

#### Quiz App

### **Submitted by**

1. Yasser Mahmoud Abd Elghany
2. Moustafa Ibrahem Abd Elbaset

##### Under Supervision of

**Eng. Amany Mamdouh**

Teaching assistant at Department of Computer Science

Misr Higher Institute of Commerce and Computers in Mansoura

**Dr. Dr Mohamed Abd Elfattah**

Lecturer at Department of Computer Science

Misr Higher Institute for Commerce and Computers in Mansoura

**Chapter 1**

**Introduction to Android**

**1\_1 What is Android?**

**1\_2 Android Development Tools**

**1\_3 Android Security and permissions**

**1\_4 Programming software for Android**

**1.1 What isAndroid?**

**1.1.1 The Android operating system**

Android is an operating system based on the **Linux kernel**. The project responsible for developing the Android system is called the **Android Open-Source Project** (**AOSP**) and is primarily lead by **Google**.

The Android system supports background processing, provides a rich user interface library, supports 2-D and 3-D graphics using the **OpenGL-ES** (short OpenGL) standard and grants access to the file system as well as an embedded **SQLite database**.

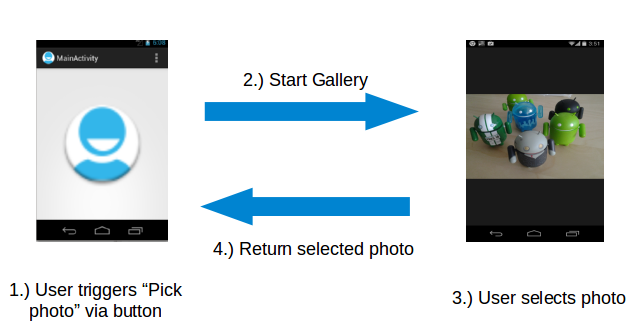
An Android application typically consists of different visual and non-visual components and **can reuse components of other applications**.

Android programmed in **JAVA** with **android configuration** which expanded the use of java classes

**1.1.2 Task**

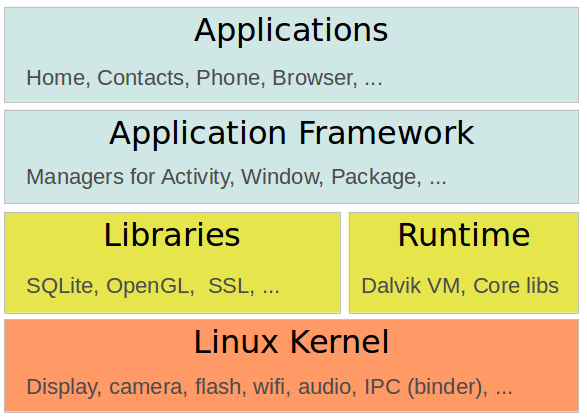
In Android the reuse of other application components is a concept known as task. An application can access other Android components to achieve a task. For example, from a component of your application you can trigger another component in the Android system, which manages photos for example, even if this component is not part of your application. In this component you select a photo and return to your application to use the selected photo.

Such a flow of events is depicted in the following graphic.



**1.1.3 Android platform components**

The Android system is a full software stack, which is typically divided into the four areas as depicted in the following graphic.



The levels can be described as:

**Applications:**

The Android Open-Source Project contains several default applications, like the Browser, Camera, Gallery, Music, Phone and more.

**Application framework:**

An API which allows high-level interactions with the Android system from Android applications.

**Libraries and runtime:**

The libraries for many common functions (e.g.: graphic rendering, data storage, web browsing, etc.) of the Application Framework and the Dalvik runtime, as well as the core Java libraries for running Android applications.

**Linux kernel:**

Communication layer for the underlying hardware.

The Linux kernel, the libraries and the runtime are encapsulated by the application framework. The Android application developer typically works with the two layers on top to create new Android applications.

**1.1.4 Google Play**

Google offers the **Google Play service**, a marketplace in which programmers can offer their Android applications to Android users. Customers use the Google Play application which allows them to buy and install applications from the Google Play service.

**Google Play** also offers an **update service**. If a programmer uploads a new version of his application to Google Play, this service notifies existing users that **an update is available** and allows them to install the update.

Google Play provides **access** to **services** and **libraries** for Android application programmers, too.

For example, it provides a service to use and display **Google Maps** and another to synchronize the application state between different Android installations. Providing these services via Google Play has the advantage that they are available for older Android releases and can be updated by Google without the need for an update of the Android release on the phone.

**1.2 Android Development Tools**

**1.2.1 Android SDK**

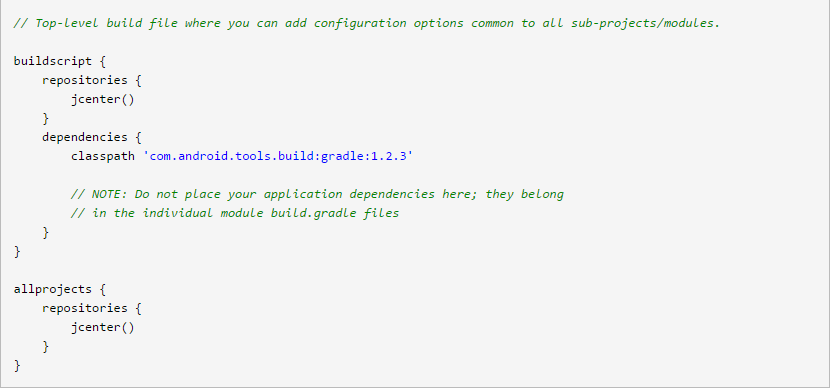
The Android Software Development Kit (Android SDK) contains the necessary tools to create, compile and package Android applications. Most of these tools are command line based. The primary way to develop Android applications is based on the Java programming language.

**1.2.2 Android debug bridge (adb)**

The Android SDK contains the Android debug bridge (adb), which is a tool that allows you to connect to a virtual or real Android device, for the purpose of managing the device or debugging your application.

**1.2.3 Gradle and the Android plug-in for Gradle**

The Android tooling uses Gradle as build system. The Android team provides a Gradle plug-in for build Android applications which is entered in the build.gradle file in the top root of the Android project. It typically looks like the following, please note that the version might be different in your case.



**1.2.4 Android Developer Tools and Android Studio**

Google provides an **Interface Development Editor** (**IDE**) called **Android Studio** as the preferred development environment for creating Android applications. This IDE is based on the **IntelliJ** **IDE**.

The Android tools provide specialized editors for Android specific files. Most of Android's configuration files are based on **XML**. In this case these editors allow you to switch between the XML representation of the file and a structured user interface for entering the data.

**1.2.5 Android RunTime (ART)**

Android 5.0 uses the Android RunTime (ART) as runtime for all Android applications.

ART uses Ahead of Time compilation. During the deployment process of an application on an Android device, the application code is translated into machine code. This results in approx. 30% larger compile code, but allows faster execution from the beginning of the application.

This also saves battery life, as the compilation is only done once, during the first start of the application.

The dex2oat tool takes the .dex file created by the Android tool change and compiles that into an Executable and Linkable Format (ELF file). This file contains the dex code, compiled native code and meta-data. Keeping the .dex code allows that existing tools still work.

The garbage collection in ART has been optimized to reduce times in which the application freezes.

**1.2.6 How to develop Android applications**

Android applications are primarily written in the Java programming language.

During development the developer creates the Android specific configuration files and writes the application logic in the Java programming language.

The Android tooling converts these application files, transparently to the user, into an Android application. When developers trigger the deployment in their IDE, the whole Android application is compiled, packaged, deployed and started.

**1.2.7 Conversion process from source code to Android application**

The Java source files are converted to Java class files by the Java compiler.

The Android **Software Development Kit** (**SDK**) contains a tool called dx which converts Java class files into a .dex (Dalvik Executable) file. All class files of the application are placed in this .dex file. During this conversion process redundant information in the class files are optimized in the .dex file.

For example, if the same String is found in different class files, the .dex file contains only one reference of this String.

These .dex files are therefore much smaller in size than the corresponding class files.

The .dex file and the resources of an Android project, e.g., the images and XML files, are packed into an .apk (Android Package) file. The program aapt (Android Asset Packaging Tool) performs this step.

The resulting .apk file contains all necessary data to run the Android application and can be deployed to an Android device via the adb tool.

**1.3 Android Security and permissions**

**1.3.1 Security concept in Android**

The Android system installs every Android application with a unique user and group ID. Each application file is private to this generated user, e.g., other applications cannot access these files. In addition, each Android application is started in its own process.

Therefore, by means of the underlying Linux kernel, every Android application is isolated from other running applications.

If data should be shared, the application must do this explicitly via an Android component which handles the sharing of the data, e.g., via a service or a content provider.

**1.3.2 Permission concept in Android**

Android contains a permission system and predefines permissions for certain tasks. Every application can request required permissions and also define new permissions. For example, an application may declare that it requires access to the Internet.

Permissions have different levels. Some permissions are automatically granted by the Android system, some are automatically rejected. In most cases the requested permissions are presented to the user before installing the application. The user needs to decide if these permissions shall be given to the application.

If the user denies a required permission, the related application cannot be installed. The check of the permission is only performed during installation, permissions cannot be denied or granted after the installation.

An Android application declares the required permissions in its AndroidManifest.xml configuration file. It can also define additional permissions which it can use to restrict access to certain components.

**1.4 Programming Software for Android**

There are two (IDE)s to implement android Applications

1\_ Android Studio

2\_ Eclipse

We will use **Android studio** as Android Studio is now **the official IDE** for Android, so you should migrate to Android Studio to receive all the **latest IDE updates**

you can check that on:

http://developer.android.com/sdk/index.html

**Installation of Android Studio**

**1- System requirements**

Windows

\* Microsoft® Windows® 8/7/Vista/2003 (32 or 64-bit)

\* 2 GB RAM minimum, 4 GB RAM recommended

\* 400 MB hard disk space

\* At least 1 GB for Android SDK, emulator system images, and caches

\* 1280 x 800 minimum screen resolution

\* Java Development Kit (JDK) 7

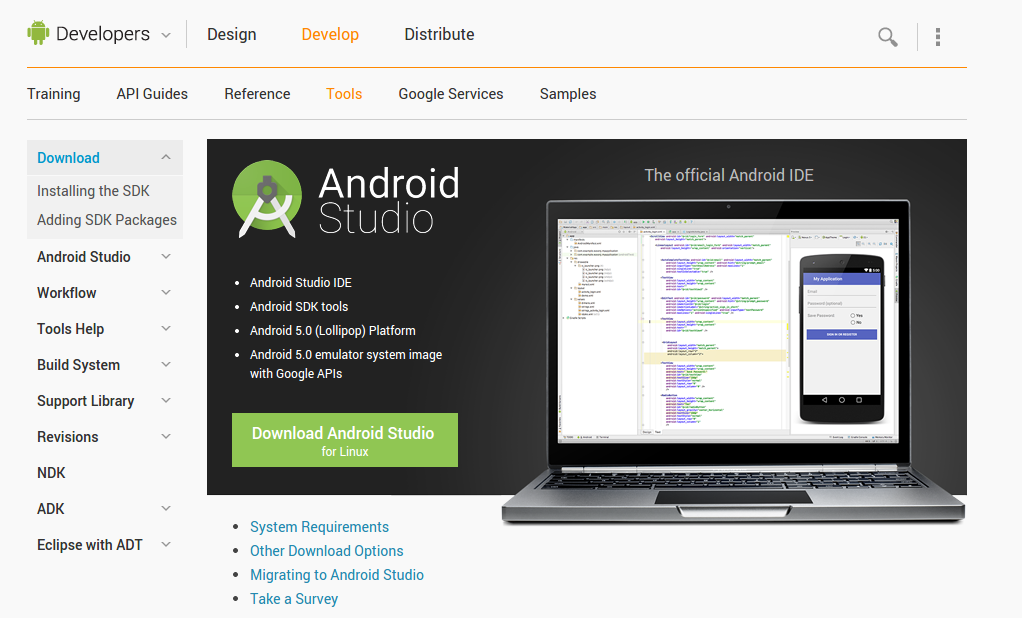
\* Optional for accelerated emulator: Intel® \* processor with support for Intel® VT-x, Intel® EM64T (Intel® 64), and Execute Disable (XD) Bit functionality.

**2- Download Android Studio**

Download Android Studio from

http://developer.android.com/sdk/index.html

The download comes in two flavors, SDK Tools only and Android Studio Packages. You want to download the Android Studio Package for your operation system.



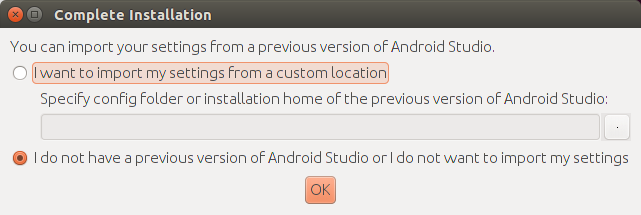
**3- Installation of Android Studio**

Installation for Windows is simple, just launch the .exe you downloaded. On Max OSX drag and drop Android Studio into the Applications folder.

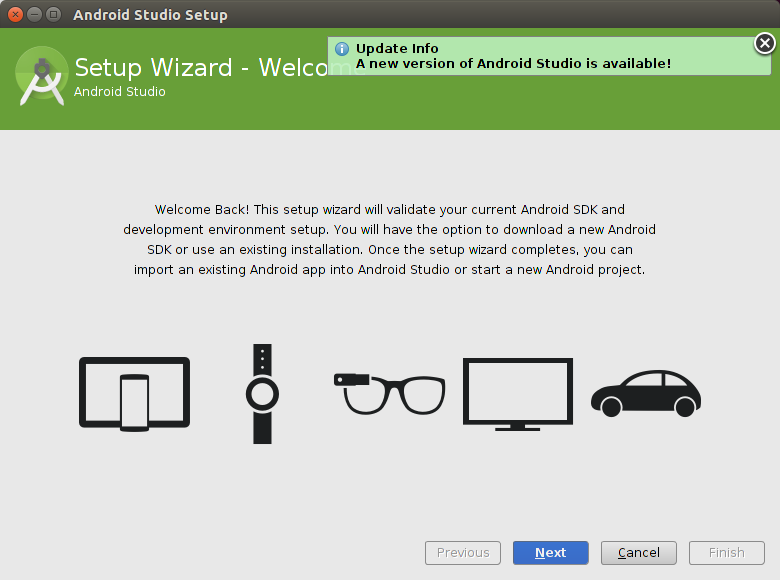
On Linux unpack the downloaded ZIP file into an appropriate location for your applications. To launch Android Studio, navigate to the android-studio/bin/ directory in a terminal and execute studio.sh.

