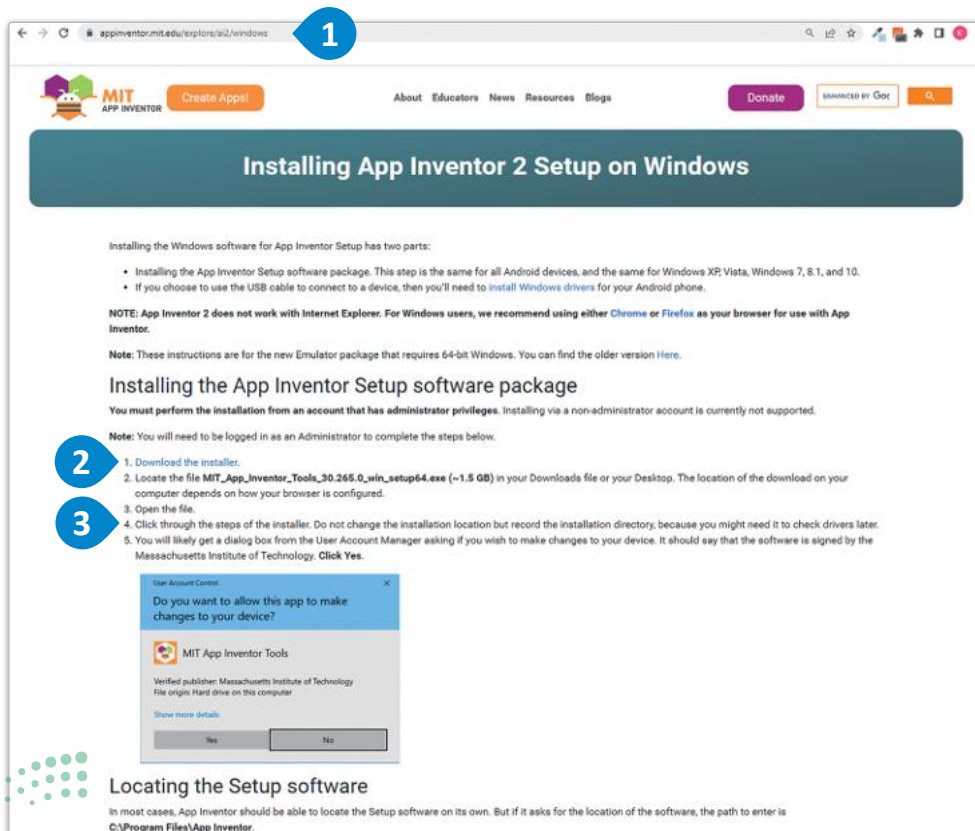


Figure 3.14: Installing the Android Emulator application

1. Run the Application with Android Emulator

To run the application:

- > Click **Connect**. ①
- > Choose **Emulator**. ②
- > Click the button to display the message. ③

The Emulator desktop application needs to be running before you initiate the connection on the App Inventor website.

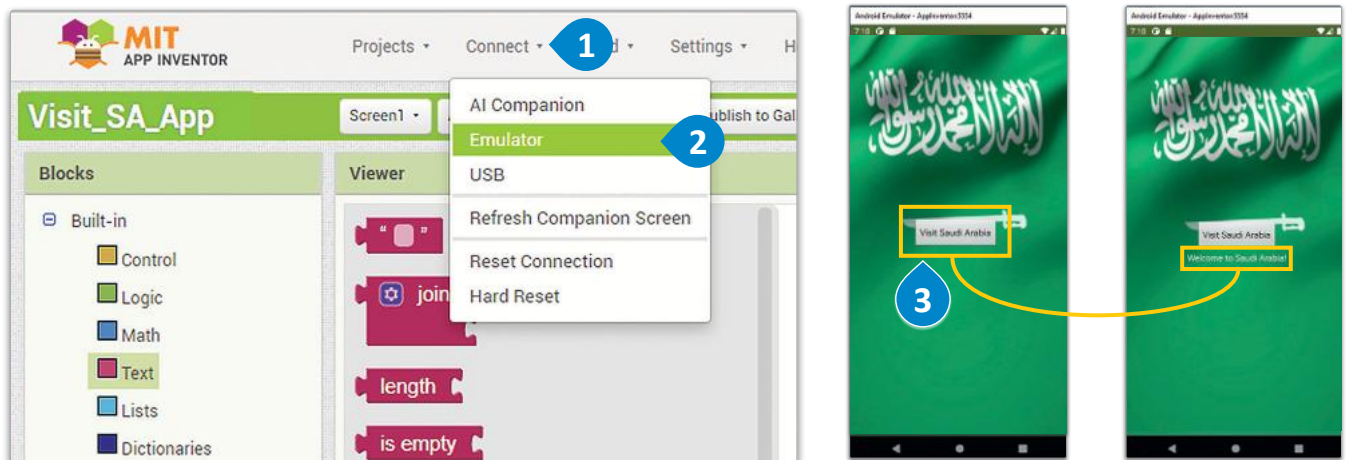


Figure 3.15: Testing an application with the Android Emulator

2. Run the Application with MIT AI2 Companion

To connect the app to the MIT AI2 Companion:

- > Click **Connect** ① and **AI Companion** from the top menu. ②
- > A dialog box with a QR code will appear on your PC screen. ③
- > On your mobile device, launch the **MIT AI2 Companion** app, then click the **Scan QR code** button on the **Companion**, ④ and scan the code in the App Inventor window and the app you are building will be displayed on your device.
- > Click the button to display the message. ⑤

Before scanning, both devices have to be connected to the same WiFi network.

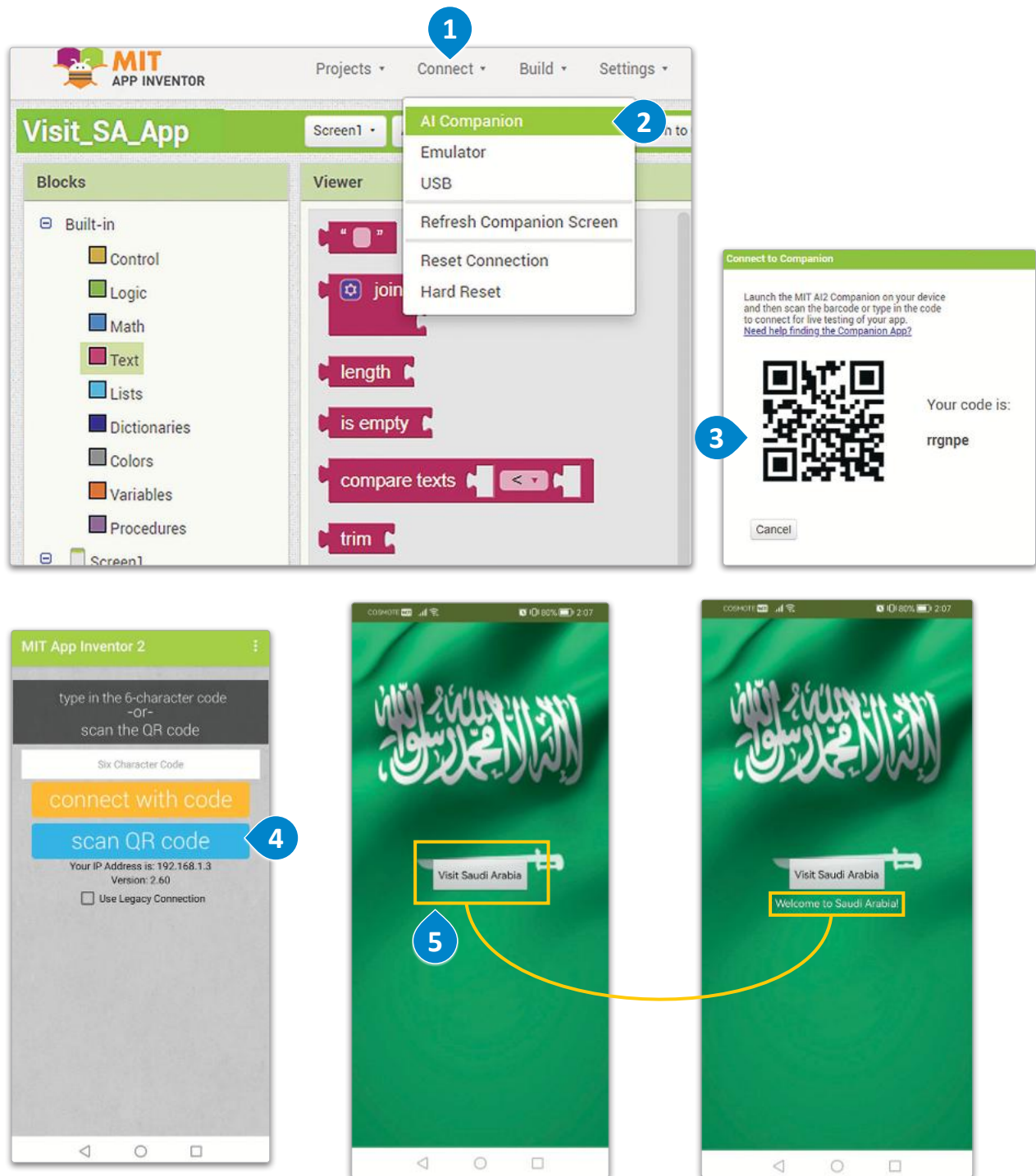


Figure 3.16: Testing an application with the MIT AI2 Companion

Exercises

- 1 Describe the four stages of developing a mobile application.

- 2 Compare how developing applications with MIT App Inventor differs from traditional mobile app development.

- 3 List the advantages of developing mobile apps with MIT App Inventor.



- 4 Create a simple application about a country you want to visit.
- Add a new screen named "Home" and add a background image with the country's flag.
 - Add two buttons named "Sightseeing" and "Useful Information".
 - Create a new screen and use the Label tool to write some useful information that will appear when the button is clicked.

- 5 Describe how a wireframe prototype will help with the development of the tourism application.



Lesson 2

Adding more Elements to the App

Link to digital lesson



www.iem.edu.sa

Designing the Home Screen

In the previous lesson, you created the Home screen of the app with a button, which when pressed, made a message appear.

Now, you will add some useful buttons on the Home screen and make some changes to its appearance.

To add a **VerticalArrangement** component:

- > From the Components section, select **Visit_KSA** button **1** and then click **Delete**. **2**
- > From the **Layout** group, add a **VerticalArrangement** component to the screen by dragging and dropping it in the viewer. **3**
- > In the **VerticalArrangement1** component, set the **AlignHorizontal** property to **Center**: **3**, **4** the **AlignVertical** property to **Center**: **2** **5** and the **BackgroundColor** property to **None**. **6**
- > Set the **Height** property to **Fill parent** **7** and the **Width** property to **Fill parent**. **8**

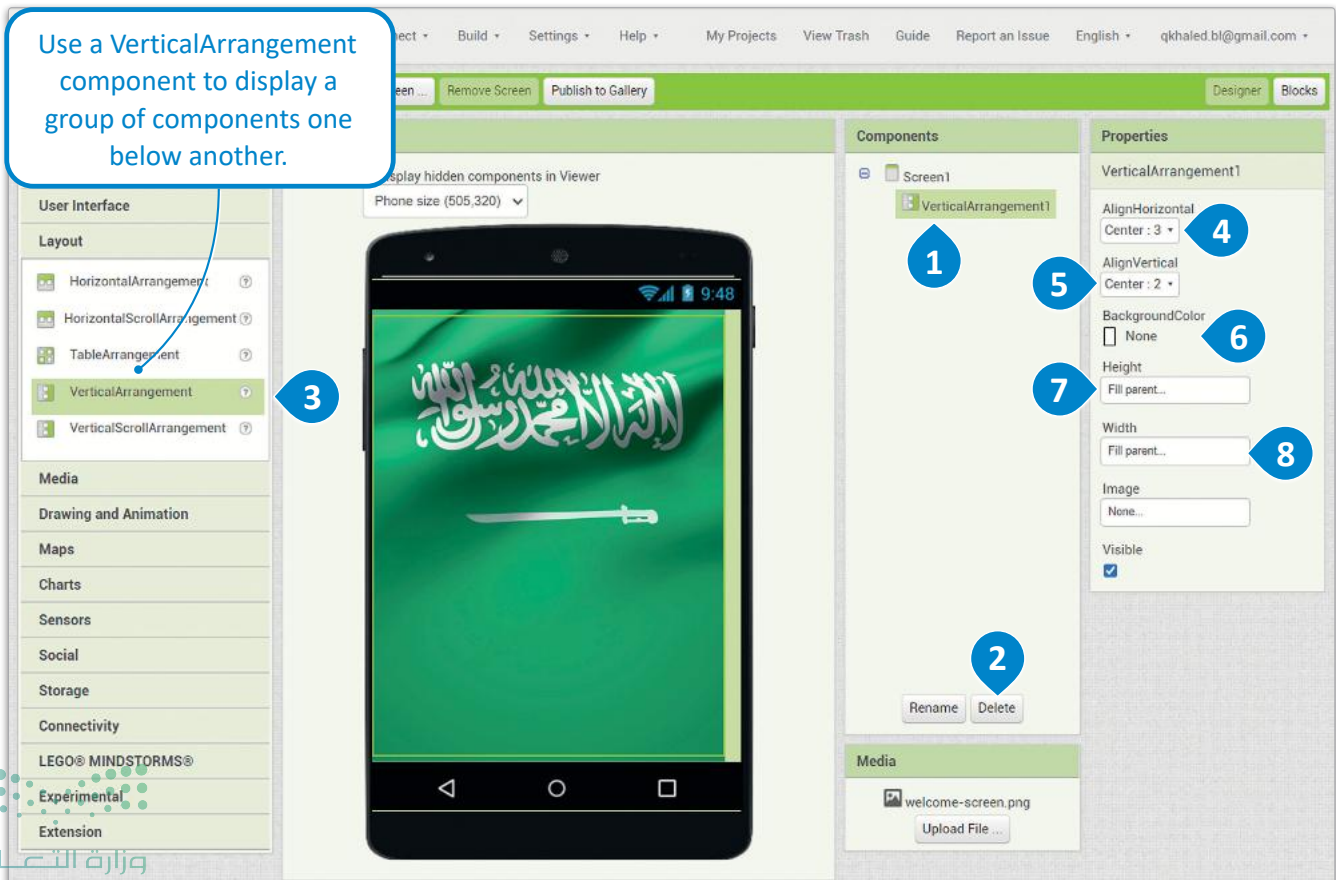


Figure 3.17: Adding a VerticalArrangement component

To add an English language button:

- > From the **User Interface** group, in the palette panel, add a **Button** component to the screen **1** and rename it **discover_button_en**. **2**
- > In the **discover_button_en** component, set the **BackgroundColor** property to **Custom** **3** and type the value **#28a595ff**, **4** set the **Text** property to **"Discover"**, **5** set the **Shape** property to **rectangular** **6** and set the **TextColor** property to **White**. **7**
- > Repeat the steps to add an Arabic language button. **8**

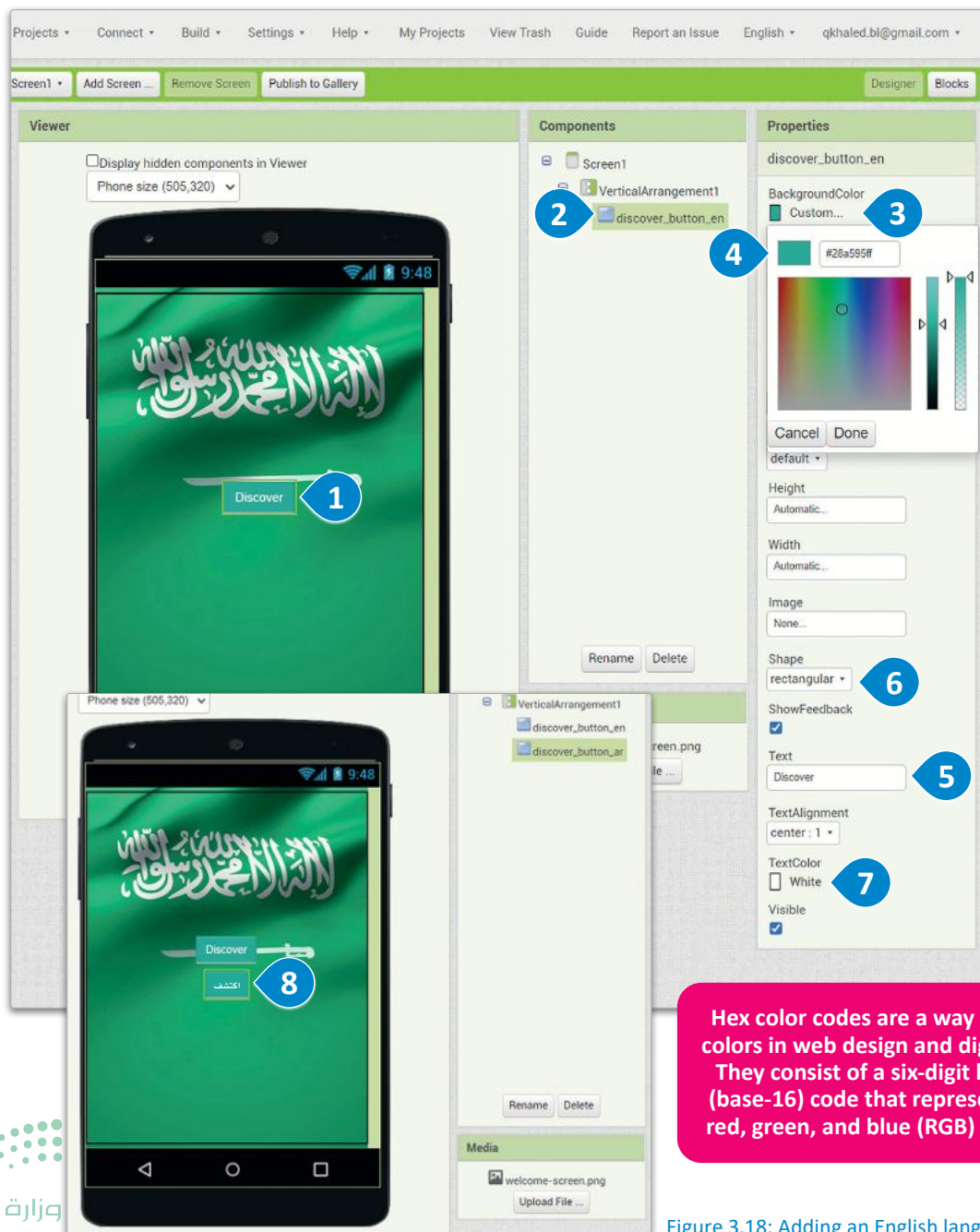


Figure 3.18: Adding an English language button

Creating the Second Screen of the App

The next screen will be the one that shows the user the cities of Riyadh and Jeddah and their highlights. When the user clicks on one of the cities, a list of available highlights will appear.

Now, you will add some useful buttons on the Home screen and make some changes to its appearance.

To add a new screen:

- > Click on the **Add Screen** button **1** and create a New Screen with the name **Cities**. **2**
- > In the **Properties** section of **Cities**, untick the **TitleVisible** property **3** and in the **BackgroundImage** property put an image of a Saudi flag. **4**

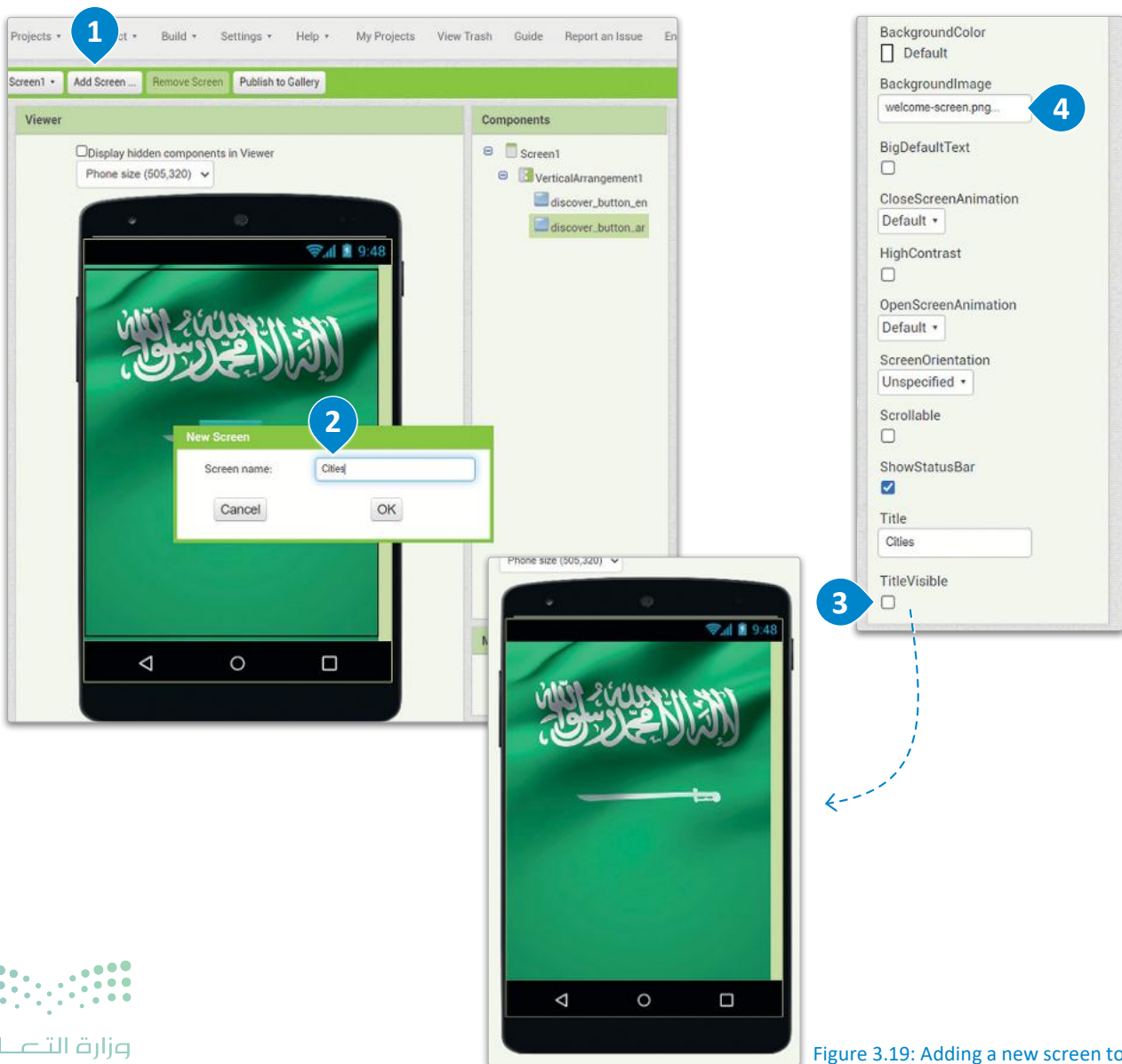


Figure 3.19: Adding a new screen to the app

To add a label:

- > From the **User Interface** group, add a **Label** component to the screen **1** and rename it **discover_label**. **2**
- > In the **discover_label** component, set the **BackgroundColor** property to **Black**, **3** set the **Width** property to **Fill parent**, **4** set the **Text** property to **"Discover Saudi Arabia"** **5** and set the **TextColor** property to **White**. **6**

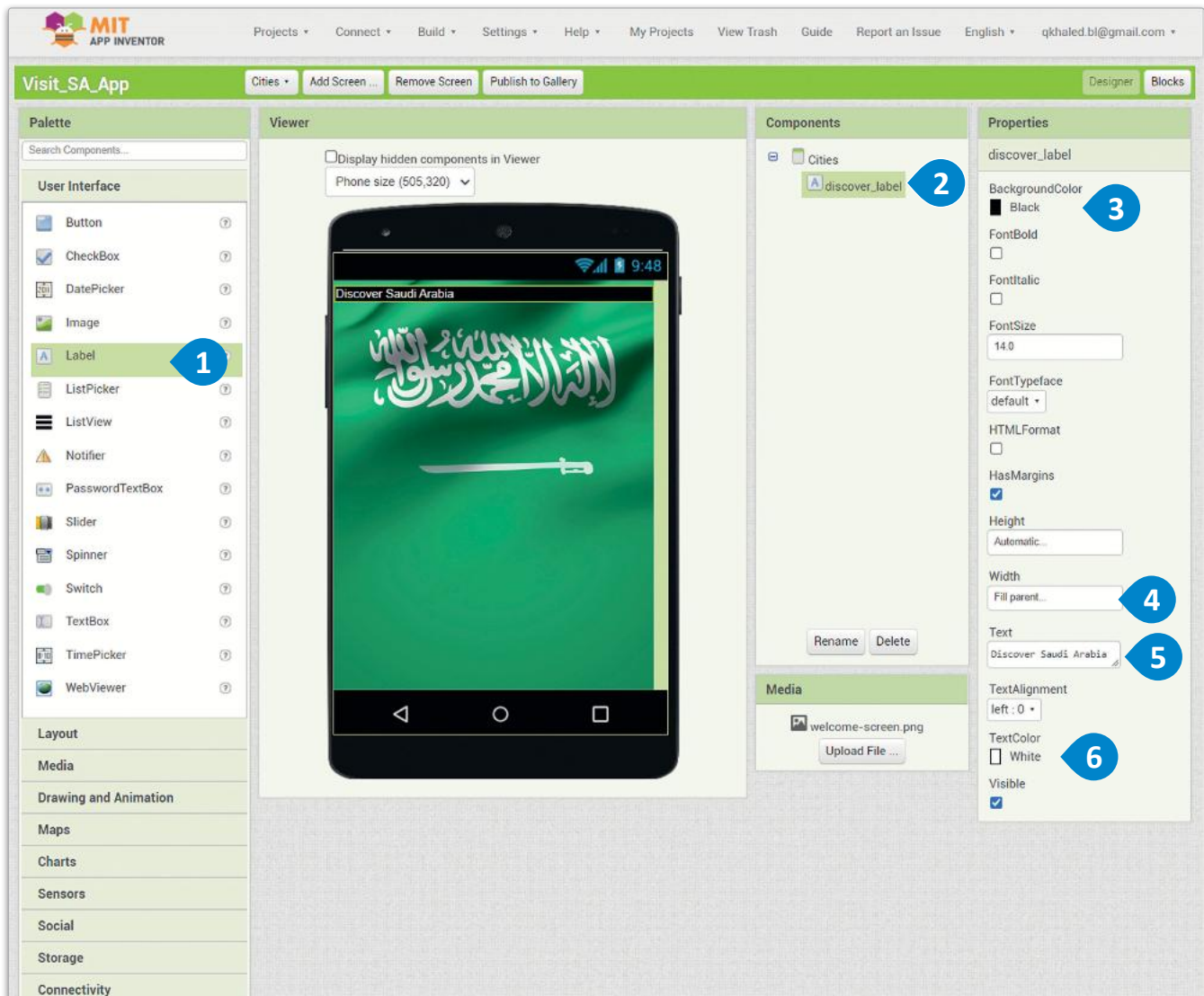


Figure 3.20: Adding a label component



To add a VerticalArrangement component:

- > From the **Layout** group, add a **VerticalArrangement** component to the screen. **1**
- > In the **VerticalArrangement1** component, set the **BackgroundColor** property to **Custom** and type the value **#11613eff**. **2**
- > In the **VerticalArrangement1** component, set the **Height** property to **Fill parent** **3** and the **Width** property to **Fill parent**. **4**

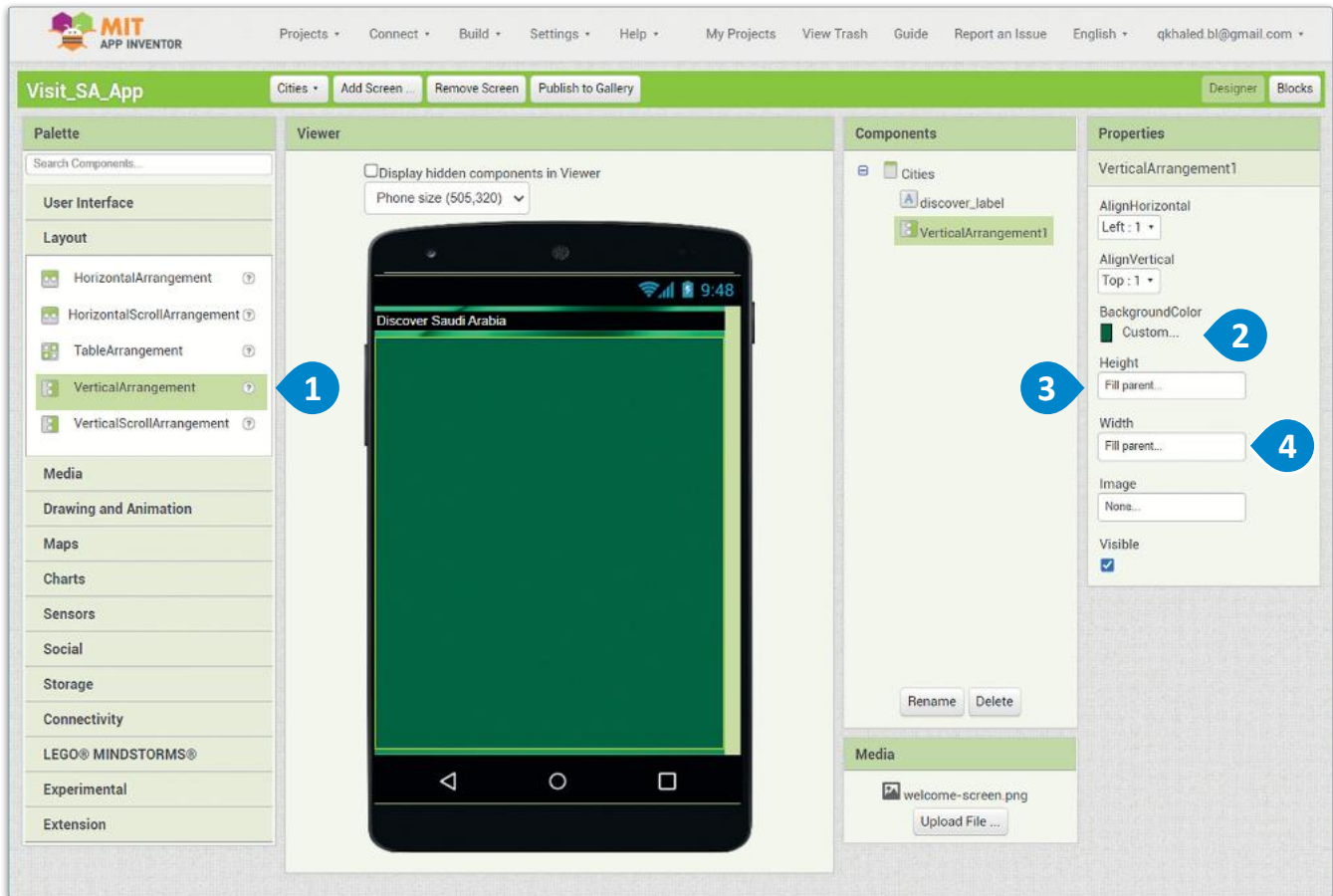


Figure 3.21: Adding a VerticalArrangement component



To add a cities label:

- > From the **User Interface** group, add a **Label** component to the screen **1** and rename it **cities_label**. **2**
- > In the **cities_label** component, set the **BackgroundColor** property to **None** **3** set the **FontSize** property to **18.0** **4** set the **Text** property to **"Cities"** and set the **TextColor** property to **White**. **5**

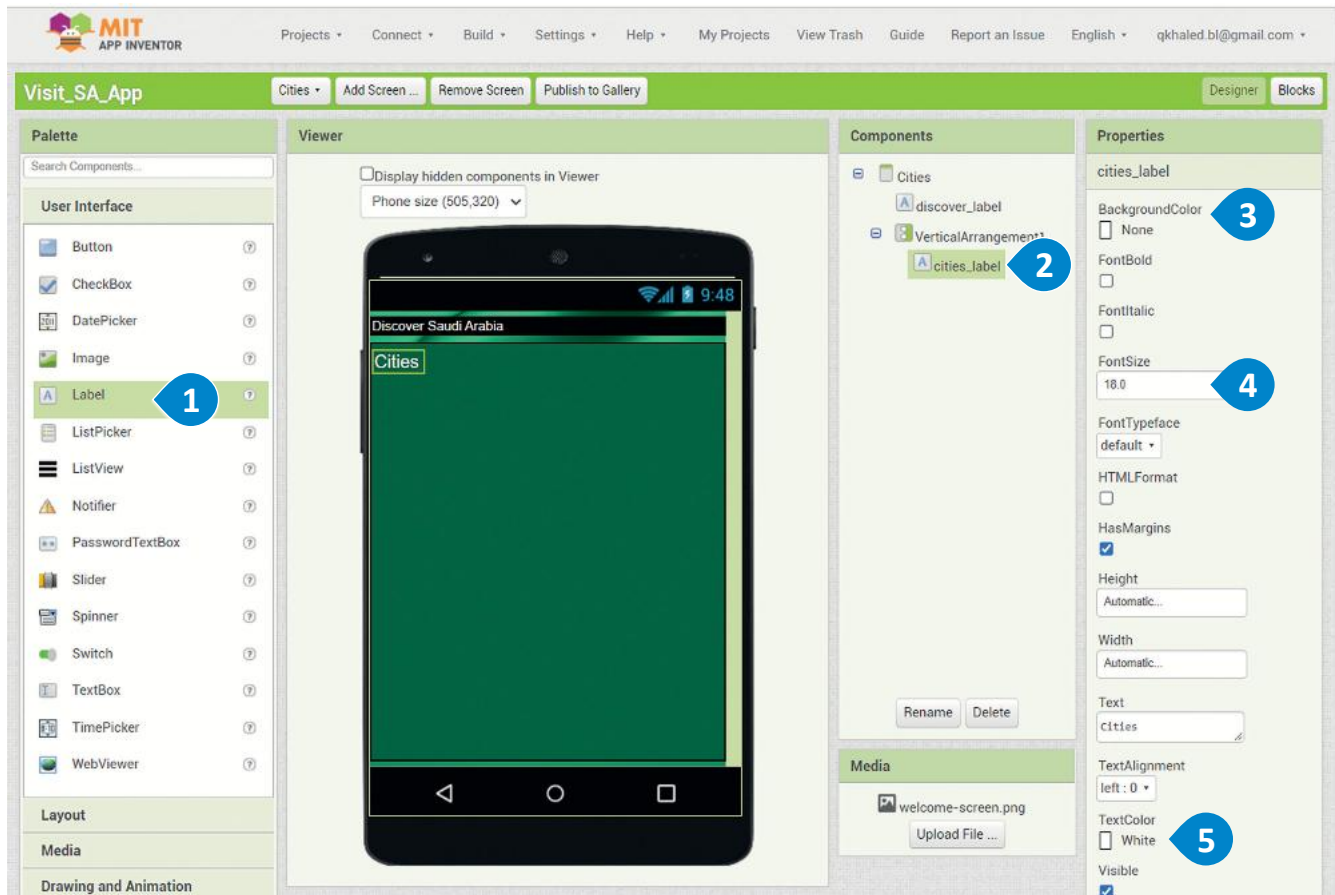


Figure 3.22: Adding a text label

Creating a List

Lists are a type of data structure we use to create and manage different combinations of values/items, and in your app, a list will be contained in each image you will add.

