



VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Tsegaye Tefera

ANDROID MOBILE APPLICATION FOR STUDENT ENROLLMENT SYSTEM

Technology and communication

2015

ACKNOWLEDGEMENT

I thank my family for their help during my study, Dr.Ghodrat Moghadampour, for his time and patience and for teaching me several software courses including the Android programming. I also thank all the instructors who have been helping me all through my study time.

Beside my supervisor, I would like to thank Vaasa Summer University who organized VAMK special programming courses during the spring and summer since the Android Programming is a course that helped me to do my thesis.

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Degree Program in Information Technology

ABSTRACT

Author	Tsegaye Tefera
Title	Android Application for Student Enrollment System
Year	2015
Language	English
Pages	59
Name of Supervisor	Ghodrat Mohgadampour

Due to the increasing saturation of the mobile technology, mobile applications have gained significant business reputation. Things become easier after we start to use Mobile or tablet Applications. Android is one of the operating system on which more than one billion devices are running.

The objective of this project was to develop an Android mobile application which allows teachers and students to use the application anywhere using an Android phone or tablets. The application allows the teachers to login, add major course information, edit the existing course information, search a student and add a student to a course. On the other hand, the applications allow students to login, search and enroll to a course and also view their current course histories.

In achieving the objective of this project, the following three major parts were designed and implemented. Firstly, the design of the graphical user interface was implemented. Secondly, a MYSQL database that connects and communicates with the PHP was designed to store the student, teacher and course data. Thirdly, the PHP scripts were written and stored in online server that is used to retrieve data from the database.

The project can be extended to a design of an efficient and reliable application that can be used with at VAMK, University of Applied Sciences. The thesis can be used also as a reference to carry out further application developments in the Android mobile university application.

Keywords Android, MySQL, VAMK, Student Enrollment System

TABLE OF CONTENTS

ACKNOWLEDGEMENT	I
LIST OF ABBREVIATIONS	V
1 INTRODUCTION	9
2 RELEVANT TECHNOLOGIES	10
2.1 Application Structure	10
2.2 Application requirements	11
2.2.1 Android	11
2.2.2 JSON	12
2.2.3 MySQL Database	12
2.2.4 PHP	12
2.3 The Application Development Environment	12
2.3.1 Installation of Eclipse IDE	13
2.3.2 Installation of Android SDK	13
2.3.3 Installation of Android Development Tools	13
2.3.4 Installation of Android Virtual Device	14
2.4 Android Phone Application Development	14
3 APPLICATION DESCRIPTIONS	16
3.1 Quality Function Deployment	16
3.1.1 Normal Requirements (Must Have)	16
3.1.2 Expected Requirements (Should Have)	17
3.1.3 Optional Requirements (Nice to Have)	17
3.2 Use Case Diagram	17
3.2.1 Student Use Case	17
The student has the following sets of use cases in detail	18
3.2.2 Student Login Review use Case	18
3.2.3 Enroll Course Use Case	19
3.2.4 View Participation Use Case	20
3.2.5 Retake Exams Use Case	20
3.2.6 Teacher Use Case Diagram	21
3.2.7 Teacher Login Use Case	22

3.2.8	Teacher Add Course Use Case.....	23
3.2.9	Teacher Edit Course Use Case.....	24
3.2.10	Add a Student to Course Use Case	25
3.3	Sequence Diagram	25
3.4	Class diagram.....	27
4	DATABASE.....	29
4.1	Design of the Database	29
5	GRAPHICAL USER INTERFACE.....	32
5.1	Launch Page.....	32
5.2	Student Login Page	32
5.2.1	Student Main Menu Page.....	33
5.2.2	Participations.....	34
5.2.3	Enroll.....	35
5.3	Teacher Login Page	37
5.3.1	Teacher Main Menu Page	37
5.3.2	View Course.....	39
5.3.3	Add Course	40
6	TESTING	42
6.1	Invalid Login Credentials	42
6.2	Student Test Cases	42
7	IMPLEMENTATION	46
7.1	User Login	46
7.2	View Registered Courses Activity Class	50
7.3	Add New Course Activity Class	52
7.4	Enroll To Course Activity.....	53
7.5	Android Manifest	55
8	CONCLUSION	57
	REFERENCES.....	58

LIST OF ABBREVIATIONS

App	Application Program
API	Application Programming Interface
JSON	JavaScript Object Notation
OS	Operating System
JVM	Java Virtual Machine
PHP	Personal home page Hypertext Preprocessor
SQL	Structured Query Language
SD card	Secure Digital Memory Card
GPL	GNU General Public license
IP	Internet Protocol
GUI	Graphical User Interface
XML	Extensible Markup Language
IOS	Iphone OS
HTML	Hypertext Markup Language

JSP	Java Server Pages
IDE	Integrated Development Environment
JAVA EE	Java Platform Enterprise Edition
SDK	Software Development Kit
ADT	Android Development Tools
APK	Android Application Package
AVD	Android Virtual Device
ER	Entity Relationship
VAMK	Vaasan ammattikorkeakoulu
OHA	Open Handset Alliance
HTTP	Hypertext Transfer Protocol

LIST OF FIGURES AND CODE SNIPPETS

Figure 1. Architecture of the application	p.11
Figure 2. Android Application Development	p.15
Figure 3. Use case for activities of the student	p.18
Figure 4: Student Login Use Case	p.18
Figure 5: Student Register Use Case	p.19
Figure 6: Student View Participations use case	p.20
Figure 7: Retake Exams use cases	p.21
Figure 8. Use case for activities of the teacher	p.22
Figure 9. Teacher Login Use case	p.22
Figure 10. Teacher Adds course use case	p.23
Figure 11. Teachers edit course use case	p.24
Figure 12: Add a student to a course use case	p.25
Figure 13: Student login Sequence diagram	p.26
Figure 14 .Teacher login Sequence diagram	p.27
Figure 15. Class diagram for the student Package	p.28
Figure 16. ER diagram of student enrollment system	p.30
Figure 17. Launch page	p.32
Figure 18. Student Login Page	p.33

Figure 19. Student menu page	p.34
Figure 20. View Participations	p.35
Figure 21. Enroll to Course	p.36
Figure 22. Teacher login page	p.37
Figure 23. Teacher main menu Page	p.38
Figure 24. View Courses	p.39
Figure 25. Student lists taking a particular course	p.40
Figure 26. Add Course	p.41
Figure 27. User Login test case	p.42
Figure 28. Main student page	p.43
Figure 29. Participations test case	p.44
Figure 30. Enroll to courses test case	p.45
Code Snippet 1. Student login xml	p.47
Code Snippet 2. Declaring and assigning buttons	p.48
Code Snippet 3. User login checking credentials	p.49
Code Snippet 4. Login process class	p.50
Code Snippet 5. View registered courses	p.51
Code Snippet 6. Assign the course main details	p.52
Code Snippet 7. Adding the course main details	p.53
Code snippet 8. Enroll Course Class in Enroll Course Activity	p.54
Code Snippet 9. Get course list class in enroll Course Activity	p.55
Code Snippet 10. Android manifest xml	p.56

1 INTRODUCTION

The Computer has been in constant evolution: the computer gets smaller in size continuously; modern computers can save a huge amount of data and can handle advanced programs. In 2008 Apple released their I-Phone that led to the new type of communication device using Apple operating system /1/. Applications are typically operated by the owner of the mobile operating system, such as Google Play, Apple App Store and Window Phone Store.

One of the giant operating systems which compute with Apple is Android mobile operating system (OS). Android is based on the Linux kernel and designed primarily for touch screen mobile devices, such as smart phones and tablet computers, and is currently developed by Google. Recently some devices like cars, televisions, wrist watches and refrigerators are running Android application with specialized user interfaces. The reason what makes many devices popular is that they are easy to use, free, and customizable and they work with all Google applications.

Android versions have been developed from the first commercial version, Android 1.0 which was released in 2008 to the current Android 5.0. Each of the versions has a code name released in alphabetic order from Alpha (1.0) to Lollipop (5.0) under ongoing development by Google and the Open OHA. /2/

This project targets to an application that the students and teachers can use more frequently rather than using a computer. By using the mobile application users can easily track the courses history at any time using a mobile or tablet.

This allows course histories to be uploaded to a central server where the data can be shared with all other Android users in the group depending on the privilege. Using Android's internet capabilities, the application communicates with the application server. The application has a server side part designed and implemented in PHP, which sends and receives data from a MySQL database.

2 RELEVANT TECHNOLOGIES

In this section an explanation is given to the application structure, application development cycle and technologies used.

2.1 Application Structure

The mobile application provides a user friendly interface for the user's. The project is composed of two main major parts which are client side and server side. In this project the Android application cannot communicate with a remote MYSQL database directly. Therefore, PHP is included which helps to communicate and allows to edit and execute scripts and interactive interpreters directly on the Android device. The Android application calls a PHP script in order to perform a data operation, let's say "add course". The PHP script then connects to the MYSQL database to perform the operation. So the data goes from the Android application to the PHP script, and then finally is stored in the MYSQL database.

Below is a diagram of the system which illustrates the interactions between the server and client side applications.

A breakdown of the steps:

1. The client makes a request using a HTTP POST to a server.
2. The PHP script queries the MYSQL server.
3. The PHP script gets the SQL data.
4. The PHP script puts the data into JSON (JavaScript Object Notation) array by assigning the keys for the values. The application parses the JSON and displays the required data to the client.

The process in visual form would look like this:

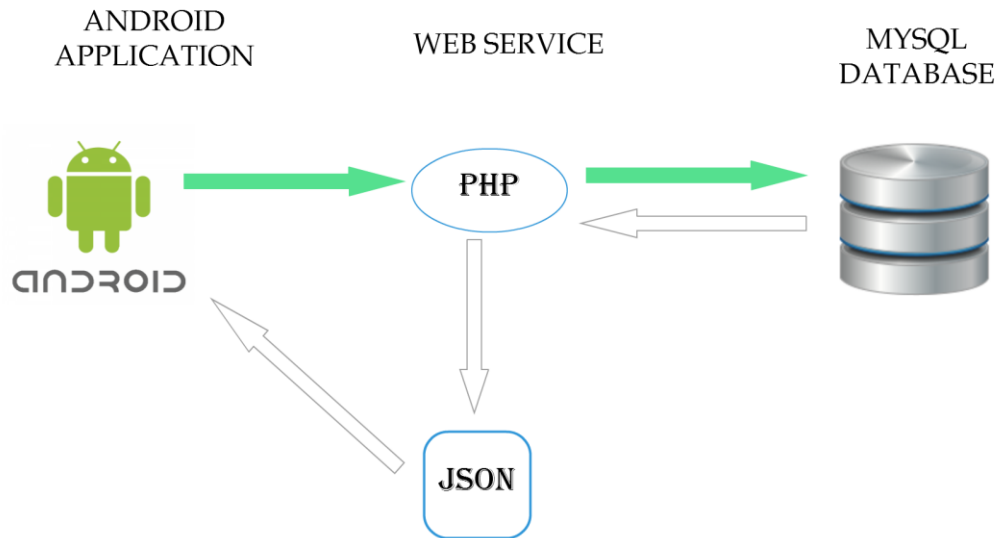


Figure 1: Architecture of the application.

