QuizaApp

By
Group - 9 :
Vinod Nagabhushan Rao Shivaram Samanth
Nayan Deep Vemula
SuryaTeja Duggi
Shriram Suryanarayanan
Nishant Bochare

Table of Contents

1.Introduction

QuizaApp is a quiz taking software for school students. Professors log into the system to create quizzes and check students' performance. Professors will post the quiz for students at a particular time. Students log into the system and simply click on a quiz to start the quiz. Professors can also view results of a quiz after students had taken a test.

Objective
To design the above System using Object Oriented design philosophy implement the design in Java.

Why this is a project related to the class?
The design of this System is based on many concepts taught in class like Object modeling, Abstraction, Encapsulation, Modularity, Inheritance etc., which are some of the most important concepts in Object Oriented Design.

Core System Functionalities:

- Allows Professors to create quizzes and check each student's performance in a particular quiz.
- Allows Students to select a quiz after log in and take the quiz.
- Provides a GUI where Professors add questions to a quiz by typing the question and various options for that question. They also pick a correct answer and a topic name for that question.
- Provides functionality for Professors to post the quiz for students whenever they want to conduct.
- Provides functionality for Students to exit the quiz at any time.
- Provides functionality for Students to check the time remaining in the examination.
- Allows Students to check their performance immediately after the quiz ends.

- The performances of the students are displayed using graphs, charts etc. The Professor can check their performance in a particular topic, related to other students in the class etc.

2.System Architecture Design

The System consists of the following important modules:

- QuizKit
- Authentication System
- TimingSystem
- LoggingSystem
- DataStorage
- UserInterface

