Al-Quds University
Department of Computer Engineering

Database Systems

Course id#: 0702325

Online Examination System

Supervised by:
Dr. Rushdi hamammreh

Team Work:

Anwar Ali Roumanen - Id#:22210719

Rawan Fraihat - Id#:22210313

Contents

Preface

Our examination system, now in version 2, provides a comprehensive and personalized assessment of each student's academic performance using a database, improving upon version 1, which used JSON files. Unlike conventional testing systems, our approach evaluates students' grades in specific fields and specializations, offering precise measurements of strengths and areas for improvement. Specialized examinations, designed by expert educators, allow students to explore career paths and academic disciplines that match their abilities. Teachers can add custom exams and questions, making the evaluation process dynamic and tailored to individual needs.
Each student and teacher has a dedicated account, giving access to an interactive learning environment that supports continuous improvement and self-directed learning.
Our system is built using a two-tier architecture, with a clear separation between the user interface and the database. This design makes the system faster, more secure, and easier to manage, while also making future updates or changes simpler. This system empowers students with meaningful insights to make informed decisions about their future.

Chapter one: introduction

1.1 Background and purpose

Choosing a university major is a critical decision that shapes a student's academic and professional future. However, many students struggle with this choice due to a lack of guidance, self-awareness, or exposure to various academic disciplines. The absence of a structured method for evaluating personal interests and cognitive strengths further complicates the decision-making process.

To address this challenge, we have developed an intelligent examination system that provides students with a clear understanding of their academic preferences and potential career paths. By utilizing a data-driven approach, our system bridges the gap between students' interests and suitable university specializations, helping them make informed choices that align with their strengths.

1.2 Problem statement

Many students experience uncertainty when selecting their university major, leading to:

- Indecision and hesitation, causing delays in choosing a specialization.
- Frequent major changes, resulting in wasted time and financial resources.
- Enrollment in unsuitable disciplines, leading to dissatisfaction and underperformance.
- Lack of career awareness, preventing students from discovering fields that match their abilities.

These issues highlight the need for a structured system that analyzes students' responses, categorizes their cognitive skills, and provides targeted academic recommendations.

1.3 Solution Approach

Our system offers a set of customized online exams that analyze students' answers and classify their competencies into specific categories. The system then uses these classifications to recommend academic majors that match the student's cognitive style, interests, and performance.

Key features of the system include:

- Personalized Assessments – Tailored exams that evaluate students' strengths in various academic disciplines.
- Data-Driven Insights – Advanced algorithms that analyze responses and suggest relevant fields of study.

- Adaptive Learning Environment – Continuous updates and refinements based on student performance and feedback.
- Customizable Exam Modules – Teachers can modify test content to enhance accuracy and relevance.

By integrating these features, the system ensures that each student receives targeted guidance to make well-informed academic decisions.

1.4 impact and benefits

Implementing this system provides multiple advantages for students, educators, and the academic sector as a whole:

- **For Students:**
  - Increased confidence in selecting a major.
  - Clear identification of academic strengths and interests.
  - Minimized risk of choosing an unsuitable specialization.
- **For Educators:**
  - Better tools to assess and guide students.
  - Insights into student performance trends.
  - Ability to refine exam content for better accuracy.
- **For Academic Institutions:**
  - Reduction in dropout rates and major-switching cases.
  - Improved graduation efficiency and student satisfaction.
  - Development of a more tailored and student-focused educational experience.

By providing structured academic guidance, our system enhances the overall quality of education and prepares students for a future in which they can thrive and contribute effectively to society.

Chapter two: Requirements

2.1. User Requirements

student:

- Should be able to log in/sign up.
- Should be able to take online exams.
- Should be able to view their scores and feedback.
- Should have an intuitive interface to navigate exams and view results.

admin(teacher):

- Should be able to log in/sign up.
- Should be able to create and modify exam questions.
- Should be able to categorize questions based on subjects.
- Should have access to student results for feedback.

2.2. System Requirements

2.2.1. Functional System Requirements

- System should allow to users (student / admin) to be able to log in/sign up securely.
- System should store and analyze student results according to their answers.
- System should provide tests from multiple fields stored in database.
- System must show the score of exams to student and to admins for recommending.
- system allow students to start and submit exams and logout from account when needed.
- System should display results and feedback after completion.
- System should give scores and initial feedback according to table 2.1 .

Table 2. 1 student's level

Score Range	Status	Feedback
Below 50	Not recommended	It is advised not to pursue this field as the score indicates insufficient aptitude.
50-74	Potential, but explore	You show capability, but it is recommended to explore other fields for better alignment.
75 and above	Strong fit	This field is highly suitable for you, and it aligns well with your skills and potential.

2.2.2. Non-functional System Requirements

- Performance: The system must get data from database within 3 second(maximum).
- usability: The UI should be user friendly and responsive.
- Reliability: The system should be available 99.9% of the time.
- Scalability: The system should handle at least 500 users at the same time.

2.2.3. Domain Requirements

- The system will be used by students and teachers (admins) to take and manage online exams.
- The system will support multiple subjects and question formats.
- A relational database will be used to store and retrieve exam data.

Chapter three: modeling system

This chapter covers the modeling techniques used to develop the Examination System, detailing how the system's components and interactions are structured to ensure efficient performance and accurate assessment.

3.1 use case diagram

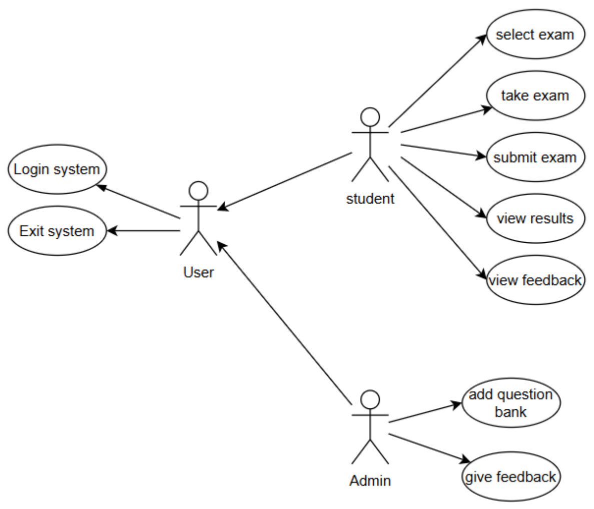


fig 1 use case diagram

The diagram in fig 1 represents how the Student and Admin interact with the system to complete their respective tasks. It also highlights the system's main functionalities and how the roles are distributed between users.

primary actors: Student and Admin.

- The student can perform the following actions:
    1. **Select Exam**: Choose an available exam to take.
    2. **Take Exam:** Answer questions and complete the assessment.
    3. **Submit Exam:** Send the completed exam for evaluation.
    4. **View Results:** Access the exam results after submission.
    5. **Receive Feedback:** View feedback provided by the admin.
- The admin can perform the following actions:

1. **Add Question Bank:** Upload or update exam questions.
2. **View Results:** Access the students' exam results.
3. **Provide Feedback:** Give feedback on student performance.

3.2 sequence diagram

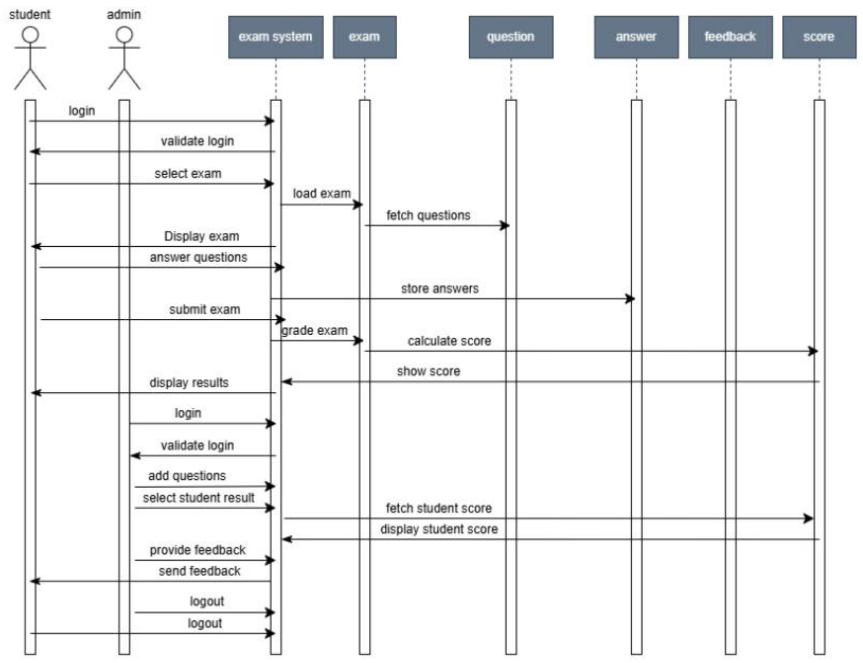


fig 2 sequence diagram

User's scenarios:

- Student Taking an Exam:

1. Student Logs In:

The student sends login credentials to the SessionManager.

The SessionManager validates the credentials and retrieves the user information from the Utility (database).

2. Select Exam:

The student selects an exam from the available list.

The Utility class retrieves the exam details from the database and presents them to the student.

3. **Take Exam:**

The student starts the exam, and questions are loaded one by one from the Utility.

For each answer, the student submits their choice, which is temporarily stored in the Answer class.

4. **Submit Exam:**

Once the exam is completed, the student submits the exam.

The Answer data is sent to the Utility class to be saved in the database.

The Score is calculated based on correct answers and stored in the database through the Utility.

5. **View Results:**

The student requests to view the results.

The Utility retrieves the exam score and feedback from the database and displays it to the student.

- Admin providing Feedback:

1. **View Results:**

The admin requests to view students' results from the Utility.

The Utility fetches the data from the database and presents it to the Admin.

2. **Provide Feedback:**

The admin writes feedback based on the student's performance. The feedback is saved to the database through the Utility.

3.3 class diagram

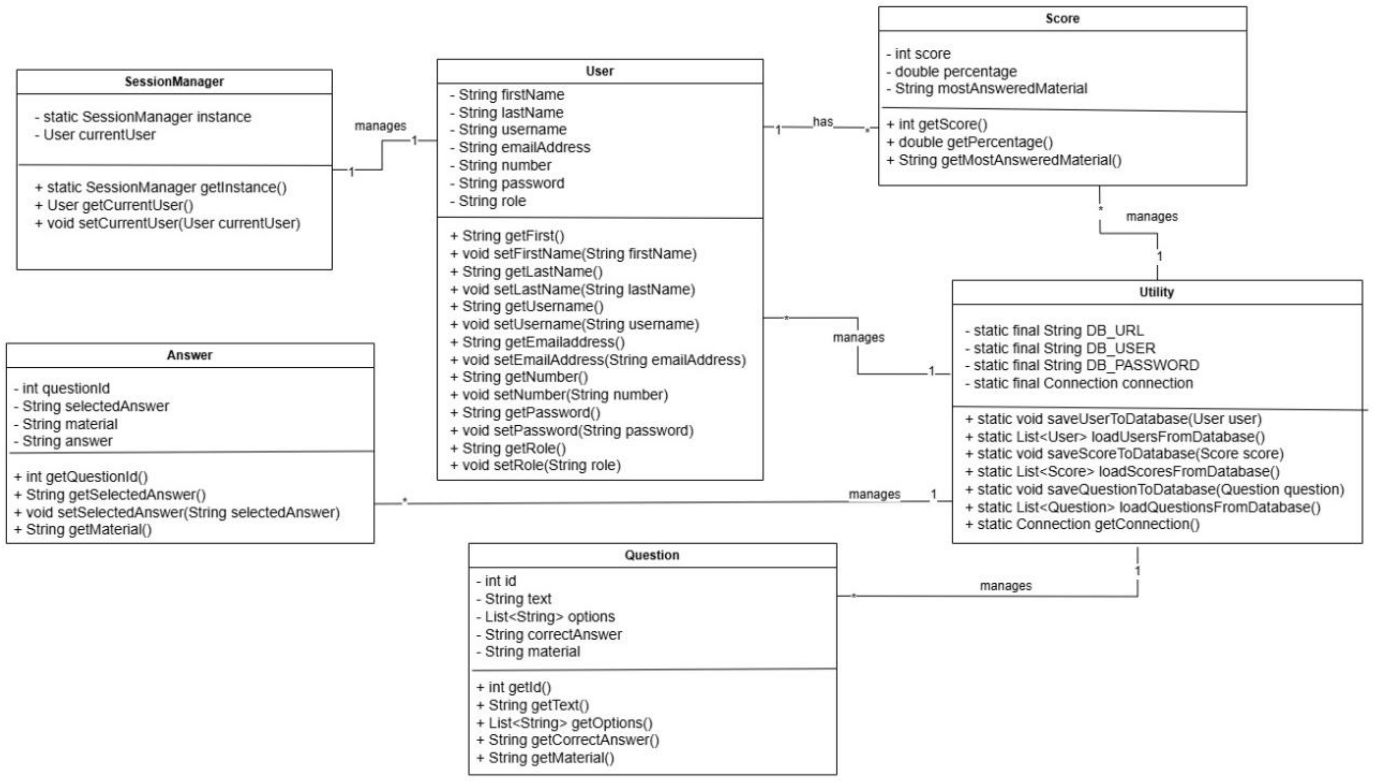

SessionManager
- static SessionManager instance
- User currentUser
+ static SessionManager getInstance()
+ User getCurrentUser()
+ void setCurrentUser(User currentUser)
manages
1
1
User
- String firstName
- String lastName
- String username
- String emailAddress
- String number
- String password
- String role
+ String getFirst()
+ void setFirstName(String firstName)
+ String getLastName()
+ void setLastName(String lastName)
+ String getUsername()
+ void setUsername(String username)
+ String getEmailaddress()
+ void setEmailAddress(String emailAddress)
+ String getNumber()
+ void setNumber(String number)
+ String getPassword()
+ void setPassword(String password)
+ String getRole()
+ void setRole(String role)
1
has
*
Score
- int score
- double percentage
- String mostAnsweredMaterial
+ int getScore()
+ double getPercentage()
+ String getMostAnsweredMaterial()
*
manages
1
Utility
- static final String DB_URL
- static final String DB_USER
- static final String DB_PASSWORD
- static final Connection connection
+ static void saveUserToDatabase(User user)
+ static List<User> loadUsersFromDatabase()
+ static void saveScoreToDatabase(Score score)
+ static List<Score> loadScoresFromDatabase()
+ static void saveQuestionToDatabase(Question question)
+ static List<Question> loadQuestionsFromDatabase()
+ static Connection getConnection()
*
manages
1
Answer
- int questionId
- String selectedAnswer
- String material
- String answer
+ int getQuestionId()
+ String getSelectedAnswer()
+ void setSelectedAnswer(String selectedAnswer)
+ String getMaterial()
*
manages
1
Question
- int id
- String text
- List<String> options
- String correctAnswer
- String material
+ int getId()
+ String getText()
+ List<String> getOptions()
+ String getCorrectAnswer()
+ String getMaterial()
*
manages
1

fig 3 class diagram

class diagram in fig 3 represents the Examination System and its core components, showing the relationships and interactions between different following classes:

1. **User Class:**
   - Represents a user in the system, storing personal details such as name, username, email, phone number, and role.
   - Provides getter and setter methods for each attribute to manage user information.
2. **Answer Class:**
   - Stores information about answers submitted by the user, including the question ID, selected answer, related material, and the correct answer.
   - Provides methods to get and set the answer details.
3. **Question Class:**

- Represents a question with its ID, text, multiple options, the correct answer, and related material.
- Includes methods to retrieve question details and options.

4. **Score Class:**
    - Stores the user's exam score, the percentage achieved, and the most answered material.
    - Provides methods to access score and performance information.
5. **SessionManager Class:**
    - Manages the current user session and holds a single instance of the session manager (Singleton pattern).
    - Provides methods to set and retrieve the current user.
6. **Utility Class:**
    - Handles database-related operations, such as saving and loading users, questions, and scores.
    - Uses static methods to interact with the database and manage persistent data.