The application is made in a way that to get the required data from a remote server whenever there is only an Internet connection. So users cannot use this application if there is no Internet connection in their device.

2.2 Application requirements

Application components are the most important building blocks of an Android application. One should need to understand these programming languages listed down below in order to achieve the goal of this project. In this project different programming technologies are used Android, PHP, MYSQL and JSON array.

2.2.1 Android

Android applications are developed using the Java language which is a very popular programming language developed by Sun Microsystems [3]. Some of the Java's important major features are that it is easy to learn and understand, it is object-oriented, designed to be platform-independent and secure. Android relies heavily on these Java fundamentals.

2.2.2 JSON

JSON stands for JavaScript object notation, which is format of data, used to send and retrieve data on the Internet. It is a lightweight data-interchange format that is easy to understand, write and read and also it is easy for machines to parse and generate. It is used to transmit data between a server and web application as to substitute XML in most applications.

Although originally derived from the JavaScript scripting language it has a data format that does not directly depend on the language. The code for parsing and generating JSON data is available in many programming languages. /4/

2.2.3 MySQL Database

Many computer applications deal with a large amount of information. A database is a collection of pieces of data and meta-information (schema) on how this data is organized. The MySQL development project has made its source code available under the terms of the GNU licence. MySQL is the most popular open source database owned by Oracle.

2.2.4 PHP

PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. It is a server-side scripting language, like JSP. PHP scripts are executed on the server and it supports many popular databases like Oracle, MySQL and PostgreSQL. It is open source software and free to download. PHP runs on different platforms (Windows, Linux, UNIX, etc.) and is compatible with almost all servers used today. /5/

2.3 The Application Development Environment

In order to achieve the objective of the project, a right working environment was needed. Therefore, the following software tools are required to set up the environment for developing Android applications.

To develop an Android application, developers need to make sure that the development environment has the recent Java versions or above Java 5. Since Java is operating system independent, the developer can freely choose the developing environment for their right operating system. In order to make this project the tools listed below should be installed on the computer. The installation of the tools is also described.

2.3.1 Installation of Eclipse IDE

For the Android programming the Eclipse IDE for Java EE Developers can be used, which is a multi-language software development environment featuring an extensible plug-in system. It is available and should be downloaded from the link (<https://eclipse.org/downloads/>). Once Eclipse IDE was downloaded a directory for the project was created and the downloaded Eclipse was put inside the directory created.

2.3.2 Installation of Android SDK

Android SDK (<http://developer.android.com/sdk/index.html>) contains all the necessary packages, class libraries, application framework, documentation and other packages. Android SDK should be downloaded and installed and the Path to locate it set up. After the Android SDK is downloaded, it is important to extract contents of the ZIP file to a directory folder, such as C:\Android folder. /6/

2.3.3 Installation of Android Development Tools

The Android Development Tool (ADT) plug-in for Eclipse is an extension to Eclipse IDE that supports the creation and debugging of Android applications. ADT allows the developer to create new Android application projects, access the tools for accessing Android virtual devices, compile and debug Android applications and export Android applications into Android Packages (APK). There are many ways of installing the ADT but the common way to install ADT on Eclipse is to go through Help->Install New Software and to type on the Install window in the Work with field : <https://dl-ssl.google.com/android/eclipse/>

After installing ADT and restarting Eclipse it is important to go through click Windows->Preferences item and in the Preferences Window select Android from the left window and make sure that the text field for SDK Location has a correct value referring to the installation folder of Android SDK. /7/

2.3.4 Installation of Android Virtual Device

The next step in setting up the development environment is to create Android Virtual Devices (AVDs) to be used during the development process. An AVD is an emulator instance that enables modeling an actual device.

Different AVDs can be created in order to test applications with different configurations which help the developer to observe the nature of the application on different devices with varying capabilities.

In order to create the Emulator we have go to Window/Android Virtual Device Manager and from the window we can check the created AVDs or move to Device Definitions and then select a device definition and then click Create AVD button./8/

2.4 Android Phone Application Development

Every phase throughout the application development process is wisely arranged to bring the highest level of customer satisfaction. The Android application development is seen as flowing through four main development phases like Setup phase, Development phase, Debugging and Testing and Publishing. These phases are described in short below.

The Setup Phase is the phase where it is important to provide the Android working environment by downloading the necessary tools .In addition the AVD's should be set up in order to test the application during the development phase.

The Development phase is when the application is built with our own source codes.

Debugging and Testing is the phase where the application is built and run in the debug mode and it will be confirmed also that the application is ready after it has been tested. The last phase is publishing the application which helps the users to use the application after it has been published in the market. The following figures describe the details.

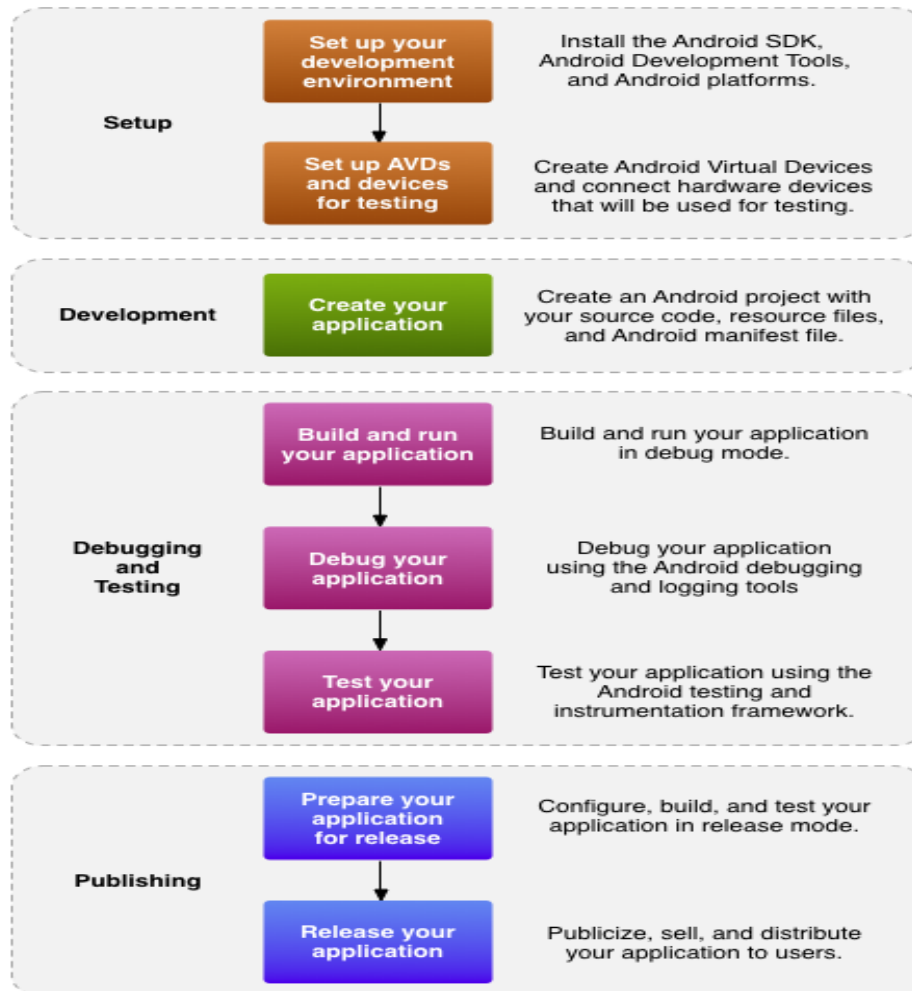


Figure 2: Android Application Development./9/

3 APPLICATION DESCRIPTIONS

In this section explanation is given to the detailed description of the project and requirements. The scope of the project encompasses both server- and client-side functionalities, so the client side is covered in detail within this section

3.1 Quality Function Deployment

The following list offers a brief outline and description of the main features and functionalities of the system. The features are split into three major categories: Must have, Should have and Nice to have. Must have or Core features are essential to the application's operation, whereas Nice to have or additional features simply add new functionalities.

3.1.1 Normal Requirements (Must Have)

Students must be able to

- Login with their existing user name and password
- Register or Enroll to a course
- Unenroll from a course
- To see the courses he/she has registered
- Enroll to retake exams.

Teachers must be able to

- Login with their existing user name and password
- Create and add courses
- Edit the added course details
- View courses given by the teacher
- Add a student to a course

3.1.2 Expected Requirements (Should Have)

The application should be user friendly .Users should be able to login easily and proceed to deal with the important features on the application. Therefore, user login page for the username and password should be included inside the application.

3.1.3 Optional Requirements (Nice to Have)

A student can view the grade for the course, a teacher can mark grade for the courses and the application let the user can view academic schedule.

3.2 Use Case Diagram

A use case describes how a system acts and responds to inputs by the primary actors. The student and the instructor have their own page to login. They should use the preregistered username and password for authentication which authorizes them to have access to all features on the application. In this project we cannot register any new student or teacher from this application. It can be done from the server side or the administrator can add a new user since the administrator has access to all the information of the registered users. All the necessary user information is stored on online database. PHP is written and stored on online server that is used later to fetch the required data.

3.2.1 Student Use Case

The following figure shows all the main activities of the Student use cases.

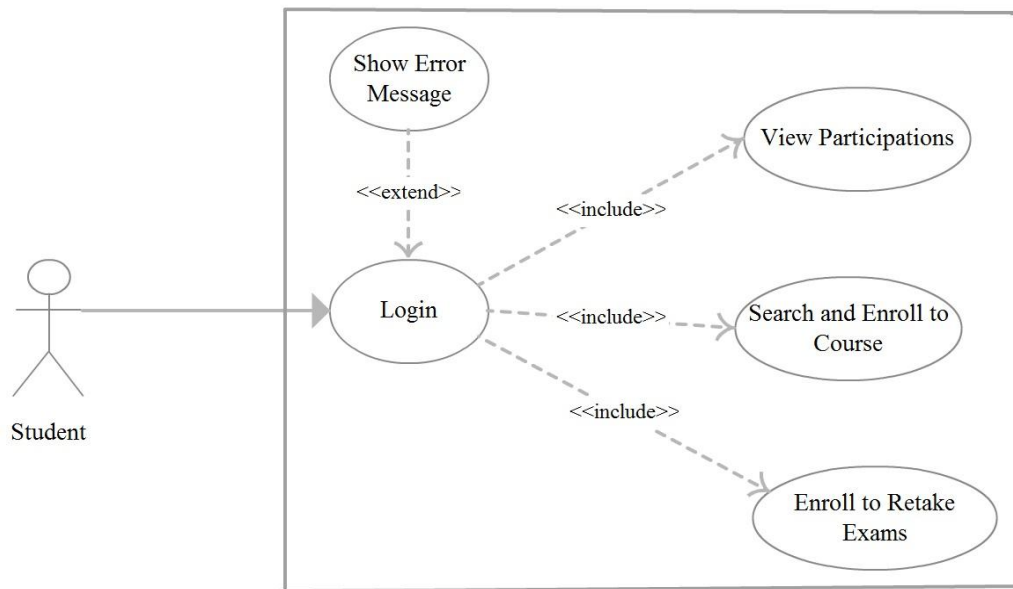


Figure 3: Use case for the activities of the student

The student has the following sets of use cases in detail.

