TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1
SOFTWARE REQUIREMENT SPECIFICATION

1. SOFTWARE REQUIREMENT SPECIFICATION

1.1 Introduction

The document aims at defining the overall design, software requirements and features of 'Eval' - A Group and Peer based Quizzing System *version 1.0*. Efforts have been made to define the requirement exhaustively and accurately.

1.2 Document Purpose

The purpose of this document is to present a detailed description of the **Group and Peer based Quizzing System - "Eval"**. It will explain the purpose and features of the system and what the system will do and also explain how the various modules work and how they communicate with each other for the successful working of the application.

1.3 Product Scope

The product aims to develop an integrated system for student quiz, evaluation and performance generation. It lets the teacher conduct a quiz amongst the students in a class. Students are asked to form groups among themselves. Each group has a leader who with the consent of all the group members sends a question to the teacher. Teacher validates the question and broadcasts to all the other groups. In this manner, each group gets a chance to ask question to all the other groups in Round Robin manner.

As the quiz proceeds, the valid questions and the performance of students are stored in the database for future references. Evaluation is done by the system on the basis of answers and the level of question that the students form. In addition to this, teacher also has the facility to upload files as reference material for students. Also, the language support in Hindi and English is one of the most beneficial features of the application

As a matter of fact, students study more when they are asked to form questions in comparison to when they have to appear for an exam. So, this system is to drive them to form quality questions which are also a parameter to evaluate their performance.

In order to enhance the learning practices, we believe that making this application available through Aakash tablet will improve the quality of education in schools. We also believe that if our methodology is adopted on a large scale, it will benefit the students throughout the country.

The ability of this application is to establish vision and direction in order to help students and teachers, to empower and inspire them to achieve results or success. It is all about learning, teaching and getting things done, providing freedom to individuals and ultimately allowing people to develop and helping them discover their own strengths. The purpose of creation of this project is to provide a platform through which students can learn, interact and teacher can view the performance of individual students in an easy and interactive way.

1.4 Intended Audience and Document Overview

The intended audience for this document is the development team, testing team and the end users of the product. The users are the students (class VI onwards to college students) and also teachers who want to use technology as an aid in their curriculum. In addition to English, the application supports Hindi, and is scalable to other languages as well.

The rest of this SRS contains first and foremost the introduction part, which is further subdivided into different sections which includes purpose, then scope of the product. Here the scope of the product is specified including relevant benefits, objectives, and goals. The second section, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next section. The third section, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

1.5 Definitions, Acronyms and Abbreviations

Term	Definition
● Android	Linux Based Operating System.
● Android Canvas	A drawing surface that handles compositing of the actual bits against Bitmap or Surface object.
● Software Requirements Specification	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document.
● Unified Modelling Language (UML)	Programming Language used for object oriented software development.
● UML Diagram	It is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence.

1.6 Document Conventions

It uses Times New Roman font style throughout the document for text. Font size 16 for headings, 14 for subheadings and 12 for text. Line spacing 1.5 and margin of 1”.

Chapter 2
OVERALL DESCRIPTION

2. OVERALL DESCRIPTION

2.1 Product Perspective

The Group and Peer Based Quizzing System is a new concept which has been introduced in Aakash tablet. This is the first time an application has been developed, which will help the students to interact amongst themselves on the topic taught by the teacher, by forming questions. It will also let the teachers evaluate the students easily. The product also has the provision for the teachers to upload assignments, evaluate the performance of the students on an individual basis, initiate a quiz and store the questions asked by the students during the quiz.

Students can view their individual and overall performance, download study materials uploaded by the teacher, view questions and participate in a quiz.

Initially, the product will be available in two languages namely English and Hindi. As the product evolves, more languages can be introduced so that the product benefits students of various regions.

2.2 Product Functionality

The product consists of two types of end users, teacher and students. Here teacher is the server and students are clients.

2.2.1 Students

1. The basic home page gives the list of all the modules.

    a) Performance
    b) Questions
    c) Notes
    d) Change password

- **Performance:** It allows the student to view the performance of their previous test. Also, it provides them the option to view their overall progress with respect to previously attempted quizzes in various subjects.
- **Questions:** It allows the student to view the questions of all the previous quiz sessions, categorized according to subject and date on which the quiz was held.
- **Notes:** It allows the student to download the study material provided by the teacher.
- **Change password:** It allows student to change their password.

2.2.2 Teachers

1**.** The basic home page gives the list of all the modules.

a) Register
b) Configure student details
c) Start a quiz
d) View performance
e) View questions
f) Upload a document

- **Register** module provides registration interface for the teacher who also acts as the administrator of the system.
- **Configure student details** module relates to adding, modifying and viewing student details. Once the teacher selects this choice, she will get an interface to do the following:

  i) Add a new student. By default, marks will be 0 for a new student.
  ii) Modify student details.
  iii) View details of students.

- **Start a quiz** module will provide the teacher with an interface to fill in the details of the quiz and start the process.

- **View Performance** module relates to have an interface to check out the performance of students. The interface lets the teacher enter any of the following 6 parameters:

  i) Student ID.
  ii) Name
  iii) Subject
  iv) Standard
  v) From date
  vi) To date

The teacher can fill in any of the 6 parameters and get the result depicting the performance of students.

- **View questions** module provides an interface for viewing the questions that have been validated by the teacher during the quiz sessions. Questions are fetched from a database based on the following parameters:
  i) Keyword
  ii) Subject
  iii) Standard
  iv) Level of question
  v) From date
  vi) To date

The teacher can fill in any of the 6 parameters and she will get back all the questions that match.

- **Upload a document** module provides an interface to upload reference material which is accessible to the students under notes module.

2.3 Users and Characteristics

2.3.1 Students:

- should have good knowledge of the topic on which the quiz is to be taken.
- should know how to use touch-screen devices.
- who are in one group should be close to each other for a better interaction between them.

2.3.2 Teachers:

- should inform the students about the quiz topic beforehand.
- should be able to configure student details.
- should have the capability to accept/reject questions that are sent by the student during the quiz.

2.4 Operating Environment

This product is designed to work specifically on Aakash Tablet with android (version Jelly Beans).

2.5 Design and implementation constraints

- The small size of the device screen limits the amount of content visible at a given time.
- The application has been developed in two different languages, so its effectiveness will depend upon the user's proficiency in these languages.
- It will work only on android devices.
- It is developed specifically for Aakash tablet having version 4.1 and above.

2.6 Assumptions and Dependencies

- Deadlines must be met.
- The product must be reliable.
- The architecture must be open so that additional functionality may be added later.
- The product must be user- friendly.

- From the very start of this project we are aware of time constraints so the main emphasis is on extensibility and parallel development. We shall try our best to ensure that project deadlines are met.
- Documentation must be lucid and clear so that further developments can take place, based on the already developed version.

Chapter 3
SPECIFIC REQUIREMENTS

3 SPECIFIC REQUIREMENTS

3.1 External Interface Requirements

3.1.1 User Interface

User interface must be user-friendly. The user interface shall be designed using various components such as Carousel Menu for switching among different modules, expandable list view to display the list of questions in the view questions module, jfree chart engine & android canvas to display performance statistics.

3.1.2 Hardware Interface

The hardware required are Aakash tablets for students, server for the teacher and an access point to which the server and the tablets will be connected. One access point is required for at most 50 tablets.

3.1.3 Software Interface

Client side:

Software interface for "Eval" application:

- Android 4.1 (Jelly Beans).

Server Side

- Computer with Windows or Linux (recommended) based operating system.
- Database Management System used is MySQL 5.5.

3.1.4 Communication Interfaces

- Product uses UDP and TCP protocols for communication between a client and the server.

3.1 Functional Requirements:

Major functions of the Client Software:

- Students can log in and select options from the home screen.
- When the teacher initiates the quiz, students go to the quiz page and the quiz continues.

Major functions of the Server software:

- New teachers, students can be registered and unregistered.
- Keep track of online and offline students.
- Launch and monitor quiz.

3.3 Performance Requirements:

- Any transaction between a client and the server will take approximately not more than 3 seconds.
- Any no. of users may be logged-in at the server at any time.
- All the users need to be connected to same/different access points of the same network.
- The “Eval” application should be light to minimize the power consumption of the device.

Chapter 4

NON-FUNCTIONAL REQUIREMENTS

4. NON FUNCTIONAL REQUIREMENT

4.1 Software Quality Attributes

4.1.1 Reliability:

System must be reliable and data should persist even after suffering some system crashes or booting of android supported devices.

4.1.2 Maintainability:

Software needs to be upgraded if required in future.

Chapter 5

DESIGN DOCUMENT AND IMPLEMENTATION

5. DESIGN DOCUMENT AND IMPLEMENTATION

5.1 Resource Requirements

5.1.1 H/W Requirements

For Client:
The following hardware configuration is required for this project:

- Minimum RAM Required : 512 MB
- Minimum Capacity Required :
  - Internal : 2 GB
  - External : 2 to 32 GB
- Display Resolution : 800 x 480 Pixels
- Touch screen
- Wi-Fi connectivity

For Server:

- Minimum RAM Required : 1 GB
- Minimum HDD Storage Required : 100 GB

5.1.2 S/W Requirements

Following software are required for this project:

- Eclipse Juno, ADT Plugin along with Android SDK.
- Microsoft Windows or Ubuntu operating system.
- We have used MySQL as the database to store the quiz questions, answers and the performance statistics of the students in the server.

5.2 Software Development Life Cycle Model

The systems development life cycle (SDLC), or software development life cycle in systems engineering, information systems and software engineering, is a process of creating or altering information systems, and the models and methodologies that people use to develop these systems.
In software engineering the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system.

A software development process is a structure imposed on the development of a software product. Similar terms include software life cycle and software process. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process. Some people consider life cycle model a general term and software life cycle development a specific term.

Iterative and Incremental development is at the heart of a cyclic software development process developed in response to the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between. Incremental development slices the system functionality into increments (portions). In each increment, a slice of functionality is delivered through cross-discipline work, from the requirements to the deployment. The unified process groups increments/iterations into phases: inception, elaboration, construction, and transition

- Inception identifies project scope, risks, and requirements (functional and nonfunctional) at a high level but in enough detail that work can be estimated.
- Elaboration delivers a working architecture that mitigates the top risks and fulfills the non-functional requirements.
- Construction incrementally fills-in the architecture with production-ready code produced from analysis, design, implementation, and testing of the functional requirements.
- Transition delivers the system into the production operating environment.

Each of the phases may be divided into 1 or more iterations, which are usually time-boxed rather than feature-boxed. Architects and analysts work an additional iteration ahead of developers and testers to keep their work-product backlog full.

The system has been developed using **Iterative Waterfall Model.**

The **waterfall model** is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation and Maintenance.

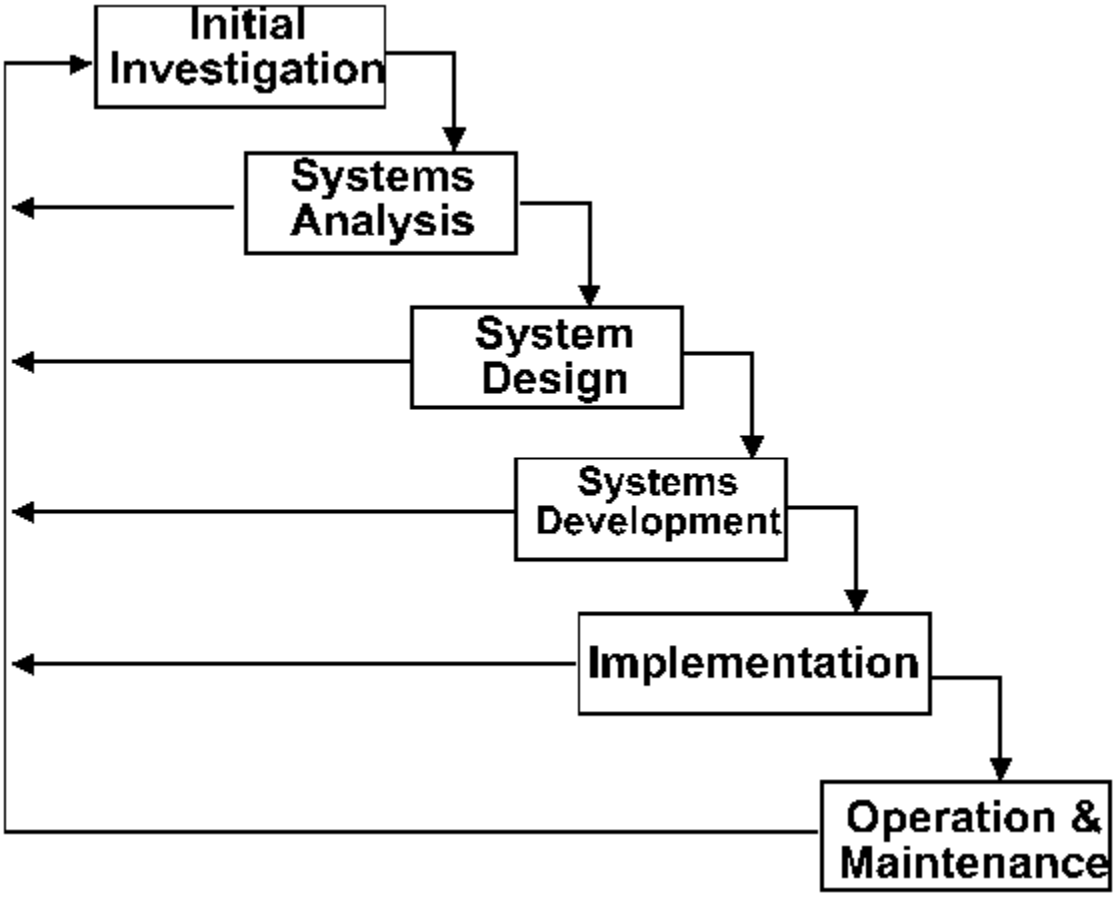


Figure 5.0: SDLC – Iterative Waterfall Model

Advantages

- Simple goal. Simple to understand and use.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.
- Easy to manage. Each phase has specific deliverable and a review.
- Works well for projects where requirements are well understood.
- Works well when quality is more important than cost/schedule.
- Customers/End users already know about it.

Disadvantages

- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.

- No working software is produced until late in the life cycle.
- Risk and uncertainty is high with this process model.
- Adjusting scope during the life cycle can end a project
- Not suitable for complex projects
- Not suitable for projects of long duration because in long running projects requirements are likely to change.
- Integration is done as a "big-bang" at the very end, which doesn't allow to identify any technological or business bottleneck or challenges early.
- Attempt to go back 2 or more phases is very costly.
- Percentage completion of functionality could not be determined in mid of the project because each functionality is undergoing some phase.
- Very risky, since one process cannot start before finishing the other.

5.3 High Level Design Document

Use Case Diagram

A use case diagram presents a graphical overview of the functionality provided by a system in terms of actors, their goals (use cases), and any dependencies between those use cases.

Client Side:

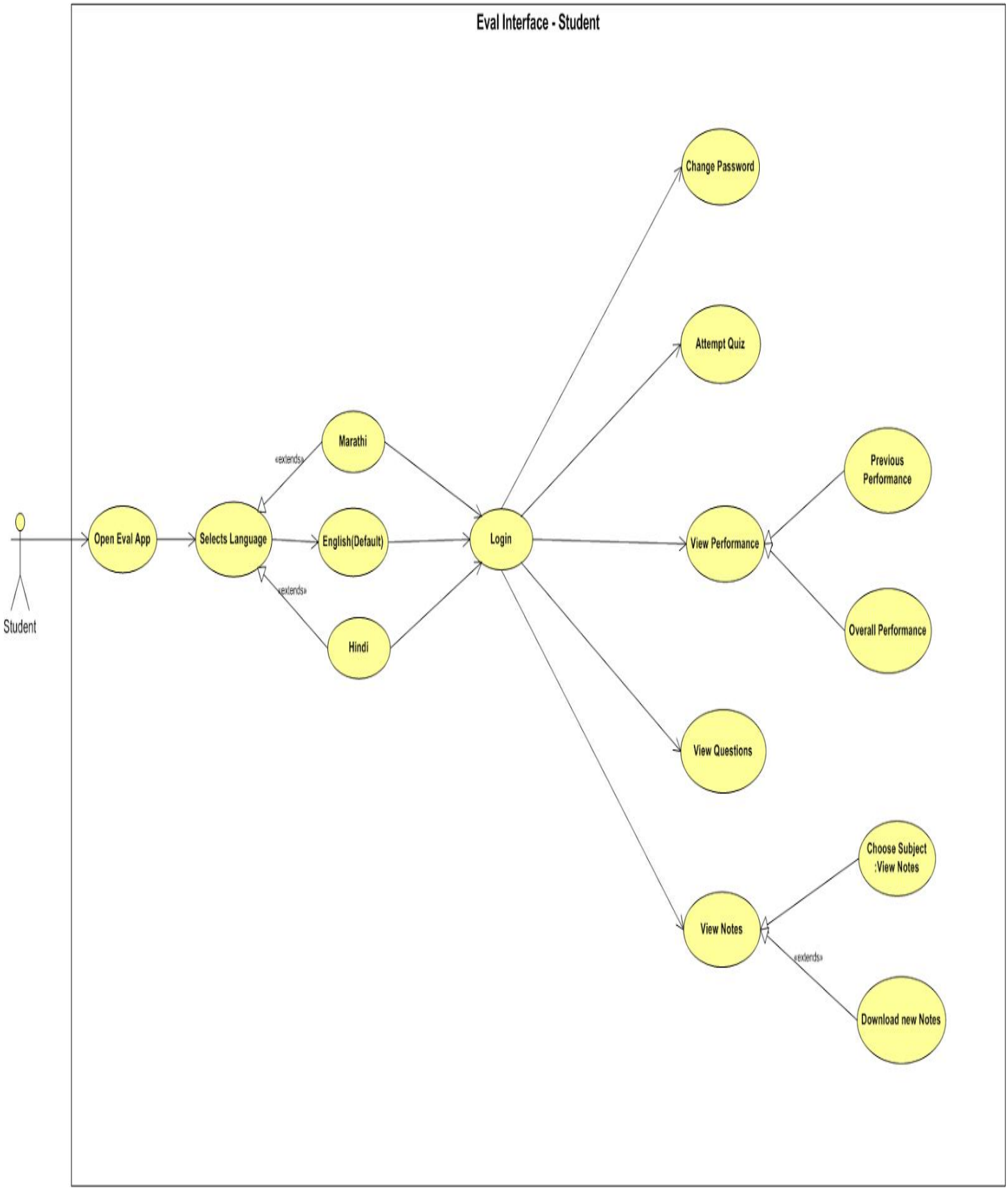


Figure 5.1: Eval Interface - Student

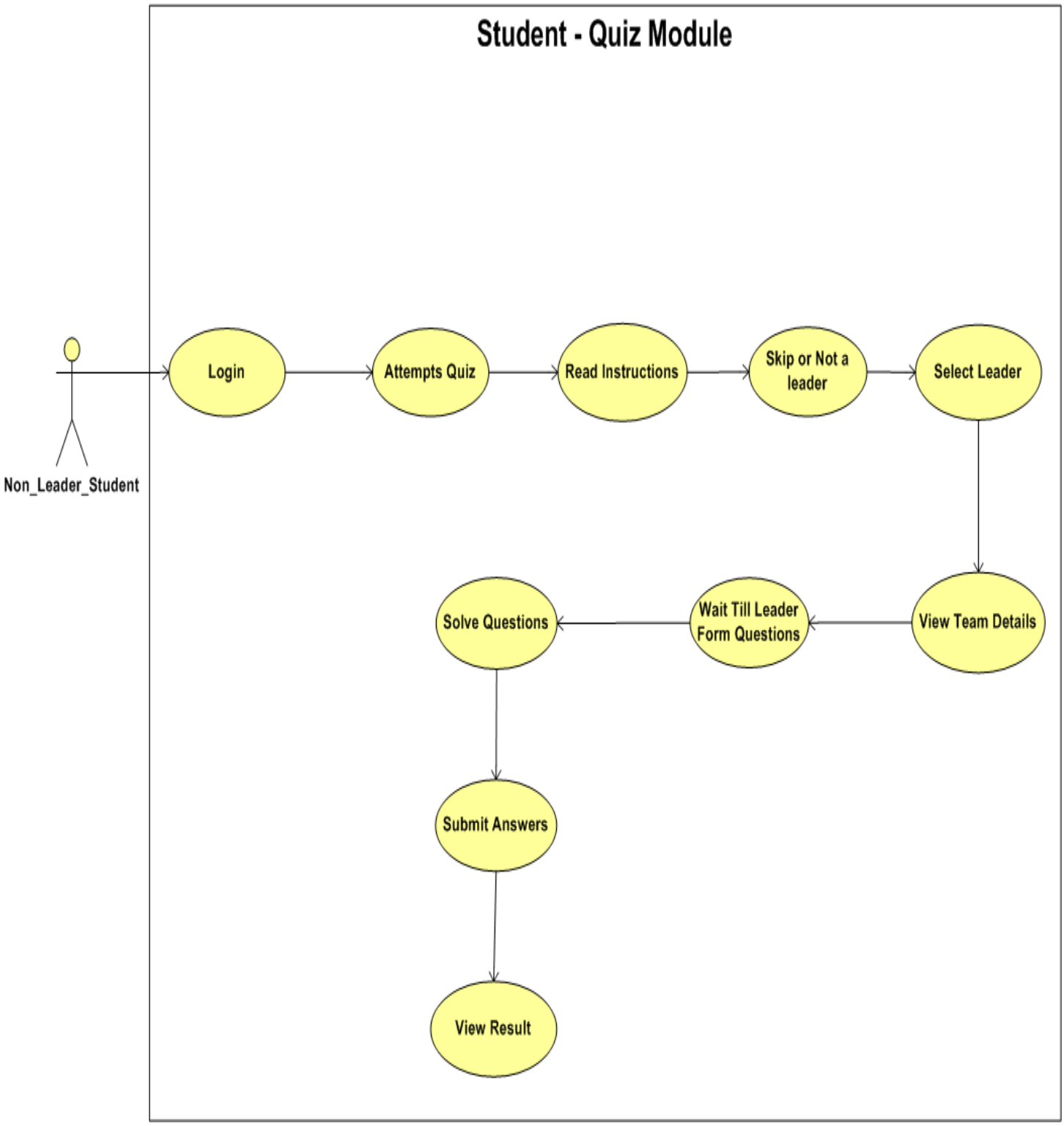


Figure 5.2: Student - Quiz Module

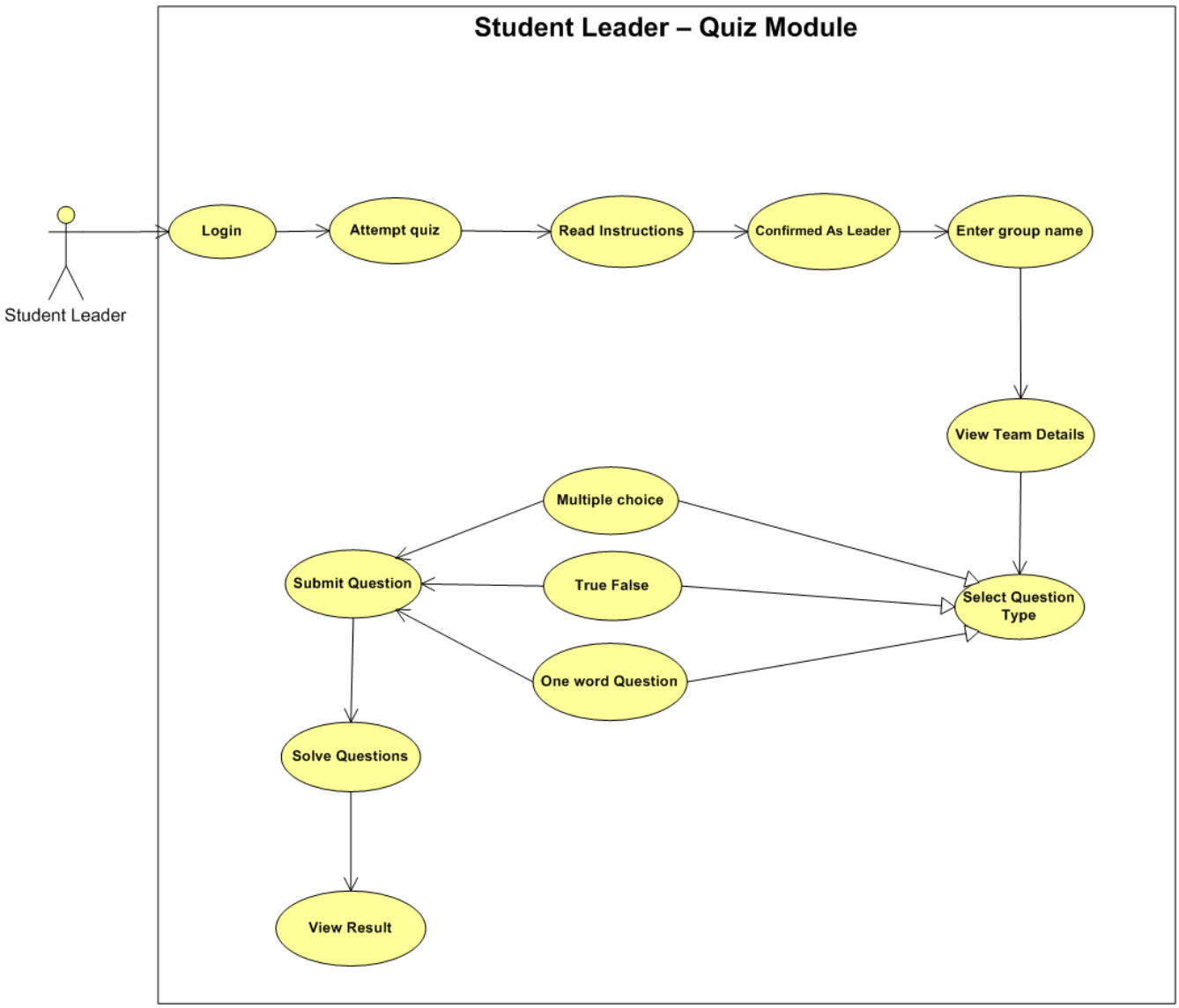


Figure 5.3: Student Leader Quiz Module

Server Side:

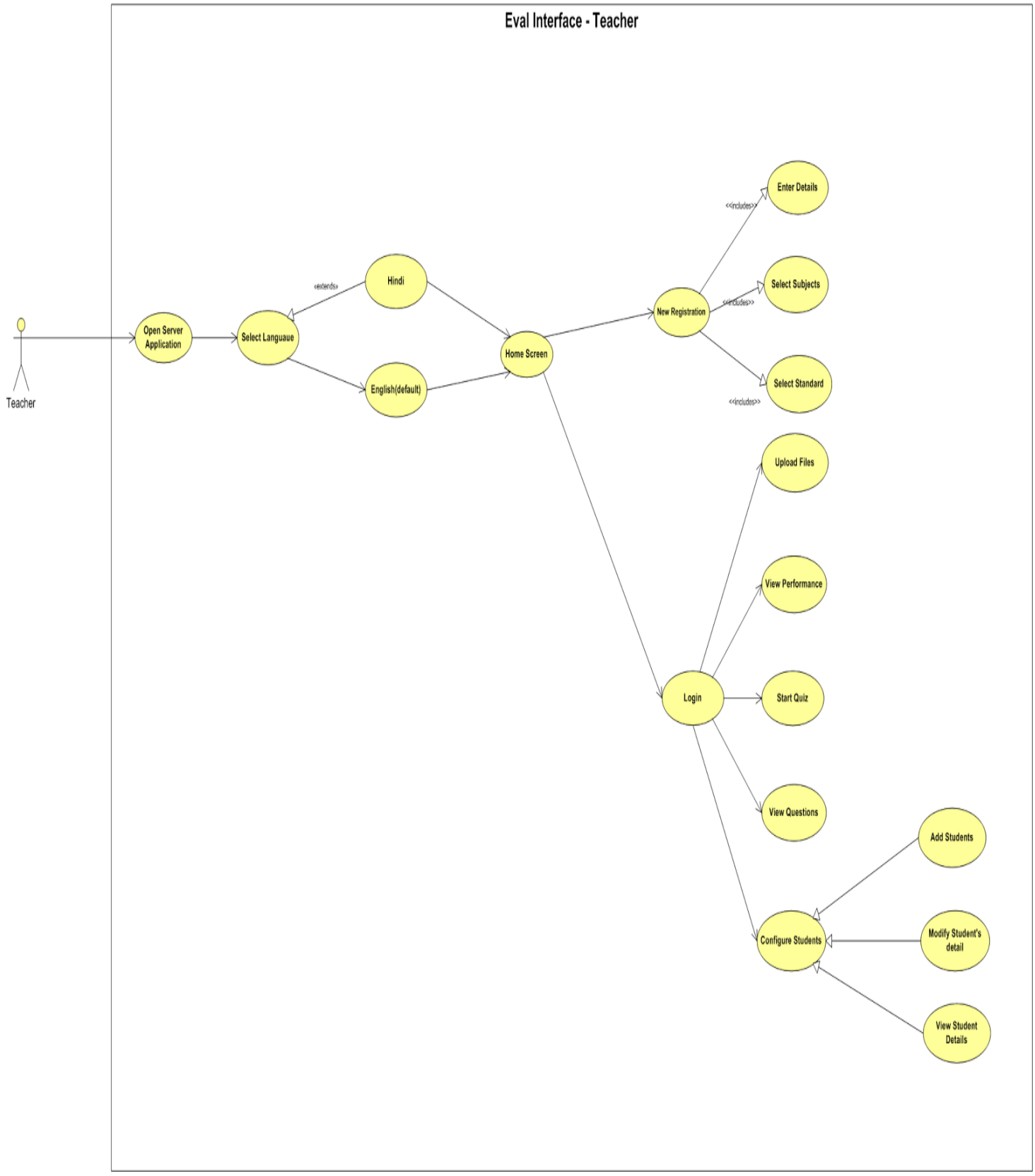


Figure 5.4: Eval Interface - Teacher

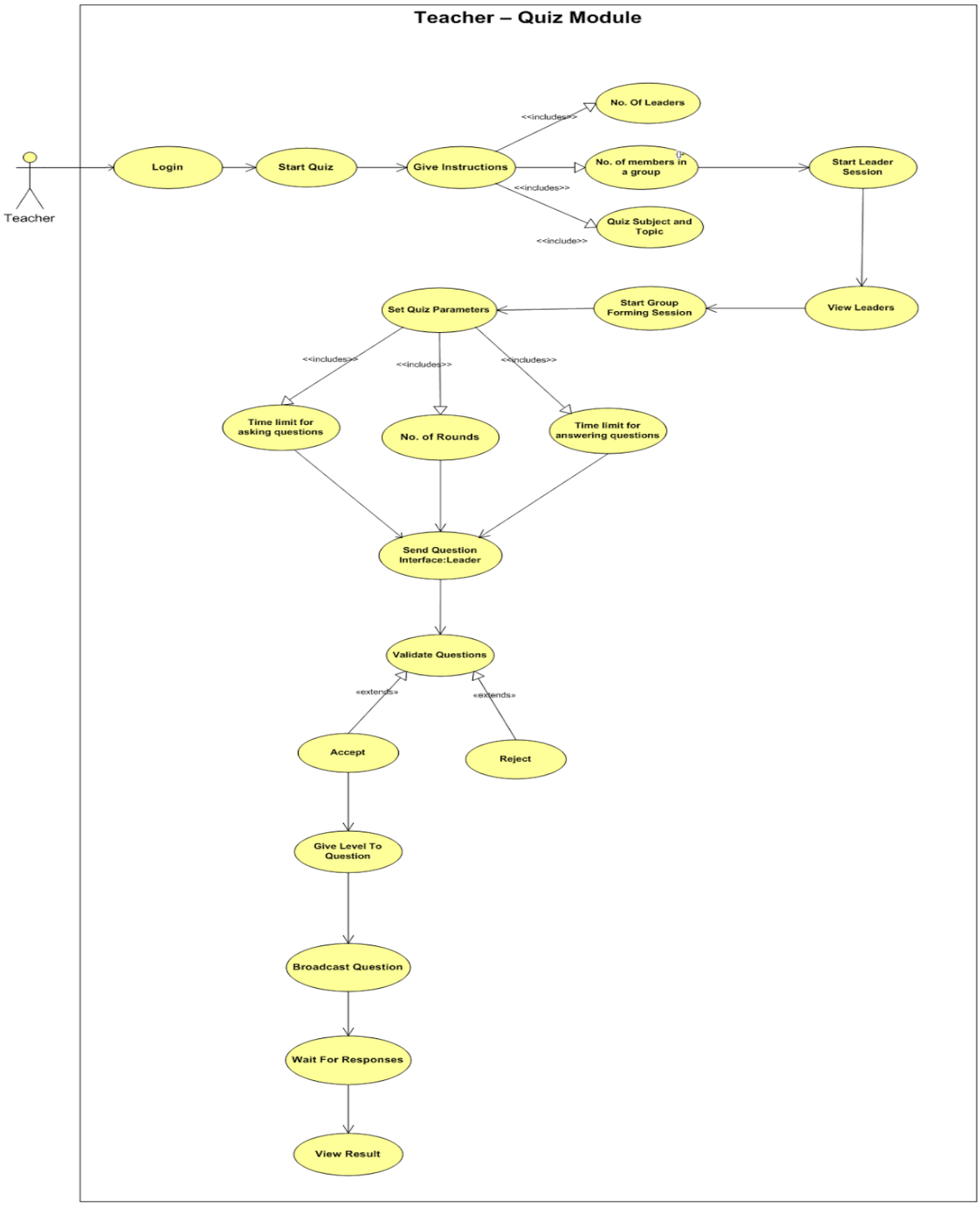


Figure 5.5: Teacher - Quiz Module

5.3.2 Use Case Descriptions:

Students and Teachers are the main actors of the system. The user launches the application and selects the appropriate language understood as per the school's medium. He then logs in and select the appropriate module as required. Then, he/she can participate in quiz and evaluate himself/herself.
Teacher when launches the application has to select the language and can have access to the application wherein he/she can select the appropriate modules required. Teacher can also monitor the student's performance.

5.3.2.1 Select Language:

Brief Description: The default language used is English. But if the user is not familiar with this language, then this option enables the user to select the language of his own choice from either Marathi or Hindi.

Flow of Events:

- **Basic Flow: :** User enters the application where he/she selects the preferred language.
- **Alternate Flow:** User can exit the app by pressing back button.

Pre-Condition: User opens Eval application.
Post-Condition: Login page is displayed.

5.3.2.2 Login:

Brief Description: Login function enables the user to enter into the main modules of the application.

Flow of Events:

- **Basic Flow:** User enters application where he/she has four options (Notes, Change Password, Performance and Questions).
- **Alternate Flow:** User can view "Application Guide" or "About Us".

Pre-Condition: User opens Eval application.
Post-Condition: Different modules are displayed.

5.3.2.3 About Us:

Brief Description: About us function displays the application development team details.

Flow of Events:

- **Basic Flow:** User tap on About us to view the respective person's detail.
- **Alternate Flow:** User can view "Application Guide" or enter into application "Eval" by logging in.

Pre-condition: User opens Eval application.
Post-condition: User views About us page.