3.4 state diagram

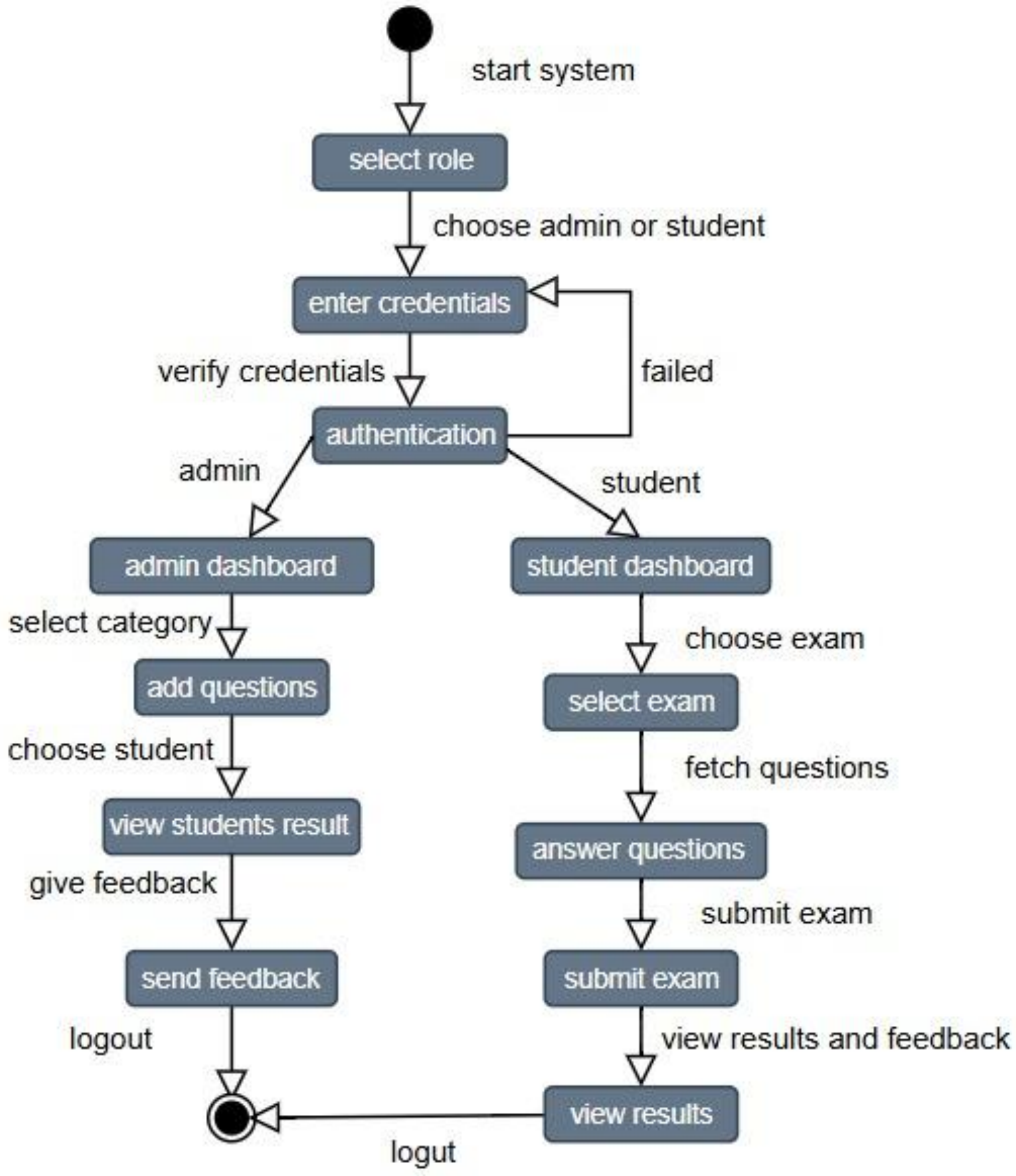


fig 4 state diagram

The State Diagram in fig 4 illustrates the different states of key components in the Examination System and their transitions based on user actions.

States and Transitions:

1. **System start:**
   The system starts and prompts the user to choose a role (Admin or Student).
2. **Enter Credentials:**

The user enters their credentials. If authentication fails, the system returns to the Enter Credentials state. If authentication is successful, the system transitions based on the selected role.

3. **Student flow:**
   - **Student Dashboard:** After successful authentication, the system shows the Student Dashboard.
   - **Choose Exam:** The student selects an exam to take.
   - **Answer Questions:** The student answers the questions.
   - **Submit Exam:** After completing the exam, the student submits their answers.
   - **View Results:** The student can view their exam results and feedback.
   - **Logout:** The student logs out, returning to the System Start state

4. **Admin flow:**
   - **Admin Dashboard:** After successful authentication, the system shows the Admin Dashboard.
   - **Add Questions:** The admin can add or manage questions in the question bank.
   - **View Student Results:** The admin can view students' results.
   - **Send Feedback:** The admin can send feedback to students.
   - **Logout:** The admin logs out, returning to the System Start state.

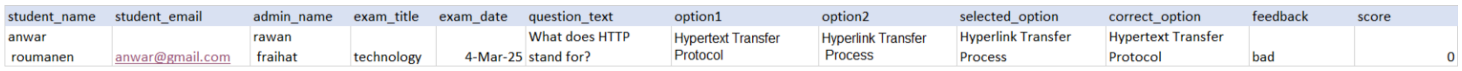
3.5Normalization

1. Unnormalized form (UNF)

This is the main user-related data required to build the database for the examination system:

- student details
- admin details
- exam information
- question details
- options and answers
- feedback and score

The table contained repeating groups like multiple options for the same question inside the same record, and it was not organized for database structure.



student_name	student_email	admin_name	exam_title	exam_date	question_text	option1	option2	selected_option	correct_option	feedback	score
anwar roumanen	anwar@gmail.com	rawan fraihat	technology	4-Mar-25	What does HTTP stand for?	Hypertext Transfer Protocol	Hyperlink Transfer Process	Hyperlink Transfer Process	Hypertext Transfer Protocol	bad	0

fig 5 UNF table

2. 1NF

I merged option1 and option2 into one column called options,now each option became one record under the options column but we still have a problem that each option is now a row, the other data is repeated for every option.

So data redundancy increased which makes updates harder and increases storage unnecessarily.

student_name	student_email	admin_name	exam_title	exam_date	question_text	options	selected_option	correct_option	feedback	score
anwar roumanen	anwar@gmail.com	rawan fraihat	technology	4-Mar-25	What does HTTP stand for?	Hypertext Transfer Protocol	Hyperlink Transfer Process	Hypertext Transfer Protocol	bad	0
anwar roumanen	anwar@gmail.com	rawan fraihat	technology	4-Mar-25	What does HTTP stand for?	Hyperlink Transfer process	Hyperlink Transfer process	Hypertext Transfer Protocol	bad	0

fig 6 1NF table

3. 2NF

Moving options into a single column made the table more flexible to add or remove options,and we separated the data into different tables because there are fields independent of options , and we include some data needed for every table

students table

student_id	full_name	email	phone	user_password	username
1	anwar roumanen	anwar@gmail.com	"0567133778"	Anwar@@669	anwartech

admins_table

admin_id	full_name	email	phone	user_password	username
1	rawan fraihat	rawan@gmail.com	"0597856478"	Rawan@@446	rawanfr

exams table

exam_id	title
1	technology

exam_attemps table

attemp_id	exam_date	selected_option	correct_answers	feedback	score
1	4-Mar-25	Hyperlink Transfer process	0	bad	0

questions table

question_id	question_text
1	What does HTTP stand for?

options table

option_id	option_text
1	Hyperlink Transfer Process
2	Hypertext Transfer Protocol

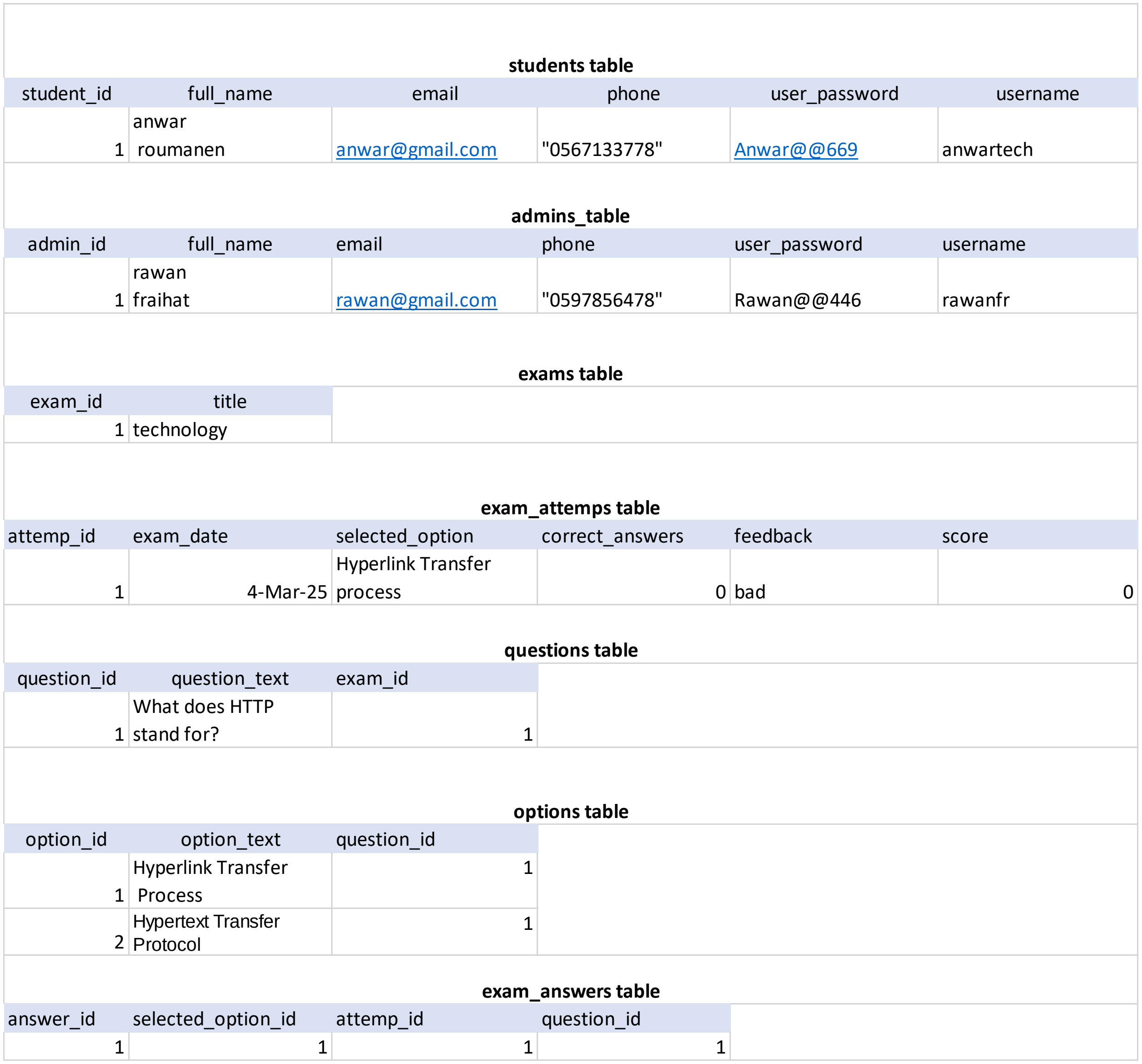
exam_answers table

exam_id	selected_option_id
1	1

fig 7 2NF table

4. 3NF

We linked table with correct foreign keys to eliminate transitive dependencies and ensure full normalization



students table

student_id	full_name	email	phone	user_password	username
1	anwar roumanen	anwar@gmail.com	"0567133778"	Anwar@@669	anwartech

admins_table

admin_id	full_name	email	phone	user_password	username
1	rawan fraihat	rawan@gmail.com	"0597856478"	Rawan@@446	rawanfr

exams table

exam_id	title
1	technology

exam_attemps table

attemp_id	exam_date	selected_option	correct_answers	feedback	score
1	4-Mar-25	Hyperlink Transfer process	0	bad	0

questions table

question_id	question_text	exam_id
1	What does HTTP stand for?	1

options table

option_id	option_text	question_id
1	Hyperlink Transfer Process	1
2	Hypertext Transfer Protocol	1

exam_answers table

answer_id	selected_option_id	attemp_id	question_id
1	1	1	1

fig 8 3NF table

5. BCNF

After completing 3NF, we verified that every determinant is a candidate key. now our schema is ready to implement DDL and DML and build our database, after reducing redundancy.

3.5 Entity-Relationship Diagram (ER)

TheER Diagram for the Examination System models the database structure, explainanig how various entities interact with each other. The system's database contains several key tables: Students, Admins, Options, Exams, Exam_Answers, Answers, Questions, and Exam_Attempts. Each of these entities plays a vital role in the operation and management of the system as shown in fig 9.

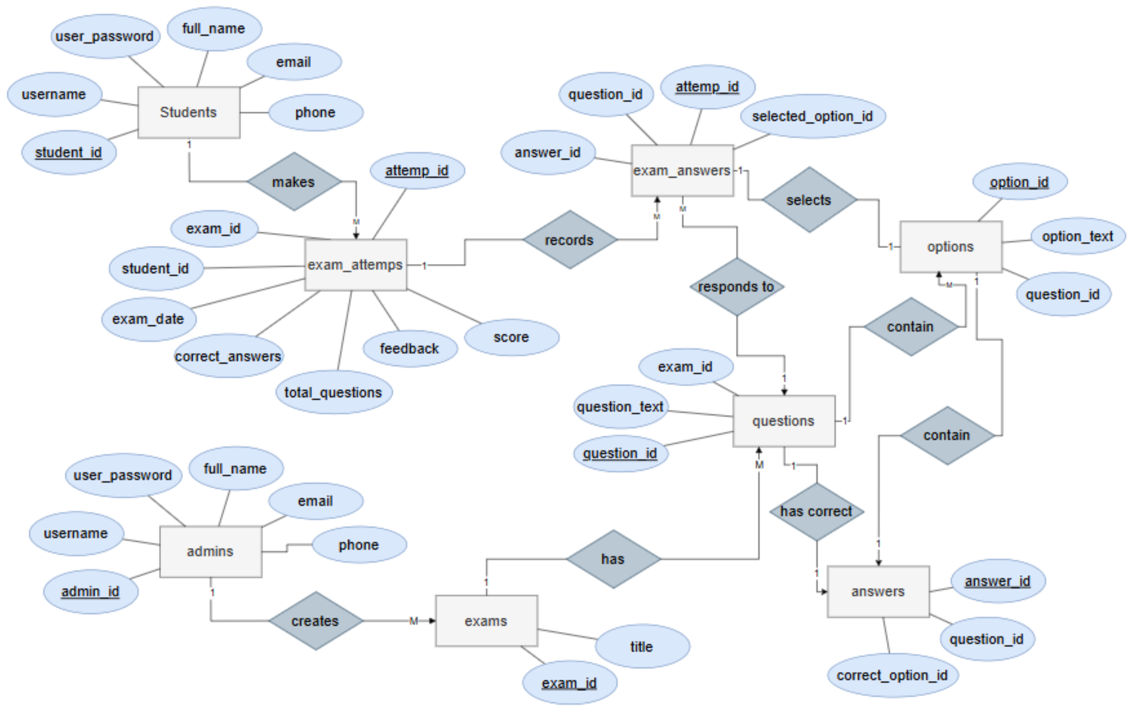


fig 9 ER diagram

Relationships Overview:

➢ **One to Many**:

- A Student can have multiple Exam_Attempts(one student can attempt multiple exams).

- An Exam can have multiple Questions and Exam_Attempts.
- A Question can have multiple Options.

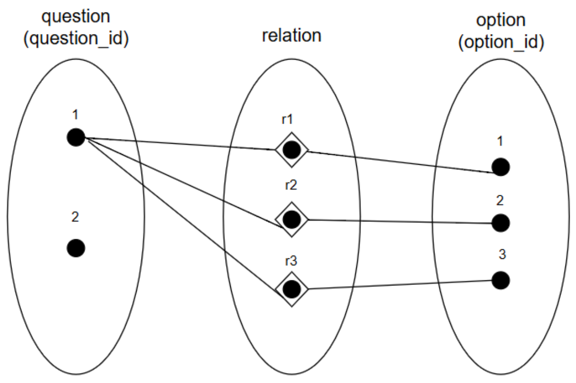


fig 10 Semantic network showing 1-M relationship

(one Question Contains more than one Option)

➢ **One to one:**

- each question is linked to a single answer.
- each answer corresponds to one selected option.
- each option is recorded in the exam attempt.

Chapter four: Design and implementation

Design system

Architectural Design

our system is built using a two tier architecture and follows a modular architecture composed of independent yet interconnected components. Each module performs a well-defined function, enhancing system scalability, maintainability, and flexibility for future updates. This section outlines the core modules and their responsibilities, providing a comprehensive view of how the system components interact to deliver a seamless experience for both students and administrators.

System Components and Their Interactions

- **User Interface (UI):**
  Description:
  The front-end interface that allows users to interact with the system, offering different experiences based on the user's role (Student or Admin).

  Technologies:
  JavaFX, Scene Builder.

  Responsibilities:
  - Accept login credentials from users.
  - Display available exams, questions, and result feedback.
  - Allow the Admin to add and manage questions.
  - Provide a clear and user-friendly navigation system.

- **Question Management module**
  Description:
  Responsible for creating, updating, and retrieving exam questions stored in the database.

  Technologies:
  Java, Oracle SQL.

  Responsibilities:
  - Store question text, options, and correct answers.
  - Validate input from Admin before insertion.
  - Fetch questions dynamically during an exam session.

- **Answer submission & Evaluation module**
  Description:
  Processes student responses and calculates the results.

  Technologies:
  Java, JDBC, Oracle DB.

  Responsibilities:
  - Collect submitted answers.
  - Compare them with correct answers.
  - Calculate scores and determine performance.
  - Store results and provide data for feedback generation.

- **Utility module**
  Description:
  A static utility class that acts as a bridge between the logic layer and the database.

  Technologies:
  Java, JDBC, Oracle SQL.

  Responsibilities:
  - Perform all database read/write operations.
  - Provide reusable functions for other modules to interact with the database.
  - Handle errors and ensure data consistency.

- **Result & Feedback module**
  Description:
  Handles result display and feedback creation.

  Technologies:
  Java, Oracle DB.

  Responsibilities:
  - Allow students to view their scores and feedback.
  - Enable the Admin to enter performance comments.
  - Store and retrieve this data reliably for future reference.

- **Database (Oracle SQL)**
  Description:
  Stores persistent data related to users, exams, questions, answers, scores, and feedback.

  Responsibilities:
  - Maintain normalized relational tables to ensure integrity.

- Support efficient querying and updates.
- Ensure security through user-role distinctions.

Module Design

The Examination System is composed of multiple well-defined modules, each responsible for specific tasks to ensure smooth and accurate operation. This section describes the internal structure, responsibilities, and interactions of each module, including the major classes and functions involved.

- **User management module**
  Purpose:
  Manages user authentication and session tracking for both Students and Admins.

  Controllers:
  - Login.java : Handles user login input and session creation.
  - Signup.java : Capture user registration data and if valid stores user to database.

  Main Classes:
  - User: Represents user data such as username, password, email, and role.
  - SessionManager: Implements the Singleton pattern to manage the current user session.