3.2.2 Student Login Review use Case

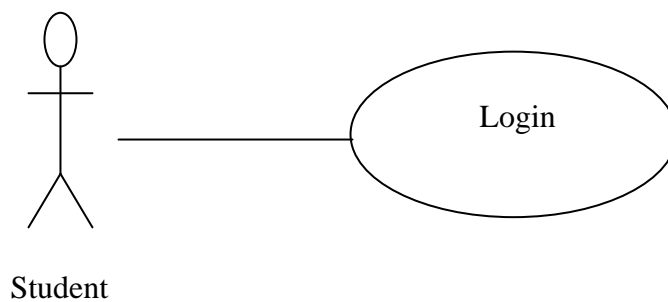


Figure 4: Student Login Use Case

Brief Description

The student logs in with his/her own user name and password.

Initial Step-By-Step Description

Before this use case can be initiated, the Student has already connected to the Website.

1. The student types his user name and password and press login button.
2. The System checks if it is a correct name and password.
3. The System generates and sends an email acknowledgement.

3.2.3 Enroll Course Use Case

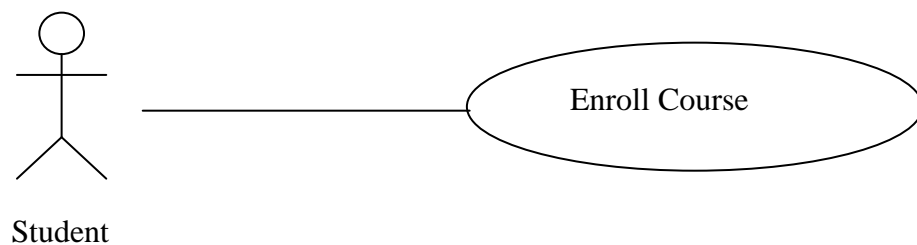


Figure 5: Student Enroll Use Case

Brief Description

The student chooses and registers into a course from the available list of courses.

Initial Step-By-Step Description

Before this use case can be initiated, the student has already accessed the main page of the register.

1. The system presents a choice of adding.
2. The student chooses a course.
3. After the student has chosen a course, the system presents options for the teacher to enroll and cancel.
4. If the course is enrolled into, the system verifies the information and returns the student to the student main page.

3.2.4 View Participation Use Case

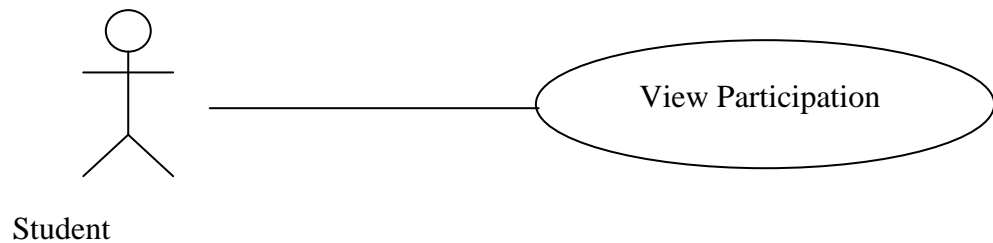


Figure 6: Student View Participations use case

Brief Description

The student views all the enrolled courses.

Initial Step-By-Step Description

Before this use case can be initiated, the student has already accessed the main page of the Register .

1. The student selects to *View Participation*.
2. The system presents the list of enrolled courses.
3. The student sees the information and clicks cancel.
4. The system verifies the information and returns the student to the register main page.

3.2.5 Retake Exams Use Case

Brief Description

The student registers for retake exams.

Initial Step-By-Step Description

Before this use case can be initiated, the student has already accessed the main page of the Register .

1. The student clicks *Retake Exams* button.
2. The system presents the list of active retake exams
3. The system presents the information about the chosen course and the student clicks the enroll button.
4. The system verifies the information and returns the student to the register main page.

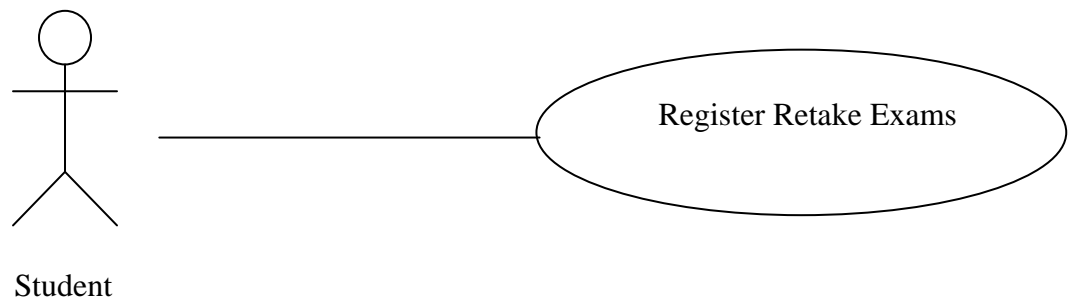


Figure 7: Retake Exams use cases

3.2.6 Teacher Use Case Diagram

The actor in the following set of use case is the teacher that performs all the actions that trigger an event in the teacher side user interface. In Figure 8, the main possible activities in the use case diagram of the teacher are presented and later each use case is presented in detail.

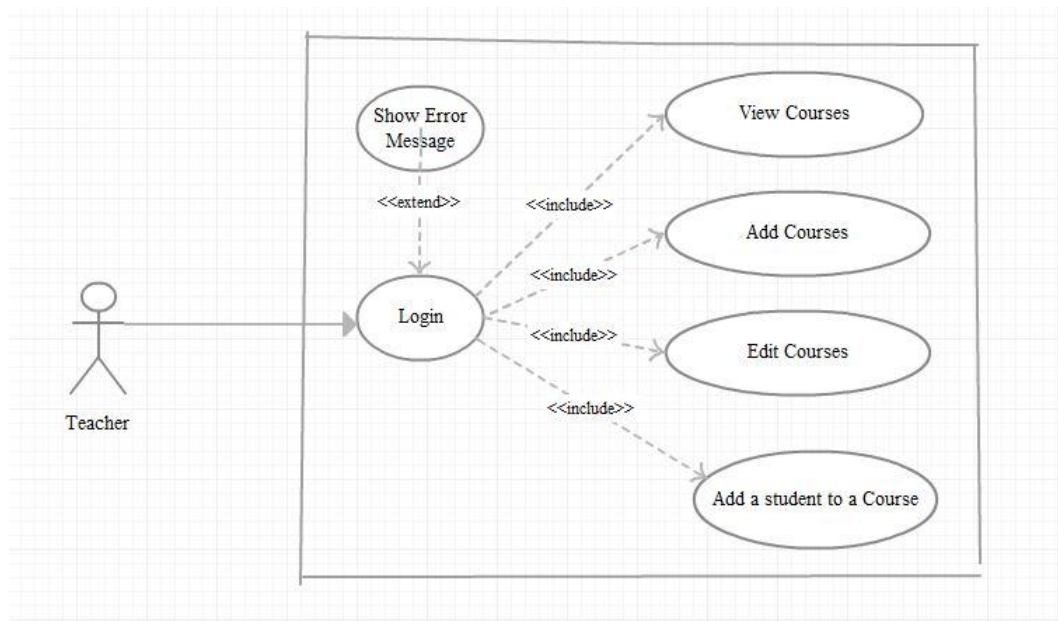


Figure 8: Use case for activities of the teacher

The teachers have the following set of use cases. In Figure 9 the teacher login use case is presented.

3.2.7 Teacher Login Use Case

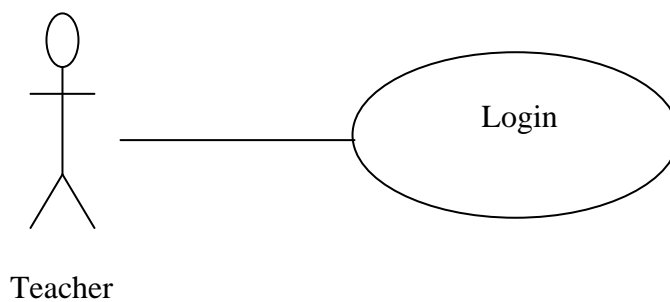


Figure 9: Teacher Login Use case

Brief Description

The teacher logs in with the user name and password.

Initial Step-By-Step Description

Before this use case can be initiated, the teacher has already connected to the main page.

1. The teacher types his user name and password and presses login button.
2. The system checks if it is a correct name and password and if the teacher has typed correctly, then Add course and View participation come up.
3. The system generates and sends an email acknowledgement.

3.2.8 Teacher Add Course Use Case

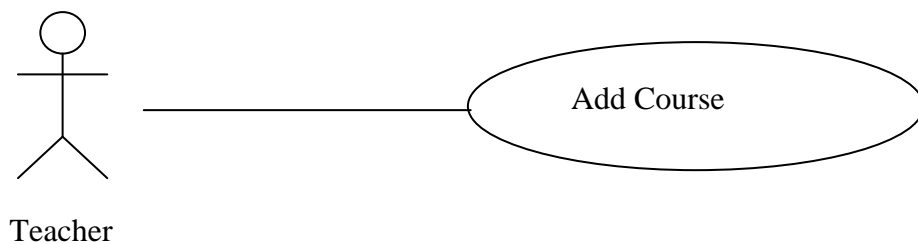


Figure 10: Teacher Adds Course Use case.

Brief Description

The teacher adds course and enters a new course name, course code and course finishing date.

Initial Step-By-Step Description

Before this use case can be initiated, the teacher has already accessed the main page of the register.

The teacher selects *Add Course*.

1. The system presents a choice of adding.
2. The teacher enters the course details .

3. After the teacher has added a course, the system presents options for the teacher to add and cancel.
4. The teacher selects and the system verifies the information and returns the teacher to the teacher main page.

3.2.9 Teacher Edit Course Use Case

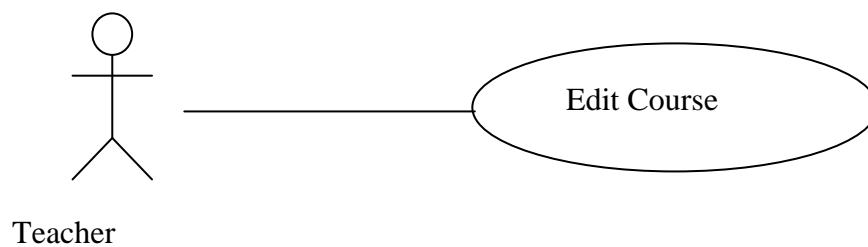


Figure 11: Teacher edit course use case

Figure 11 above is explained in detail in the following brief description.