For example, when you press the image of Riyadh city, a list of two highlights of this city will appear as follows:

- Al Masmak
- Boulevard Riyadh City

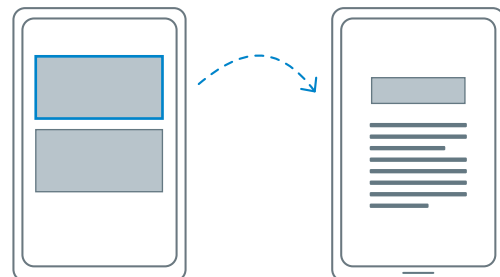


Figure 3.23: Opening a new page from the ListPicker

The **ListPicker** component will be used to select the highlight from each city. Each city will be represented by a **ListPicker**. So, there will be a **ListPicker** for **Riyadh** and a **ListPicker** for **Jeddah**.

To add a ListPicker component for Riyadh:

- > From the **User Interface** group, add a **Label "RIYADH"** ¹ and then a **ListPicker** component to the screen ² and rename it **riyadh_list**. ³
- > In the **riyadh_list** component, set the **Height** property to **Fill parent** ⁴ and the **Width** property to **Fill parent**, ⁵ upload an image of Riyadh from the **Image** property ⁶ and clear the **Text** property. ⁷
- > Repeat the steps to add a **Label "JEDDAH"** and a **ListPicker** for the city of Jeddah. ⁸

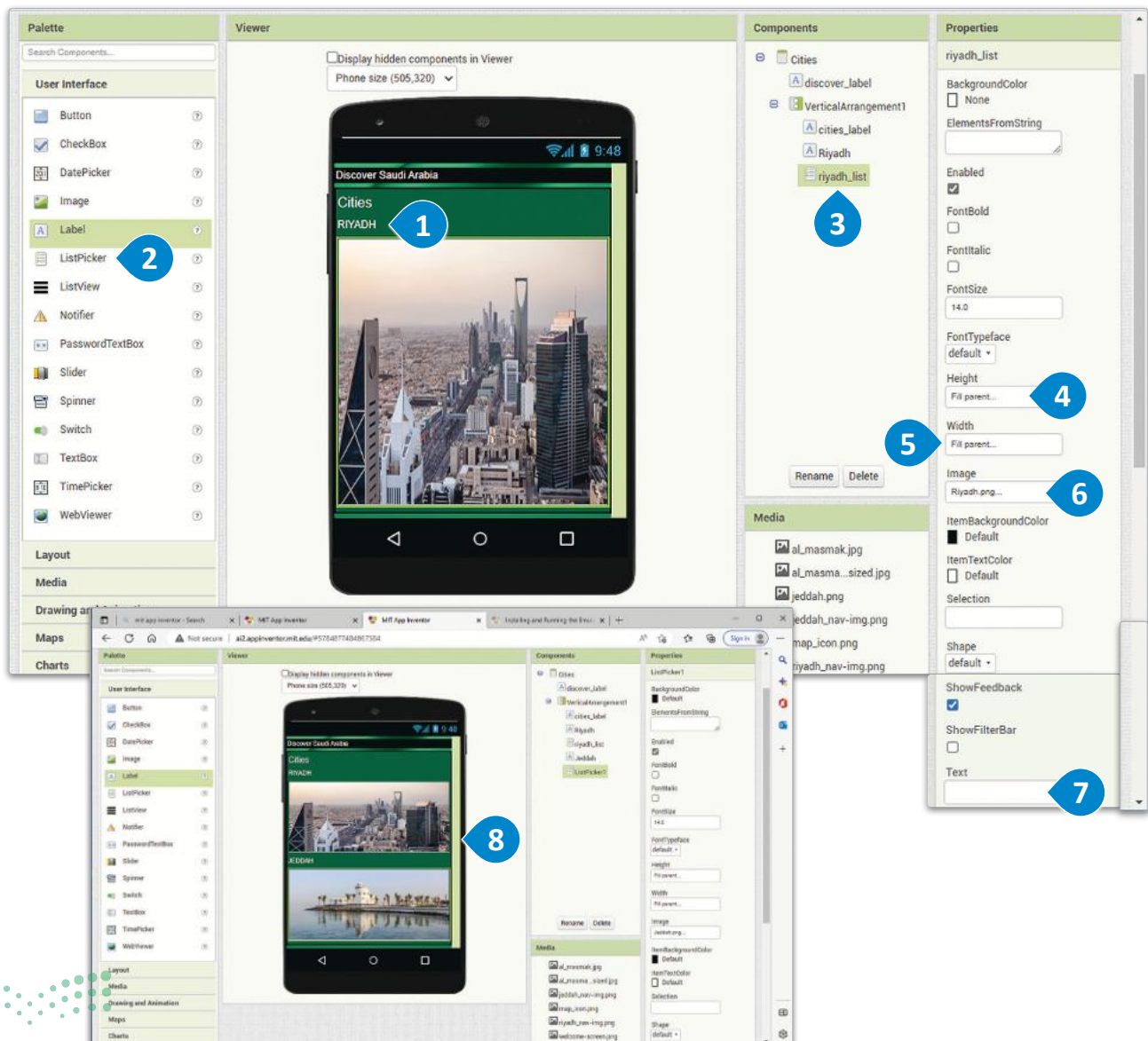


Figure 3.24: Adding a ListPicker component

When you run the finished application on your mobile phone, the **ListPicker** component works in the following way. When you select an image of each **ListPicker**, the screen contents will change to show the list of options. For example, when you click on the ListPicker component for **Riyadh**, the application will have the following behavior:

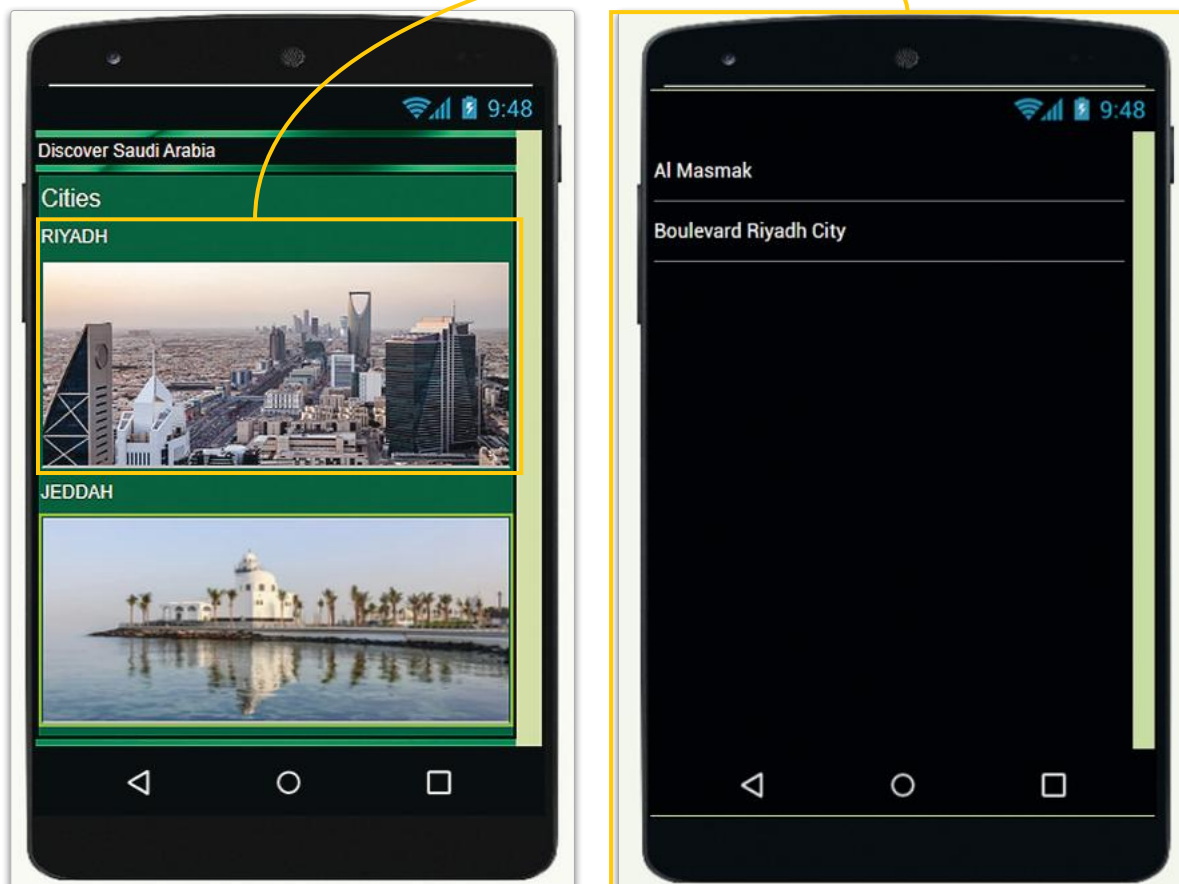


Figure 3.25: The ListPicker component

Creating the Third Screen of the App

The final screen will be the screen that displays information about the selected highlight from the previous screen. This screen will show the title of the highlight, a representative image, a text description and a map button which will launch an interactive map to view the location of the highlight in the respective city. Each highlight will have its dedicated screen, and in this lesson, you will create a screen for **Al Masmak**.



You will need to remove the original screen title label and replace it with the Discover Saudi Arabia label.

Add a new screen as you have already learned and rename it "AlMasmak". Then add the KSA flag as a background and the screen title as shown below.

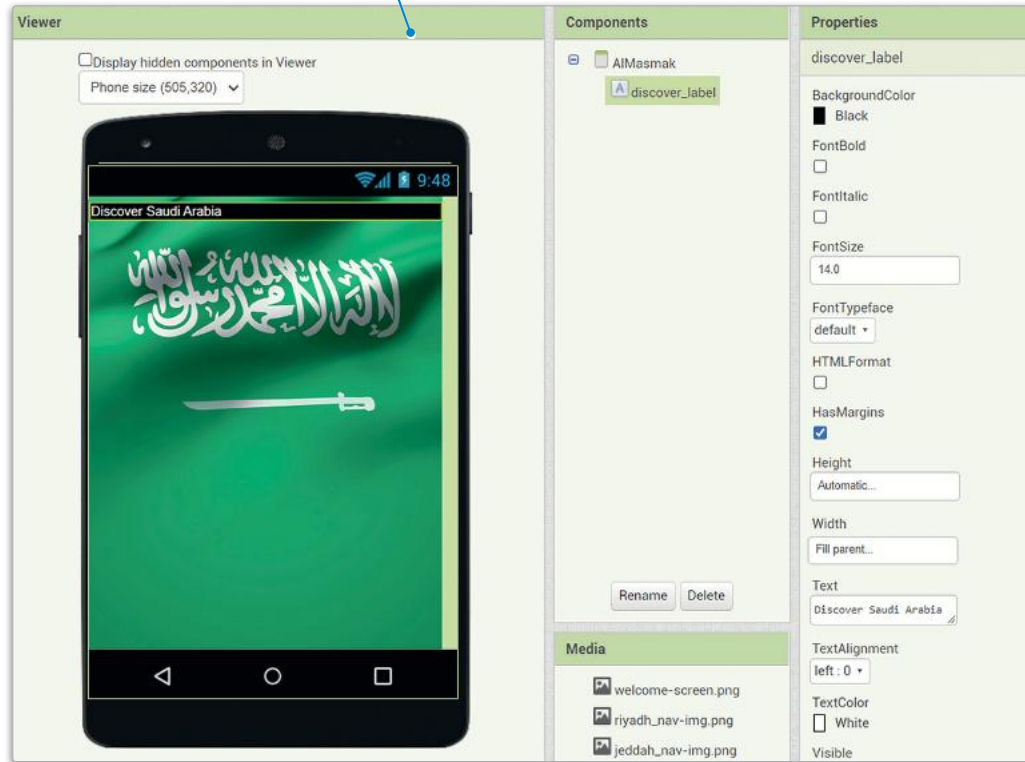


Figure 3.26: Creating the third screen

To add elements on the screen:

- > From the **Layout** group, add a **VerticalArrangement** component to the screen. ❶
- > In the **VerticalArrangement1** component set the **BackgroundColor** property to **Custom** ❷ and type the value **#11613eff**.
- > In the **VerticalArrangement1** component, set the **Height** property to **Fill parent** ❸ and the **Width** property to **Fill parent**. ❹
- > From the **User Interface** group, add a **Label** component to the screen and rename it **title_label**. ❺
- > In the **title_label** component, set the **BackgroundColor** property to **None**, ❻ set the **FontSize** property to **18.0** ❼ set the **Text** property to **"Al Masmak"** ❽ and set the **TextColor** property to **White**. ❾



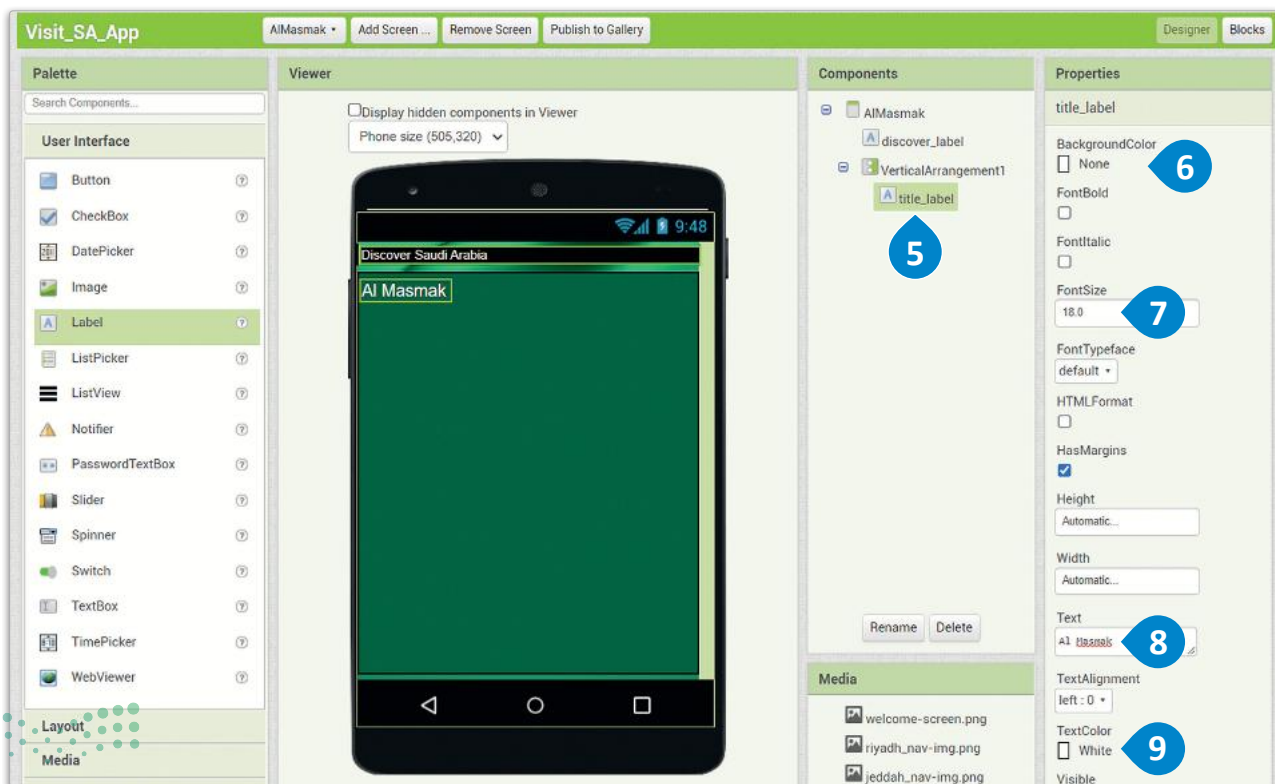
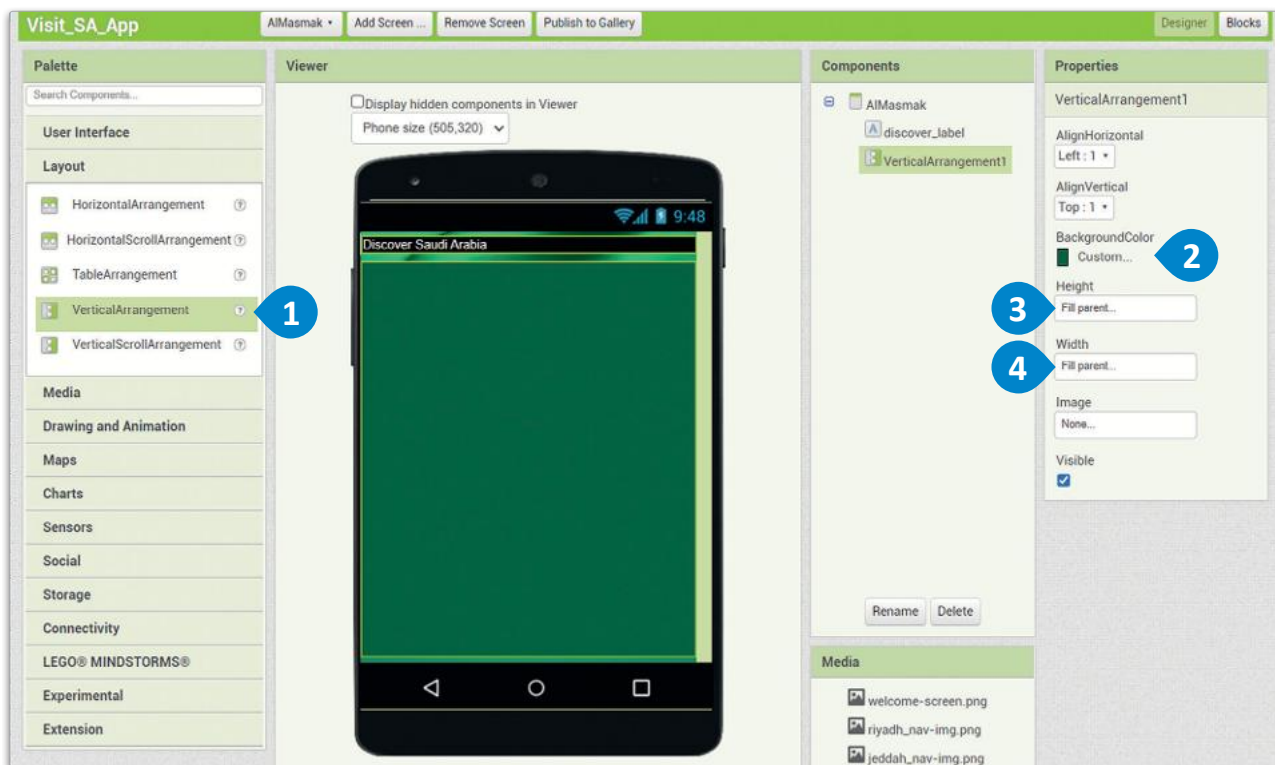


Figure 3.27: Adding elements to the third screen

To add an image component:

- > From the **User Interface** group, add an **Image** component to the screen **1** and rename it **image**. **2**
- > In the **image** component, set the **Height** property to **Fill parent** **3** and the **Width** property to **Fill parent** **4** and set the **Picture** property to an image of **Al Masmak**. **5**

The component instance name cannot be the same as the name of the component type, but **image** and **Image** are different names.

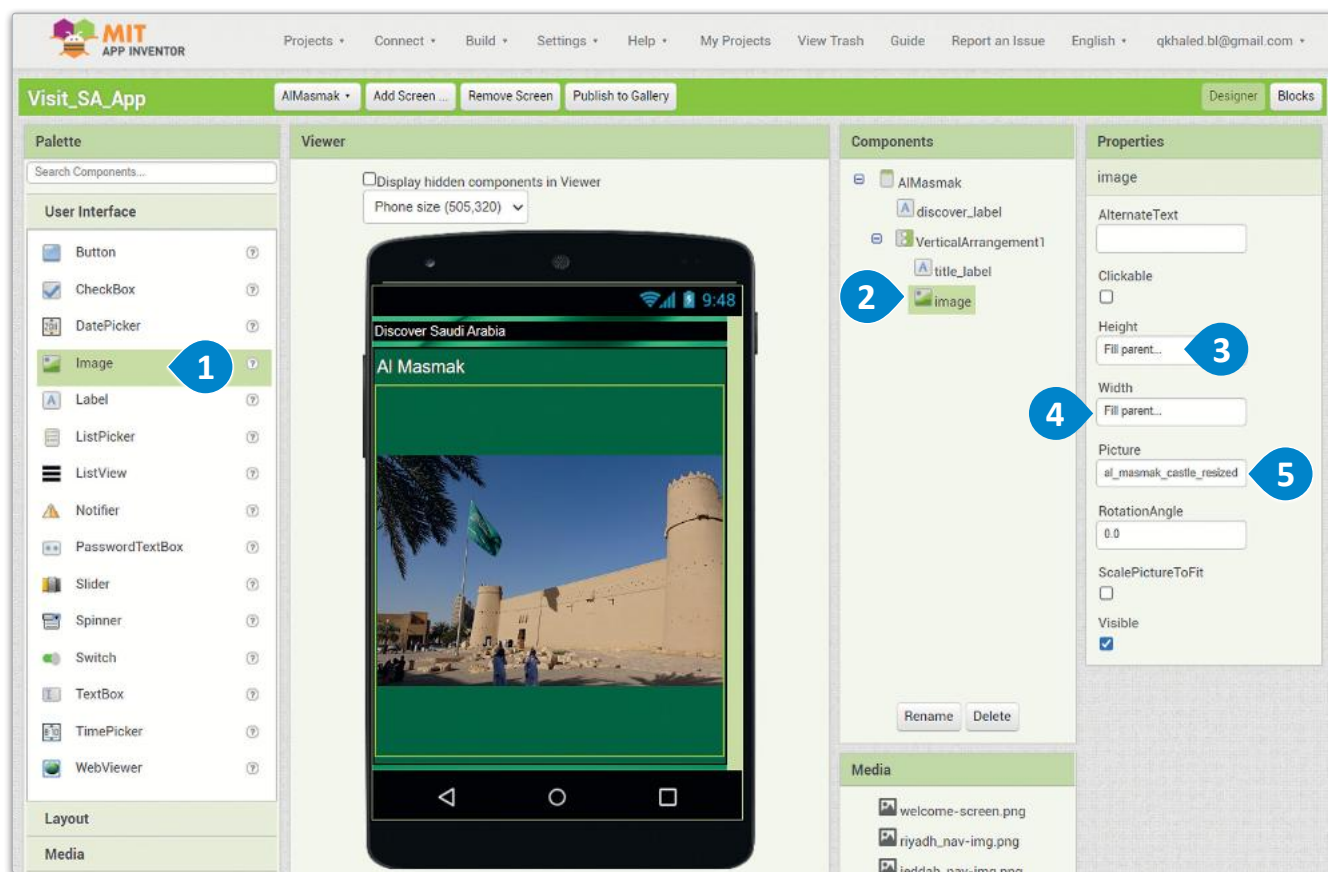


Figure 3.28: Adding an image component

Now, you will add a Label that will contain a description of Al Masmak. But, at this point, you will add a placeholder for the text that will be added in the next lesson.

To add a text description component:

- > From the **User Interface** group, add a **Label** component to the screen **1** and rename it **description_label**. **2**
- > In the **description_label** component, set the **Height** property to **Fill parent**, **3** set the **BackgroundColor** property to **None** **4** and the **TextColor** property to **White**. **5**

When more components are added, the image will be scaled properly.

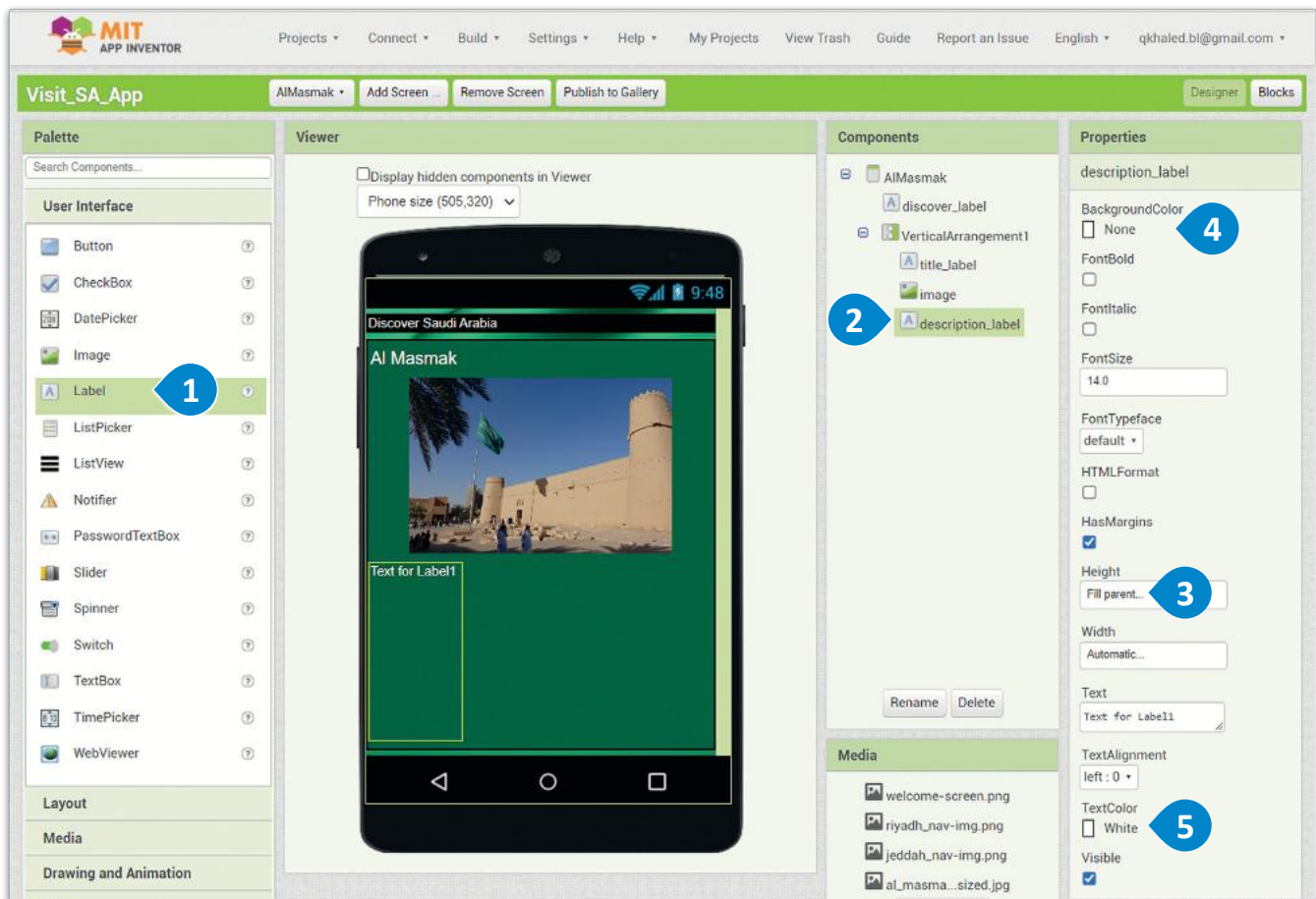


Figure 3.29: Adding a text description component

Adding an Interactive Map to the Application

On the screen of each highlight, the users will be able to launch an interactive map that displays the accurate location of the highlight so that they can see where each highlight is located in each city. You will first create a container for a button that will activate the interactive map, and then you will add the component.

Horizontal Arrangement Component

- With the HorizontalArrangement component, objects are arranged horizontally along the horizontal axis and aligned vertically in the center.
- If the Height or Width property is set to Automatic, then the actual height of the component is specified as the length of the tallest object within it.
- If the Height property for the HorizontalArrangement component is blank, the Height will be 100.
- If the Height or Width property of the HorizontalArrangement component is specified by Fill Parent or pixels (in pixels), any Width property specified by the Fill Parent will equally take up space not occupied by the other components.