  Internal Logic:
  - SessionManager.getInstance() ensures that only one session is active at a time.
  - On successful login, the user object is stored and accessible by other modules for authorization checks.

- **Question management module**
  Purpose:
  Enables the Admin to manage the question bank by adding, editing, and retrieving questions.

  Controllers:
  - addQuestions.java : let admins insert new questions to database.
  - GenExam.java : while generating exam take questions for user from DB.

  Main Class:
  Question: Contains attributes such as question Text, options, and correctAnswer.

  Internal Logic:
  - The Question class interacts with the Utility class to perform insert and select operations.
  - Input validation ensures that every question has at least one correct answer and no duplicate options.

- **Answer Submission & Evaluation module**
  Purpose:
  Captures and processes student answers, then calculates scores based on correctness.

  Main Classes:
  - Answer: Holds selected answers and related metadata.
  - Score: Stores calculated results and performance metrics.

  Internal Logic:
  - Student answers are collected in a list of Answer objects.
  - Upon submission, the list is sent to the Utility class for scoring and saving.
  - The Score class calculates the percentage of correct answers and passes it to the result module.

- **Result & feedback module**
  Purpose:
  Handles result generation, display, and feedback exchange between Admins and Students.

  Controllers:
  Results.java : display the results for user and feedback coming from admins.
  Results2.java: display the results and feedback for specialization exams not general one.

  Main Class:
  - Score: Stores exam scores and links to feedback.
  - Admin UI or Controller class (GUI): Allows feedback entry.

  Internal Logic:
  - After result generation, the Admin can write comments using a dedicated input area.
  - Feedback is saved in the same or a related table to the Score, allowing easy retrieval.

- **Utility module**
  Purpose:
  Provides reusable methods for interacting with the Oracle Database.

  Main Class:
  Utility: A static class containing SQL methods.

  Internal Logic:
  - Methods like getConnection(),saveUser(), loadRandomQuastions(), saveResults(), and loadtResults() interact with the DB via JDBC.
  - Errors are caught and logged to prevent application crashes.

- **User interface module**
  Purpose:
  Acts as the front-end of the system, facilitating all user interactions.
  Internal Logic:
  - FXML-based views are linked with controller classes.
  - Controllers call the appropriate methods from other modules (like Utility or SessionManager) based on user actions.

Database design

The database of the Examination System is designed to efficiently store and manage user data, exam content, student submissions, scores, and feedback. It follows a relational structure and is built using Oracle Database. The schema ensures data integrity, scalability, and ease of retrieval for different system operations.

- **ERD** (see in fig 9)
- **Table descriptions**
  - **Users**: Stores user information including name, username, password, email, phone, and role (student or admin).
  - **Exams**: Contains exam metadata such as exam title.
  - **Questions**: Stores individual questions linked to specific exams.
  - **Options**: Holds multiple-choice answers for each question, with a flag for the correct option.
  - **Exam_Attempts**: Tracks student exam submissions, including score and date of attempt.
  - **Exam_Answers**: Records selected options for each question in an exam attempt and whether they were correct.
  - **Feedback**: Stores admin comments related to each exam attempt for personalized student feedback.

- **Normalization Justification**
  The database design follows Boyce Codd Normal Form to eliminate redundancy and ensure data integrity:
  - **1NF:** All fields contain atomic values (no repeating groups).
  - **2NF:** All non-key attributes are fully dependent on the primary key.
  - **3NF:** There are no transitive dependencies (e.g., correct answers are stored in the Options table, not repeated in Answers).
  - **BCNF:** Every determinant is a candidate key.

- **Technologies used For DB**
  - Oracle Database: A powerful relational DBMS used to implement and manage the entire system schema. It supports advanced SQL queries and constraints.
  - SQL Developer: Used for designing tables, writing queries, and managing data.
  - JDBC: Used to interact with Oracle DB from Java.

User interface design

The Examination System includes a role-based graphical user interface designed using JavaFX and Scene Builder. The interface is simple and user-friendly, allowing students and admins to navigate easily between different features.

- Login interface

  The screen in fig 11 (see Appendix A) allows both students and admins to log in using their username and password. If credentials are valid, the user is redirected based on their role.

- Signup interface

  If students don't have an account can create one in signup interface (fig 12 in Appendix A)

- Student Dashboard

  After login, students can view a list of available exams. They can select an exam, answer questions, submit their answers, and view their results and feedback (fig 13 in Appendix A).

- Exam interface

  In fig 14 (in Appendix A) Screen displays all exams available to students.

after select an exam the screen in fig. 15 (in Appendix A) displays questions one by one along with multiple-choice options. The student selects their answers and submits the exam once finished.

- Results and feedback Screen

  Students can view their scores and the feedback provided by the admin after completing an exam. (fig 16 for bas results and fig 1 for good results in Appendix A)

- Admin Dashboard

  Admins can add new questions to the question bank, view student submissions, and provide feedback based on performance.(in fig 18 and 19 in Appendix A)

Technology Stack

- **Languages**
  - **Java**: Core programming language for implementing system logic and functionality.

    - **SQL**: Used for defining, manipulating, and querying data in the Oracle relational database.
    - **CSS**: style sheet language, to style user interfaces to be more user friendly.
    - **FXML**: markup language, For separating UI layout from logic.

- **Tools and Platforms**
    - **JavaFX**: Framework for building a modern and interactive GUI.
    - **Scene Builder**: Visual layout tool for designing JavaFX interfaces using FXML.
    - **Oracle Database**: Relational database for persistent storage of users, exams, questions, answers, and results.
    - **JDBC**: API to connect the Java application to the Oracle database.
    - **IDE (IntelliJ IDEA)**: Integrated Development Environment for writing, compiling, and debugging Java code.
    - **SQL Developer**: Tool for managing Oracle database tables, queries, and data structure.
- **Version Control**
    - **Git**: Local version control system used to track changes and manage code history.
    - **GitHub**: Cloud-based platform used to host the project repository, share code, and maintain backup and documentation.

Implementation

- **Code structure**
  The project follows a clean and organized structure that separates logic, user interface, and styling into dedicated folders to improve maintainability and scalability.
    - Main.java
      This is the entry point of the application (see fig 20 in appendix A). It launches the JavaFX application and loads the initial login screen using FXML.
    - classes/
      This folder contains the core data models of the system, including classes such as User, Question, Answer, Score, and SessionManager. These classes define the structure and behavior of the application's main entities.

- controllers/
  This folder contains the controller classes linked to FXML views. Each controller handles user interactions, validates inputs, manages UI updates, and calls backend logic through the utility class. Examples include LoginController, ExamController, and AdminPanelController.

- fxmlFiles/
  This folder includes all the .fxml files that define the layout and components of the user interface. Each screen, such as login, exam view, or admin panel, has its own FXML file, which is linked to a corresponding controller.

- Styles/
  Contains CSS files used to style the application and ensure a consistent visual appearance across all scenes.

- **Algorithms/Key functions**
  - Login Validation:
    Ensure that username and password entered by user are exist in DB (see fig 21 in appendix A)
  - Signup functionality:
    Make sure the phone number and username are not present to avoid duplication.(see fig 22 in appendix A)

- **Integration**
  - UI and Logic:
    Each FXML file is linked to a controller class which handles UI logic and events.

  - Logic and Database:
    Controller classes call methods in Utility.java which use JDBC to perform SQL queries.

Chapter five: Testing

5.1 Testing Objectives

The objective of the testing is to ensure that the Online Examination System functions as intended across its core components. This includes verifying:

- User interaction through the graphical interface (JavaFX)
- Business logic implemented in the data classes
- Functional correctness of controller methods

5.2 Tools & Technologies Used

- JUnit 5 : Unit testing framework for Java classes and controllers
- JavaFX : For manual UI testing and user flow validation
- IntelliJ IDEA : Development and test execution

5.3 Test Cases and Scenarios

This section outlines the various test cases designed to verify the functionality, correctness, and reliability of both the backend logic using JUnit and the user interface of the Online Examination System. Each test case includes a description of the scenario, the expected result, and the actual result obtained during testing.

The tests are divided into two main categories:

- JUnit Test Cases for unit testing and integration testing the core Java classes and methods.

- UI Test Cases for evaluating user interactions, visual elements, and system behavior during different user journeys.

5.4 Unit Testing

code samples present the unit testing outcomes using JUnit for the core classes and some methods of the Online Examination System. Each unit test checks the correctness of a specific functionality to ensure the system behaves as expected.

Classes: we test the classes that contains setters and getters and constructor and test if everything works perfectly. **(see test results in Appendix B)**

Classes tested: user, answer, question, sessionManager, score.

Methods: some methods need to test to see if method is run correctly or not as expected. **(see test results in Appendix B).**

Tested methods: loadRandomQuestion() in utility to test questions coming from database and connection is works as expected.

5.5 UI Testing

UI testing focuses on verifying that the graphical user interface elements of the Online Examination System function properly and provide a smooth user experience. Tests were conducted to check the behavior of forms, buttons, input fields, and navigation between screens. The goal was to ensure that the UI is intuitive, responsive, and free of errors. Manual testing was performed to simulate real user interactions, and any inconsistencies or malfunctions were recorded and corrected. **(See detailed UI test results in Appendix B.)**

5.6 Integration testing

Integration testing was done to check if different parts of the system work together correctly. We focused on testing the SessionManager class with the User class to make sure that users can log in, start a session, and that their information is handled properly. The goal was to see if the connection between these two classes works without any problems. **(See test results in Appendix B.)**

Appendices

Appendix A: Design and implementation

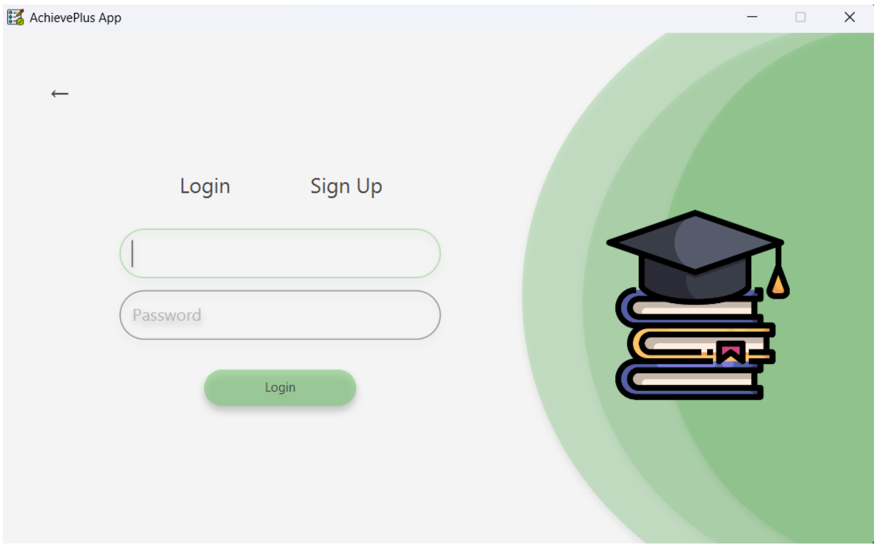


fig 11 login screen

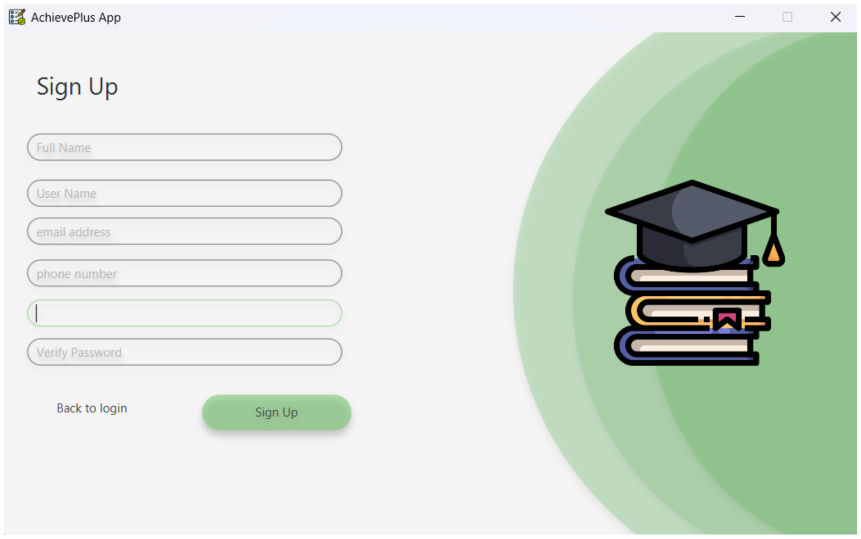
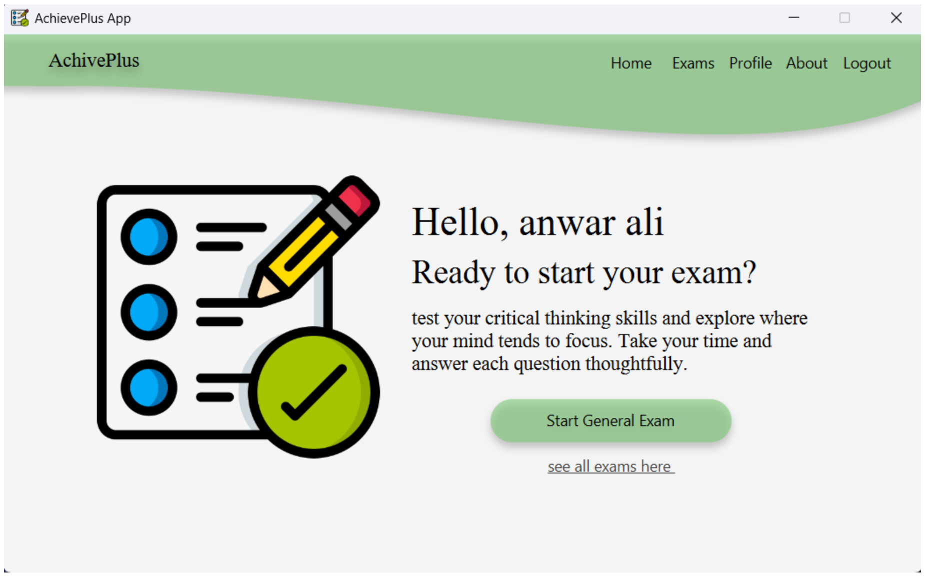


fig. 12 signup screen



Hello, anwar ali

Ready to start your exam?

test your critical thinking skills and explore where your mind tends to focus. Take your time and answer each question thoughtfully.