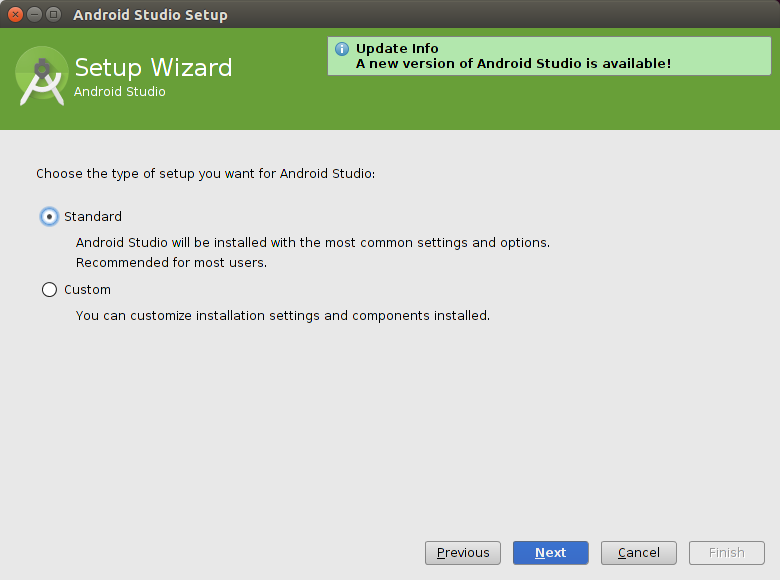
**4- Configuration**

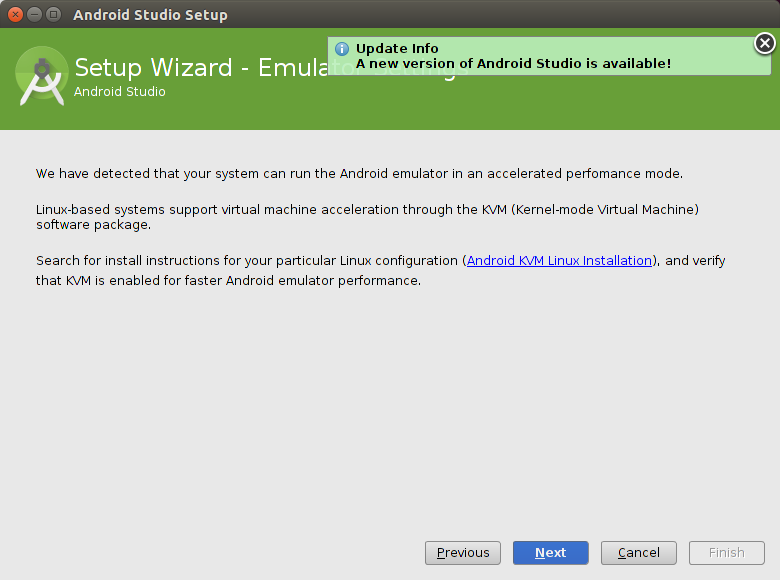
The first time you start Android Studio you can select if you want to import your setting from an existing installation.



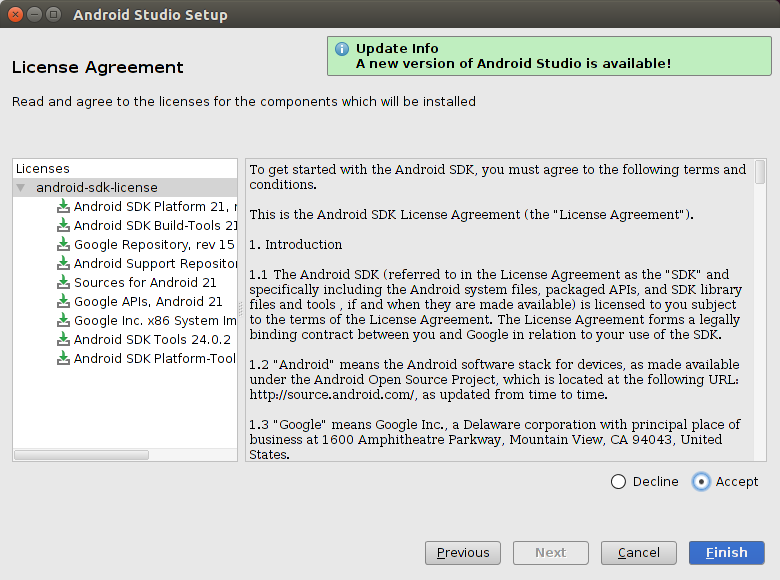
Afterwards click through the setup guide.







Once you reach the last page, press the Finish button.



**5- Downloading and Using the Android SDK manager**

The Android SDK Manager allows you to install specific versions of the Android **API**.

The Android SDK Manager allows you to install and delete Android packages.

**Chapter 2**

**Introduction to Agile**

**2\_1 SW Engineering**

**2\_2** **Traditional SW Development Methodologies**

**2\_3 Agile**

**2\_4 Agile Methods**

**2.1 SW Engineering**

SW Engineering is a set of tools, methods and practices used to produce SW product.

**2.2 Traditional SW Development Methodologies**

**2.2.1 Types of Traditional SW Development Methodologies**

1. Waterfall Model
2. Spiral Model
3. Incremental Model
4. Iterative Model
5. V-Model

**2.2.2 Traditional Methods Drawbacks**

* developers can’t go back to make changes.
* testing is only done at the end with a very high delay.
* very high maintenance effort.
* Low customer communication and satisfaction.
* very high documentation effort.
* Requirements should be fully understood.
* Risk analysis requires highly specific domain expertise.

**2.3 Agile**

**2.3.1 What is Agile?**

Agile software development methodology is an incremental software development methodology, that provides a fast and simple way of software developing based on the continuous customer involvement which grantee better understanding for project requirement, project quality and customer satisfaction.

Agile methods focus on the following:

* + Customer satisfaction.
  + To Move Quickly.
  + To Be Creative.
  + To Adapt Changes.
  + To Work on What on Hand.

**2.3.2 Agile Manifesto**

A formal proclamation of 4 key values and 12 principles to guide an iterative and people-centric approach to software development.

Agile Values and Agile Principles

**2.3.2.1 Agile Values**

* Individuals and interactions over processes and tools
* Working software over comprehensive documentation
* Customer collaboration over contract negotiation
* Responding to change over following a plan

**2.3.2.1 Agile Principles**

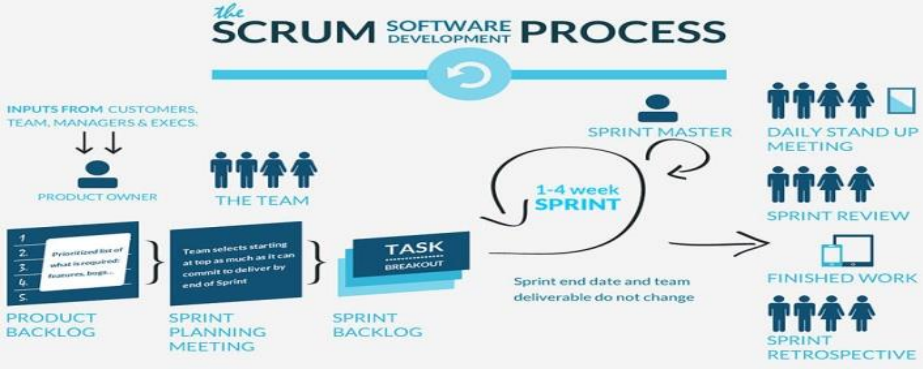
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

**2.4 Agile**

**2.4.1 Extreme Programming (XP)**

* XP is the most successful method of developing agile software because of its focus on customer satisfaction.
* XP requires maximum customer interaction to develop the software. It divides the entire software development life cycle into several number of short development cycles.
* It welcomes and incorporates changes or requirements from the customers at any phase of the development life cycle.
* XP has introduced many new things for developers like pair programming, extensive code review, code refactoring and open workspace

**2.4.2 Scrum**



**2.4.2.1 Principles**

* Burndown chart: shows the work remaining within the sprint. Used both to track sprint progress and to decide when items must be removed from the sprint backlog and deferred to the next sprint.
* Product backlog. Product backlog is the complete list of requirements—including bugs, enhancement requests, and usability and performance improvements—that are not currently in the product release.
* ScrumMaster. is the person responsible for managing the Scrum project. Sometimes it refers to a person who has become certified as a ScrumMaster by taking ScrumMaster training.
* Sprint backlog. Sprint backlog is the list of backlog items assigned to a sprint, but not yet completed. In common practice, no sprint backlog item should take more than two days to complete. The sprint backlog helps the team predict the level of effort required to complete a sprint.
* product owner: handle communication with customer.

**2.4.2.2 Daily Scrum**

1. Leaded by scrum master.
2. 15 minutes duration.
3. each team member should answer three questions
   1. What did I accomplish yesterday?
   2. What will I do today?
   3. What obstacles are impeding my progress?

**2.4.2.3 User story**

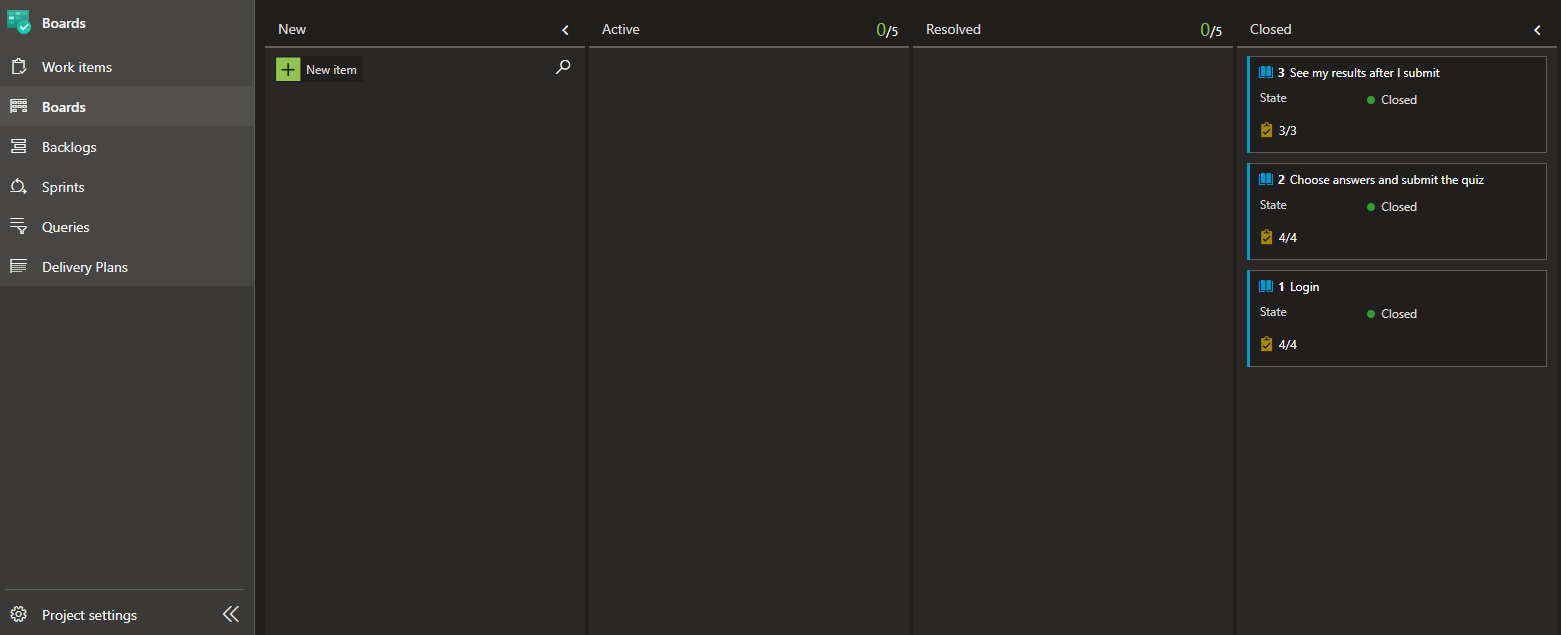
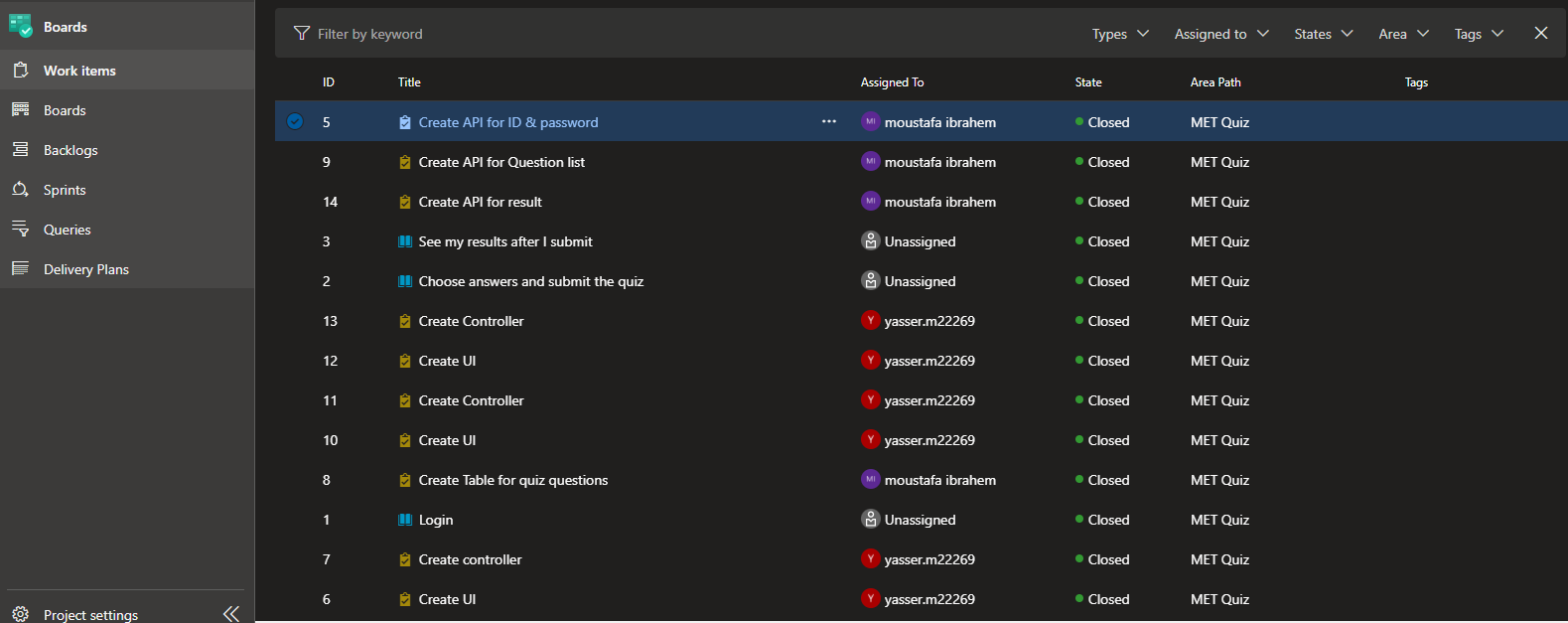
* User Story is a small (actually, the smallest) piece of work that represents some value to an end user and can be delivered during a sprint.
* As a [type of user], I want [an action] so that [a benefit/a value]
* Great User Stories always fit the INVEST set of criteria:
  + Independent – they can be developed in any sequence and changes to one UserStory don’t affect the others.
  + Negotiable – it’s up for the team to decide how to implement them; there is no rigidly fixed workflow.
  + Valuable – each User Story delivers a detached unit of value to end users.
  + Estimable – it’s quite easy to guess how much time the development of a UserStory will take.
  + Small – it should go through the whole cycle (designing, coding, testing) during one sprint.
  + Testable – there should be clear acceptance criteria to check whether a User Story is implemented appropriately.