To add a HorizontalArrangement component:

- > From the **Layout** group, add a **HorizontalArrangement** component to the screen. **1**
- > In the **HorizontalArrangement1** component, set the **BackgroundColor** property to **None**, **2** set the **AlignHorizontal** property to **Center : 3**, **3** and set the **AlignVertical** property to **Bottom : 3**. **4**
- > Set the **Height** property to **Fill parent** **5** and the **Width** property to **Fill parent**. **6**

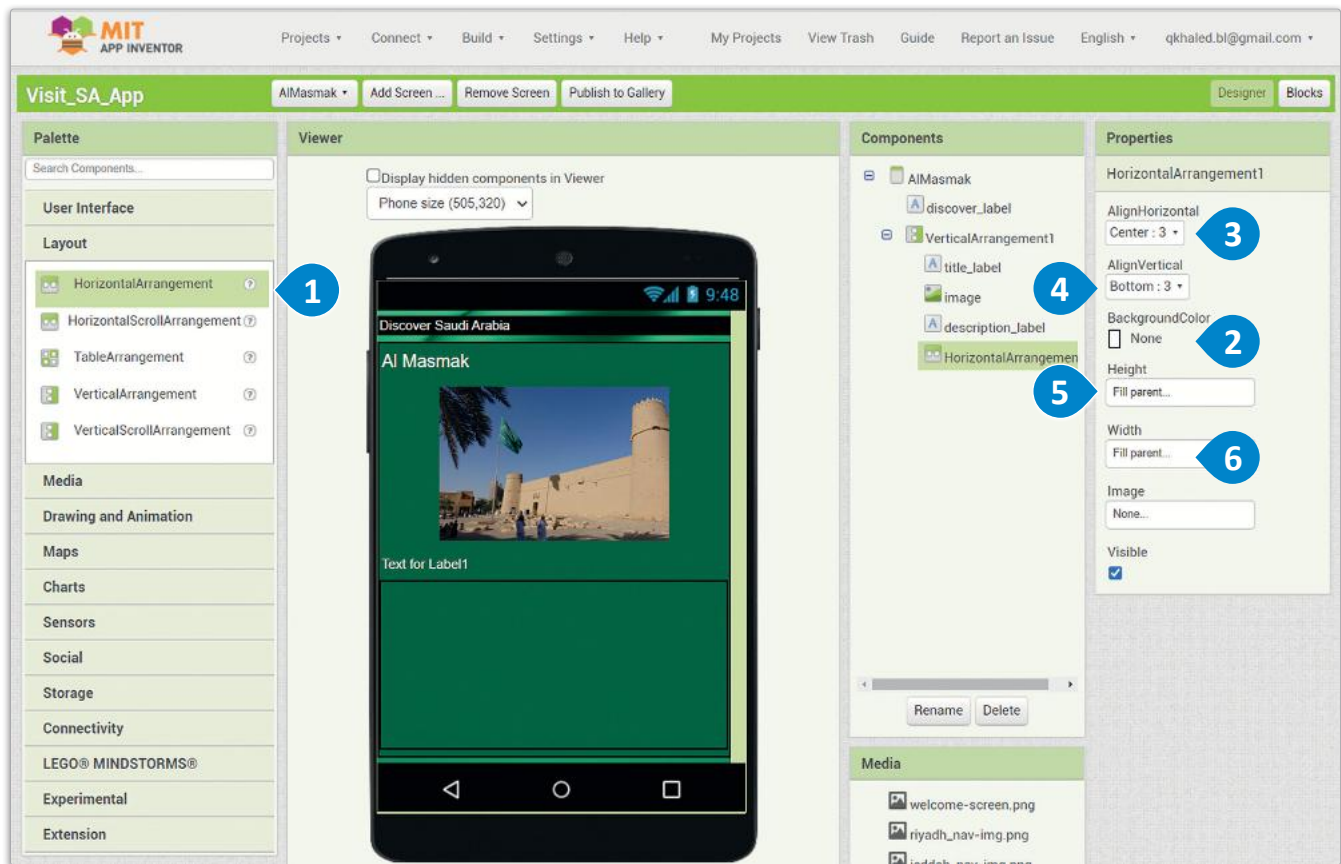


Figure 3.30: Adding a HorizontalArrangement component

To add a map button:

- > From the **User Interface** group, add a **Button** component to the screen **1** and rename it **map_button**. **2**
- > In the **map_button** component, set the **BackgroundColor** property to **None**, **3** clear the **Text** property **4** and set the **Image** property to an icon of a map. **5**

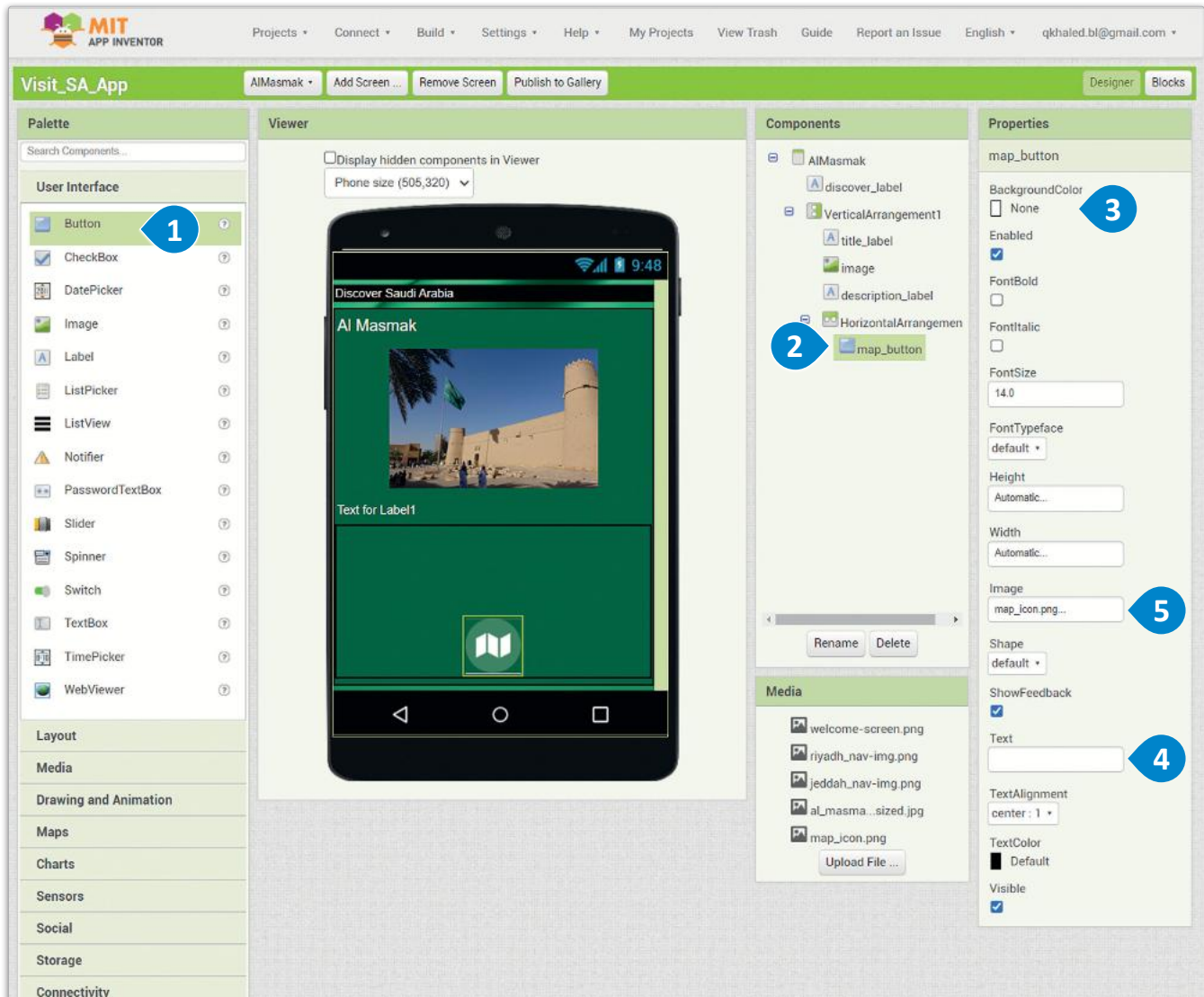


Figure 3.31: Adding a map button



To add a map component:

- > From the **Maps** group, select the **Map** component **1** and place it under **HorizontalArrangement1**. **2**
- > Set the **Height** property to **Fill Parent** **3** and the **Width** property to **Fill Parent**. **4**
- > Untick the **Visible** property. **5**
- > Set the **ZoomLevel** property to **16**. **6**

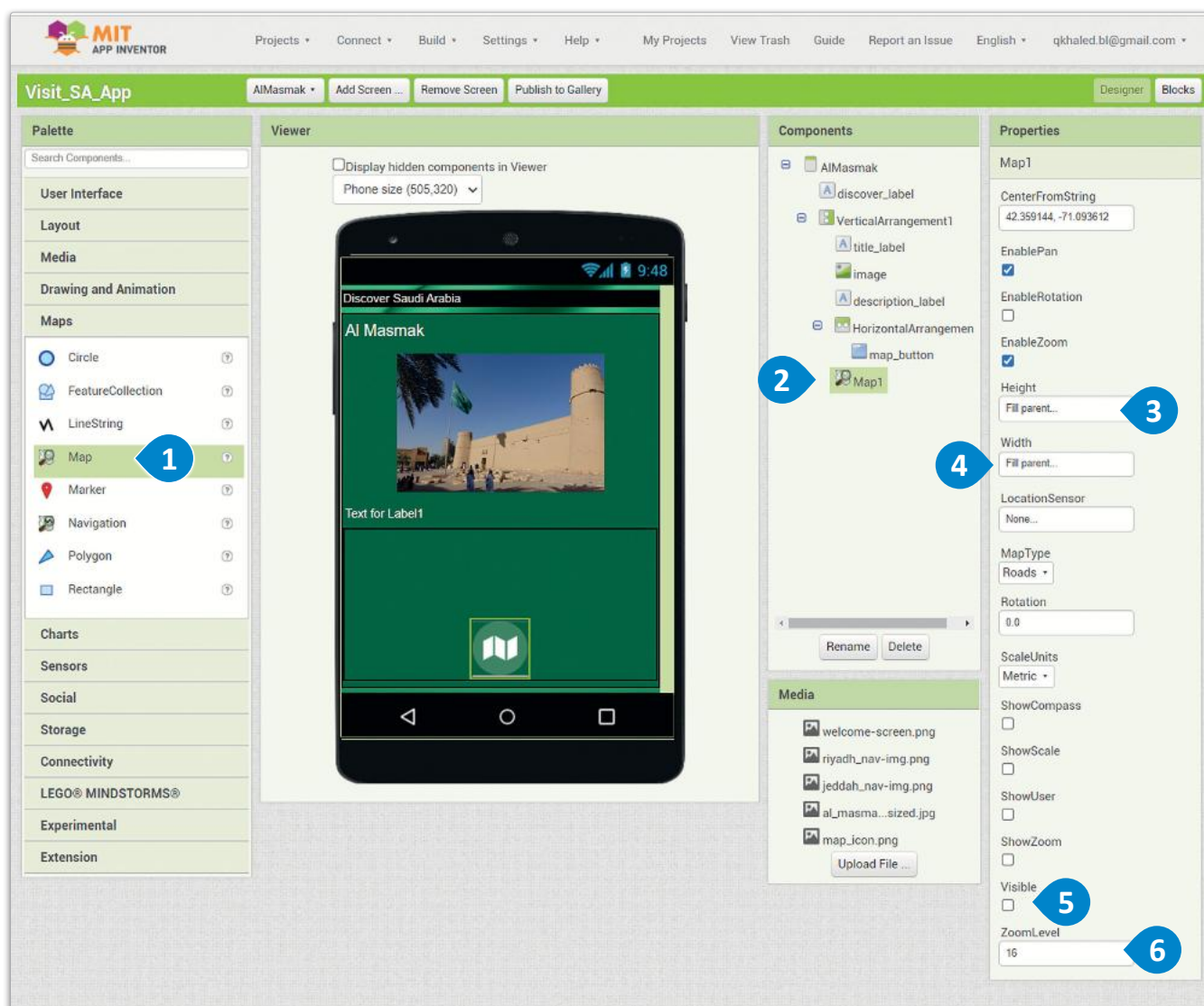


Figure 3.32: Adding a Map component

When you run the finished application on your mobile phone, the map component opens at the location of the selected highlight. In the next lesson, you will add the coordinates depending.

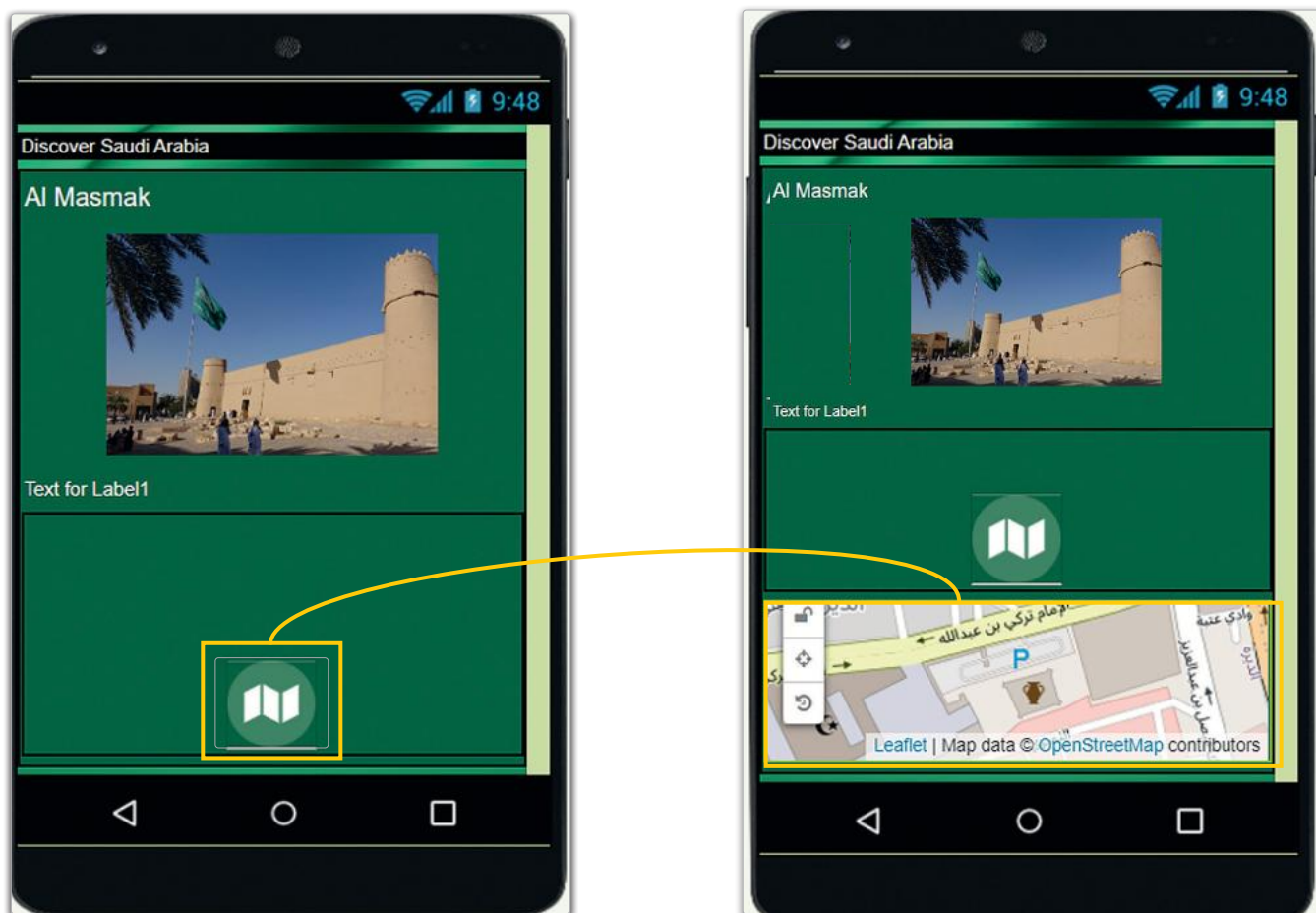


Figure 3.33: Activation of the Map component by the map button

You can interact with the map component as you would with any other complete map application.



Exercises

- 1 Describe how HorizontalArrangement and VerticalArrangement components help you form the layout of a mobile screen.

- 2 State the difference between a ListPicker and a Button component.



- 3 Design an application with a `VerticalArrangement` and two `HorizontalArrangements`. Each `HorizontalArrangement` has two buttons. All of the components are in the center of their container. Use the `Alignment` properties of the appropriate components.

- 4 Design another screen for the above application which has a `HorizontalArrangement` as an external container and two `VerticalArrangements` with buttons inside. All of the components are in the center of their container. Use the `Alignment` properties of the appropriate components.

- 5 Design another screen for the above application which has a `VerticalArrangement` and three rows of `HorizontalArrangements`. Each `HorizontalArrangement` row will have two photos. Each photo will be of a different sport. Make sure that all the components are arranged in the center of their containers and all the photos have the same dimensions.





Programming Applications in App Inventor

Before developing applications with the codeblocks, there are certain concepts and commands that need to be explained, such as working with variable data, and implementing logic and program flow.

Variables in App Inventor

App Inventor allows you to create and interact with variables. Variables can be initialized with multiple data types, like floating point numbers and strings. Variables in App Inventor can have the following scopes:

- **Global:** The variables are accessible by all control and procedure codeblocks.
- **Local:** The variables are accessible only by the procedure that includes them.

Local variables are used to save the memory of the device because they are created and accessed when their procedure needs them.

For the purposes of this project, you will use only global variables because you will not create complex procedures that need local variables.

Lists in App Inventor

Lists are a simple and useful data structure that you can use to implement your application logic. App Inventor also offers methods to interact with and manipulate the data on the lists.

Initializing a global variable

initialize global x to " "

Getting a global variable

get global x

Setting a global variable

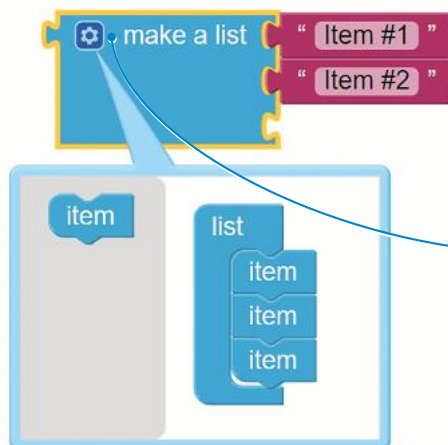
set global x to " test "

Creating an empty list

create empty list

Initializing a list with data

make a list " Item #1 " " Item #2 "



In order to set the number of items of the list, click on the gear icon and drag and drop items to remove and add list items.

The ListPicker Component

The ListPicker allows you to create custom application logic depending on the selection that you have made. When you click on a ListPicker component, the screen changes its appearance to show the list contents. The property of the component that stores the list data is called **Elements**.

Accessing ListPicker elements



A ListPicker will take a variable which contains a list as elements.

Initializing a ListPicker Elements with list Data



Interacting with the ListPicker element selection



The application will open the screen that has the name of the ListPicker selection.

Sending Variables to Another Screen

You want to send the value of the language variable to the next screen in order to initialize the text in the appropriate variable. In App Inventor, when you use a command to open another screen, you can send an initializer value that can be accessed by the next screen.



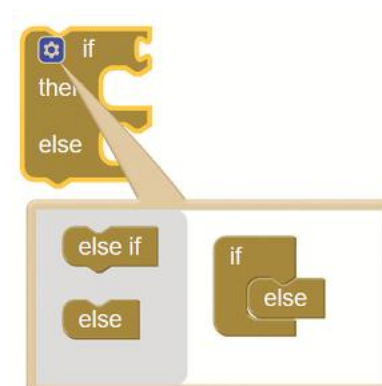
Conditional If Statements in App Inventor

If codeblock statements in App Inventor are constructed similarly to lists. You can add **else if** or **else** statements to the codeblock using the gear icon.

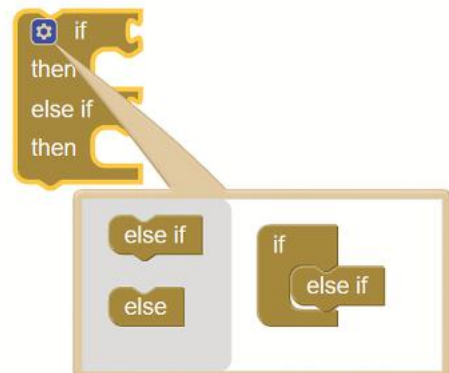
Simple if statement



Adding an else statement



Adding an else if statement



The VerticalScrollArrangement Component

The **VerticalScrollArrangement** component is used to format other components on the vertical axis and make a container for alignment. It also supports scrolling for the components that do not completely fit onto the screen. The alignment of the components inside the container can be changed with the following codeblocks.

```
set VerticalArrangement1 . AlignHorizontal to HorizontalAlignment Left
```

```
set VerticalArrangement1 . AlignVertical to VerticalAlignment Top
```

The HorizontalArrangement Component

The **HorizontalArrangement** component is used to format other components on the horizontal axis and make a container for alignment. The alignment of the components inside the container can be changed with the following codeblocks.

```
set HorizontalArrangement1 . AlignHorizontal to HorizontalAlignment Left
```

```
set HorizontalArrangement1 . AlignVertical to VerticalAlignment Top
```

Programming the Home Screen

The **Home** screen (**Screen1**) will send the user to the **Cities** screen and define the language that will be used on the next screens as well.

Programming Language Support Buttons

You will now program the two language buttons to change the text on the home page and store a variable for the next screen to know in which language to initialize the text. Each page will default to the English language.

To program the two language buttons in the home screen:

- > Select the **when.Click** block for the **discover_button_en** component. **1**
- > Select the Control group and open the screen **Cities** with a **startValue**. **2**
- > Set the **startValue** to **"en"**. **3**
- > Repeat the above step for the **discover_button_ar** and set the startValue to **"ar"**. **4**

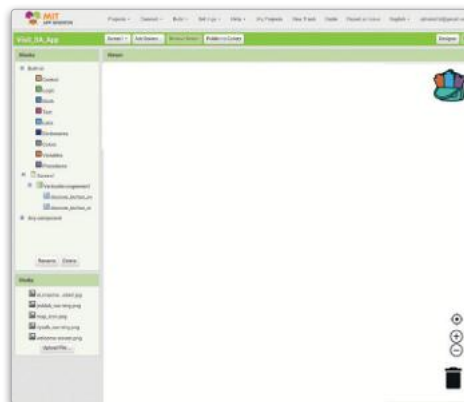
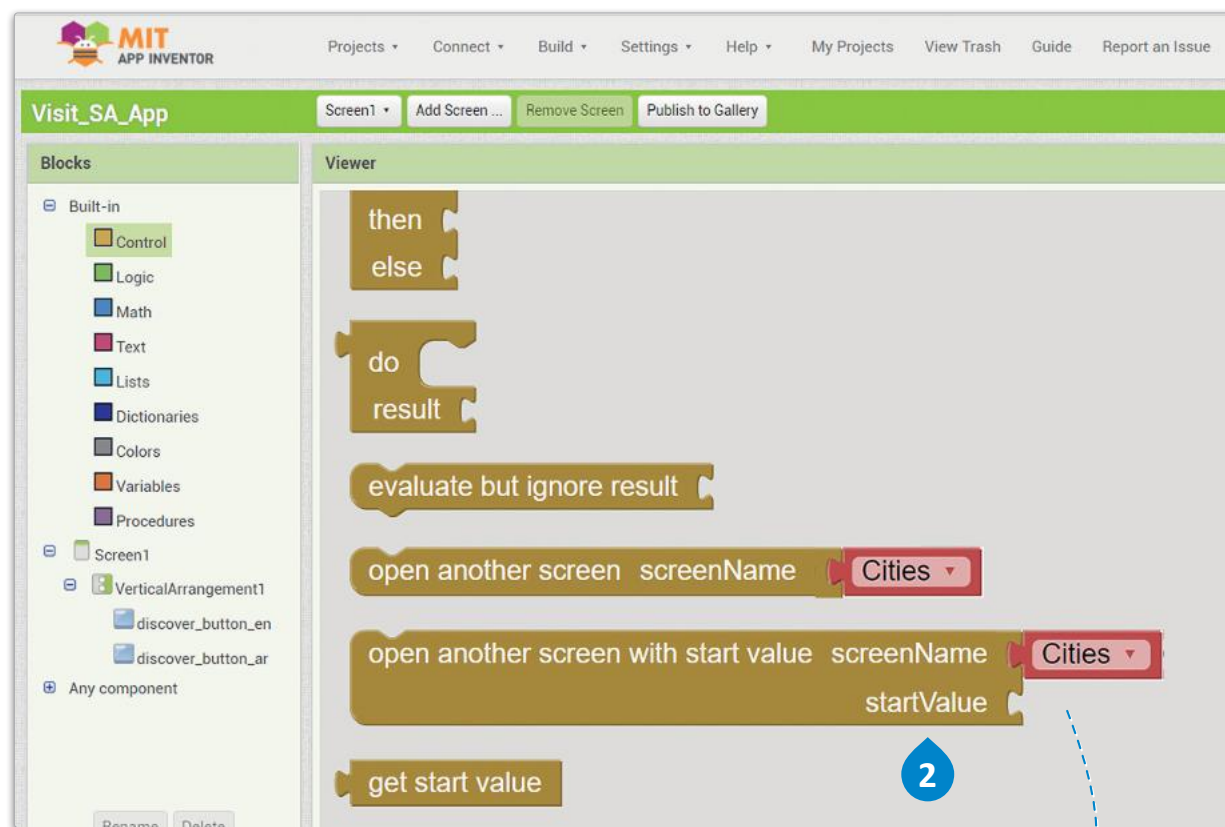
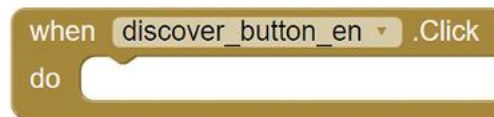
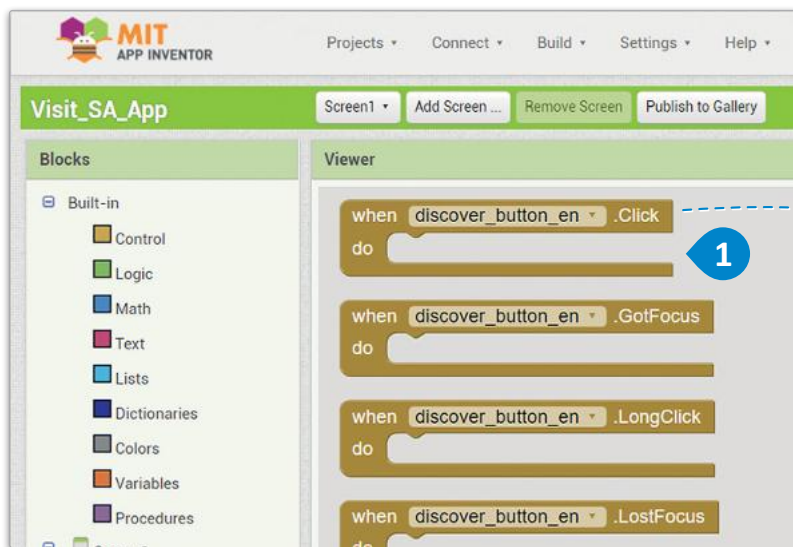


Figure 3.34: Initial blocks page





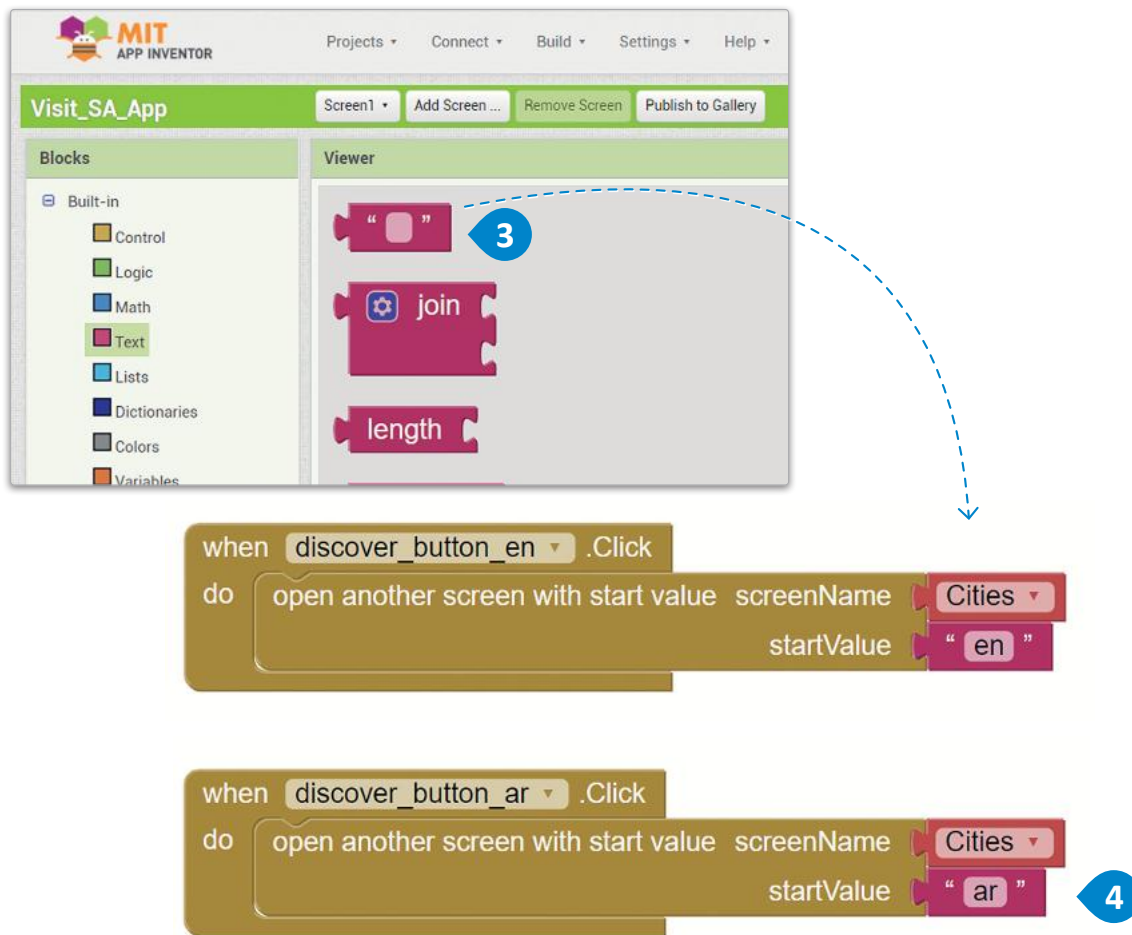


Figure 3.35: Programming the two language buttons in the home screen

The Complete Code for the First Screen (Screen1)

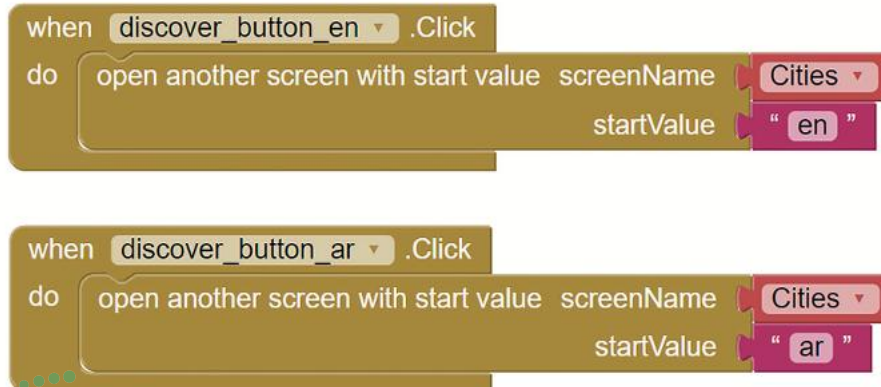


Figure 3.36: Complete code for the first screen

Programming the Cities Screen

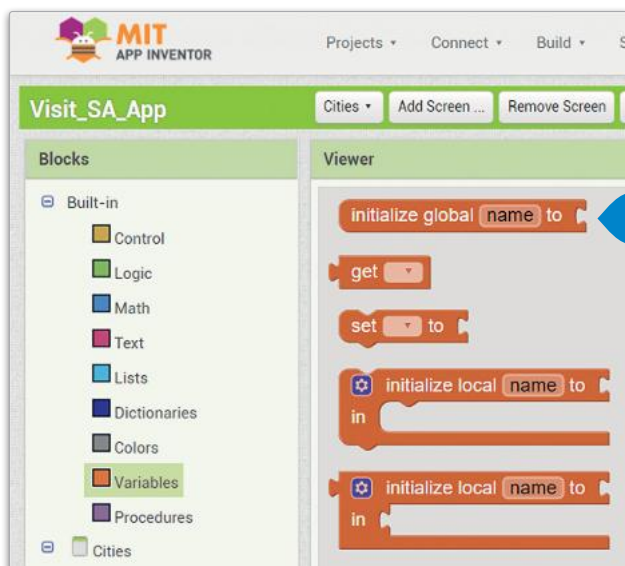
As mentioned earlier, the home screen will send the user to the Cities page and define the language that will be used on the next screens as well.