Brief Description

The teacher edits information about an existing course.

Initial Step-By-Step Description

Before this use case can be initiated, the teacher has already accessed the main page of the Teacher.

1. The teacher selects to *Edit Course*.
2. The system presents the list of active courses.
3. The system presents the information about the chosen course.
4. The teacher sees the information and selects the course detail that the teacher wishes to edit.
5. The system verifies the information and returns the teacher to the teacher main page.

3.2.10 Add a Student to Course Use Case

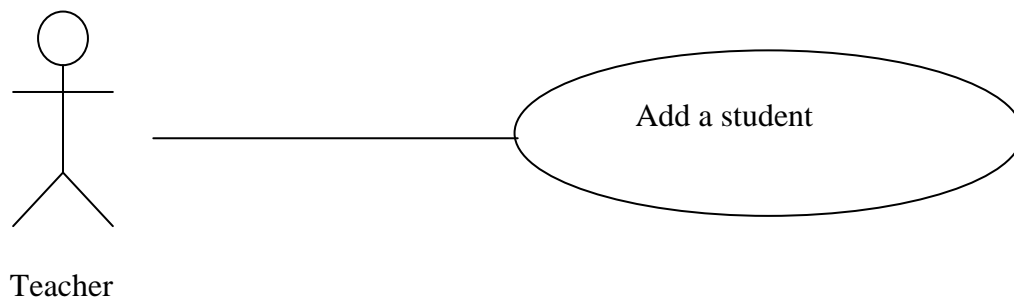


Figure 12: Add a student to a course use case.

Brief Description

The teacher adds a student to an existing course.

Initial Step-By-Step Description

Before this use case can be initiated, the teacher has already accessed the main page of the teacher.

1. The system presents the list of active courses and student lists.
2. The teacher chooses a course.
3. The system presents the students enrolled to the chosen course.
4. The teacher views the information and clicks Add student to a course.
5. The system verifies the information and returns the teacher to the main page.

3.3 Sequence Diagram

The following sequence diagram describes in detail the involvement of different participants in managing the student login instance on the application. The sequence is described through those main classes for this case. Note that some important calls to the Android API were skipped to help to keep our focus on the main login student sequence.

Figure 13 describes the login sequence diagram of the student side feature. It can

be easily seen that the username and password is checked if it exists already in the database for further feature of the application to proceed.

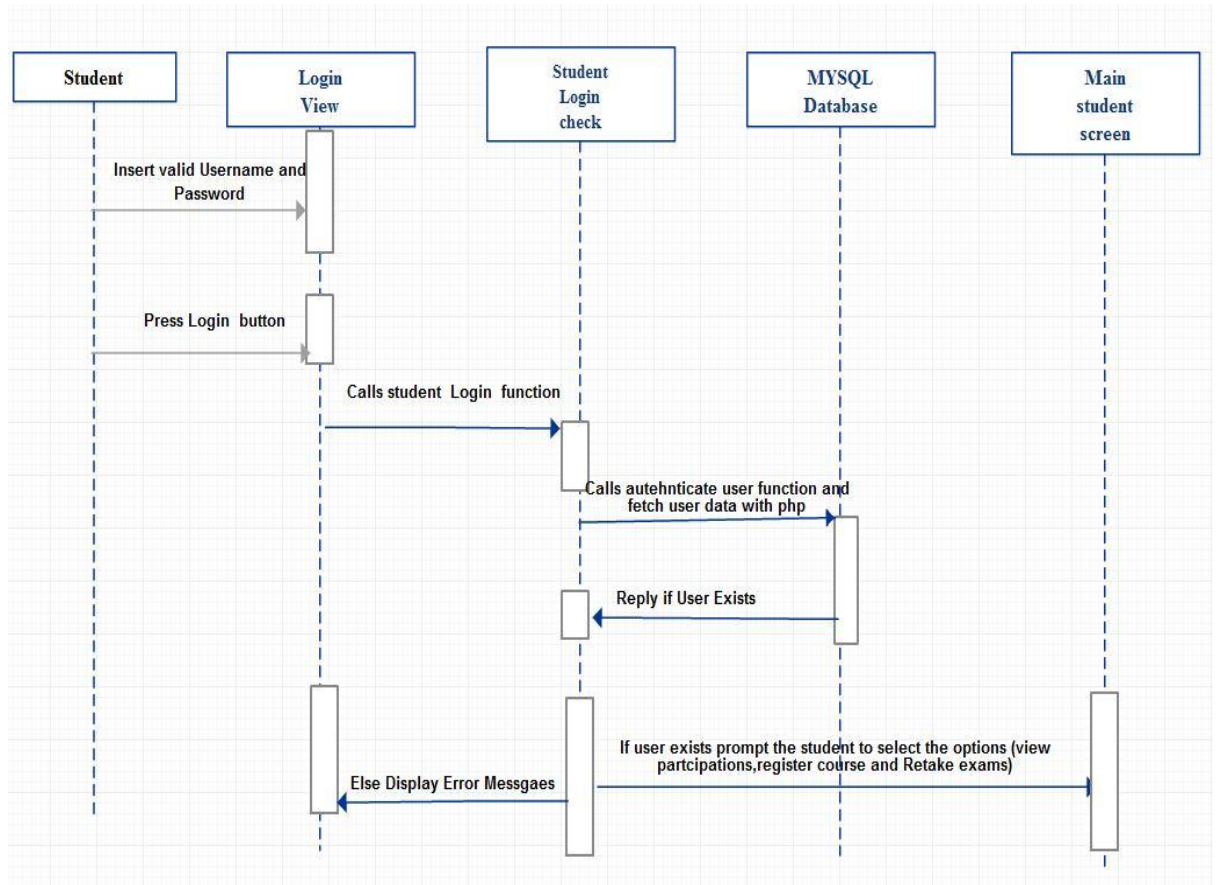


Figure 13 .Student login Sequence diagram

Figure 14 presents the login sequence through which the teacher accesses the main teacher page interface. The main menu teacher page contains the important features of the teacher side application.

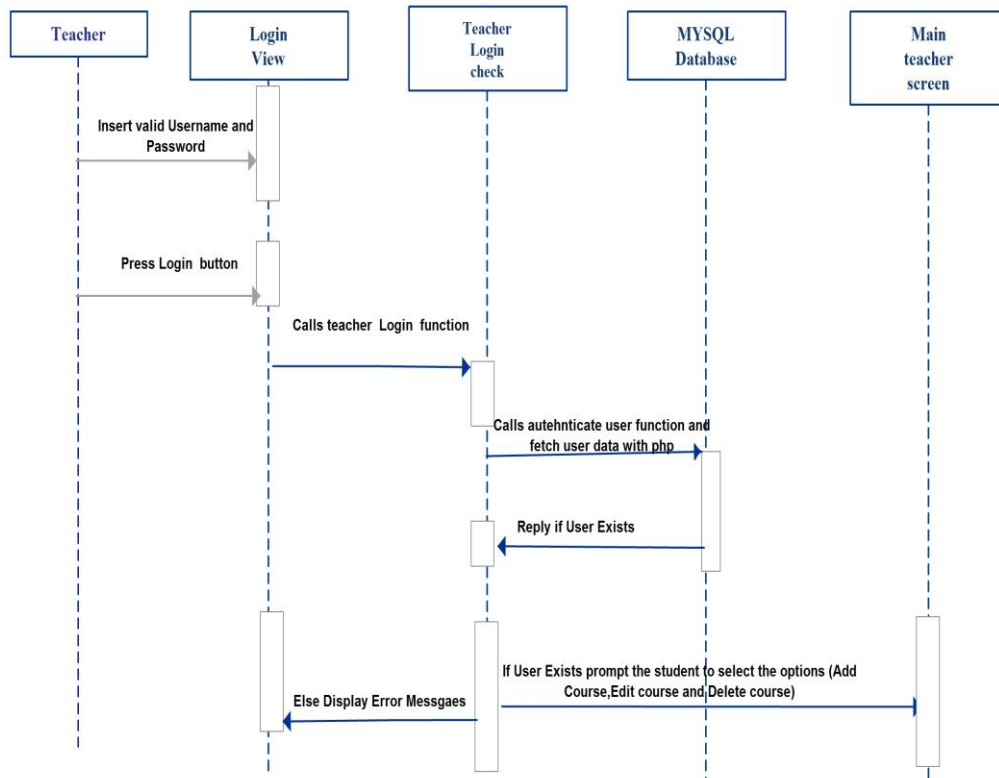


Figure 14 .Teacher login Sequence diagram

3.4 Class diagram

The class diagram which is also one of the most popular UML diagrams shows a collection of classes, interfaces, associations, collaborations and constraints. The purpose of the class diagram is to analyze and design the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction /10/. Figure 15 describes the declaration of each parameter of different classes and shows the relation among the classes that are used inside the student package.