Example for UserStory for Uber app:

* As a driver, I want to block badly behaved passengers so they are never shown me again.
* As a passenger, I want to link the credit card to my profile so that I can pay for a ride faster, easier and without cash.
* As a driver, I want to add photos of my car in my profile so that I can attract more users.
* As a passenger, I want several available drivers to be displayed so that I can choose the most suitable option for me.
* An acceptance criterion is a set of conditions that are used to confirm when a Story is completed.

Each UserStory should have priority – weight When Development starts each user store is broken into tasks.

**2.4.2.4 Scrum Board**

****

**Chapter 3**

**Requirement Definition Phase**

**3\_1 Description for quiz app  
3\_2 UseCase3.1 Description for quiz app**

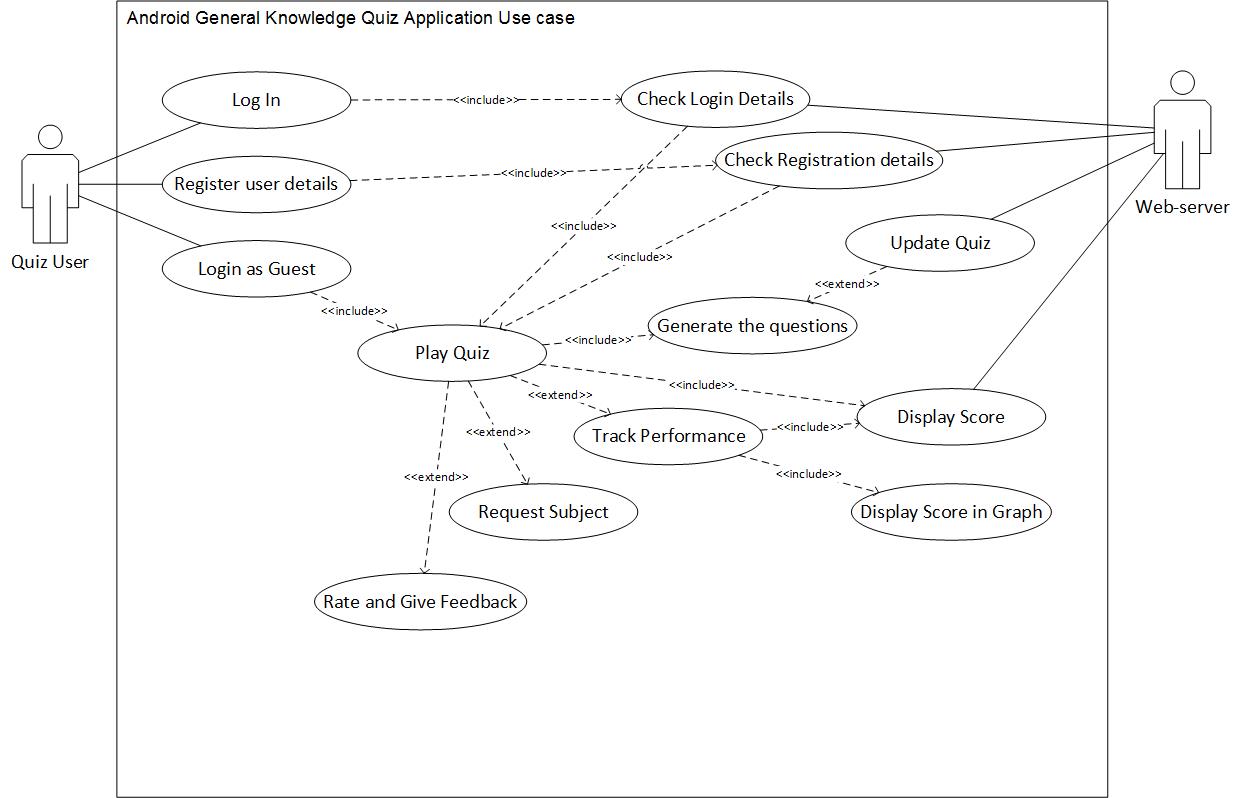
This work deals with the development of an android-based multiple-choice question examination system, namely: METQ.

This app is developed for educational purposes, allowing users to test multiple choice questions for different exams that are taken at the university level.

The main objective of the application is to enable users to practice exams in subjects that are conducted by the university, focusing on the field of computer science.

This quiz application includes three main modules, namely (i) computer science, (ii) Systems and information management, and (iii) engineering. The computer science and engineering modules contain various types of subcategories.

**3.2 UseCase**

****

**Chapter 4**

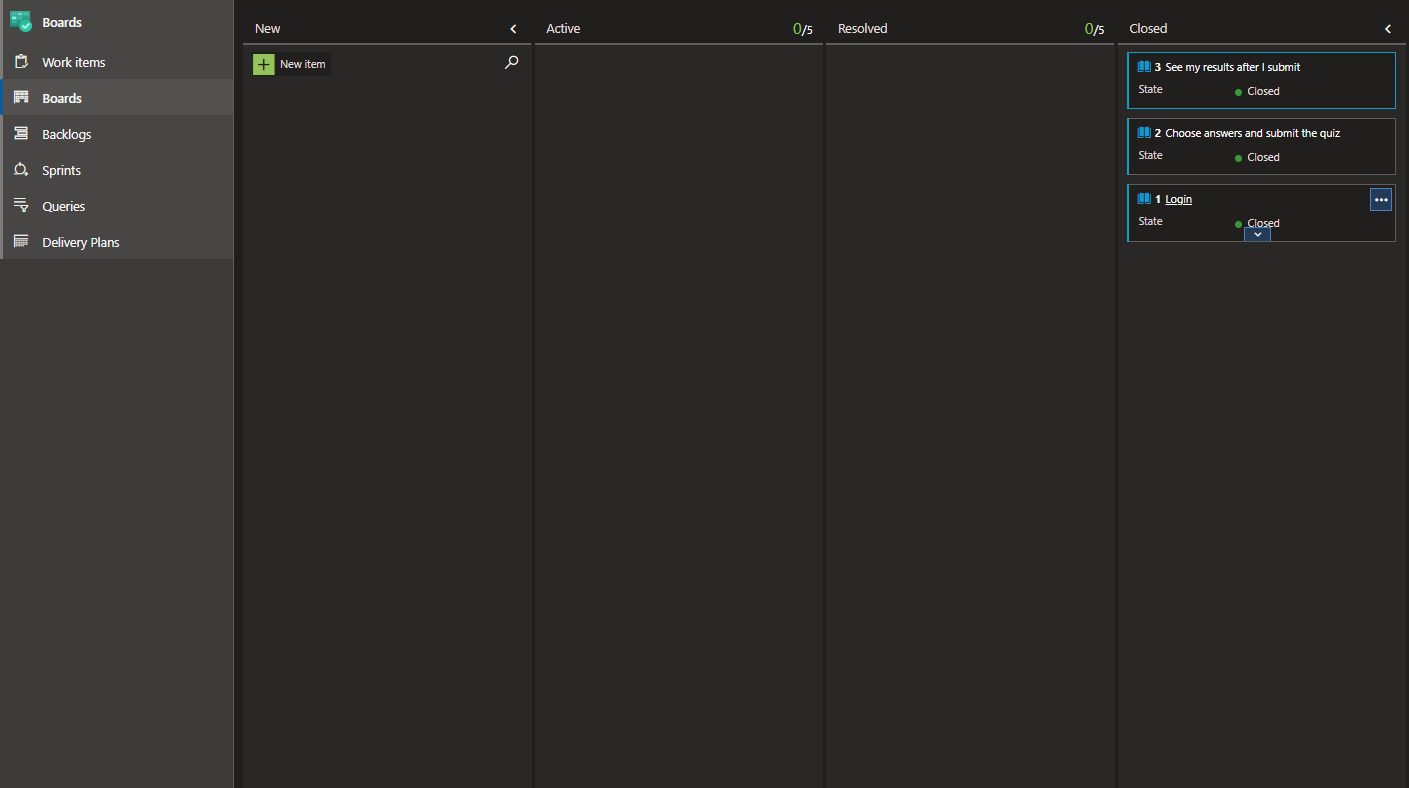
**Development Phase**

**4\_1 Azure  
4\_2 Implementation**

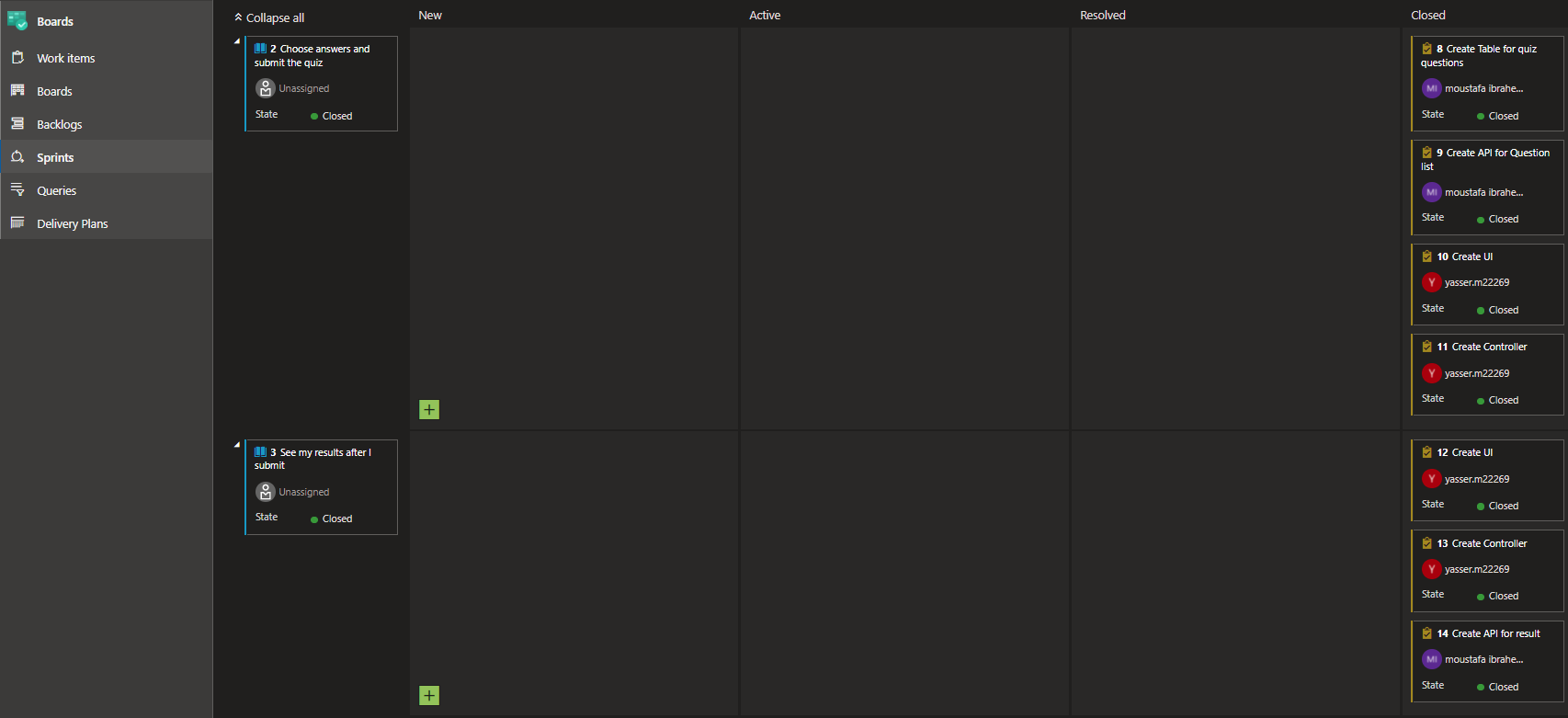
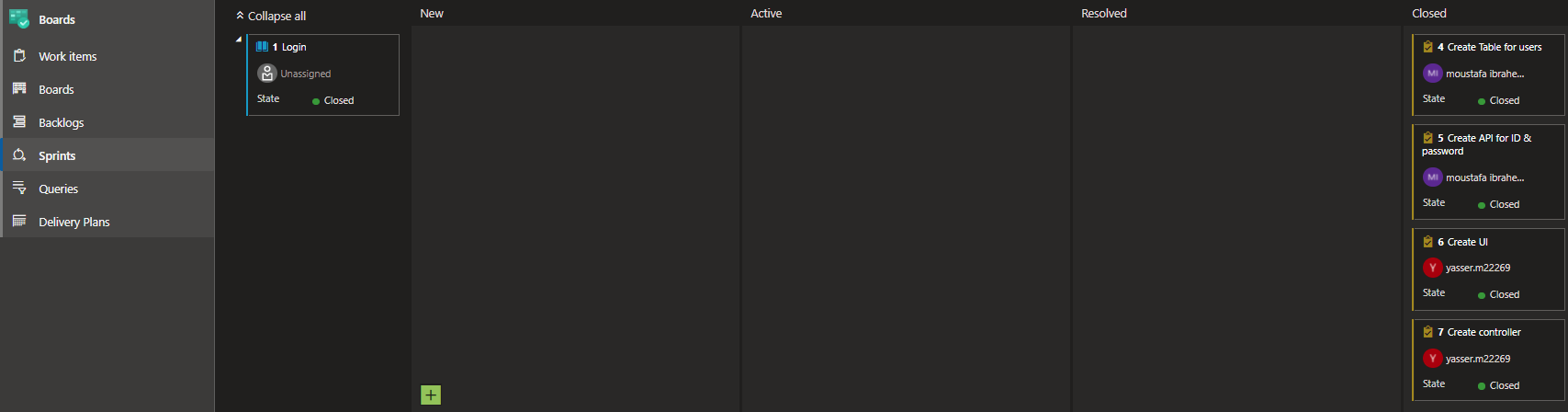
**4.1 Azure**

Work has been done on Azure to achieve agile to the fullest by dividing the project into several sprints and dividing the user stories on them and dividing them into tasks and determining the work of each of them on the programmers inside the project.