Creating the Content for the ListPicker

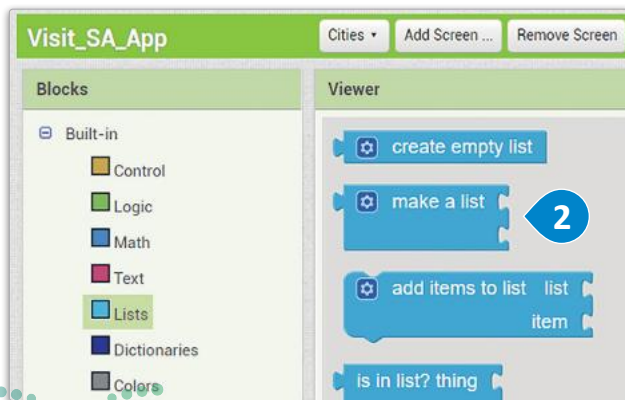
You want to fill the ListPicker Elements with the appropriate text depending on the language. The first step is to define the lists of the highlights for a destination both in English and in Arabic. The second step is to initialize the ListPicker components with the corresponding language.

To create the content lists:

- > Create a new **variable** named **riyadh_highlights_en**. ¹
- > Create a **make a list** block and place it on **the riyadh_highlights_en** variable. ²
- > Fill the list with highlight names in English. ³
- > Repeat the process for a variable named **riyadh_highlights_ar**. ⁴



initialize global **riyadh_highlights_en** to



initialize global **riyadh_highlights_en** to **make a list**

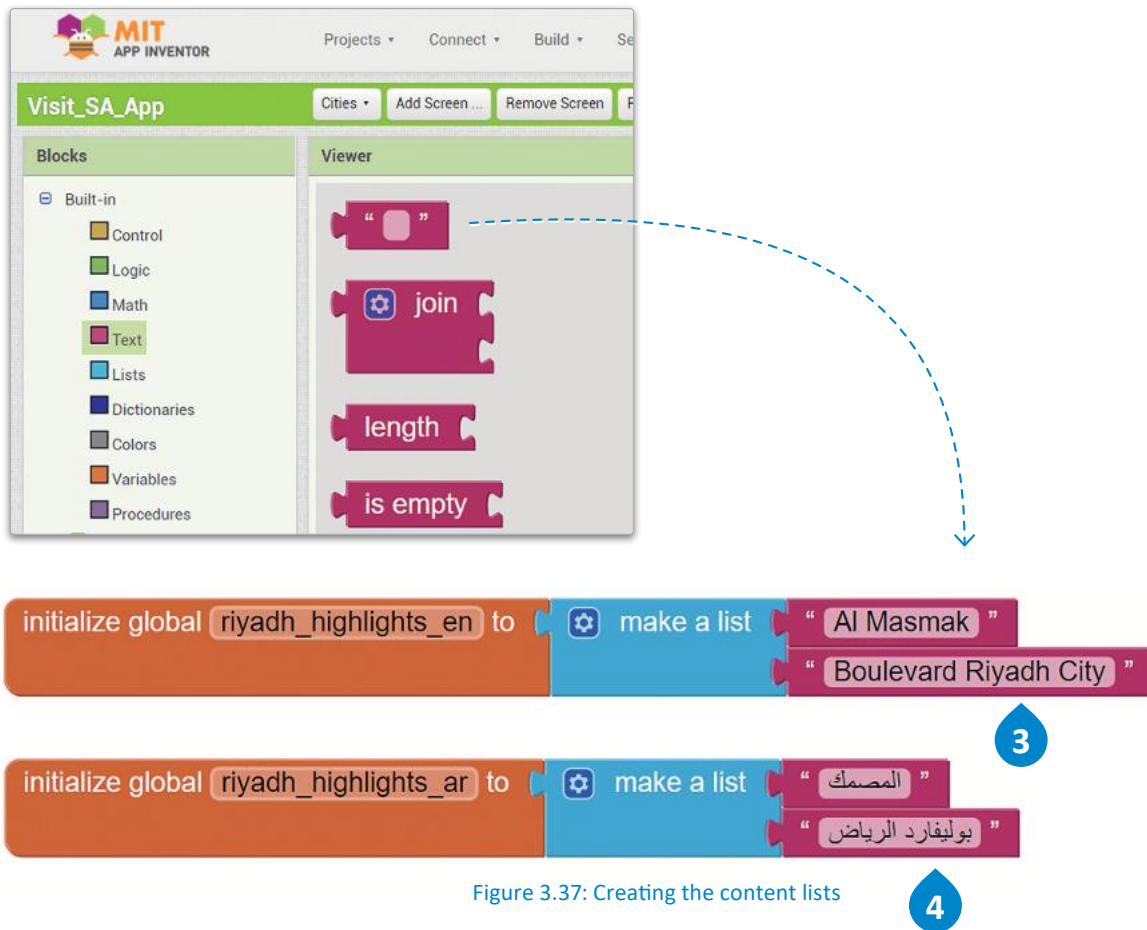
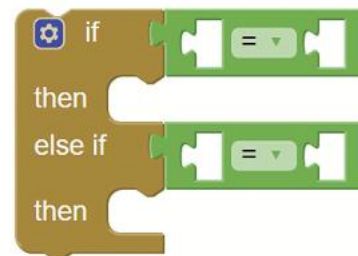
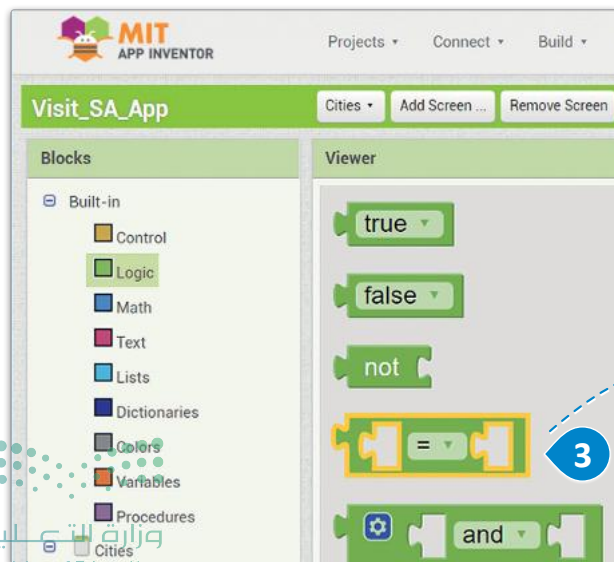
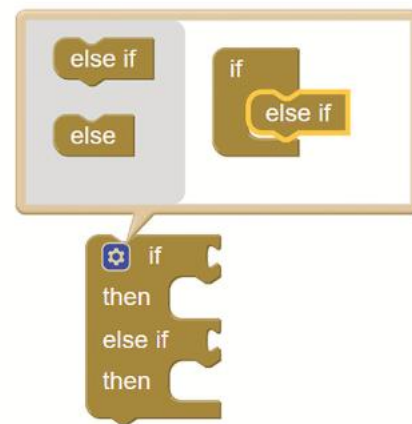
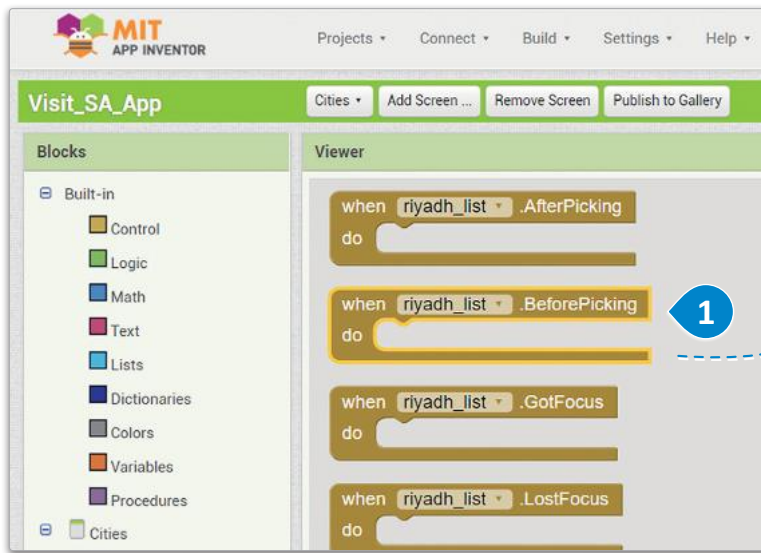


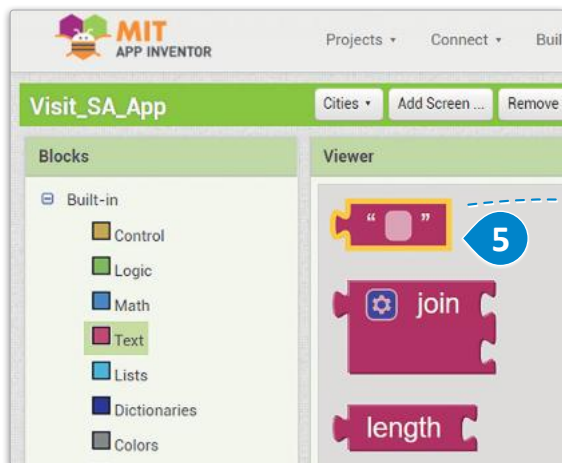
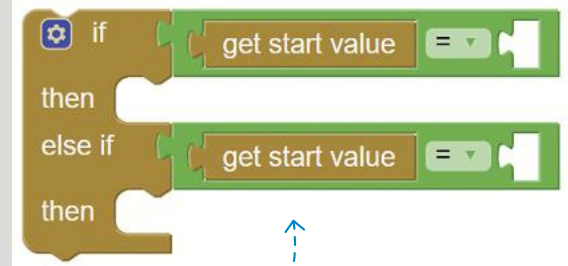
Figure 3.37: Creating the content lists

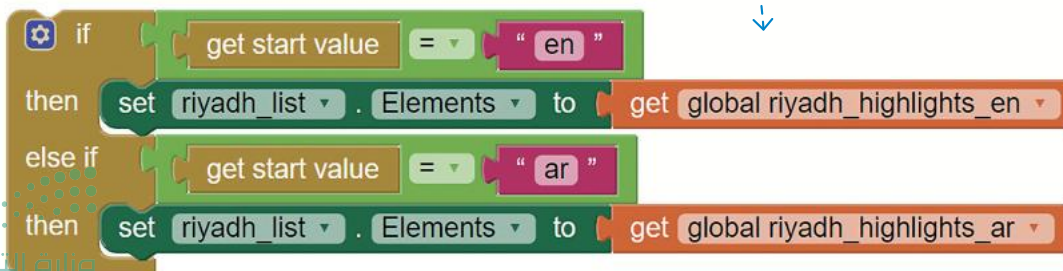
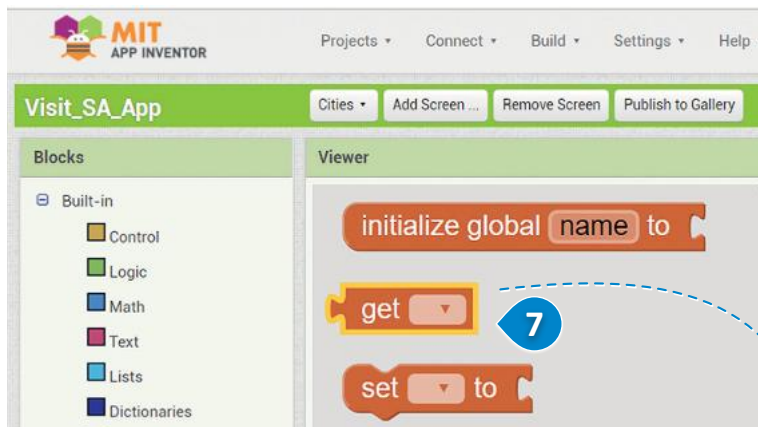
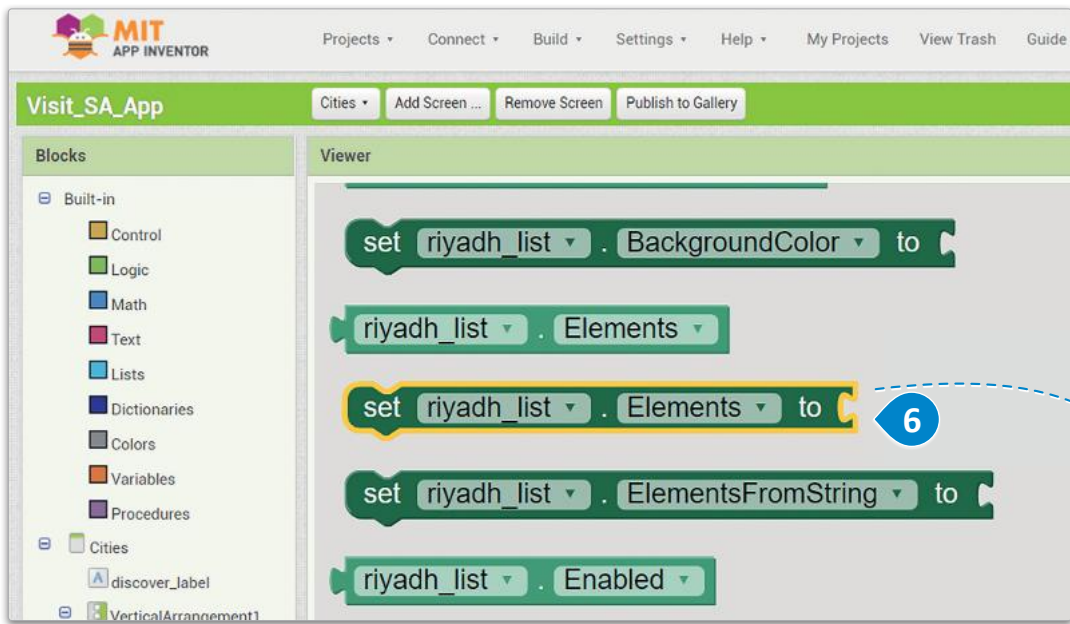
The **if-then** block is used to program the menu item so that when pressed it opens the screen associated with it. If we click on Al Masmak, the corresponding page should open.

To select the content for the list:

- > Select the **BeforePicking** block for the **riyadh_list** component. ①
- > Create an **if else if** statement. ②
- > Add one **equals condition** on the **if** and one on the **else if** statements. ③
- > Add the **get start value** variable on the **left** side of the two **equals** clauses. ④
- > Add the **"en"** in the **right** side of the first **equals** clause and **"ar"** in the **right** side of the second **equals** clause. ⑤
- > Select the **Set Elements to** command of the **riyadh_list**. ⑥
- > Add the appropriate **list variables** for the above command. ⑦
- > Add the **if else if** codeblock to the **BeforePicking** event. ⑧







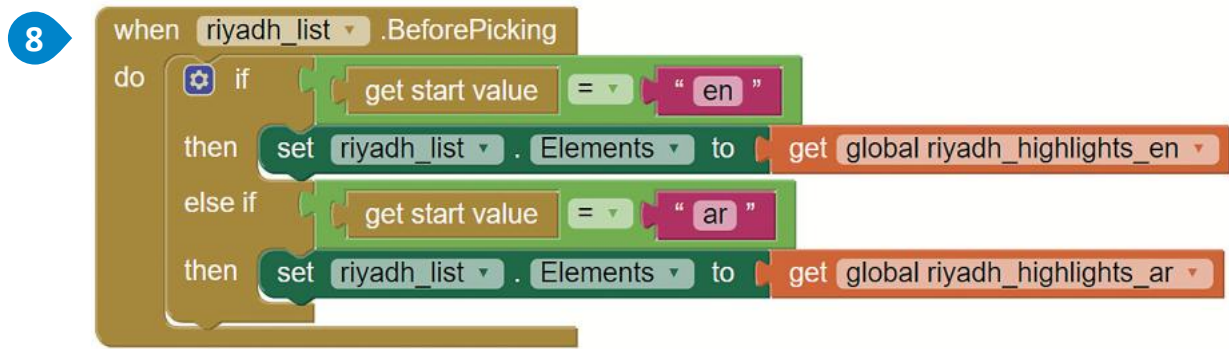


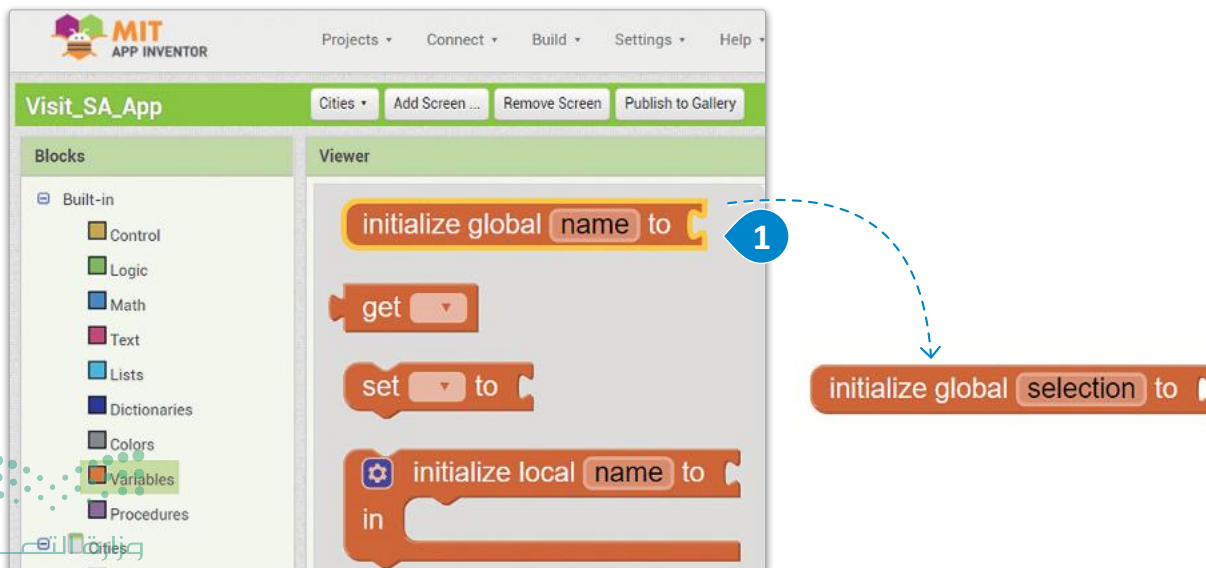
Figure 3.38: Initializing the ListPicker content

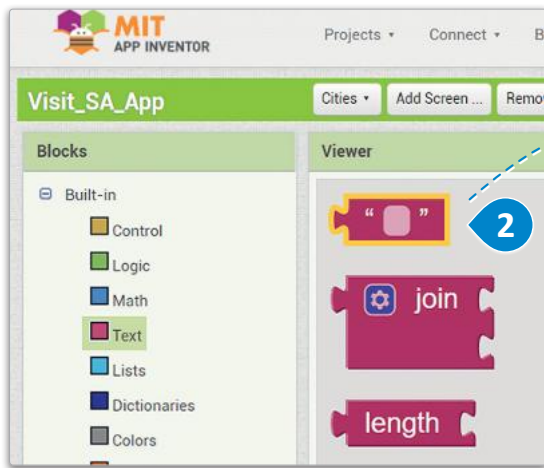
Opening a New Page from the ListPicker

You have the list of highlights from the city you choose and you want to open a specific highlight. Each highlight will have its unique page. Each selection from the ListPicker will detect the highlight and then open it. The page for each highlight will also support English and Arabic language.

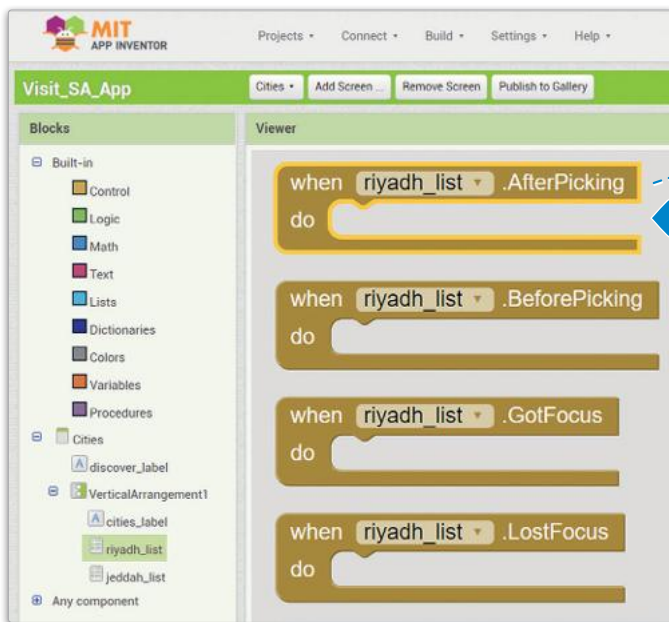
To get the new highlight page name from the ListPicker selection:

- > Create a new **variable** named **selection**. ①
- > Add an **empty string** on the **selection** variable. ②
- > Select the **AfterPicking** event for the **riyadh_list** component. ③
- > Add the **selection** variable inside the **AfterPicking** event. ④
- > Set the **selection** variable to the **Selection** property of the **riyadh_list**. ⑤
- > Add an **if** codeblock inside the **AfterPicking** event. ⑥

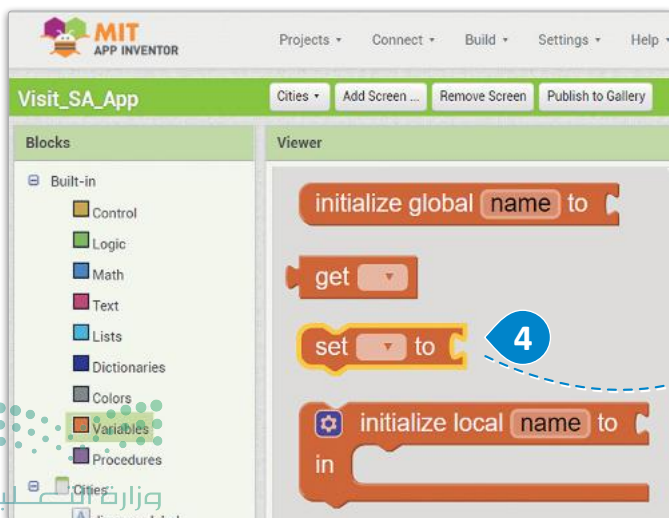




initialize global selection to " "



when riyadh_list .AfterPicking do



when riyadh_list .AfterPicking do set global selection to

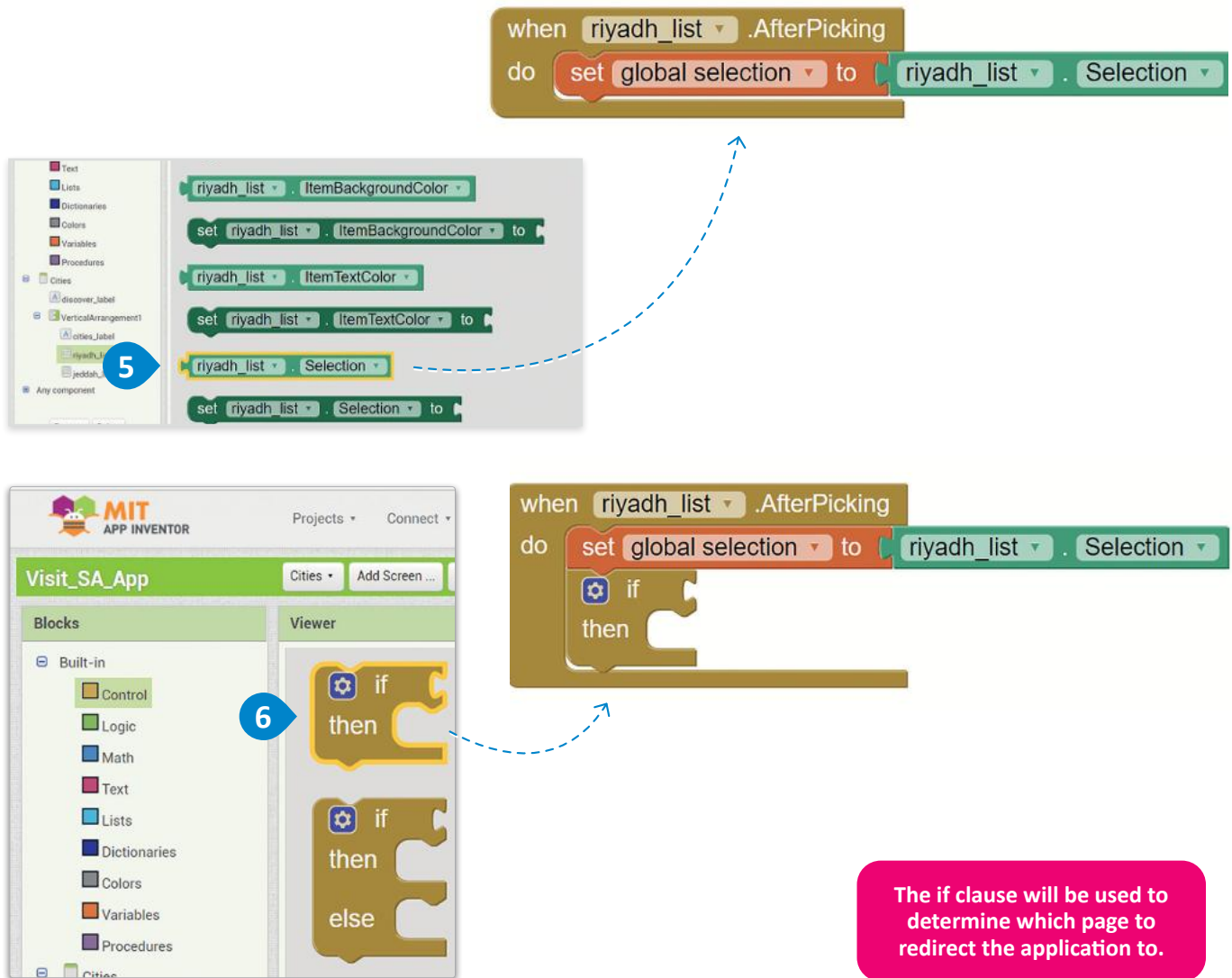


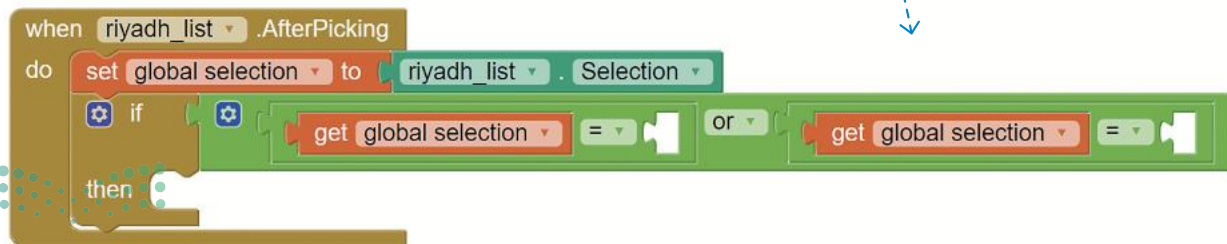
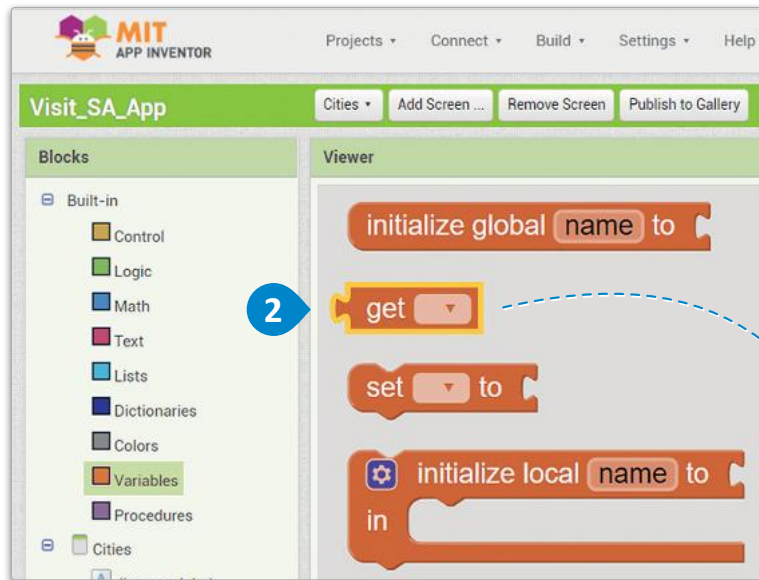
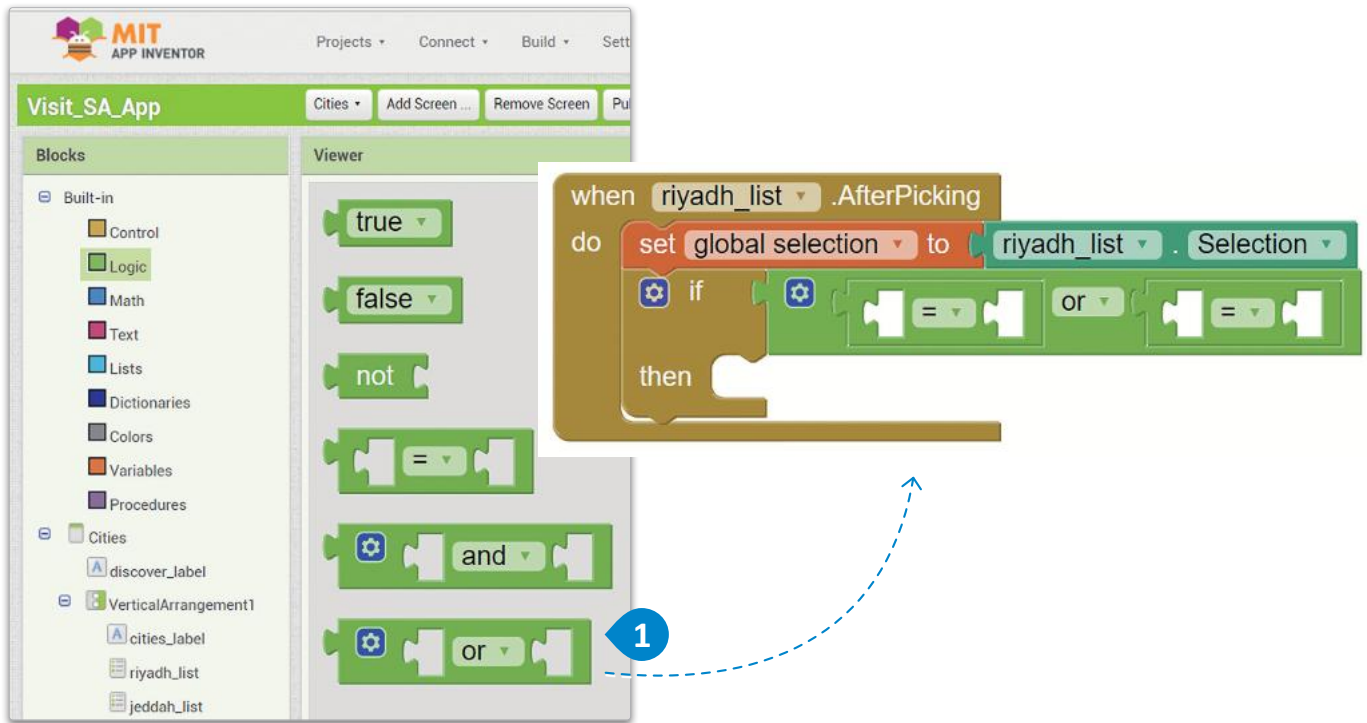
Figure 3.39: Selecting the page from the ListPicker

Opening the Appropriate Page for the ListPicker Selection

A new page for a highlight will be displayed depending on the selection of the ListPicker, and the codeblock will recognize the destination whether it is in English or Arabic. The codeblock will also send language configuration to the screen as well.