5.3.2.4 Application Guide:

Brief Description: This function displays instructions for efficient use of the application.

Flow of Events:

- **Basic Flow:** User tap on Application Guide & read the instructions.
- **Alternate Flow:** User can view "About Us" or enter into application by logging in .

Pre-condition: User opens Eval application.
Post-condition: User views the instructions.

5.3.2.5 Change Password:

Brief Description: This enables the user to change his/her password where he/she has to provide his/her respective id to successfully change the password.

Flow of Events:

- **Basic Flow:** User enters change password where he has to give his id, current password, new password and confirm his new password and then presses the submit button
- **Alternate Flow:** User can select logout, Performance, Notes, Questions or may be in the quiz page .

Pre-condition: User is logged in and has access to the main menu screen.
Post-condition: Password is changed successfully and back to the main screen.

5.3.2.6 Performance:

Brief Description: This enables the user to keep a track of his performance. This module is further divided in two more modules, previous performance and overall performance. In the first module he can check his last performance in the quiz subject wise and in the second module he can view the overall progress of his performance till date of various subjects.

Flow of Events:

- **Basic Flow:** User enters Performance where he has two choices: check previous performance or check overall performance.
- **Alternate Flow:** User can select logout, change password, notes, and questions or may be in the quiz page.

Pre-condition: User is logged in and has access to the main menu screen.
Post-condition: User has two options, select previous performance or selects overall performance.

5.3.2.7 Previous Performance:

Brief Description: This enables the user to select a particular subject and check his last performance in that subject. He comes to know about the number of questions he answered correctly and how many he answered incorrectly. He also gets the score of his quiz.

Flow of Events:

- **Basic Flow:** User selects the subject and views a pie chart.
- **Alternate Flow:** User can go to the home screen or can take a screenshot of his pie chart shown.

Pre-condition: User must enter the performance module and should have clicked check previous performance.
Post-condition: User can select subject and can view his performance in form of a pie chart.

5.3.2.8 Overall Performance:

Brief Description: This enables the user to have a look at his progress and performance in different subjects test wise.

Flow of Events:

- **Basic Flow:** User views a progress graph where he can view all his subject's marks.
- **Alternate Flow:** User can go to the home screen or can take a screenshot of his progress graph shown.

Pre-condition: User must enter the performance module and should have clicked check overall performance.
Post-condition: User can see a graph of his performance for all the subjects.

5.3.2.9 Questions:

Brief Description: By selecting this user can have a look at all the questions subject wise that were asked in the previous quiz. He can also view the answer, level and date of the question asked. These questions are stored in a file locally in the application so whenever user logs in, he can view the questions easily.

Flow of Events:

- **Basic Flow:** User has to select the particular subject to view the questions of that subject. He can press on refresh button to get the new questions from the database.
- **Alternate Flow:** User can go to the home screen.

Pre-condition: User must enter the Question module
Post-condition: User has to select the subject and he can view the questions asked in the previous quizzes.

5.3.2.10 Notes:

Brief Description: By selecting this module user can have a look at all the files downloaded by him from the server subject wise. He can also download new files uploaded by the teacher by clicking on the refresh button. The notes downloaded are being stored in the sd card of the tablet.

Flow of Events:

Basic Flow: User has to select the particular subject to view the notes of that subject. He can also press the refresh button to get the new files which are uploaded by the teacher.
Alternate Flow: User can go to the home screen by clicking the home button.

Pre-condition: User must enter the Notes module
Post-condition: User has to select the subject and he can view the files or can download new files.

5.3.2.11 Attempts Quiz:

Brief Description: This enables the students to give quiz; this page is broadcasted by the teacher to all the students. The page consists of general instructions like how many leaders will be there and what will be the maximum group size along with the name of the topic and subject. Here students can opt for leader or press the skip button if they don't want to become a leader.

Flow of Events:

- **Basic Flow:** User enters Quiz where he has to select the button leader or press the skip button and has to wait until the leader session expires.
- **Alternate Flow:** There is no alternate option.

Pre-condition: User is logged in and has access to the main menu screen.
Post-condition: User can be selected as a leader after clicking the leader button.

5.3.2.12 Enter Group Name:

Brief Description: Students who are selected as leader give a group name to their respective groups while other students are selecting their leaders.

Flow of Events:

- **Basic Flow:** User enters group name.
- **Alternate Flow:** There is no alternative option.

Pre-condition: User is confirmed that he is a leader.
Post-condition: User submits the group name and can view his group.

5.3.2.13 Select Leader:

Brief Description: Students who are not selected as leader or who pressed skip button selects their leader by clicking on their name. If the group is full, they will have to opt for a different leader.

Flow of Events:

- **Basic Flow:** User selects their team leader.
- **Alternate Flow:** There is no alternative option.

Pre-condition: User is not a leader.
Post-condition: User selects the leader and can view his Team.

5.3.2.14 Team Details:

Brief Description: Here all the students are broadcasted to team details page which displays the team members' name.

Flow of Events:

- **Basic Flow:** Users view their leader name and all the members who are present in their team.
- **Alternate Flow:** There is no alternative option.

Pre-condition: The group formation session should expire.
Post-condition: Students now wait for the quiz to start.

5.3.2.15 Select Question Type:

Brief Description: The group which has been selected for asking question to other groups, their respective leader will receive this page, where he has to select the question type like multiple choice question, one word question or true false question type. He has to type the question along with the correct answers and submit the question to the teacher. Teacher validates the question when she receives it.

Flow of Events:

- **Basic Flow:** Leader discuss the question and question type among his team members and then type the question and send it to the server**.**
- **Alternate Flow:** There is no alternative option.

Pre-condition: Quiz session has started.
Post-condition: Students of that group wait for the teacher to validate the question and also wait for other groups to answer their question.

5.3.2.16 Solves Question:

Brief Description: All the groups will receive the question except the group which has formed the question. According to the level provided to that question by the teacher, students of the question forming group will be marked and the students of other groups will be marked according to their performance. Answers are sent to the server and checked.

Flow of Events:

- **Basic Flow:** Students answer their questions individually and their marks are recorded .
- **Alternate Flow:** There is no alternative option.

Pre-condition: Quiz session should start.
Post-condition: If the maximum number of rounds are not completed, then the other group will get a chance to ask question else a pie chart is displayed which show their current performance.

5.3.2.17 View Results:

Brief Description: When the quiz is over i.e. maximum no. of rounds have been completed then students will get a pie chart of their previous performance. They can view total questions attempted, wrong and correctly answered and also their score.

Flow of Events:

- **Basic Flow:** Students can view their marks in form of a pie chart and their performance for that particular subject is recorded in the database.
- **Alternate Flow:** There is no alternative option.

Pre-condition: Quiz is completed.
Post-condition: After viewing the pie chart, students will be directed to the home screen.

Server Side:

5.3.2.18 Selects Language:

Brief Description: This enables teacher to select the language. They can opt for either Hindi or English. English is taken as a default language.

Flow of Events:

- **Basic Flow:** User enters application where he/she selects the preferred language.
- **Alternate Flow:** User can exit the app by closing it.

Pre-condition: User opens ‘Eval’ application.
Post-condition: Home page is displayed.

5.3.2.19 Login:

Brief Description: Login function enables teacher to enter into the main modules of the application.

Flow of Events:

- **Basic Flow:** User enters application by entering the username and password. After logging in she can view different modules present.
- **Alternate Flow:** If the user is new, she can go for new registration.

Pre-condition: User opens 'Eval' application.
Post-condition: Different modules are displayed.

5.3.2.20 New Registration:

Brief Description: If teacher is new she can register herself by giving her details and select subject and standards she is going to teach.

Flow of Events:

- **Basic Flow:** User can register herself and then login to have access to different modules.
- **Alternate Flow:** User can also go for the login option.

Pre-condition: User opens Eval application.
Post-condition: Registration done successfully.

5.3.2.21 Performance:

Brief Description: This module relates to give an interface to the teacher to check out the performance of students by their roll no, date, subject, view all records of a particular student or view all the performance of one particular subject ordered by date.

Flow of Events:

- **Basic Flow:** Teacher enters Performance where she gives the parameters and accordingly the records are filtered out from the database.
- **Alternate Flow:** User can also select logout, Configure Students, Upload notes, View Questions or Start quiz.

Pre-condition: Teacher is logged in and has access to the main menu screen.
Post-condition: Teacher can view the performance in tabular form or a graphical form(i.e. in the form of a bar graph).

5.3.2.22 Questions:

Brief Description: This module relates to give an interface to the teacher to check out the all the questions that were asked in the previous quizzes ordered by date, subject, level and standard.

Flow of Events:

- **Basic Flow:** Teacher enter Question module where she gives the parameters and accordingly the questions are filtered out from the database.
- **Alternate Flow:** User can also select logout, Configure Students, Upload notes, View Performance or Start quiz.

Pre-condition: Teacher is logged in and has access to the main menu screen.
Post-condition: Teacher can view the Questions in tabular form.

5.3.2.23 Configure Students:

Brief Description: This module provides teacher with an interface to add a new student, modify details of a new student or view students' details. Teacher can modify the student's password by giving his id. She can view a student detail like his overall performance till date.

Flow of Events:

- **Basic Flow:** Teacher can add student, modify student details or view student's details.
- **Alternate Flow:** User can also select logout, View Questions, Upload notes, View Performance or Start quiz.

Pre-condition: Teacher is logged in and has access to the home screen.
Post-condition: Teacher can configure students.

5.3.2.24 Upload Notes:

Brief Description: This module provides teacher with an interface to upload assignments or give notes to the students.

Flow of Events:

- **Basic Flow:** Teacher can upload files for students.
- **Alternate Flow:** User can also select logout, View Questions, Configure Students, View Performance or Start quiz.

Pre-condition: Teacher is logged in and has access to the main menu screen.
Post-condition: Teacher can upload files.

5.3.2.25 Start Quiz:

Brief Description: This module provides teacher with an interface to conduct quiz in the class. She can start it whenever she wants. Students who are logged in are eligible to take part in the quiz.

Flow of Events:

- **Basic Flow:** Teacher initiates the quiz by clicking the Start quiz button. All the students are broadcasted the quiz page.

- **Alternate Flow:** User can also select logout, View Questions, Configure Students, View Performance or Upload Files.

Pre-condition: Teacher is logged in and has access to the main menu screen.
Post-condition: Teacher gets an interface to give instructions for the quiz.

5.3.2.26 Give Instructions:

Brief Description: Teacher gives general instructions before starting the actual test. She provides information regarding no. of leaders, no. of members in a group and subject of the quiz.
This information is used to form team leaders and also to form group. Teacher also has to give the time for leader session and group forming session.
If leader session or group formation session expires teacher can repeat the process again or simply click on the continue button to proceed further.

Flow of Events:

- **Basic Flow:** Teacher gives general instructions to all the students.
- **Alternate Flow:** Teacher can only have access to the quiz page.

Pre-condition: Teacher presses Start Quiz button and gives time limit for leader session.
Post-condition: Teacher gets an interface where she can view all the leaders and their respective group members.

5.3.2.27 Set Quiz Parameters:

Brief Description: Teacher gets an interface to configure quiz where she has to enter parameters like maximum no. of rounds to be conducted, time limit for question and answers.

Flow of Events:

- **Basic Flow:** Teacher sets all the quiz parameters.
- **Alternate Flow:** No alternative choices provided.

Pre-condition: Teacher should make sure that all the groups are formed..
Post-condition: Teacher gets an interface to receive question from leader.

5.3.2.28 Validate Questions:

Brief Description: Teacher receives the questions given by the leader and checks it whether it is a valid question or not. If it is a valid question she can proceed with that or she has a option to reject the question. When the question is rejected leader has to form the question again.

Flow of Events:

- **Basic Flow:** Teacher can validate a question by accepting or rejecting it.
- **Alternate Flow:** No alternative choices provided.

Pre-condition: Question should reach within the specified time limit.
Post-condition: Teacher gives level to the question.

5.3.2.29 Enter Level:

Brief Description: After validating the question teacher can provide level to the question and accordingly the marks get awarded to the group members whose team leader has formed the question.

Flow of Events:

- **Basic Flow:** Teacher gives level to the question.
- **Alternate Flow:** No alternative choices provided.

Pre-condition: Question should be accepted by the teacher.
Post-condition: Teacher broadcast question to all other students.

5.3.2.30 Wait For Responses:

Brief Description: In this interface teacher wait for the responses from other students. After all the students have given the answers, teacher can have a graphical view of performance for that particular question in form of a pie chart.

Flow of Events:

- **Basic Flow:** Teacher wait until answering session expires.
- **Alternate Flow:** No alternative choices provided.

Pre-condition: Answers should reach within the specified time limit.
Post-condition: Teacher can view a pie chart of the performance.

5.3.2.31 View Results:

Brief Description: When the quiz is over i.e. maximum no. of rounds have been completed, then the teacher can view the average performance of the class in form of a graphical representation.

Flow of Events:

- **Basic Flow:** Teacher can view the performance of the class, and their average performance.
- **Alternate Flow:** There is no alternative option.

Pre-condition: Quiz is completed.
Post-condition: After viewing the graphical chart, teacher will be directed to the home screen.

Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

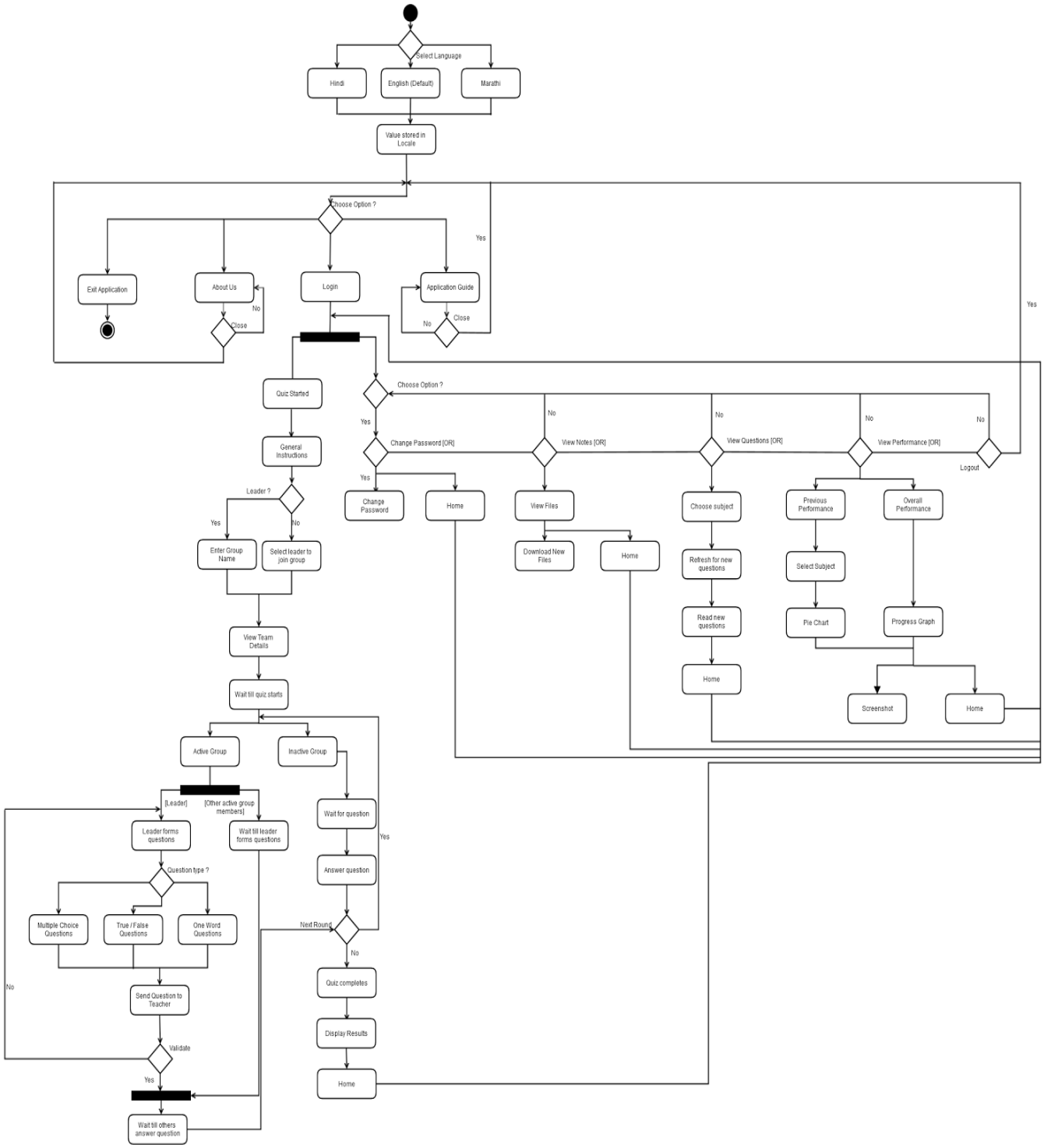


Figure 5.6: Activity Diagram of student

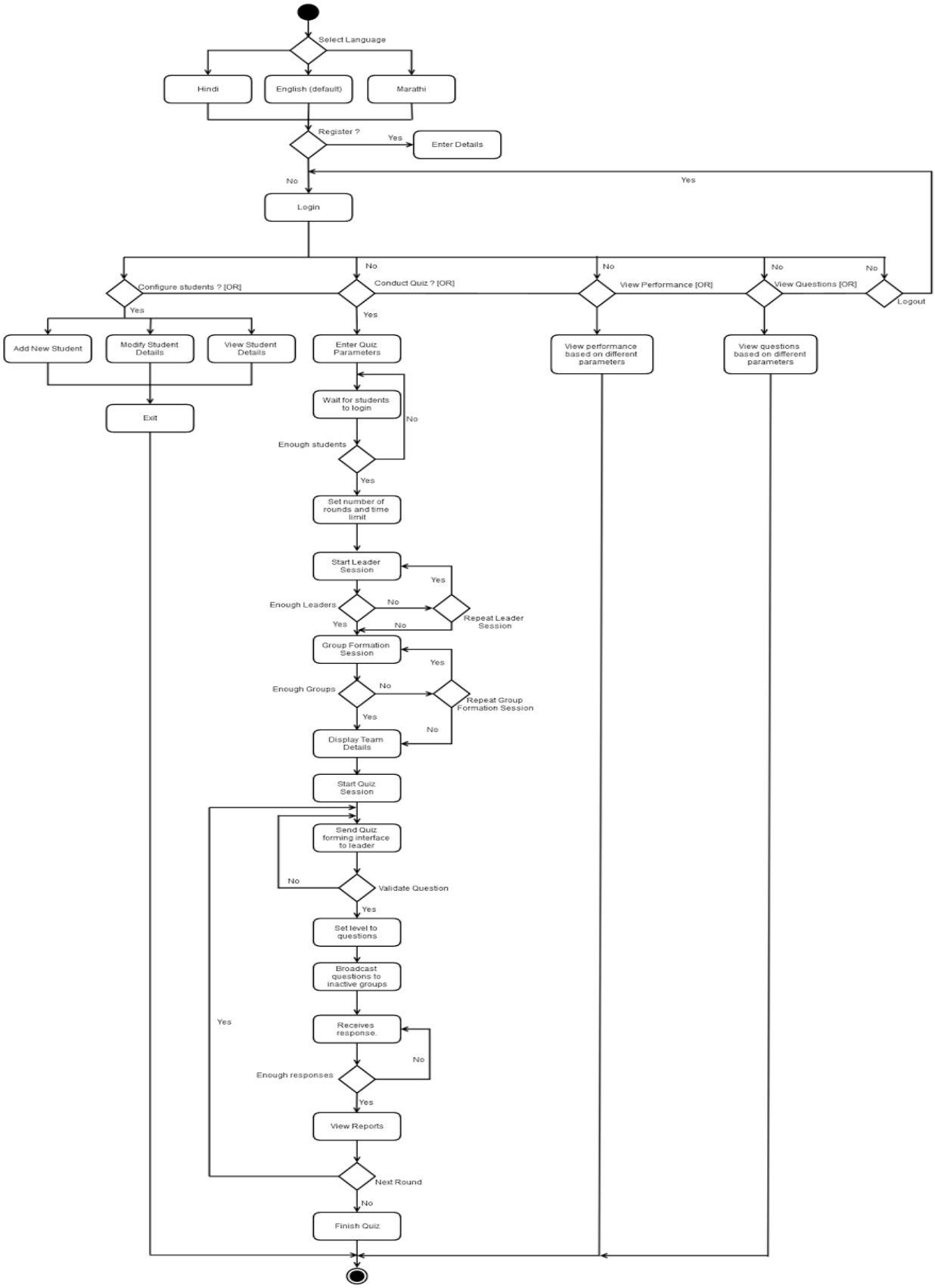


Figure 5.7: Activity Diagram of Teacher

Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or) methods and the relationships between the classes.

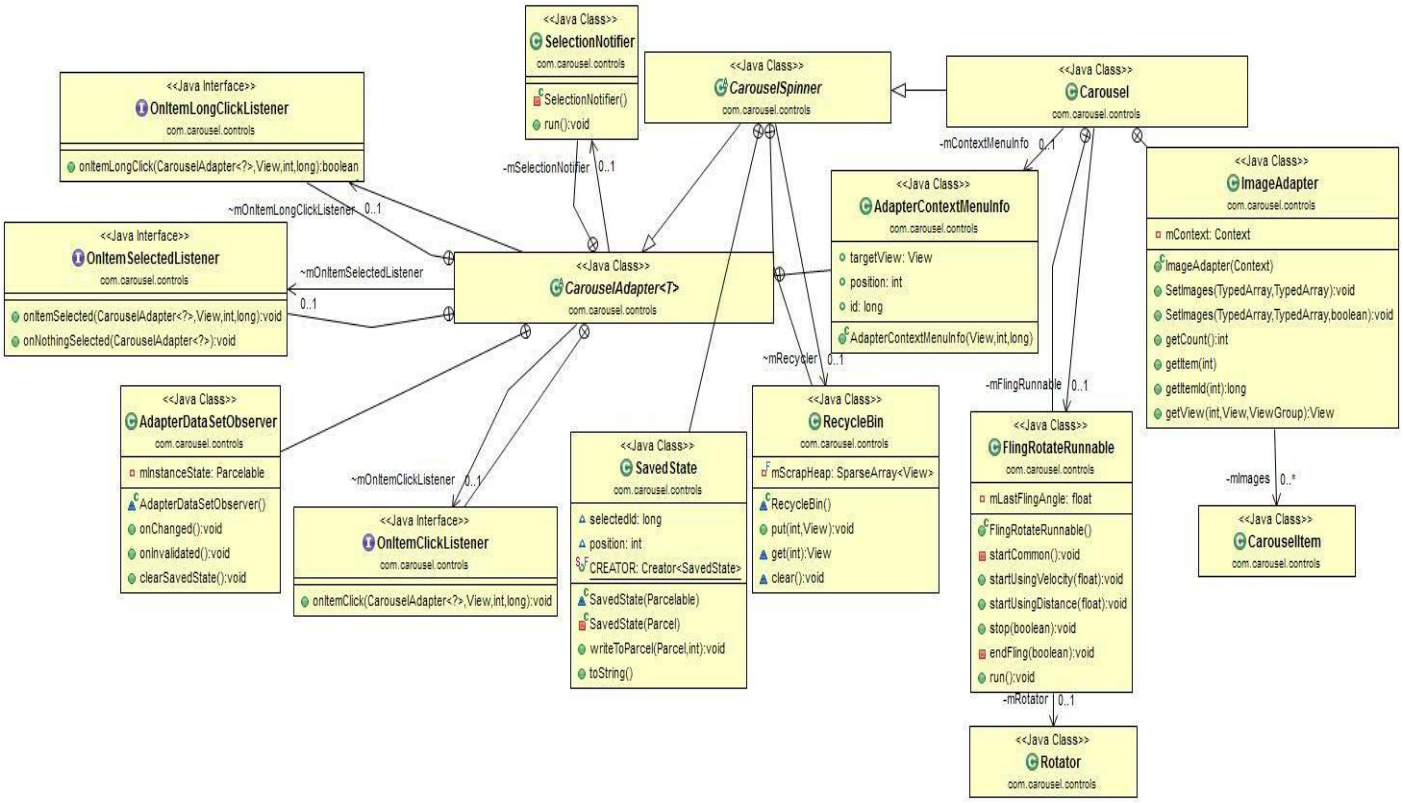


Figure 5.8: Class Diagram for 'Eval'

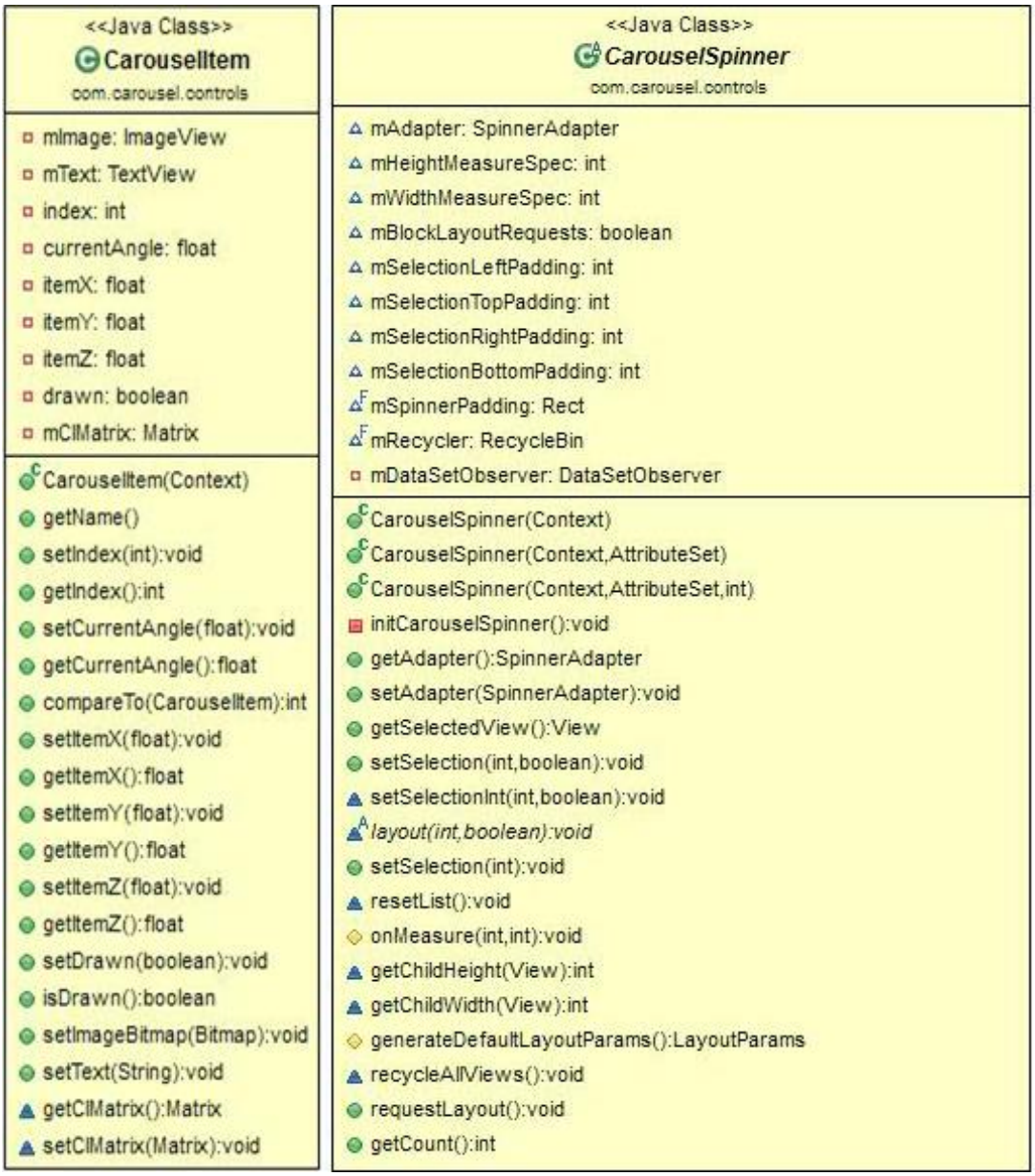
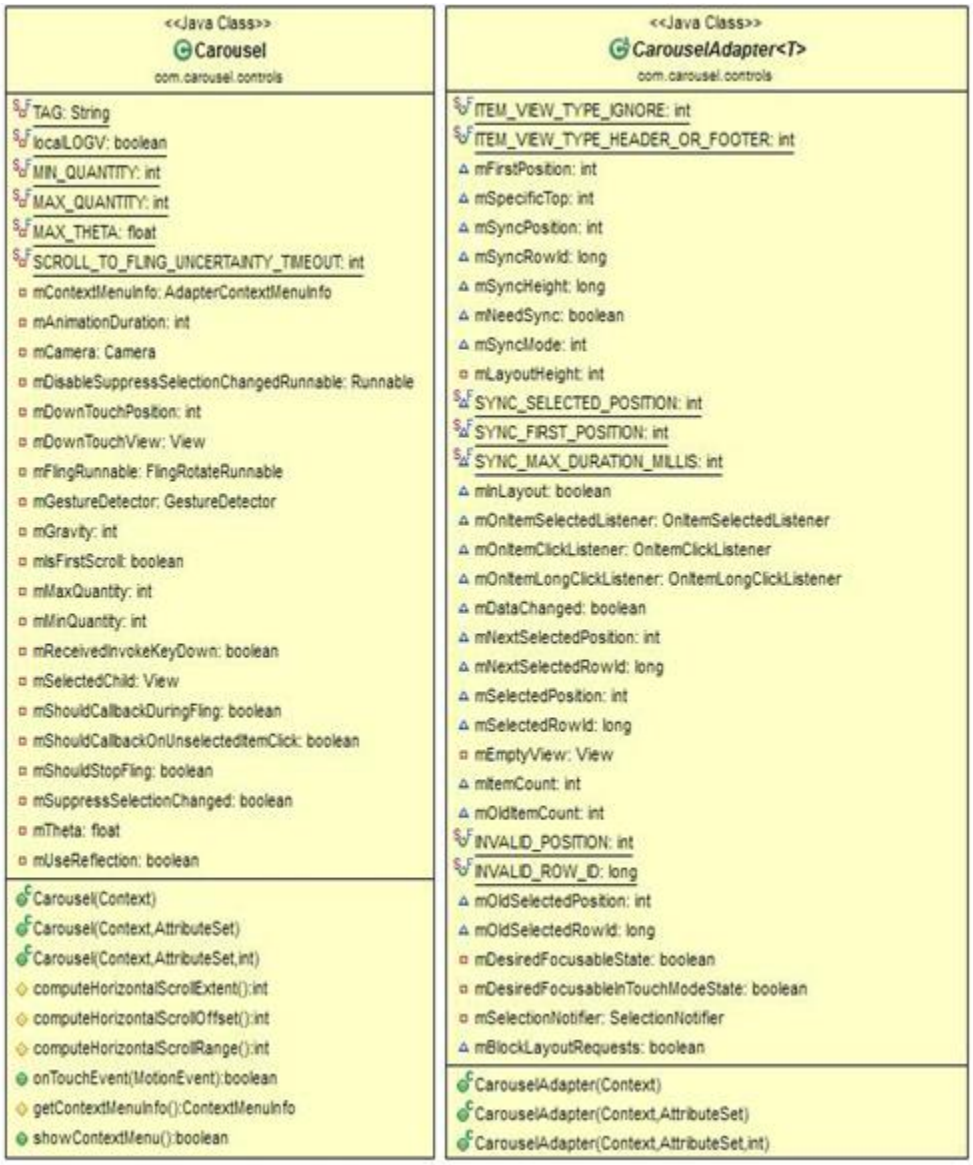
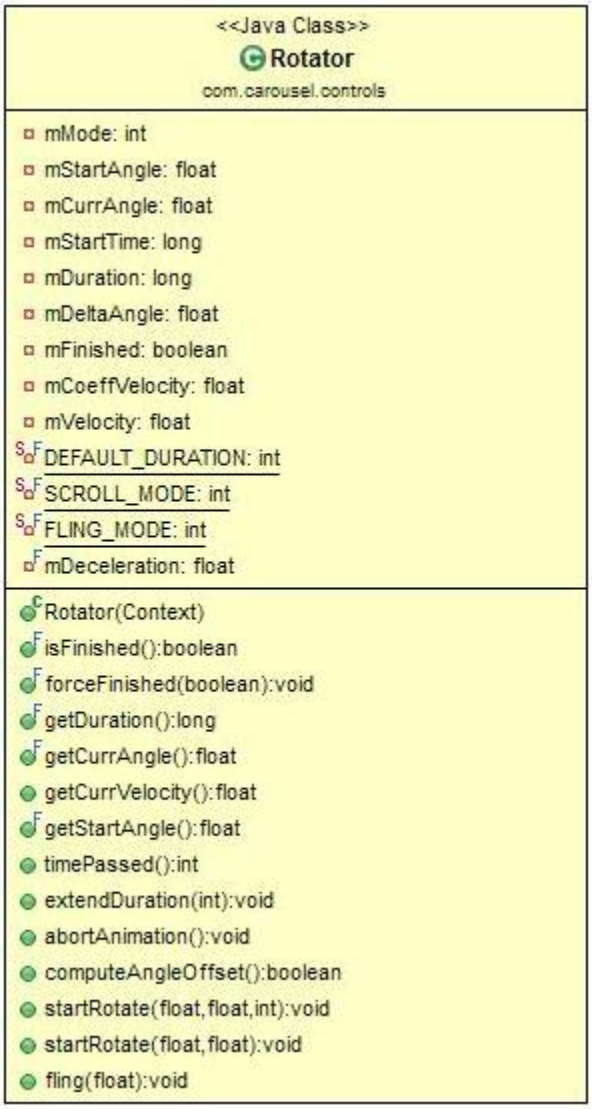


Figure5.9: Class diagram for 'Eval'


<<Java Class>>
Carousel
com.carousel.controls
TAG: String
localLOGV: boolean
MIN_QUANTITY: int
MAX_QUANTITY: int
MAX_THETA: float
SCROLL_TO_FLING_UNCERTAINTY_TIMEOUT: int
mContextMenuInfo: AdapterContextMenuInfo
mAnimationDuration: int
mCamera: Camera
mDisableSuppressSelectionChangedRunnable: Runnable
mDownTouchPosition: int
mDownTouchView: View
mFlingRunnable: FlingRotateRunnable
mGestureDetector: GestureDetector
mGravity: int
mIsFirstScroll: boolean
mMaxQuantity: int
mMinQuantity: int
mReceivedInvokeKeyDown: boolean
mSelectedChild: View
mShouldCallbackDuringFling: boolean
mShouldCallbackOnUnselectedItemClick: boolean
mShouldStopFling: boolean
mSuppressSelectionChanged: boolean
mTheta: float
mUseReflection: boolean
Carousel(Context)
Carousel(Context,AttributeSet)
Carousel(Context,AttributeSet,int)
computeHorizontalScrollExtent():int
computeHorizontalScrollOffset():int
computeHorizontalScrollRange():int
onTouchEvent(MotionEvent):boolean
getContextMenuInfo():ContextMenuInfo
showContextMenu():boolean
<<Java Class>>
CarouselAdapter<T>
com.carousel.controls
ITEM_VIEW_TYPE_IGNORE: int
ITEM_VIEW_TYPE_HEADER_OR_FOOTER: int
mFirstPosition: int
mSpecificTop: int
mSyncPosition: int
mSyncRowId: long
mSyncHeight: long
mNeedSync: boolean
mSyncMode: int
mLayoutHeight: int
SYNC_SELECTED_POSITION: int
SYNC_FIRST_POSITION: int
SYNC_MAX_DURATION_MILLIS: int
mInLayout: boolean
mOnItemSelectedListener: OnItemSelectedListener
mOnItemClickListener: OnItemClickListener
mOnItemLongClickListener: OnItemLongClickListener
mDataChanged: boolean
mNextSelectedPosition: int
mNextSelectedRowId: long
mSelectedPosition: int
mSelectedRowId: long
mEmptyView: View
mItemCount: int
mOldItemCount: int
INVALID_POSITION: int
INVALID_ROW_ID: long
mOldSelectedPosition: int
mOldSelectedRowId: long
mDesiredFocusableState: boolean
mDesiredFocusableInTouchModeState: boolean
mSelectionNotifier: SelectionNotifier
mBlockLayoutRequests: boolean
CarouselAdapter(Context)
CarouselAdapter(Context,AttributeSet)
CarouselAdapter(Context,AttributeSet,int)