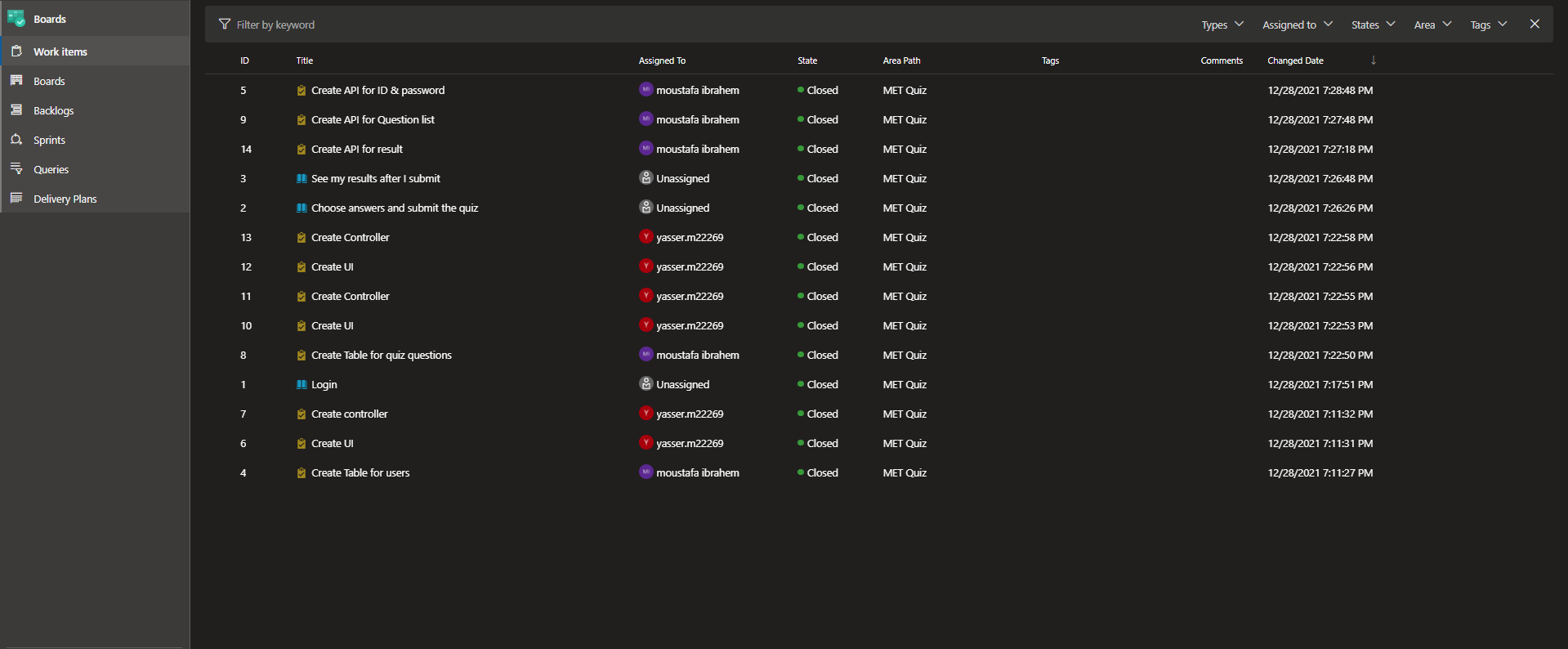
**4.1.1 Boards**

****

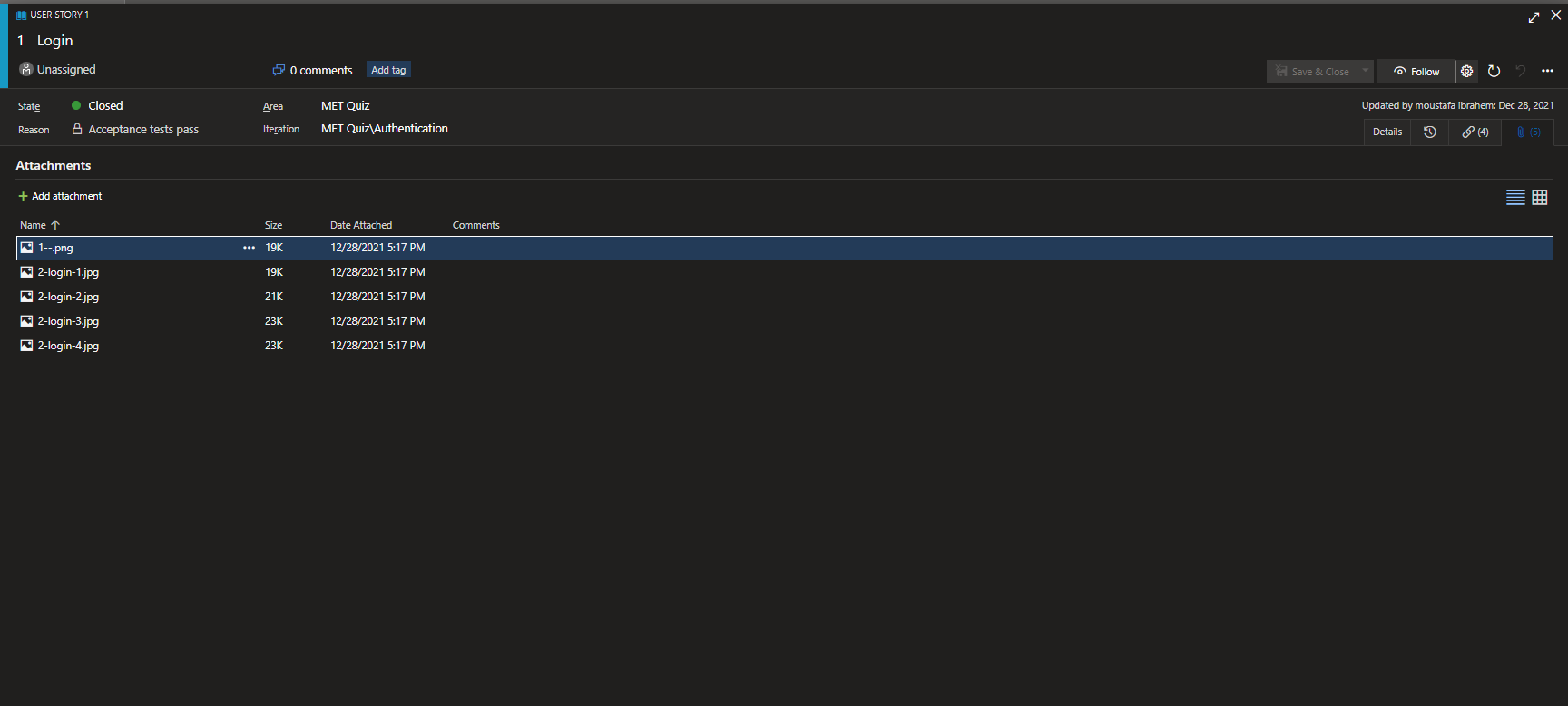
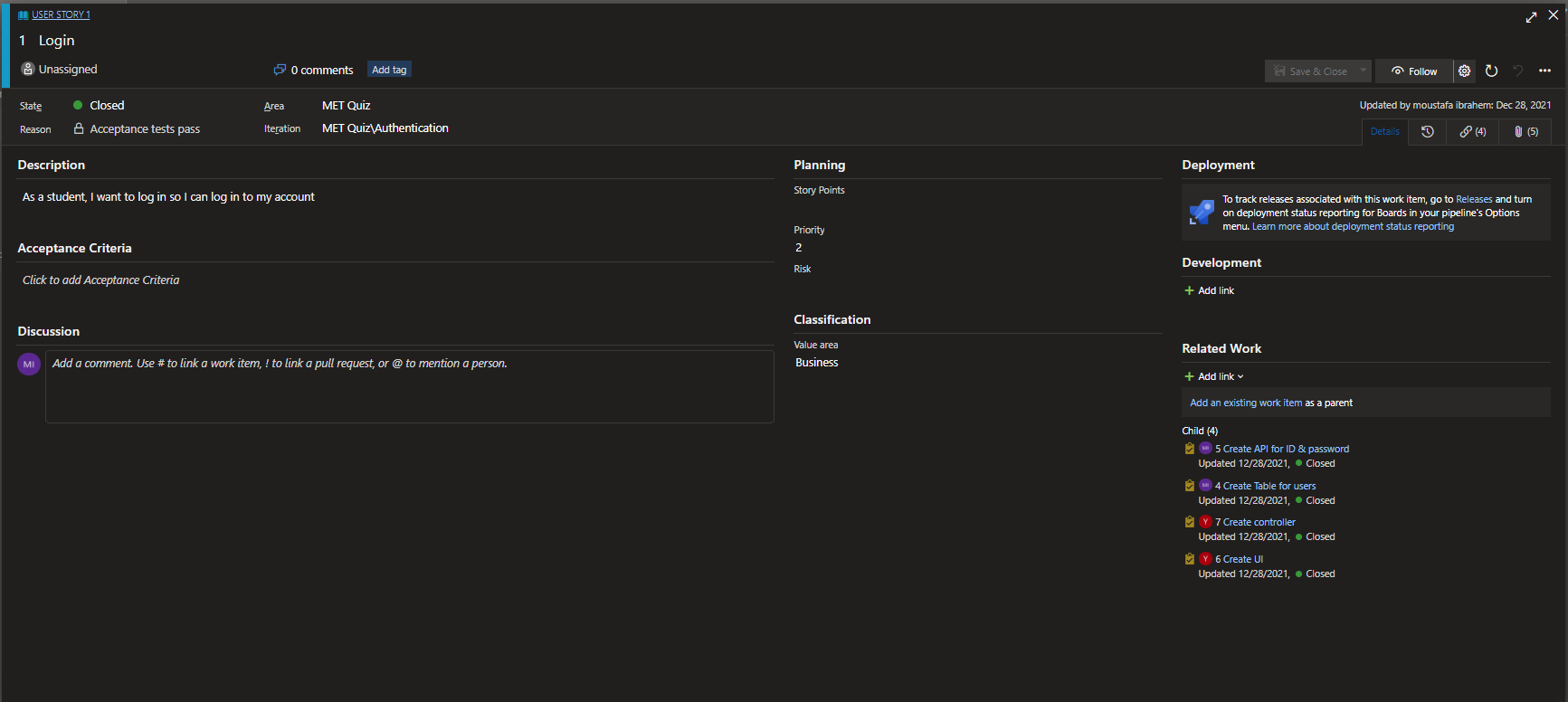
**4.1.2 Sprints**



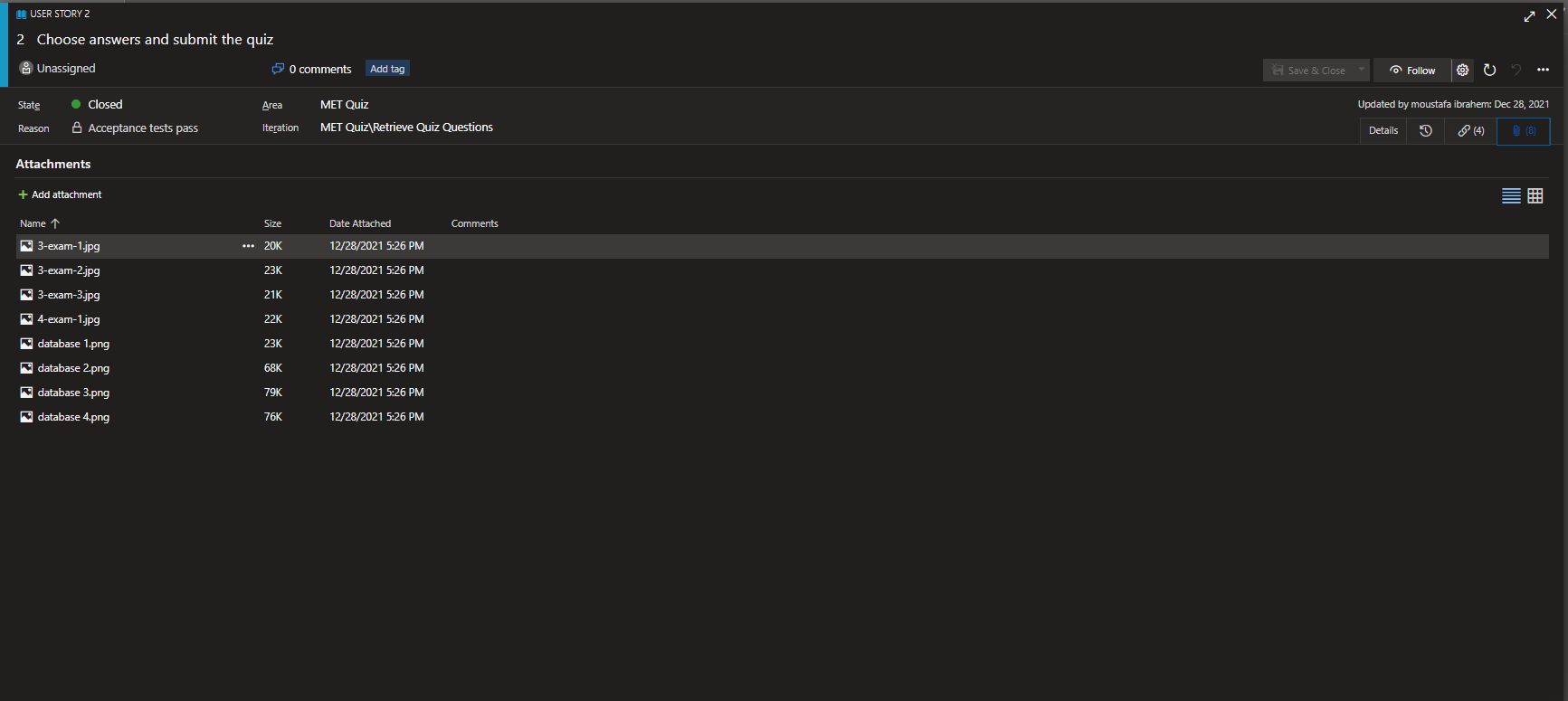
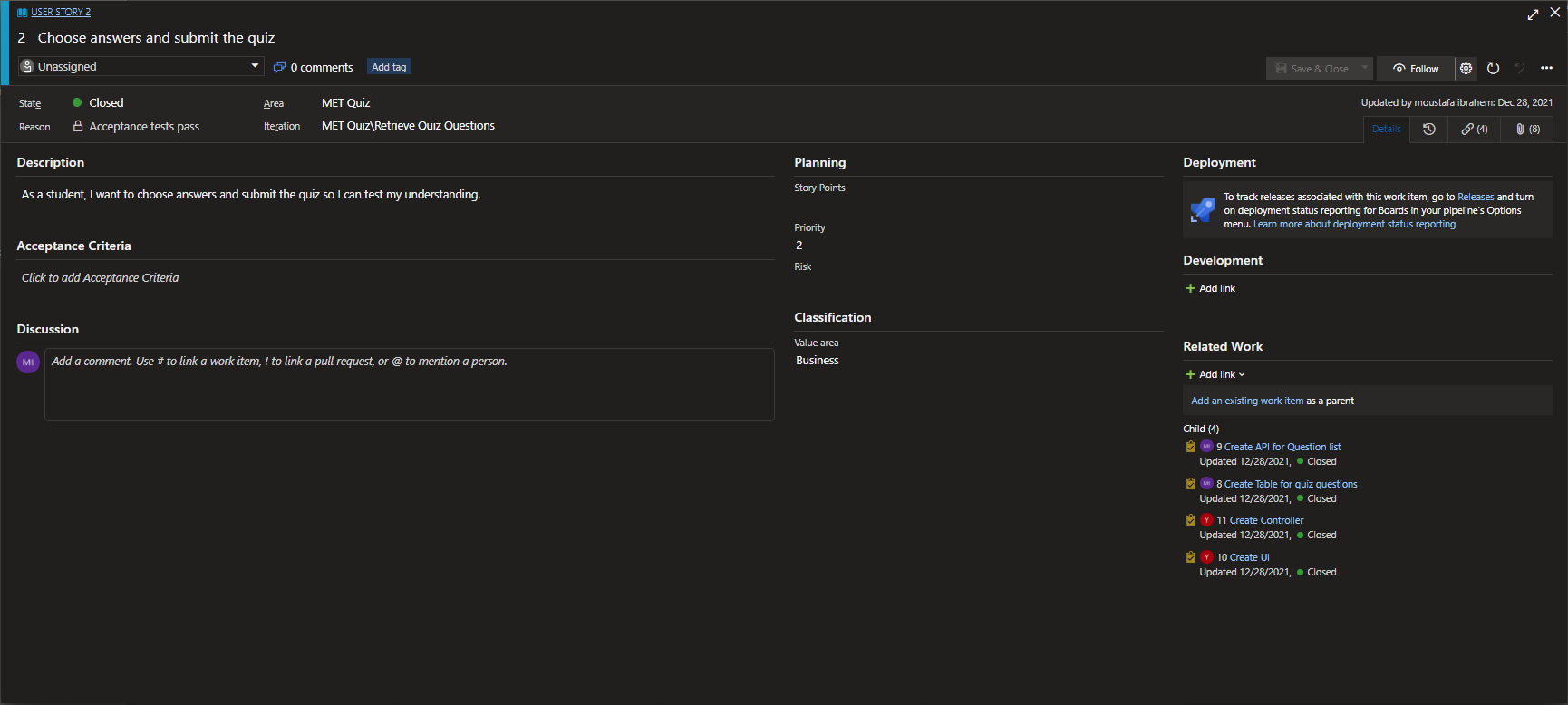
**4.1.3 Work Items**