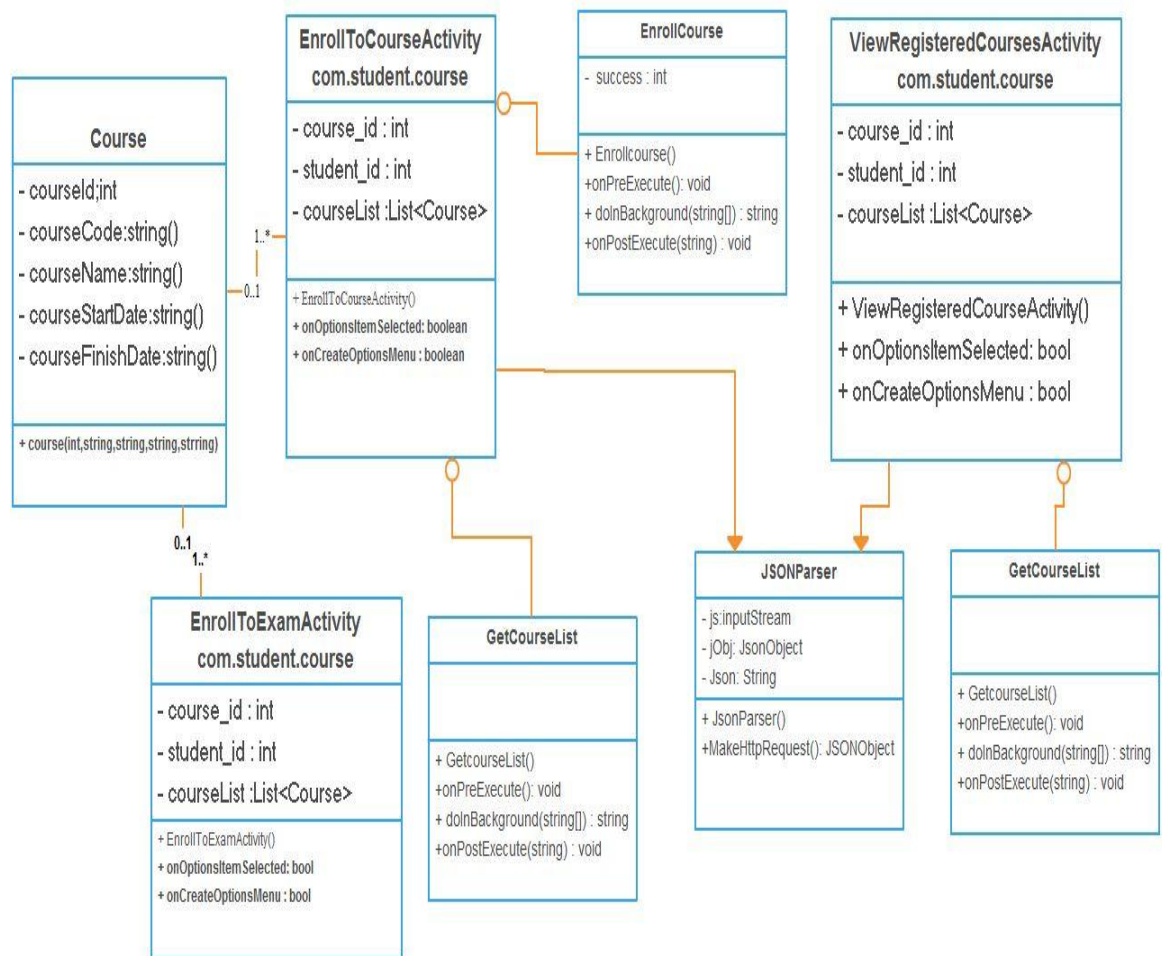


Figure 15: Class diagram for the student Package

In the above figure it is shown that course, Enroll to Course Activity, Enroll to Exam Activity View Registered Courses are identified as the main elements of the system and their relationship is shown between these elements. For example Course and Enroll to Course Activity have a one-to-many relationship because a course can be called by one or many enroll to course activity.

4 DATABASE

This application basically uses the database on online server. Therefore, the database that is used to store and retrieve data records is found in the remote school server (www.mysql.cc.puv.fi). For this application, a database named the 0800841_e0800841 was created.

4.1 Design of the Database

After creating the database, it is also important to create the database tables. The database consists of six tables. These are student, teacher, course, enrollment, enrollment for exam and retake exam tables. Each table has its own attributes that are restricted by possible values of the domain and we also need to have relationships between these tables. For instance, teachers add a course, and a course is enrolled by students. These relationships need to be represented in the database. Also, when fetching data with SQL which are written inside the PHP script in this project, certain types of JOIN queries need to be used to get the information needed. The relationships between the tables are as shown in Figure 16 diagram below.

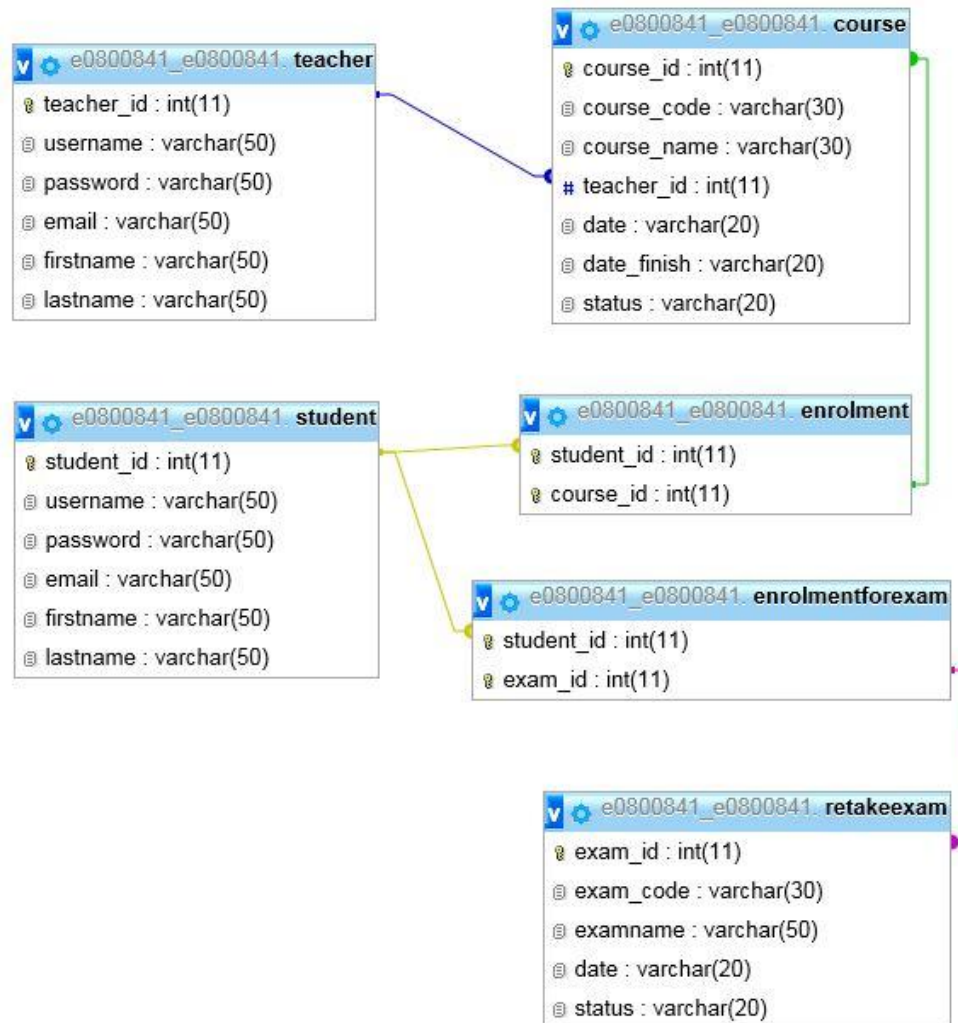


Figure 16: ER diagram of the student enrollment database

In the above Figure 16 it is shown that the teacher table with primary key `teacher_id` has a one-to-many relationship with the course table that has a primary key `course_id`. The relationship between these tables is designed on the idea that teacher can have multiple courses.

The student table with the primary key `student_id` has many-to-many relationship with the course table. In a many-to-many relationship; one or more rows in a student table can be related to 0, 1 or many rows in course table. Therefore, each row in student table is linked to 0, 1 or many rows course table and vice versa. Since in

relational database design, a many-to-many relationship is not allowed, a new table called an enrollment table is required in order to implement such a relationship. Similarly, enrollment for exam table is created in order to substitute a many-to-many relationship between the student table and the retake exam table.

In Figure 15 it can be seen the primary keys and foreign keys are used in order to prevent invalid data from being inserted into the tables, and also used to establish and enforce a link between the data in these tables.

The Android application cannot directly connect to the MYSQL database. Therefore, PHP is used to create a perfect development environment because of it is an easy way of interaction with databases.

The actual server side code is written using PHP. The PHP files are stored in the public html directory which can be accessed in the school network. For making a connection to the PHP script, it uses the HTTP protocol from the Android system. The following paragraph explains the details how it works.

When the server name, the database name, the password and username that have used for making the database are known then it is possible to start writing the PHP script. These fields will be required for making the PHP script that will be used for login.php.

The user inputs his username and password (in case of login) and presses the login button. The Android application calls a URL:

<http://www.cc.puv.fi/~e0800841/android/login.php> on calling the URL, a PHP script runs, that receives the data that has been sent from the Android application, with the method called POST.

After that the PHP script runs query to the database and compares the username and password with already existing username and password in the database. If the result is true, it will output its data with a value of true in the JSON format. Once the returned data is true, the user application will then move to the next activity that is to be shown to the user. In case the user login information is invalid an error message is sent to the user that the information that he enters is invalid.

5 GRAPHICAL USER INTERFACE

This chapter provides an overview of the user interface. The easiest mechanism for designing a user interface for an Android application is using editing and writing XML files.

5.1 Launch Page

The Launch page was the first page which will be displayed every time whenever we want to use the application. There are two buttons which lead the teacher or student to choose their own page.

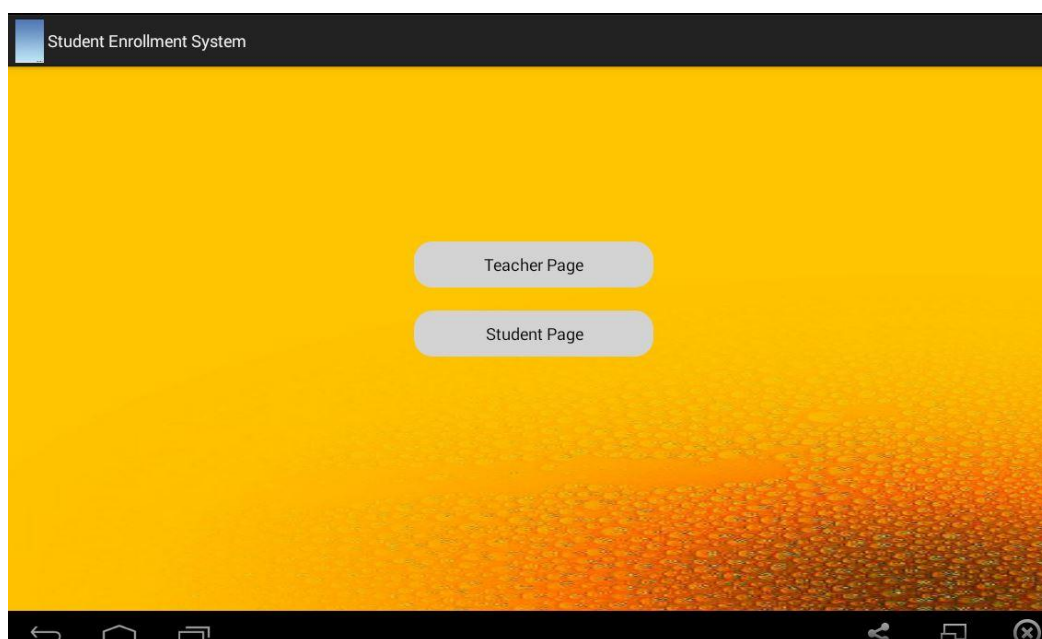


Figure 17: Launch page

At this stage the page was not linked to server. It will only guide the student or teacher to choose page for authentication.

5.2 Student Login Page

The login page interface was designed using android xml. It allows the stu-

dents to enter their registered username and password.