<<Java Class>>
Rotator
com.carousel.controls
mMode: int
mStartAngle: float
mCurrAngle: float
mStartTime: long
mDuration: long
mDeltaAngle: float
mFinished: boolean
mCoeffVelocity: float
mVelocity: float
DEFAULT_DURATION: int
SCROLL_MODE: int
FLING_MODE: int
mDeceleration: float
Rotator(Context)
isFinished():boolean
forceFinished(boolean):void
getDuration():long
getCurrAngle():float
getCurrVelocity():float
getStartAngle():float
timePassed():int
extendDuration(int):void
abortAnimation():void
computeAngleOffset():boolean
startRotate(float,float,int):void
startRotate(float,float):void
fling(float):void

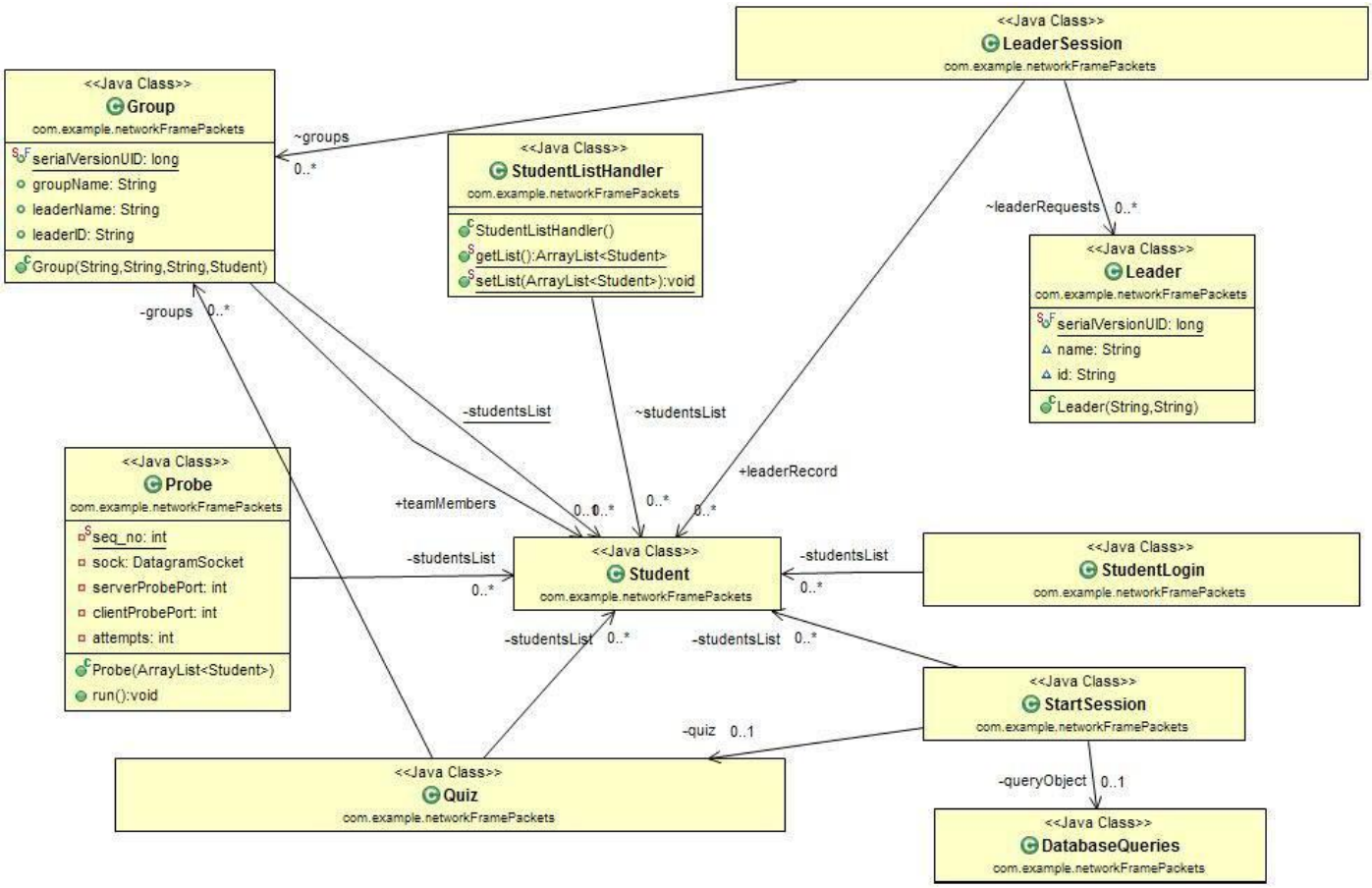
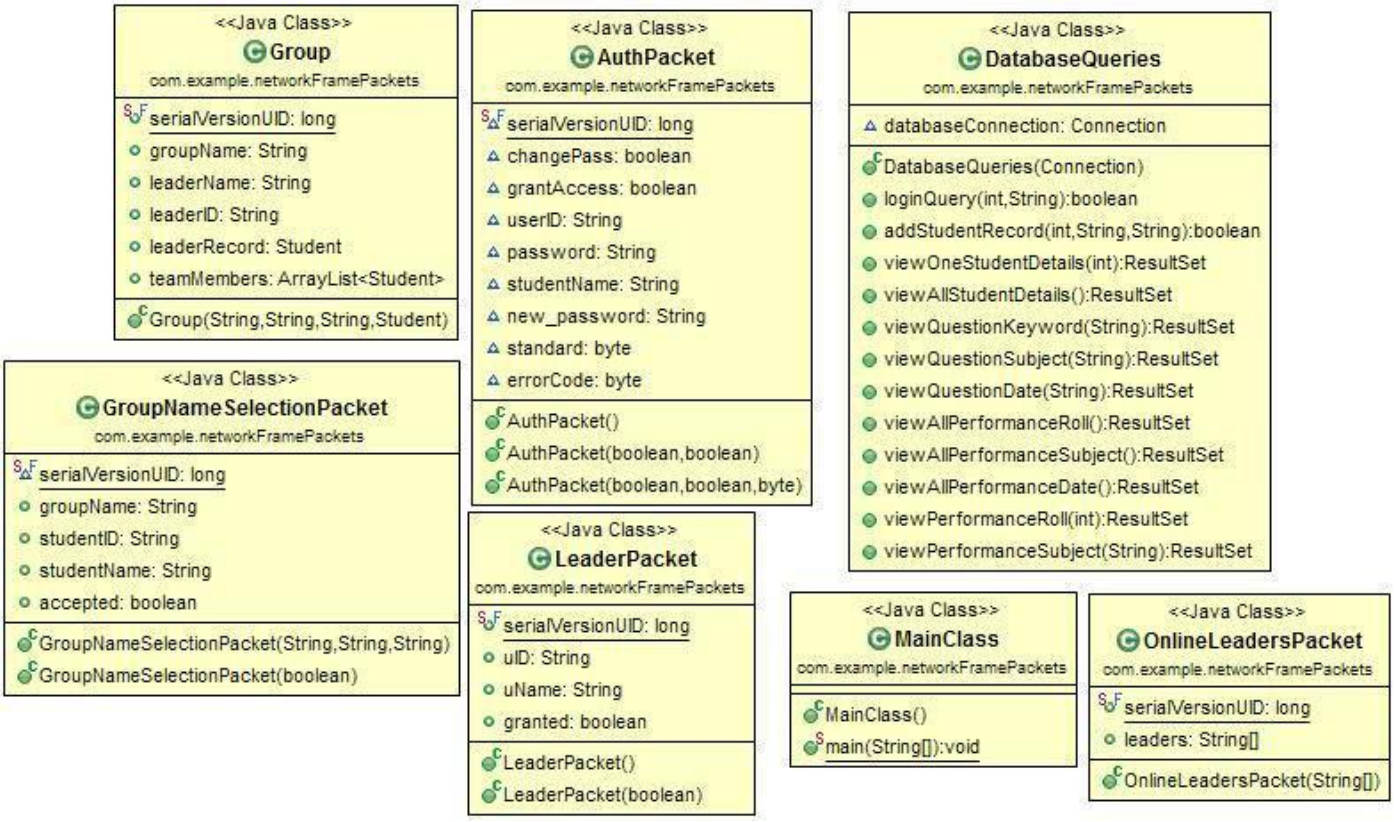
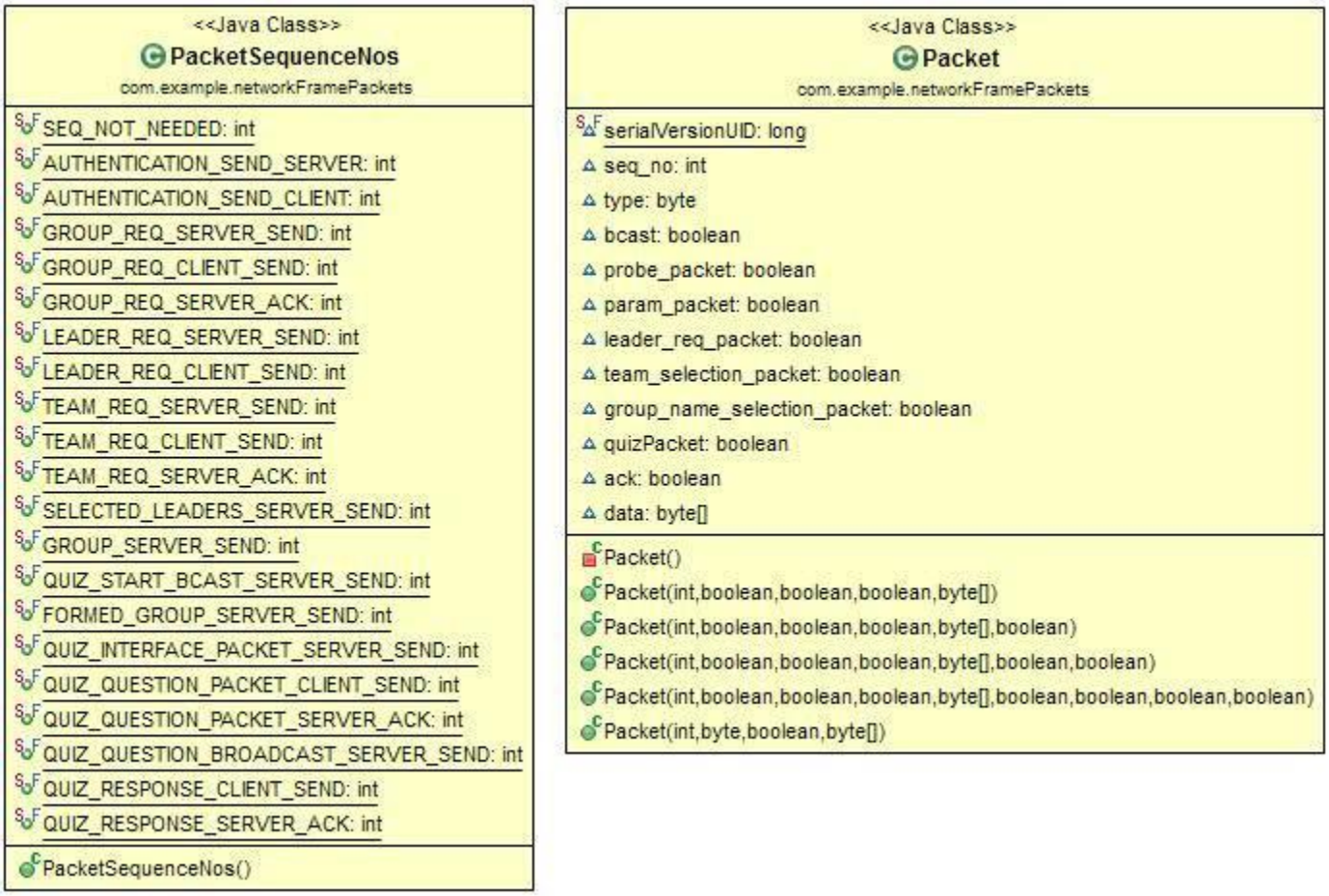
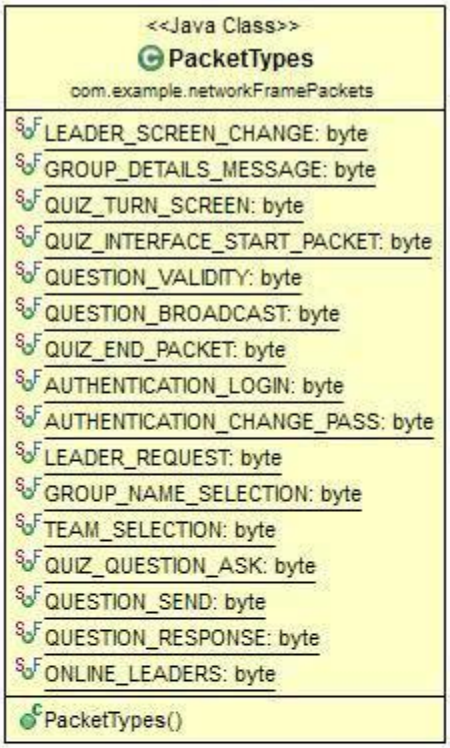
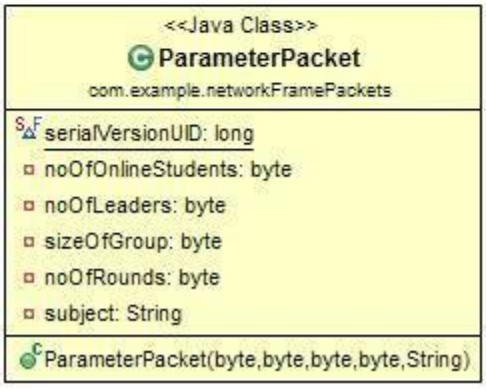


Figure 5.10: Class diagram for 'Eval'

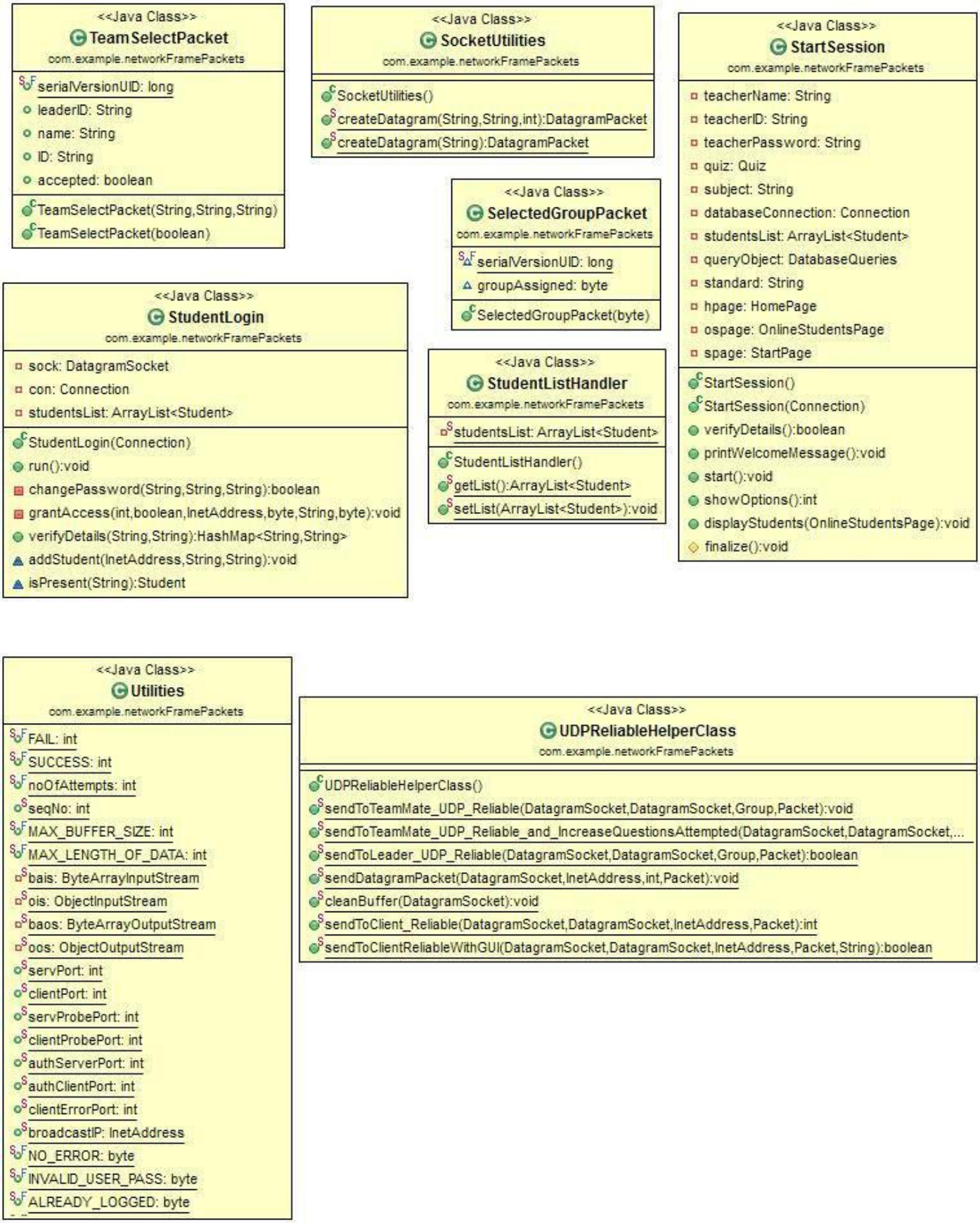

<<Java Class>>
Group
com.example.networkFramePackets
serialVersionUID: long
groupName: String
leaderName: String
leaderID: String
leaderRecord: Student
teamMembers: ArrayList<Student>
Group(String,String,String,Student)
<<Java Class>>
AuthPacket
com.example.networkFramePackets
serialVersionUID: long
changePass: boolean
grantAccess: boolean
userID: String
password: String
studentName: String
new_password: String
standard: byte
errorCode: byte
AuthPacket()
AuthPacket(boolean,boolean)
AuthPacket(boolean,boolean,byte)
<<Java Class>>
DatabaseQueries
com.example.networkFramePackets
databaseConnection: Connection
DatabaseQueries(Connection)
loginQuery(int,String):boolean
addStudentRecord(int,String,String):boolean
viewOneStudentDetails(int):ResultSet
viewAllStudentDetails():ResultSet
viewQuestionKeyword(String):ResultSet
viewQuestionSubject(String):ResultSet
viewQuestionDate(String):ResultSet
viewAllPerformanceRoll():ResultSet
viewAllPerformanceSubject():ResultSet
viewAllPerformanceDate():ResultSet
viewPerformanceRoll(int):ResultSet
viewPerformanceSubject(String):ResultSet
<<Java Class>>
GroupNameSelectionPacket
com.example.networkFramePackets
serialVersionUID: long
groupName: String
studentID: String
studentName: String
accepted: boolean
GroupNameSelectionPacket(String,String,String)
GroupNameSelectionPacket(boolean)
<<Java Class>>
LeaderPacket
com.example.networkFramePackets
serialVersionUID: long
uID: String
uName: String
granted: boolean
LeaderPacket()
LeaderPacket(boolean)
<<Java Class>>
MainClass
com.example.networkFramePackets
MainClass()
main(String[]):void
<<Java Class>>
OnlineLeadersPacket
com.example.networkFramePackets
serialVersionUID: long
leaders: String[]
OnlineLeadersPacket(String[])


<<Java Class>>
PacketSequenceNos
com.example.networkFramePackets
SEQ_NOT_NEEDED: int
AUTHENTICATION_SEND_SERVER: int
AUTHENTICATION_SEND_CLIENT: int
GROUP_REQ_SERVER_SEND: int
GROUP_REQ_CLIENT_SEND: int
GROUP_REQ_SERVER_ACK: int
LEADER_REQ_SERVER_SEND: int
LEADER_REQ_CLIENT_SEND: int
TEAM_REQ_SERVER_SEND: int
TEAM_REQ_CLIENT_SEND: int
TEAM_REQ_SERVER_ACK: int
SELECTED_LEADERS_SERVER_SEND: int
GROUP_SERVER_SEND: int
QUIZ_START_BCAST_SERVER_SEND: int
FORMED_GROUP_SERVER_SEND: int
QUIZ_INTERFACE_PACKET_SERVER_SEND: int
QUIZ_QUESTION_PACKET_CLIENT_SEND: int
QUIZ_QUESTION_PACKET_SERVER_ACK: int
QUIZ_QUESTION_BROADCAST_SERVER_SEND: int
QUIZ_RESPONSE_CLIENT_SEND: int
QUIZ_RESPONSE_SERVER_ACK: int
PacketSequenceNos()
<<Java Class>>
Packet
com.example.networkFramePackets
serialVersionUID: long
seq_no: int
type: byte
bcast: boolean
probe_packet: boolean
param_packet: boolean
leader_req_packet: boolean
team_selection_packet: boolean
group_name_selection_packet: boolean
quizPacket: boolean
ack: boolean
data: byte[]
Packet()
Packet(int,boolean,boolean,boolean,byte[])
Packet(int,boolean,boolean,boolean,byte[],boolean)
Packet(int,boolean,boolean,boolean,byte[],boolean,boolean)
Packet(int,boolean,boolean,boolean,byte[],boolean,boolean,boolean,boolean)
Packet(int,byte,boolean,byte[])


<<Java Class>>
PacketTypes
com.example.networkFramePackets
LEADER_SCREEN_CHANGE: byte
GROUP_DETAILS_MESSAGE: byte
QUIZ_TURN_SCREEN: byte
QUIZ_INTERFACE_START_PACKET: byte
QUESTION_VALIDITY: byte
QUESTION_BROADCAST: byte
QUIZ_END_PACKET: byte
AUTHENTICATION_LOGIN: byte
AUTHENTICATION_CHANGE_PASS: byte
LEADER_REQUEST: byte
GROUP_NAME_SELECTION: byte
TEAM_SELECTION: byte
QUIZ_QUESTION_ASK: byte
QUESTION_SEND: byte
QUESTION_RESPONSE: byte
ONLINE_LEADERS: byte
PacketTypes()


<<Java Class>>
ParameterPacket
com.example.networkFramePackets
serialVersionUID: long
noOfOnlineStudents: byte
noOfLeaders: byte
sizeOfGroup: byte
noOfRounds: byte
subject: String
ParameterPacket(byte,byte,byte,byte,String)


<<Java Class>>
Quiz
com.example.networkFramePackets
noOfStudents: byte
noOfGroups: byte
noOfStudentsInGroup: byte
noOfRounds: byte
studentsList: ArrayList<Student>
leaderList: ArrayList<String>
answeredStuds: ArrayList<String>
groups: ArrayList<Group>
questionTimelimitInSeconds: int
AnswerTimeLimitInSeconds: int
subject: String
teacherName: String
standard: String
sendSocket: DatagramSocket
recvSocket: DatagramSocket
broadcastIP: InetAddress
con: Connection
questions: ArrayList<Question>
questionLevel: byte
running: boolean
qwp: QuestionWaitPage
mcf: MultipleChoiceFrame
tff: TrueOrFalseFrame
owf: OneWordFrame
lf: LevelFrame
rw: ResponseWait
ext: QuestionTimeExtension
rps: RetryPacketSend
hp: HomePage
waitbackground: waitPageGUI


<<Java Class>>
TeamSelectPacket
com.example.networkFramePackets
serialVersionUID: long
leaderID: String
name: String
ID: String
accepted: boolean
TeamSelectPacket(String,String,String)
TeamSelectPacket(boolean)
<<Java Class>>
SocketUtilities
com.example.networkFramePackets
SocketUtilities()
createDatagram(String,String,int):DatagramPacket
createDatagram(String):DatagramPacket
<<Java Class>>
StartSession
com.example.networkFramePackets
teacherName: String
teacherID: String
teacherPassword: String
quiz: Quiz
subject: String
databaseConnection: Connection
studentsList: ArrayList<Student>
queryObject: DatabaseQueries
standard: String
hpage: HomePage
ospage: OnlineStudentsPage
spage: StartPage
StartSession()
StartSession(Connection)
verifyDetails():boolean
printWelcomeMessage():void
start():void
showOptions():int
displayStudents(OnlineStudentsPage):void
finalize():void
<<Java Class>>
SelectedGroupPacket
com.example.networkFramePackets
serialVersionUID: long
groupAssigned: byte
SelectedGroupPacket(byte)
<<Java Class>>
StudentLogin
com.example.networkFramePackets
sock: DatagramSocket
con: Connection
studentsList: ArrayList<Student>
StudentLogin(Connection)
run():void
changePassword(String,String,String):boolean
grantAccess(int,boolean,InetAddress,byte,String,byte):void
verifyDetails(String,String):HashMap<String,String>
addStudent(InetAddress,String,String):void
isPresent(String):Student
<<Java Class>>
StudentListHandler
com.example.networkFramePackets
studentsList: ArrayList<Student>
StudentListHandler()
getList():ArrayList<Student>
setList(ArrayList<Student>):void
<<Java Class>>
Utilities
com.example.networkFramePackets
FAIL: int
SUCCESS: int
noOfAttempts: int
seqNo: int
MAX_BUFFER_SIZE: int
MAX_LENGTH_OF_DATA: int
bais: ByteArrayInputStream
ois: ObjectInputStream
baos: ByteArrayOutputStream
oos: ObjectOutputStream
servPort: int
clientPort: int
servProbePort: int
clientProbePort: int
authServerPort: int
authClientPort: int
clientErrorPort: int
broadcastIP: InetAddress
NO_ERROR: byte
INVALID_USER_PASS: byte
ALREADY_LOGGED: byte
<<Java Class>>
UDPReliableHelperClass
com.example.networkFramePackets
UDPReliableHelperClass()
sendToTeamMate_UDP_Reliable(DatagramSocket,DatagramSocket,Group,Packet):void
sendToTeamMate_UDP_Reliable_and_IncreaseQuestionsAttempted(DatagramSocket,DatagramSocket,...
sendToLeader_UDP_Reliable(DatagramSocket,DatagramSocket,Group,Packet):boolean
sendDatagramPacket(DatagramSocket,InetAddress,int,Packet):void
cleanBuffer(DatagramSocket):void
sendToClient_Reliable(DatagramSocket,DatagramSocket,InetAddress,Packet):int
sendToClientReliableWithGUI(DatagramSocket,DatagramSocket,InetAddress,Packet,String):boolean

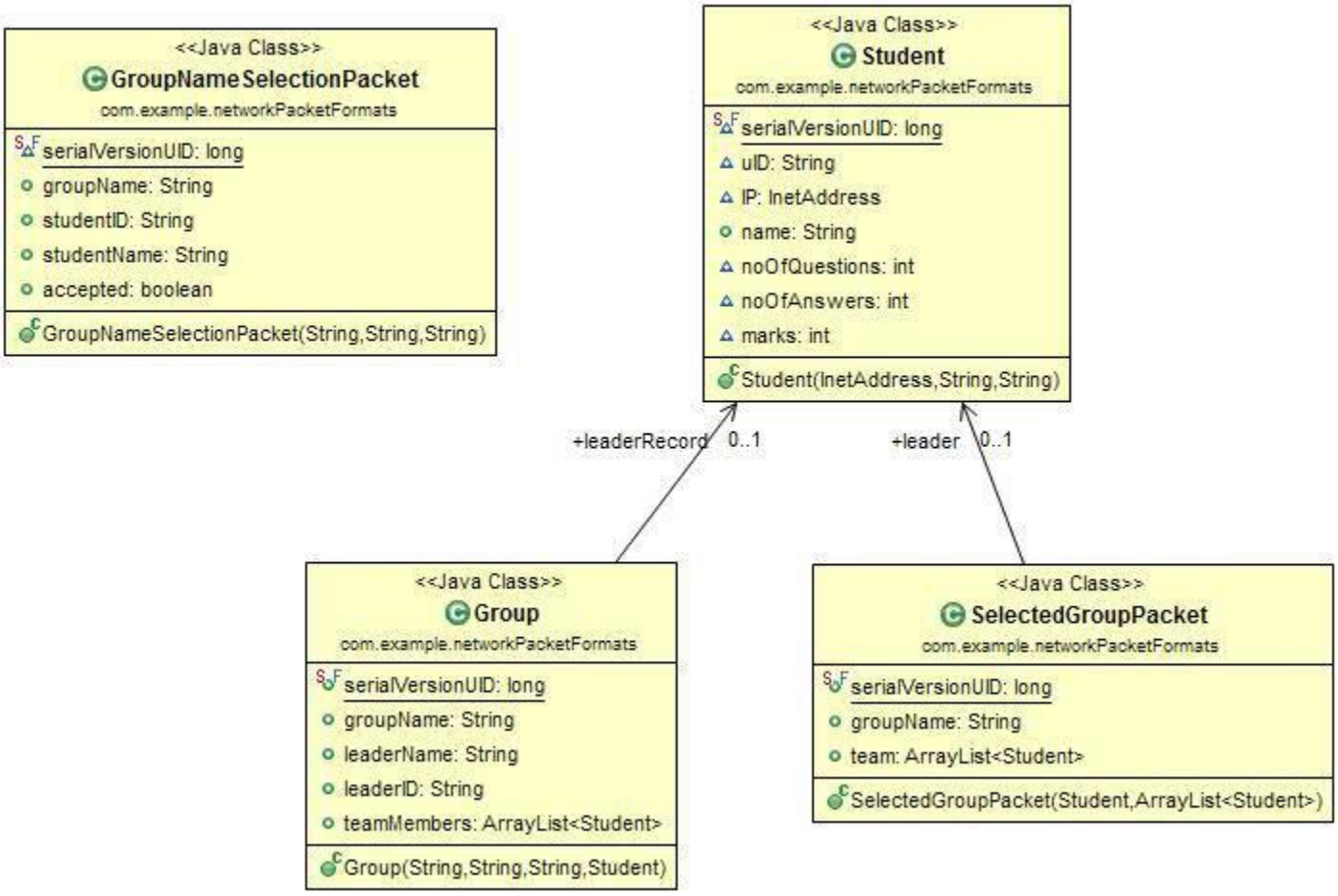
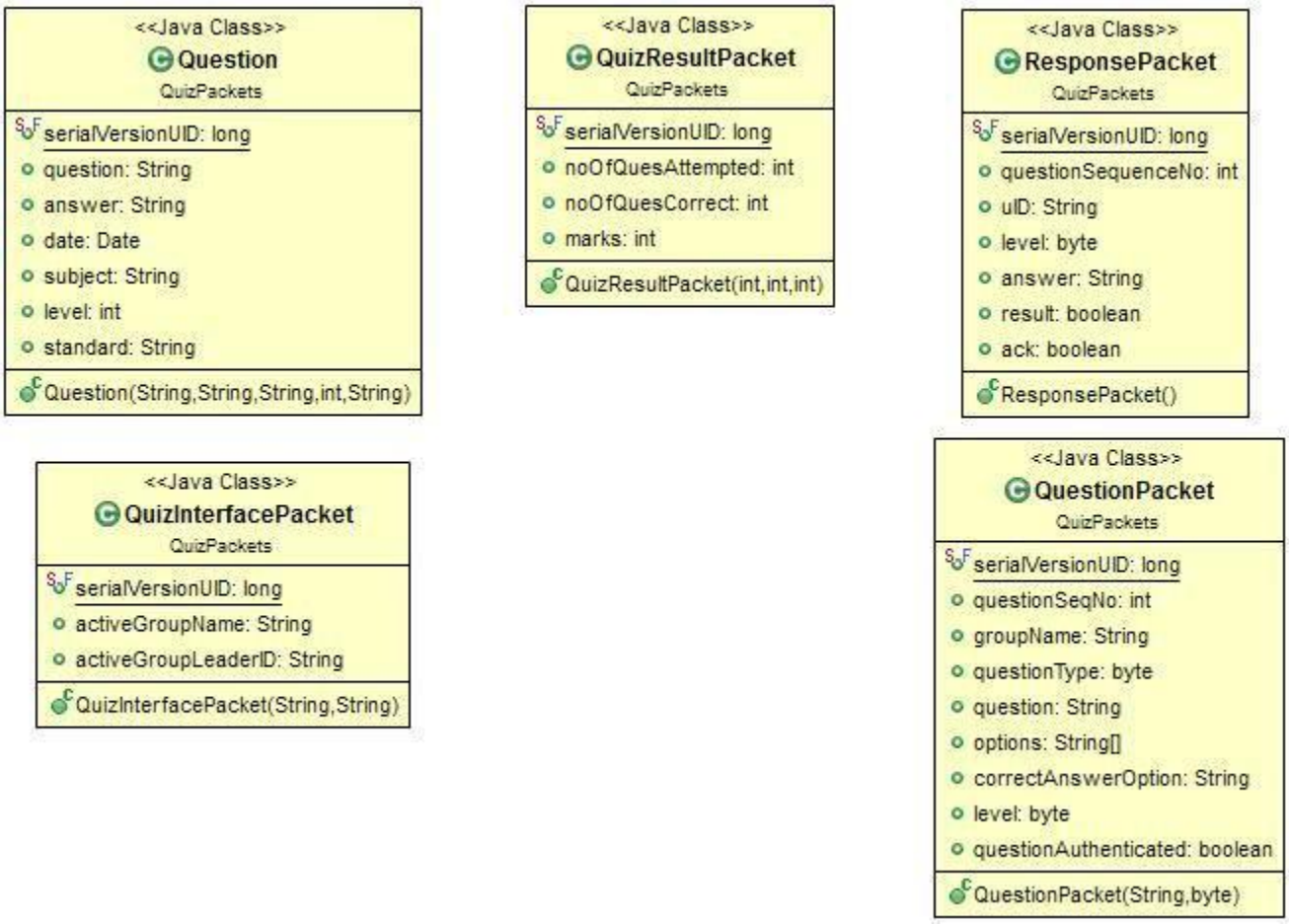
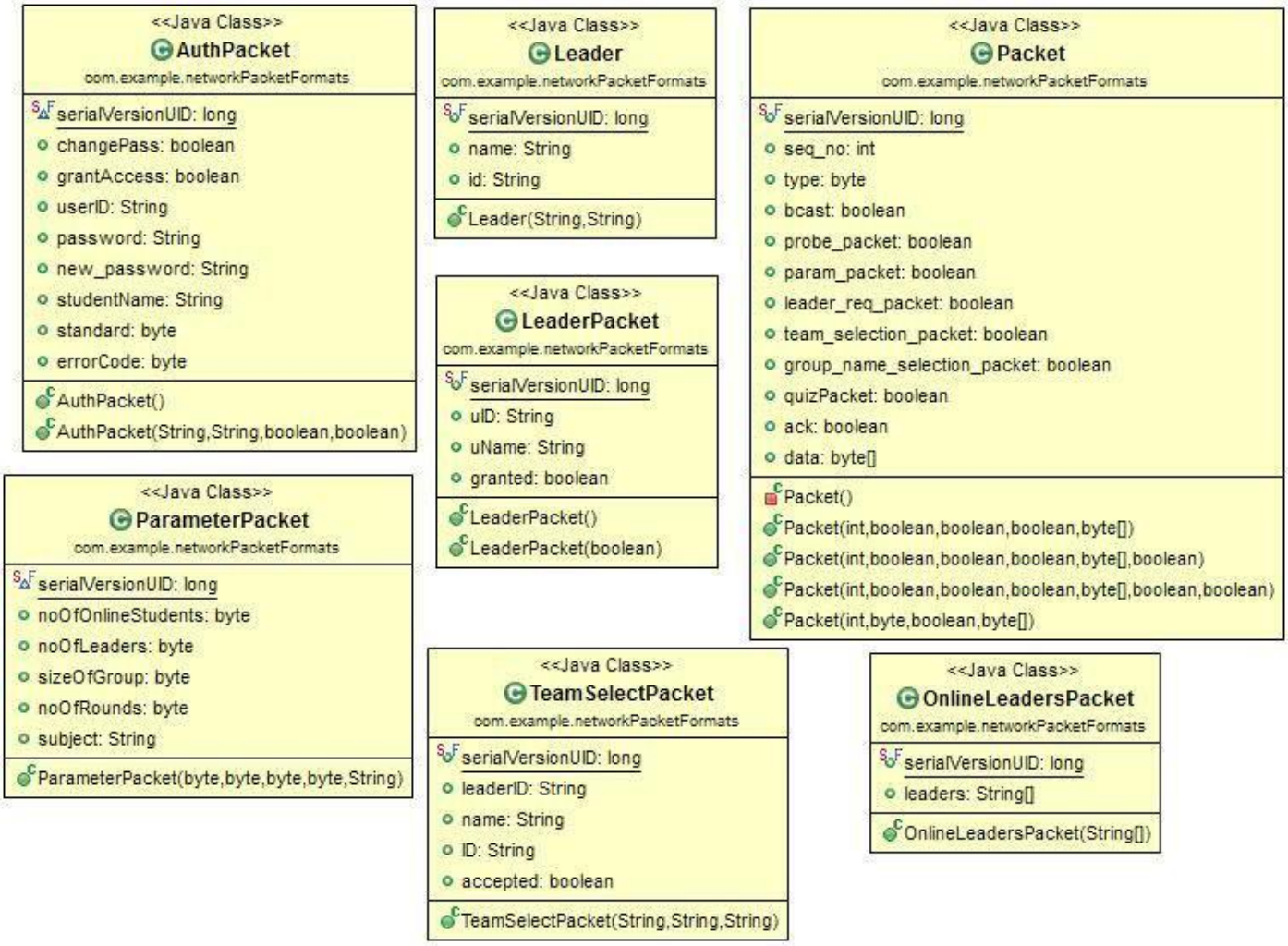