To open the appropriate page from the ListPicker selection:

- > Add an **or** clause that contains two more **equals** clauses and put them in the **if statement** inside the **AfterPicking** event. **1**
- > Add the **selection** variable to the **left** side of the **equals** clauses. **2**
- > Add the **empty strings** of the **AI Masmak** highlight both in **English** and in **Arabic** on the **right** side of the **equals** clauses. **3**
- > Open the **AI Masmak** screen with the **startValue** being the one you used in the **previous** screen. **4**



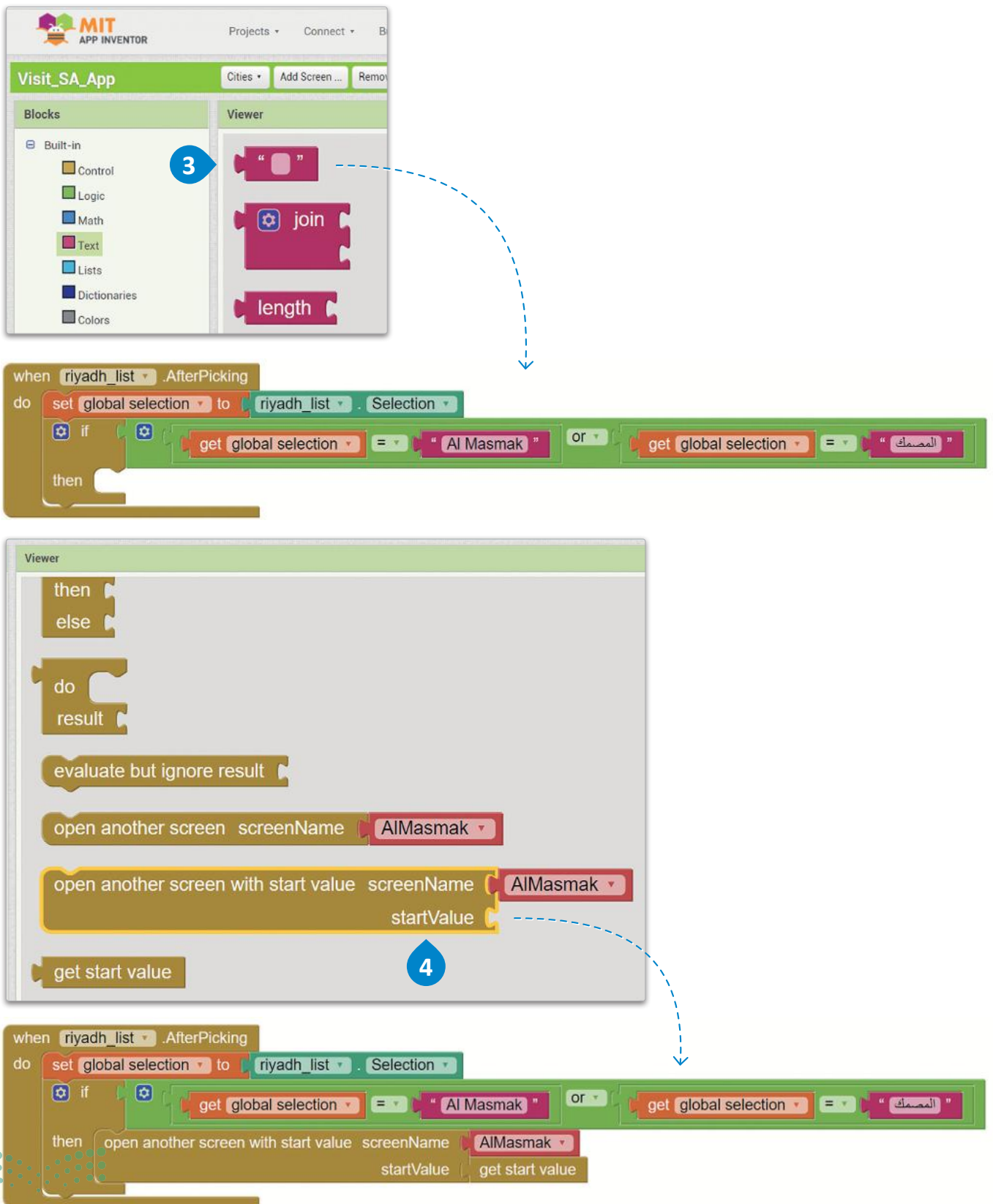


Figure 3.40: Opening the appropriate page from the ListPicker selection

The Complete Code for the Second Screen (Cities)

```

initialize global selection to " "

initialize global riyadh_highlights_en to
  make a list
  " Al Masmak "
  " Boulevard Riyadh City "

initialize global riyadh_highlights_ar to
  make a list
  " المصمك "
  " بوليفارد الرياض "

when riyadh_list .BeforePicking
do
  if
    get start value = " en "
  then
    set riyadh_list . Elements to get global riyadh_highlights_en
  else if
    get start value = " ar "
  then
    set riyadh_list . Elements to get global riyadh_highlights_ar

when riyadh_list .AfterPicking
do
  set global selection to riyadh_list . Selection
  if
    get global selection = " Al Masmak " or get global selection = " المصمك "
  then
    open another screen with start value
    screenName AlMasmak
    startValue get start value
  
```

Figure 3.41: Complete code for the second screen

Programming the Highlight Screen (Al Masmak)

A highlight screen will change the language and the formatting of the text depending on the selection of the home page and will also offer the option to display an interactive map with the location of the highlight.

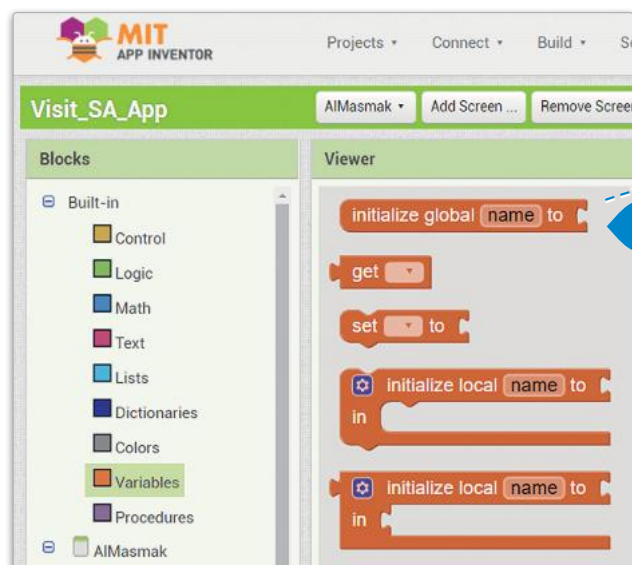
Dynamically Changing the Language for the Highlight Page

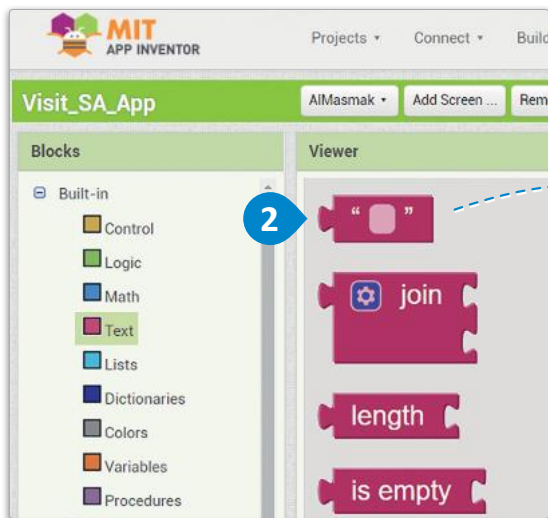
The user will be directed to a page that contains text and an image for the selected highlight. The text language will change according to the selected language. The text will also need to be aligned correctly depending on the original language selection on the home page. If the language selection is English, the text will need to be aligned to the left, and if it is Arabic, the text will need to be aligned to the right.



To change the language dynamically:

- > Create variables for the label texts. **1**
- > Add **empty string** for the **title** and **description** labels for the **AI Masmak** highlight both in **English** and in **Arabic**. **2**
- > Select the **Initialize** event for the **AI Masmak** screen. **3**
- > Add an **if else if** codeblock inside the **Initialize** event. **4**
- > Add an **equals** clause inside each **if** statement. **5**
- > Add the **get start value** on the **left** side of each of the two equals clause. **6**
- > Add the **"en"** in the **right** side of the first **equals** clause and **"ar"** in the **right** side of the second **equals** clause. **7**
- > Set the **Text** property of the **title_label** to the **title variable** of the appropriate language. **8**
- > Set the **Text** property of the **description_label** to the **description variable** of the appropriate language. **9**
- > Set the **AlignHorizontal** property of **VerticalArrangement1** to the **text direction** of the appropriate language. **10**



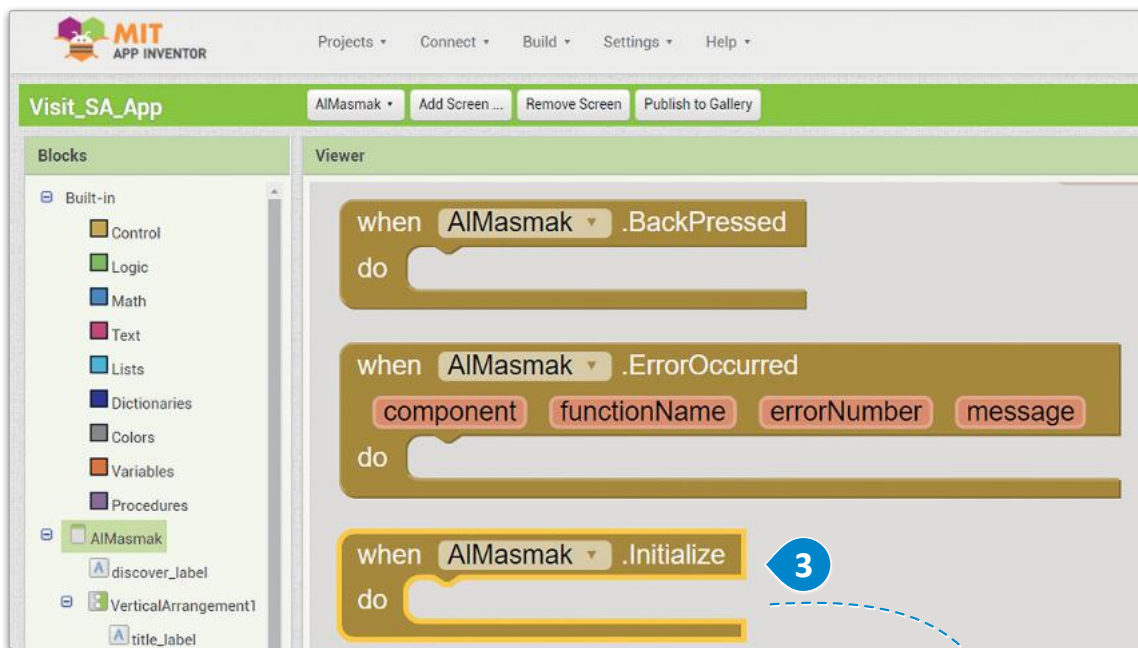


Search the Internet for descriptions of the Al Masmak castle both in English and in Arabic.

```

initialize global title_en to " Al Masmak "
initialize global title_ar to " المصمك "

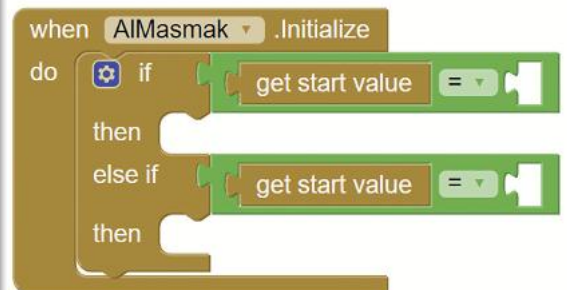
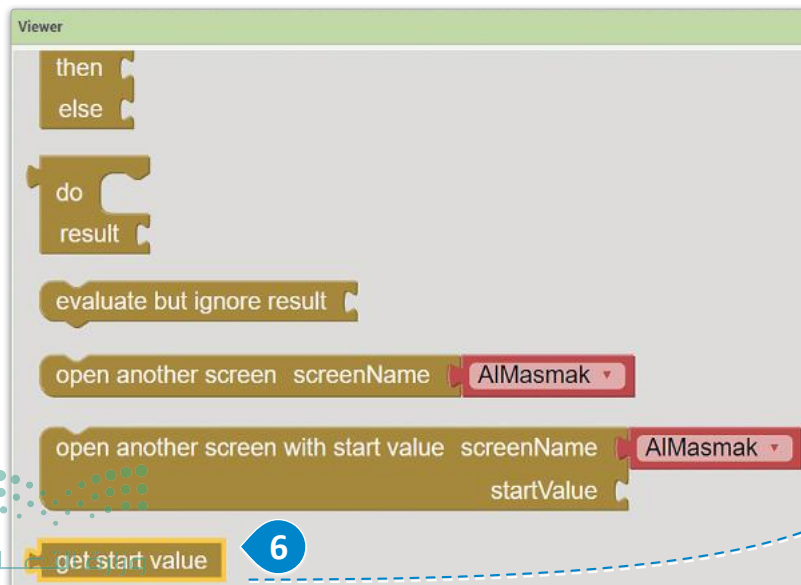
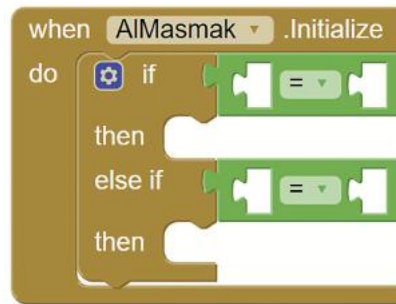
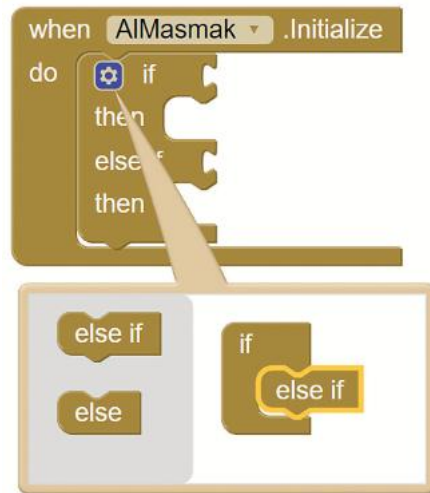
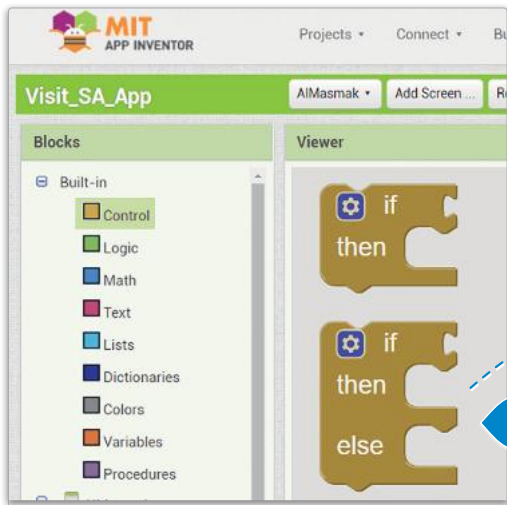
initialize global description_en to " Located in the heart of the old quarter in Riyadh... "
initialize global description_ar to " ...يقع قصر المصمك في وسط مدينة الرياض، وهو عبارة عن "
  
```

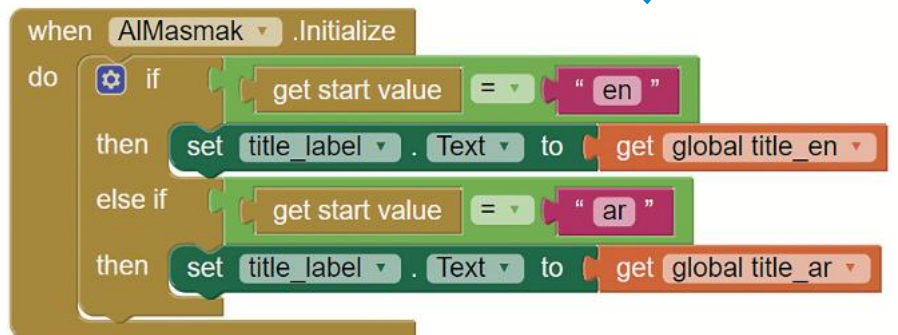
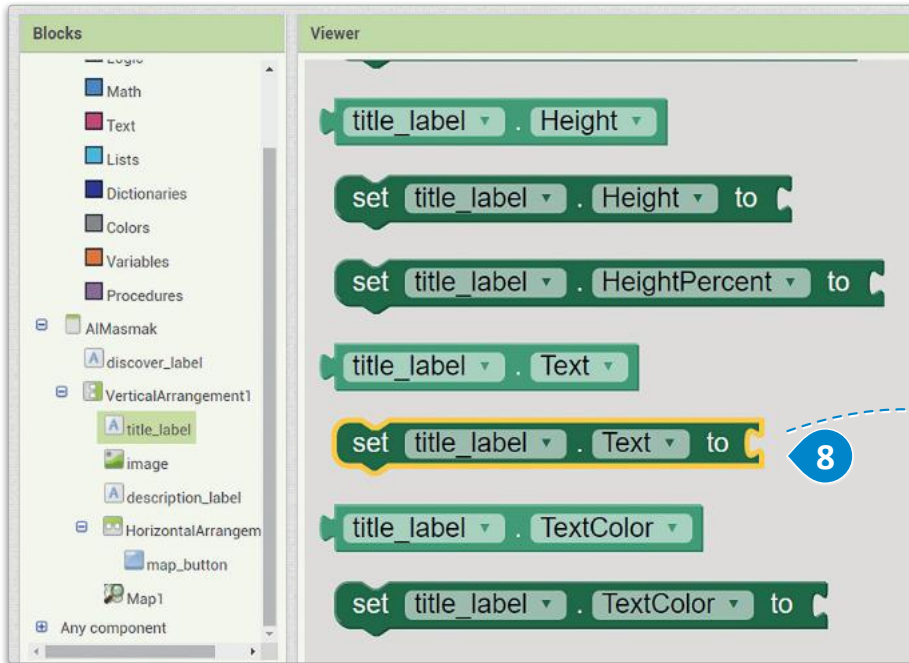
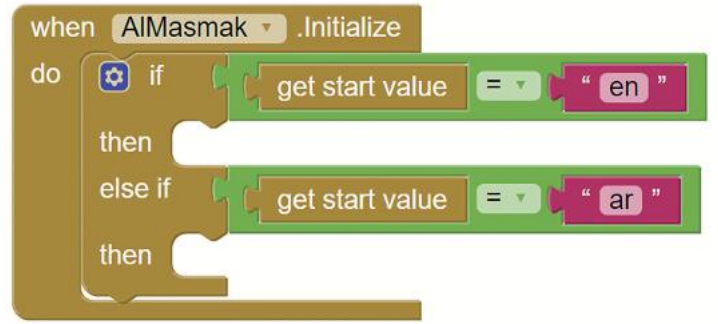


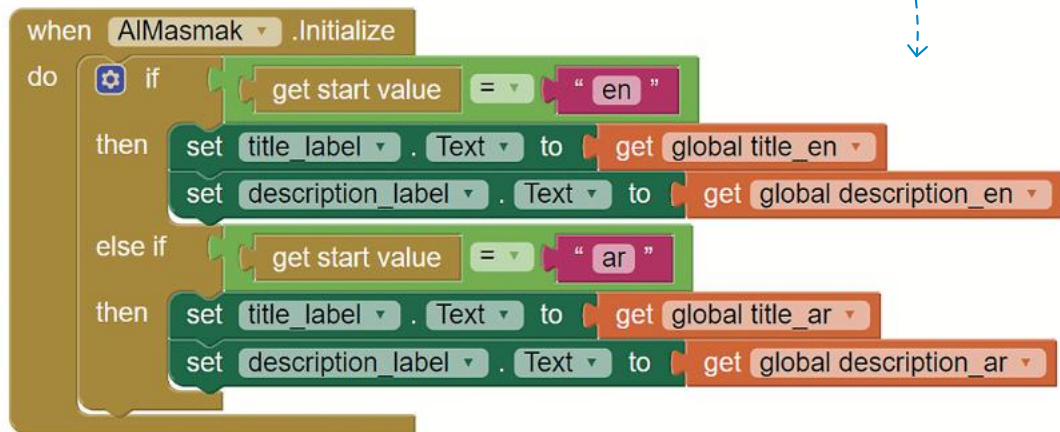
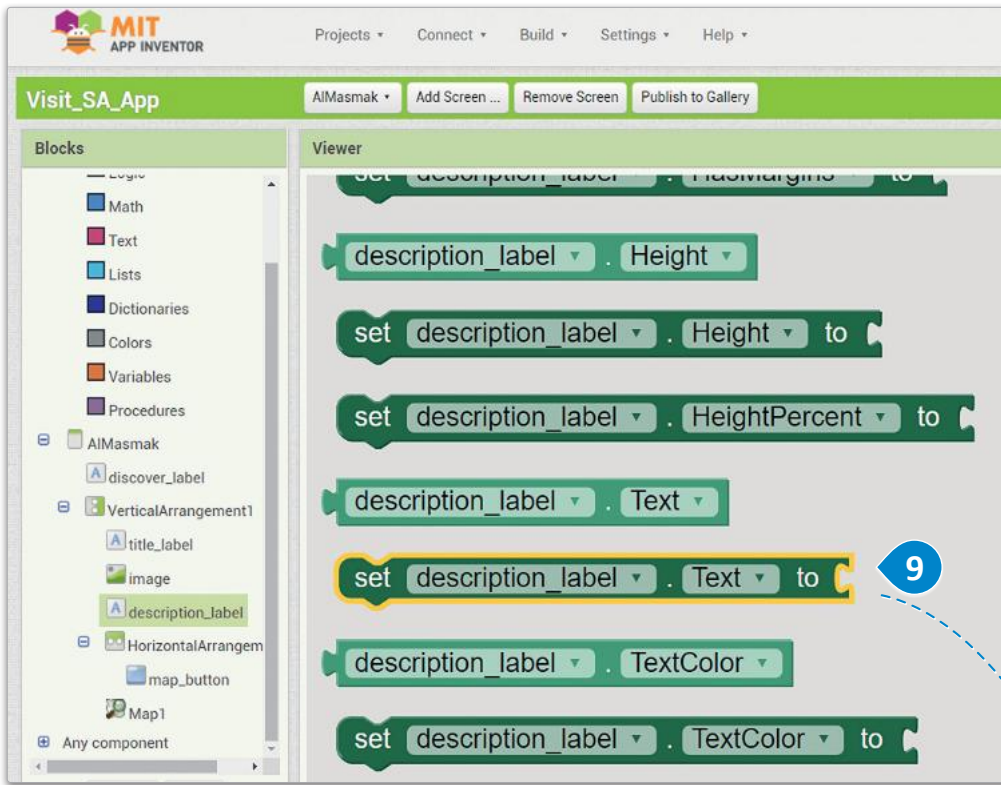
```

when AlMasmak .Initialize
do
  
```









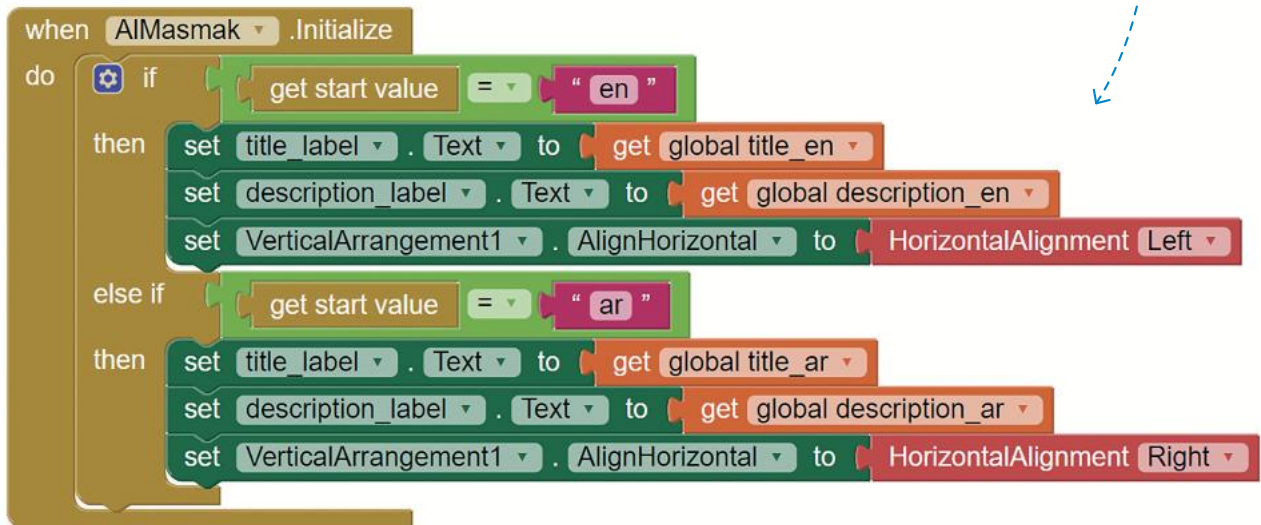
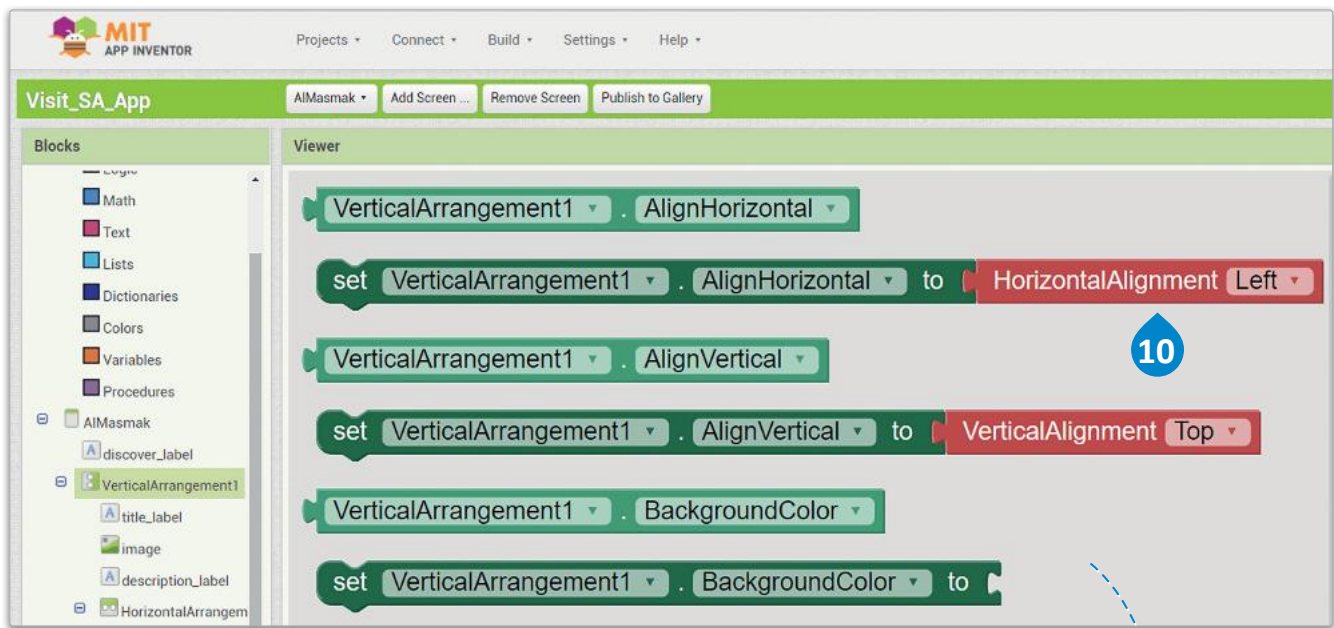


Figure 3.42: Changing the language dynamically

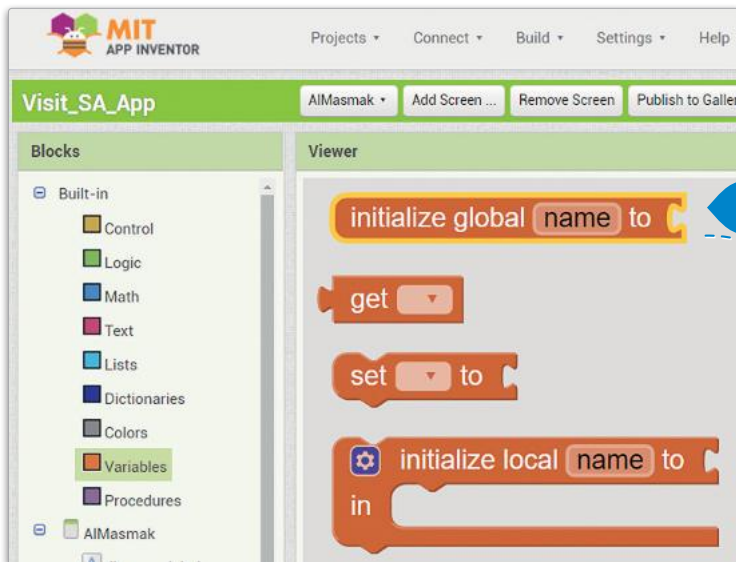


Programming the Interactive Map

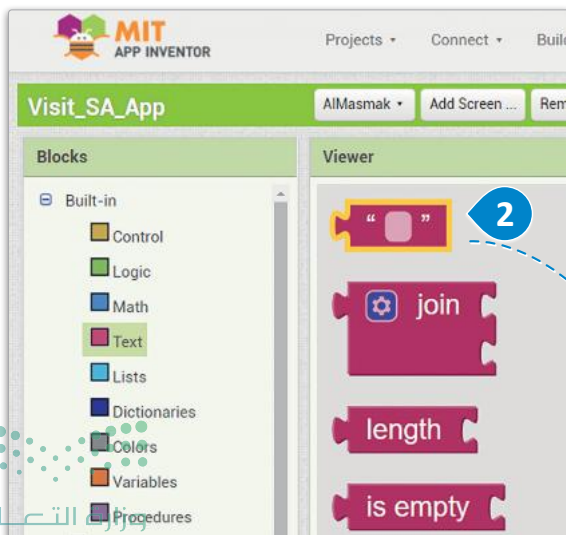
The application will open the interactive map when the user presses the map button. The map will be initialized to the coordinates of the respective highlight.