fig. 13 student dashboard

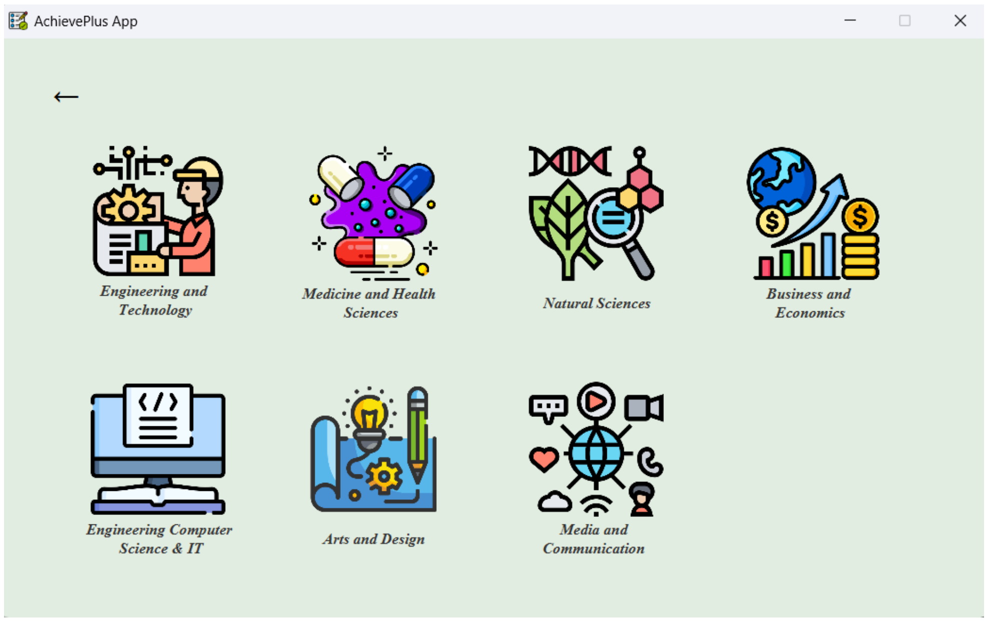


fig. 14 exams created

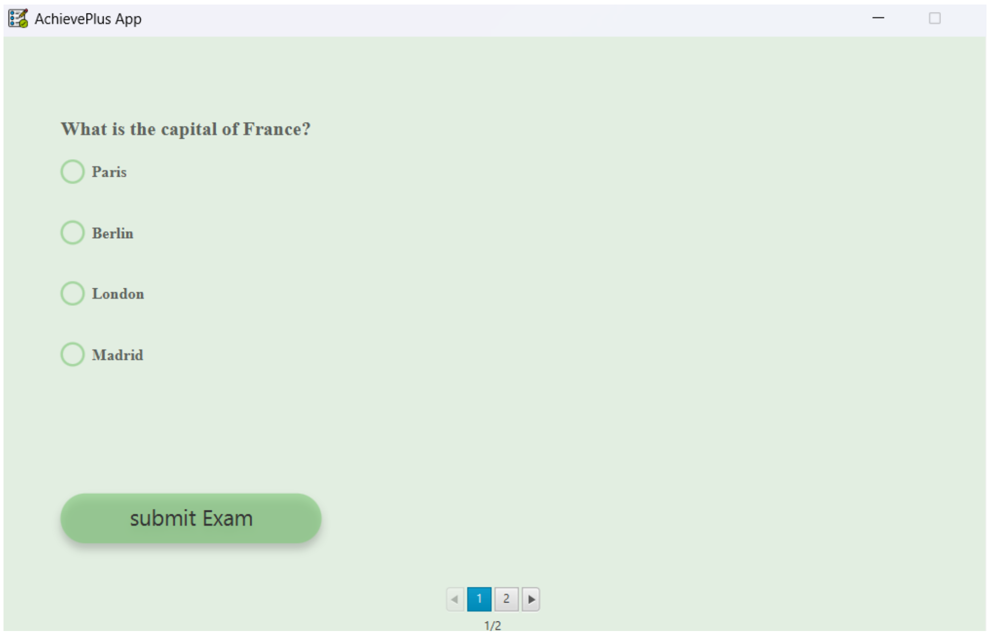
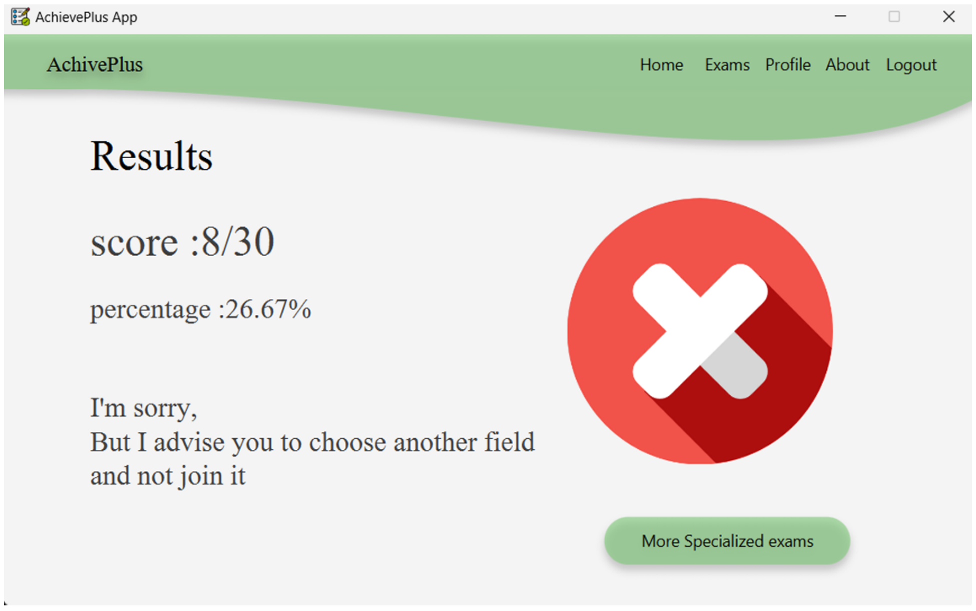


fig. 15 exam questions screen

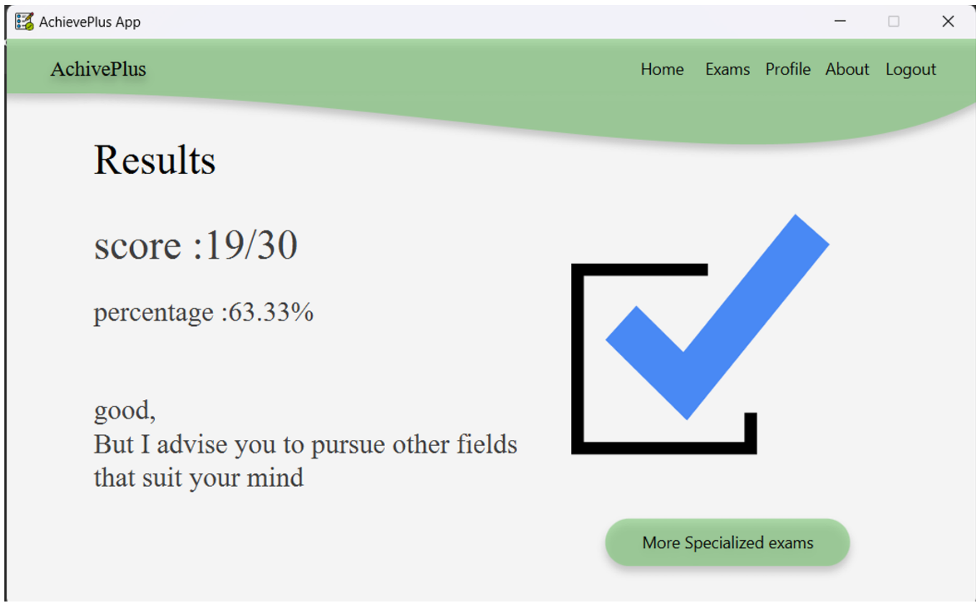


score :8/30

percentage :26.67%

I'm sorry,
But I advise you to choose another field
and not join it

fig. 16 bad result screen


AchievePlus App

AchivePlus

Home Exams Profile About Logout

Results

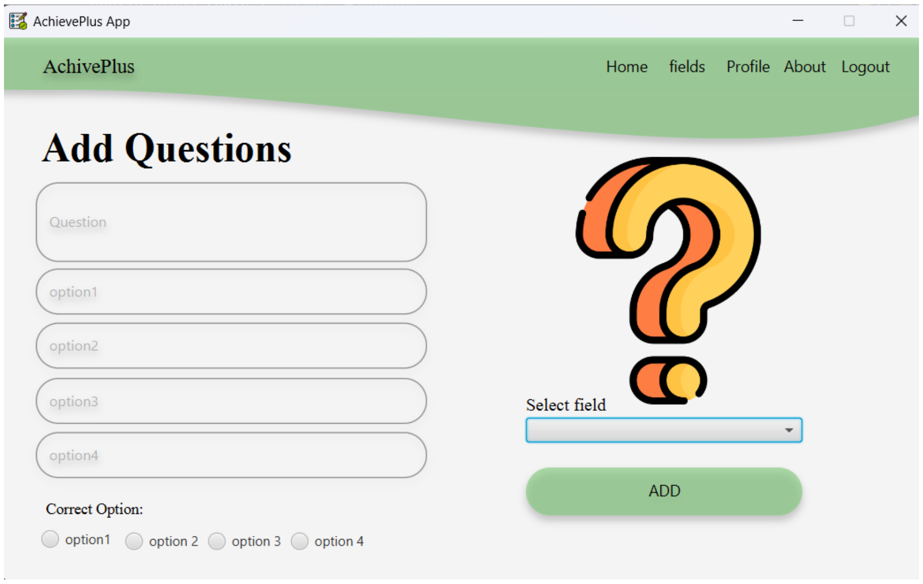
score :19/30

percentage :63.33%

good,
But I advise you to pursue other fields that suit your mind

More Specialized exams

fig. 17 good result screen


AchievePlus App

AchivePlus

Home fields Profile About Logout

Add Questions

Question

option1

option2

option3

option4

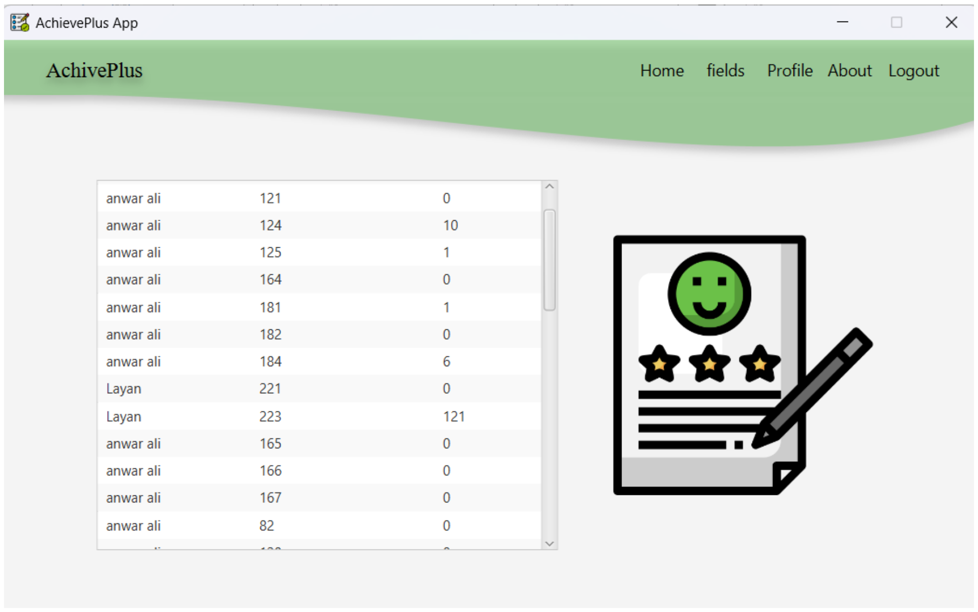
Correct Option:

option1 option 2 option 3 option 4

Select field

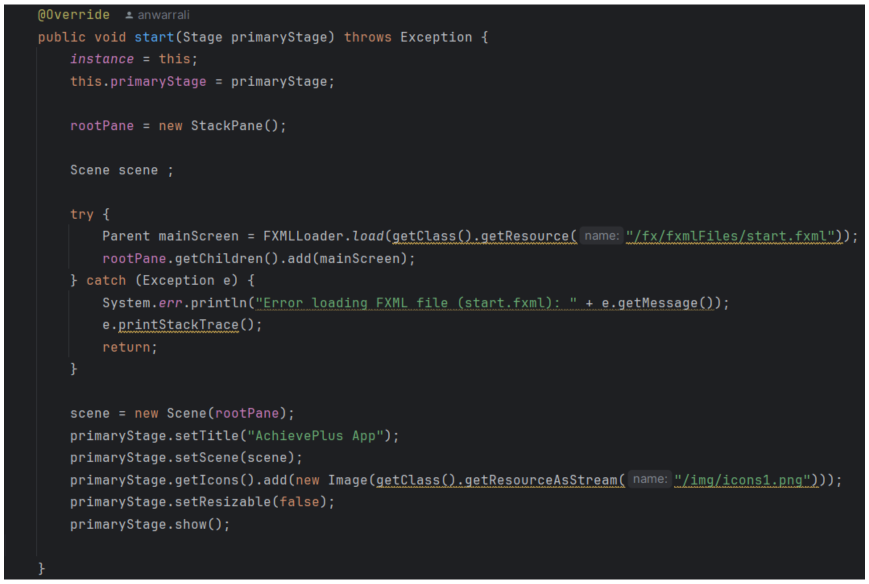
ADD

fig. 18 adding questions screen



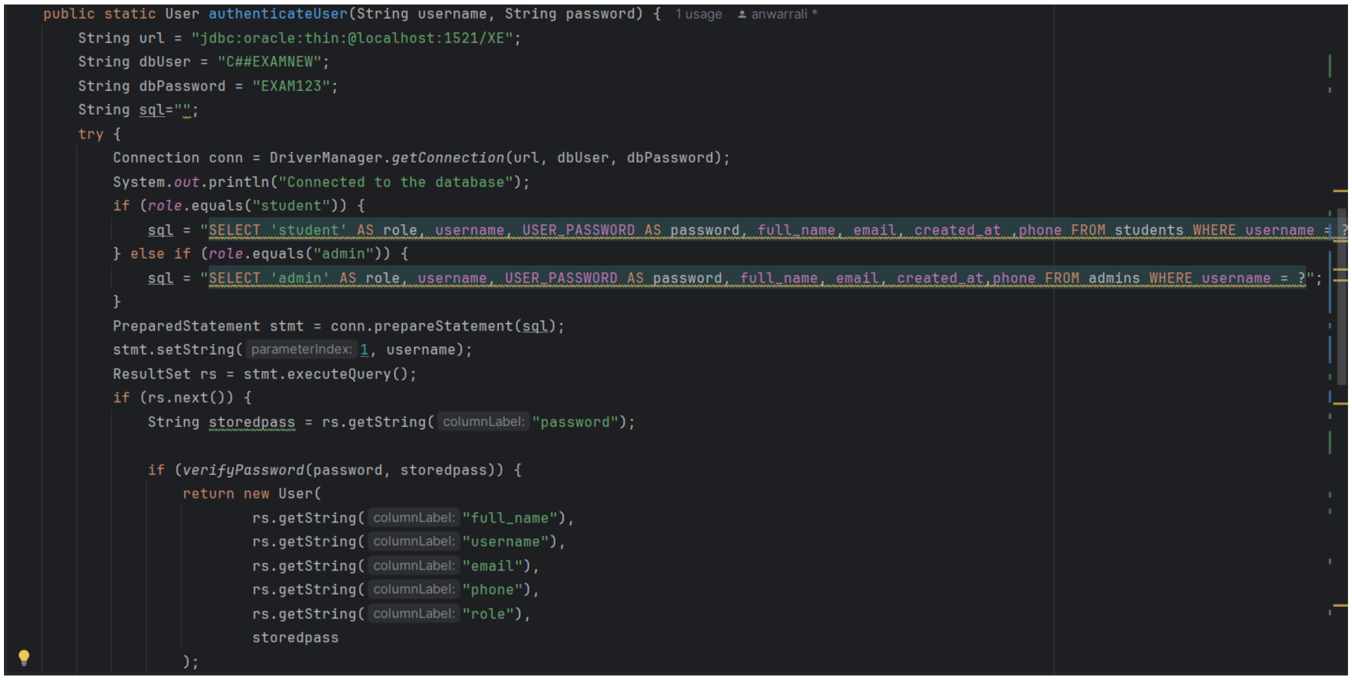
anwar ali	121	0
anwar ali	124	10
anwar ali	125	1
anwar ali	164	0
anwar ali	181	1
anwar ali	182	0
anwar ali	184	6
Layan	221	0
Layan	223	121
anwar ali	165	0
anwar ali	166	0
anwar ali	167	0
anwar ali	82	0

fig. 19 select attempt to send feedback to students



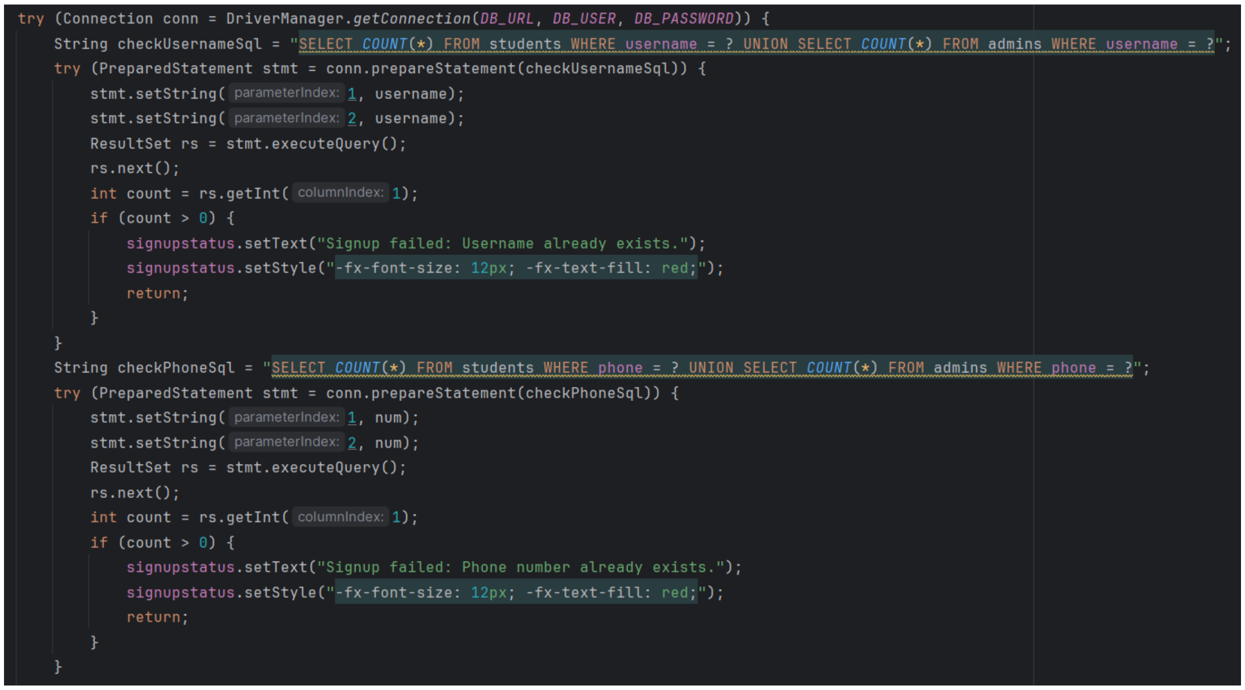
```java
@Override
public void start(Stage primaryStage) throws Exception {
    instance = this;
    this.primaryStage = primaryStage;

    rootPane = new StackPane();

    Scene scene ;

    try {
        Parent mainScreen = FXMLLoader.load(getClass().getResource("/fx/fxmlFiles/start.fxml"));
        rootPane.getChildren().add(mainScreen);
    } catch (Exception e) {
        System.err.println("Error loading FXML file (start.fxml): " + e.getMessage());
        e.printStackTrace();
        return;
    }

    scene = new Scene(rootPane);
    primaryStage.setTitle("AchievePlus App");
    primaryStage.setScene(scene);
    primaryStage.getIcons().add(new Image(getClass().getResourceAsStream("/img/icons1.png")));
    primaryStage.setResizable(false);
    primaryStage.show();

}
```

fig. 20 entry point code



```java
public static User authenticateUser(String username, String password) {  1 usage  anwarrali *
    String url = "jdbc:oracle:thin:@localhost:1521/XE";
    String dbUser = "C##EXAMNEW";
    String dbPassword = "EXAM123";
    String sql="";
    try {
        Connection conn = DriverManager.getConnection(url, dbUser, dbPassword);
        System.out.println("Connected to the database");
        if (role.equals("student")) {
            sql = "SELECT 'student' AS role, username, USER_PASSWORD AS password, full_name, email, created_at ,phone FROM students WHERE username = ?
        } else if (role.equals("admin")) {
            sql = "SELECT 'admin' AS role, username, USER_PASSWORD AS password, full_name, email, created_at,phone FROM admins WHERE username = ?";
        }
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setString( parameterIndex: 1, username);
        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
            String storedpass = rs.getString( columnLabel: "password");

            if (verifyPassword(password, storedpass)) {
                return new User(
                        rs.getString( columnLabel: "full_name"),
                        rs.getString( columnLabel: "username"),
                        rs.getString( columnLabel: "email"),
                        rs.getString( columnLabel: "phone"),
                        rs.getString( columnLabel: "role"),
                        storedpass
                );
```

fig 21 login validation code



```java
try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD)) {
    String checkUsernameSql = "SELECT COUNT(*) FROM students WHERE username = ? UNION SELECT COUNT(*) FROM admins WHERE username = ?";
    try (PreparedStatement stmt = conn.prepareStatement(checkUsernameSql)) {
        stmt.setString( parameterIndex: 1, username);
        stmt.setString( parameterIndex: 2, username);
        ResultSet rs = stmt.executeQuery();
        rs.next();
        int count = rs.getInt( columnIndex: 1);
        if (count > 0) {
            signupstatus.setText("Signup failed: Username already exists.");
            signupstatus.setStyle("-fx-font-size: 12px; -fx-text-fill: red;");
            return;
        }
    }
    String checkPhoneSql = "SELECT COUNT(*) FROM students WHERE phone = ? UNION SELECT COUNT(*) FROM admins WHERE phone = ?";
    try (PreparedStatement stmt = conn.prepareStatement(checkPhoneSql)) {
        stmt.setString( parameterIndex: 1, num);
        stmt.setString( parameterIndex: 2, num);
        ResultSet rs = stmt.executeQuery();
        rs.next();
        int count = rs.getInt( columnIndex: 1);
        if (count > 0) {
            signupstatus.setText("Signup failed: Phone number already exists.");
            signupstatus.setStyle("-fx-font-size: 12px; -fx-text-fill: red;");
            return;
        }
    }
```

fig 22 signup validation code

test cases table:

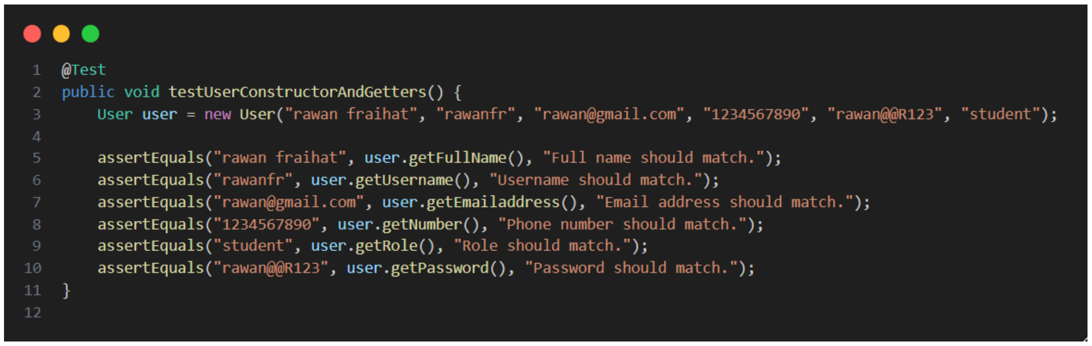
	scenarios	steps	Expected result
1	Admin login	1. Go to login page 2. Enter correct credentials 3. Click login	Admin is logged in and redirected to admin dashboard
2	Admin Adds Question to Exam	1. Click “Add Question” 2. Fill form with valid data 3. Save	Question is saved and shown in UI
3	Student Login	1. Go to login page 2. Enter student credentials	Student is redirected to available exams
4	Student Takes Exam	1. Login 2. Select exam 3. Answer all questions 4. Submit	Score is calculated, result displayed
5	Validation	1. Try submitting login or question form with empty fields or invalid credentials	Error message shown, form not submitted
6	Password Masking	1. Go to login 2. Enter password 3. Observe input	Password is hidden
7	User Feedback	1. Submit exam 2. Check feedback page	Student sees score + feedback

Appendix B: Testing

Unit testing results:

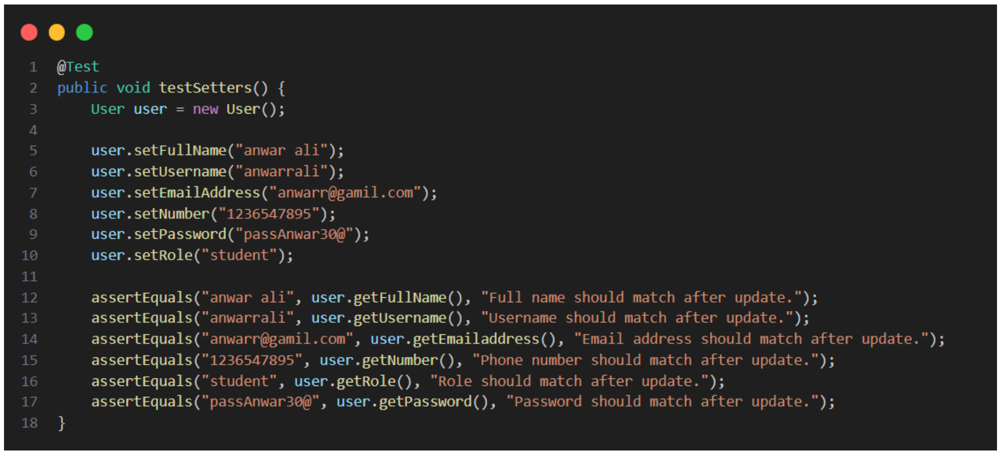
User class

Method Tested: Constructor & Getters



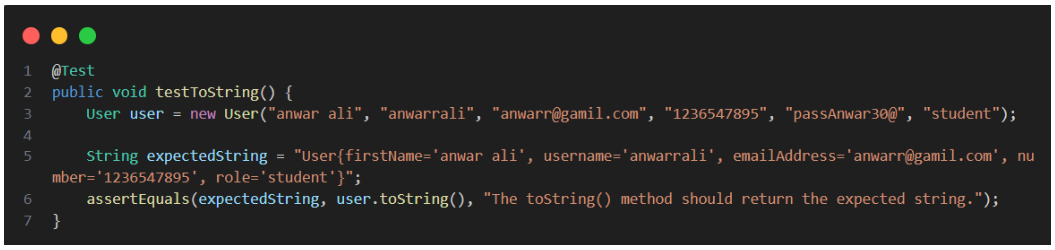
```java
@Test
public void testUserConstructorAndGetters() {
    User user = new User("rawan fraihat", "rawanfr", "rawan@gmail.com", "1234567890", "rawan@@R123", "student");

    assertEquals("rawan fraihat", user.getFullName(), "Full name should match.");
    assertEquals("rawanfr", user.getUsername(), "Username should match.");
    assertEquals("rawan@gmail.com", user.getEmailaddress(), "Email address should match.");
    assertEquals("1234567890", user.getNumber(), "Phone number should match.");
    assertEquals("student", user.getRole(), "Role should match.");
    assertEquals("rawan@@R123", user.getPassword(), "Password should match.");
}

```

Method Tested: Setters



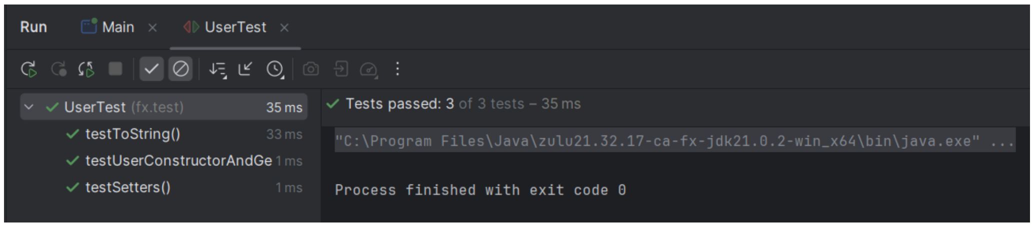
```java
@Test
public void testSetters() {
    User user = new User();

    user.setFullName("anwar ali");
    user.setUsername("anwarrali");
    user.setEmailAddress("anwarr@gamil.com");
    user.setNumber("1236547895");
    user.setPassword("passAnwar30@");
    user.setRole("student");

    assertEquals("anwar ali", user.getFullName(), "Full name should match after update.");
    assertEquals("anwarrali", user.getUsername(), "Username should match after update.");
    assertEquals("anwarr@gamil.com", user.getEmailaddress(), "Email address should match after update.");
    assertEquals("1236547895", user.getNumber(), "Phone number should match after update.");
    assertEquals("student", user.getRole(), "Role should match after update.");
    assertEquals("passAnwar30@", user.getPassword(), "Password should match after update.");
}
```

Method Tested: toString()



```java
@Test
public void testToString() {
    User user = new User("anwar ali", "anwarrali", "anwarr@gamil.com", "1236547895", "passAnwar30@", "student");

    String expectedString = "User{firstName='anwar ali', username='anwarrali', emailAddress='anwarr@gamil.com', number='1236547895', role='student'}";
    assertEquals(expectedString, user.toString(), "The toString() method should return the expected string.");
}
```

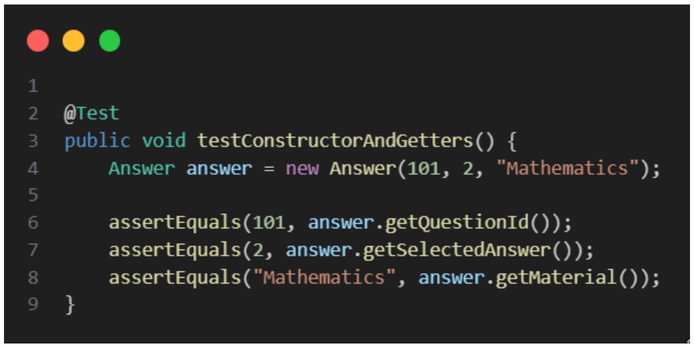
Test results



```
Run   Main ×   UserTest ×

UserTest (fx.test)                  35 ms    Tests passed: 3 of 3 tests – 35 ms
  testToString()                    33 ms    "C:\Program Files\Java\zulu21.32.17-ca-fx-jdk21.0.2-win_x64\bin\java.exe" ...
  testUserConstructorAndGe          1 ms
  testSetters()                     1 ms     Process finished with exit code 0
```

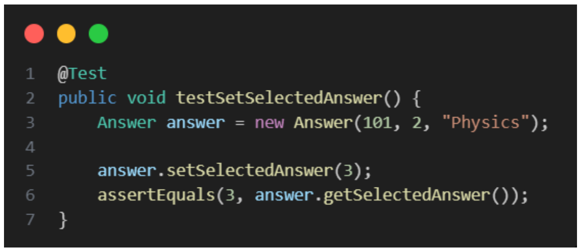
Answer class

Method Tested: Constructor & Getter



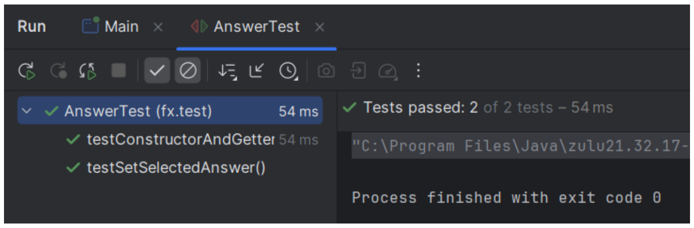
```java

@Test
public void testConstructorAndGetters() {
    Answer answer = new Answer(101, 2, "Mathematics");

    assertEquals(101, answer.getQuestionId());
    assertEquals(2, answer.getSelectedAnswer());
    assertEquals("Mathematics", answer.getMaterial());
}
```

Method Tested: selectedAnswer setter



```java
@Test
public void testSetSelectedAnswer() {
    Answer answer = new Answer(101, 2, "Physics");

    answer.setSelectedAnswer(3);
    assertEquals(3, answer.getSelectedAnswer());
}
```

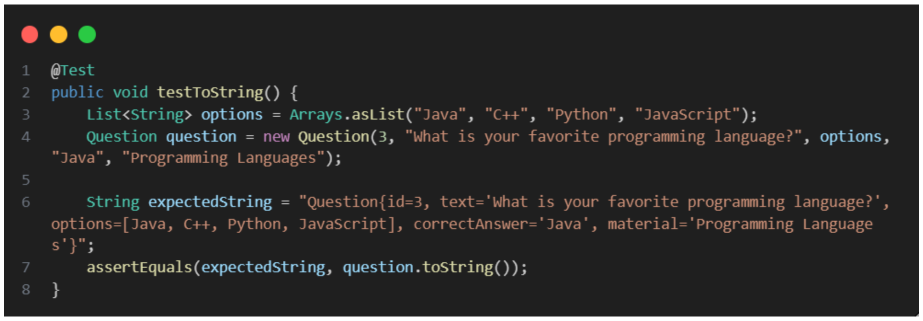
Test results



```
Run   Main ×   AnswerTest ×

AnswerTest (fx.test)          54 ms    Tests passed: 2 of 2 tests – 54 ms
  testConstructorAndGetter    54 ms    "C:\Program Files\Java\zulu21.32.17-
  testSetSelectedAnswer()
                                       Process finished with exit code 0
```

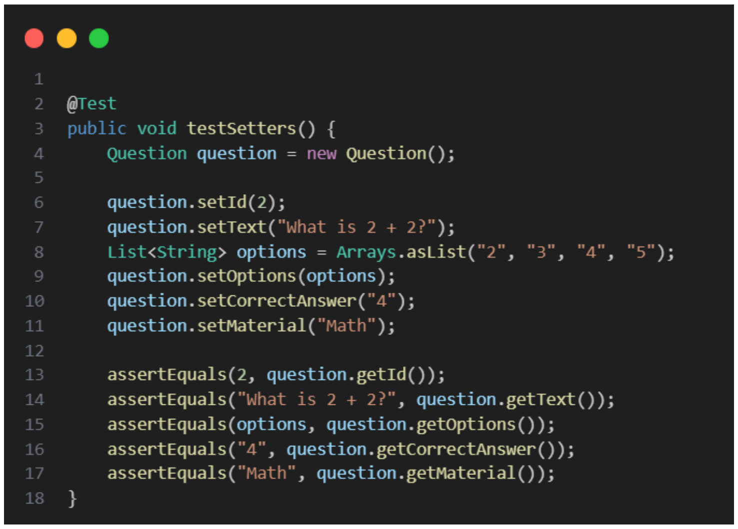
Question class

Method Tested: toString()



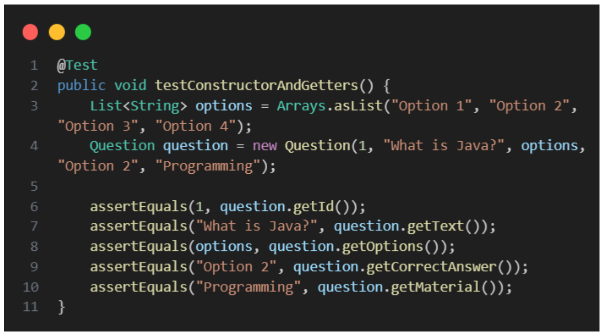
```java
@Test
public void testToString() {
    List<String> options = Arrays.asList("Java", "C++", "Python", "JavaScript");
    Question question = new Question(3, "What is your favorite programming language?", options, "Java", "Programming Languages");

    String expectedString = "Question{id=3, text='What is your favorite programming language?', options=[Java, C++, Python, JavaScript], correctAnswer='Java', material='Programming Languages'}";
    assertEquals(expectedString, question.toString());
}
```

Method Tested: Setters



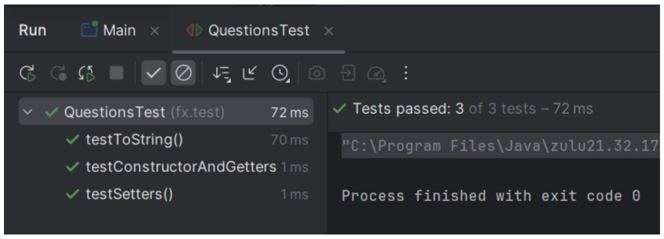
```java

@Test
public void testSetters() {
    Question question = new Question();

    question.setId(2);
    question.setText("What is 2 + 2?");
    List<String> options = Arrays.asList("2", "3", "4", "5");
    question.setOptions(options);
    question.setCorrectAnswer("4");
    question.setMaterial("Math");

    assertEquals(2, question.getId());
    assertEquals("What is 2 + 2?", question.getText());
    assertEquals(options, question.getOptions());
    assertEquals("4", question.getCorrectAnswer());
    assertEquals("Math", question.getMaterial());
}
```

Method Tested: Constructor & Getters



```java
@Test
public void testConstructorAndGetters() {
    List<String> options = Arrays.asList("Option 1", "Option 2", "Option 3", "Option 4");
    Question question = new Question(1, "What is Java?", options, "Option 2", "Programming");

    assertEquals(1, question.getId());
    assertEquals("What is Java?", question.getText());
    assertEquals(options, question.getOptions());
    assertEquals("Option 2", question.getCorrectAnswer());
    assertEquals("Programming", question.getMaterial());
}
```