Figure 5.11: Class diagram for 'Eval'


<<Java Class>>
Question
QuizPackets
serialVersionUID: long
question: String
answer: String
date: Date
subject: String
level: int
standard: String
Question(String,String,String,int,String)
<<Java Class>>
QuizResultPacket
QuizPackets
serialVersionUID: long
noOfQuesAttempted: int
noOfQuesCorrect: int
marks: int
QuizResultPacket(int,int,int)
<<Java Class>>
ResponsePacket
QuizPackets
serialVersionUID: long
questionSequenceNo: int
uID: String
level: byte
answer: String
result: boolean
ack: boolean
ResponsePacket()
<<Java Class>>
QuizInterfacePacket
QuizPackets
serialVersionUID: long
activeGroupName: String
activeGroupLeaderID: String
QuizInterfacePacket(String,String)
<<Java Class>>
QuestionPacket
QuizPackets
serialVersionUID: long
questionSeqNo: int
groupName: String
questionType: byte
question: String
options: String[]
correctAnswerOption: String
level: byte
questionAuthenticated: boolean
QuestionPacket(String,byte)


<<Java Class>>
AuthPacket
com.example.networkPacketFormats
serialVersionUID: long
changePass: boolean
grantAccess: boolean
userID: String
password: String
new_password: String
studentName: String
standard: byte
errorCode: byte
AuthPacket()
AuthPacket(String,String,boolean,boolean)
<<Java Class>>
Leader
com.example.networkPacketFormats
serialVersionUID: long
name: String
id: String
Leader(String,String)
<<Java Class>>
Packet
com.example.networkPacketFormats
serialVersionUID: long
seq_no: int
type: byte
bcast: boolean
probe_packet: boolean
param_packet: boolean
leader_req_packet: boolean
team_selection_packet: boolean
group_name_selection_packet: boolean
quizPacket: boolean
ack: boolean
data: byte[]
Packet()
Packet(int,boolean,boolean,boolean,byte[])
Packet(int,boolean,boolean,boolean,byte[],boolean)
Packet(int,boolean,boolean,boolean,byte[],boolean,boolean)
Packet(int,byte,boolean,byte[])
<<Java Class>>
LeaderPacket
com.example.networkPacketFormats
serialVersionUID: long
uID: String
uName: String
granted: boolean
LeaderPacket()
LeaderPacket(boolean)
<<Java Class>>
ParameterPacket
com.example.networkPacketFormats
serialVersionUID: long
noOfOnlineStudents: byte
noOfLeaders: byte
sizeOfGroup: byte
noOfRounds: byte
subject: String
ParameterPacket(byte,byte,byte,byte,String)
<<Java Class>>
TeamSelectPacket
com.example.networkPacketFormats
serialVersionUID: long
leaderID: String
name: String
ID: String
accepted: boolean
TeamSelectPacket(String,String,String)
<<Java Class>>
OnlineLeadersPacket
com.example.networkPacketFormats
serialVersionUID: long
leaders: String[]
OnlineLeadersPacket(String[])

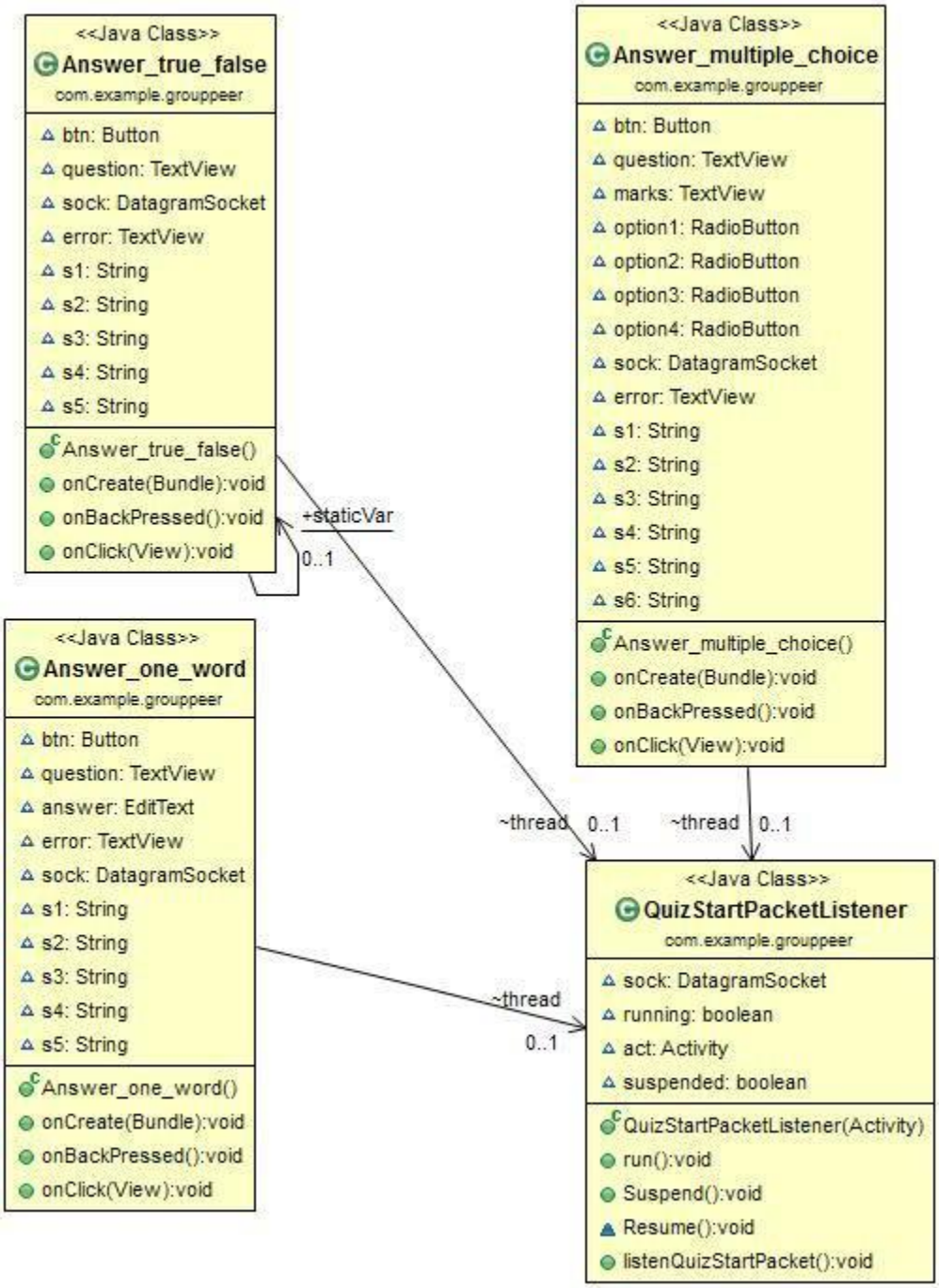
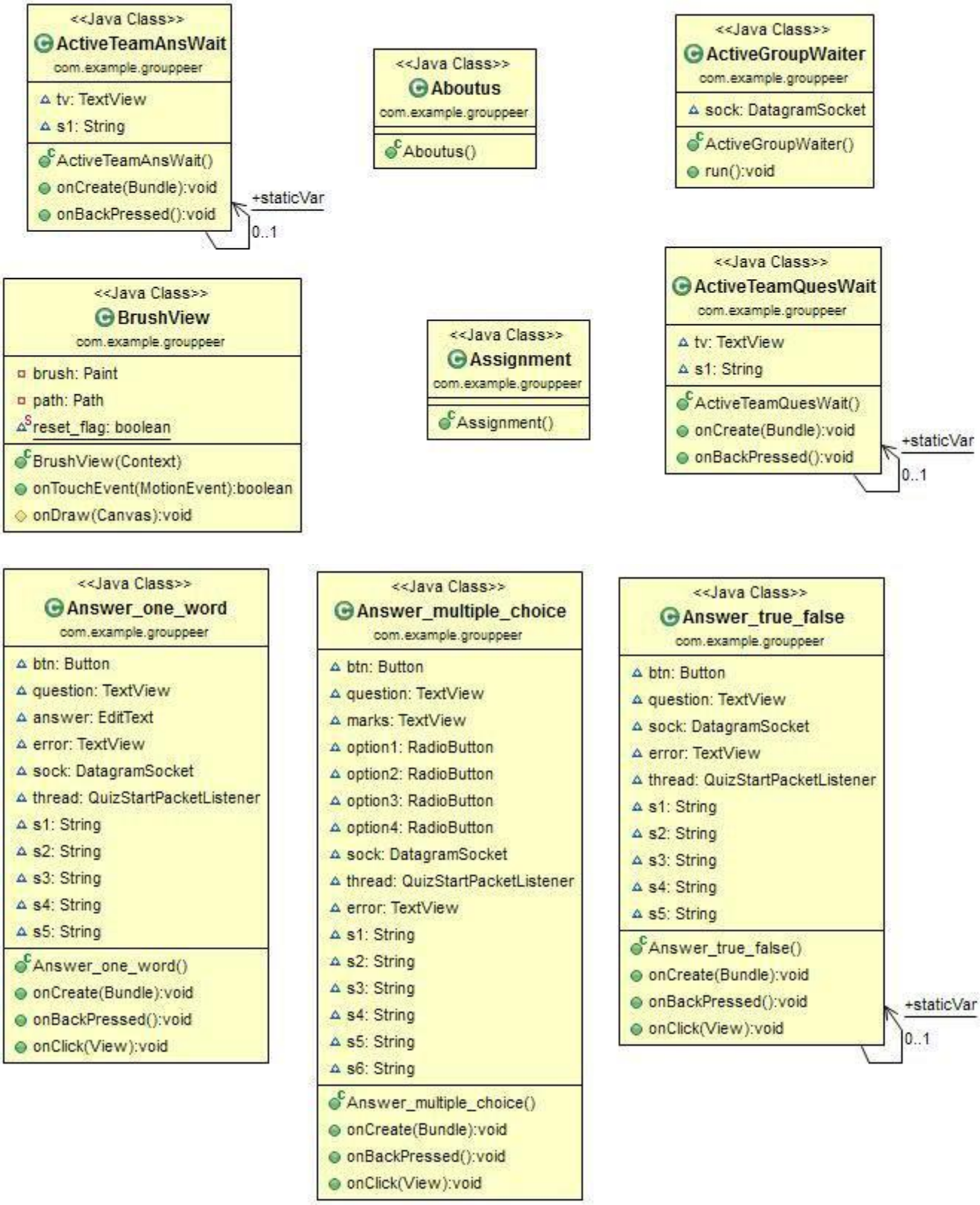
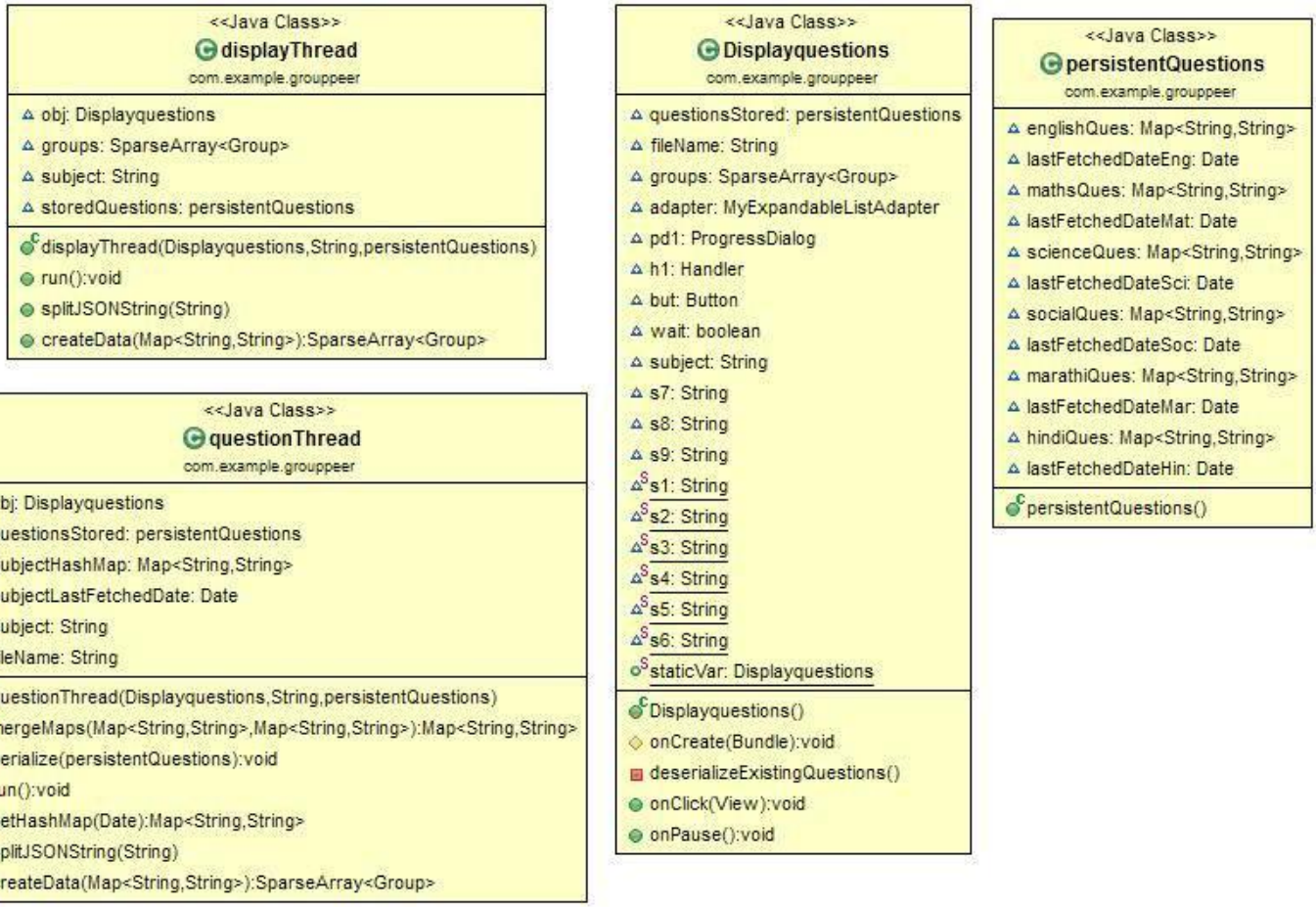
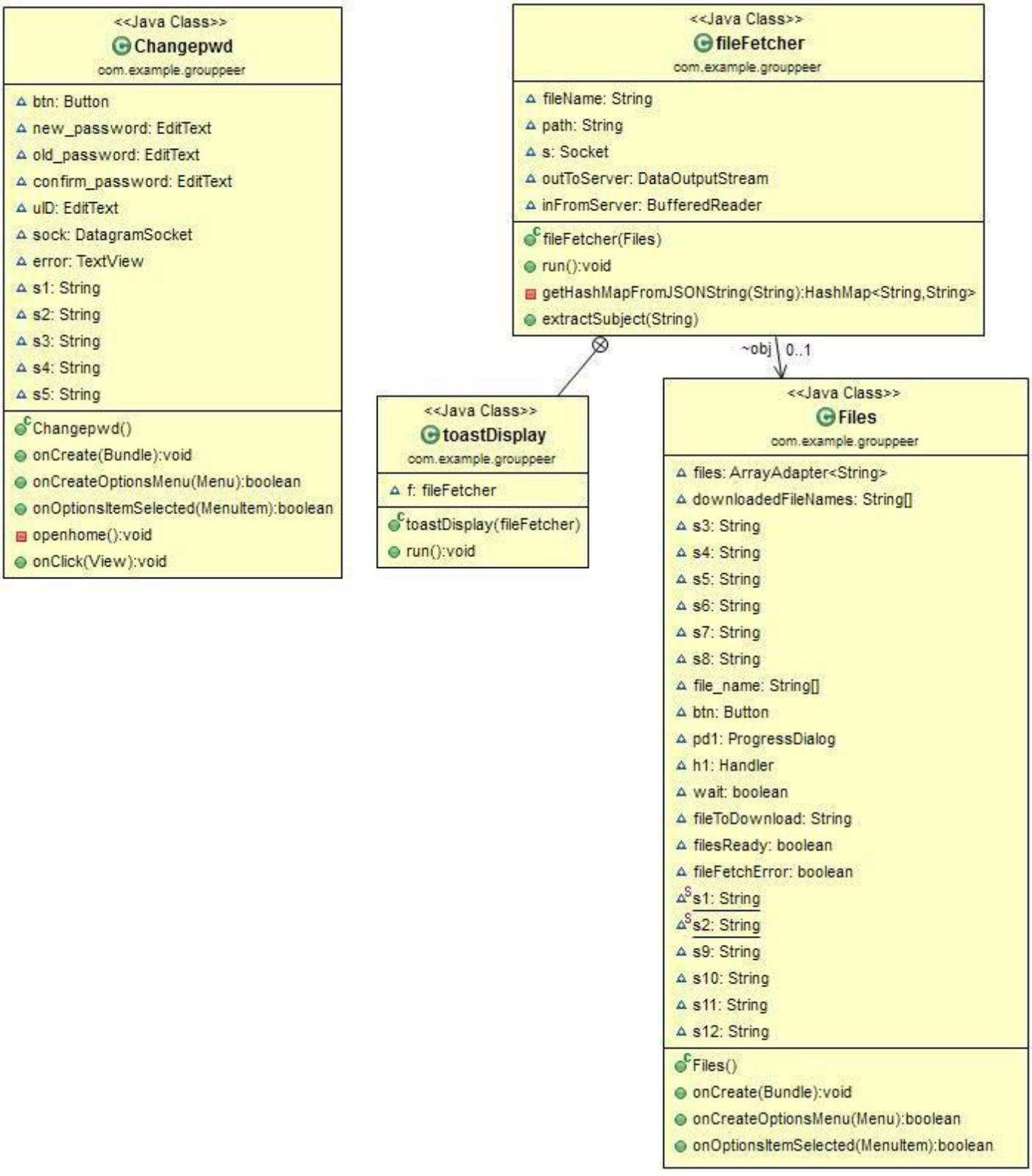
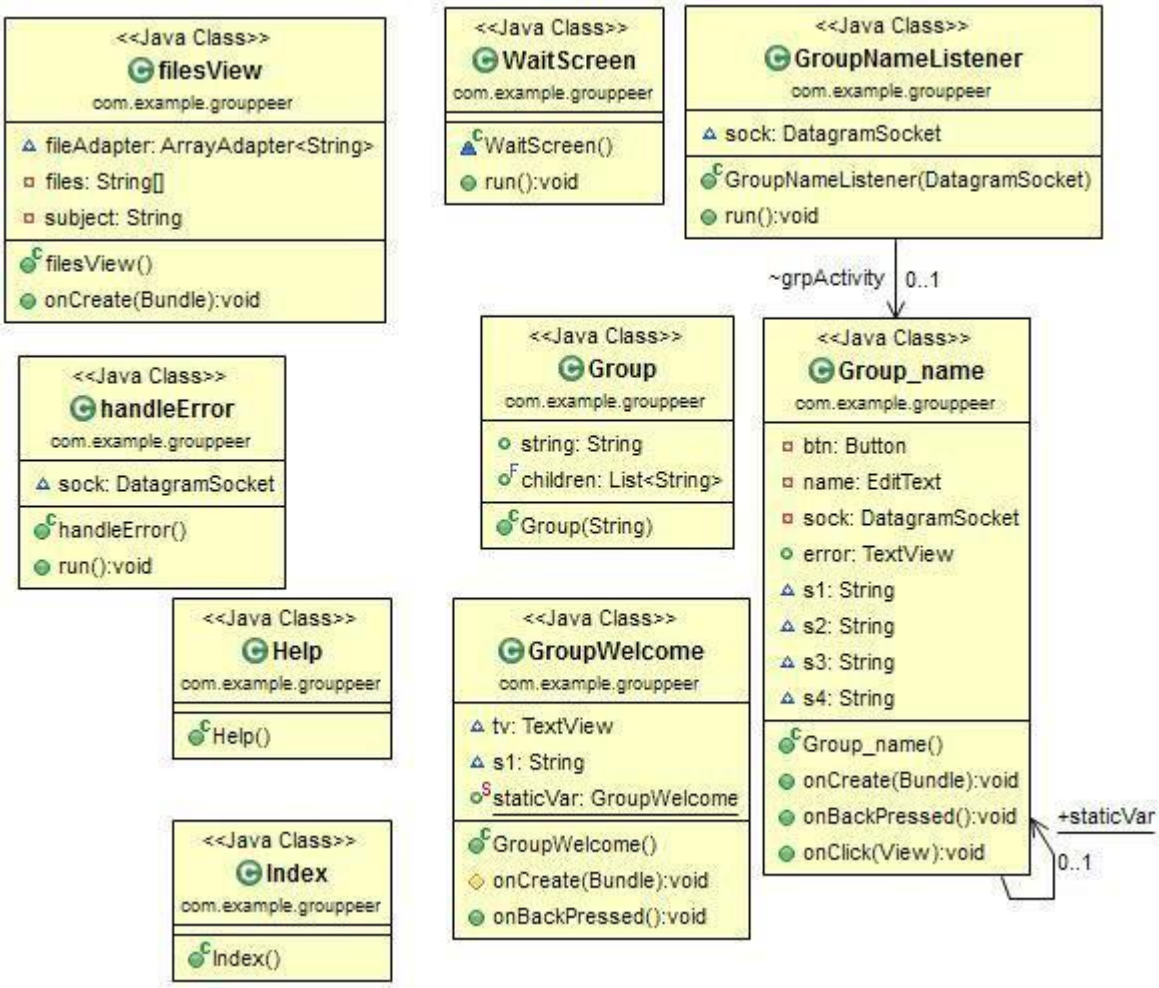


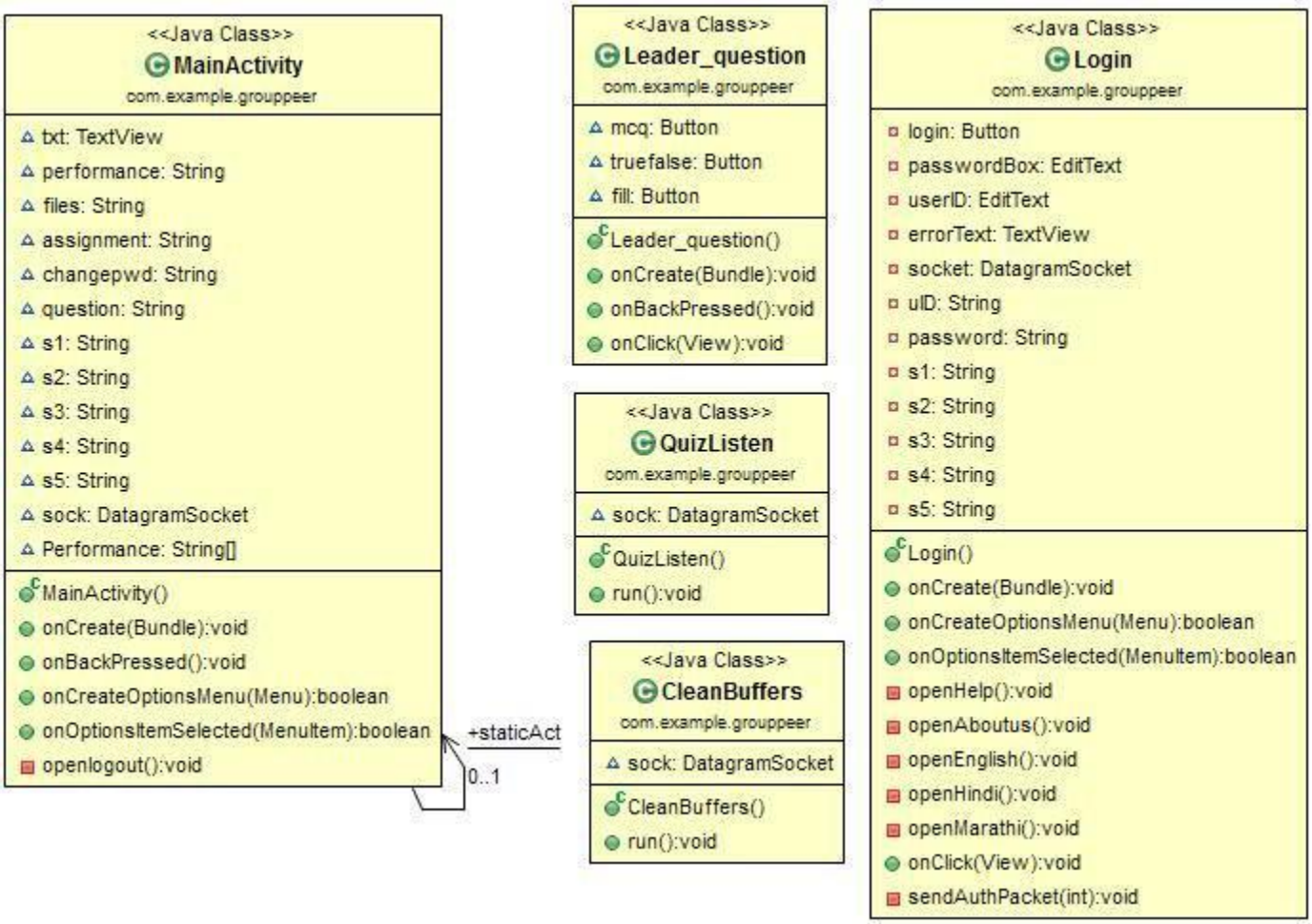
Figure 5.12: Class diagram for 'Eval'

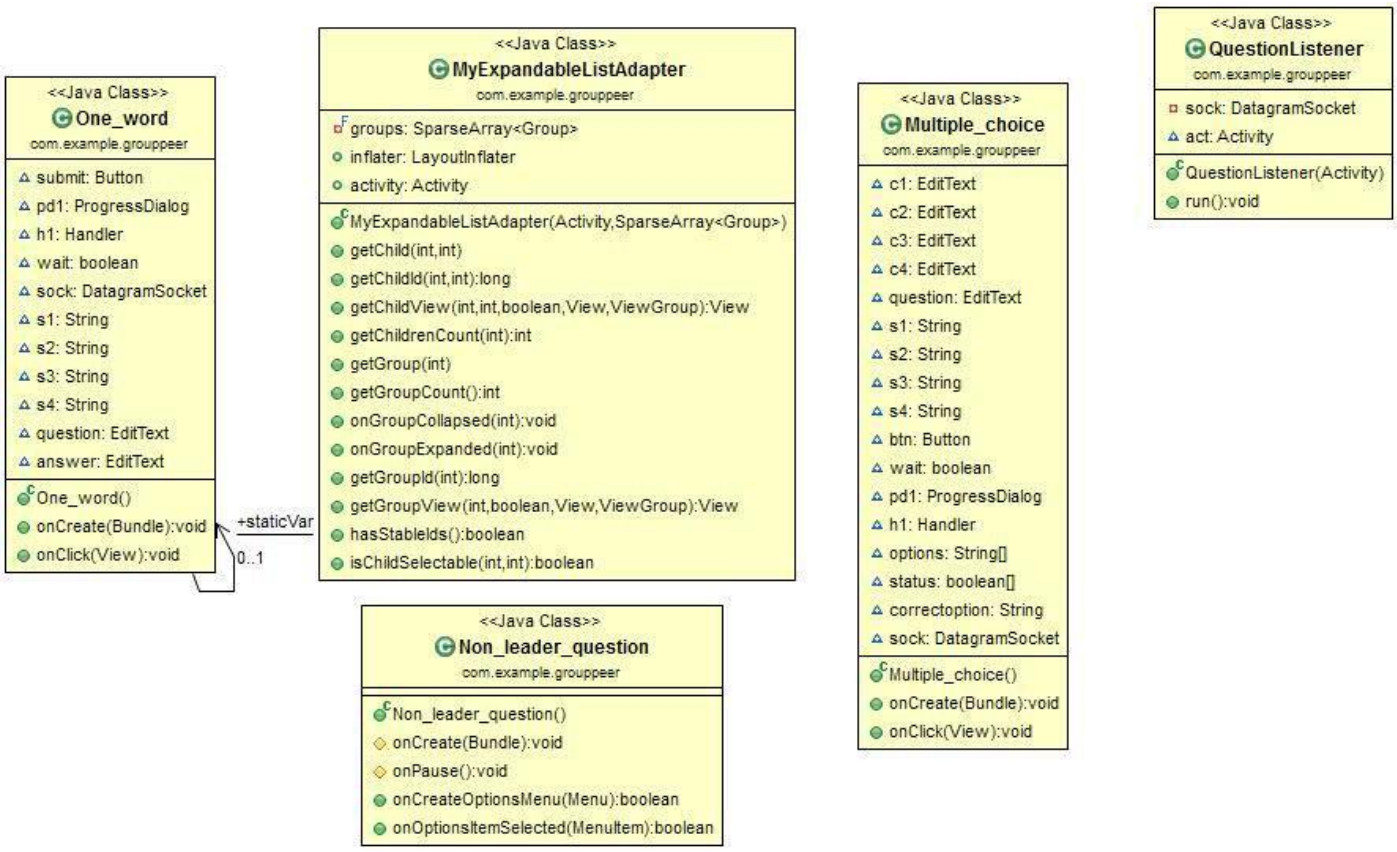

<<Java Class>>
ActiveTeamAnsWait
com.example.grouppeer
tv: TextView
s1: String
ActiveTeamAnsWait()
onCreate(Bundle):void
onBackPressed():void
+staticVar
0..1
<<Java Class>>
Aboutus
com.example.grouppeer
Aboutus()
<<Java Class>>
ActiveGroupWaiter
com.example.grouppeer
sock: DatagramSocket
ActiveGroupWaiter()
run():void
<<Java Class>>
BrushView
com.example.grouppeer
brush: Paint
path: Path
reset_flag: boolean
BrushView(Context)
onTouchEvent(MotionEvent):boolean
onDraw(Canvas):void
<<Java Class>>
Assignment
com.example.grouppeer
Assignment()
<<Java Class>>
ActiveTeamQuesWait
com.example.grouppeer
tv: TextView
s1: String
ActiveTeamQuesWait()
onCreate(Bundle):void
onBackPressed():void
+staticVar
0..1
<<Java Class>>
Answer_one_word
com.example.grouppeer
btn: Button
question: TextView
answer: EditText
error: TextView
sock: DatagramSocket
thread: QuizStartPacketListener
s1: String
s2: String
s3: String
s4: String
s5: String
Answer_one_word()
onCreate(Bundle):void
onBackPressed():void
onClick(View):void
<<Java Class>>
Answer_multiple_choice
com.example.grouppeer
btn: Button
question: TextView
marks: TextView
option1: RadioButton
option2: RadioButton
option3: RadioButton
option4: RadioButton
sock: DatagramSocket
thread: QuizStartPacketListener
error: TextView
s1: String
s2: String
s3: String
s4: String
s5: String
s6: String
Answer_multiple_choice()
onCreate(Bundle):void
onBackPressed():void
onClick(View):void
<<Java Class>>
Answer_true_false
com.example.grouppeer
btn: Button
question: TextView
sock: DatagramSocket
error: TextView
thread: QuizStartPacketListener
s1: String
s2: String
s3: String
s4: String
s5: String
Answer_true_false()
onCreate(Bundle):void
onBackPressed():void
onClick(View):void
+staticVar
0..1