To program the interactive map:

- > Add a new variable named **coordinates**. ¹
- > Add the following text to the **coordinates** variable: **24.6312, 46.7134**. ²
- > Select the **Click** event for the **map_button** component. ³
- > Set the **CenterFromString** property of the **map** component to the **coordinates** variable. ⁴
- > Set the **Visible** property of the **map** component to a **true** codeblock. ⁵



initialize global **coordinates** to



initialize global **coordinates** to "24.6312, 46.7134"

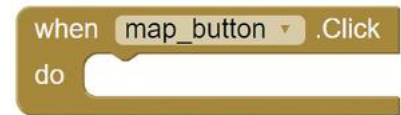
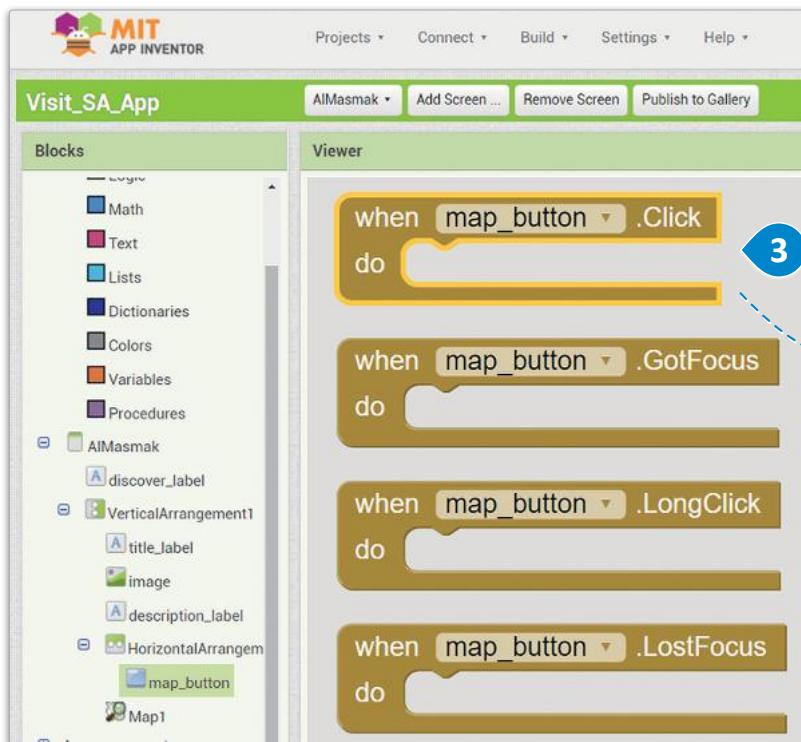




Figure 3.43: Programming the interactive map

The Complete Code for the Third Screen (Al Masmak)

```

initialize global title_en to " Al Masmak "
initialize global description_en to " Located in the heart of the old quarter in Riyad... "
initialize global title_ar to " المصمك "
initialize global description_ar to " يقع قصر المصمك في وسط مدينة الرياض، وهو عبارة عن ... "
initialize global coordinates to " 24.6312, 46.7134 "

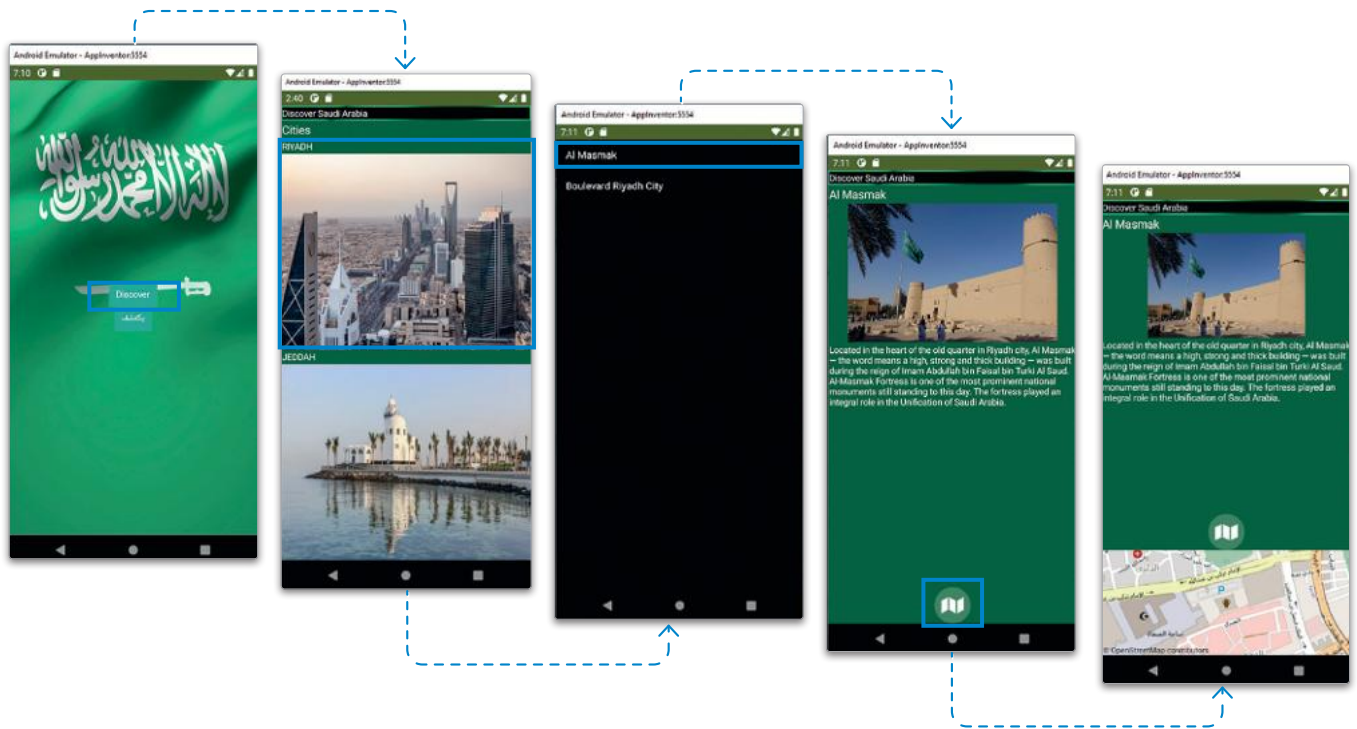
when AIMasmak.Initialize
do
  if
    get start value = " en "
  then
    set title_label.Text to get global title_en
    set description_label.Text to get global description_en
    set VerticalArrangement1.AlignHorizontal to HorizontalAlignment Left
  else if
    get start value = " ar "
  then
    set title_label.Text to get global title_ar
    set description_label.Text to get global description_ar
    set VerticalArrangement1.AlignHorizontal to HorizontalAlignment Right

when map_button.Click
do
  set Map1.CenterFromString to get global coordinates
  set Map1.Visible to true
  
```

Figure 3.44: Complete code for the third screen

The software is ready and you have to test it. For this you can use the Android Emulator, or you can download a package format for deployment and run it on your Android device. You can also scan the QR code with the Android device to preview. In the pictures below, the screens of your application are shown when you run the program using the Emulator.





Here are the screens shown when pressing the Arabic button:

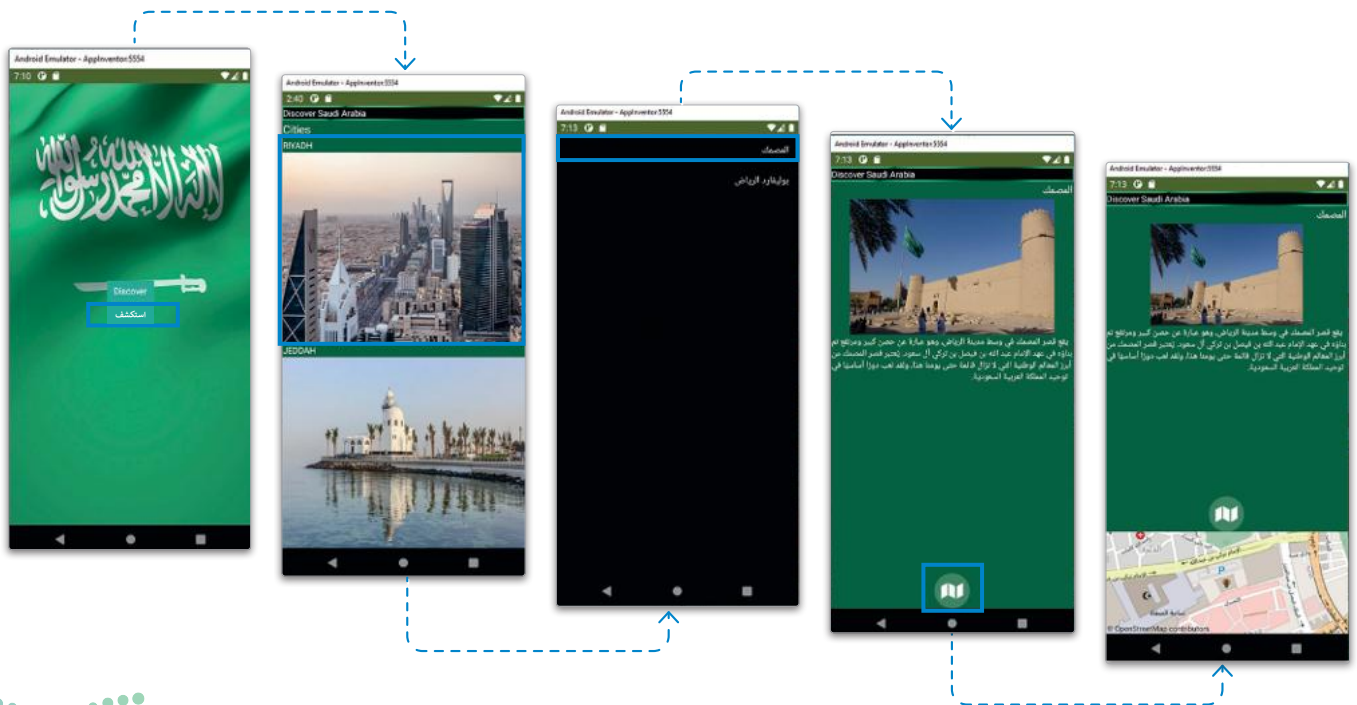


Figure 3.45: Testing the application in the Emulator

Exercises

- 1

[illegible]

- 2



- 3 Add another row with two buttons that change the language between English and Arabic and modify each page's code to accommodate this feature.

- 4 On a highlight page, add a new text label that will display the coordinates of the interactive map component. You can find the coordinates properties by clicking the map component on the blocks page.

- 5 On a highlight page, add two new buttons that will change the type of the interactive map between Aerial and Road views. You can find the map type properties by clicking the map component on the blocks page.



Project

1

In previous Units, you started a project about Vision 2030. You have already created a prototype with Pencil Project. Now you will build your app.

2

Create an application in App Inventor that will present Vision 2030 megacities to the user. These cities will be Amaala, NEOM and Qiddiya.

3

The user will be able to choose a megacity and then be presented with images and descriptions of each project.

4

Develop the app with the appropriate layout and navigation controls.



Wrap up

Now you have learned to:

- > Design an application UI with a wireframe guide.
- > Develop a functional and interactive UX application for users.
- > Utilize feedback to iteratively improve an application.
- > Program complex logic for an application.
- > Compile content for an application and display it appropriately.

KEY TERMS

Blocks	Event	ListPicker
Coordinates	Event handler	Variables
Emulator	HorizontalArrangement	VerticalArrangement



4. Software Accessibility and Digital Inclusion

In this unit, you will learn about the process of testing applications and then you will deploy and test the application you built in the previous unit. Then you will understand the digital divide problem and how digital inclusion can be used to solve it. Finally, you will develop accessibility features for the mobile application you built in order to make it more inclusive for users with disabilities.

Learning Objectives

In this unit, you will learn to:

- > Explain the various stages and methodologies of testing software.
- > Deploy and test mobile applications in MIT App Inventor.
- > Define the problems created by the digital divide.
- > Distinguish the solutions for fostering digital inclusion.
- > Analyze the various accessibility features for both hardware and software.
- > Enhance mobile applications with accessibility features.

Tools

- > MIT App Inventor
- > Pencil Project

Lesson 1

Testing and deploying applications

Link to digital lesson



www.ien.edu.sa

The Importance of Application Testing

When you complete developing an application, you may feel confident that everything has worked as you planned. However, mistakes can happen, and the initial application may not achieve the desired results, so you must verify that the program works as expected without errors or defects. The purpose of testing is to show that the program works correctly and to find previously undiscovered errors or errors related to incorrect application usage.

What is Software Quality

Software quality is the study of a software product to check whether it meets the user specifications, if it is usable and functional, if it has few errors and if it can be properly maintained and improved. The following table illustrates the main quality attributes of a software product.

Table 4.1: Main software quality attributes

Attribute	Description
Functionality	A software product is functional when it meets the end-user requirements and can accomplish all the specified tasks.
Reliability	A software product is reliable when it is not prone to errors and can perform well under restricted resources.
Usability	A software product is more usable if different users can easily access its capabilities.
Efficiency	A software product is efficient when it does not waste resources such as processing power, memory or network capacity.
Maintainability	A software product is maintainable if bugs can be easily fixed as they appear and new features can be easily implemented.
Portability	A software product is portable if it can work under other operating systems, on many devices, with other software, etc.

Difference between Application Debugging and Testing

Debugging is the process of removing bugs and malfunctions from the software, which occurs after testing. Testing is the process of validating the software. For example, during testing, it may occur that a specific component in the UI of an application does not render the correct information to the user. During debugging, it is found that a false calculation in program logic causes the error in the rendering, and it is corrected. Therefore, proper testing is needed to proceed to the debugging phase.

Who Performs Tests

The programmer or developer tests the code they write most often. But the programmer who wrote the code is not the best option. Sometimes he may not be able to see the mistakes he made. For this reason, we need another person called a tester to do the testing focusing on the program's functionality and testing the results by entering different input data sets, for example. Programmers do the initial tests, but the tester judges the software's quality and that it works as expected. Some software development companies include a testing department, usually called the Quality Assurance Department, which specializes in validating the correct behavior of the program.

A test does not guarantee that the program is 100% correct, but it does reveal potential errors.

Choosing Test Data

The best way to test a program is to manually compute its output before running it and see if the results match what you calculated or not. In other words, write the expected outcome of the program before running it and compare the running result with the real output of the program.

Whatever data the user enters, the program must function correctly. If invalid data is entered, the program must state that the data is not acceptable and request a re-enter. The developer needs to consider all the possible values of the entered data. To properly test the program, we need to choose test data representing all possibilities of user input. Test data is divided into the following categories:

- **Normal data:** The data is usually used when dealing with the program by the user, and it includes a set of values of the same type as the expected data. For example, if you have to enter a month as an integer from 1 to 12, then the normal data is an integer from 1 to 12.
- **Boundary data:** It is data located on the outliers of the range of expected values. For example, if you expect to enter a year between 1900 to 2020, then the outliers are 1900 and 2020, and so you are testing the program when you enter 1900 or 2020 as numbers into the program to see if any errors happen.
- **Erroneous data:** Data that is outside the range of expected values or is of an incorrect data type. In the previous example, if the user entered 0 or 13 as the month, or entered the word January instead of the integer 1, there would be an error.

Automated Testing

Often the software is large and complex, with many updates and modifications that may change its functionality after its initial release. In this case, the testing procedure must perform the same tests and some new ones to verify that the software is working properly. But if the tests are many, it will take a lot of time and effort to do it manually. The tester can create a set of automatic tests updated each time the program is changed. The tester writes their code to test the program through several

tools to automate this process. For example, some of the most popular testing frameworks for Android are **Appium**, **Espresso** and **UI Automator**.

Testing Strategies

Testing is divided into several categories depending on the complexity of the code or application being tested. Most testing time is consumed on programming functions and the main program at the lowest level. Developers, both small and large, use many different testing strategies. The following table illustrates the most common testing strategies.

Table 4.2: Common testing strategies

Name	Description
Dry Run Testing	In this method, you follow the logic of the program as the computer executes each statement in the code and records the value of each variable in the tracking table.
Usability Testing	Usability or user experience (UX) tests are conducted to ensure that the software is easy and understandable to the end user.
Black-Box Testing	Black box tests consider the parts of the program that you test as a closed box, so the code is ignored, and the input and output data are dealt with only to see if the tester gets the expected results when entering the data or not.
White-Box Testing	In white-box testing, the tester has access to the code, so the testing process is focused on verifying the correctness of the code implementation. This includes testing the code logic, data structures, algorithms, error handling, and boundary conditions.
Unit Testing	Unit testing is testing each program's function separately to ensure that each process works before fully verifying that the program works.
Integration Testing	Integration tests check how different parts of a program behave when they work together as an integrated system.
Performance Testing	Performance tests check the performance of a program or system under data loads or increased user numbers. Issues that need to be fixed to ensure scalability will emerge.
Acceptance Testing	Acceptance tests will show whether the product meets all the requirements of users with different needs, and this test is suitable for large multi-user software.
Penetration Testing	Penetration tests focus on the security of a program or system. It checks how to protect the program from attacks and intrusions.
Stress Testing	Stress testing is a testing technique that focuses on evaluating the performance of a software system under extreme conditions. The goal of stress testing is to identify the breaking point of the system and to determine how it behaves when it is under maximum stress.

Test Planning

A test plan or test schedule is a list of tests planned to be carried out to check the accuracy of the program and then to record the results of each test.

- A table that includes test data, the purpose of each test, expected results, and actual results when the program is run. Each row in this test table is called a test case.

- A test scenario validates a specific part of a program's functionality and may contain a set of test cases.
- Strictly defined acceptance criteria in each test scenario.

The test case should be distinct from the use case. As we saw earlier, the use case defines how the program or system is used to perform a specific task. It is usually a diagram showing the sequence of actions the user will follow when interacting with the program. The goal of testing is intentionally creating error states with valid and invalid data. Test scenarios and cases are often planned before the actual programming is done.

Test Documentation

The testing process needs to be carefully documented to benefit from it in the tests of the following versions, and the testing documents include the following:

- **Testing Policy** - Describe the principles, methods, and objectives of testing.
- **Test Plan** - Description of the software, its functions, parts to be tested, and scope of tests.
- **Test Specifications** - Details of each test scenario and its evaluation criteria.
- **Test Description** - Test data and procedures for each test case.
- **Test Analysis Report** - The results of each test scenario.
- **Bug Report** - A report of any software bug, error, or problem.
- **Test Summary Report** - The final report that summarizes the completed testing process.

Testing Visit Saudi Tourism Application

After you have created your application, it is important to distribute your app for testing. The testers should vary in their profiles. This variety gives the developer more information on what to fix in the application and which features to implement. For example, in the current state of the application, users with difficulty in their vision or hearing impairments, will have trouble getting the correct information. In a later lesson, you will implement accessibility features for the mobile application you have created.

Packaging and Distributing an Application

After you have developed and tested your application, you need to package it into a file format that can be downloaded and installed on Android mobile devices. Then it can be distributed with two methods.

- **Downloading a package on your phone:** Downloading the package from your computer or from a website link and installing the application directly on your physical mobile phone.
- **Publishing to a store application:** Uploading the package to a store application so users can find it from wherever they are.

Application Versioning

Whichever method you choose for distributing your application, the first step in packaging is versioning. Each application does not stay the same forever. Fixes and updates are being implemented continuously. You have the same application name, but there may be changes to the UI or the code functionality. So there needs to be a way to differentiate those apps. Versioning is an identifier code that indicates which version of the application the user has currently installed on their mobile device. On applications that are meant for android, versioning is defined by the following properties:

- **VersionCode:** An identifier number that is defaulted to 1. It should be incremented every time that you upload a new version to a store application.
- **VersionName:** A string that can be set to any value but is defaulted to "1.0". The industry-standard convention is to increment the first digit every time that you make a major update to the app and increment the second digit every time that you make a minor update to the app. For example, an app that starts with the **VersionName** "1.0" and makes a minor update gets the **VersionName** "1.1". When a major update is implemented, then the **VersionName** becomes "2.0".

In MIT App Inventor, many general application properties including the versioning of the application are modified from the **Properties** section of **Screen1**. This is the reason why **Screen1** cannot be removed as a screen from the application.

VersionCode
1

VersionName
1.0

Application Appearance

There is also other information that is presented to the user when they want to download your application from the store. These are the following:

- **Application Name:**
This is the name that appears on the store and on your phone in your list of applications.
- **Application Description:**
Text that gives a brief overview of the application.
- **Application Logo:**
The logo icon that appears on the store and on your phone.

This information is defined in the **Properties** section of **Screen1**. The properties that must be defined are the following:

AppName
Visit_SA_App

AboutScreen

Icon
None...

Application Packaging

In order to install your application on a physical mobile phone, you need to convert what you have created in the browser of the App Inventor into a file that can be installed by Android devices. For Android mobile applications there are two package types.

.apk

Standard Android package format that has been used since the creation of the Android operating system. It is the simplest way to distribute an Android application. APKs can be downloaded directly from a website link or can be uploaded to the Google Play store.

.aab

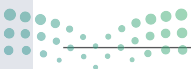
Android App Bundles (AABs) are a newer file format that is used to package Android applications. They contain an APK and metadata that allows the application to run smoothly on a wide range of devices. They can only be distributed through the Google Play store.

Exercises

1 Describe the difference between testing and debugging application software.

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

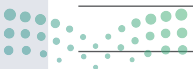
2 Classify the common types of testing strategies.

[illegible]

3 Outline the main components of testing documentation.

4 State the two methods of distributing an application for Android phones.

5 Describe the differences between the two main types of Android packages.





What is the Digital Divide?

The digital or technological divide is a social problem that refers to the disparity in the amount of information and skills between those with and without access to computers. The availability of high-speed Internet access at an affordable cost and quality is one of the most discussed issues these days. The term digital divide was popularized in the late nineties of the last century, and this gap was supposed to shrink in our time, but things have gotten worse.

This problem must be discussed in an international context, as many countries are more prepared than developing countries to benefit from the increasing growth in technology development. Appropriate use and access to technology and communications are vital to improving quality of life. In society, groups vary in their ability to benefit from the available technology. Research and studies have shown that the differences may be due to the presence of low-performance or low-quality computers, the poor quality or high cost of communication networks, or the difficulty of obtaining training or accessing high-quality content via the Internet. Also, having access to technical support is very important.

The Kingdom of Saudi Arabia aims to become one of the leading countries in digital transformation. A very important step in this process is bridging the digital divide as much as possible. So, legislation and policies like the following are passed to aid the Kingdom's citizens.

- **E-Learning:** The Saudi government has made significant investments in e-learning initiatives to provide access to quality education for all citizens, regardless of their location. The goal is to use technology to reach students in remote and underserved areas and to provide them with equal access to quality education.
- **National Broadband Plan:** The National Broadband Plan aims to increase broadband access across the country and reduce the digital divide. The plan includes providing subsidies for broadband services, increasing investment in digital infrastructure, and improving broadband access in rural and underserved areas.
- **Digital Inclusion Programs:** There are several digital inclusion programs aimed at providing digital skills training and access to digital devices for marginalized and low-income communities in Saudi Arabia. These programs aim to improve access to digital services and bridge the digital divide.



Figure 4.1: The digital divide affecting elderly people

Factors Affecting the Digital Divide

Many factors contribute to the increase in the digital divide. These difficulties are not limited to a specific country but extend to entire continents, making the digital divide a global issue. The following is a list of common factors affecting the digital divide.

Education

Studies show that those with a university degree can access the Internet at work ten times more than those with a secondary education. Literacy also plays a major role in facilitating dealing with a computer and accessing the Internet. Access to more resources and information speeds up an individual's learning process.



Users Age

Most of the elderly are far from technology compared to the younger generations. They also need some training and rehabilitation through family members and using tools that make it easier for them to deal with modern technologies.



Geographical Location

Households in urban and suburban areas have more than ten times more computers available in their homes than households in rural areas. Telecom companies prefer to build infrastructure in urban areas to serve many customers at a lower cost than in vast rural areas with a limited population. Companies and governments can overcome this issue through alternative solutions such as internet connection power lines and satellite communications.



Economic Level

Wealthier societies have better chances of adopting new technologies than poorer ones. It is common for richer public areas to offer free internet access, unlike in more impoverished areas where the need is higher.



Language Differences

Language directly affects the user's experience with the Internet, as it governs the amount and quality of information he can access and the communities he can communicate with. For example, a Google search may give you ten times more specific information in one particular language than when searching in another language. And if your language isn't popular, it's likely that not enough content will be available in this language on the Internet.



Special Needs

Modern technology may be available to some people. However, the presence of a disability, such as a limb disability, and audio-visual impairments, prevents them from using technology, like interacting with screens or smartphones. According to research, about 15% of the world's population lives with some disability, and 2% to 4% have difficulty performing their daily activities independently.



What is Digital Inclusion?

Digital inclusion ensures equitable access to and use of information and communication technology for social and economic engagement, including education, social services, health, and social and community participation. Digital inclusion includes inexpensive broadband Internet, Internet-enabled devices, digital literacy training, competent technical support, and applications and online content that foster self-sufficiency, participation, and cooperation. They are the solution to the gap created by the digital divide.

There are many solutions to the digital divide, but first, to reduce it we must address the issues of weak infrastructure and deal with the repercussions of low levels of education and poverty. The following are some solutions that can help reduce this gap.

The Cost of Technology

- Reducing the cost of users' devices and the costs of connecting to the Internet and using its services.
- Providing financing to help low-income people bear the burdens of modern technology and reducing customs tariffs to encourage them to acquire digital tools.
- Lowering the cost of ancillary services (e.g. solar chargers rather than electricity chargers).

Convenience

- Develop content and applications in local languages to increase the ability of local people to understand them easily.
- Address privacy and data security issues that make users suspicious of evolving technology.
- Adjusting workplaces to accommodate people with special needs and developing assistive software.
- Computers should be available to all persons, regardless of their physical or learning abilities.

Efficiency

- Training on ICT means and skills.
- Focusing on education and updating the curricula periodically.
- There is a need to have a certain level of information literacy to use computer technology, which can pose serious challenges for people. These challenges include the exponential increase in information and the ability to find and use information.

Infrastructure

- Expanding and upgrading the network to increase its capacity.
- Develop solutions for rural areas in terms of cost-effectiveness on a large scale.
- Develop stable infrastructures to support the digital economy, such as fourth and fifth-generation mobile network operators.
- Using wireless technology to provide Internet service to those who own computers or devices capable of connecting to the network in public places, cafes, and libraries, as is the case in some cities that provide wireless connections.

Advantages of Reducing the Digital Divide

- Increasing the percentage of segments of society accessing the automated services the state provides to its citizens, such as government e-services.
- Allowing space for the participation of different parts of society in questionnaires and opinion polls related to the services offered by various institutions.
- Expanding the reach of education to a larger number of community members through the use of various educational resources available on the Internet.
- It allows entrepreneurs to market their products, introduce their projects, and create ideas for new projects based on customers' use of technology.



Figure 4.2: The importance of reducing the digital divide

Accessibility for People with Special Needs

Accessibility is the process of designing products, devices, services, or environments so that all people can use them. This concept focuses on enabling or facilitating access for people with disabilities through assistive technology.

In the world of technology, accessibility is hardware and software designed to help people overcome their disabilities, just like the equipment and supplies that help people overcome disabilities, such as wheelchairs, hearing aids, ramps that facilitate entry to a building, and Braille.

The following areas should be accessible to people with special needs.

Hardware Accessibility

Providing accessibility accessories, such as large-character keyboards, large mice, and keys that can be activated with a little force, as well as other devices enables users with disabilities to use computers in alternative ways. The following are guidelines for designing accessible input and control devices.

- Switches and controls must be within reach, easily accessible, identifiable by touch, and usable with one hand in an easy manner.
- The switches and controls in these devices are designed to be touched and recognized without activating them.
- Allow alternative control methods using different senses for touchscreen devices, such as voice commands.
- Control devices for people with special needs are designed to recognize their status (opening/closing) and their response to commands using different senses other than the sense of sight, such as touch or hearing.
- Accessibility control devices should connect to computers and smart devices using standard wires and ports available on common devices.

Knowledge of accessibility principles and tools are essential for developers and organizations who want to create high-quality websites and web tools in a way that ensures their products and services are accessible to people with disabilities.

The following is a list of alternate input devices for people with various disabilities.

- **Braille computer keyboard:** A Braille computer keyboard is a specialized keyboard that allows users with visual impairments to input text and to control their computer. It typically consists of six or eight raised dots representing Braille characters, and the user can enter text by pressing the dots with their fingers. The keyboard is designed to be used with screen reader software that converts the Braille input into text that can be displayed on the screen.
- **Head-mouse control:** A head mouse control, also known as a head-tracking mouse, is a device that allows people with physical disabilities to control the movements of a computer mouse using head movements. It typically consists of a small camera that tracks the user's head movements and translates them into mouse movements on the screen. The camera can be mounted on a headband, a hat, or a pair of glasses, and it is connected to the computer via USB or Bluetooth.
- **Foot-mouse control:** A foot-mouse control, also known as a foot-operated mouse, is a device that allows users with physical disabilities to control the movements of a computer mouse using their feet. It typically consists of a small platform with two pedals that the user operates to control the movement of the mouse cursor on the screen. The foot mouse control is connected to the computer via USB or Bluetooth.
- **Brain EEG control:** Brain EEG control, also known as a brain-computer interface (BCI), is a technology that enables users to control devices or applications using their brain activity. It typically involves measuring the electrical signals generated by the brain, known as electroencephalography (EEG) signals, and translating them into commands that the computer can understand.
- **Eye tracking control:** Eye tracking control, also known as eye gaze control, is a technology that allows users to control devices or applications by tracking their eye movements. It works by using a specialized camera or sensor to track the movements of the user's eyes and then translating those movements into commands that the computer can understand.

Software Accessibility

Modern operating systems such as Windows and macOS provide display adjustment options, which include tools such as the ability to enlarge screen contents and invert colors. These settings help those with vision problems and the ability to enable text-to-speech and describe objects and text on the screen more accurately, as well as the possibility of using voice commands to perform basic tasks.

Table 4.3: Accessibility settings in various operating systems

Operating System	Setting
Windows	Settings → Ease of Access
macOS	System Preferences → Accessibility
iOS	Settings → General → Accessibility
Android	Settings → Accessibility

Web Accessibility

The web should be accessible to all people, regardless of their hardware, software, language, location, or abilities, to be accessible to people of different hearing, motor, visual, or cognitive abilities.

Therefore, accessibility can overcome the impact of disability on the web because it removes the barriers facing users in the real world. Accordingly, websites and their various applications should be designed to take into account all groups (taking into account inclusiveness in design) and allow them to use the web effectively.

What is Web Accessibility?

Sites, tools, and technologies are designed to enable people with disabilities to use these sites, and specifically to enable these people to perceive, understand, navigate and interact with the web over the Internet.

Web accessibility includes all disabilities that can affect a person's ability to access the web, including:

- Speech difficulties
- Physical disability
- Hearing disability
- Vision difficulties
- Cognitive difficulties



Figure 4.3: Web accessibility



COGNITIVE



VISION



HEARING



MOTOR



SPEECH

Figure 4.4: Types of disabilities

Web access is essential for groups other than those with special needs, such as the elderly with diminished capabilities due to age and those who suffer from permanent disabilities or temporary disabilities due to certain accidents, such as arm fractures and others.

Principles of Website Development for Accessibility

Web accessibility aims to meet the needs of each visitor to the site to choose a certain level of use, so the following are some conditions to achieve this goal.

Clarify Vision through Careful Color Selection and Increased Contrast

People with visual impairments may find it difficult to read a text without a high-contrast background, whether it is a plain background or text embedded within an image.

Not Relying Only on Colors to Clarify Information

Using designs that rely only on color discrimination is not enough for people who cannot distinguish color differences. The correct position of the active light in the traffic light is appropriate to provide the necessary information regarding stopping or moving forward to individuals with color blindness. Therefore, designers should use more than one method to express the meaning intended by design.