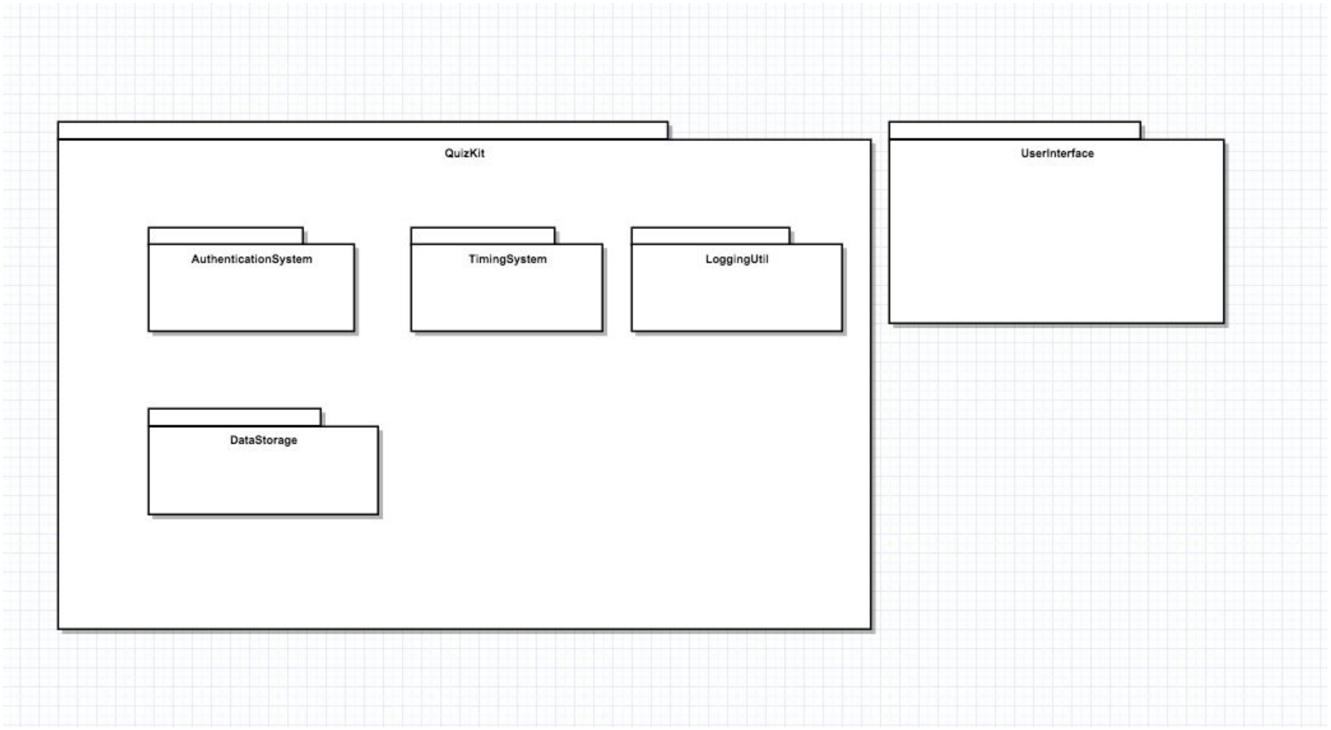

Fig 1: Architecture of QuizaApp

The System is architected in such a way that the application can be ported to other Java platforms like Android with little modifications. The User Interface module will would vary depending on the type of platform on which the application is run.

3.Detailed Description of Components:

The following package diagram describes the contents of each module of the System.

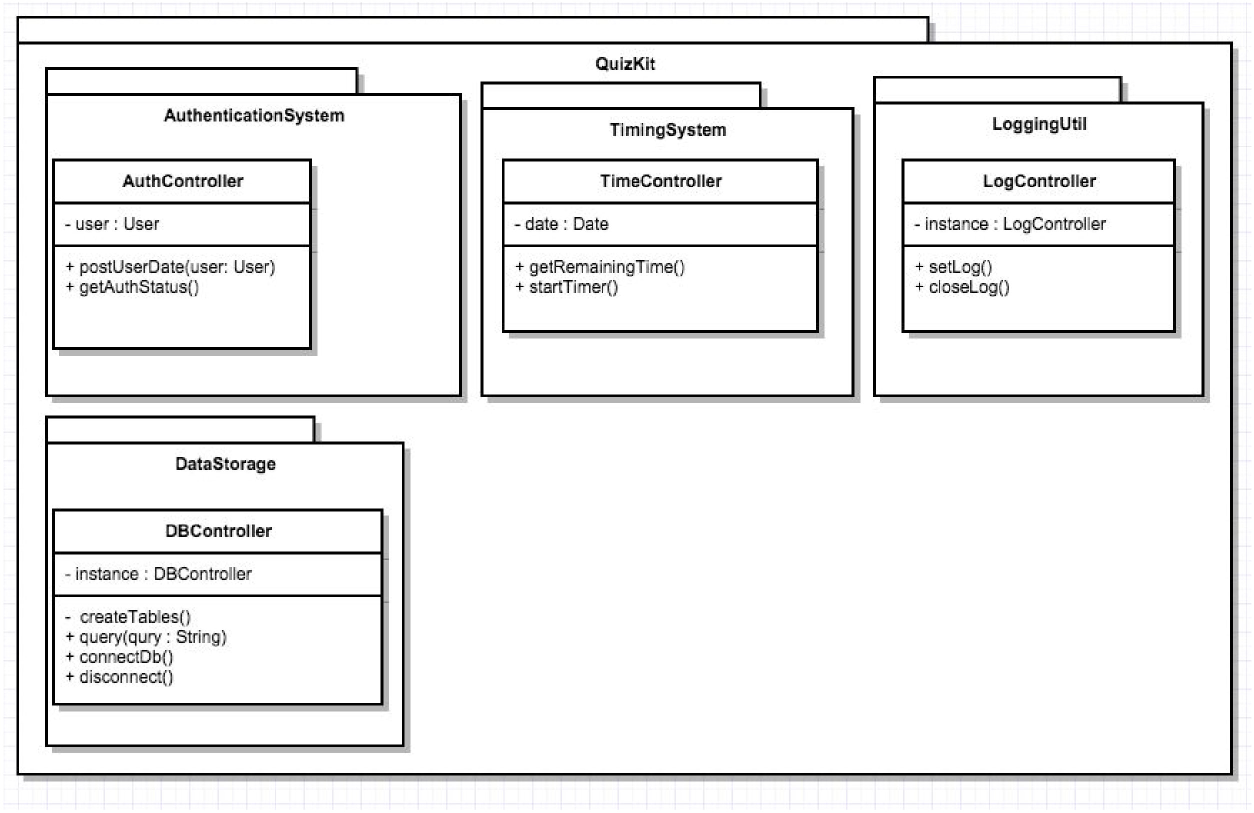

Fig 2: Detailed Component description of the System

QuizKit

QuizKit consists of the following core classes of the System:

- User
- Student
- Professor
- Quiz
- Question
- Options.