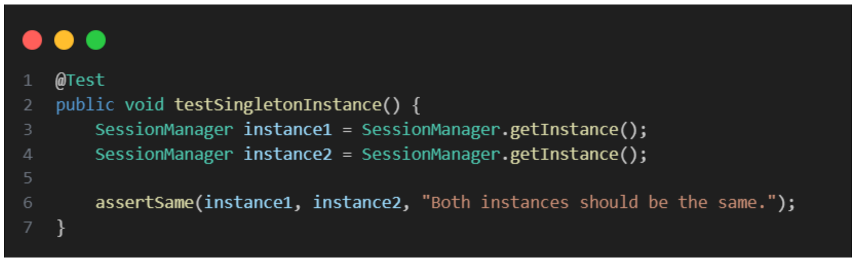
Test results



```
Run    Main ×    QuestionsTest ×

QuestionsTest (fx.test)        72 ms    Tests passed: 3 of 3 tests – 72 ms
  testToString()               70 ms    "C:\Program Files\Java\zulu21.32.17
  testConstructorAndGetters     1 ms
  testSetters()                 1 ms    Process finished with exit code 0
```

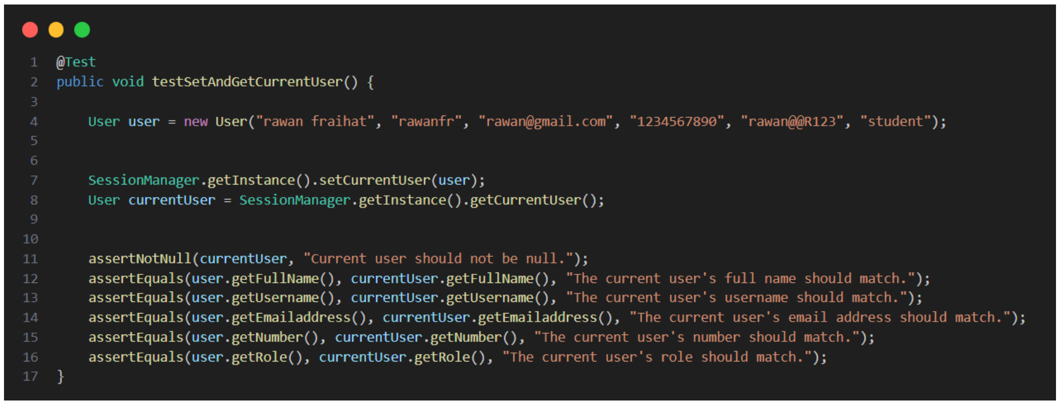
SessionManager class

Test case: Singleton Instance Verification



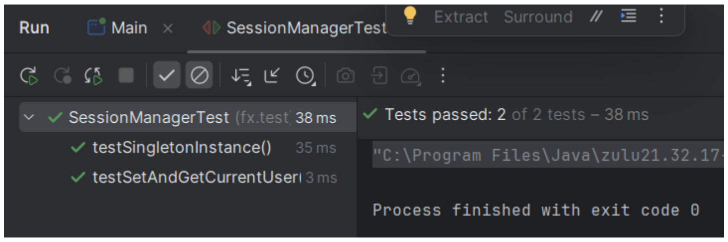
```java
@Test
public void testSingletonInstance() {
    SessionManager instance1 = SessionManager.getInstance();
    SessionManager instance2 = SessionManager.getInstance();

    assertSame(instance1, instance2, "Both instances should be the same.");
}
```

Test Case: Set and Get Current User



```java
@Test
public void testSetAndGetCurrentUser() {

    User user = new User("rawan fraihat", "rawanfr", "rawan@gmail.com", "1234567890", "rawan@@R123", "student");


    SessionManager.getInstance().setCurrentUser(user);
    User currentUser = SessionManager.getInstance().getCurrentUser();


    assertNotNull(currentUser, "Current user should not be null.");
    assertEquals(user.getFullName(), currentUser.getFullName(), "The current user's full name should match.");
    assertEquals(user.getUsername(), currentUser.getUsername(), "The current user's username should match.");
    assertEquals(user.getEmailaddress(), currentUser.getEmailaddress(), "The current user's email address should match.");
    assertEquals(user.getNumber(), currentUser.getNumber(), "The current user's number should match.");
    assertEquals(user.getRole(), currentUser.getRole(), "The current user's role should match.");
}
```

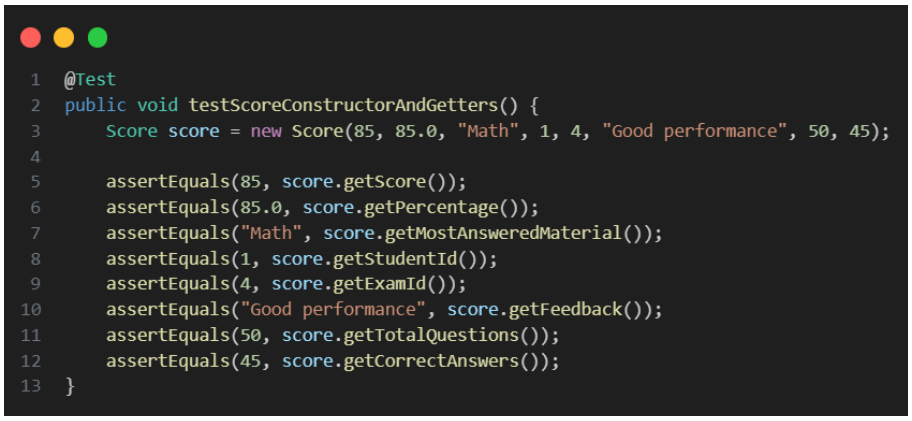
Test results



```
Run    Main ×    SessionManagerTest    Extract  Surround

SessionManagerTest (fx.test  38 ms    Tests passed: 2 of 2 tests – 38 ms
  testSingletonInstance()    35 ms    "C:\Program Files\Java\zulu21.32.17
  testSetAndGetCurrentUser(   3 ms
                                      Process finished with exit code 0
```

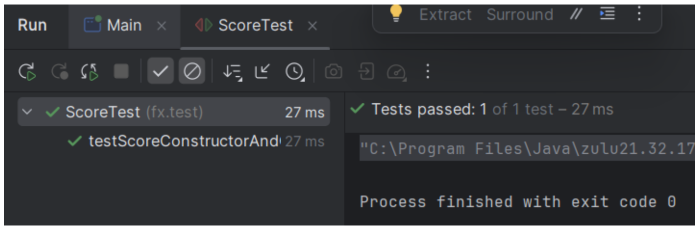
Score

Method Tested: Constructor & Getters



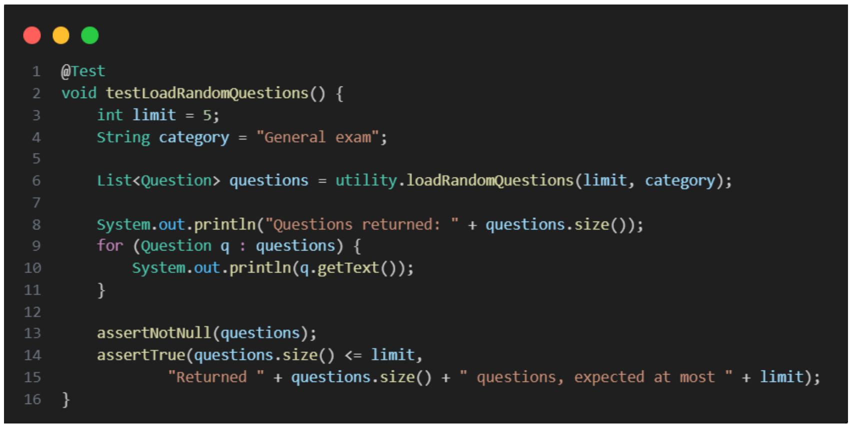
```java
@Test
public void testScoreConstructorAndGetters() {
    Score score = new Score(85, 85.0, "Math", 1, 4, "Good performance", 50, 45);

    assertEquals(85, score.getScore());
    assertEquals(85.0, score.getPercentage());
    assertEquals("Math", score.getMostAnsweredMaterial());
    assertEquals(1, score.getStudentId());
    assertEquals(4, score.getExamId());
    assertEquals("Good performance", score.getFeedback());
    assertEquals(50, score.getTotalQuestions());
    assertEquals(45, score.getCorrectAnswers());
}
```

Test results



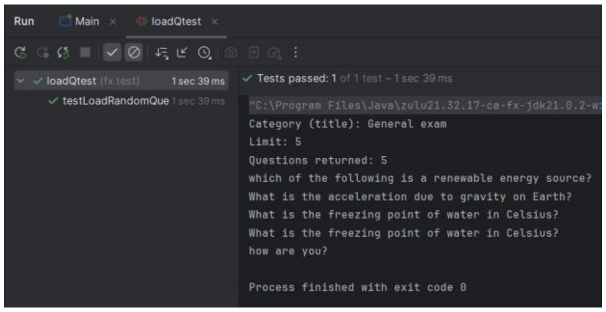
```
ScoreTest (fx.test)  27 ms
  testScoreConstructorAnd  27 ms

Tests passed: 1 of 1 test – 27 ms
"C:\Program Files\Java\zulu21.32.17

Process finished with exit code 0
```

Load questions from DB method in utility



```java
@Test
void testLoadRandomQuestions() {
    int limit = 5;
    String category = "General exam";

    List<Question> questions = utility.loadRandomQuestions(limit, category);

    System.out.println("Questions returned: " + questions.size());
    for (Question q : questions) {
        System.out.println(q.getText());
    }

    assertNotNull(questions);
    assertTrue(questions.size() <= limit,
            "Returned " + questions.size() + " questions, expected at most " + limit);
}
```

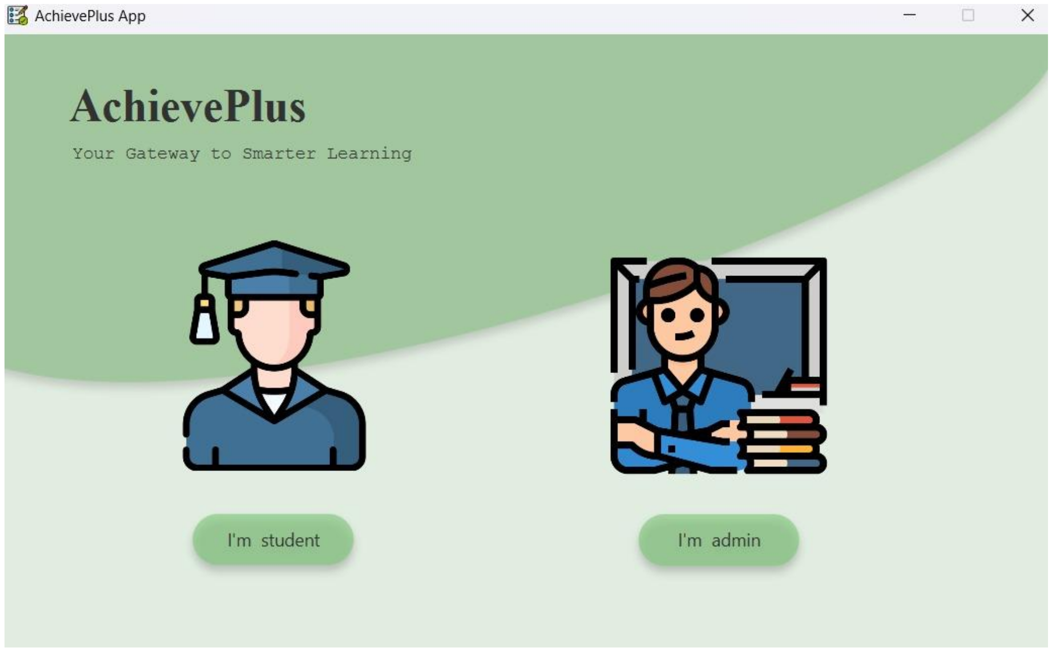
Test result



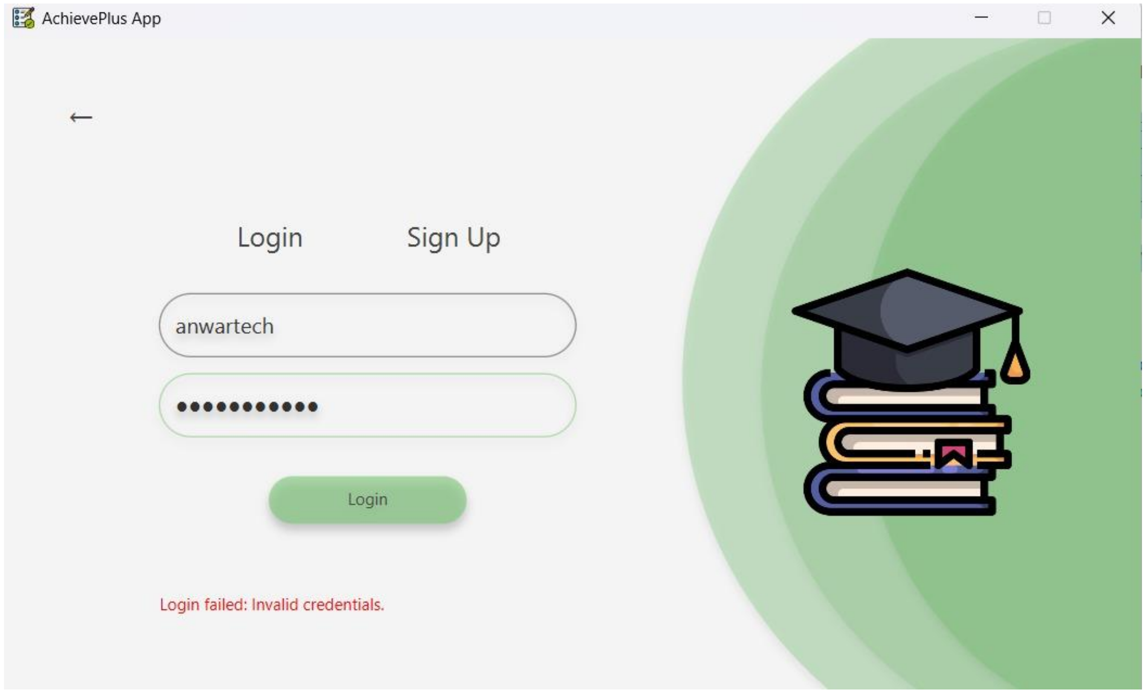
```
loadQtest (fx.test)  1 sec 39 ms
  testLoadRandomQue  1 sec 39 ms

Tests passed: 1 of 1 test – 1 sec 39 ms
"C:\Program Files\Java\zulu21.32.17-ca-fx-jdk21.0.2-w
Category (title): General exam
Limit: 5
Questions returned: 5
which of the following is a renewable energy source?
What is the acceleration due to gravity on Earth?
What is the freezing point of water in Celsius?
What is the freezing point of water in Celsius?
how are you?

Process finished with exit code 0
```

UI testing results:

Entry point of the system start from the following UI to slect the user role if admin or student:

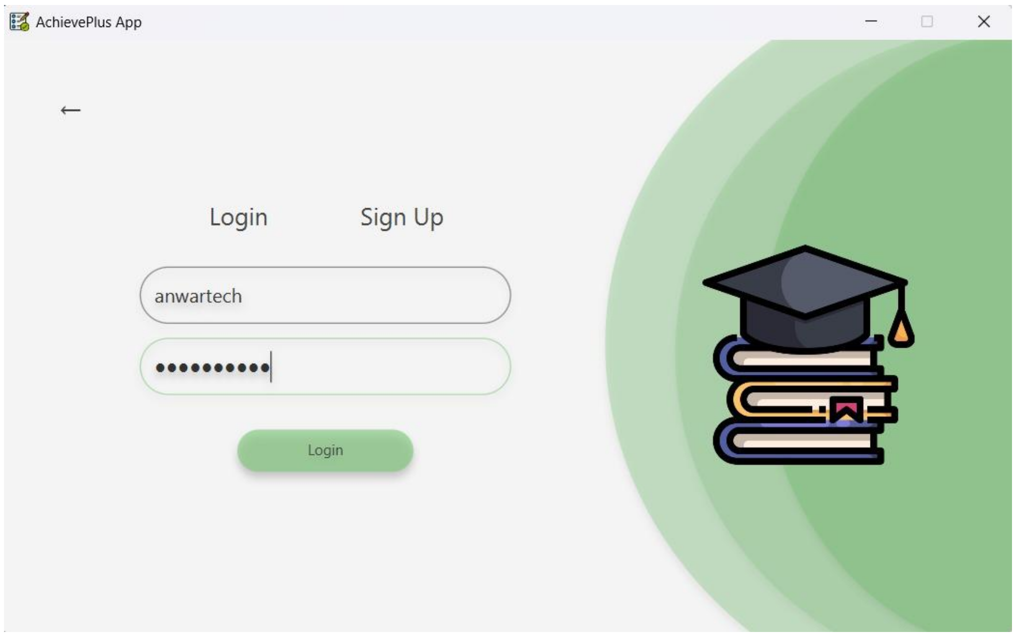


If student write invalid credentials system shows the message in UI for student

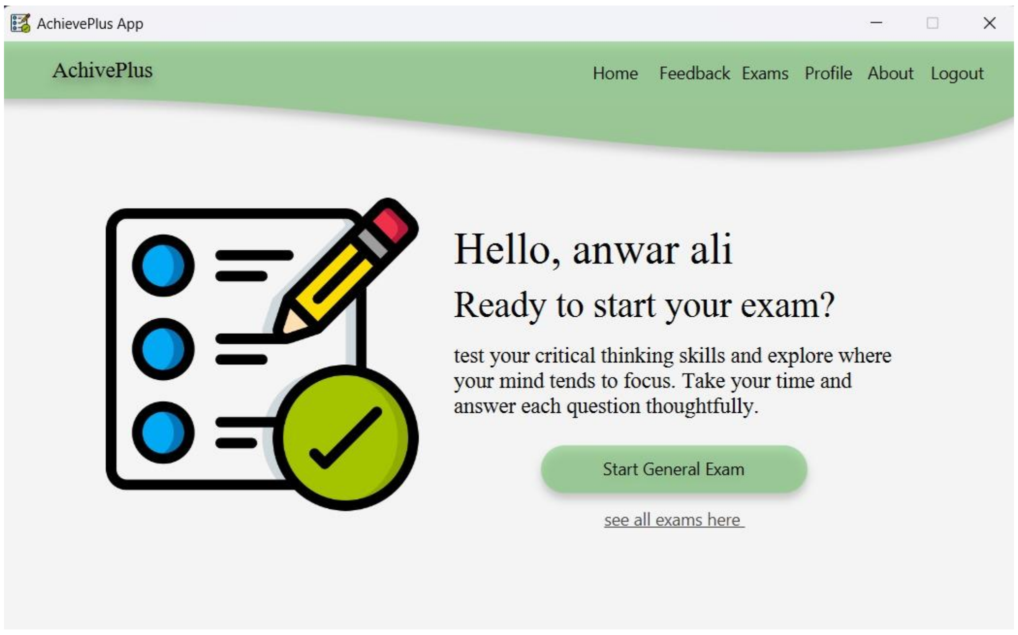


Login failed: Invalid credentials.