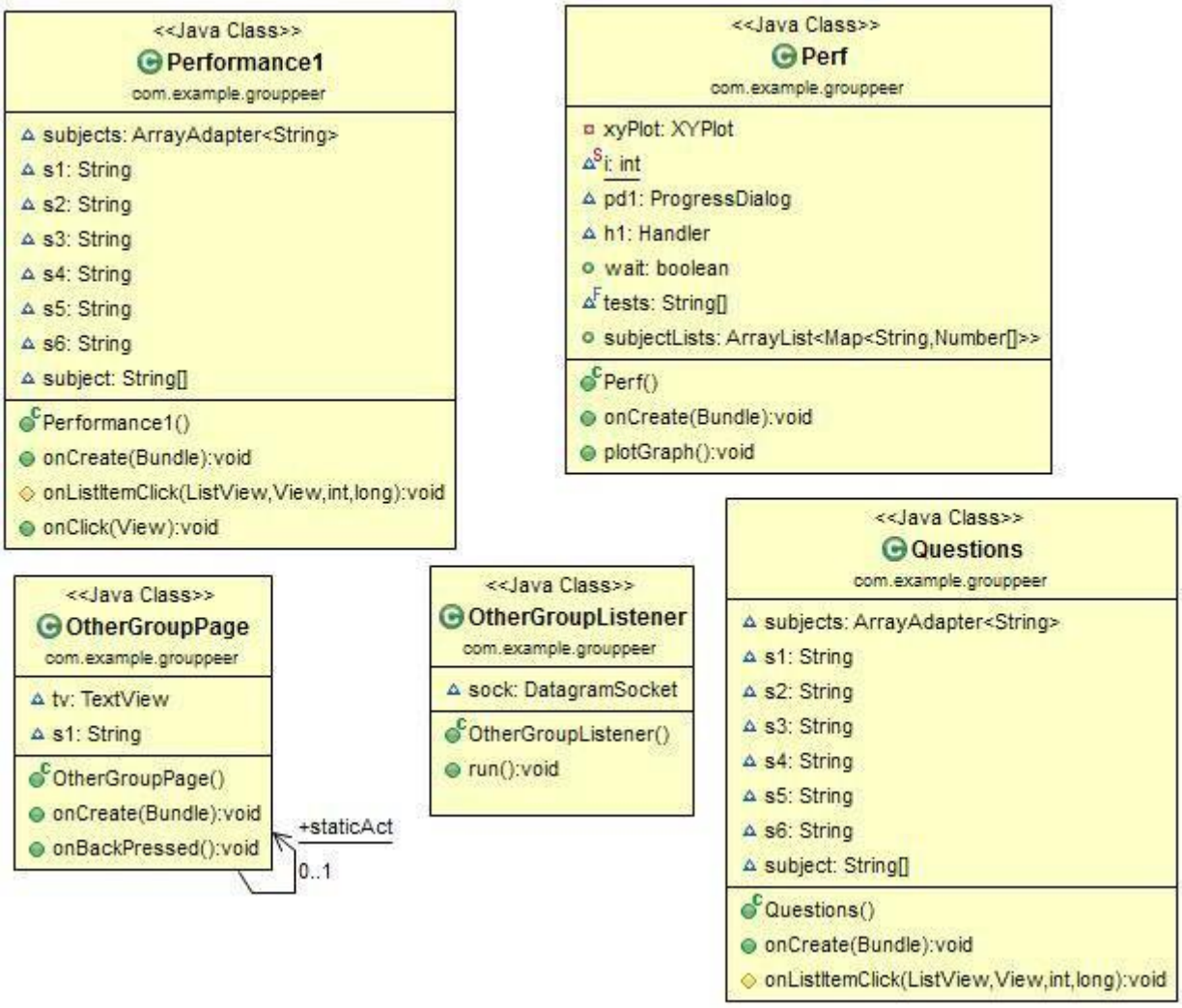

<<Java Class>>
displayThread
com.example.grouppeer
obj: Displayquestions
groups: SparseArray<Group>
subject: String
storedQuestions: persistentQuestions
displayThread(Displayquestions,String,persistentQuestions)
run():void
splitJSONString(String)
createData(Map<String,String>):SparseArray<Group>
<<Java Class>>
questionThread
com.example.grouppeer
obj: Displayquestions
questionsStored: persistentQuestions
subjectHashMap: Map<String,String>
subjectLastFetchedDate: Date
subject: String
fileName: String
questionThread(Displayquestions,String,persistentQuestions)
mergeMaps(Map<String,String>,Map<String,String>):Map<String,String>
serialize(persistentQuestions):void
run():void
getHashMap(Date):Map<String,String>
splitJSONString(String)
createData(Map<String,String>):SparseArray<Group>
<<Java Class>>
Displayquestions
com.example.grouppeer
questionsStored: persistentQuestions
fileName: String
groups: SparseArray<Group>
adapter: MyExpandableListAdapter
pd1: ProgressDialog
h1: Handler
but: Button
wait: boolean
subject: String
s7: String
s8: String
s9: String
s1: String
s2: String
s3: String
s4: String
s5: String
s6: String
staticVar: Displayquestions
Displayquestions()
onCreate(Bundle):void
deserializeExistingQuestions()
onClick(View):void
onPause():void
<<Java Class>>
persistentQuestions
com.example.grouppeer
englishQues: Map<String,String>
lastFetchedDateEng: Date
mathsQues: Map<String,String>
lastFetchedDateMat: Date
scienceQues: Map<String,String>
lastFetchedDateSci: Date
socialQues: Map<String,String>
lastFetchedDateSoc: Date
marathiQues: Map<String,String>
lastFetchedDateMar: Date
hindiQues: Map<String,String>
lastFetchedDateHin: Date
persistentQuestions()

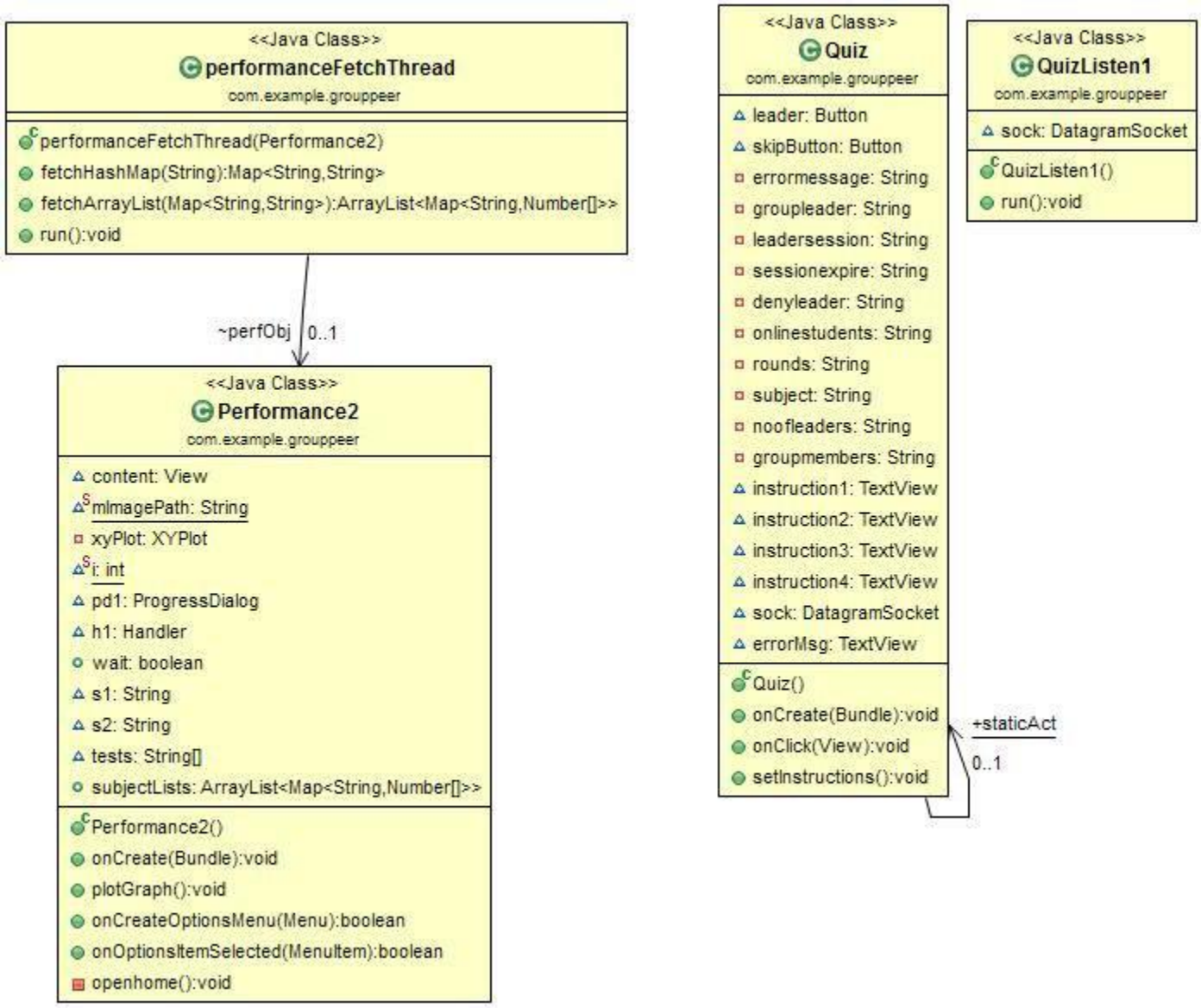

<<Java Class>>
Changepwd
com.example.grouppeer
btn: Button
new_password: EditText
old_password: EditText
confirm_password: EditText
uID: EditText
sock: DatagramSocket
error: TextView
s1: String
s2: String
s3: String
s4: String
s5: String
Changepwd()
onCreate(Bundle):void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
openhome():void
onClick(View):void
<<Java Class>>
fileFetcher
com.example.grouppeer
fileName: String
path: String
s: Socket
outToServer: DataOutputStream
inFromServer: BufferedReader
fileFetcher(Files)
run():void
getHashMapFromJSONString(String):HashMap<String,String>
extractSubject(String)
~obj
0..1
<<Java Class>>
toastDisplay
com.example.grouppeer
f: fileFetcher
toastDisplay(fileFetcher)
run():void
<<Java Class>>
Files
com.example.grouppeer
files: ArrayAdapter<String>
downloadedFileNames: String[]
s3: String
s4: String
s5: String
s6: String
s7: String
s8: String
file_name: String[]
btn: Button
pd1: ProgressDialog
h1: Handler
wait: boolean
fileToDownload: String
filesReady: boolean
fileFetchError: boolean
s1: String
s2: String
s9: String
s10: String
s11: String
s12: String
Files()
onCreate(Bundle):void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean

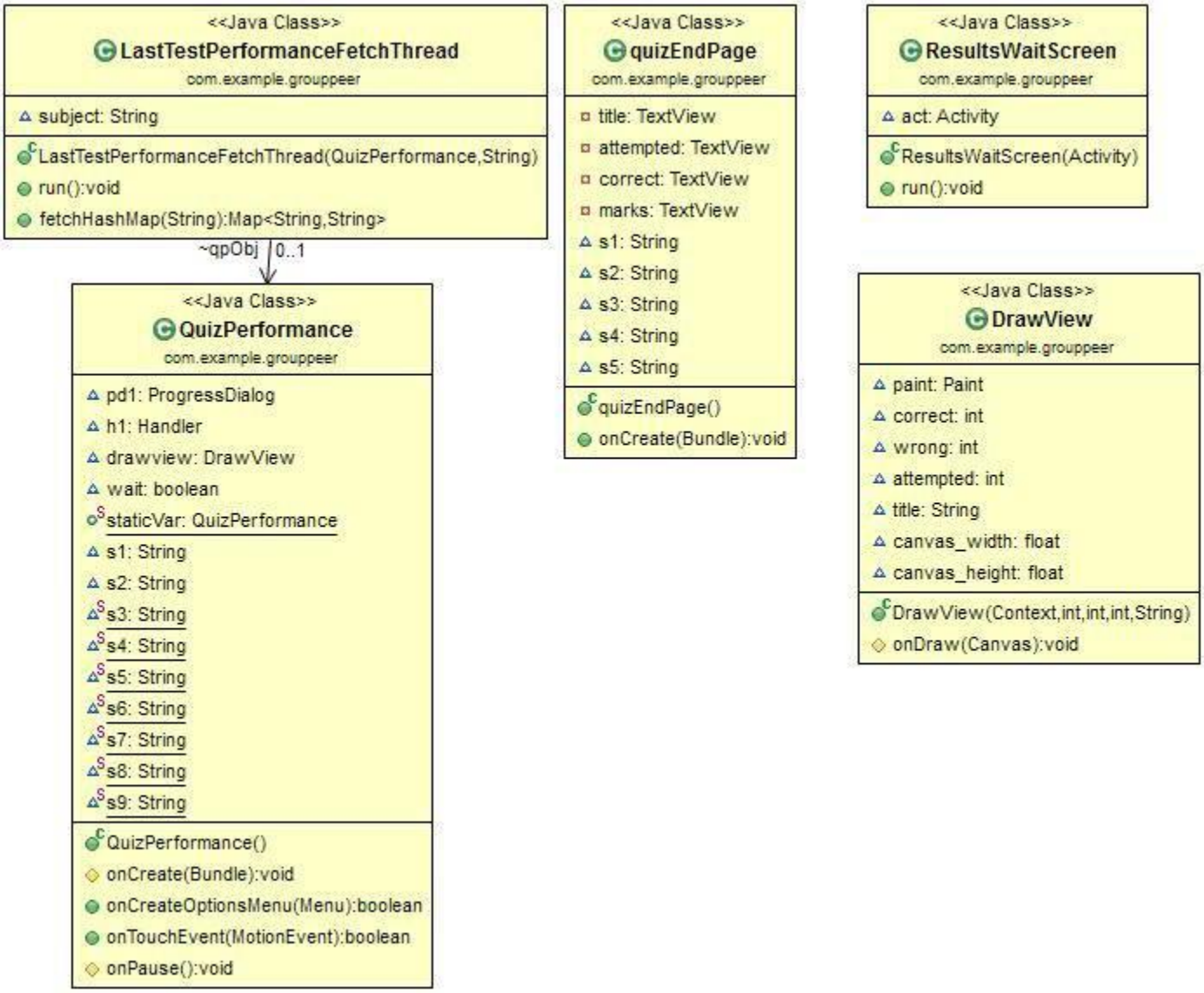

<<Java Class>>
filesView
com.example.grouppeer
fileAdapter: ArrayAdapter<String>
files: String[]
subject: String
filesView()
onCreate(Bundle):void
<<Java Class>>
WaitScreen
com.example.grouppeer
WaitScreen()
run():void
<<Java Class>>
GroupNameListener
com.example.grouppeer
sock: DatagramSocket
GroupNameListener(DatagramSocket)
run():void
~grpActivity
0..1
<<Java Class>>
handleError
com.example.grouppeer
sock: DatagramSocket
handleError()
run():void
<<Java Class>>
Group
com.example.grouppeer
string: String
children: List<String>
Group(String)
<<Java Class>>
Group_name
com.example.grouppeer
btn: Button
name: EditText
sock: DatagramSocket
error: TextView
s1: String
s2: String
s3: String
s4: String
Group_name()
onCreate(Bundle):void
onBackPressed():void
onClick(View):void
+staticVar
0..1
<<Java Class>>
Help
com.example.grouppeer
Help()
<<Java Class>>
GroupWelcome
com.example.grouppeer
tv: TextView
s1: String
staticVar: GroupWelcome
GroupWelcome()
onCreate(Bundle):void
onBackPressed():void
<<Java Class>>
Index
com.example.grouppeer
Index()

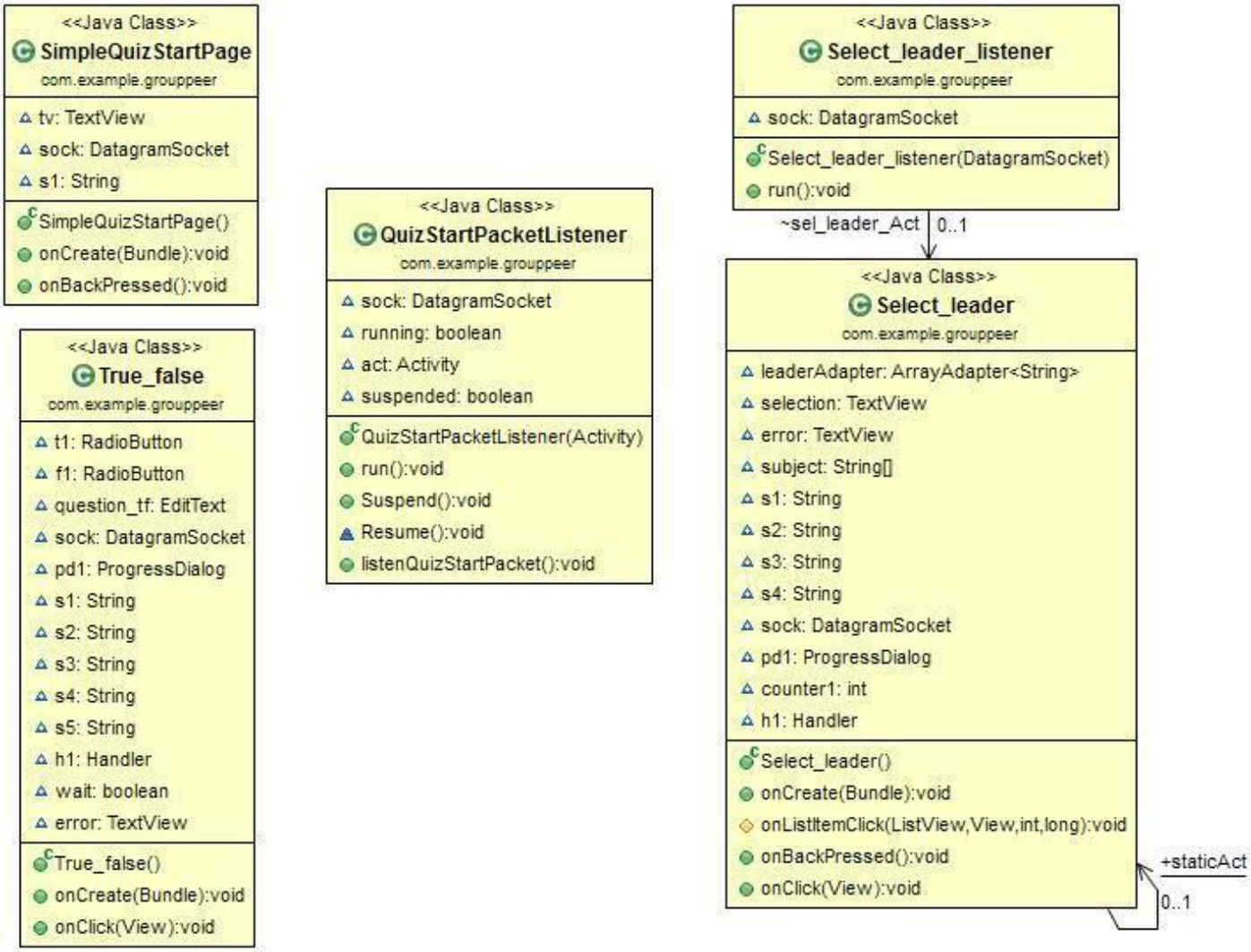

<<Java Class>>
MainActivity
com.example.grouppeer
txt: TextView
performance: String
files: String
assignment: String
changepwd: String
question: String
s1: String
s2: String
s3: String
s4: String
s5: String
sock: DatagramSocket
Performance: String[]
MainActivity()
onCreate(Bundle):void
onBackPressed():void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
openlogout():void
+staticAct
0..1
<<Java Class>>
Leader_question
com.example.grouppeer
mcq: Button
truefalse: Button
fill: Button
Leader_question()
onCreate(Bundle):void
onBackPressed():void
onClick(View):void
<<Java Class>>
QuizListen
com.example.grouppeer
sock: DatagramSocket
QuizListen()
run():void
<<Java Class>>
CleanBuffers
com.example.grouppeer
sock: DatagramSocket
CleanBuffers()
run():void
<<Java Class>>
Login
com.example.grouppeer
login: Button
passwordBox: EditText
userID: EditText
errorText: TextView
socket: DatagramSocket
uID: String
password: String
s1: String
s2: String
s3: String
s4: String
s5: String
Login()
onCreate(Bundle):void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
openHelp():void
openAboutus():void
openEnglish():void
openHindi():void
openMarathi():void
onClick(View):void
sendAuthPacket(int):void

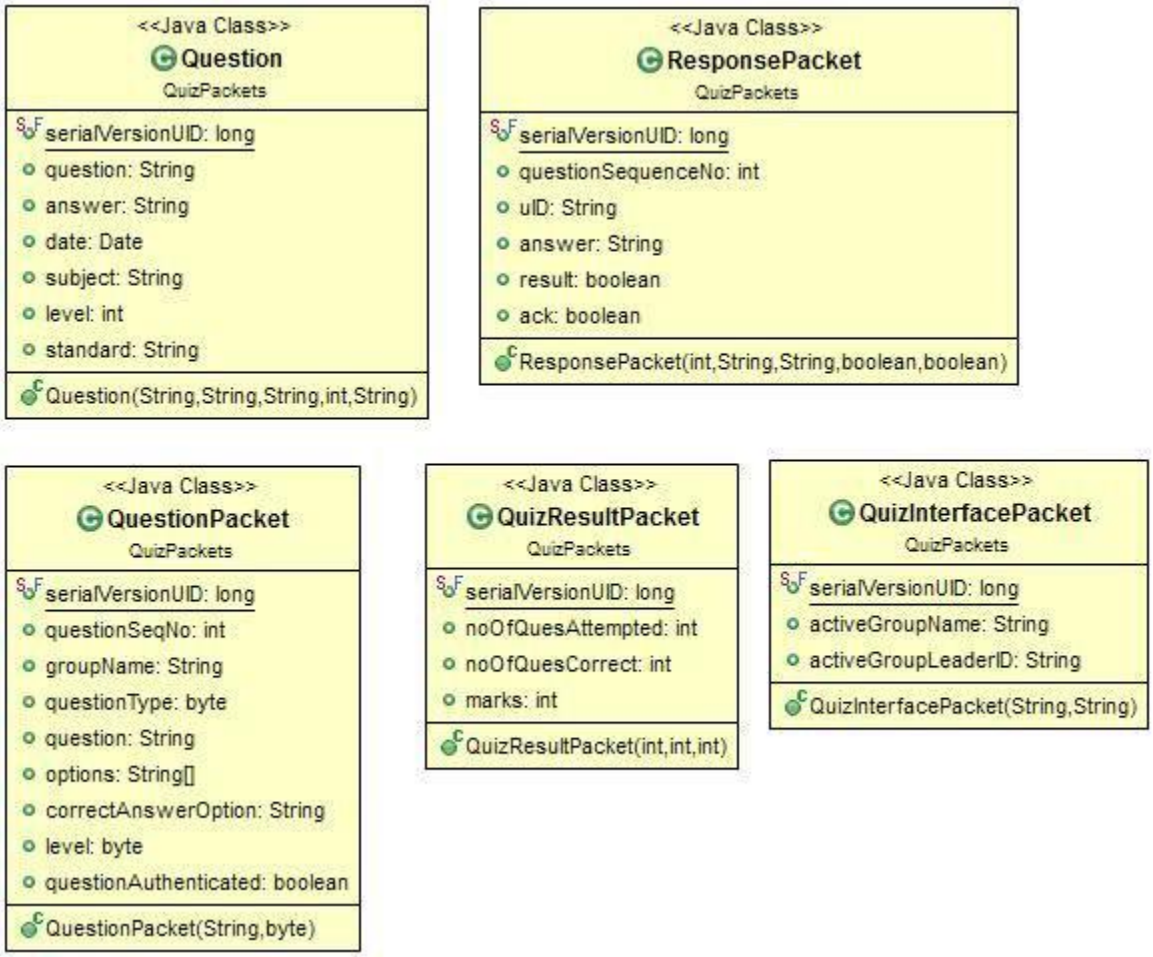

<<Java Class>>
One_word
com.example.grouppeer
submit: Button
pd1: ProgressDialog
h1: Handler
wait: boolean
sock: DatagramSocket
s1: String
s2: String
s3: String
s4: String
question: EditText
answer: EditText
One_word()
onCreate(Bundle):void
onClick(View):void
+staticVar
0..1
<<Java Class>>
MyExpandableListAdapter
com.example.grouppeer
groups: SparseArray<Group>
inflater: LayoutInflater
activity: Activity
MyExpandableListAdapter(Activity,SparseArray<Group>)
getChild(int,int)
getChildId(int,int):long
getChildView(int,int,boolean,View,ViewGroup):View
getChildrenCount(int):int
getGroup(int)
getGroupCount():int
onGroupCollapsed(int):void
onGroupExpanded(int):void
getGroupId(int):long
getGroupView(int,boolean,View,ViewGroup):View
hasStableIds():boolean
isChildSelectable(int,int):boolean
<<Java Class>>
Non_leader_question
com.example.grouppeer
Non_leader_question()
onCreate(Bundle):void
onPause():void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
<<Java Class>>
Multiple_choice
com.example.grouppeer
c1: EditText
c2: EditText
c3: EditText
c4: EditText
question: EditText
s1: String
s2: String
s3: String
s4: String
btn: Button
wait: boolean
pd1: ProgressDialog
h1: Handler
options: String[]
status: boolean[]
correctoption: String
sock: DatagramSocket
Multiple_choice()
onCreate(Bundle):void
onClick(View):void
<<Java Class>>
QuestionListener
com.example.grouppeer
sock: DatagramSocket
act: Activity
QuestionListener(Activity)
run():void

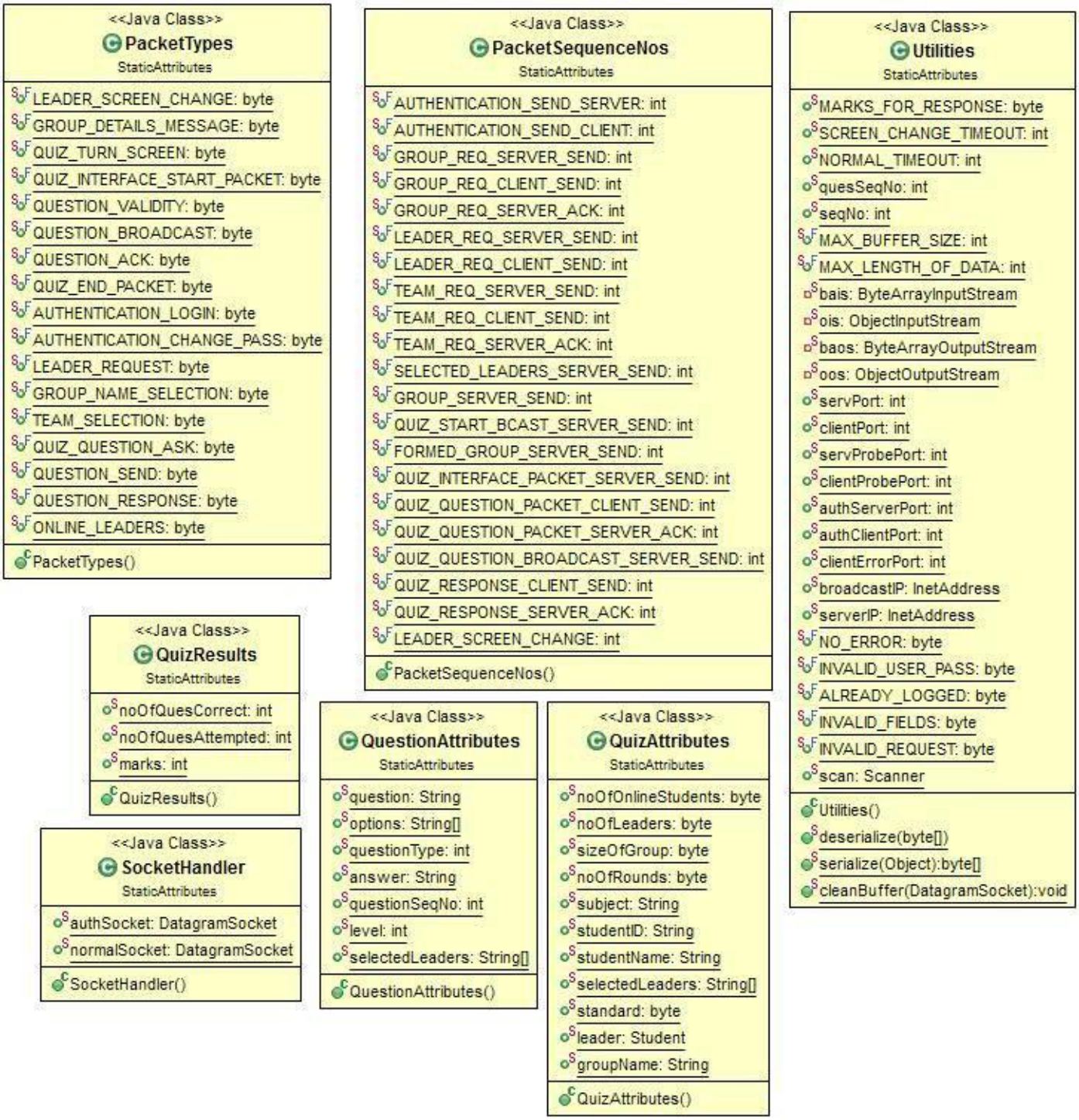

<<Java Class>>
Performance1
com.example.grouppeer
subjects: ArrayAdapter<String>
s1: String
s2: String
s3: String
s4: String
s5: String
s6: String
subject: String[]
Performance1()
onCreate(Bundle):void
onListItemClick(ListView,View,int,long):void
onClick(View):void
<<Java Class>>
Perf
com.example.grouppeer
xyPlot: XYPlot
i: int
pd1: ProgressDialog
h1: Handler
wait: boolean
tests: String[]
subjectLists: ArrayList<Map<String,Number[]>>
Perf()
onCreate(Bundle):void
plotGraph():void
<<Java Class>>
OtherGroupPage
com.example.grouppeer
tv: TextView
s1: String
OtherGroupPage()
onCreate(Bundle):void
onBackPressed():void
+staticAct
0..1
<<Java Class>>
OtherGroupListener
com.example.grouppeer
sock: DatagramSocket
OtherGroupListener()
run():void
<<Java Class>>
Questions
com.example.grouppeer
subjects: ArrayAdapter<String>
s1: String
s2: String
s3: String
s4: String
s5: String
s6: String
subject: String[]
Questions()
onCreate(Bundle):void
onListItemClick(ListView,View,int,long):void


<<Java Class>>
performanceFetchThread
com.example.grouppeer
performanceFetchThread(Performance2)
fetchHashMap(String):Map<String,String>
fetchArrayList(Map<String,String>):ArrayList<Map<String,Number[]>>
run():void
~perfObj
0..1
<<Java Class>>
Performance2
com.example.grouppeer
content: View
mImagePath: String
xyPlot: XYPlot
i: int
pd1: ProgressDialog
h1: Handler
wait: boolean
s1: String
s2: String
tests: String[]
subjectLists: ArrayList<Map<String,Number[]>>
Performance2()
onCreate(Bundle):void
plotGraph():void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
openhome():void
<<Java Class>>
Quiz
com.example.grouppeer
leader: Button
skipButton: Button
errormessage: String
groupleader: String
leadersession: String
sessionexpire: String
denyleader: String
onlinestudents: String
rounds: String
subject: String
noofleaders: String
groupmembers: String
instruction1: TextView
instruction2: TextView
instruction3: TextView
instruction4: TextView
sock: DatagramSocket
errorMsg: TextView
Quiz()
onCreate(Bundle):void
onClick(View):void
setInstructions():void
+staticAct
0..1
<<Java Class>>
QuizListen1
com.example.grouppeer
sock: DatagramSocket
QuizListen1()
run():void


<<Java Class>>
LastTestPerformanceFetchThread
com.example.grouppeer
subject: String
LastTestPerformanceFetchThread(QuizPerformance,String)
run():void
fetchHashMap(String):Map<String,String>
~qpObj
0..1
<<Java Class>>
QuizPerformance
com.example.grouppeer
pd1: ProgressDialog
h1: Handler
drawview: DrawView
wait: boolean
staticVar: QuizPerformance
s1: String
s2: String
s3: String
s4: String
s5: String
s6: String
s7: String
s8: String
s9: String
QuizPerformance()
onCreate(Bundle):void
onCreateOptionsMenu(Menu):boolean
onTouchEvent(MotionEvent):boolean
onPause():void
<<Java Class>>
quizEndPage
com.example.grouppeer
title: TextView
attempted: TextView
correct: TextView
marks: TextView
s1: String
s2: String
s3: String
s4: String
s5: String
quizEndPage()
onCreate(Bundle):void
<<Java Class>>
ResultsWaitScreen
com.example.grouppeer
act: Activity
ResultsWaitScreen(Activity)
run():void
<<Java Class>>
DrawView
com.example.grouppeer
paint: Paint
correct: int
wrong: int
attempted: int
title: String
canvas_width: float
canvas_height: float
DrawView(Context,int,int,int,String)
onDraw(Canvas):void


<<Java Class>>
SimpleQuizStartPage
com.example.grouppeer
tv: TextView
sock: DatagramSocket
s1: String
SimpleQuizStartPage()
onCreate(Bundle):void
onBackPressed():void
<<Java Class>>
True_false
com.example.grouppeer
t1: RadioButton
f1: RadioButton
question_tf: EditText
sock: DatagramSocket
pd1: ProgressDialog
s1: String
s2: String
s3: String
s4: String
s5: String
h1: Handler
wait: boolean
error: TextView
True_false()
onCreate(Bundle):void
onClick(View):void
<<Java Class>>
QuizStartPacketListener
com.example.grouppeer
sock: DatagramSocket
running: boolean
act: Activity
suspended: boolean
QuizStartPacketListener(Activity)
run():void
Suspend():void
Resume():void
listenQuizStartPacket():void
<<Java Class>>
Select_leader_listener
com.example.grouppeer
sock: DatagramSocket
Select_leader_listener(DatagramSocket)
run():void
~sel_leader_Act
0..1
<<Java Class>>
Select_leader
com.example.grouppeer
leaderAdapter: ArrayAdapter<String>
selection: TextView
error: TextView
subject: String[]
s1: String
s2: String
s3: String
s4: String
sock: DatagramSocket
pd1: ProgressDialog
counter1: int
h1: Handler
Select_leader()
onCreate(Bundle):void
onListItemClick(ListView,View,int,long):void
onBackPressed():void
onClick(View):void
+staticAct
0..1


<<Java Class>>
Question
QuizPackets
serialVersionUID: long
question: String
answer: String
date: Date
subject: String
level: int
standard: String
Question(String,String,String,int,String)
<<Java Class>>
ResponsePacket
QuizPackets
serialVersionUID: long
questionSequenceNo: int
uID: String
answer: String
result: boolean
ack: boolean
ResponsePacket(int,String,String,boolean,boolean)
<<Java Class>>
QuestionPacket
QuizPackets
serialVersionUID: long
questionSeqNo: int
groupName: String
questionType: byte
question: String
options: String[]
correctAnswerOption: String
level: byte
questionAuthenticated: boolean
QuestionPacket(String,byte)
<<Java Class>>
QuizResultPacket
QuizPackets
serialVersionUID: long
noOfQuesAttempted: int
noOfQuesCorrect: int
marks: int
QuizResultPacket(int,int,int)
<<Java Class>>
QuizInterfacePacket
QuizPackets
serialVersionUID: long
activeGroupName: String
activeGroupLeaderID: String
QuizInterfacePacket(String,String)