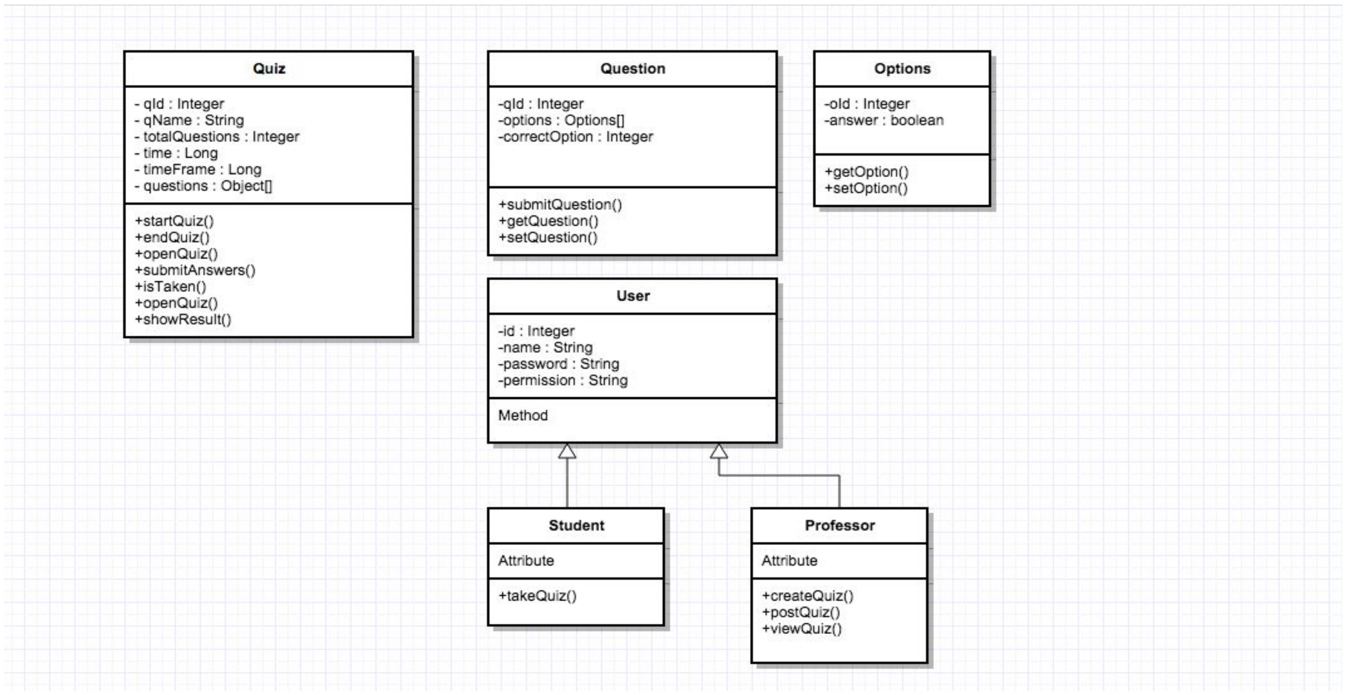

Fig 3: Architecture of QuizKit module

The User class is the base class which contains the variables and methods shared by both the Student class and Professor class. For example, both the students and professors will have an id, name, password. So all these variables are present in User class. The User class also contains the get and set methods for these variables. Encapsulation is used so that only the necessary classes can access these properties.

The Student class contains its own unique methods like takeQuiz() and it extends the User class. In the application, the UI guides each kind of user's behavior.

The Professor class extends the User class and contains methods like createQuiz(), postQuiz(), viewQuiz(). It also inherits the get and set methods from the User class just like the Student class.

Quiz class is where all the action takes place. Each quiz will have a name, description, number of questions, time limit and questionnaire. The Quiz class' design is used inside the database schema as well. The class contains methods like startQuiz(), stopQuiz(), submitAnswers() etc.

The Question class contains variables like questionId, question, options and a correct option. It also contains the get and set methods for initialising the variables.

The Options class is used inside the Questions class. As the System is based on Multiple Choice Questions (MCQs), each question will have four options. So, the Options class will have variables like answer, option name.