****

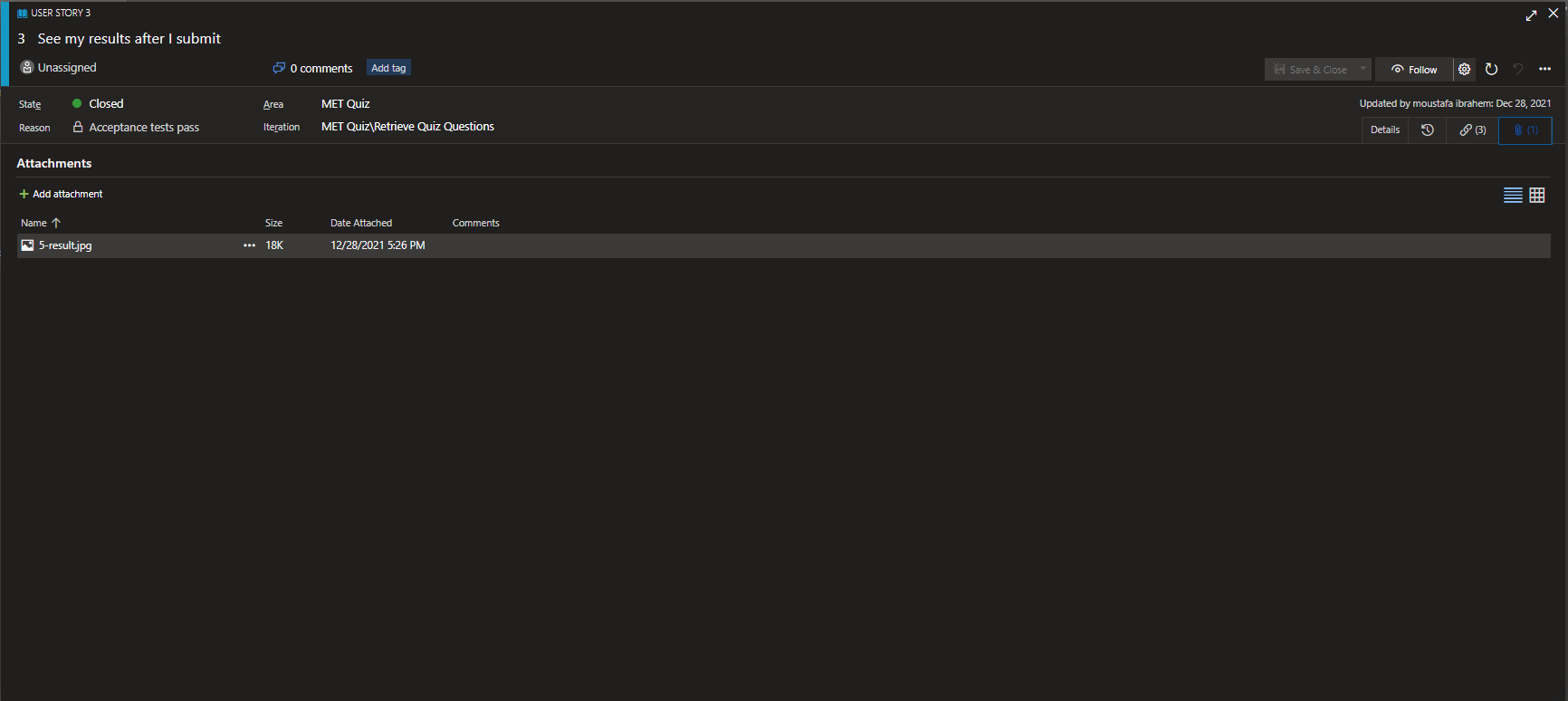
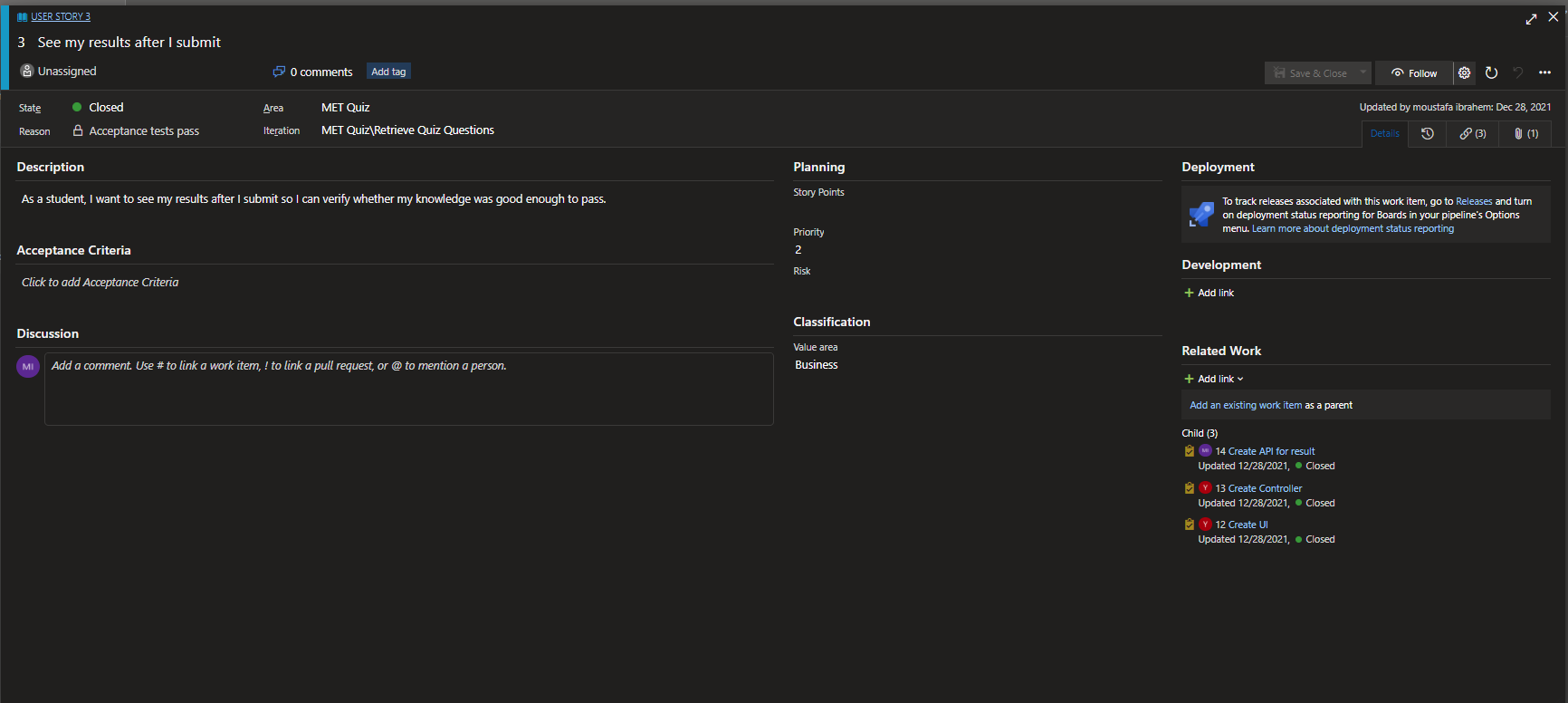
**login**

****

**Choose answers and submit the quiz**

****

**See my results after I submit**

****

And through the last steps, I made work among programmers better and faster in work and productivity.

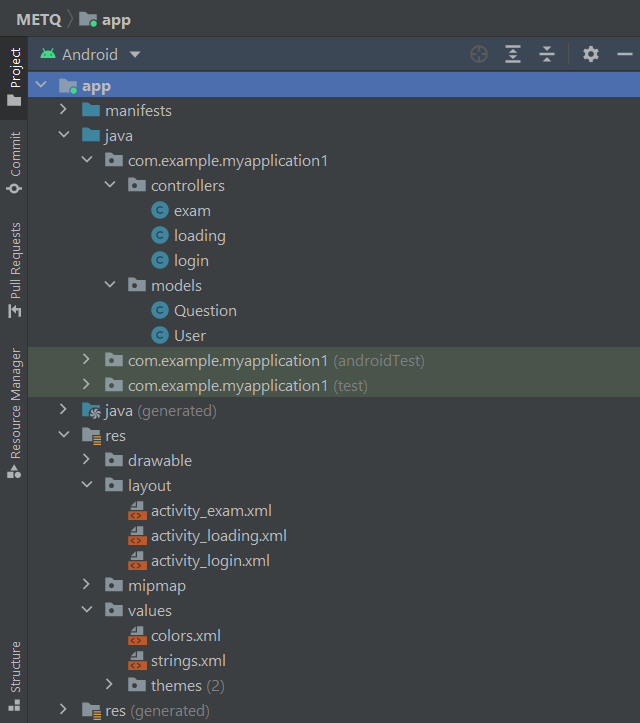
**4.2 Implementation**

The pattern has been followed in the project's design which is the MVC pattern

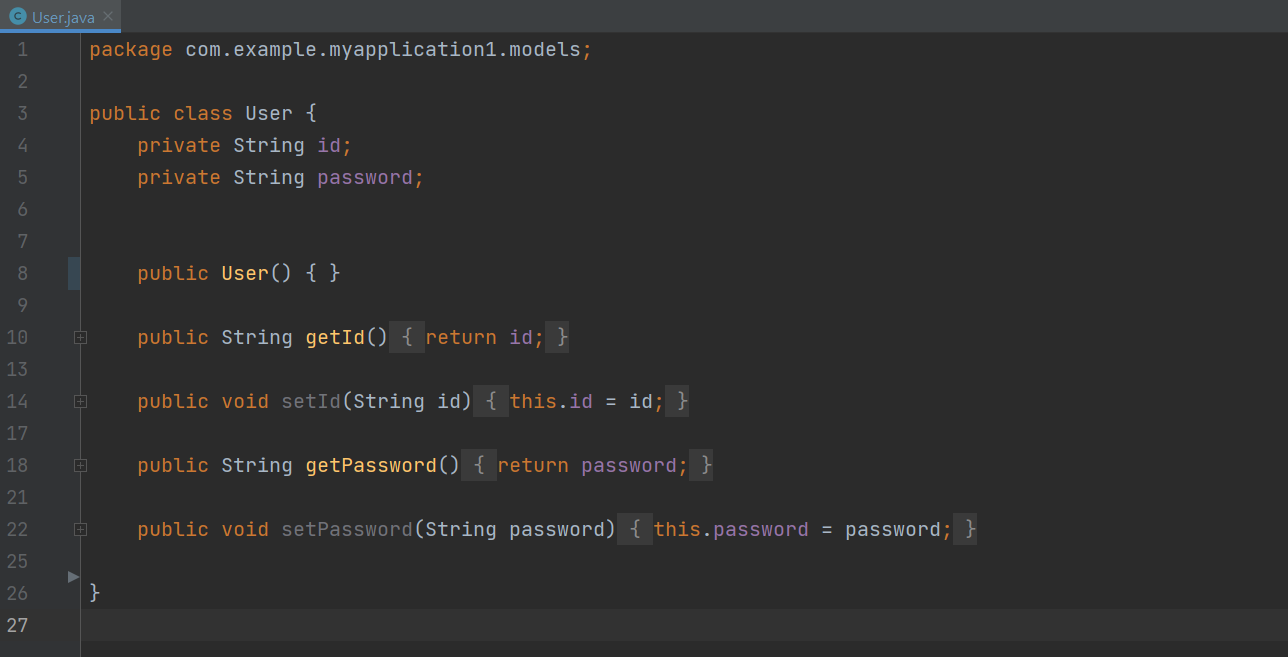
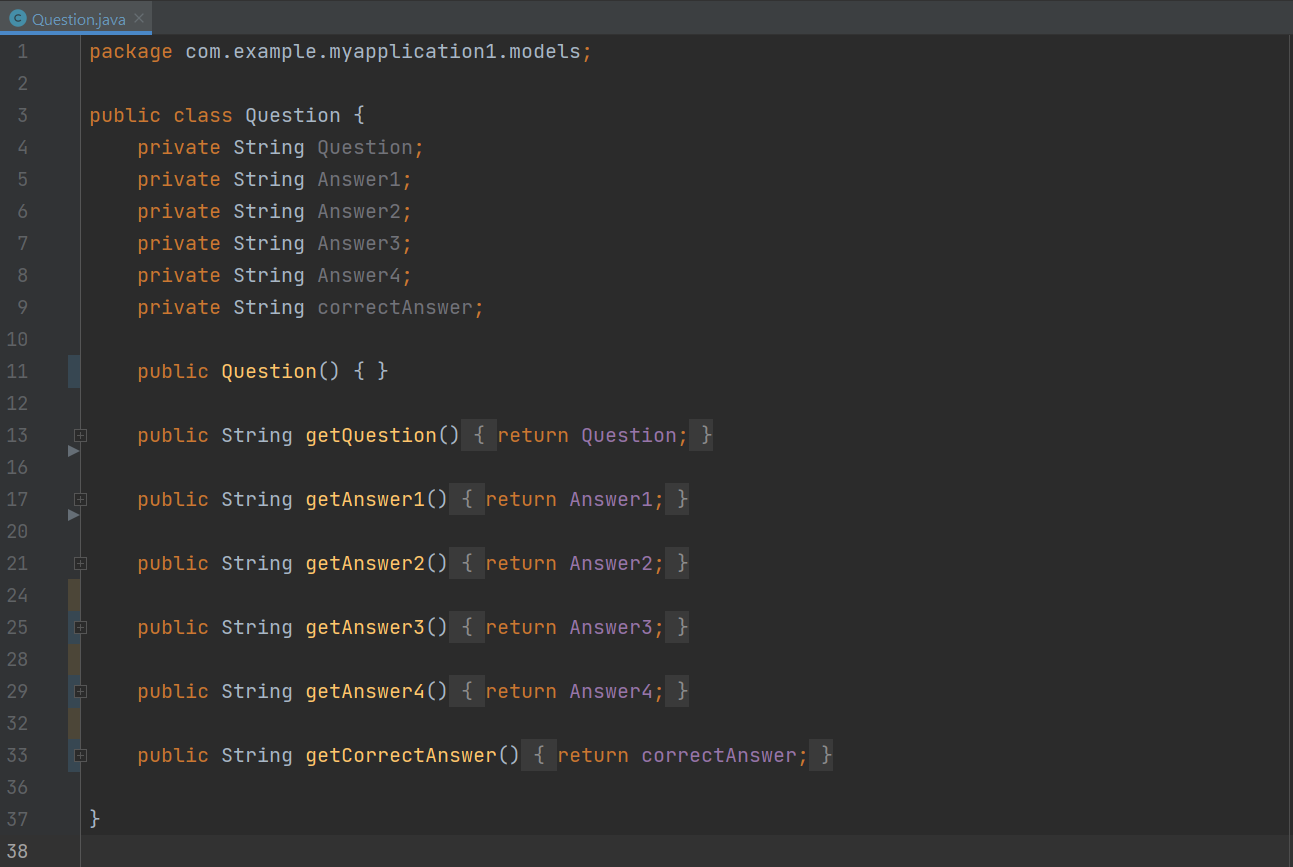
MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