<<Java Class>>
PacketTypes
StaticAttributes
LEADER_SCREEN_CHANGE: byte
GROUP_DETAILS_MESSAGE: byte
QUIZ_TURN_SCREEN: byte
QUIZ_INTERFACE_START_PACKET: byte
QUESTION_VALIDITY: byte
QUESTION_BROADCAST: byte
QUESTION_ACK: byte
QUIZ_END_PACKET: byte
AUTHENTICATION_LOGIN: byte
AUTHENTICATION_CHANGE_PASS: byte
LEADER_REQUEST: byte
GROUP_NAME_SELECTION: byte
TEAM_SELECTION: byte
QUIZ_QUESTION_ASK: byte
QUESTION_SEND: byte
QUESTION_RESPONSE: byte
ONLINE_LEADERS: byte
PacketTypes()
<<Java Class>>
PacketSequenceNos
StaticAttributes
AUTHENTICATION_SEND_SERVER: int
AUTHENTICATION_SEND_CLIENT: int
GROUP_REQ_SERVER_SEND: int
GROUP_REQ_CLIENT_SEND: int
GROUP_REQ_SERVER_ACK: int
LEADER_REQ_SERVER_SEND: int
LEADER_REQ_CLIENT_SEND: int
TEAM_REQ_SERVER_SEND: int
TEAM_REQ_CLIENT_SEND: int
TEAM_REQ_SERVER_ACK: int
SELECTED_LEADERS_SERVER_SEND: int
GROUP_SERVER_SEND: int
QUIZ_START_BCAST_SERVER_SEND: int
FORMED_GROUP_SERVER_SEND: int
QUIZ_INTERFACE_PACKET_SERVER_SEND: int
QUIZ_QUESTION_PACKET_CLIENT_SEND: int
QUIZ_QUESTION_PACKET_SERVER_ACK: int
QUIZ_QUESTION_BROADCAST_SERVER_SEND: int
QUIZ_RESPONSE_CLIENT_SEND: int
QUIZ_RESPONSE_SERVER_ACK: int
LEADER_SCREEN_CHANGE: int
PacketSequenceNos()
<<Java Class>>
Utilities
StaticAttributes
MARKS_FOR_RESPONSE: byte
SCREEN_CHANGE_TIMEOUT: int
NORMAL_TIMEOUT: int
quesSeqNo: int
seqNo: int
MAX_BUFFER_SIZE: int
MAX_LENGTH_OF_DATA: int
bais: ByteArrayInputStream
ois: ObjectInputStream
baos: ByteArrayOutputStream
oos: ObjectOutputStream
servPort: int
clientPort: int
servProbePort: int
clientProbePort: int
authServerPort: int
authClientPort: int
clientErrorPort: int
broadcastIP: InetAddress
serverIP: InetAddress
NO_ERROR: byte
INVALID_USER_PASS: byte
ALREADY_LOGGED: byte
INVALID_FIELDS: byte
INVALID_REQUEST: byte
scan: Scanner
Utilities()
deserialize(byte[])
serialize(Object):byte[]
cleanBuffer(DatagramSocket):void
<<Java Class>>
QuizResults
StaticAttributes
noOfQuesCorrect: int
noOfQuesAttempted: int
marks: int
QuizResults()
<<Java Class>>
QuestionAttributes
StaticAttributes
question: String
options: String[]
questionType: int
answer: String
questionSeqNo: int
level: int
selectedLeaders: String[]
QuestionAttributes()
<<Java Class>>
QuizAttributes
StaticAttributes
noOfOnlineStudents: byte
noOfLeaders: byte
sizeOfGroup: byte
noOfRounds: byte
subject: String
studentID: String
studentName: String
selectedLeaders: String[]
standard: byte
leader: Student
groupName: String
QuizAttributes()
<<Java Class>>
SocketHandler
StaticAttributes
authSocket: DatagramSocket
normalSocket: DatagramSocket
SocketHandler()