When student write valid credentials login to the system:



The main page student will see after login:

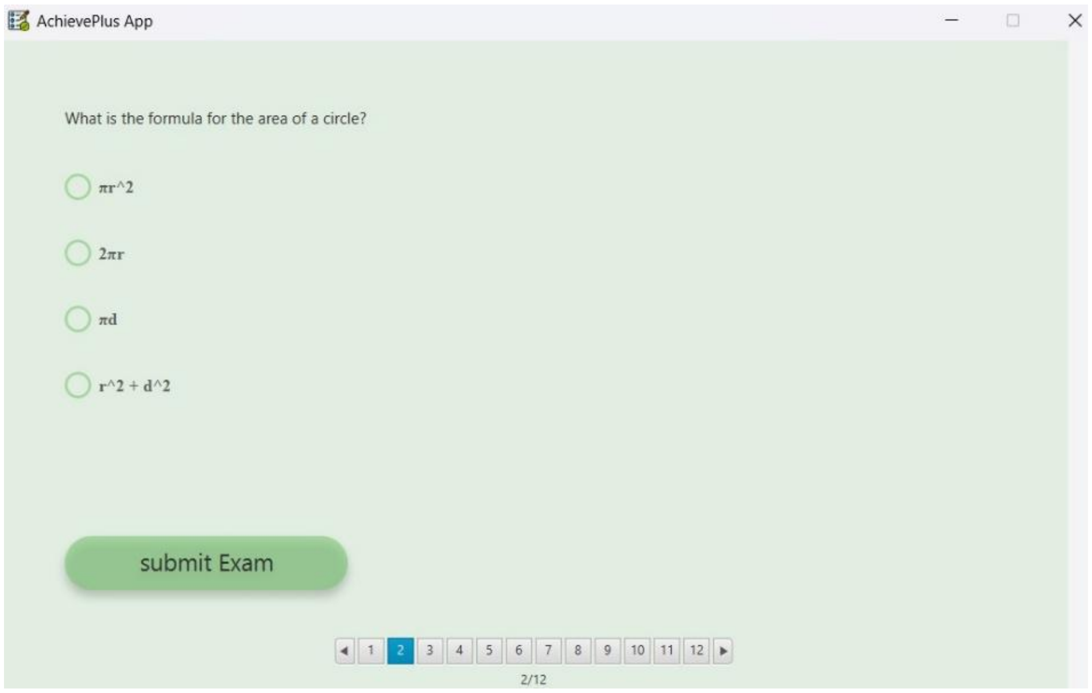


Hello, anwar ali

Ready to start your exam?

test your critical thinking skills and explore where your mind tends to focus. Take your time and answer each question thoughtfully.

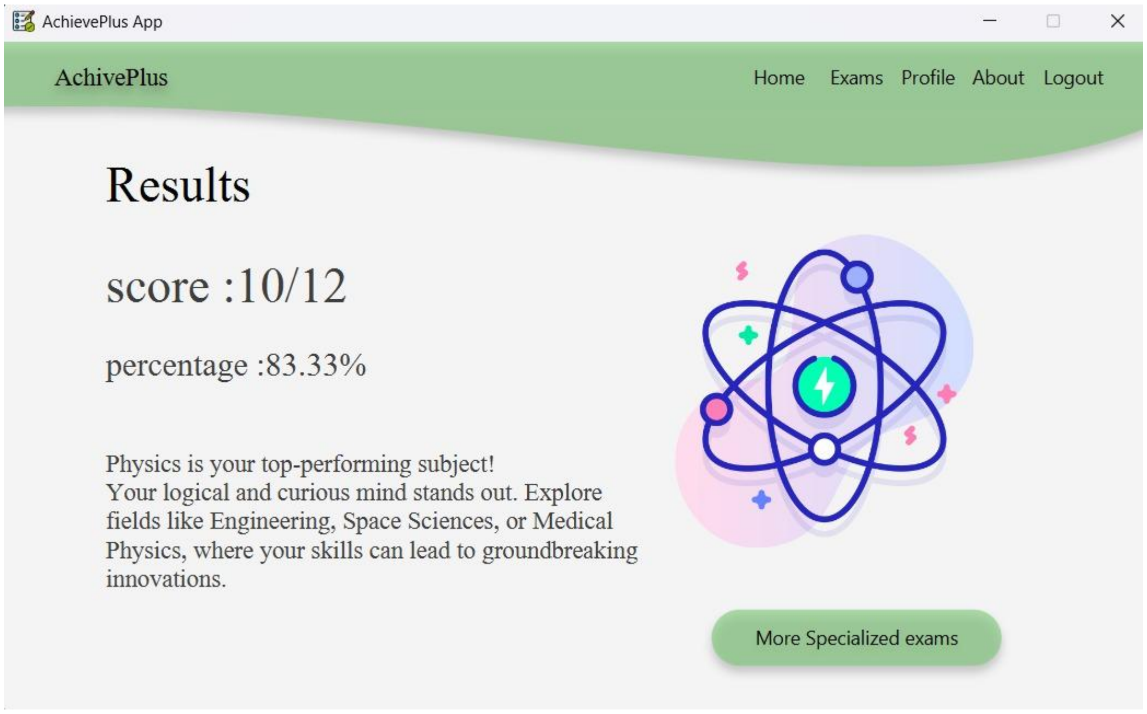
When click to "start general exam " button will start the exam , After loading the exam from database and submit the answers will save the answer to the database and recorded as an attempt to let admin see the results and give feedback


AchievePlus App

What is the formula for the area of a circle?

- πr^2
- 2πr
- πd
- r^2 + d^2

Then results UI shown :


AchievePlus App

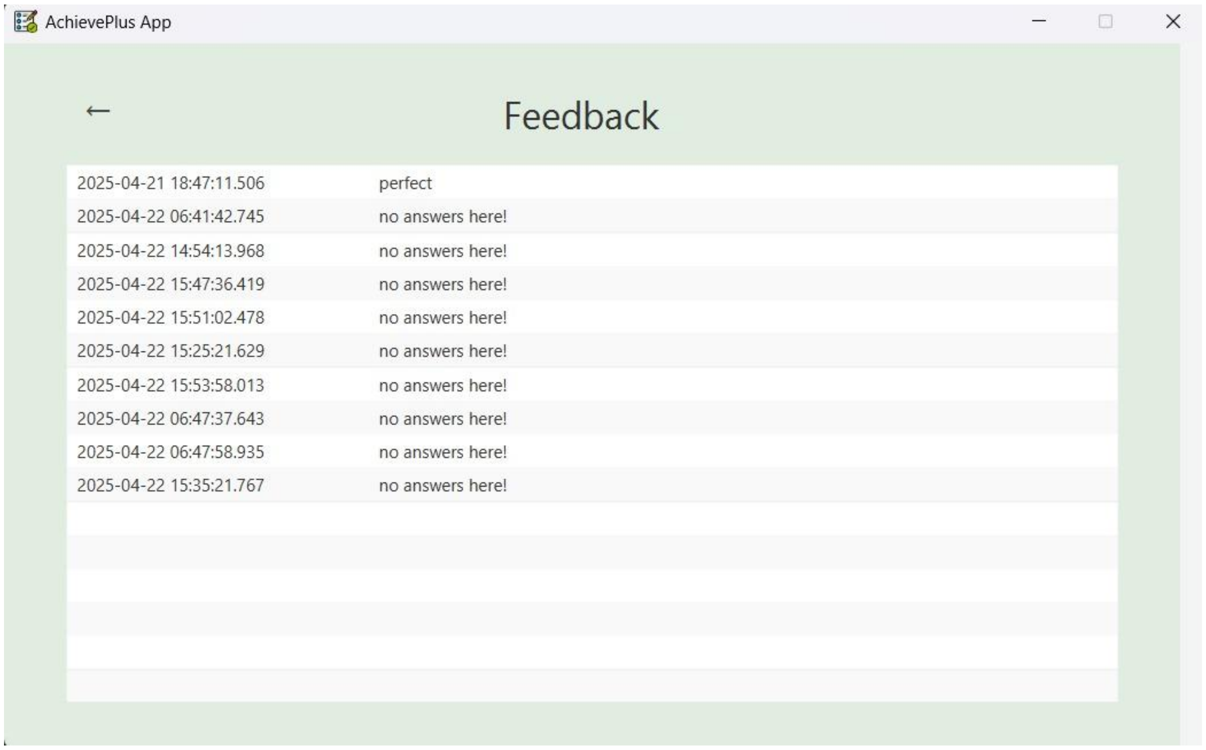
Results

score :10/12

percentage :83.33%

Physics is your top-performing subject!
Your logical and curious mind stands out. Explore fields like Engineering, Space Sciences, or Medical Physics, where your skills can lead to groundbreaking innovations.

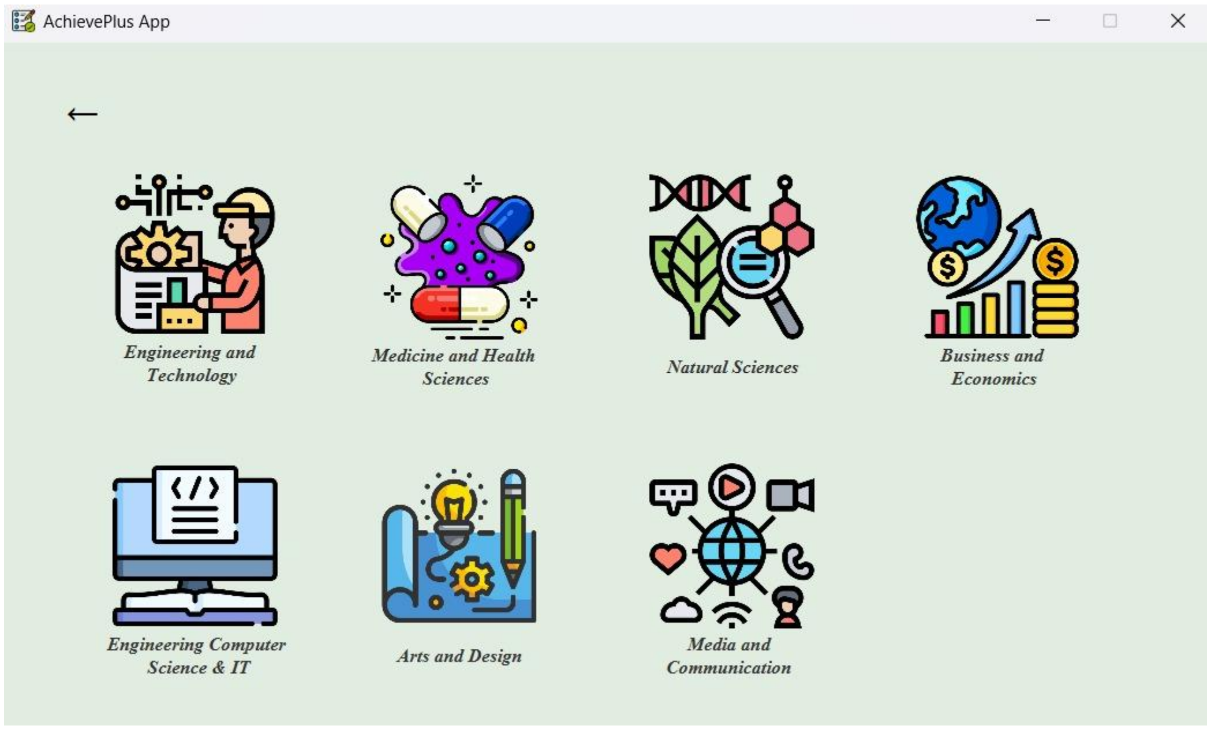
Every Student can see feedback comes from admins for every exam attempt


AchievePlus App

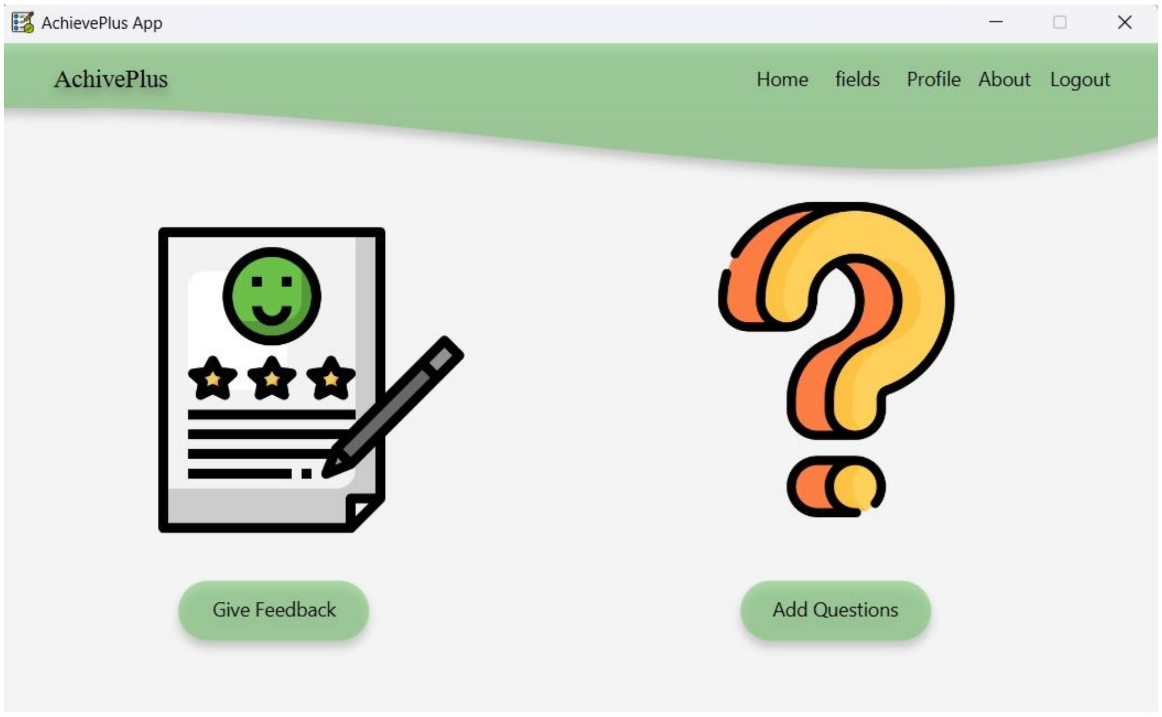
Feedback

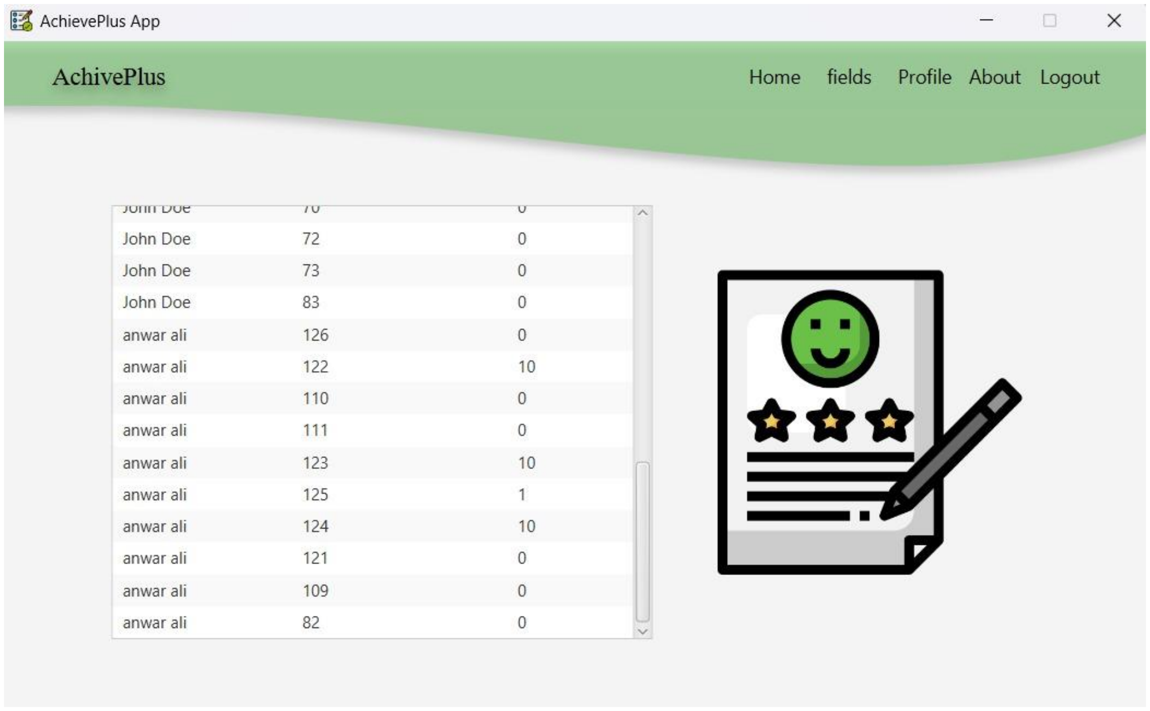
2025-04-21 18:47:11.506	perfect
2025-04-22 06:41:42.745	no answers here!
2025-04-22 14:54:13.968	no answers here!
2025-04-22 15:47:36.419	no answers here!
2025-04-22 15:51:02.478	no answers here!
2025-04-22 15:25:21.629	no answers here!
2025-04-22 15:53:58.013	no answers here!
2025-04-22 06:47:37.643	no answers here!
2025-04-22 06:47:58.935	no answers here!
2025-04-22 15:35:21.767	no answers here!