Authentication System

The Authentication System is used to identify if a user is a Student or a Professor after a user enters his credentials. Currently, users cannot register as in most of the universities, the system administrators maintain all the accounts.

Based on the Authentication system, the application redirects users to the proper page inside the application i.e Professors get a different page view to create quiz, view results etc., when they login and Students get a different page view to take quizzes.

Timing System

The Timing System is one of the most important classes of the System as the whole quiz depends on the time remaining. It is responsible for starting the clock once a user starts taking a quiz. It stops the quiz when the time finishes and tells its listeners i.e the other modules to end the quiz.

Logging System

The Logging System is developed to track user behavior and to improve the System. We used it for development purpose in the application and the logs mostly are used for debugging by us.

This logging module can be also used to generate log files which can be uploaded for improving the system when the application crashes.

Data Storage Module

The Data Storage Module is developed to connect storage system i.e MySQL to other modules within the QuizaApp application. This Module provides APIs for other modules to read, insert, delete data from the database. The professor module uses it for functionalities to create quiz, update quiz, view result among others. It helps student module to show list of quizzes, view quiz and take quiz among others.

Here the Data Storage Module connects to MySQL server using JDBC drivers provided by Oracle.

User Interface Module

User interface is one of the most important modules in our application as it bind both UI and backend and eases the user interaction with the application. The below image gives the UIinterface module. We will describe each class in UIInterface module as follows.

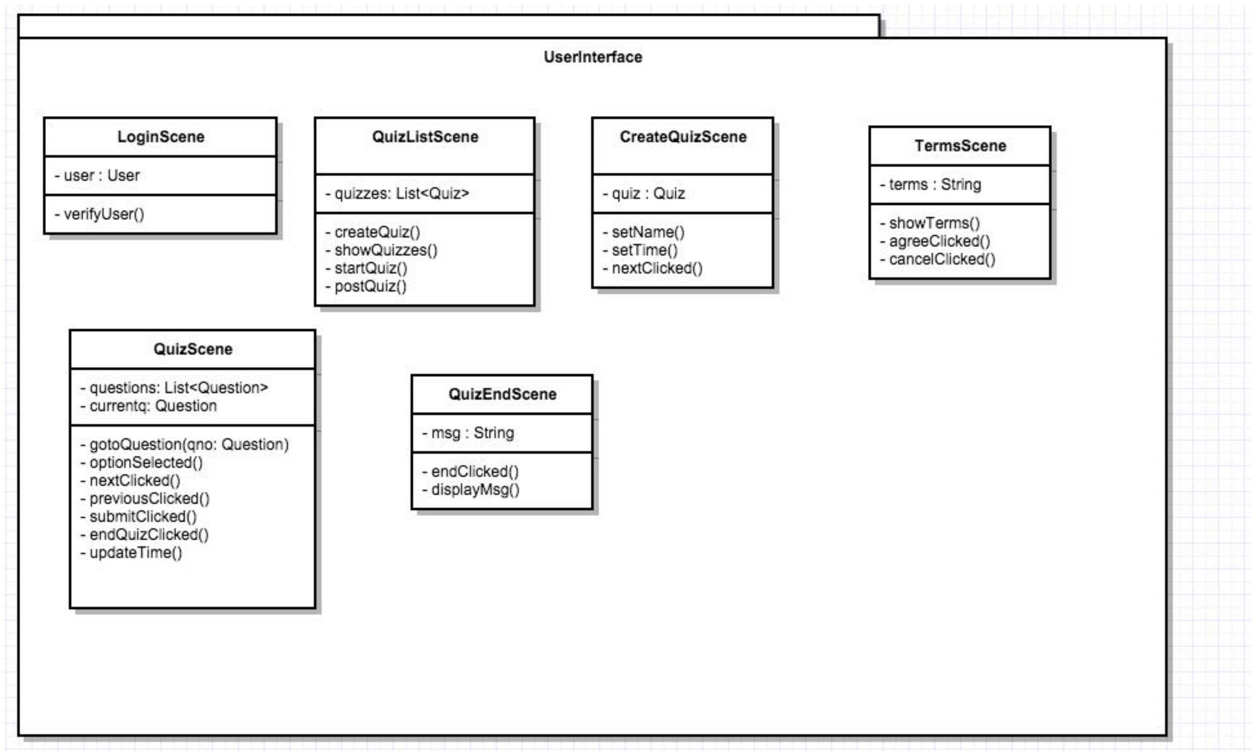

Fig 4: Architecture of UserInterface module

LoginScene Class

Login scene class is responsible for initiating the initial scene when the application is started. Login scene contains two user input fields one for username and other for password. User will enter the credentials and hit login button based on what kind of authenticated user he is we will follow the flow of scenes as needed. Like currently we have two different flow one is for student and other is for professor. Login Scene will use verifyUser() method which intern uses database to validate the user to start the flow.