Data Flow Diagram

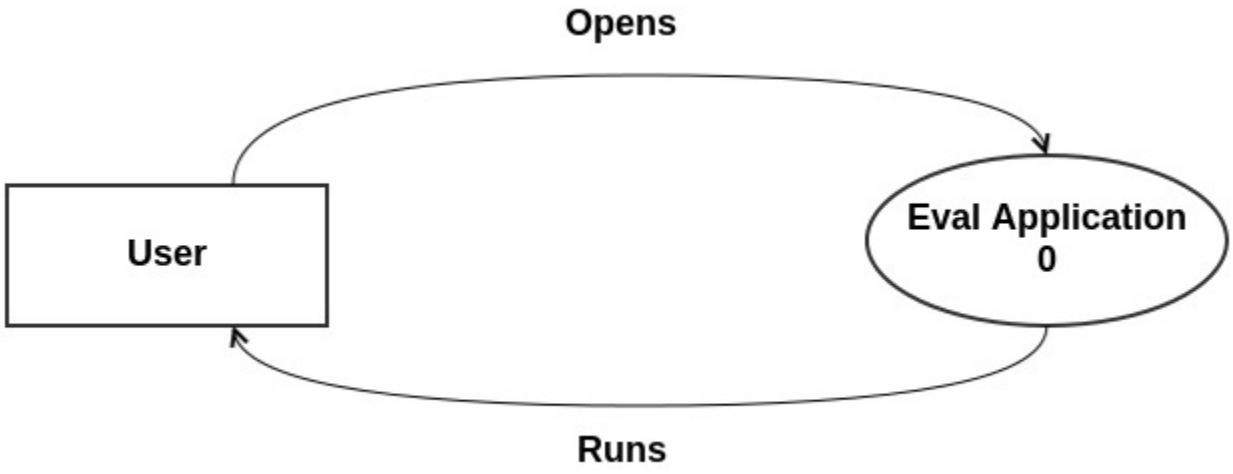


Figure 5.13: Level 0 DFD

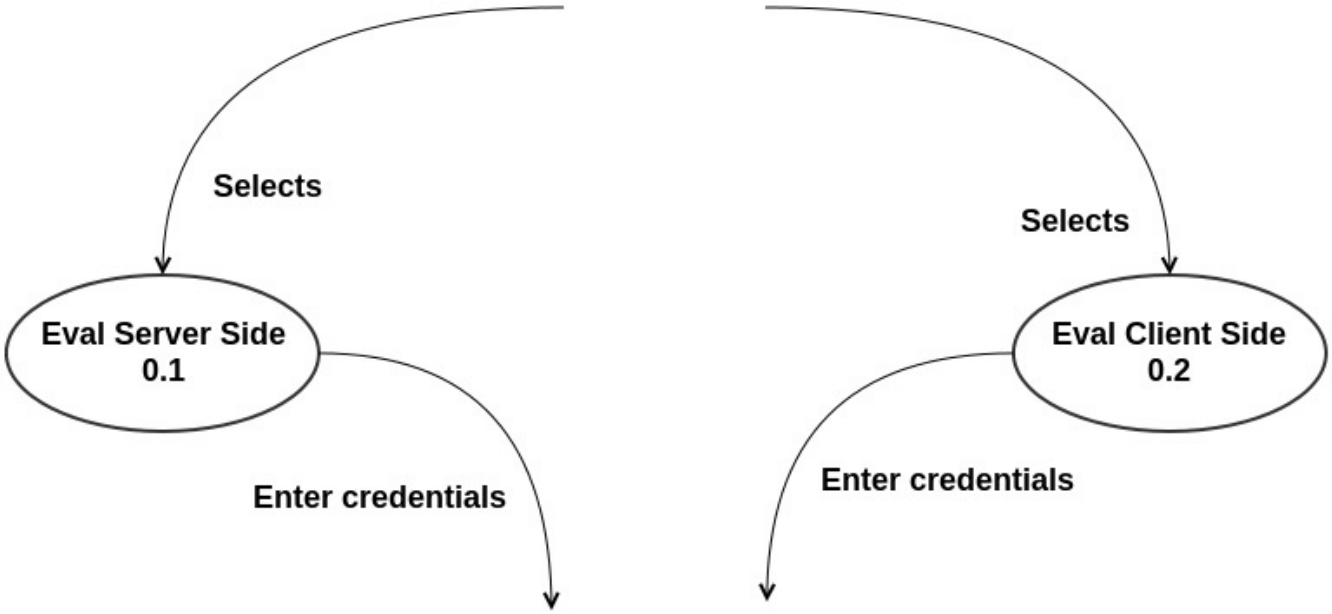


Figure 5.14: Level 1 DFD

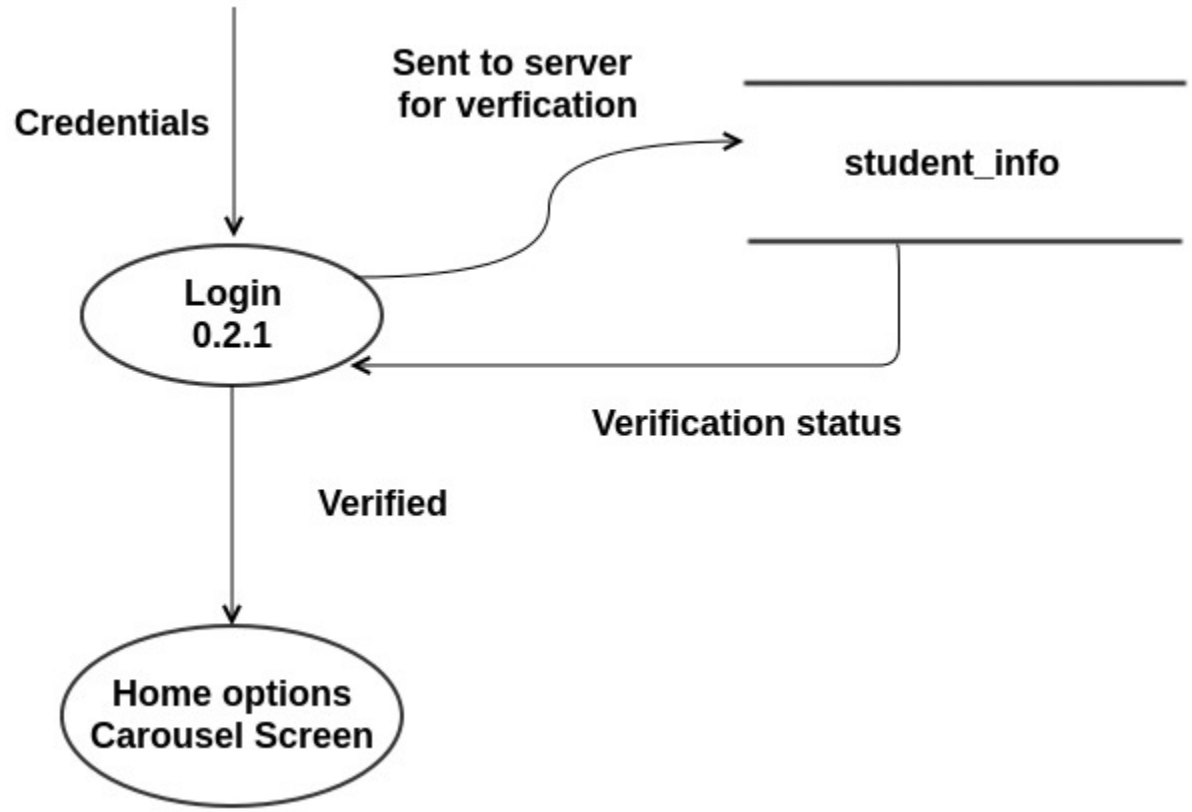


Figure 5.15: Level 2 DFD, Client

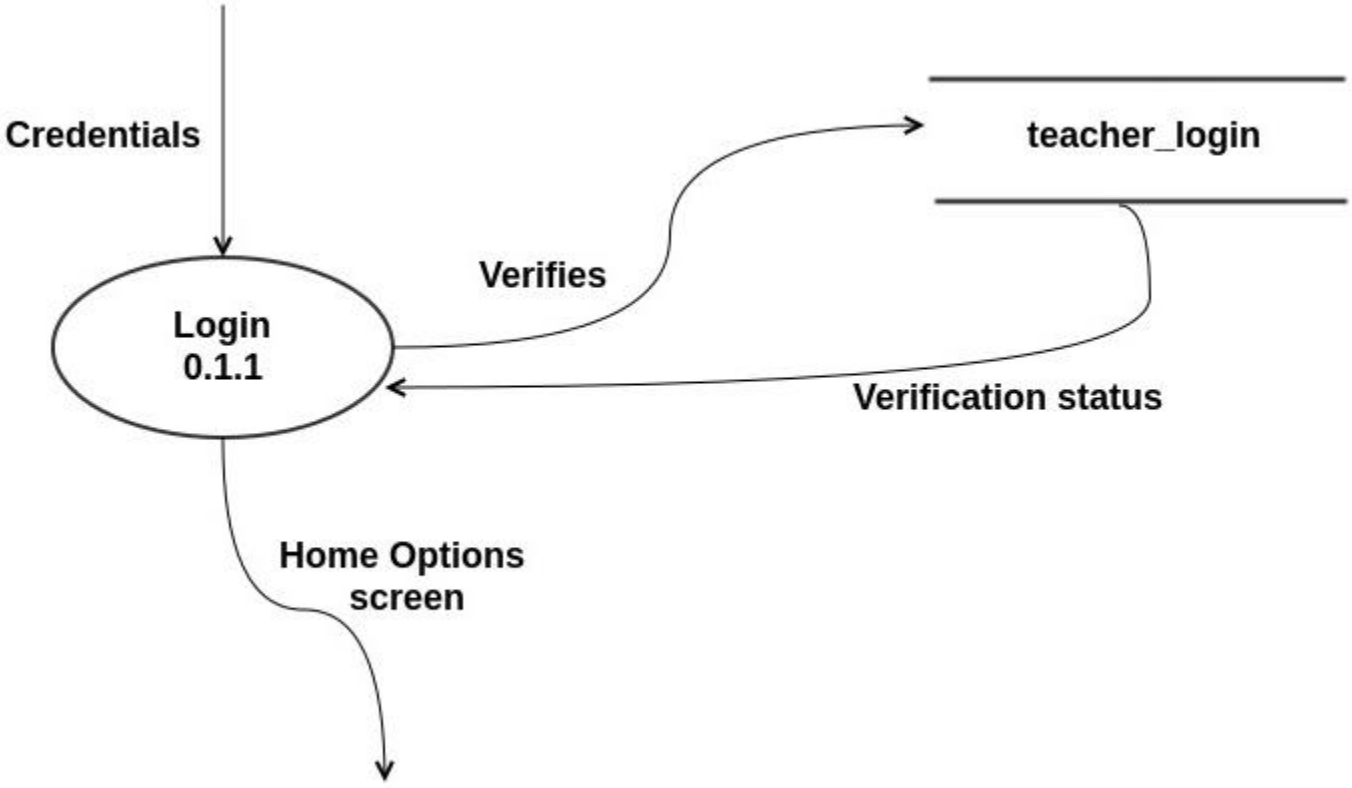


Figure 5.16: Level 2 DFD, Server

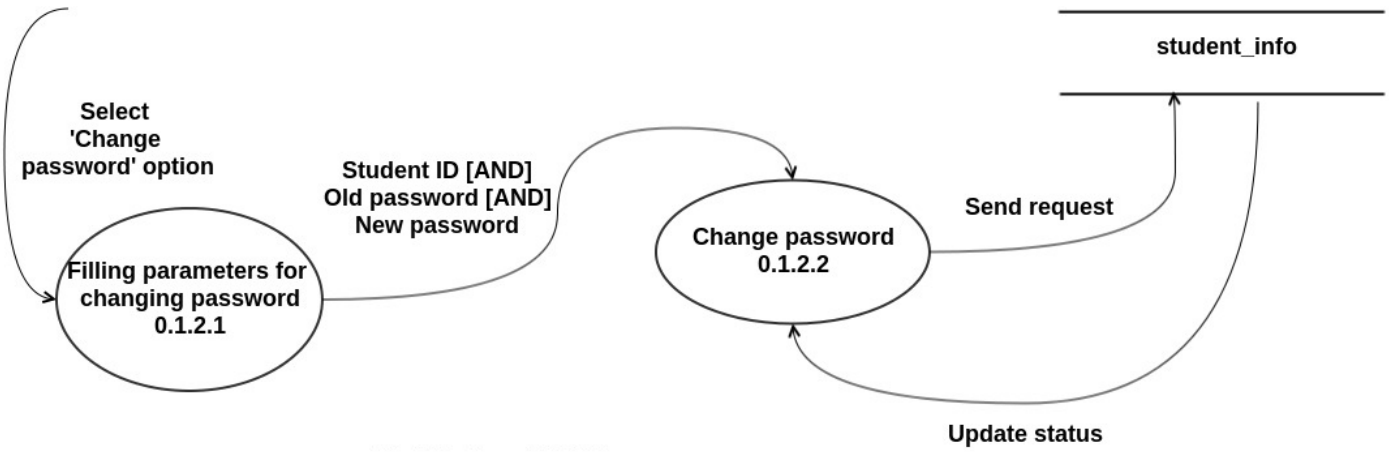


Figure5.17: Level 3 DFD, Client

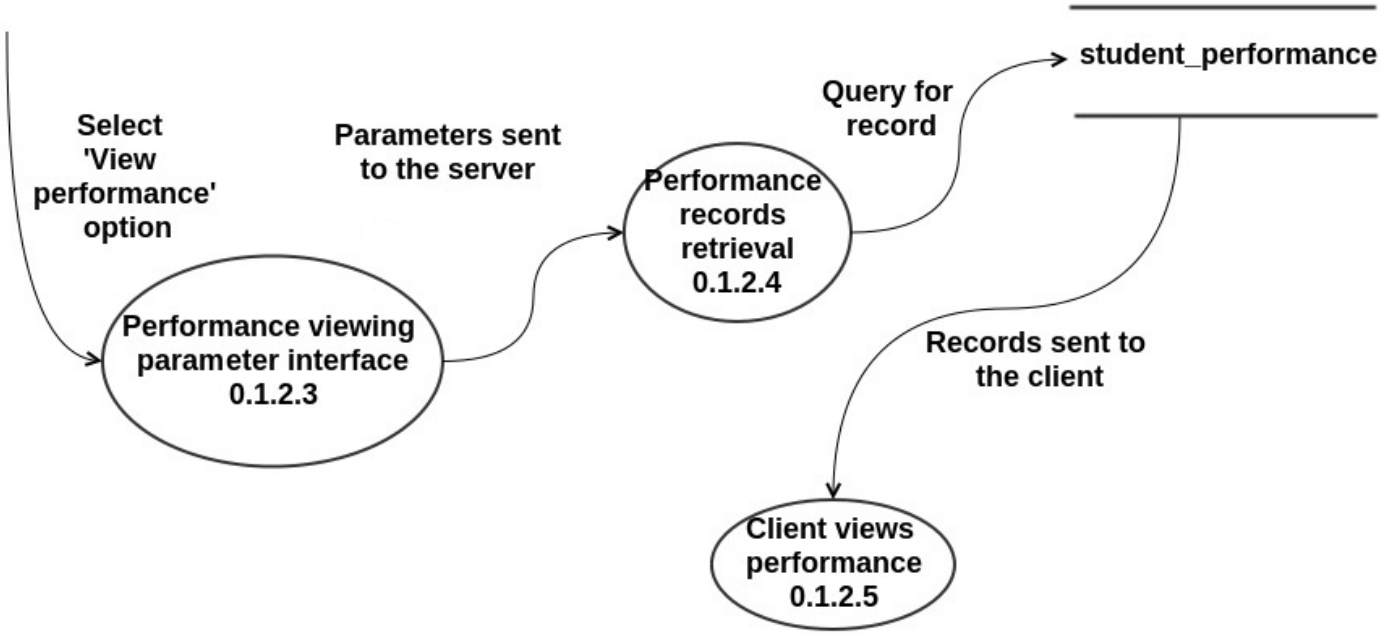


Figure 5.18: Level 3 DFD, Client

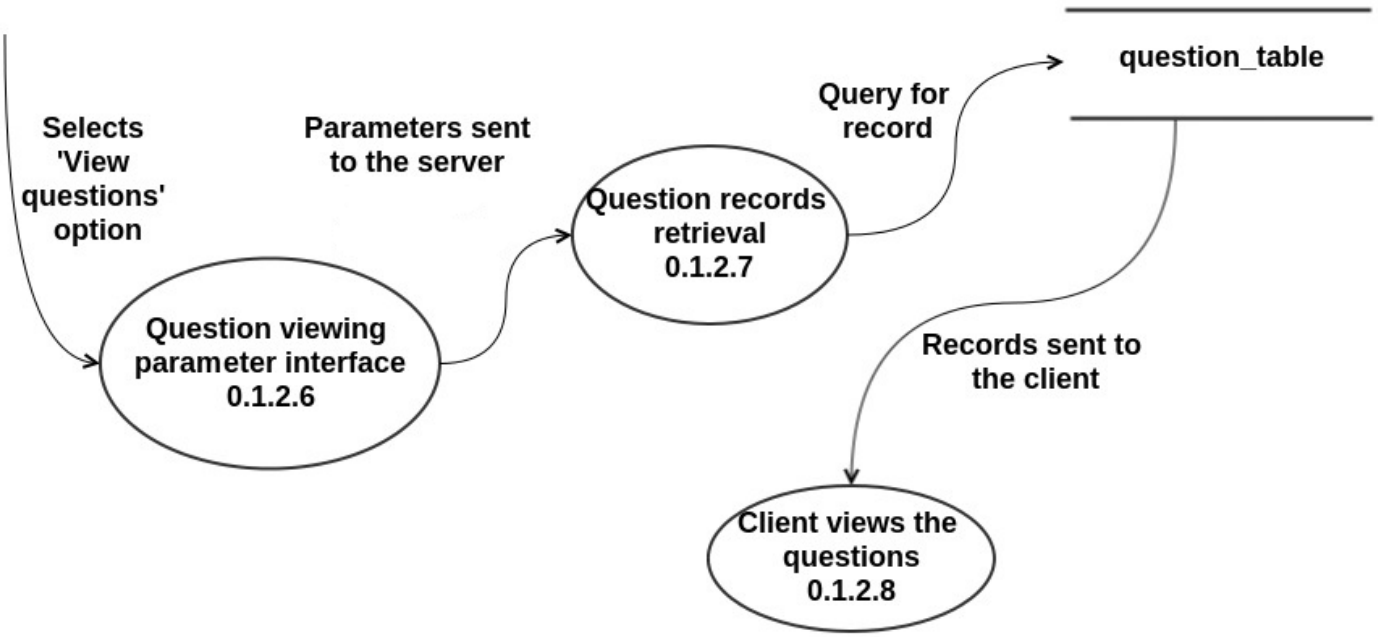


Figure 5.19: Level 3 DFD, Client

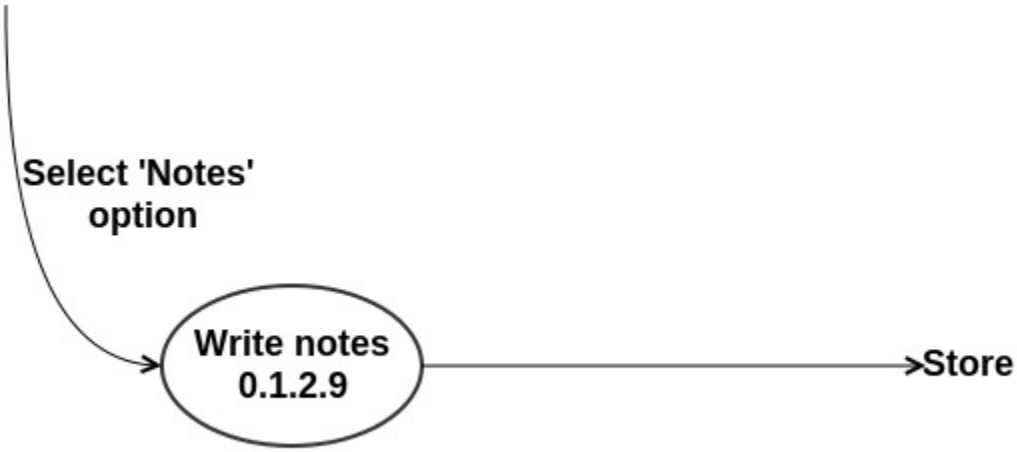


Figure 5.20: Level 3 DFD, Client

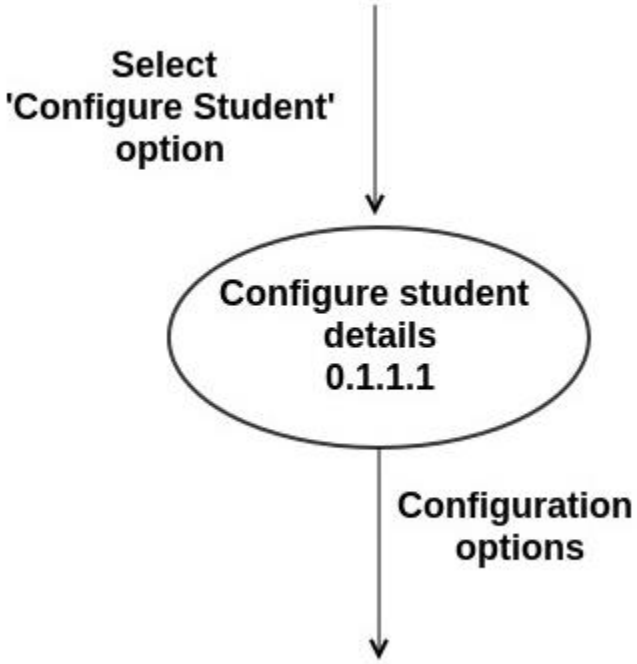


Figure 5.21: Level 3 DFD, Server

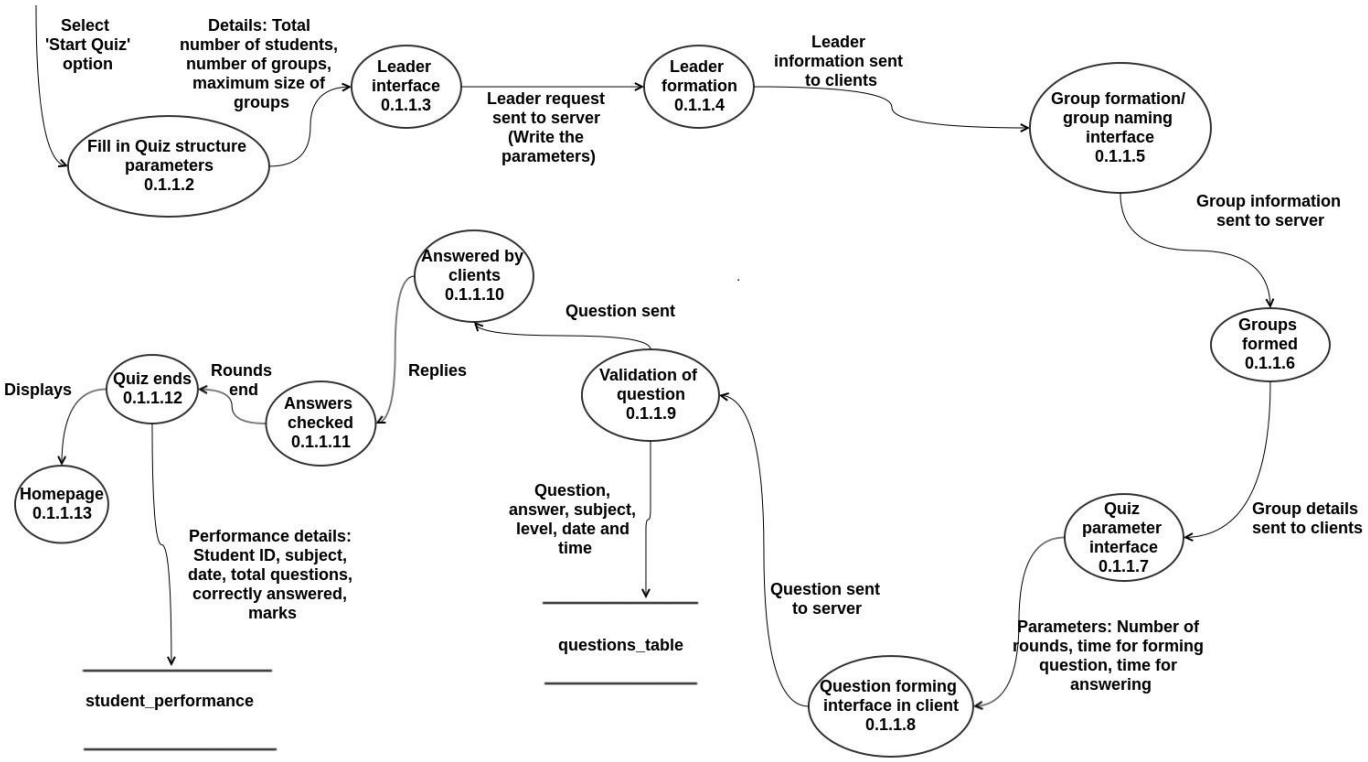


Figure5.22: Level 3 DFD, Server

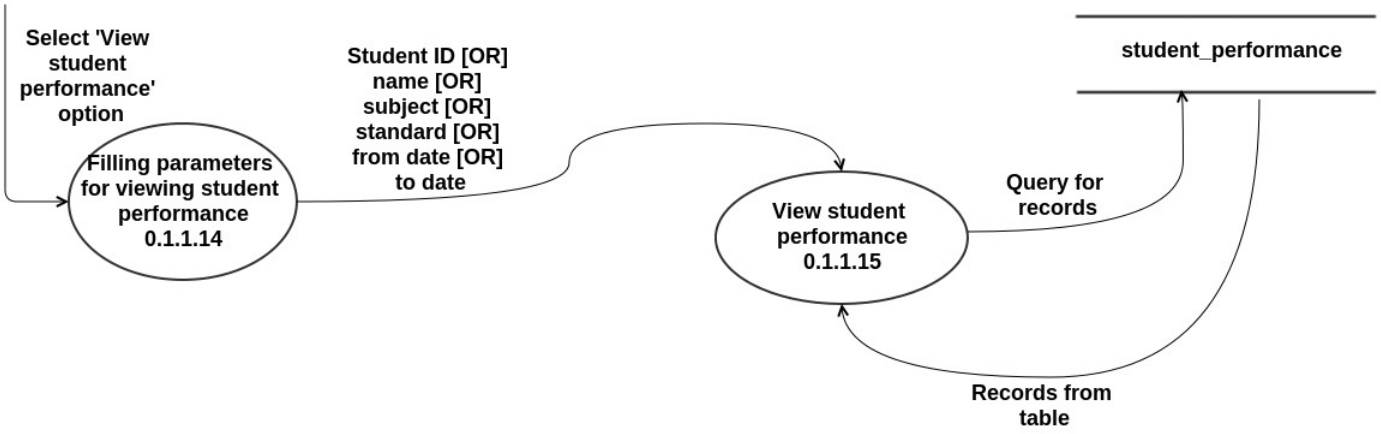


Figure 5.23: Level 3 DFD, Server

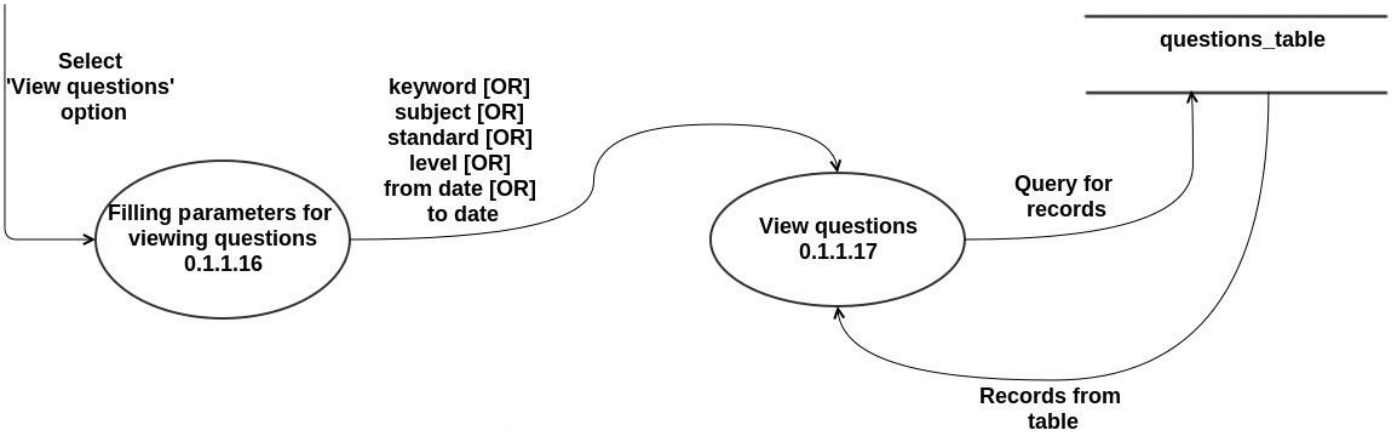


Figure 5.24: Level 3 DFD, Server

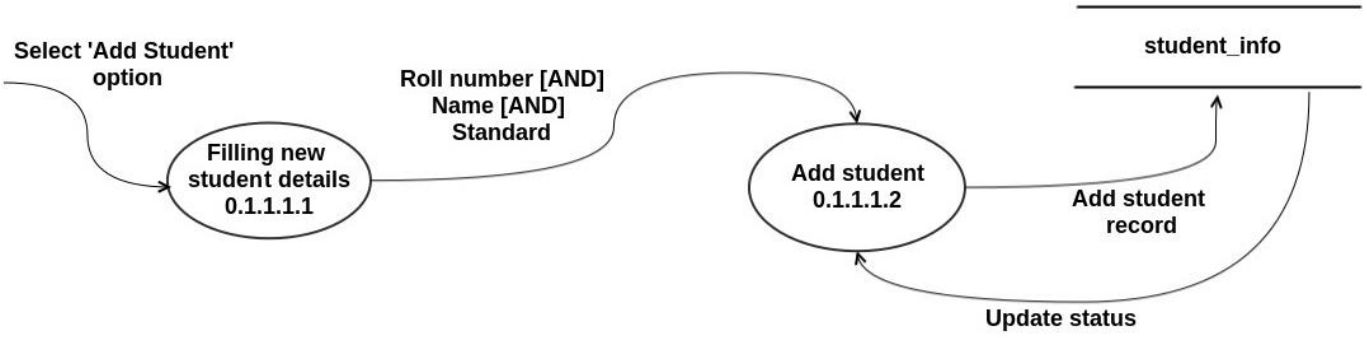


Figure 5.25: Level 4 DFD, Server

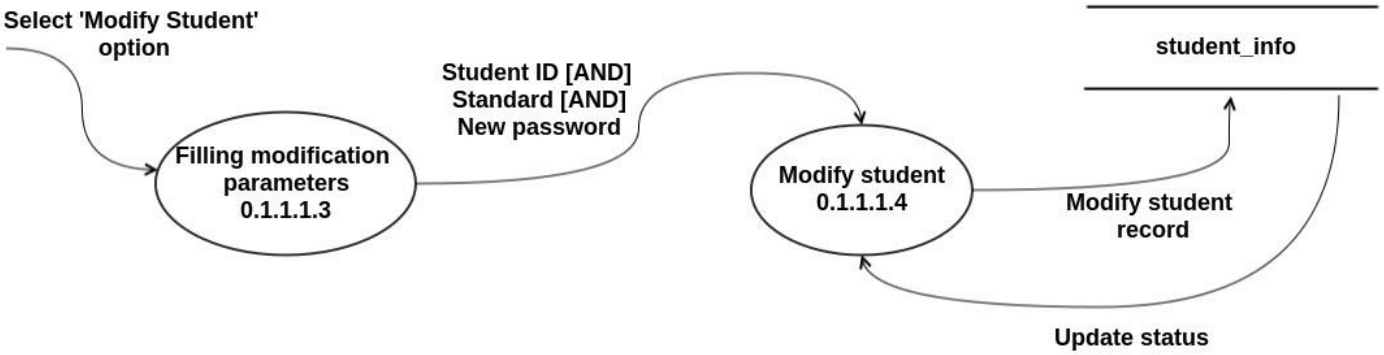


Figure 5.26: Level 4 DFD, Server

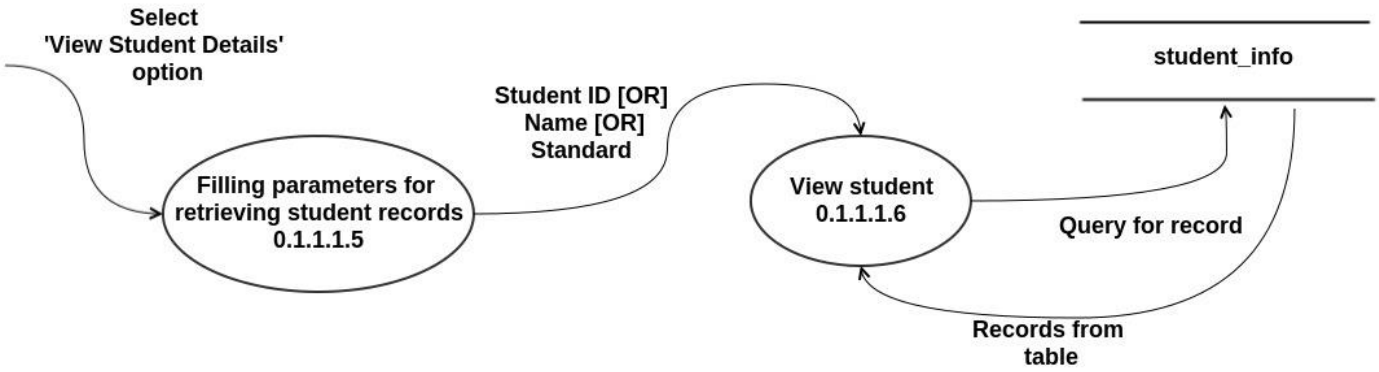


Figure 5.27: Level 4 DFD, Server

Sequence Diagram

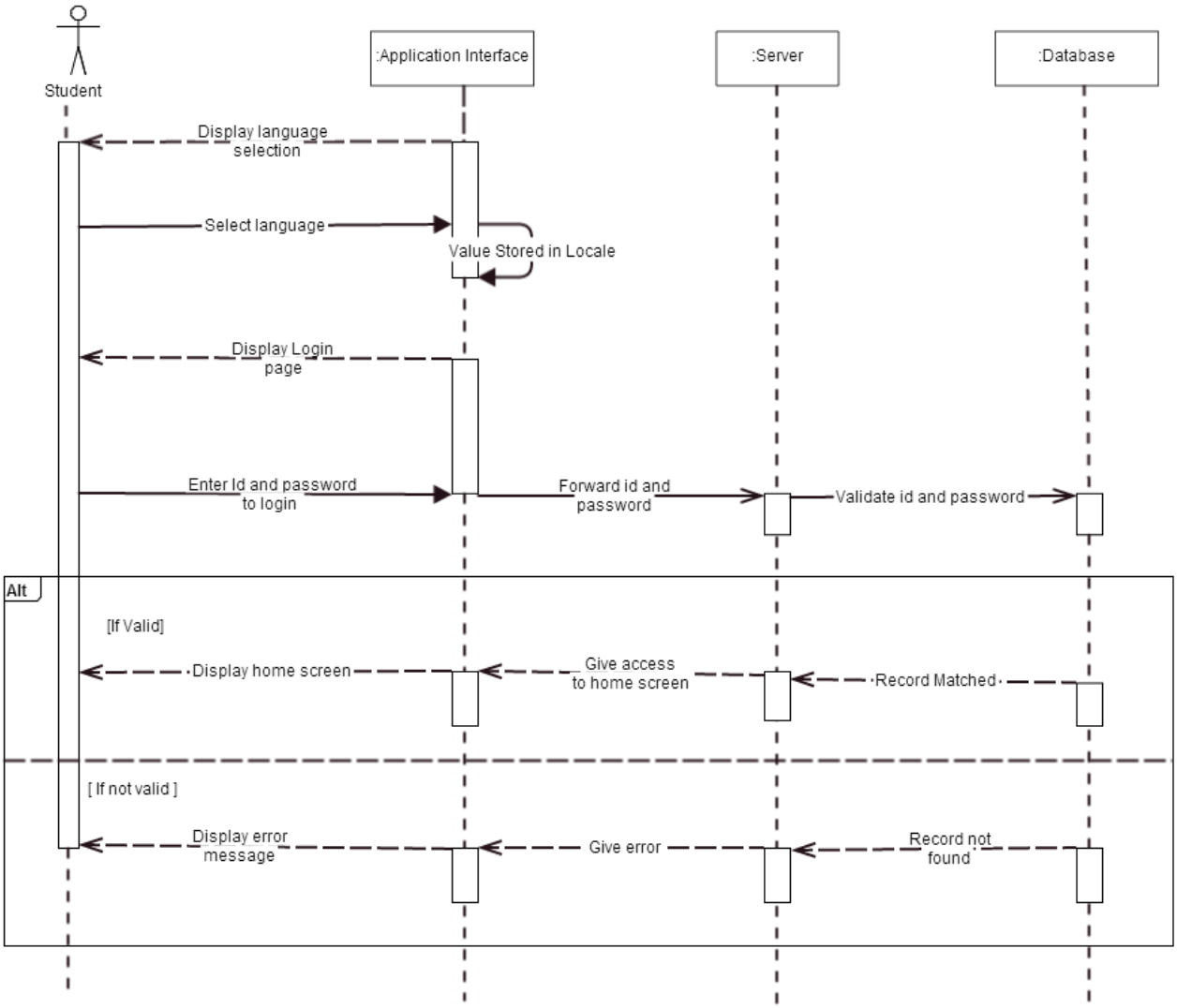


Figure 5.28: Sequence diagram for student login

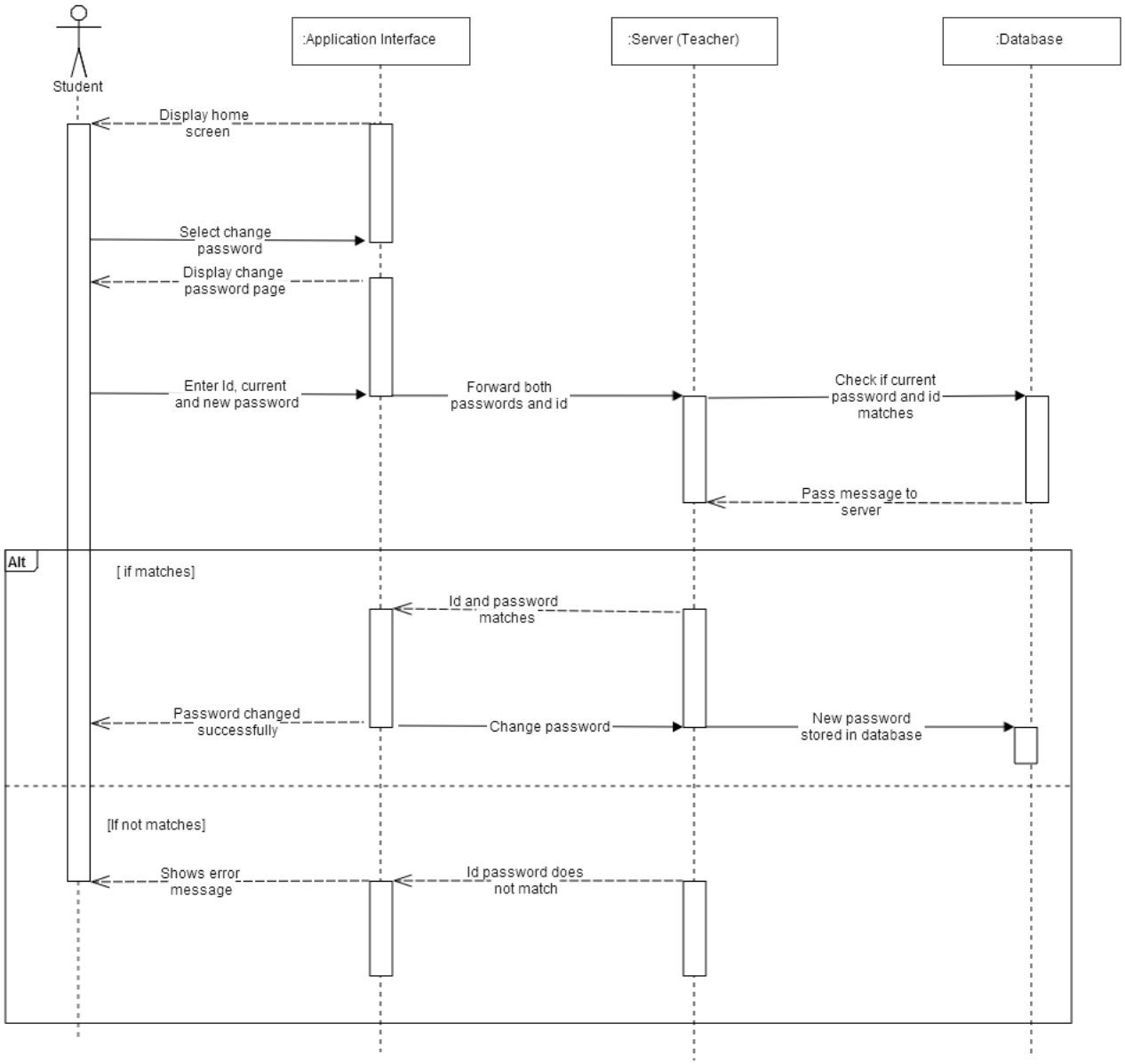


Figure 5.28: Sequence diagram for student - Change password

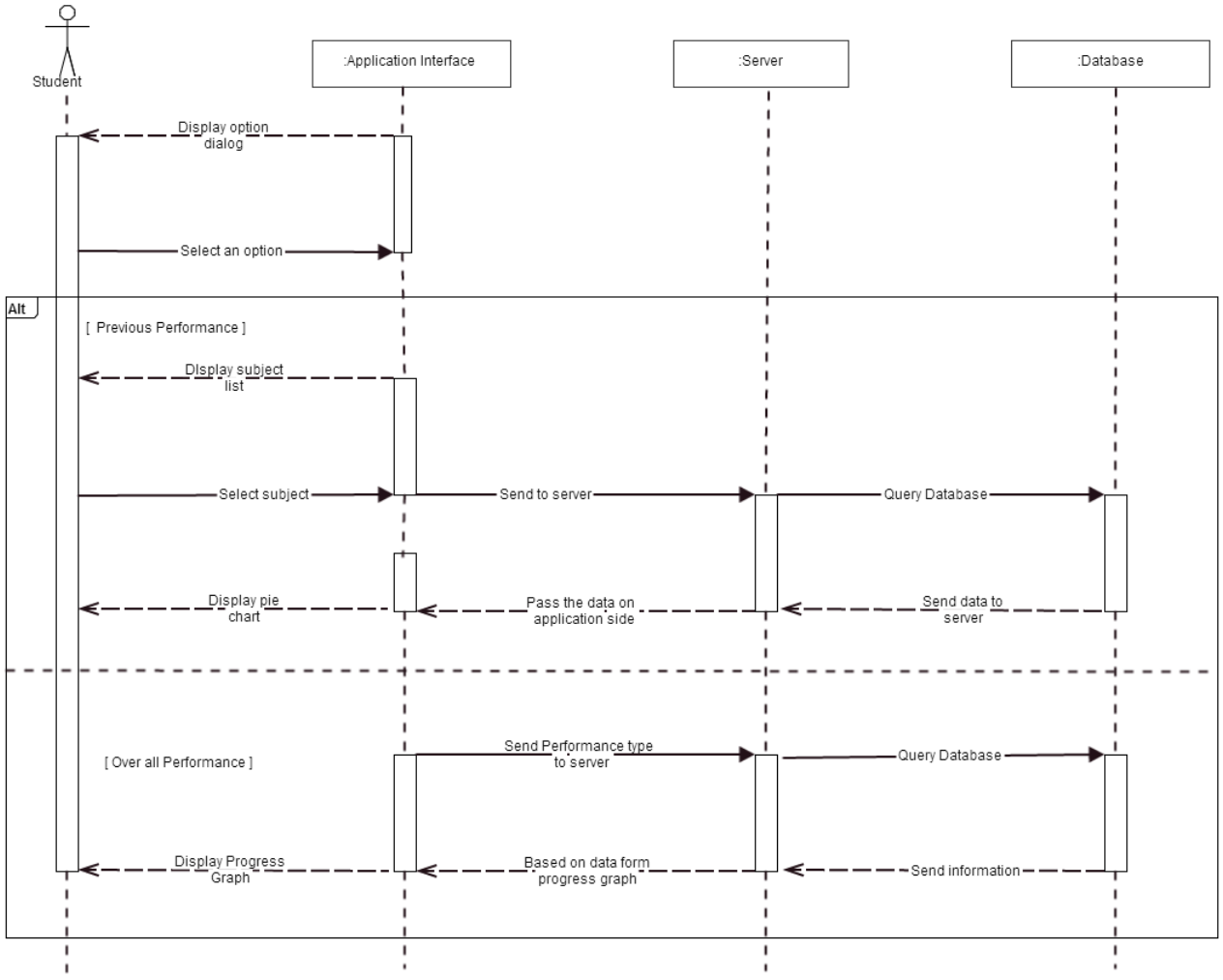


Figure 5.29: Sequence diagram for student - Performance

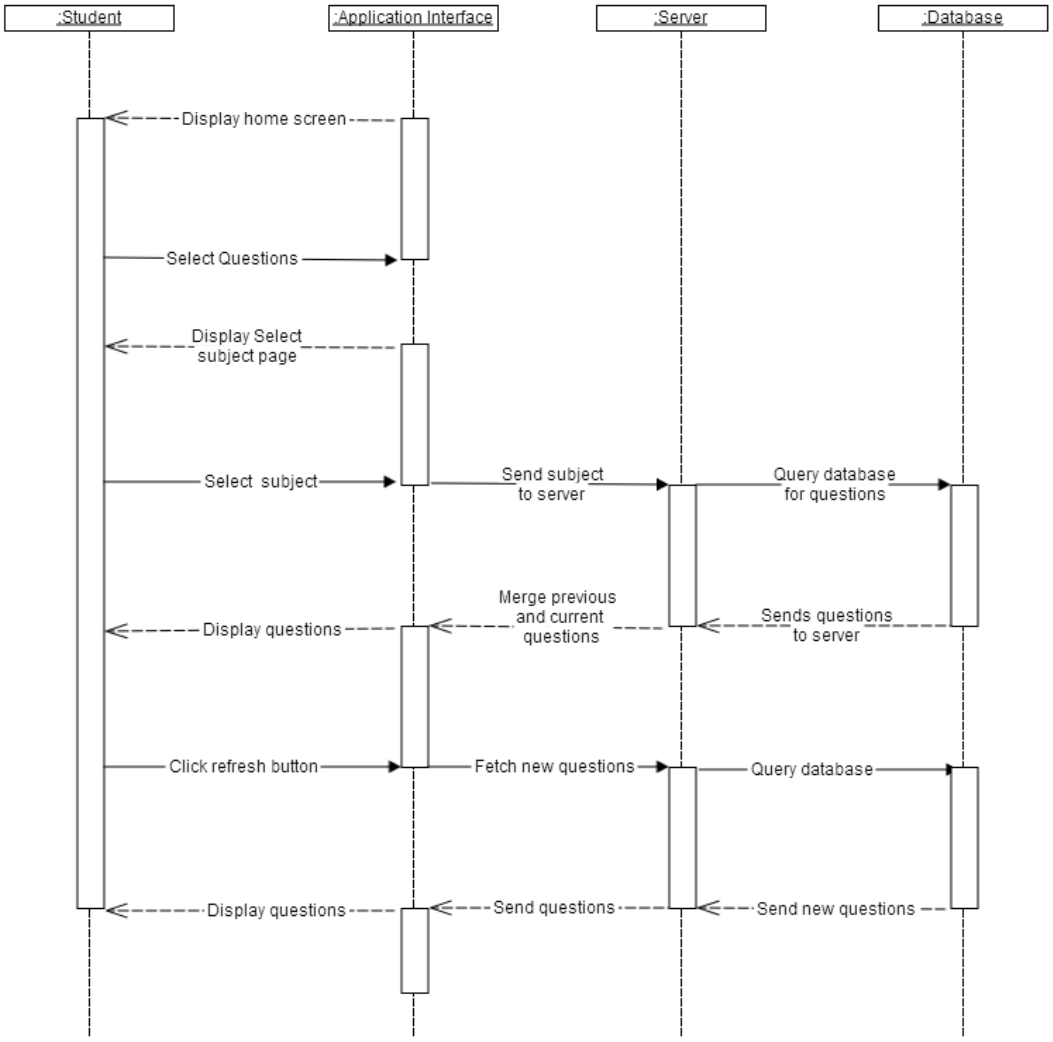


Figure 5.30: Sequence diagram for student - Questions


Student
:Application Interface
:Server
:Database
Display Home page
View Options
Read Instructions
Server Broadcast Quiz Page
Opt for leader or press skip button
Information passed to server
Alt
[Leader]
Views Group name page
Broadcast Group-name page
Give group name
pass name to server
[Non-Leader]
View all leaders
Broadcast select leader page
select a leader
forward leader name to server
View Team details
Broadcast team details page
View Message "Wait until Quiz starts"
Broadcast wait for quiz page
Alt
[If Leader]
View three options (multiple choice, one word and true false)
Broadcast Question Interface page
Select option and type the question
Send question to server
[If not a Leader]
View message "wait until leader form question
Broadcast page wait till leader forms question
[If member of other group]
View message "Be ready you will receive a question"
Broadcast page saying be ready for quiz
Leader give question
Passed to server
Alt
[If valid Question]
Give Level to Question
Store the question in database
View question
Broadcast question to all other groups
Stored Successfully
[If not valid]
Leader Views question interface
Broadcast question interface to leader
Form question again
Send for validation
Answer questions
forward answer
Submitted successfully
Answer submitted successfully
Check if no. of rounds completed
Alt
[If yes]
Quiz complete
Store performance in database
View performance
Send results
[If no]
Next group gets chance to ask question
View home Screen
Quiz over, broadcast home screen