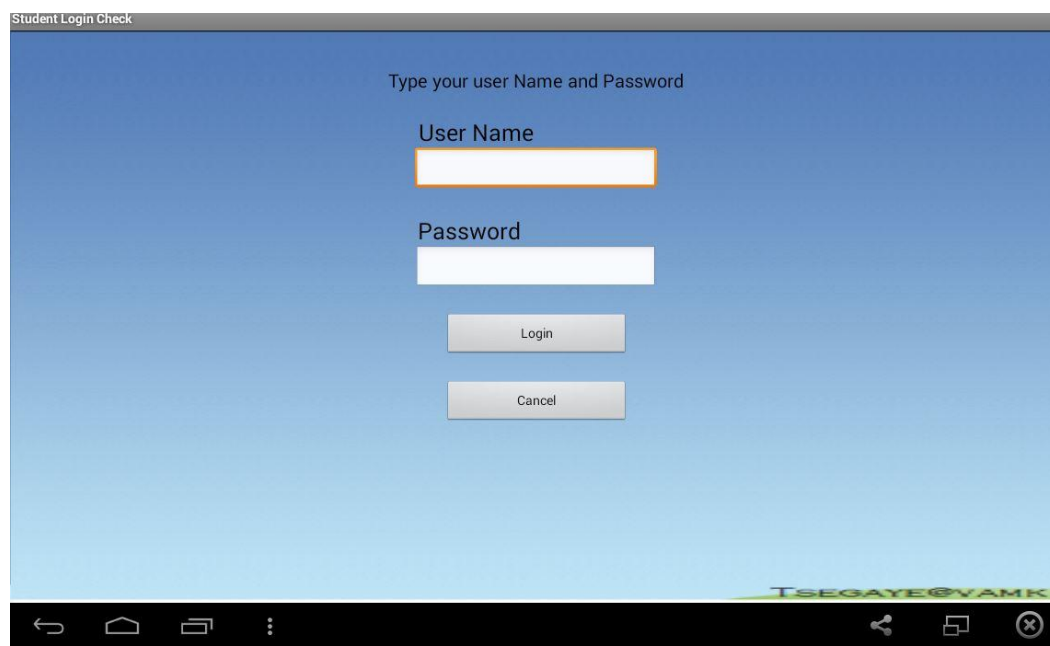


Figure 18: Student login Page.

The page was designed in such a way that all the information that was entered here is checked on the remote database before it proceeds to the next page. It consists of two buttons (login and cancel) and the buttons have clickable attributes which help the buttons to do an action whenever it is clicked. In this page the student should fill the necessary information in order to proceed to the next page.

5.2.1 Student Main Menu Page

The Student Menu list page contains all the important features of the student side application. This was designed using xml. The following figure shows how the student home page looks after the student has logged in on the authentication page.

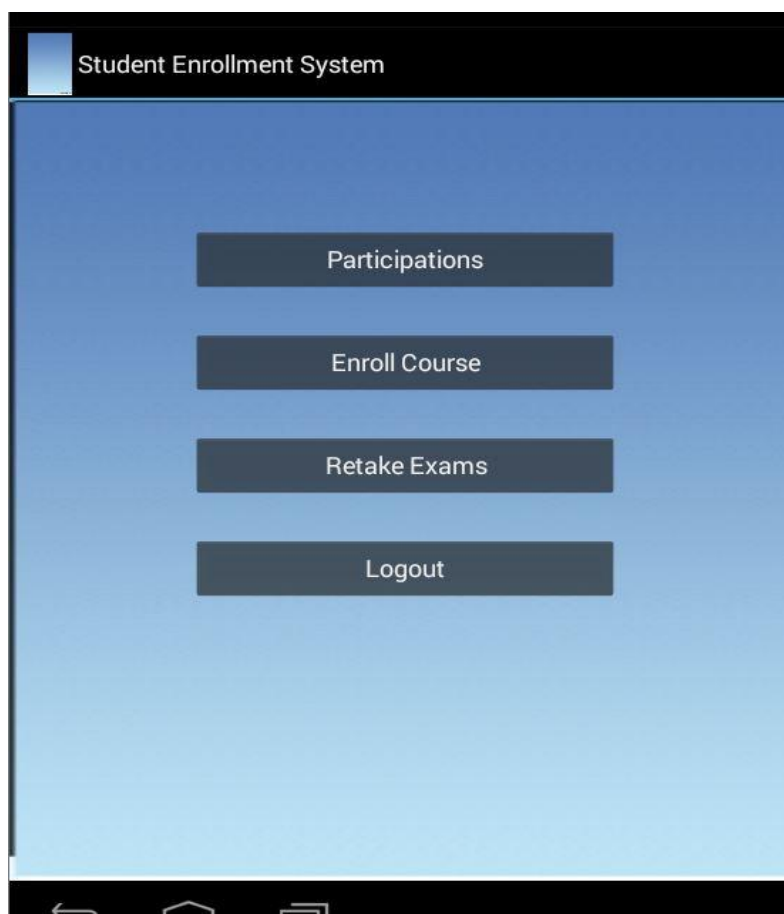


Figure 19: Student menu page.

The page is the communication link between the student and application. The student will be able to access the features by clicking on the buttons on the menu list page. These four buttons are: Participations, Enroll, Retake Exams and Logout. These buttons are the most important buttons for the student to provide the medium to interact with the student side features of the application. Clicking these buttons leads to other pages of the application. The function of the buttons is described down below in detail.

5.2.2 Participations

When Participation button is clicked, it displays the list of enrolled courses. The application will display the list of enrolled courses. Once the courses are enrolled, the application should be able to display the courses. The following figure illustrates this.

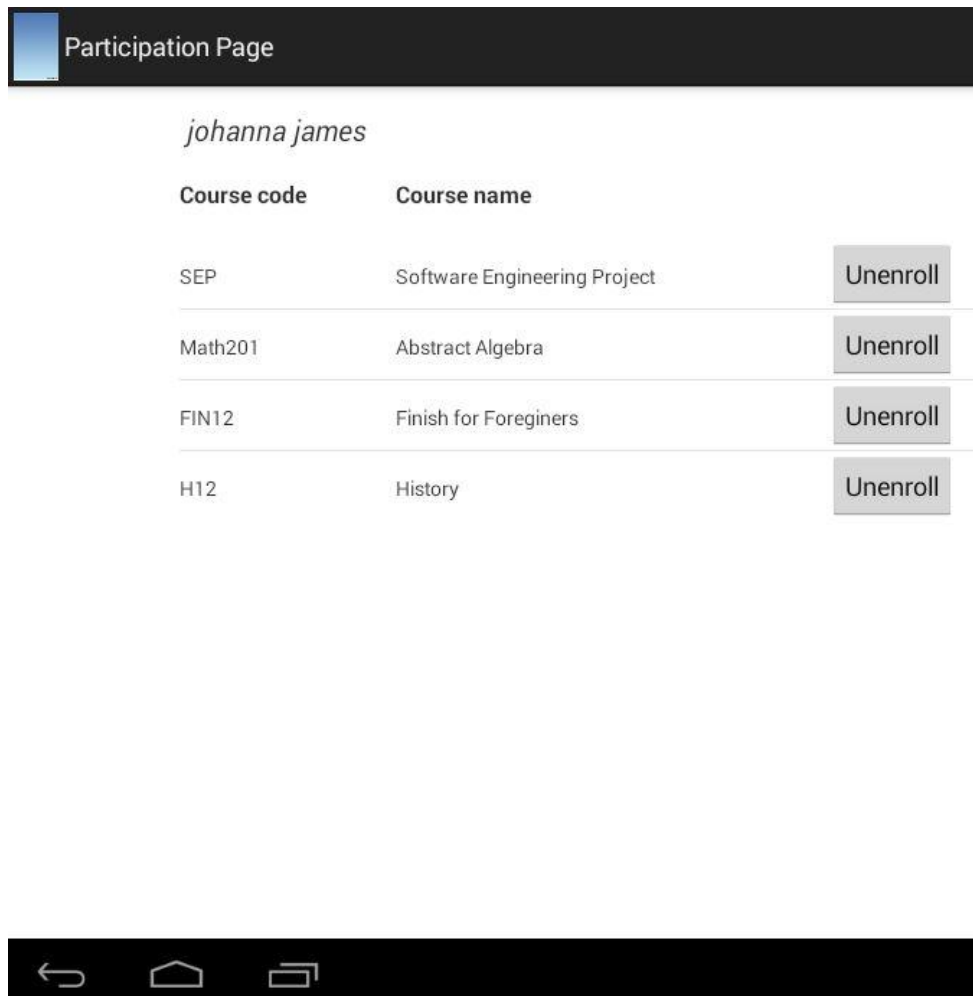


Figure 20: View Participation.

5.2.3 Enroll

This button is linked to a function, which displays the list of available courses. The application will display the available courses. Once the courses are displayed, the student can choose the course he wants to register. The following figure illustrates the view after the Enroll button has been clicked from the student Main page.

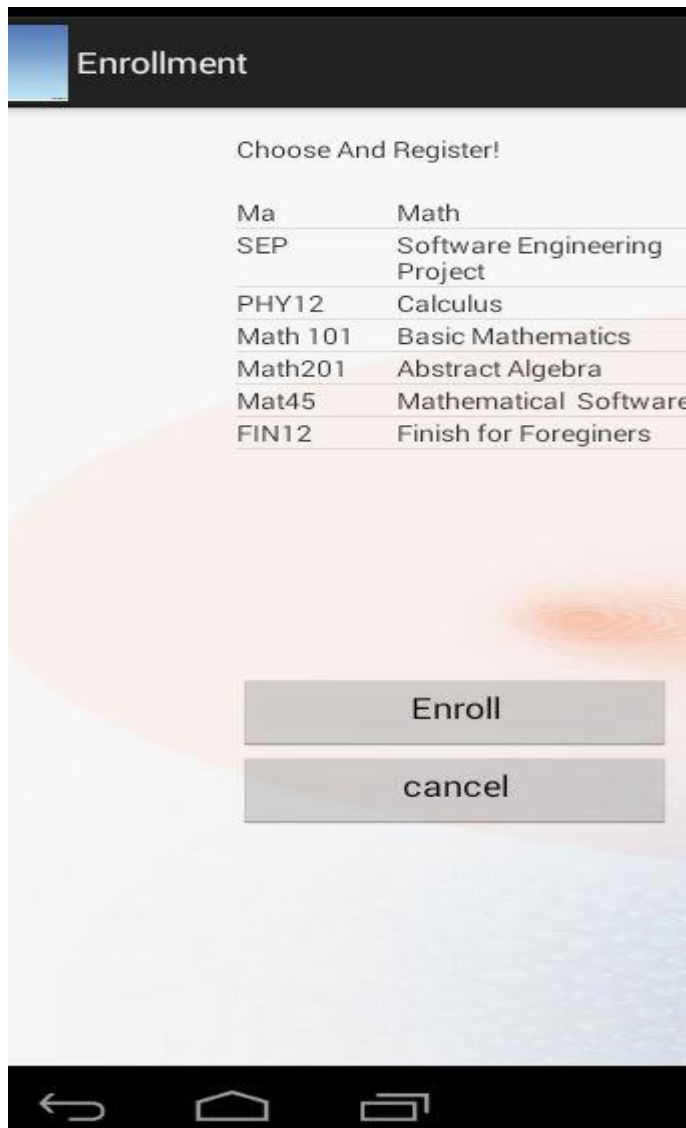


Figure 21: Enroll to Course.

Students can view and register into courses that are available to take in that current semester or period.

In addition, if the user selects the course and clicks the enroll button, then the course will be added to the list of enrolled courses and can be viewed later when the user click the Participations button from the main student page. And if the student clicks the cancel button then the application will direct the student to the student main page.