QuizListScene Class

QuizListScene class scene is shown when the user otherwise student flow is active.In this flow student next scene after the loginScene is QuizListScene where student sees all the activated quiz which he can take multiple times. Only quiz’s that belong the courses he has taken are visible in this scene. Student can select either of quiz that is available to start the quiz. QuizListScene is also responsible for other scenes like create quiz, show quiz, and post quiz.

CreateQuizScene

CreateQuizScene class is responsible for the start of creating quiz, this comes in professor flow of scenes as we described before. This scene filed for quiz name, questions in quiz, time for the quiz. Class modules such as setTitle(), setTIme(), setQuestionNumber() are used to update the database when professor clicks save and continue.

TermsScene

TermsScene class is responsible for show terms and condition before taking quiz, as you expected this comes in student flow and after user reads and accepts the terms and conditions he will click on agree and continue button to proceed to actual quiz.This are achieved by using the class modules showTerms(), agreeCLick(), cancleClick().

QuizScene

QuizScene class is responsible for showing the actual where student takes quiz based on the question set by the professor. There is also time module as to show how much time is left to finish the test. All these are achieved by using modules in QuizScene class.

QuizEndScene

QuizEndScene class is responsible for showing the last scene in quiz scene after the user has submitted the quiz or when the time is done for taking the quiz. This scene will show total score of the quiz. All this are achieved by the modules that are in the quizEndScene class.

4.Tools and Technologies used

Tools and technologies used for this project are as mentioned below.

Java was the main programming that we choose to build this application as it revolves around OOAD features as discussed in this course. Many designs and ideas discussed in this course can only be implemented by JAVA that is the reason we have chosen this as our programming language.[1]

JavaFx was our next technology for building ui for quizaapp application. We choose javaFx as it was latest and new technology to learn and develop application and moreover it is build on OO concepts which helped us to implements and use those concepts in our application. JavaFx helped us build ui much faster than we expected but there were few challenges that we faced using JavaFX. Scene Builder provided by JavaFx has helped us to drag and drop the UI which made us build our UI much faster than we expected.[2]

Github was our tool for doing all the code management stuff as it was free and available. OOAD has helped us divide the modules among all the teammates and later use git for merging individual code. Here we have clean concise about what each module will do and return so that other team mate can code the module based on what is going to be returned to him, we tested our system with lot of test cases based on this method [3].

Editors played a very important role in our project. We used several editors like Intellij by IDEA and Eclipse which help us to generate lot of OOAD code like setters and getters and so on [4].

We used MySQL as our backend database as it was required for us to store the quiz and later map the use according to their courses and professors for their current taught courses. We used either mamp or wamp or xamp as our software application as they come with UI start and stop of mysql server than command line. They also help use the phpmyadmin tool where creating, modifying and inserting data into the database was much easier and error free [5].

6.Challenges

The System design was a very big challenge for us as during the design phase we had lot of designs in mind and it took time to put those on paper. After drawing the diagrams, we iterated on the design repeatedly and the system design changed from the initial design we had planned. The current design suits the needs and can be extended easily to add other capabilities to the system.

JavaFx is one of the alternatives to the historical Java Swing framework and we chose it to build the application like we described. We chose it to learn something but it proved to be a challenge. We faced lots of difficulties in building the interface like using Scene Builder which sometimes didn't save changes. Also, we couldn't fix the JavaFX bug where the screen is sometimes not painted after opening a window.

Each of our team members had involved in the development and initially we faced difficulties when we had to integrate various modules. But, we soon found a solution by using git and it solved all the integration problems.

7.References

1. https://java.com/en/download/
2. http://www.oracle.com/technetwork/java/javafx/overview/index.html
3. https://github.com/
4. https://www.jetbrains.com/idea/
5. https://www.mysql.com/