Browse Using the Keyboard

We usually use the mouse to browse the web, but sometimes using the mouse is difficult. Hence, the keyboard provides us with options to navigate the web page to suit users with limited mobility. For navigation using the keyboard, special methods are used in designing links, such as highlighting them in color and defining designs for different cases, such as pressing, scrolling, and others.

Provide the Correct Naming of the Fields

A descriptive label is provided for all form fields. It is important to note that some users with cognitive disabilities may not be able to understand the meaning of form fields.

Variety of Feedback for Errors

To enable users to correct their errors quickly as they interact with your website, alert them to errors using text, icons, and colors. Designers can use color to provide appropriate feedback to the user.

Providing Several Alternatives for the Media Used

The variety of media used, including images, audio, text, and video, provides equal access to information for users with different disabilities. Composite picture texts and audio and text versions make the content more attractive to users with hearing or vision impairments. It is good to provide a text version with audio information, which helps people. People who are deaf or hard of hearing cannot understand the content, and this also applies to search engines and other technologies that are not audiologically available.

Write Useful Alt Text for Images and Other Non-Text Content

People with low vision often benefit from screen readers to obtain information on web pages audibly, as these tools convert text into speech that enables the person to hear the words on the website. When image alt texts are available, the Auto Reader will describe the image using the alt text rather than just indicating the picture's existence to search engines and other technologies that are not audiologically available.

Usability-Focused Design

Usability-Focused design incorporates a user-centered design approach, where the user's needs, behaviors, and expectations are central to the design process. This approach involves close collaboration between designers, developers, and stakeholders to ensure that the software meets the needs of its intended audience. The usability-focused design process includes user research, prototyping, testing, and iteration. User research may involve surveys, focus groups, or usability tests to better understand the user's needs, goals, and pain points.

Examples of Accessible Applications

Various applications have been created with certain standards to be friendly to users with different types of permanent or temporary disabilities, and the following are some examples.

AccessNow accessnow

AccessNow shares accessibility information about places around the world. You can search for specific businesses like a restaurant, hotel, or store or browse a map to see accessibility features nearby that a person needs. For example, a person using a wheelchair can get a list of restaurants available for wheelchair users in a specific area. If the information is not already on the map, the application allows users to add it and contribute to community service worldwide.

RogerVoice

RogerVoice is designed to help deaf people communicate over the phone. Usually, voice messages are with a person who is deaf on one side. This app allows deaf people to participate in a discussion using voice recognition technology and convert it into written text that a deaf person can read.

Envision AI

The Envision AI application uses the camera to describe what is happening around the person. For example, you can point your phone at your companion sitting in front of you, and the program will notify you that a person is sitting there and will describe things to you. The application can also read documents, recognize handwriting, and scan barcodes, and the program supports 60 languages. You can also have Envision AI recognize photos of your family members, and it will remember them in the future.

Exercises

1

Read the sentences and tick ✓ True or False.	True	False
1. The digital divide is exclusively an economic problem.	☐	☐
2. Geographical restrictions do not contribute to the digital divide.	☐	☐
3. The digital divide affects both illiterate people and people with disabilities.	☐	☐
4. All technological applications have been appropriated to local languages.	☐	☐
5. The rising cost of hardware components contributes to the digital divide.	☐	☐
6. Providing fast Internet access to remote areas helps combat the digital divide.	☐	☐
7. Reducing the digital divide helps create more markets for businesses.	☐	☐
8. Only software accessibility is needed to reduce the digital divide for people with disabilities.	☐	☐
9. There are no standard guidelines for designing accessible websites.	☐	☐
10. Usability-focused design is used only for people with vision disabilities.	☐	☐



2 Describe what the digital divide is.

3 Illustrate how reducing the cost of technological components helps bridge the digital divide.

4 Classify the main societal advantages of reducing the digital divide.



5 Explain how hardware products can be modified to be accessible to people with disabilities.

[illegible]

6 Describe three principles of website development for accessibility.



Lesson 3

Accessibility Features in an Application

Link to digital lesson



www.ien.edu.sa

The Suitability of the Application for People with Special Needs

Not all users have the same needs, that's why applications must take into account these differences and modify their user interface and functionality based on them.

You are going to improve the application you created in the previous unit to help elderly people navigate through the screen so that they can read information about the different tourist spots that they can visit in Saudi Arabia, as well.

Specifically, because elderly people have vision problems, they will have the ability to adjust the size of the text so that they can read it easily. Others may have problems stabilizing hand movements, so it will be difficult for them to press a button on the screen, and that's why you have to give them access to adjust the size of the app buttons to their liking.

Making your mobile app accessible to people with poor vision and visual difficulties does not necessarily require a huge amount of work. The most important thing here is to remember that users have different needs. To adapt to the needs of a user who has difficulty seeing, it is important to improve the initial application by adding the following features:

Zoom In and Out Function

You can add the Zoom In and Zoom Out feature so that the user can adjust the text size according to his needs.

Colored Interface

You can add the option to change between colored interfaces or those that display in black and white only in order to facilitate reading for older users or people with disabilities, as well as for general users, especially on mobile devices.

Text to Speech

It is important that the application supports as much interaction with the human senses as possible, so that users are able to understand and assimilate information contained in various media such as images, audio, video, animation and presentations. We will modify our application to support a sense other than sight, by adding the option for the user to listen to information if they are unable to read it for any reason.



Adjusting the Prototype to Improve the Accessibility of the App

Before starting to make the appropriate changes to the mobile app in App Inventor, you must make the changes in the prototype you created with Pencil Project.

Open Pencil Project and redesign the last screen of the app.

You will add four images as you have already learned to do and make the screen that displays Al Masmak as shown below:

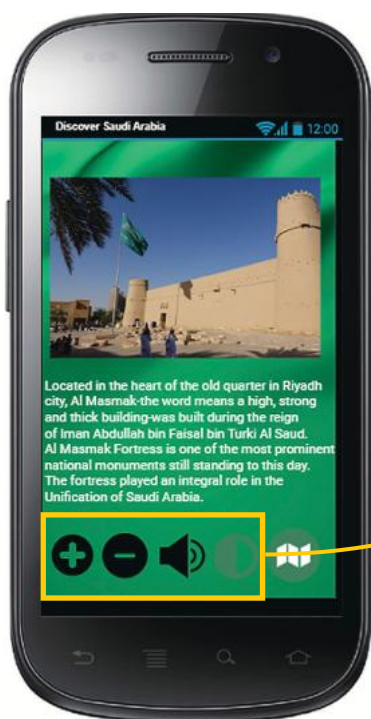
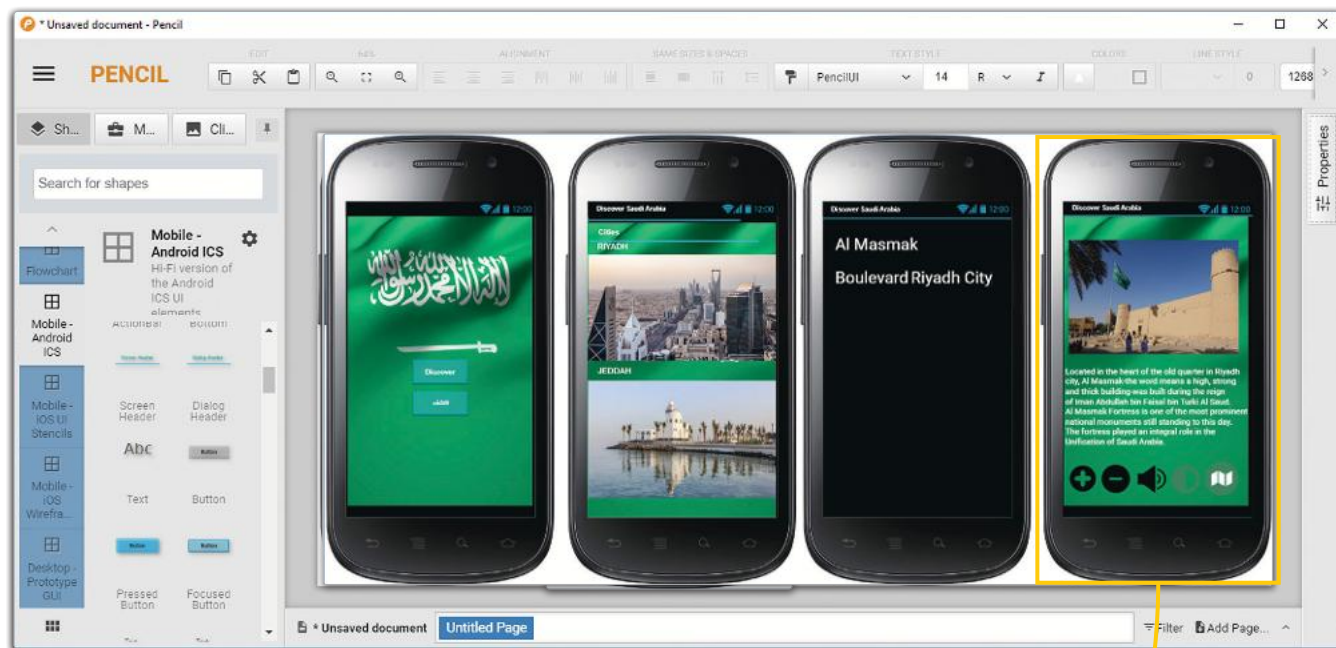


Figure 4.5: Design modifications to improve the accessibility in Pencil Project



Enhancing the UI with Accessibility Features

You will now enhance the screen with the **AI Masmak** highlight with accessibility features. You will add buttons to enlarge or minimize the text size, a button to change the color theme of the screen and a button to add text-to-speech functionality on the screen. You will first add the components on the **Designer** page and then you will program their functionality on the **Blocks** page.

Adding a Zoom In Button to the Application

You will now add a button that will enlarge the font size of all the text components every time you click on it.

To add a zoom in button:

- > From the **User Interface** group, add a **Button** component to the screen ¹ and rename it **zoomin_button**. ²
- > In the **zoomin_button** component, set the **BackgroundColor** property to **None**, ³ clear the **Text** property ⁴ and set the **Image** property to an icon of a plus sign. ⁵

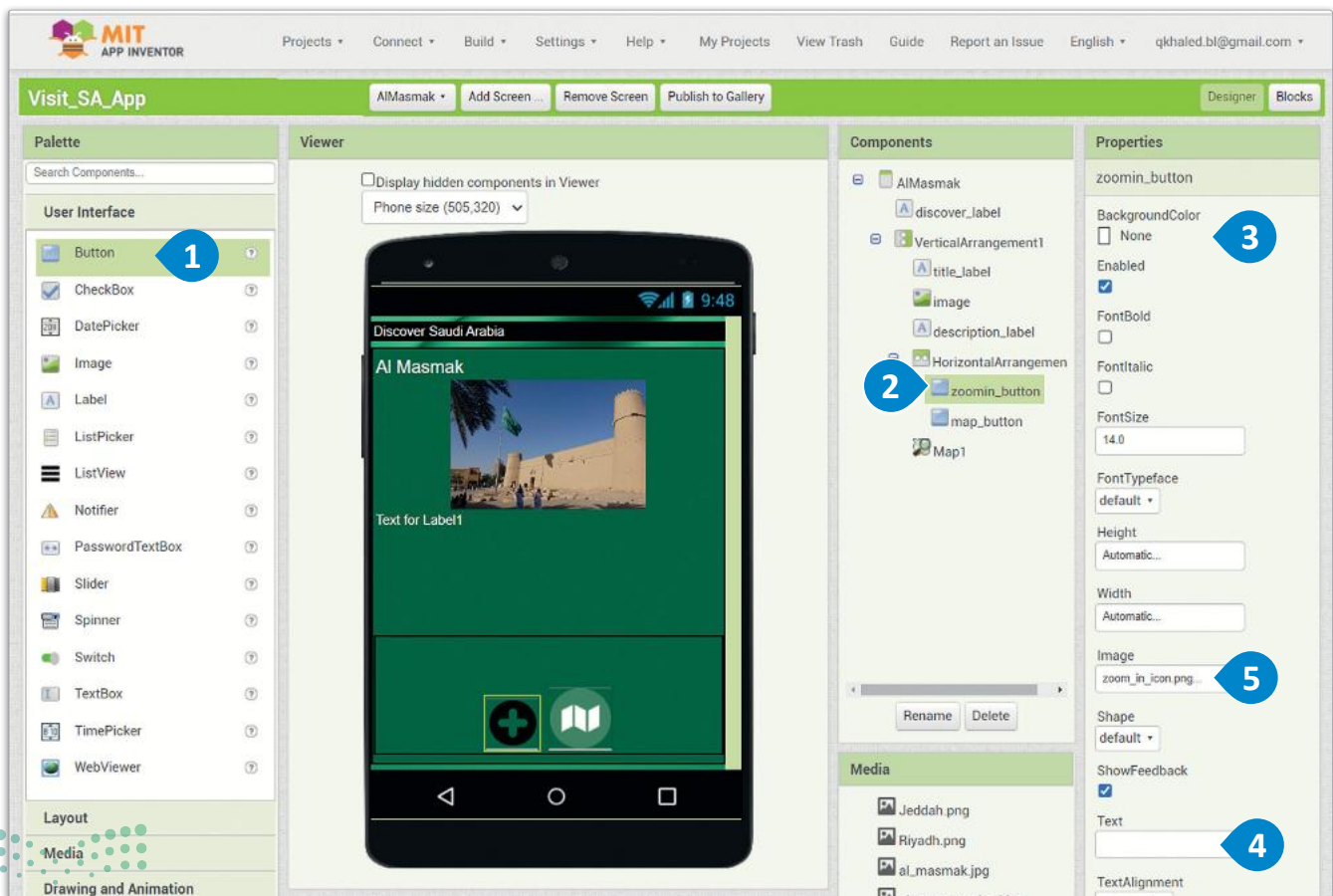


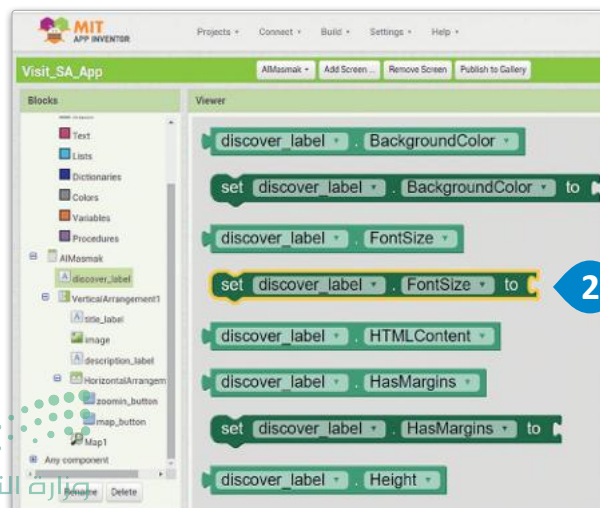
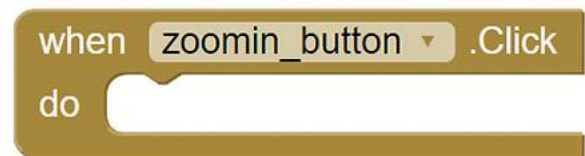
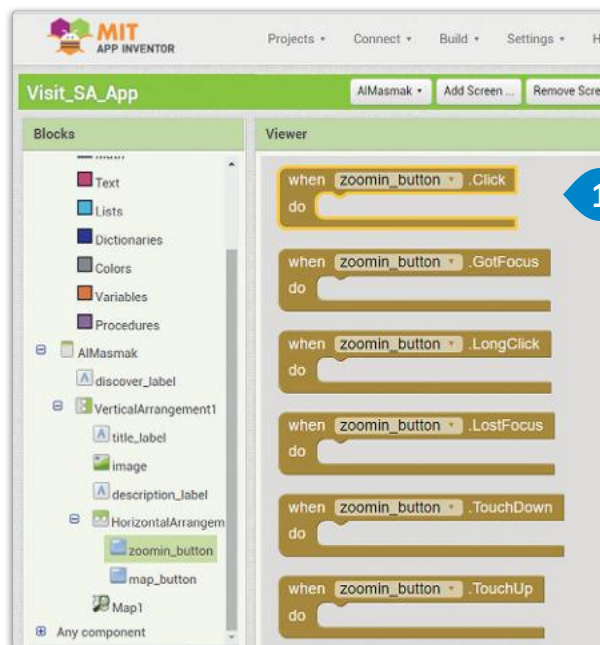
Figure 4.6: Adding a zoom-in button

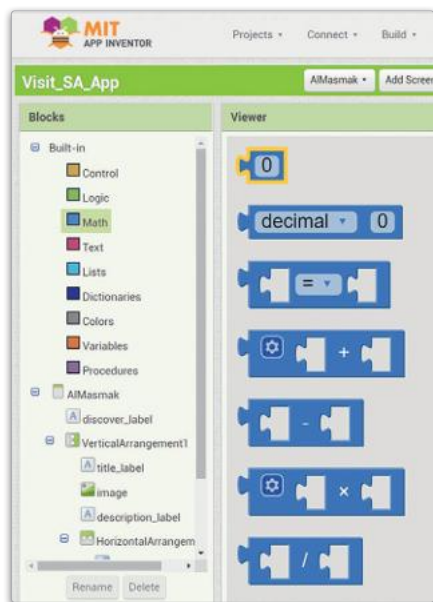
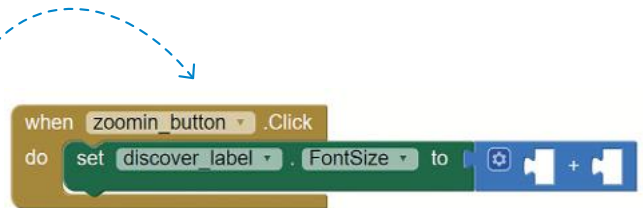
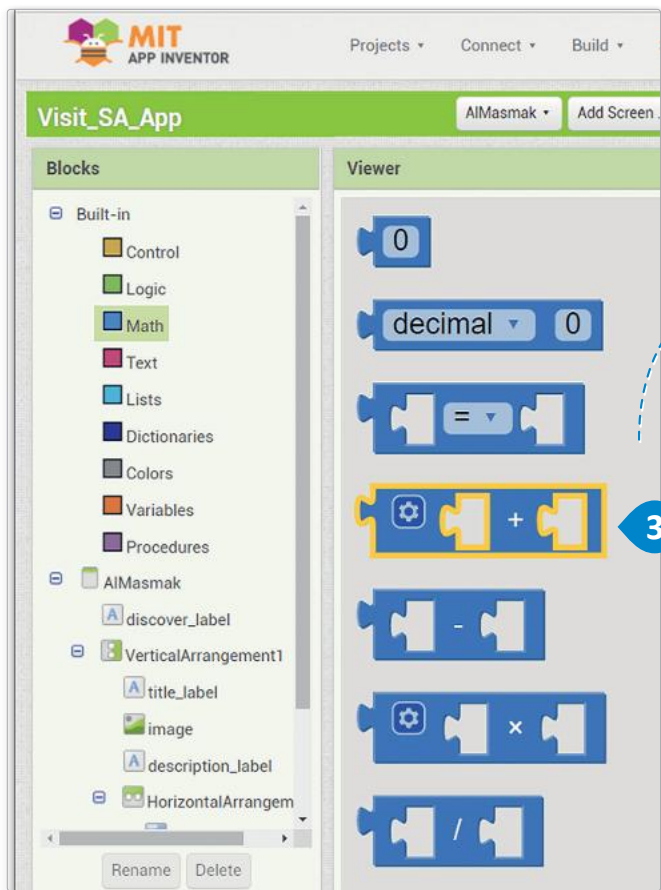
Programming the Zoom In Button

This button will increment the existing value of the **FontSize** property of all the text components by 1 point each time that it is clicked.

To program the zoom in button:

- > Select the **Click** event for the **zoomin_button** component. 1
- > Select the **Set FontSize** command of the **discover_label** component. 2
- > Select an **addition** codeblock from the **Math** command group. 3
- > Add the value of **1** to the existing **FontSize** property of the **discover_label** component. 4
- > Repeat the same process for the remaining text components of the screen. 5





4

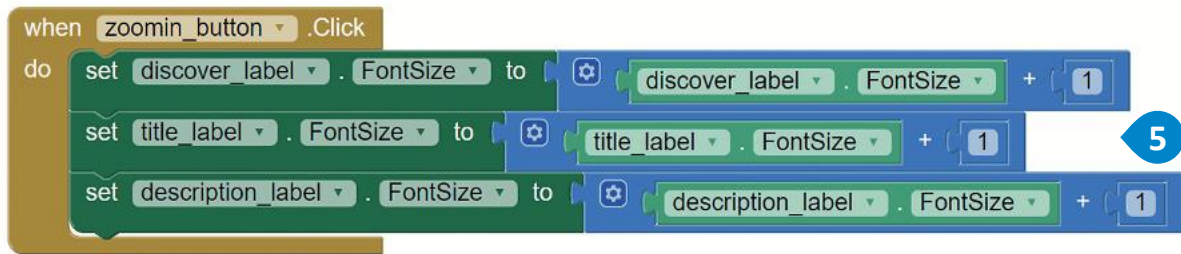


Figure 4.7: Programming the zoom-in button

Adding a Zoom Out Button to the Application

You will now add a button that will reduce the font size of all the text components every time you click on it.

To add a zoom out button:

- > From the **User Interface** group, add a **Button** component to the screen ¹ and rename it **zoomout_button**. ²
- > In the **zoomout_button** component set the **BackgroundColor** property to **None**, ³ clear the **Text** property ⁴ and set the **Image** property to an icon of a minus sign. ⁵

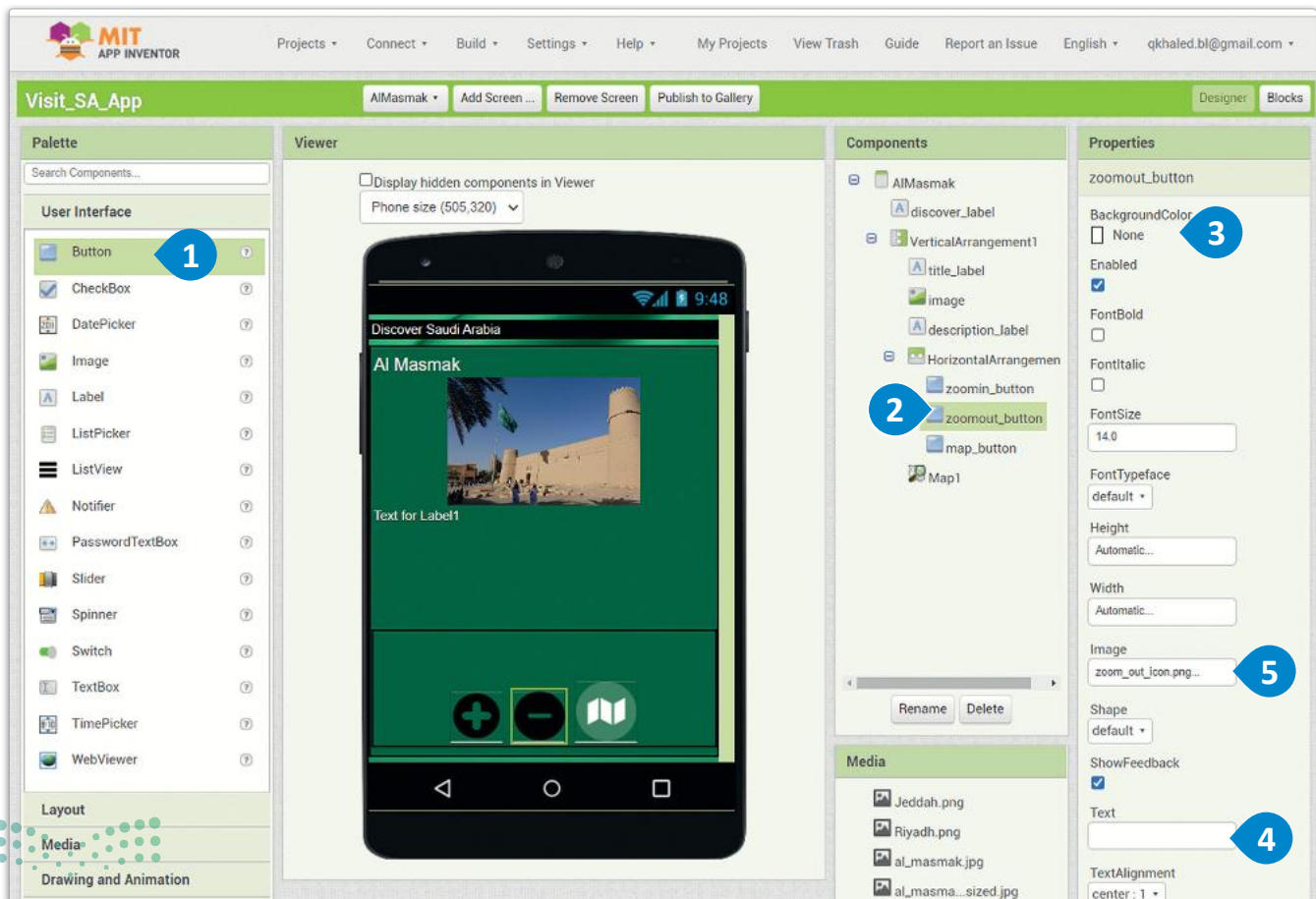


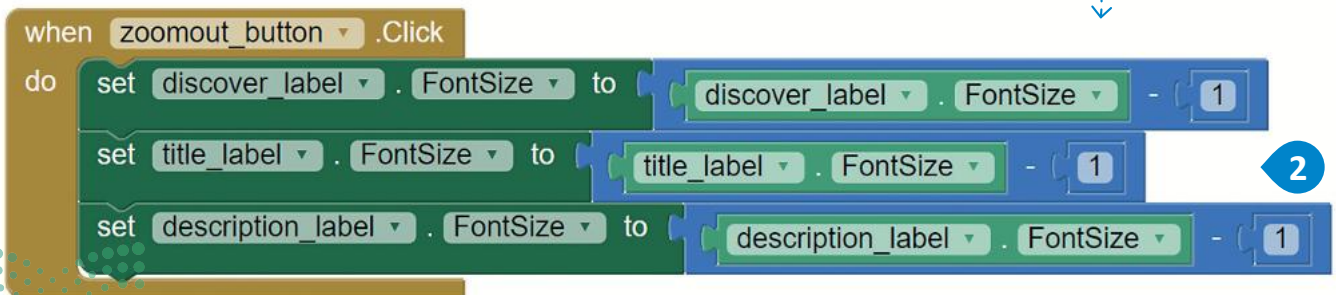
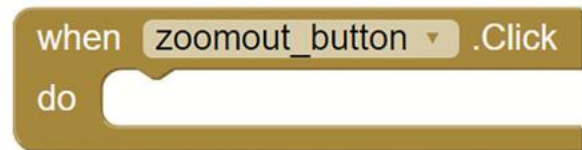
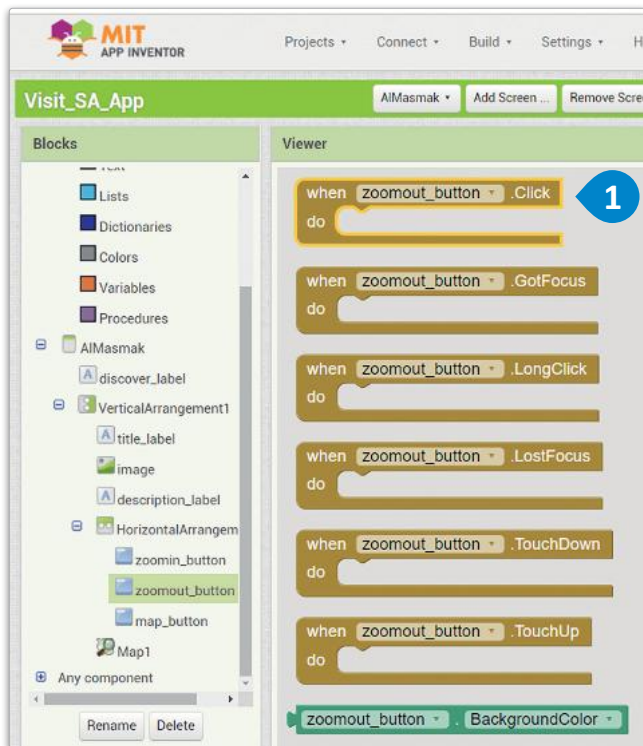
Figure 4.8: Adding a zoom-out button

Programming the Zoom Out Button

This button will reduce the existing value of the **FontSize** property of all the text components by 1 point each time that it is clicked.

To program the zoom out button:

- > Select the **Click** event for the **zoomout_button** component. **1**
- > Repeat the same process you followed for the **zoomin_button** component, changing only the mathematical operation from an **addition** to a **subtraction**. **2**



Adding a Text-To-Speech Button to the Application

You will now add a button that will activate a **TextToSpeech** component that uses the mobile device's sound system to read a specified text aloud.