5.3 Teacher Login Page

The teacher login page was designed using xml. It allows the teachers to enter their registered username and password. The page was made the same way as the student login page. It was designed in such a way that it does not accept null variables or wrong combinations. It consists of two buttons (login and cancel) and the buttons have clickable attributes which help the buttons to do an action whenever it is clicked. On this page the teacher should fill the necessary information in order to proceed to the next page. Figure 22 is the page which appears when the teacher wants to login.

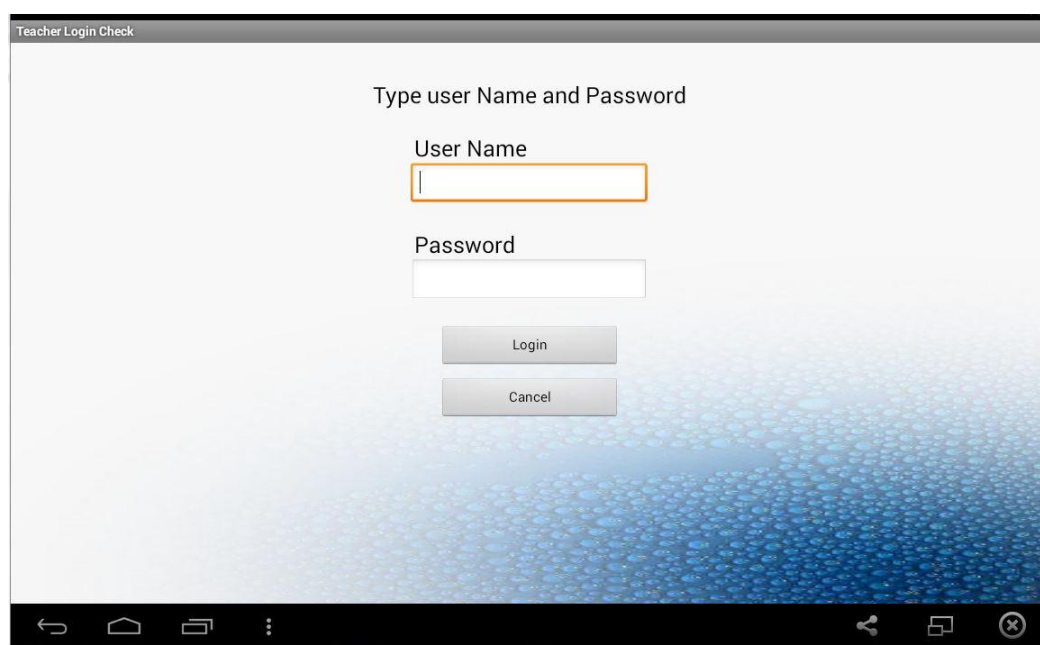


Figure 22: Teacher login page

5.3.1 Teacher Main Menu Page

The Teacher Menu list page contains all the buttons which lead to the important features of the teacher side application. There are four buttons which lead the teacher to other pages as shown in the diagram below. The following figure shows how the teacher home page looks like after the teacher has logged on with the authentication page.

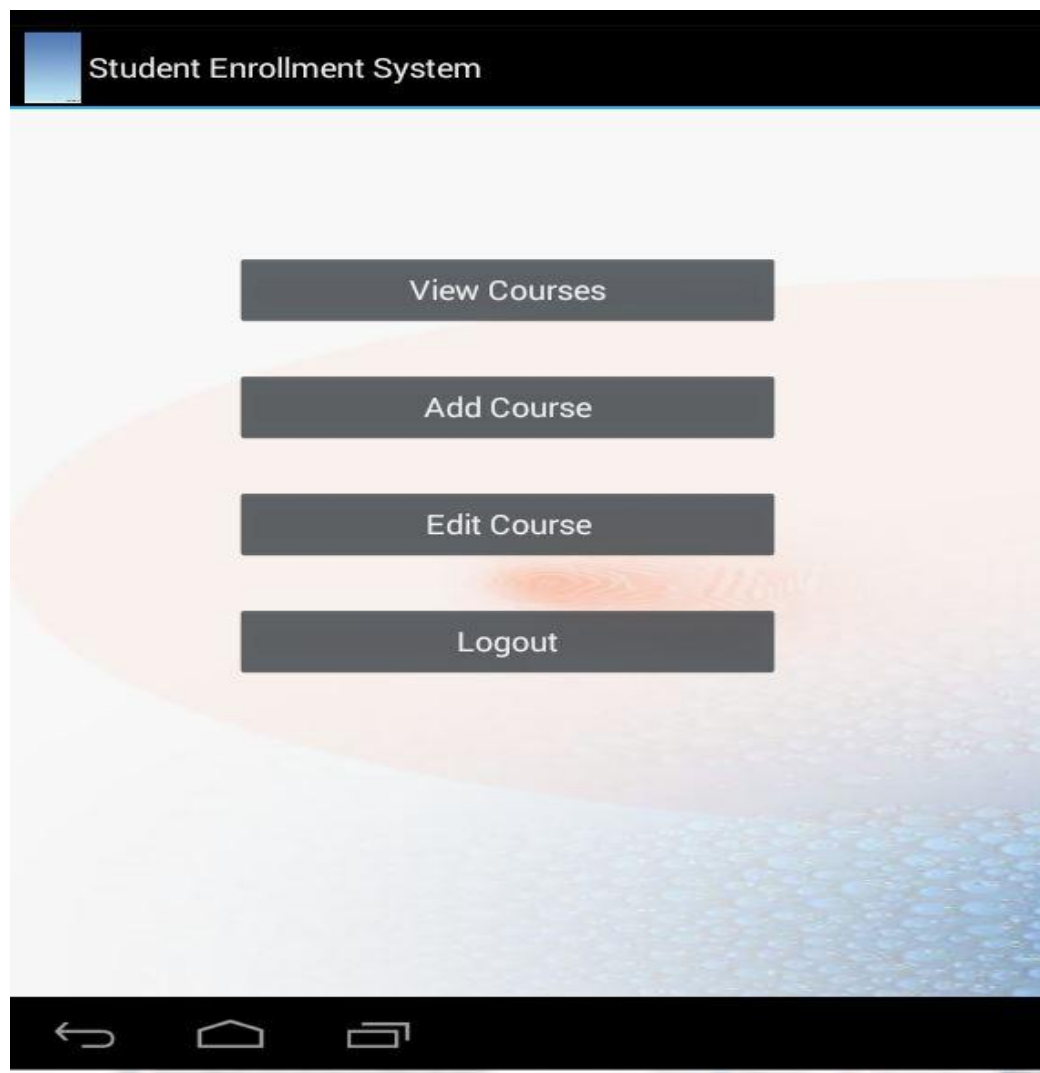


Figure 23: Teacher main menu Page

The page is the communication link between the teacher and application. The teacher will be able to access the main features by clicking on the buttons of the teacher main menu page.

These four buttons are: View courses, Add course, Edit course and logout .These buttons are the most important buttons for the teacher to provide the medium to interact with the teacher side application. When these buttons are clicked, it leads to other pages of the application. The function of the buttons is described down below in detail.

5.3.2 View Course

This button is linked called in a class, which displays the list of courses given by the teacher. The application will display the list of courses and on this page the teacher can make some minor changes in the course details, for example, the course name, course id or course description. The teacher can do it easily in this application and the changes will be directly updated in the database. Figure 24 illustrates when the View courses button is clicked.

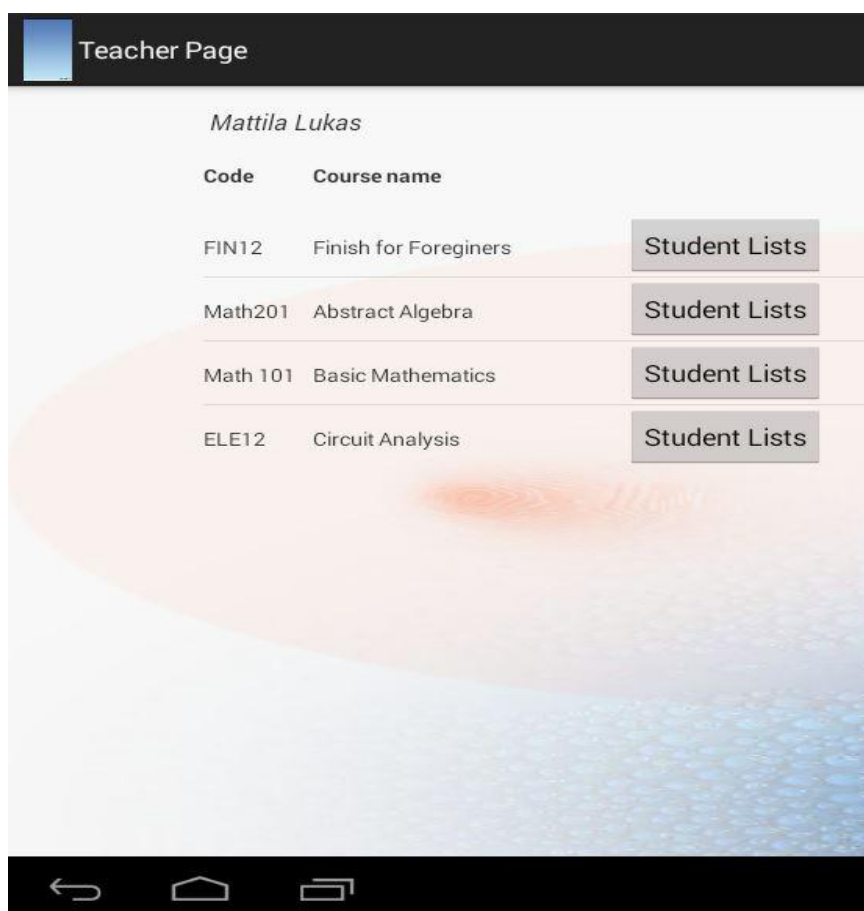
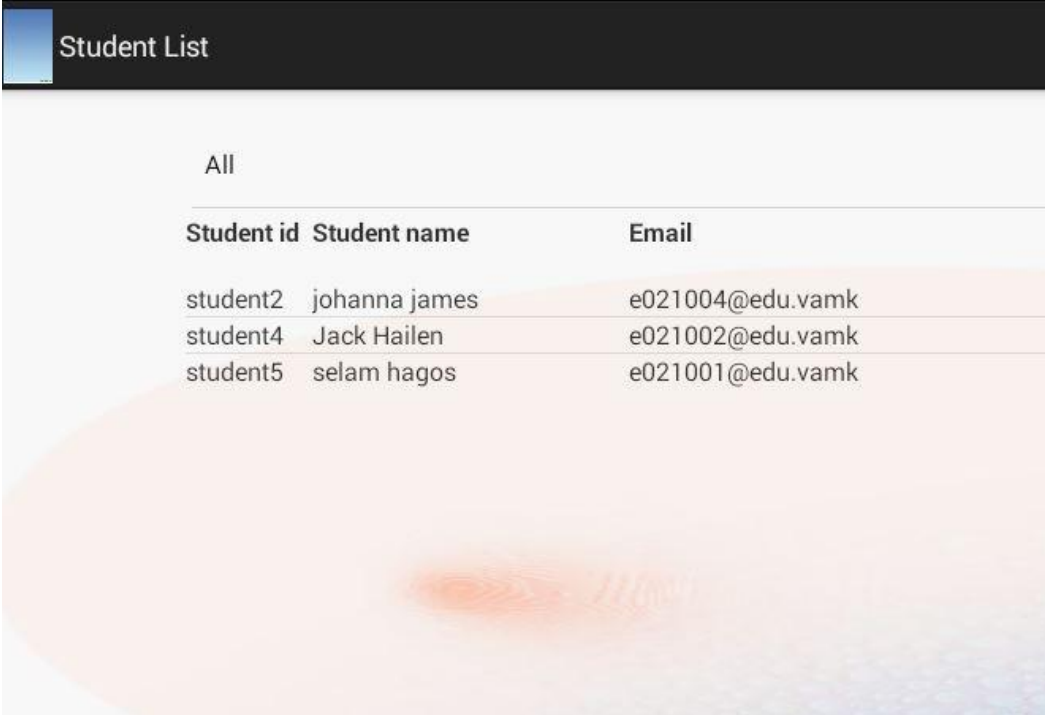


Figure 24: View Courses.