Student can select any specialized exam and will do the same process as general exam



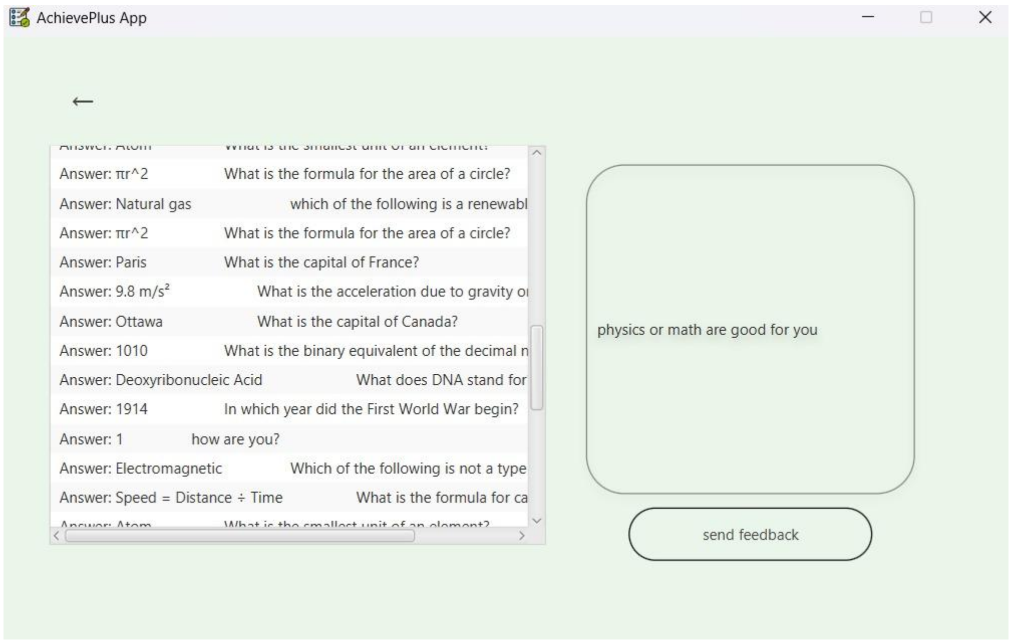
If from the start admin was chosen and login to admin account the main UI will shown as the following, let admin select if want to add questions to database or send feedback for students attempts:





John Doe	72	0
John Doe	73	0
John Doe	83	0
anwar ali	126	0
anwar ali	122	10
anwar ali	110	0
anwar ali	111	0
anwar ali	123	10
anwar ali	125	1
anwar ali	124	10
anwar ali	121	0
anwar ali	109	0
anwar ali	82	0

After double click for any student attempt will show the answers of the student and let the admin to send feedback to database



AchievePlus App

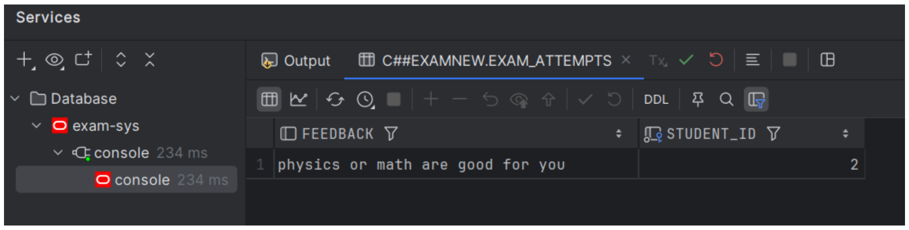
←

Answer: πr^2	What is the formula for the area of a circle?
Answer: Natural gas	which of the following is a renewabl
Answer: πr^2	What is the formula for the area of a circle?
Answer: Paris	What is the capital of France?
Answer: 9.8 m/s²	What is the acceleration due to gravity o
Answer: Ottawa	What is the capital of Canada?
Answer: 1010	What is the binary equivalent of the decimal n
Answer: Deoxyribonucleic Acid	What does DNA stand for
Answer: 1914	In which year did the First World War begin?
Answer: 1	how are you?
Answer: Electromagnetic	Which of the following is not a type
Answer: Speed = Distance ÷ Time	What is the formula for ca

physics or math are good for you

send feedback

As shown at the following ,feedback sent to DB for attempt_id=109:



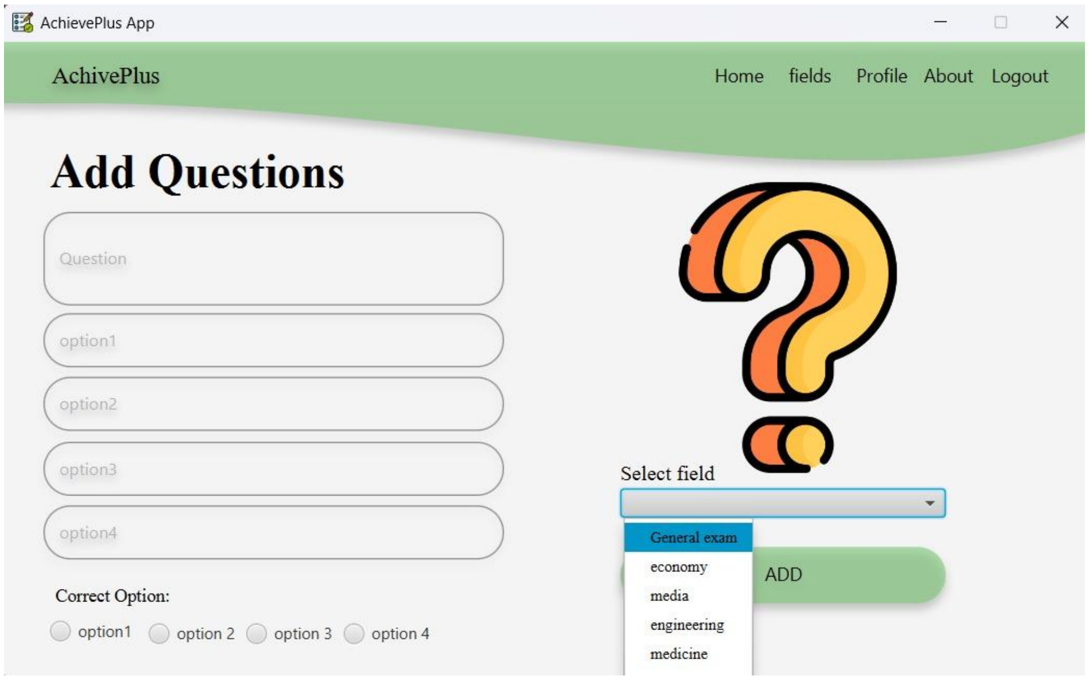
Services

Database
exam-sys
console 234 ms
console 234 ms

Output   C##EXAMNEW.EXAM_ATTEMPTS

```
  FEEDBACK                            STUDENT_ID
1 physics or math are good for you             2
```

If admin select add question so the following UI will displayed:



AchievePlus App

AchivePlus Home fields Profile About Logout

Add Questions

Question

option1

option2

option3

option4

Correct Option:

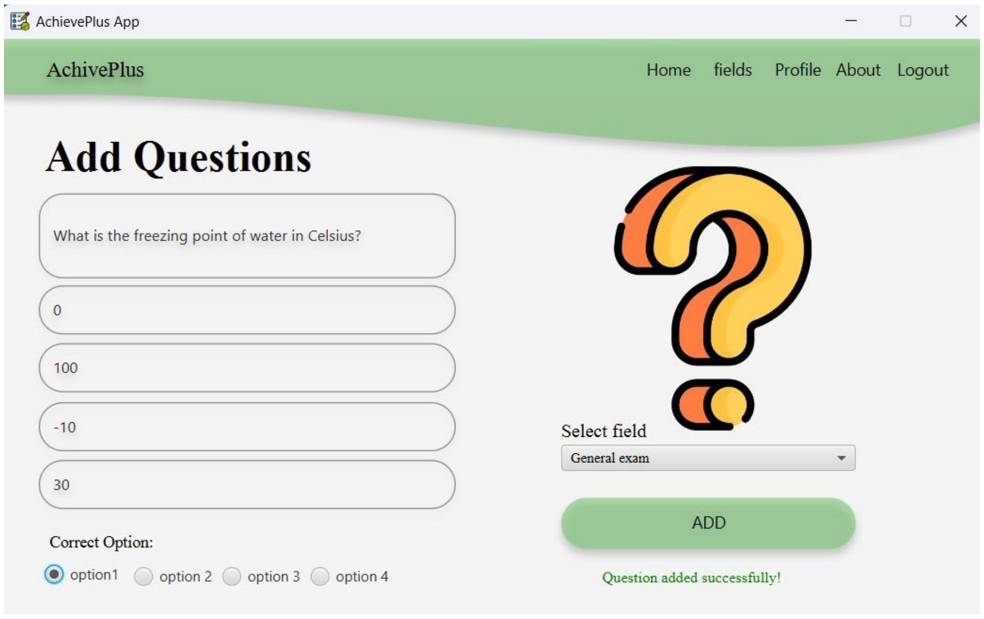
option1 option 2 option 3 option 4

Select field

General exam
economy
media
engineering
medicine

ADD

Then can add questions and give a message that question added successfully :



Add Questions

What is the freezing point of water in Celsius?

0

100

-10

30

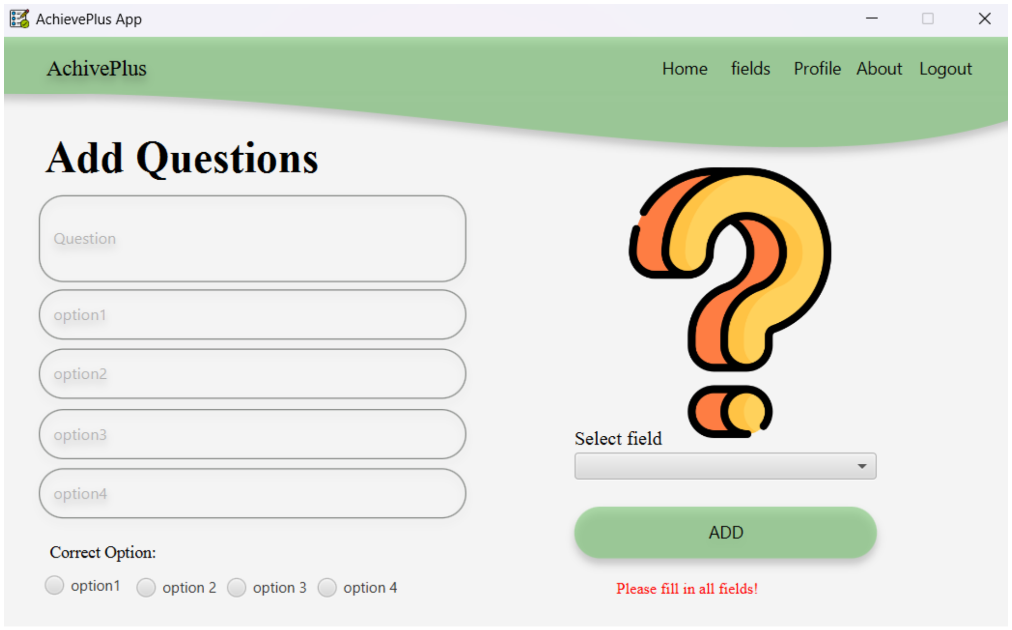
Correct Option:

[✓ option1] option 2 option 3 option 4

Select field: General exam

Question added successfully!

If the admin click on add button with empty fields will give the user an error message :



Add Questions

Question

option1

option2

option3

option4

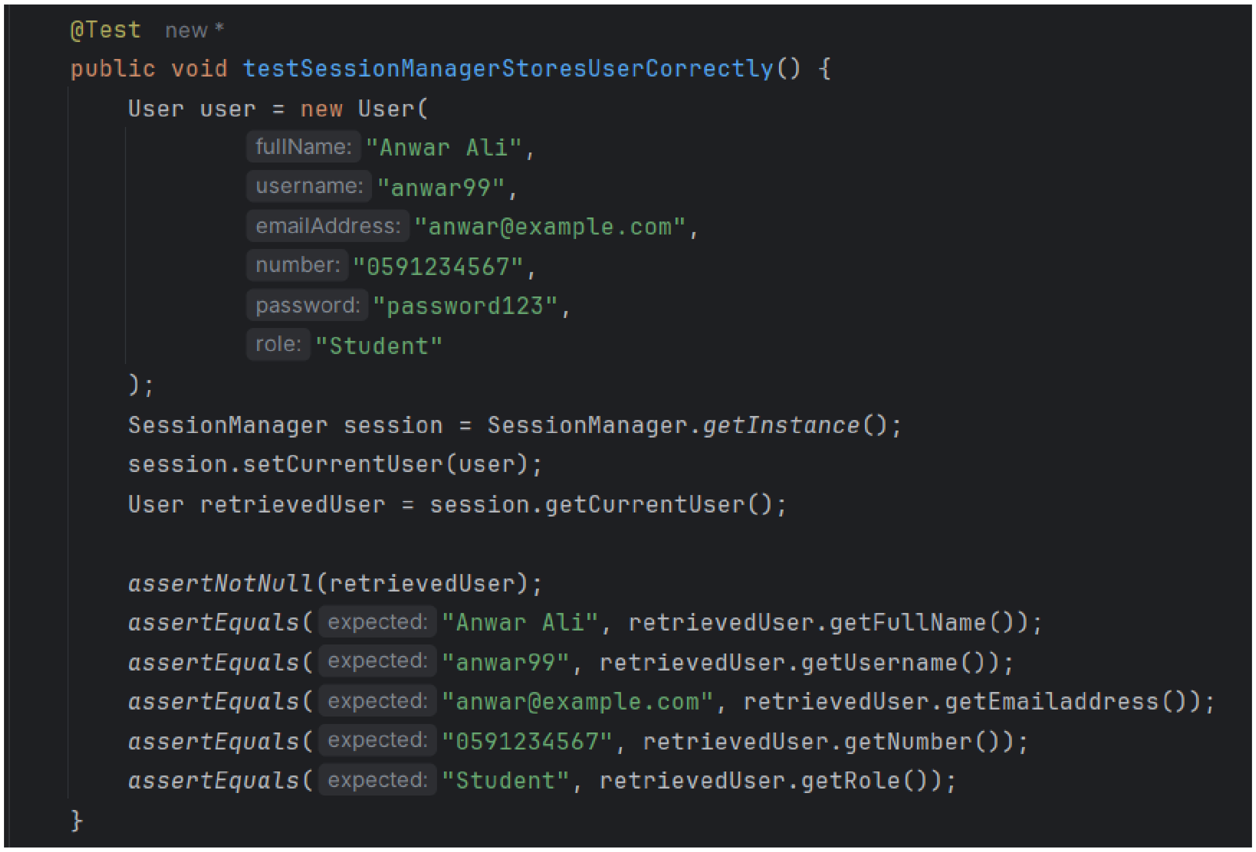
Correct Option:

option1 option 2 option 3 option 4

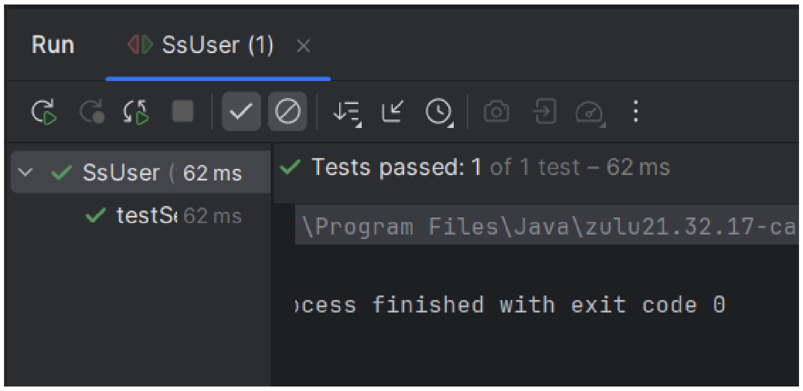
Select field

Please fill in all fields!

Integration testing results



```
@Test  new *
public void testSessionManagerStoresUserCorrectly() {
    User user = new User(
            fullName: "Anwar Ali",
            username: "anwar99",
            emailAddress: "anwar@example.com",
            number: "0591234567",
            password: "password123",
            role: "Student"
    );
    SessionManager session = SessionManager.getInstance();
    session.setCurrentUser(user);
    User retrievedUser = session.getCurrentUser();

    assertNotNull(retrievedUser);
    assertEquals( expected: "Anwar Ali", retrievedUser.getFullName());
    assertEquals( expected: "anwar99", retrievedUser.getUsername());
    assertEquals( expected: "anwar@example.com", retrievedUser.getEmailaddress());
    assertEquals( expected: "0591234567", retrievedUser.getNumber());
    assertEquals( expected: "Student", retrievedUser.getRole());
}
```



```
✓ Tests passed: 1 of 1 test – 62 ms

\Program Files\Java\zulu21.32.17-ca

cess finished with exit code 0
```