* Model - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
* View - View represents the visualization of the data that model contains.
* Controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

**MVC Pattern:**



**Models:**

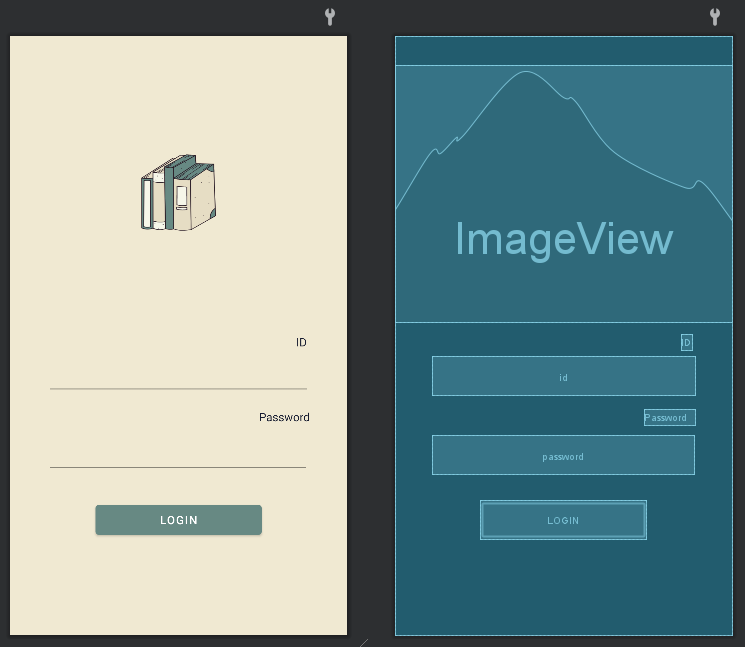


**views:**

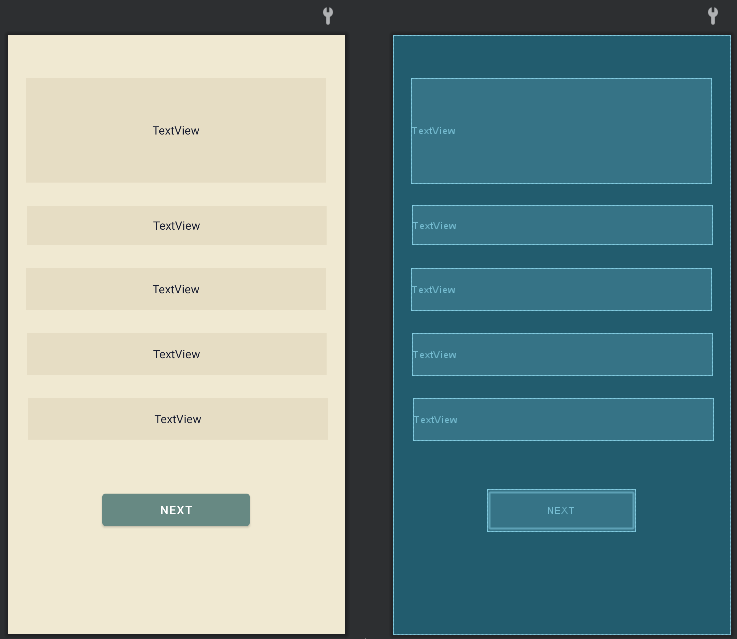
Loading ui



Login ui

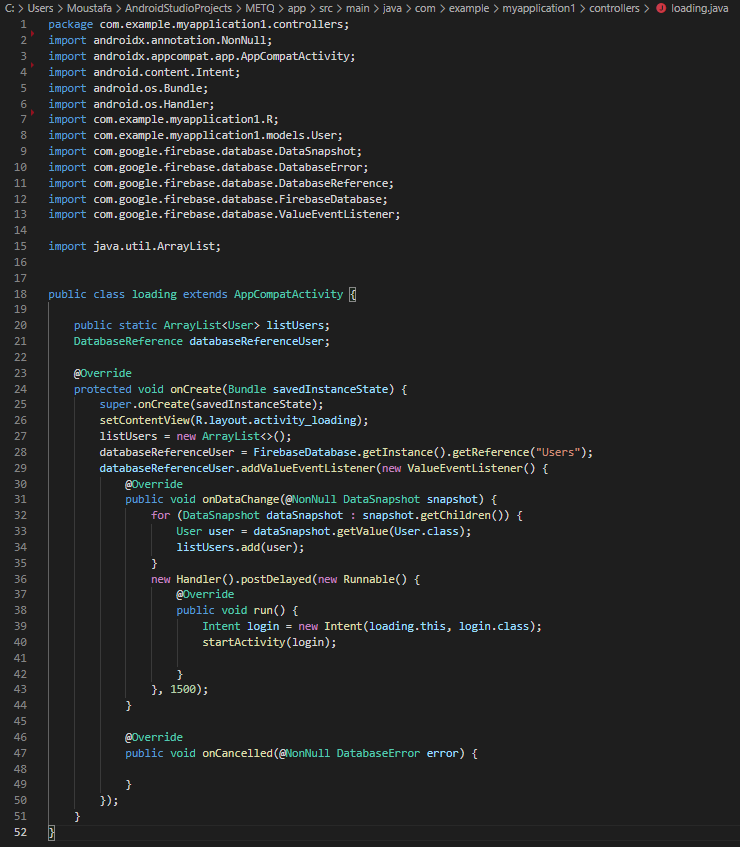


Exam ui



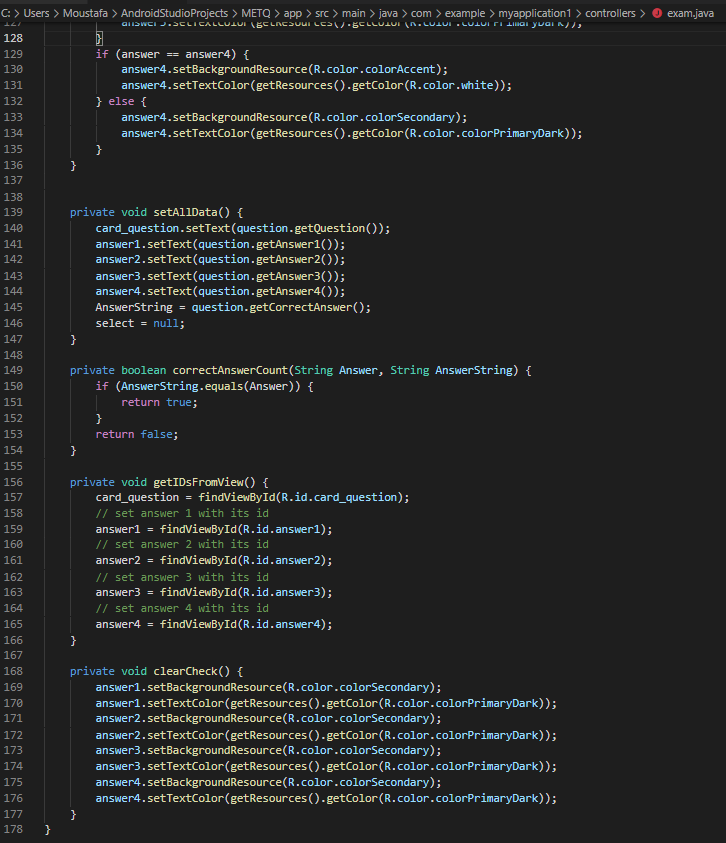
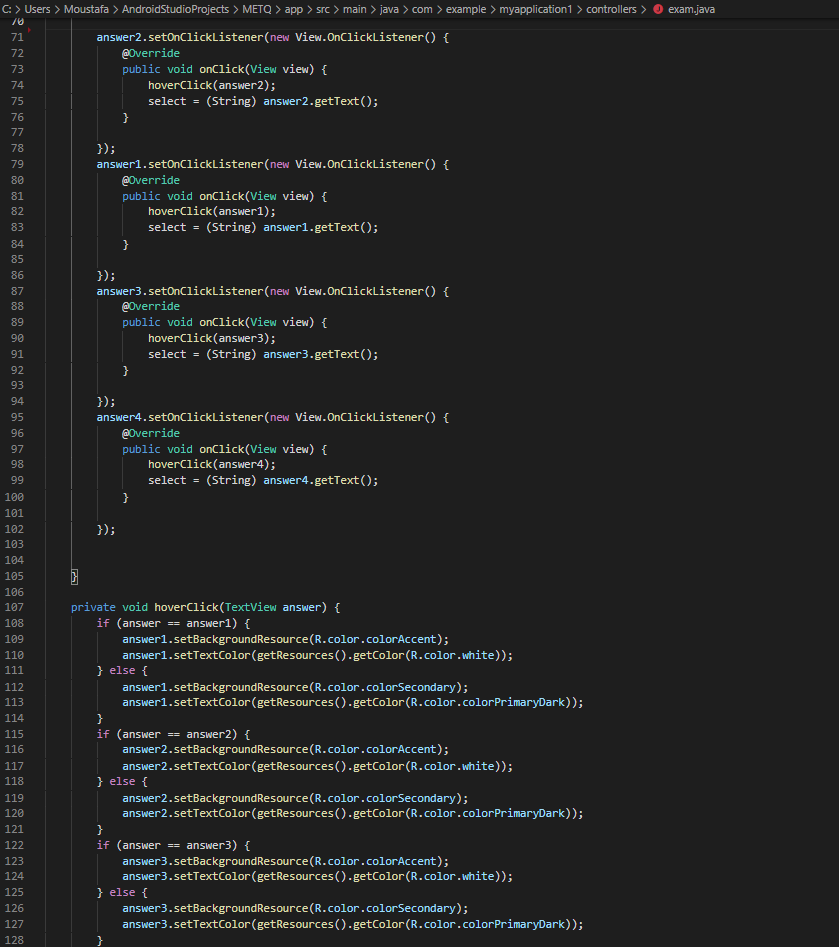
**controllers:**