Figure 5.31: Sequence diagram for student - Quiz

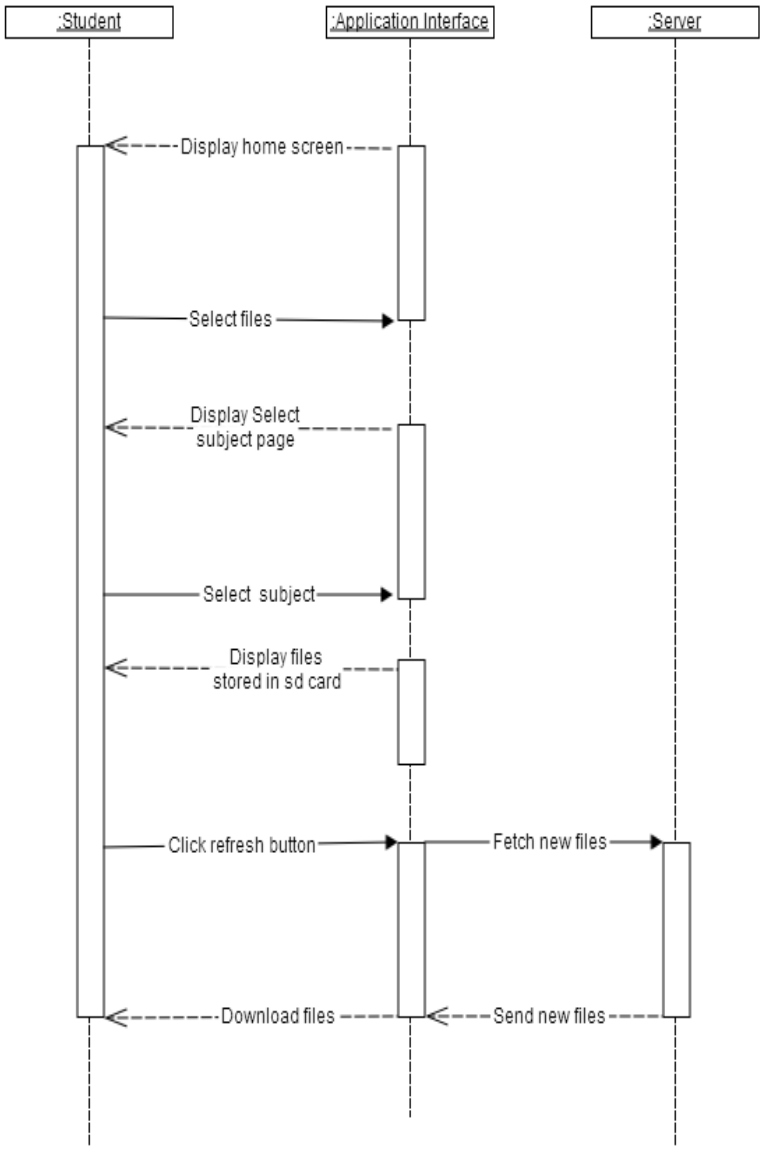


Figure 5.32: Sequence diagram for student - Files

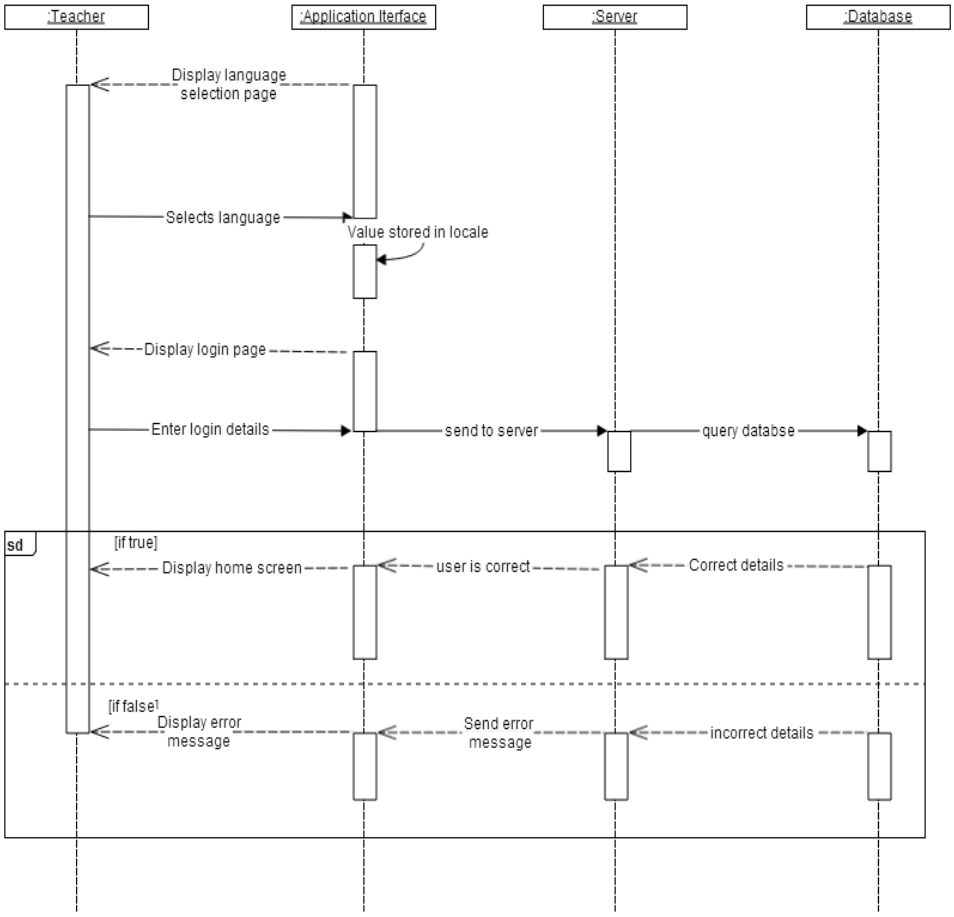


Figure 5.33: Sequence diagram for teacher - Login

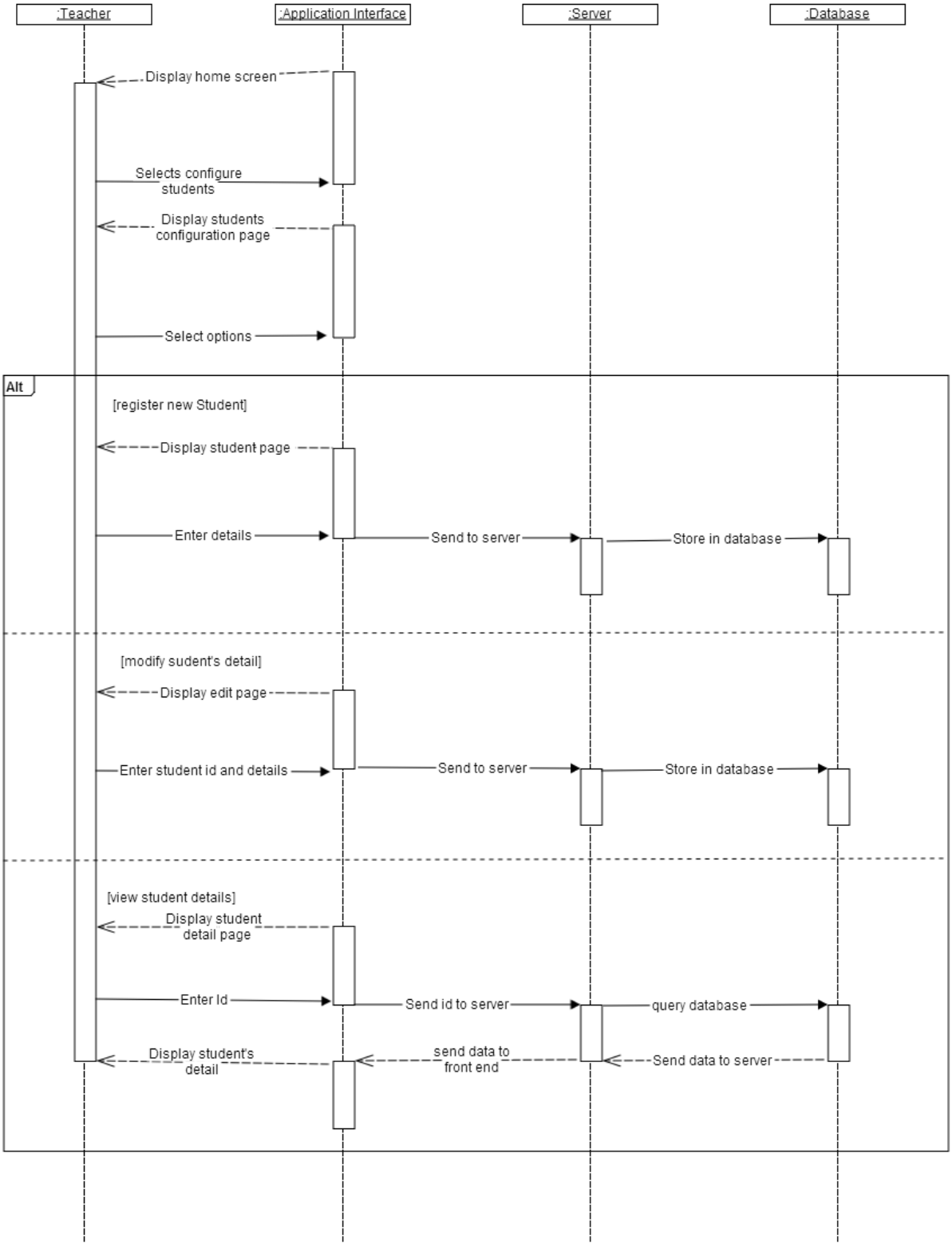


Figure 5.34: Sequence diagram for teacher - Configure Students

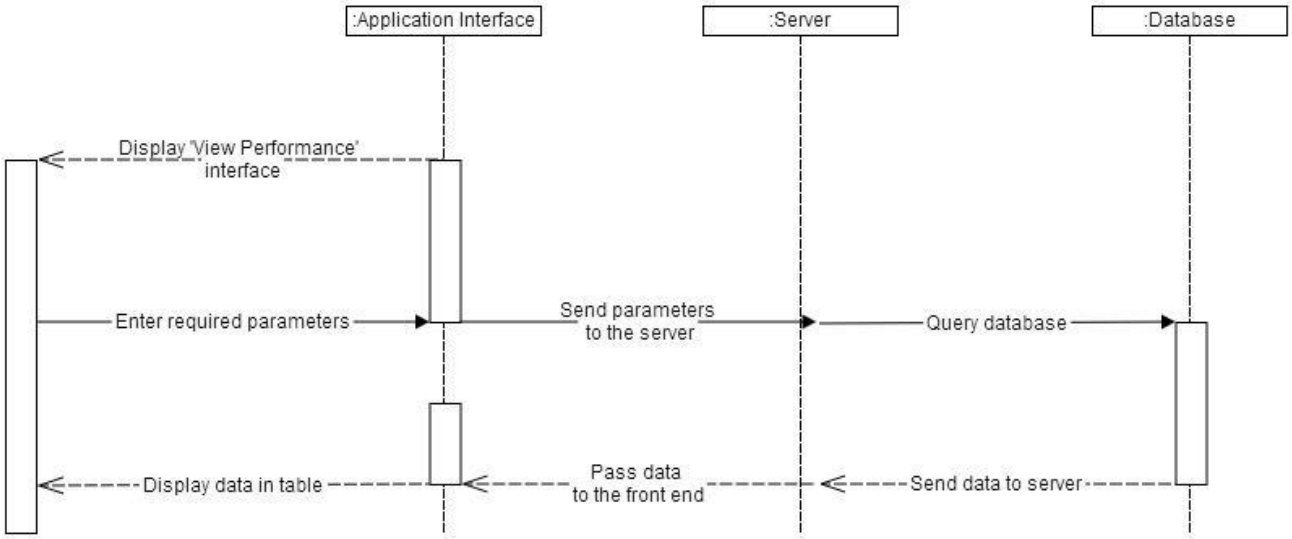


Figure 5.35: Sequence diagram for teacher - Performance

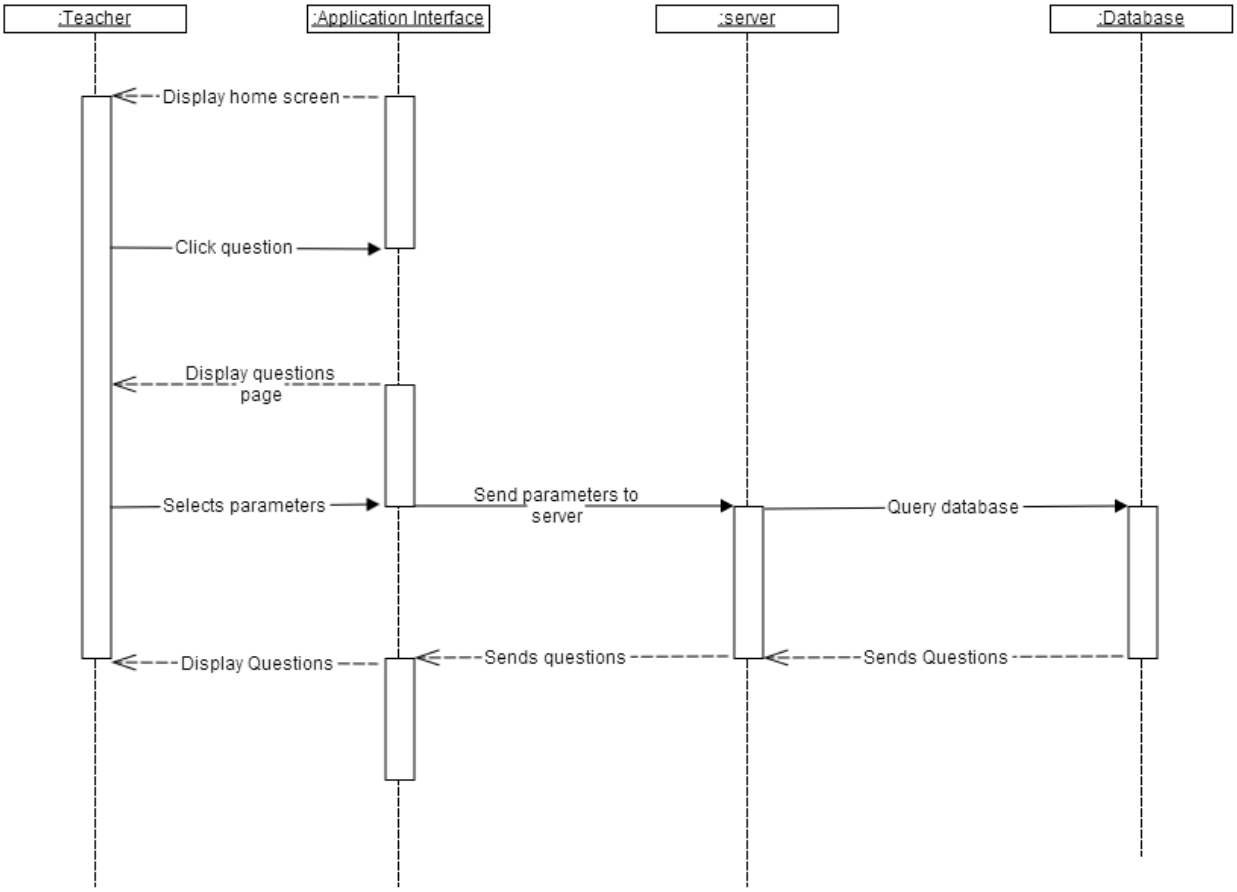


Figure 5.36: Sequence diagram for teacher - Questions

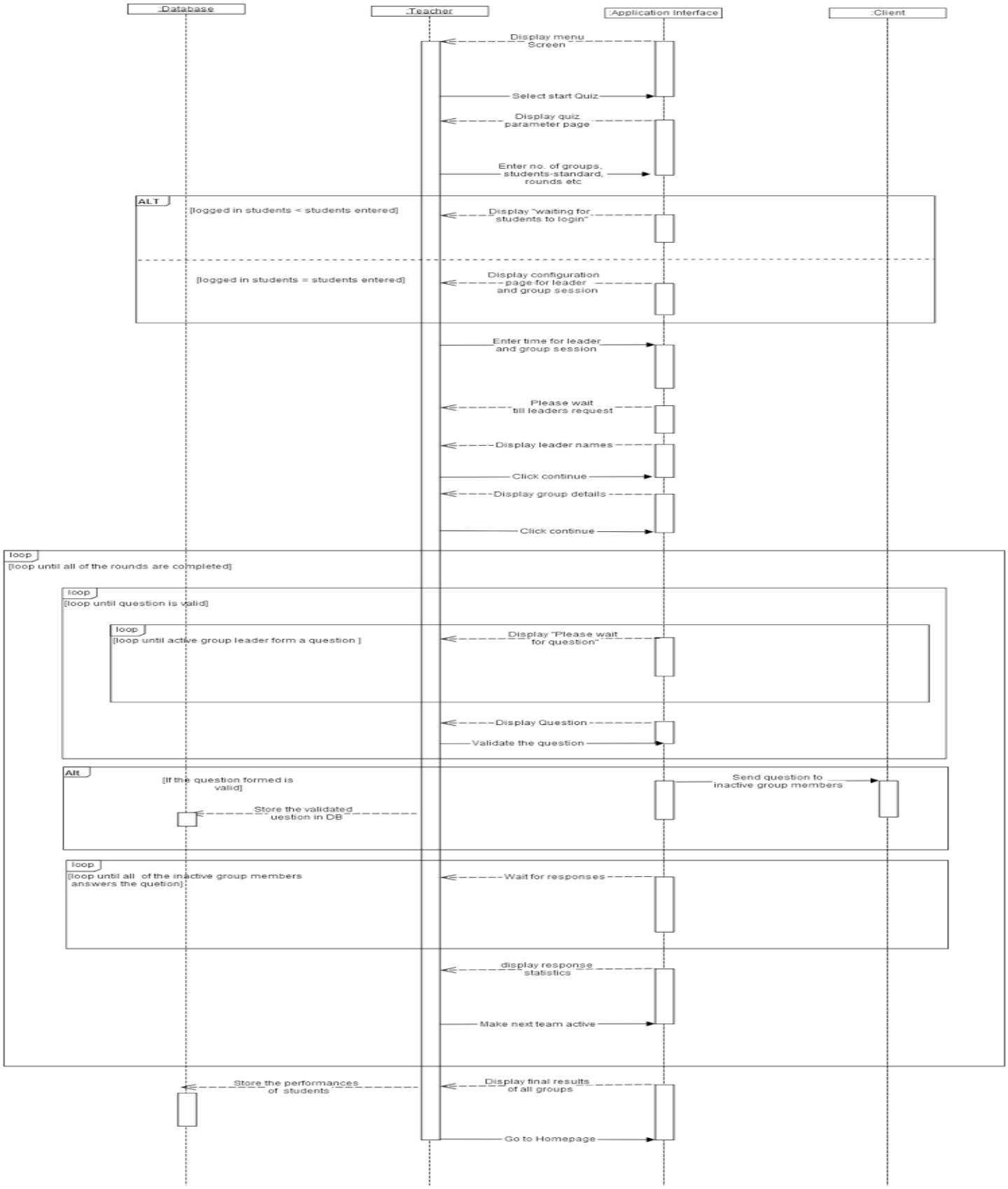


Figure 5.37: Sequence diagram for teacher - Quiz

E-R Diagram

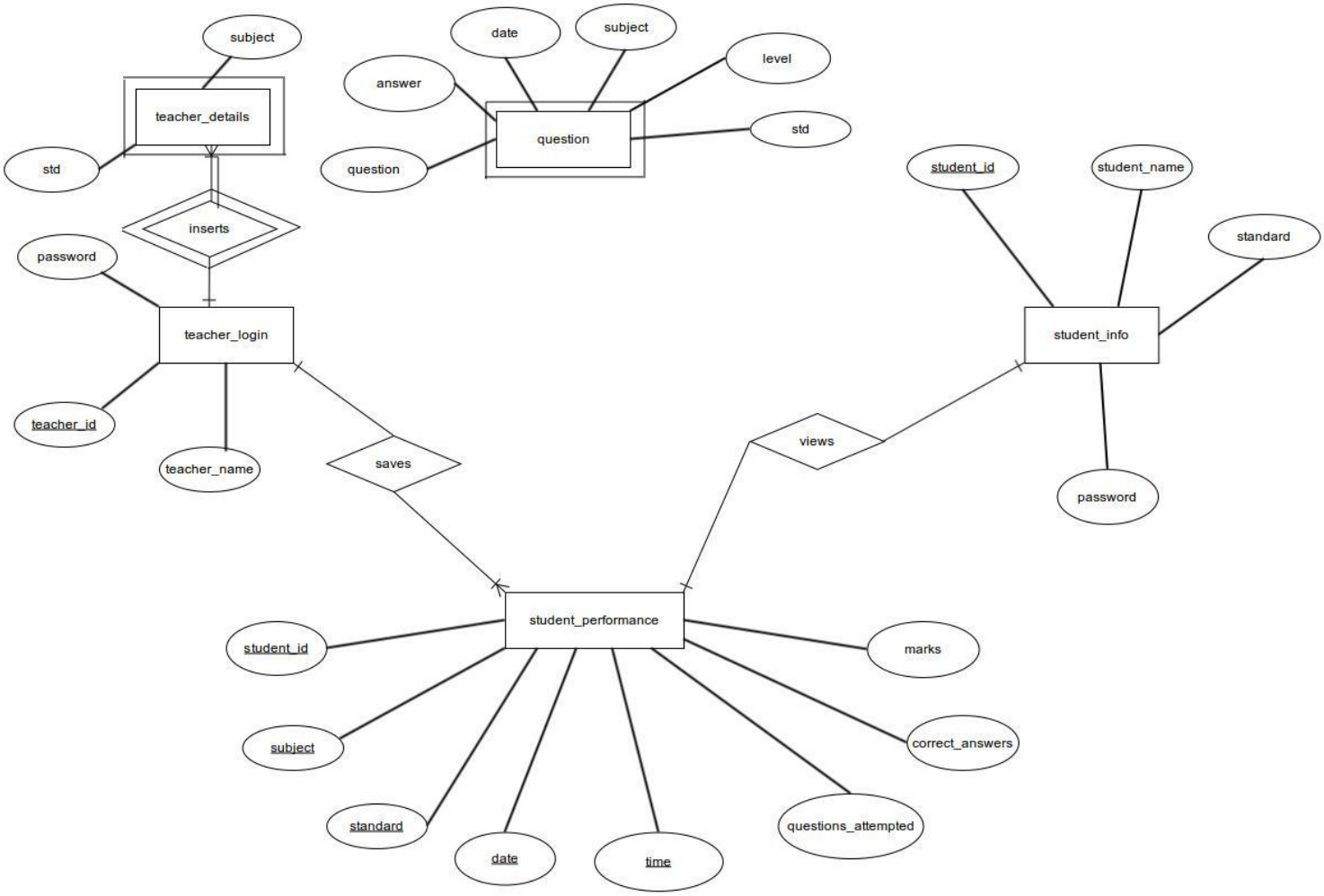


Figure 5.38: ER diagram of 'Eval'

Database Schema

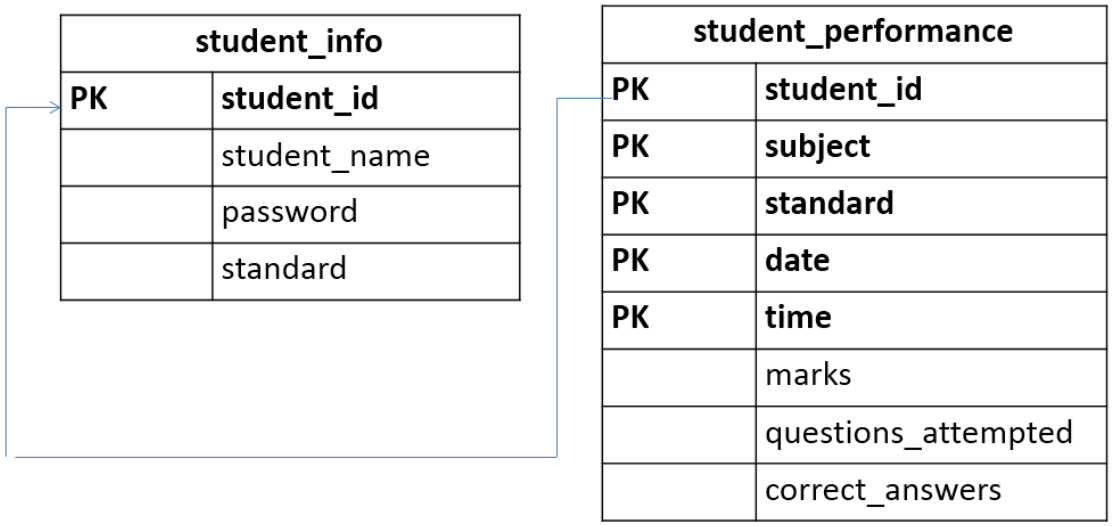


Figure 5.39: 'student_info' and 'student_performance'

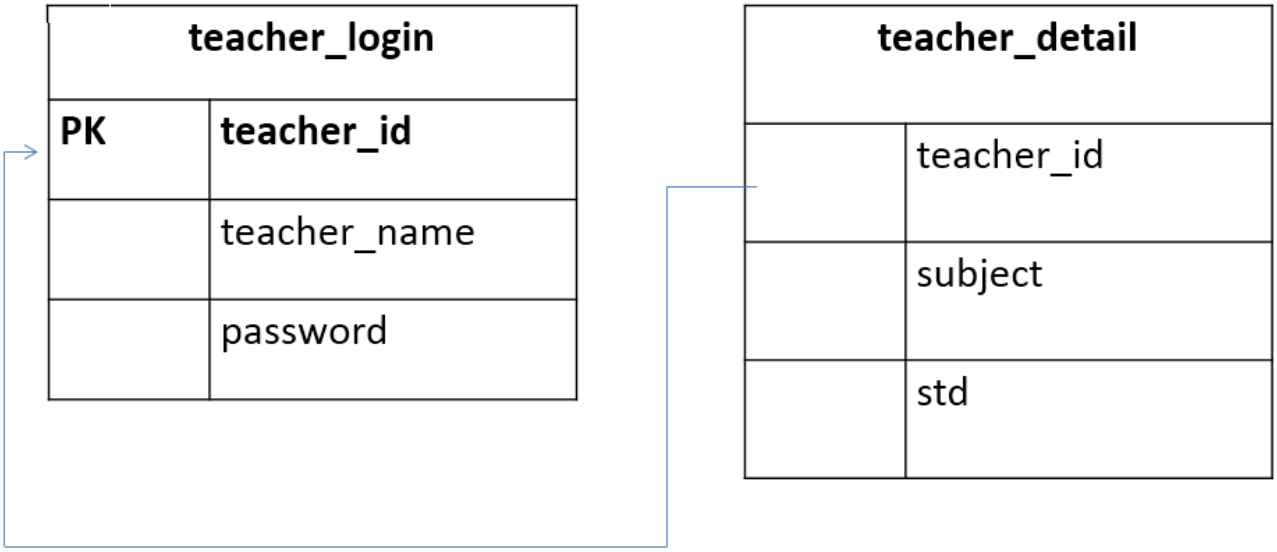


Figure 5.40: 'teacher_login' and 'teacher_detail'

question_table	
	question
	answer
	date
	subject
	level
	std

Figure 5.41: 'question_table'

student_info table: **student_id** field is the primary key

student_id (PK) - This field is formed by the concatenation of standard and roll number of a student
student_name - contains name of the student.
password - password is also auto generated. It is formed by concatenating the first name and the roll number.
standard - denotes the standard in which the student is studying.

student_performance table: **student_id**, **subject**, **standard**, **date** and **time** together form the primary key

student_id (PK)- denotes the unique id of every student, which is assigned at the time of registration.
subject (PK) - denotes the subject of the quiz.
standard (PK) - denotes the standard in which the student is studying.
date (PK) - denotes the date of the quiz.
time (PK) - denotes the time at which the quiz ended.

teacher_login table: **teacher_id** is the primary key

teacher_id (PK) - every teacher selects a teacher id at the time of registration.
teacher_name - denotes the name of the teacher
password - denotes the password that a teacher chooses while registering

teacher_detail table: **teacher_id** is foreign key to teacher_login table

teacher_id - the unique id of every teacher.
subject - the subject that the teacher teaches
std - denotes the standard in which the teacher teaches the above subject

question_table table:

question - contains the question
answer - contains the answer
date - denotes the date on which
subject - denotes the subject of the question
level - denotes the level of the question. This value is allotted by the teacher at the time of authentication
std - denotes the standard, in which the quiz was played

5.4 Quiz Module Description

- Quiz module consists of interaction between the client (Student) and the server (Teacher).
- Quiz will be initiated by the teacher. Students who are logged in at that moment will get an interface in which the instructions are displayed.
- The instructions contain number of students participating in the quiz, number of groups in the class, size of each group, subject of the quiz.
- These parameters are entered by the teacher in the server when he/she starts the quiz.
- Students also get an option of becoming a leader by pressing the 'Become a leader' button. If any student wants to become a leader, then he/she on pressing the button will send a packet to the server for becoming leader.
- For selecting this button to become a leader, there will a time limit set by the teacher. Students who want to become leaders should discuss with their teammates and give their requests before the time limit completes.
- After the time limit completes, the teacher will get to know who all gave leader requests and who all are assigned as leaders.
- Leaders are assigned on first come first serve basis, where the server will keep track of all the students who gave the requests to become leaders in a queue.
- Excess requests to become leaders are rejected and appropriate message is displayed to student that he/she can't become a leader.
- If any of the students in the class couldn't give their request in the time-span assigned to them for becoming a leader, then the teacher can repeat the session until all of the students (Who want to become leader) give their requests.
- After this leader session is completed, the teacher can go ahead and start group session.
- Before starting this, teacher knows who all are leader, so the leaders' will get an interface of selecting their group name.
- The other non leader people will get a listing of leader names to select the leader team to be in.

- After this group session completed, the students who opted for a specific leader will be assigned to that leader group and similarly everyone is assigned to their respective chosen groups
- Now, Teacher will get a screen in which all the formed group's, their leaders and team mates are displayed.
- Teacher can start the quiz now among these assigned groups.
- The quiz session will run in a round robin fashion where all the groups are given a chance to ask a question in a round.
- When a group gets a chance to ask a question, the people who are in the same group will physically discuss with each other and come to a conclusion in forming a question.
- Groups have an option of forming a multiple choice, one word, true or false questions.
- Only the leader of a group can enter the question which has been discussed with his/her teammates.
- After entering the question, the question will reach the server and will be displayed on the teacher interface, where he/she will validate the question.
- If the question is invalid, then teacher invalidates it by pressing invalid button. This will give a chance to the same group again for forming the question.
- This will be repeated until a valid and appropriate question is formed.
- If the teacher thinks that question is valid, then he/she validates it by pressing 'valid' button.
- After that, level of the question should also be entered by the teacher for the validated question, so that this level will be used as a mark criterion for the student group who asked this question.
- If the question is validated by the teacher, then all the people in the group who formed the question will get the marks according to the level of the question they formed.
- Now, the question is stored in the database with its answer, level, date on which is has been formed.
- Server will take this validated question and send it to all the other groups who are waiting for the question.
- All inactive groups (who are waiting for the question) will get an interface for answering the question which was formed by the active group (who got the turn and formed the question).
- Students in all the inactive groups will get +2 marks for answering the question correctly and 0 marks for answering it wrong.
- There is a time duration assigned by the teacher for both forming the question by the active group and for answering the question by all other passive groups.
- After the response time duration has been complete, Teacher will get a detailed analysis of answers given by students in a form of a pie chart.
- The pie chart contains how many students in the class answered option A, B, C, D in multiple choice, True or False in T/F questions and correct, incorrect in one word answers.
- If the student couldn't give his/her response within the time limit, then he/she will get 0 marks for that question. Immediately after the answering time completes, the next group randomly will be chosen and made active and all others are made passive.
- The active group leader will get the question interface and all other members of the same group will discuss the question and this continues as the same thing mention above.
- After every group gets turn, a round will be completed.
- The session will continue in the same manner until numbers of rounds specified by the teacher are finished.
- After all the rounds are finished, teacher will get a screen which shows the performance of the all the groups in terms of total marks.
- Now, the student records will be entered into the database in student_performance table with the no of questions attempted, no of answer's given correct and overall marks.
- This will be the end of the quiz and the teacher will be redirected to homepage.

- Students will get their performance stats after the quiz gets finished and will be redirected to their home page.
- Student's can see their performance in this test any time later by clicking the view performance option in both Student interfaces.
- Similarly teacher can also query for a specific student performance or the performance of a bunch of students.
- Questions which are stored in the server DB can be viewed later by students or teacher.

Chapter 6
TECHNICAL DETAILS

6. TECHNICAL DETAILS

6.1 Terminology:

Client: Student with Aakash tablet assigned a random IP by Access point.

Server: Teacher with Home Computer is a Server which is assigned IP 192.168.56.113 by the access point.

6.1.1 Abbreviations Used:

TCP: Transmission Control Protocol
JSON: JavaScript Object Notation
UDP: User Datagram Protocol
GUI: Graphical User Interface
DB: Database

6.2 Network Protocols used:

TCP: Transmission control protocol is used by the server and clients for transferring huge data like questions, performance statistics of all the tests and study materials (files).

UDP: User datagram protocol is used by the server and the client while performing the quiz and short request-response messages. Overhead incurred by using TCP for a large number of tablets is avoided by using UDP where there are no state full connection messages in the network.

Added features to the UDP protocol in Application Layer:
To scale our application to handle inconsistencies in the network and to work in the conditions where there is no reliability in the delivery of the packet, we have decided to add some reliable features over the existing UDP protocol rather than using TCP which has more overhead.

* UDP protocol is reliable, that means it can't assure that whatever is sent by using it will reach destination. Our application needed some form of reliability or acknowledgement that the packet which we sent is delivered.
* We used acknowledgements to assure that the packet which is sent by the server to the clients has reached.
* If the client doesn't reply in a specific timeout, then the server retransmits the same packet until the ack is received.
* Sequence numbers are used for every packet to discard all the stale packets.

6.3 Network Architecture:

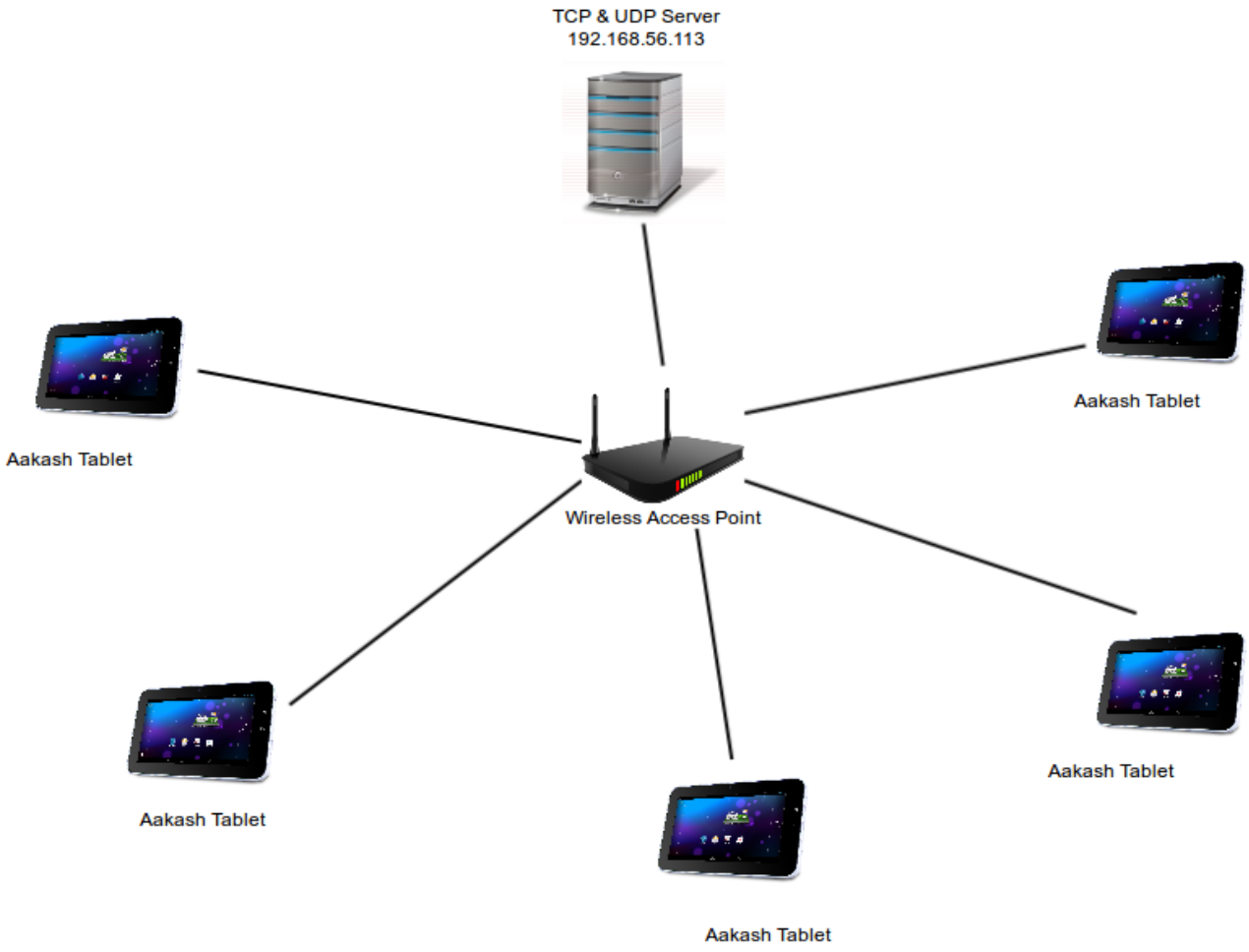


Figure 6.1: Network Topology

Server side:

Server uses Linux Operating system running TCP Multithreaded Server and UDP server as threads in the same Application using Java Sockets API.

Client side:

Client uses Android Sockets API with Internet Permissions enabled.

Network Ports used:

Client side:

Client uses 2 UDP ports.

1. 6789 Port for authentication requests.

2. 5555 Port for all the other functions
Client uses 1 random port for TCP connection.

Server side:
Server uses 1 TCP port 6711 for serving the TCP requests.
It also uses 2 UDP Ports.
1. 9876 for serving authentication requests
2. 4444 for all the other functions.

6.4 Technologies used:

1. Java Serialization is used to convert the Object representations in the client and in the server side to a byte representation for transferring through the network.
2. JSON Parser and Packers are used in the TCP server and TCP client for exchanging the data.

6.4.1 External Libraries used:

1. JfreeChart: This is used in the server side for displaying the detailed response statistics after every question
2. Simple-JSON: Used to convert the data to be sent through the network into a meaningful packet representation which will make the other end easy to extract the information.
3. Jackson- JSON: Used to convert the data to be sent through the network into a meaningful packet representation.
4. MySQL-Jdbc-Connector: Used to connect the MySQL Database to the Java Application
5. Android-Plot: This is used by the client to plot the performance graph generated by the server.

6.5 Technical Overview:

Server code is roughly divided into two modules. One is TCP server and the other is UDP server. UDP server serves the clients starting from login to the end of the application. TCP server is only used when the clients need to access their performances in the tests, fetch files or questions asked previously from the server.

Initially, server should be connected to the access point and it should be assigned an IP of 192.168.56.113. This can be done by adding a MAC entry in the Router's table to that IP address. This is mandatory because

the clients are required to connect or query to this IP address. Similarly, clients should be connected to the access point and get an IP from it and it can be random.

After everyone connects to the Access point, Server is started which in turn starts TCP and UDP server in the backend. Client's login is served by the UDP server as there is no connection needed to keep the client in the system. Client will give his/her credentials through GUI in the Aakash tablet. These details are stored in a packet which is serialized into byte stream. This packet contains a well structured format for accessing the data and interpreting the data. The serialized byte stream is sent on UDP to the server. Server knows the packet format and serializes the data sent by the client and authenticates the tablet by checking in the local database. Server also notes down the IP of the client in its local database for future use.

As soon as a client enters into the system, the server creates a special object for student which contains client information such as name, id, IP which will be used in the quiz session.

If the authentication is successful, then the client is entered into the system and will be displayed on the home screen where he can check his/her performance statistics, question asked in the previous quizzes and files uploaded by the teacher.

Meanwhile, Teacher also logs in to the system by entering his/her ID, Password, and Standard. The entered standard should match with the student's standard as they are in the same class. After teacher gets authenticated, the home screen is displayed.

Teacher can initiate the Quiz or check the performances of the students by clicking the appropriate option.

- All of the computationally expensive processing is done at the server side leaving client application very light weighted.
- There is no database stored in the client application and needs internet connectivity for logging in.
- All the questions are stored in the local database in the client side when they are fetched from the server.

Files:

- This option is provided at the client side of the application to allow students in the classroom to view the study materials uploaded by the teacher according to the standard. It would be helpful for teacher where in he/she can post some study materials for the upcoming quiz and students will download and go through it.
- Students can view only the files which are of their standard and can download them in to their local SD card for later viewing.
- Teacher will put the files in the respective folders on server.

  File Formats supported:
  Pdf, Doc, txt, Docx, Img.

File Path for storage:
Aakash Tablet (Student):
File Path: /mnt/sdcard/Eval

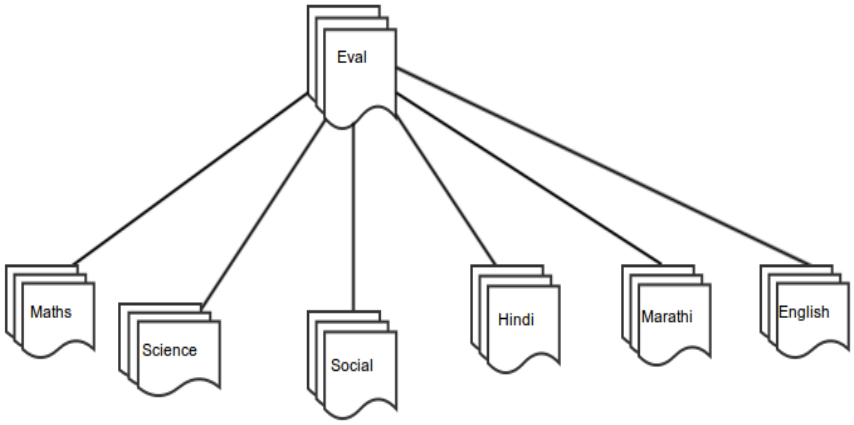


Figure 6.2: Directory Structure

- Eval directory is created when the application is installed for the first time.
- This directory consists of all the subjects like English, Hindi, Math, Marathi, Science, Social.
- Student can see all the downloaded files which were downloaded from the Teacher here.
- When the application starts for the first time, Teacher's Applications home directory will be used to create and organize folders according to the standard of the teacher.

File Download:

- Teacher makes the files available for the students by placing them in the appropriate standard directory as mentioned above
- Teacher has to use this format for naming the files in the directory mentioned above.
    - **Ex**: If teacher has a file named **Lesson1.doc** and its of subject **Hindi of 10th standard**, Then he/she will place the file in the directory **standard10/Lesson1_Hindi.doc**
- Students when they connect to the TCP server, they will get all the files which are currently present in the respective standard folder in server
- To perform this file transfer, we have used TCP connection as its reliable for the file exchange.

Procedure:

- Student clicks on the 'Files' option in the Home Screen and gets all the subject's and their files which have already been downloaded by him earlier.
- Student can download new files by selecting 'Download New Files' option.
- On selecting 'Download New Files' option, the tablet makes a TCP Connection with the server.
- Server will check the student standard and send back the files which are present right at that moment.
- Student will get a file listing sent by the server and can download files from it.
- Files downloaded will be saved in the file path /mnt/Eval/<Subject>
- Student can use file Manager to browse through the files and view them.

Questions:

- During the Quiz session, after the question is formed by the leader of the current group the question is stored in the database table named 'question_table' with its standard, level, date & time at which its formed.
- Questions are fetched by the students according to the standard in which they are in and are restricted to their standard questions only.
- Client uses TCP connection to query as the questions list can be huge.
- JSON is used in sending the questions from server to client, which gives a good way to exchange and interpret the data.
- Teacher can also view the questions that were formed by the students.
- When transferring lot of questions on each and every time when the student wants it is a big overhead and consumes a lot of bandwidth in the network. This will make network less usable to other clients. We handled the usage of the network efficiently by storing questions in a local database after fetching it from the server. Whenever the questions are queried by the students, the packet will be sent along with the last fetched date. This will be interpreted by the server and serves the client by giving only the questions which were formed after the last fetched date supplied by the client. This way the client will get new questions ans update the last fetched date. If there aren't any questions stored on the server since last fetched date, then nothing is sent back to the client. Now, client doesn't need to depend on the server each and every time for looking at the questions.
- For question's storage we are assuming that the tablet will be used by the same standard students every time. Otherwise the questions should be flushed out.

Student Performance:

- Quiz performances are recorded after all of the rounds get finished in every quiz. These performances will directly go into the 'student_performance' table from where the student or the teacher can fetch later.
- Number of questions attempted, Correct answers, Overall marks are stored after every quiz session.
- Teacher can view the performance of the students present in the class by choosing the 'Student performance' option in the home screen.

- Teacher can get a detailed report of the questions attempted, questions answered and marks for every student in the class. This will be presented to teacher in a pictorial form using pie charts.
- Teacher can use any of the parameters like student ID, date, and name, standard and date to query for performances.
- Student can view his/her performance by sending a JSON query string to the server by specifying the type of performance he/she wants. If 'overall performance' is selected, then the string is sent with the overall flag enabled or else it is not. The performance fetched by the client from the server is in the form of raw string and is converted to a graphical Pie Chart representation for last test performances else a graph for the entire subject's overall presentation. For this presentation we have used Android-plot Library.
- Student can log in through any tablet and can look at his last test performance of all the subjects or overall performance of all the subjects. This is made possible by storing all of the data in the server and keeping the tablets very light.

6.6 Challenges faced in the application:

- Connecting large number of tablets to the access point and running the system properly. In the worst case, if the access point switches off, TCP will consume lot of time and bandwidth for the connection establishment. This will be detrimental to the overall performance of the system. So, decision to use UDP broadcast as the transmission protocol was agreed upon.
- Decision to use UDP gave rise to another problem. UDP is unreliable and hence there is no guarantee whether a packet will be received by the other side. Whenever, the server sent packets, often a client or two didn't receive the packet and the system got stuck. Also, low signal strength of UDP broadcast packet is a problem. Hence, a challenge to make UDP reliable emerged. We went about the process of making UDP reliable by shifting from UDP broadcasting to UDP unicasting, in the process, introducing acknowledgement packets. Now, the server sent a packet to a client and waited for an acknowledgement from that client. If the server received a packet, then it transmitted to the next client, else it retransmitted the packet to the same client. After a fixed number of failed attempts to a client, the server deemed it to be out of the network and struck off that client's entry from its list. This approach solved our problem and now, the application scaled to a large number of tablets without any drops.
- Other challenge we faced during the application testing was to make sure the quiz continues even if the Leader who forms the question is not reachable through the network. This problem was quite challenging and has been successfully solved by our team. We have used the above described protocol methodology to send to the leader. If the leader is not reachable within some specific number of attempts, then his record from the current quiz will be removed and randomly a teammate will be picked and made as a leader for that group. If there are no teammates to be picked then the entire group will be removed.

- When using UDP on a Link with 1500 as MTU, we have to make sure that our packet won't be fragmented to avoid increasing packet loss. To avoid fragmentation, we have tried to keep all the packets with in 1000 bytes so that the packet's will reach to the destination without any penalties.

Server Source Code details:

Description	Lines of Code	Source file
This class initializes the Database and starts the Application session.	50	MainClass
This class authenticates the teacher and also displays the home screen for teacher.	265	StartSession
This Class is the core class of the Application which contains all the functions which are required for the Quiz session to perform seamlessly.	1521	Quiz
This class performs functions for allocating leaders and groups. This will be invoked by the Quiz class	687	LeaderSession
Has all the static methods for sending packet using UDP with reliability by using timeout's and retransmissions.	317	UDPReliableHelperClass
This Class has all static methods and variables which are used throughout the application by all the other classes.	150	Utilities
This contains all the methods required for sending performance details, questions and Files using TCP and JSON.	500	TCPServer
This class has all the methods for authenticating the client requests by verifying it in the DB. It uses UDP.	400	StudentLogin
This has static methods to create Datagram.	30	SocketUtilities

Chapter 7
TESTING

7. TESTING

Software testing is an investigation conducted to provide information about the quality of the product or service under test. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test technique includes the process of executing a program or application with the intent of finding software bugs.
Software testing can be stated as the process of verifying and validating that a software program/application/product:

- Meet the requirements that guide its design and development,
- Work as expected, and
- Can be deployed with the same characteristics.

Software testing can be implemented at any time depending on the testing method employed in the development process. However, most of the test effort traditionally occurs after the requirements have been defined and the coding process has been completed. It is observed that fixing a bug is less expensive if found earlier in the development process. Although in the agile approaches most of the test effort is, conversely, on-going. As such, the methodology of the test is governed by the software development methodology adopted.

7.1 Testing technique used

Several testing techniques have been performed on the system such as unit testing, integration testing and system testing.

7.1.1 Unit Testing:

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs.

7.1.2 Integration Testing:

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules).

1. Black-box testing- It is a method of software testing that examines the functionality of an application (e.g. what the software does) without peering into its internal structures or workings.

2. White-box testing- It is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. In this testing, we performed Boundary Value Analysis.

7.1.3 System Testing

System testing tests a completely integrated system to verify that it meets its requirements. In addition, the software testing should ensure that the program, as well as working as expected, does not also destroy or partially corrupt its operating environment or cause other processes within that environment to become inoperative.

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. In this testing we performed Alpha –testing.

Statistics of the tests performed at OSL lab

No of clients completed Quiz Successfully	No of clients (Tablets)	Quiz
10	10	1
19	20	2
28	30	3
37	40	4

CHAPTER 8
CONCLUSION

8. CONCLUSION

8.1 Concluding Text

The application aims to develop student-student interactive software. Students tend to learn faster when they work in groups.

8.2 Future Enhancement

- Embedding regional languages in the application, so that the state board schools are also benefitted
- Feature of uploading and submission of assignments, where the evaluation will be done by students only.

CHAPTER 9
REFERENCES

9. REFERENCES

➢ Web References:

- Android Developer Website, http://developer.android.com/
- Code Project Website, http://www.codeproject.com/Articles/146145/Android-D-Carousel
- Android Tutorial Website, http://en.wikipedia.org/wiki/Android
- Json format representation, http://www.mkyong.com/java/json-simple-example-read-and-write-json/
- Stack overflow, http://stackoverflow.com/

➢ Book References:

- Fundamentals Of Software Engineering – Rajib Mall
- Unix Network Programming – Richard Stevens
- Complete Reference for Java – Herbert Schildt

CHAPTER 10
APPENDIX

10. APPENDIX

Acronym or Abbreviation	Description
ADT	Android Development Tools
API	Application Programming Interface
APK file	Android Application Pac**k**age file
AVD	Android virtual Device
GUI	Graphical user Interface
IDE	Integrated Development Environment
SDD	Software Design Document
SDK	Software Development Kit
SRS	Software Requirement Specification
SDLC	Software Development Life Cycle