To add a text-to-speech button:

- > From the **User Interface** group, add a **Button** component to the screen **1** and rename it **text_to_speech_button**. **2**
- > In the **text_to_speech_button** component, set the **BackgroundColor** property to **None**, **3** clear the **Text** property **4** and set the **Image** property to an icon of a speaker. **5**

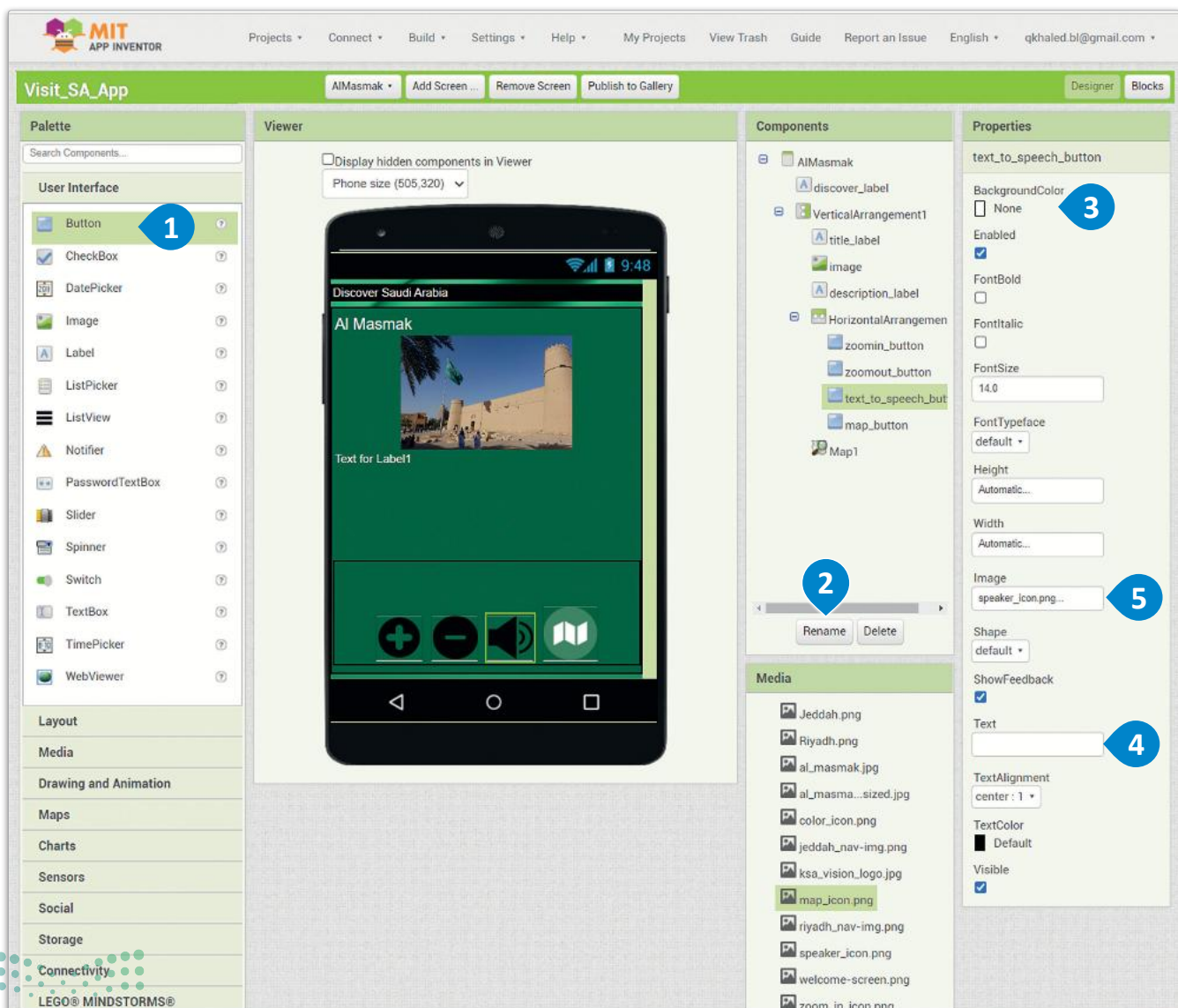


Figure 4.10: Adding a text-to-speech button

To add a TextToSpeech component:

- > From the **Media** group, add a **TextToSpeech** component to the screen **1** and rename it **text_to_speech**. **2**
- > In the **text_to_speech** component set the **Language** property to **en**, **3** and set the **SpeechRate** property to **0.5**. **4**

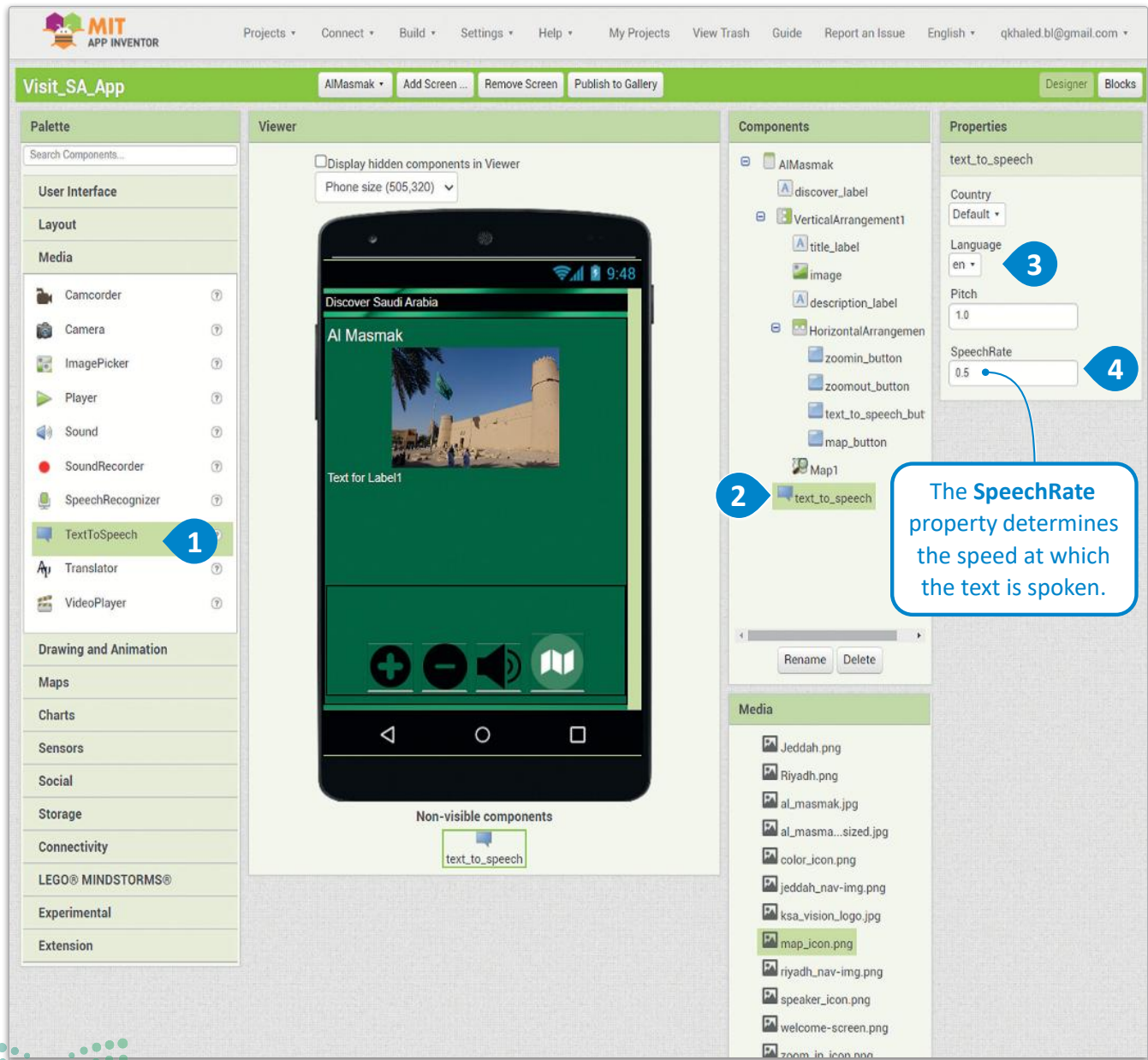


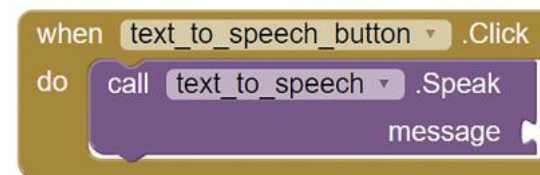
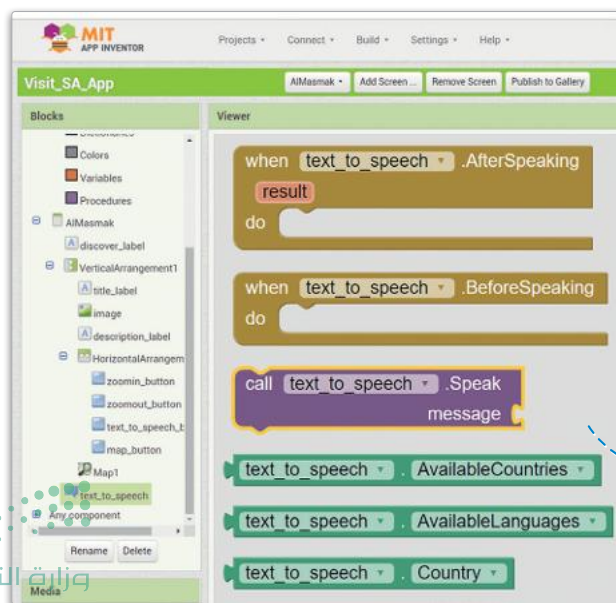
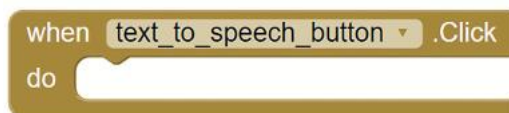
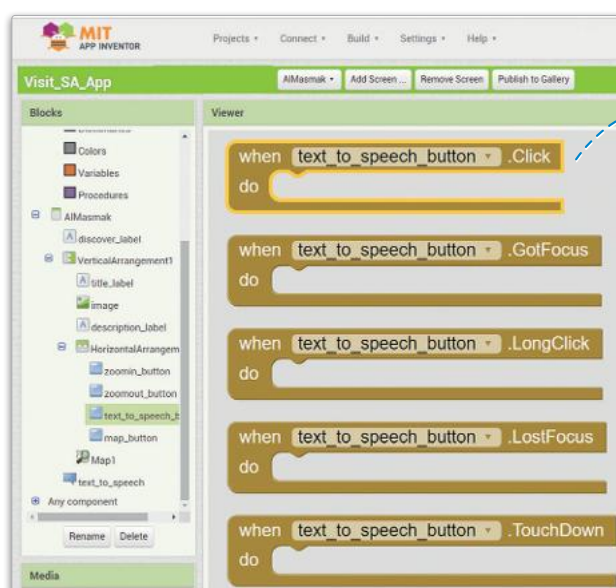
Figure 4.11: Adding a text-to-speech component

Programming the Text-To-Speech Button

This button will create a text from the contents of all the text components on the screen which will be provided to the **TextToSpeech** component to produce the spoken message.

To program the text-to-speech button:

- > Select the **Click** event for the **text_to_speech_button** component. ¹
- > Select the **Speak message** command for the **text_to_speech** component. ²
- > Select the **join** command from the **Text** group in order to join texts. ³
- > Add the **Text** property of the **discover_label** component to the first input of the **join** command. ⁴
- > Repeat the above process for the remaining text components. ⁵



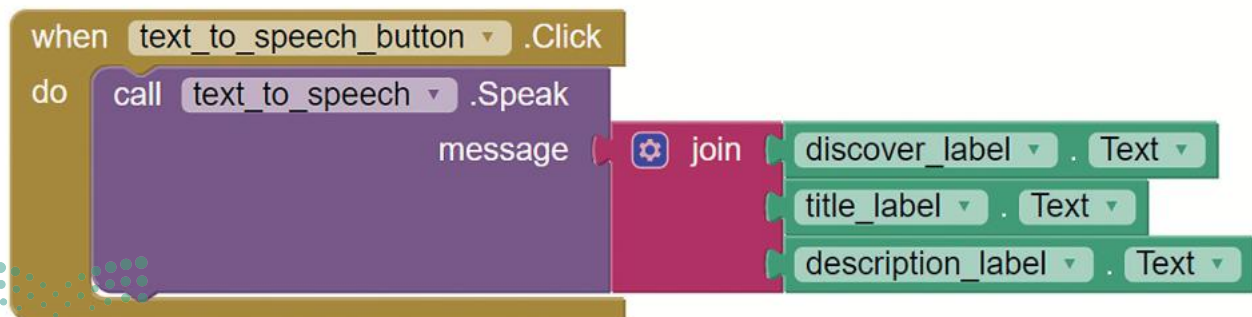
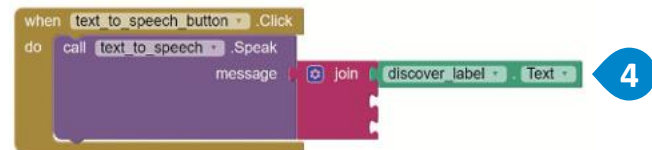
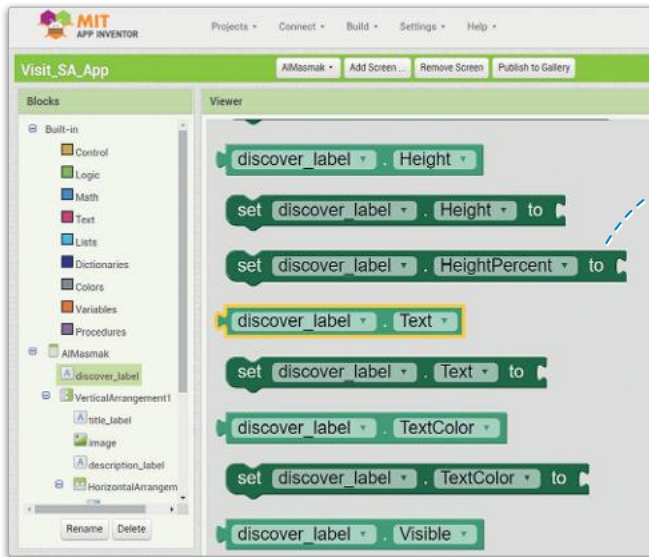
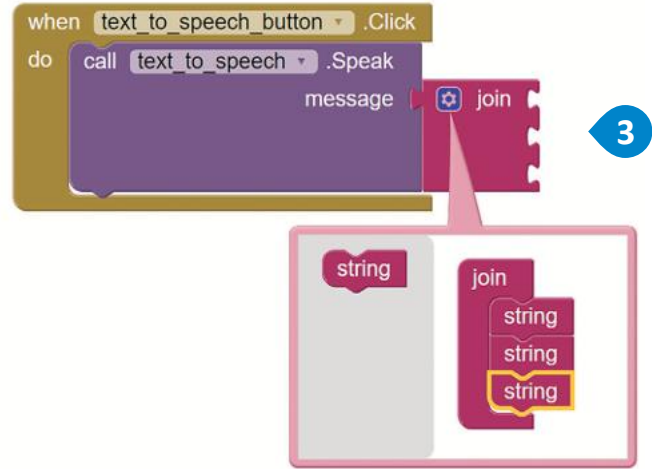


Figure 4.12: Programming the text-to-speech functionality

Adding a Button to Change the Color Theme of the Application

You will now add a button that will change the theme of the screen between a light theme and a green theme.

To add a color theme button:

- > From the **User Interface** group, add a **Button** component to the screen **1** and rename it **color_button**. **2**
- > In the **color_button** component, set the **BackgroundColor** property to **None**, **3** clear the **Text** property **4** and set the **Image** property to an icon of a half circle. **5**

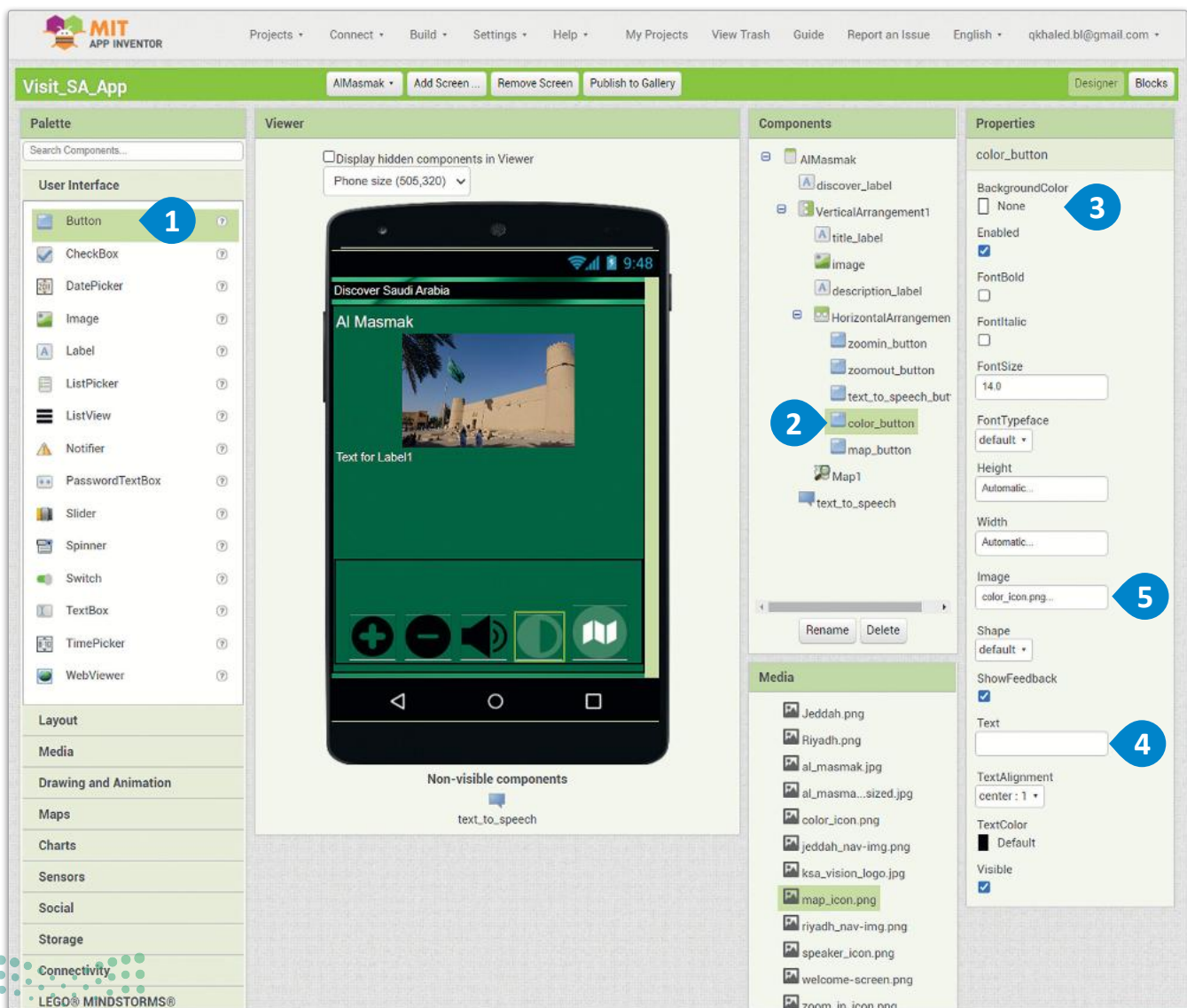


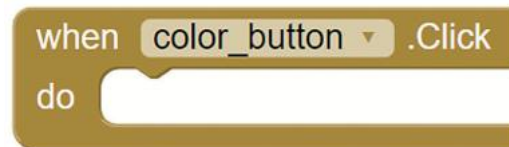
Figure 4.13: Adding a color theme button

Programming the Color Theme Button

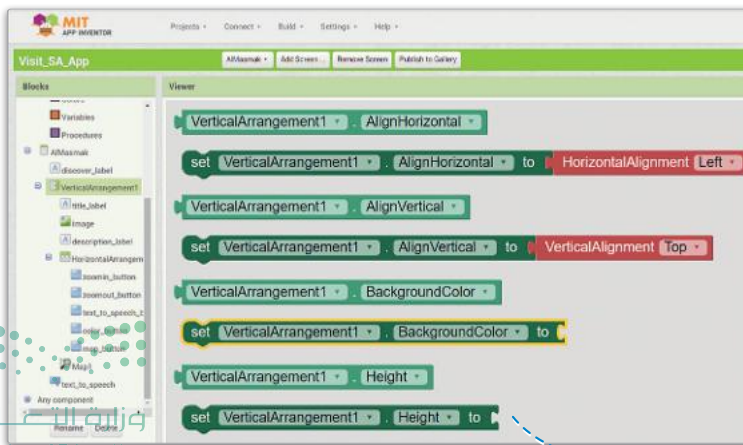
When this button is clicked, the background of the screen will become white and the text color will become black. Whereas, when the button is long clicked, the background of the screen will become dark green and the text color will become white.

To program the color theme button:

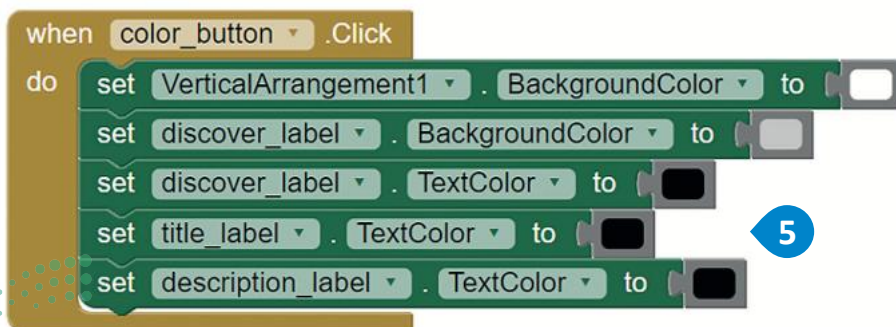
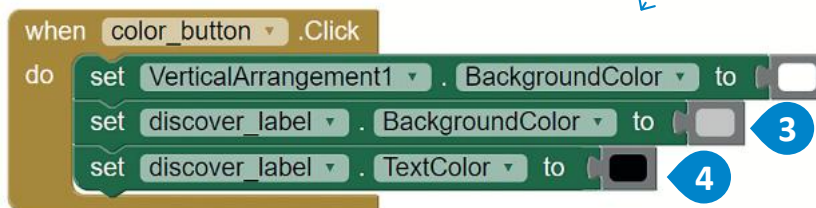
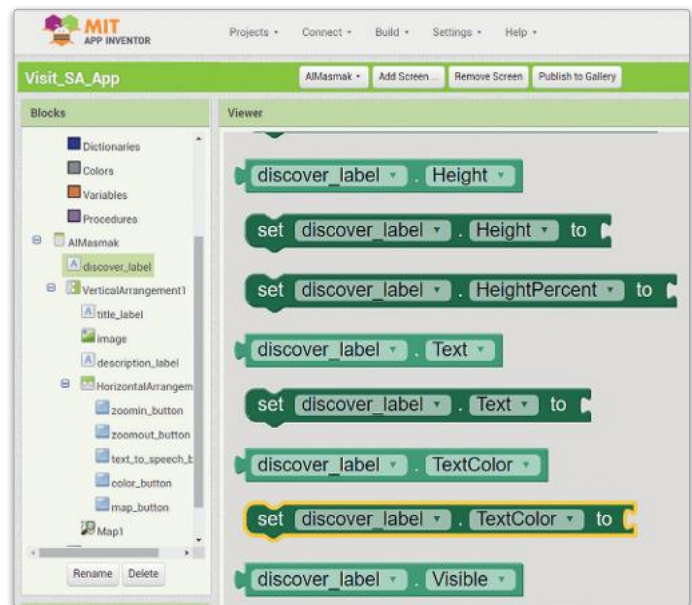
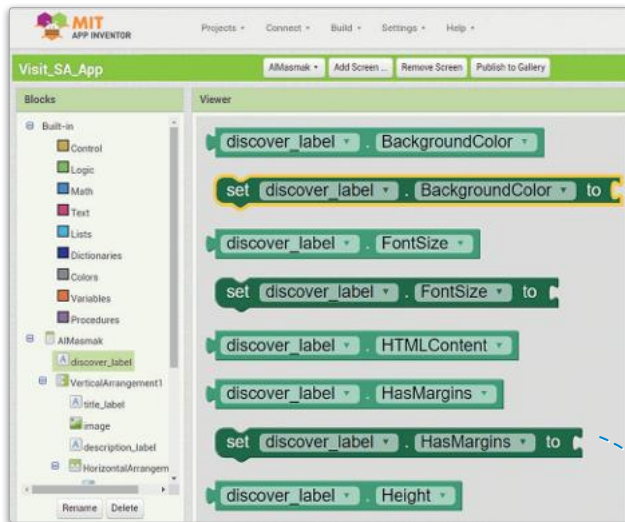
- > Select the **Click** event for the **color_button** component. **1**
- > Set the **BackgroundColor** property of the **VerticalArrangement1** component to **White**. **2**
- > Set the **BackgroundColor** property of the **discover_label** component to **Light Grey**. **3**
- > Set the **TextColor** property of the **discover_label** component to **Black**. **4**
- > Repeat the above process for the remaining text components. **5**
- > Select the **Long Click** event for the **color_button** component. **6**
- > Set the appropriate colors for the components as in the above process. **7**



1



2



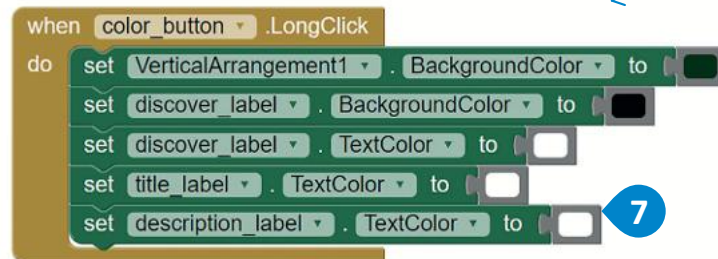
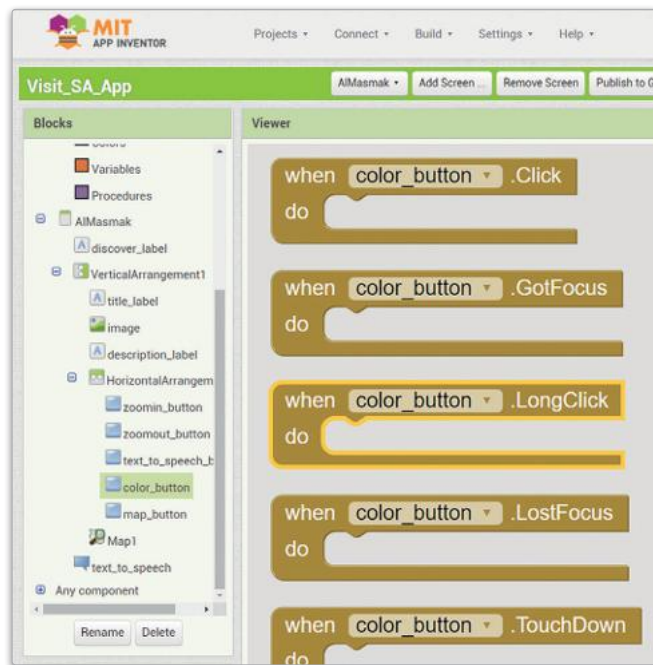
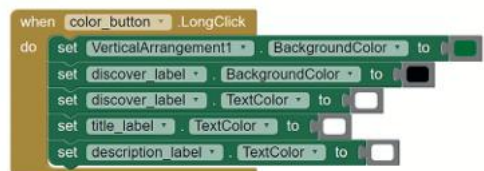
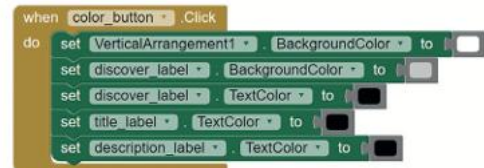
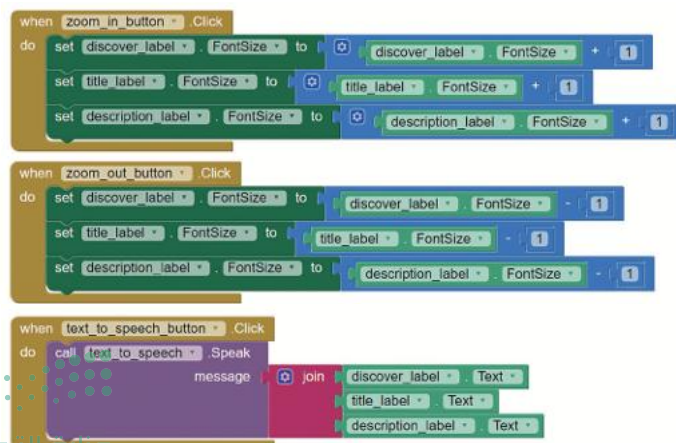


Figure 4.14: Programming the color theme button

The Complete Code for the Accessibility Features on the Third Screen (AI Masmak)



The software is ready and you have to test it. Run the application using the Android Emulator or scan the QR code with the Android device to preview. The images below show how the accessibility features will appear on a mobile device.

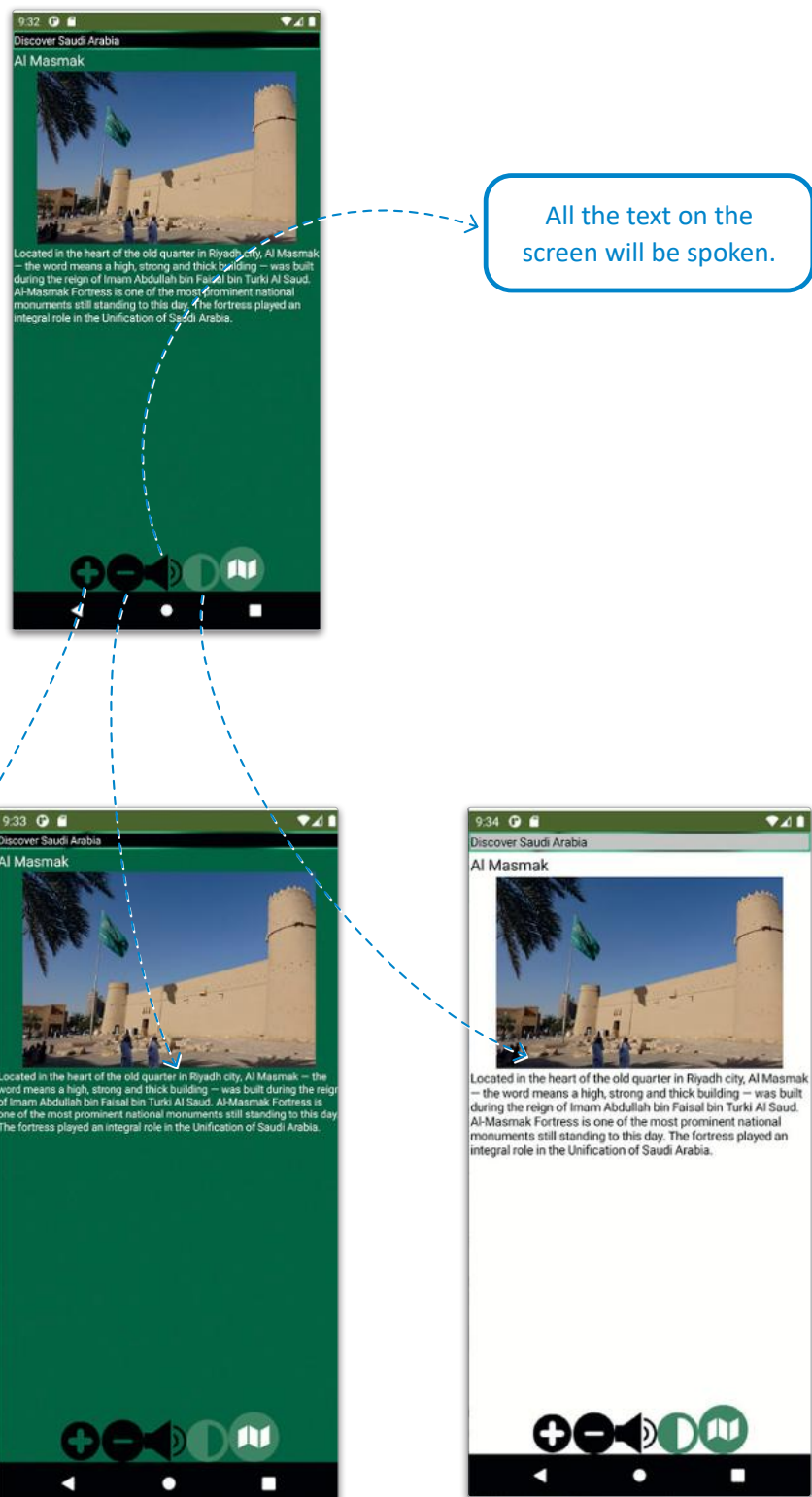


Figure 4.15: The accessibility features in the Emulator

Exercises

- 1 Describe how each accessibility feature added will help people with various disabilities use the application.

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and extend across the width of the page. There are no margins, text, or other markings on the paper.

- 2** Add the zoom in and zoom out buttons on the remaining screens of the application.

- 3 Extend the zoom in and zoom out button functionality to enlarge the **Button** and **ListPicker** components of each screen.

- 4** Add the color theme button on the remaining screens of the application.

- 5** Add the text-to-speech button on the remaining screens of the application.

- 6 Extend the text-to-speech button functionality to add predefined messages on every screen to introduce each component before speaking the content of each component.

Project

1

In a previous Unit you built the mobile application for the tourism in KSA part of Vision 2030 and now you will enhance it to make it more accessible.

2

People with disabilities need to have access to the information concerning the Kingdom's newest megacity projects. You will implement the necessary features to make your application more inclusive for people with various disabilities.

3

More specifically, you will implement features to accommodate users with color blindness, vision difficulties and general blindness.

4

Think which features need to be implemented in order to help the above users and add them to the existing application.



Wrap up

Now you have learned to:

- > Demonstrate the processes and methodologies of testing software applications.
- > Build, deploy and test a mobile application in MIT App Inventor.
- > Examine how the digital divide affects society.
- > Outline how the digital divide can be combated through digital divide solutions.
- > Distinguish the solutions for providing accessibility features in both hardware and software.
- > Develop an application with accessibility features in mind.

KEY TERMS

Android App Bundle

Android Package

Application Versioning

Automated Testing

Debugging

Digital Divide

Digital Inclusion

Hardware Accessibility

Software Accessibility

Software Quality

Testing

Text-To-Speech

Usability-Focused Design

Web Accessibility





Notes