Loading

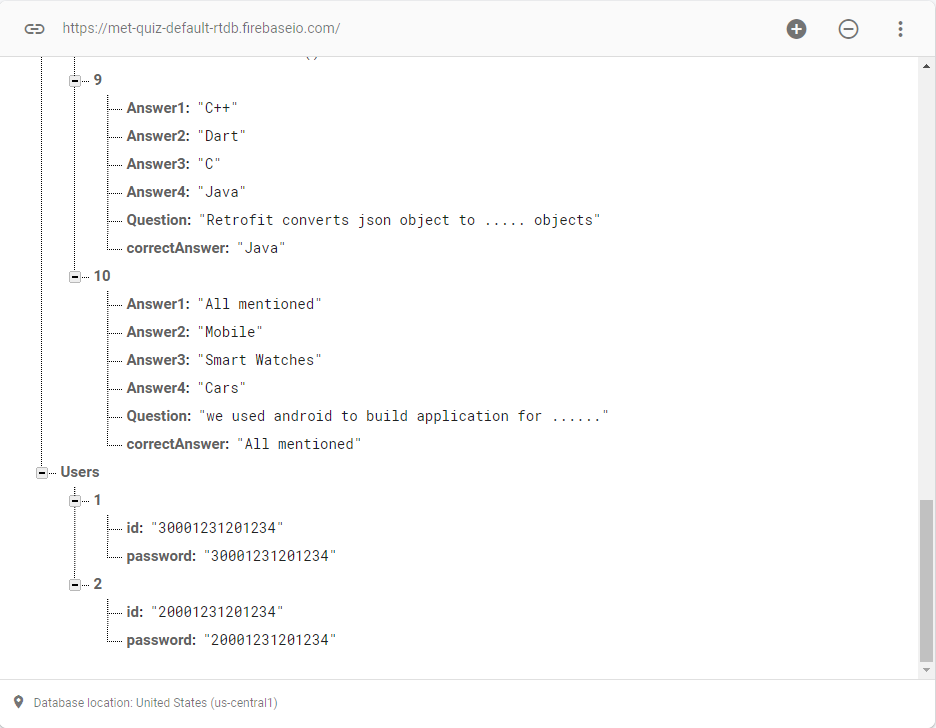
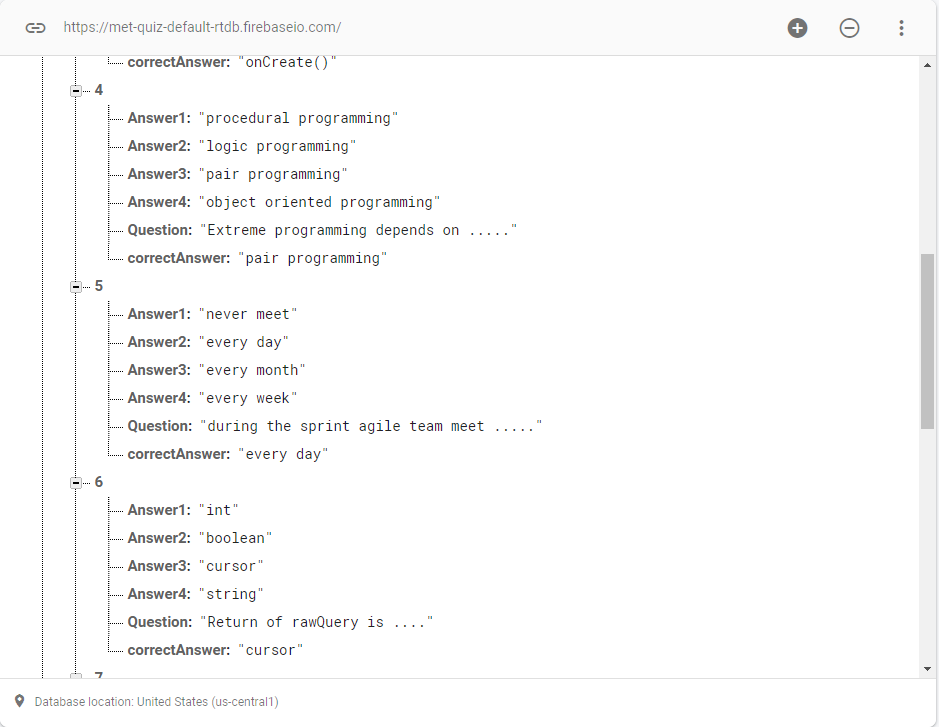
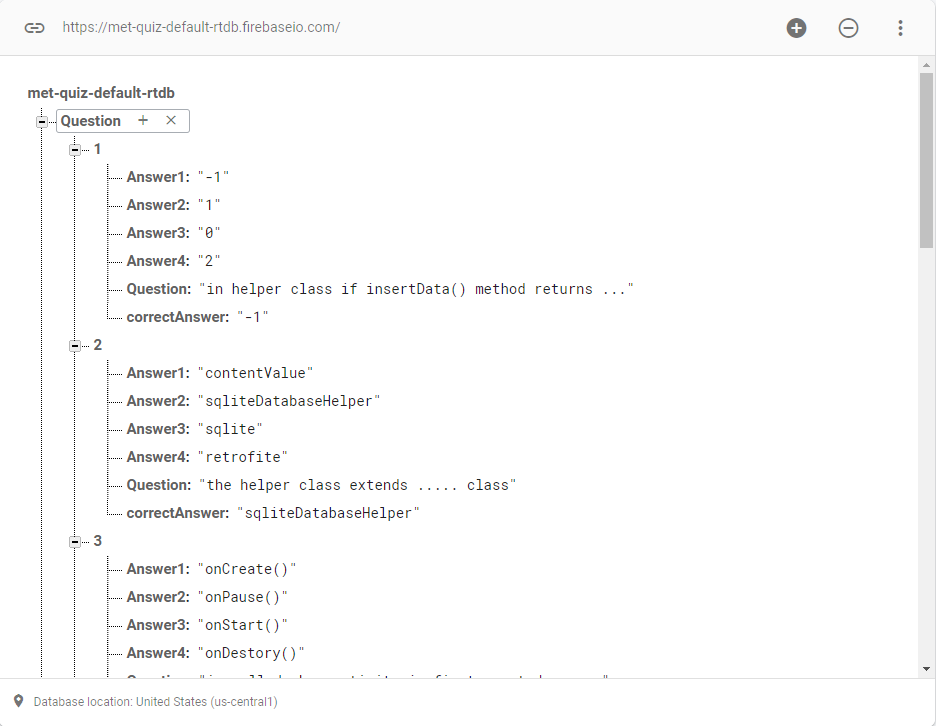
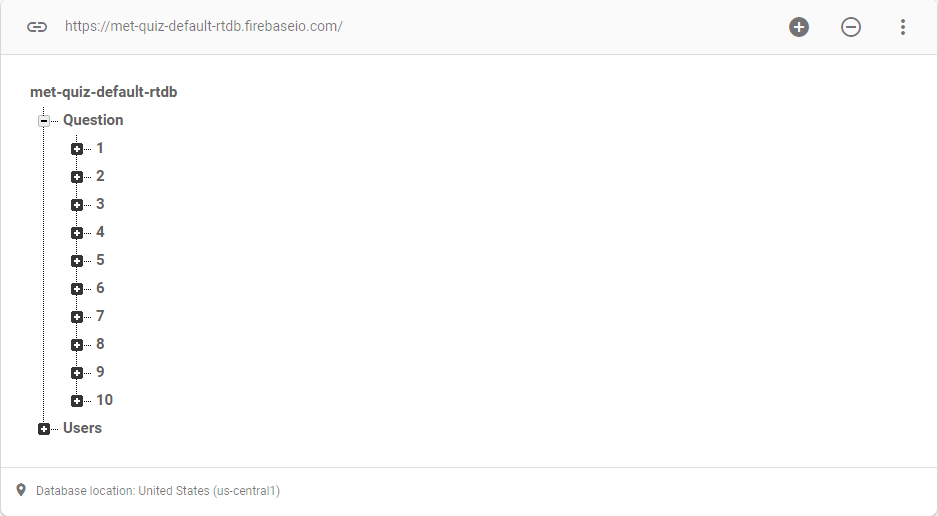
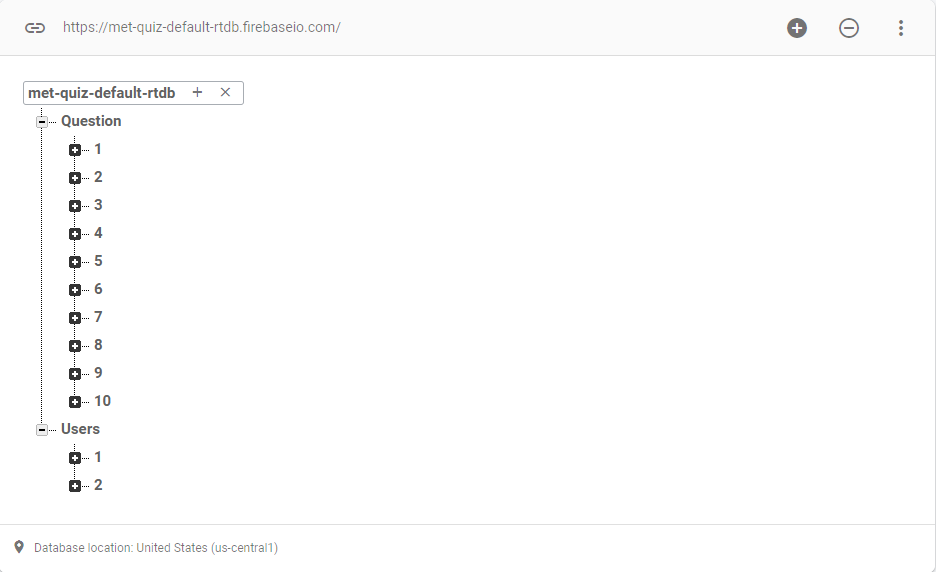
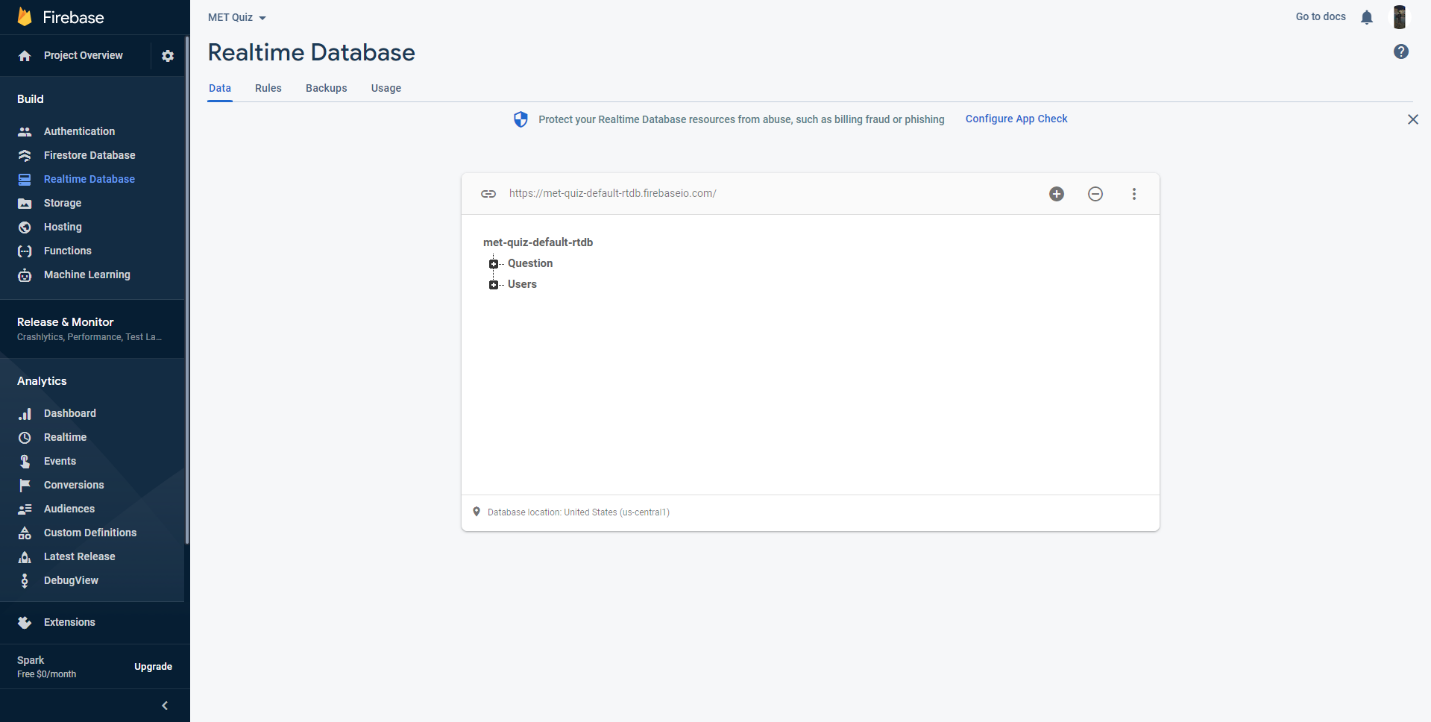


Loging



Exam

And we use and here we were using firebase because the database keeps Realtime



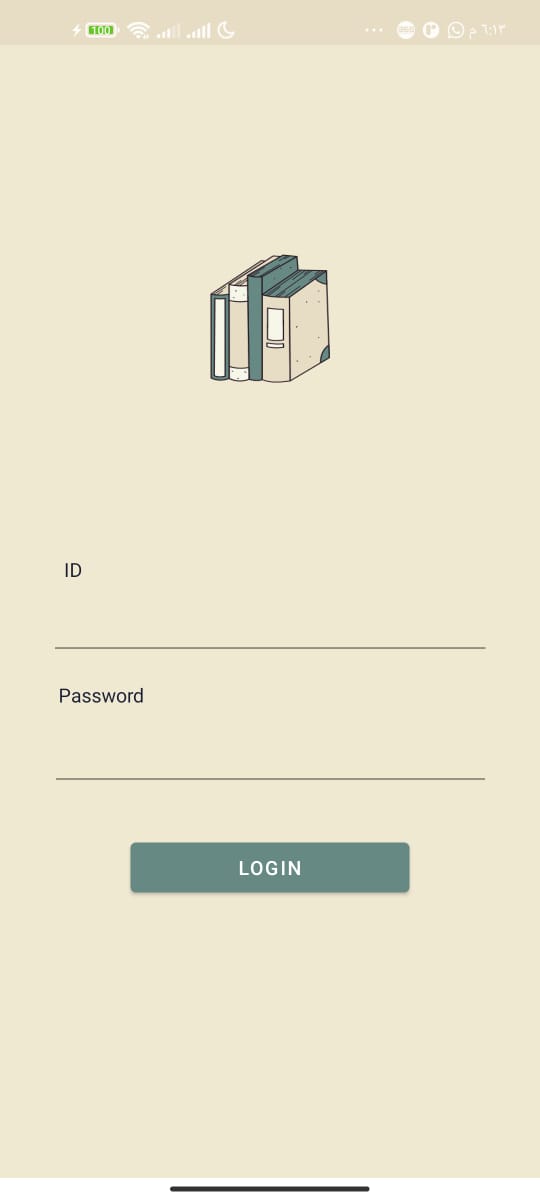
**Chapter 5**

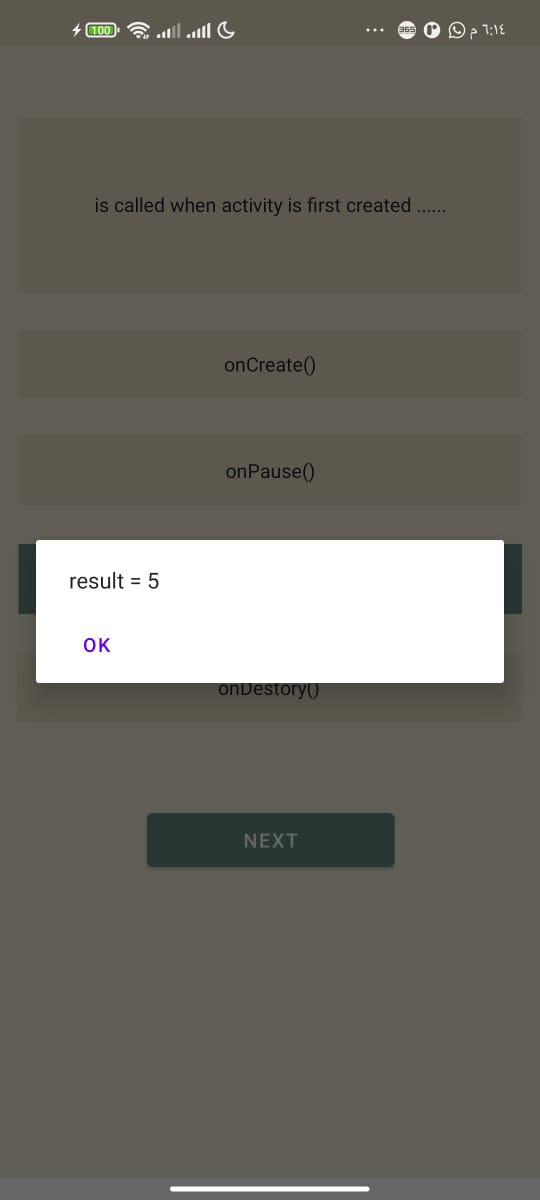
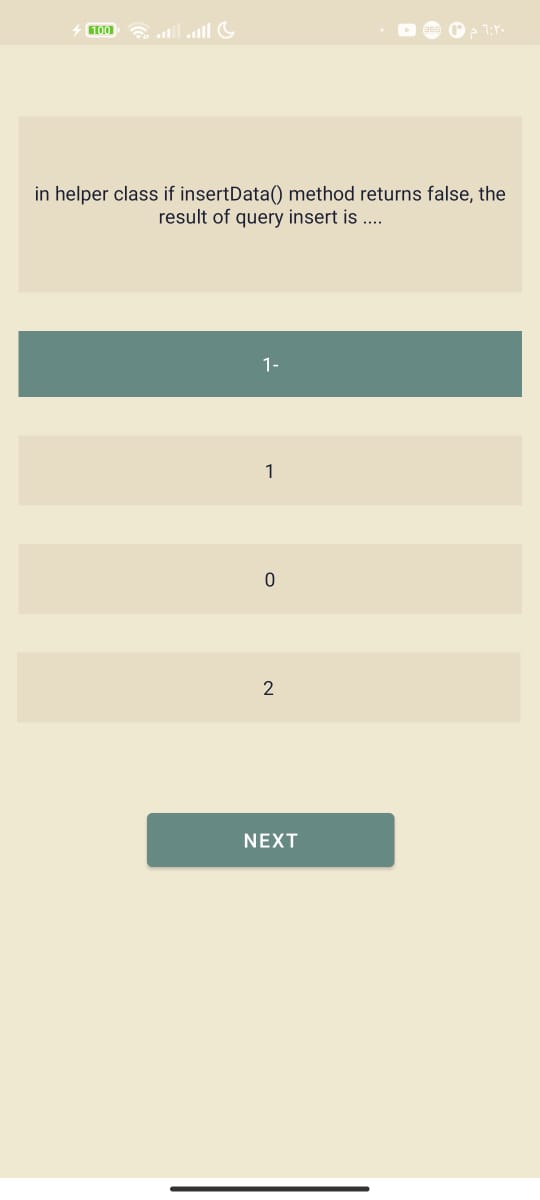
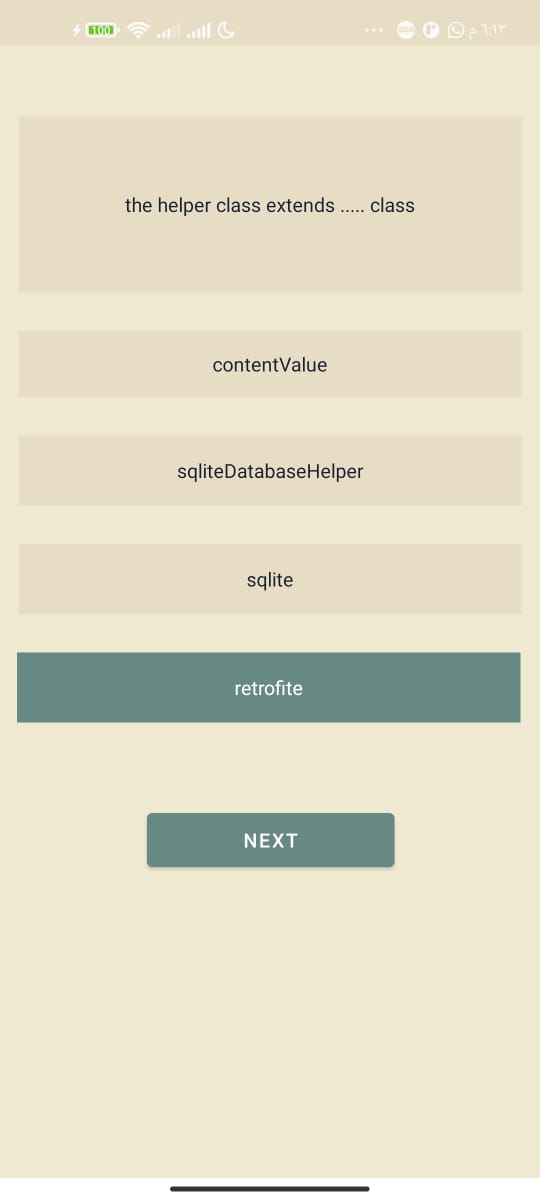
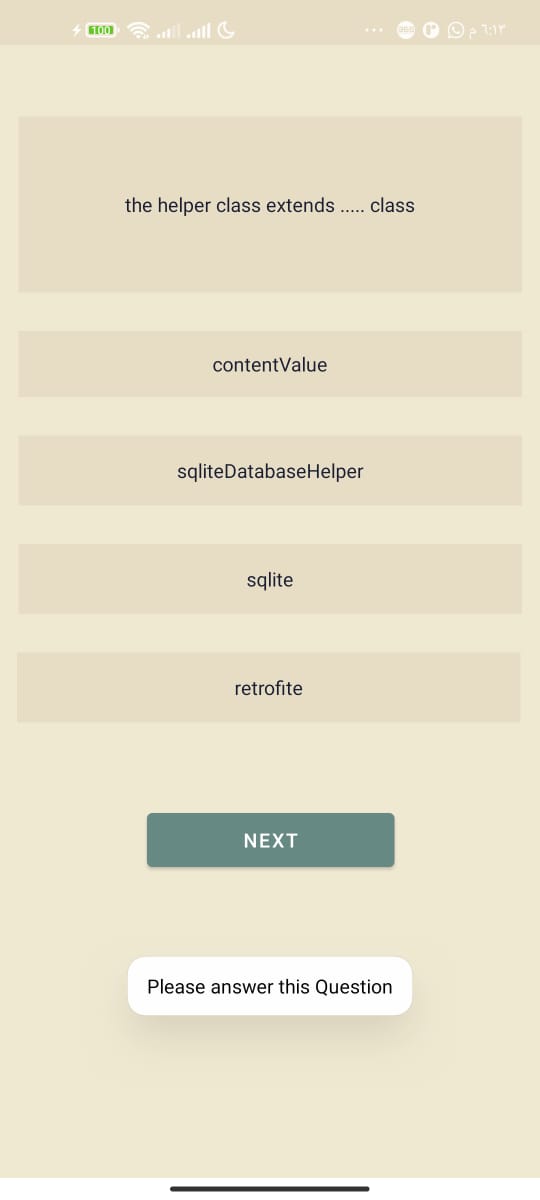
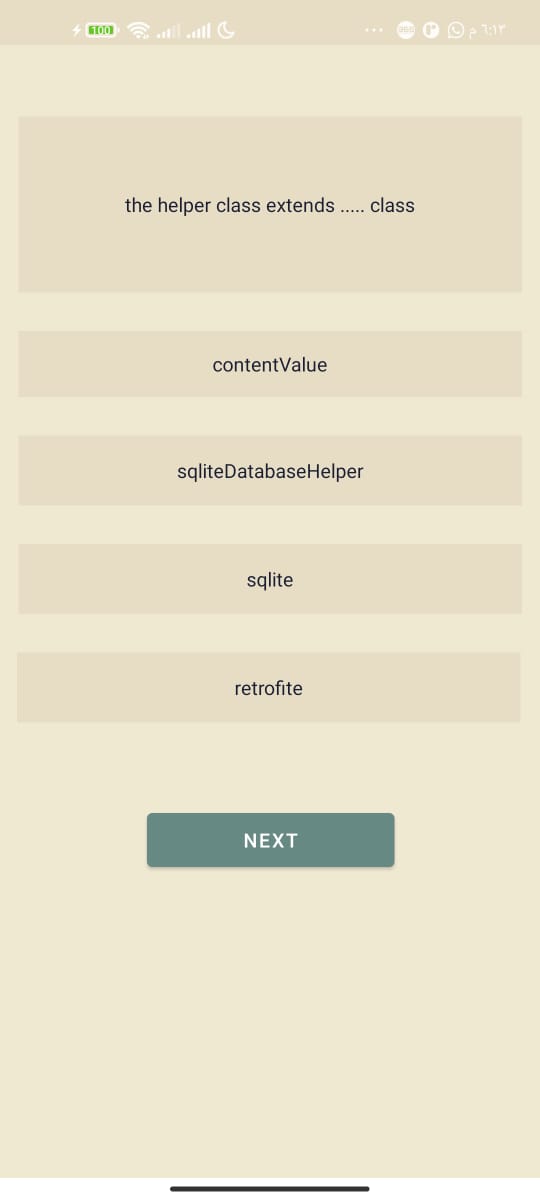
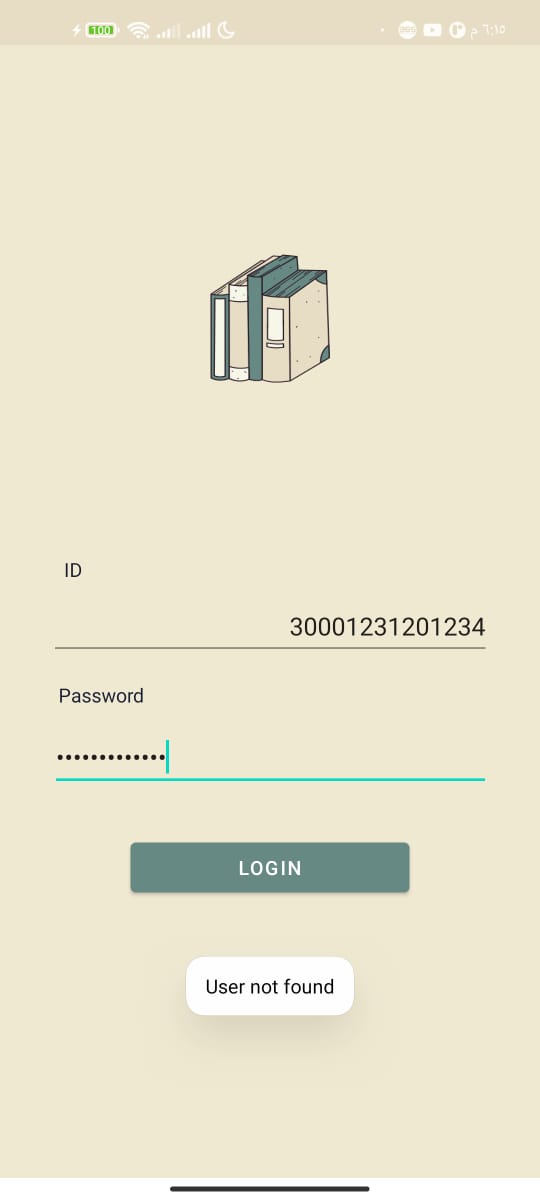
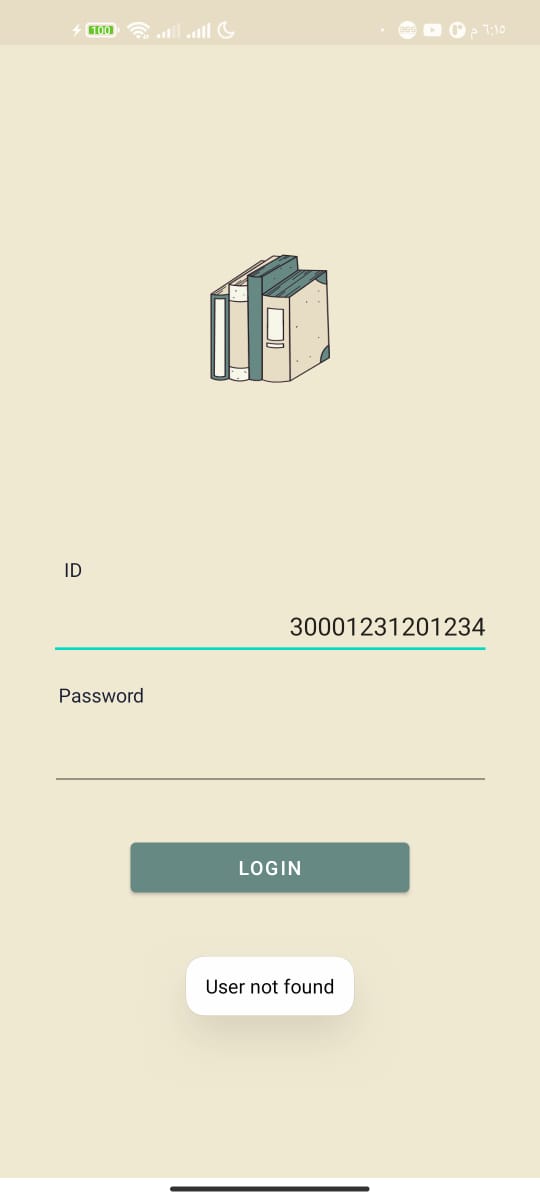
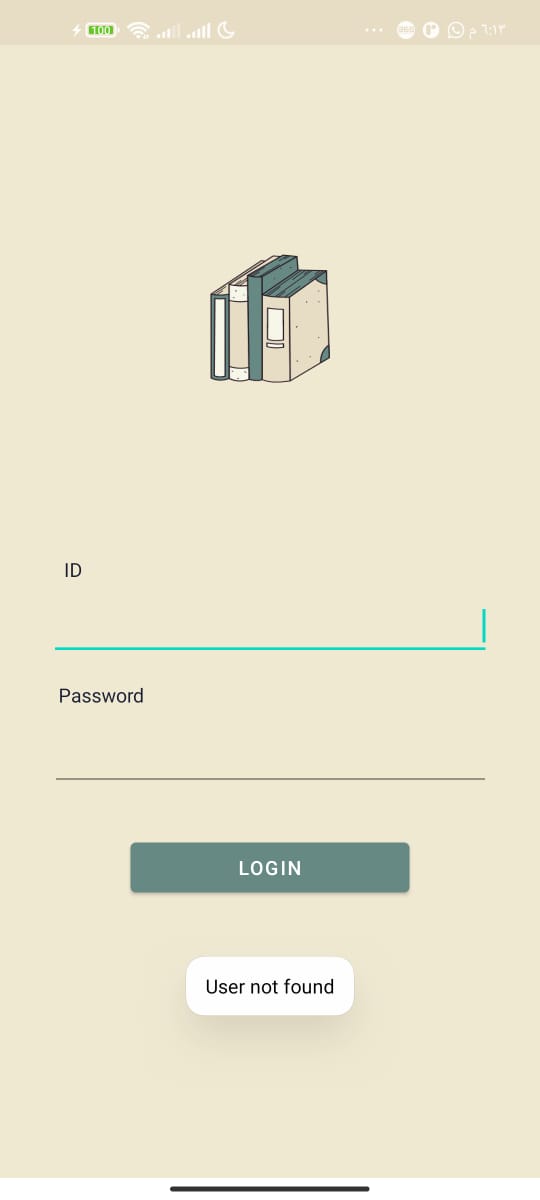
**Testing Phase**

**5\_1 UI**

**5\_2 Test cases**

**5.1 UI**





**5.2 Test cases**

A suitable ui has been used for all users, not forgiving of interface errors

User cannot login without ID and password is correct and must make sure that the text box is filled in

He cannot skip a question without answering it

**Chapter 6**

**Future Work**

In the nearest future, we hope to turn this project into a bigger one.

Some future work:

- An admin system, adding questions, modifying an exam, and starting an exam and all results for students.

- Improvement in design.

- Right and wrong option.