In addition, if the teacher clicks on the student lists button it is linked to a class that brings an environment for the teacher to view the list of students taking that particular course. Figure 25 below shows how the button appears during this event.



The screenshot shows a web interface titled "Student List". Below the title, there is a filter button labeled "All". A table displays the following student information:

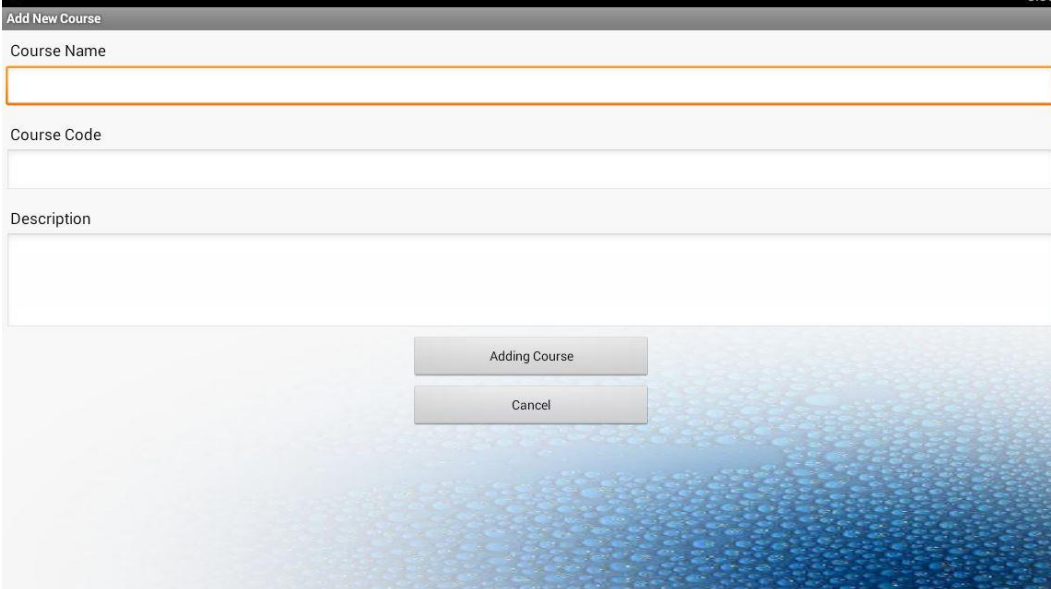
Student id	Student name	Email
student2	johanna james	e021004@edu.vamk
student4	Jack Hailen	e021002@edu.vamk
student5	selam hagos	e021001@edu.vamk

Figure 25: Student lists taking a particular course.

5.3.3 Add Course

On this page the teacher can add new courses that will be given to the students in the current period or the coming period. It was built in a way that the teacher can add the course major details which are course name, course id and course date.

This button is linked to the Add course class, which displays editable text views for important course details. The three editable text views are course name, course ID and Course description. When the Add course button is clicked on the teacher main page, the application displays a page with editable Text Area for the user to enter a valid course details shown in Figure 27 below.



The image shows a web application window titled "Add New Course". It contains three input fields: "Course Name" (with an orange border), "Course Code", and "Description". At the bottom, there are two buttons: "Adding Course" and "Cancel". The background of the window has a blue, textured pattern.

Figure 26: Add Course.

In addition, if the teacher adds the right information on the course details and clicks the submit button, a new activity, which saves a full description of the course details will be launched and will store the data in the database. If the teacher clicks the cancel button then the application will direct to the teacher main page.

6 TESTING

Testing can provide an objective that the application meets the requirements that guided its design and development responds correctly to all kinds of inputs and ensure the correct functionality of the application. The testing done in the Client side of the application was carried out on a Samsung Galaxy S4 phone.

6.1 Invalid Login Credentials

To view the main features of the application, the username and password should be given as shown in Figure 27. In the case of inputting invalid login credentials (missing data or wrong username and password combination), the application will not proceed to the main page, instead the application will display error messages that ask the user to enter the missing data or the correct user credentials.

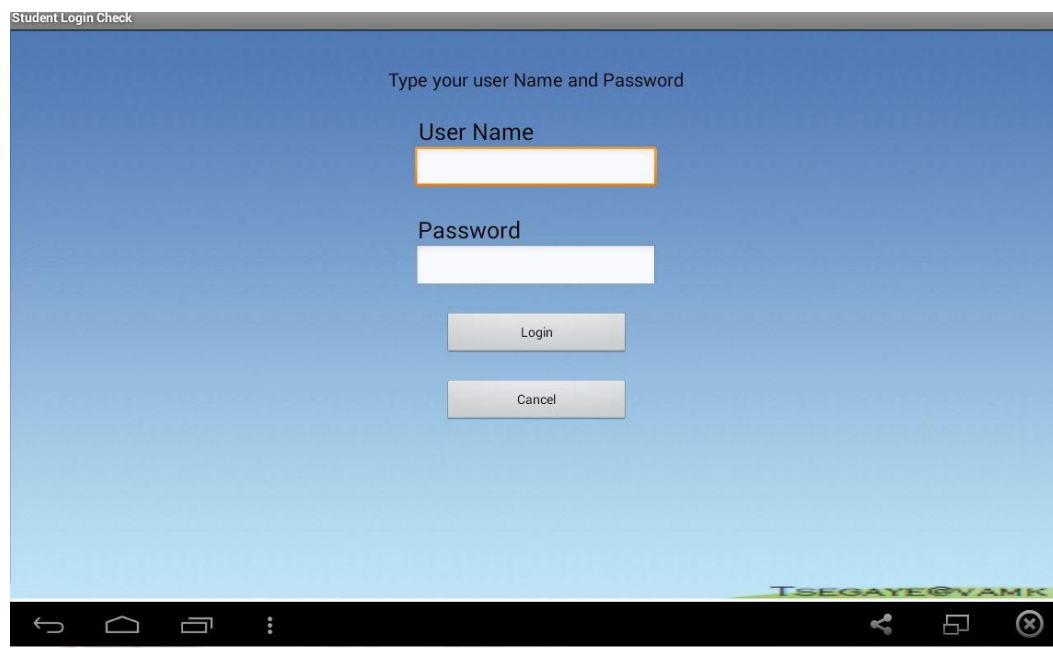


Figure 27.User Login test case

6.2 Student Test Cases

- Test: Main student page

As shown in Figure 28, the student touches the buttons in order to get the desired feature of the application.

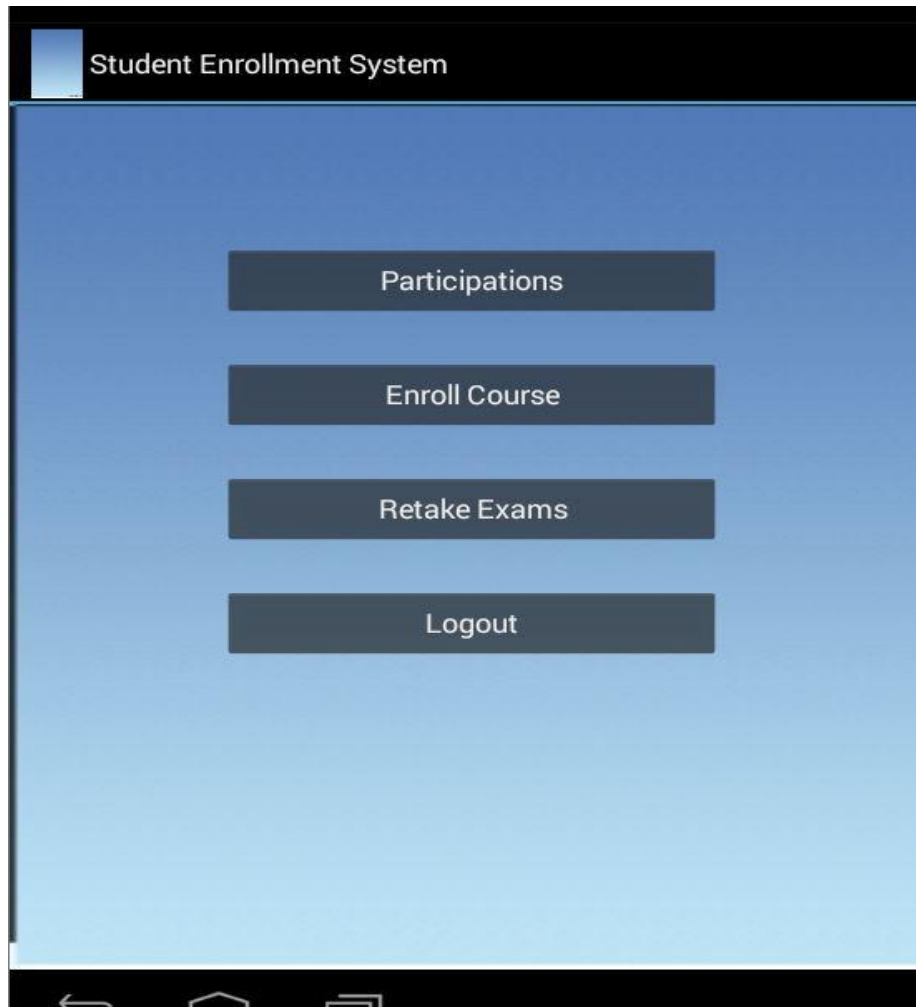


Figure 28: Main student page

- Test: Participation

When the Participation button which is shown in Figure 29 is pressed, then the following page which is shown in figure 30 will be displayed. This test case is carried out assuming that a student has already registered for few courses.



Figure 29: Participations test case

On this page, as shown in Figure 31, the user can also undo the enrollment by pressing Unenroll course button.

- Test: Enroll To Courses

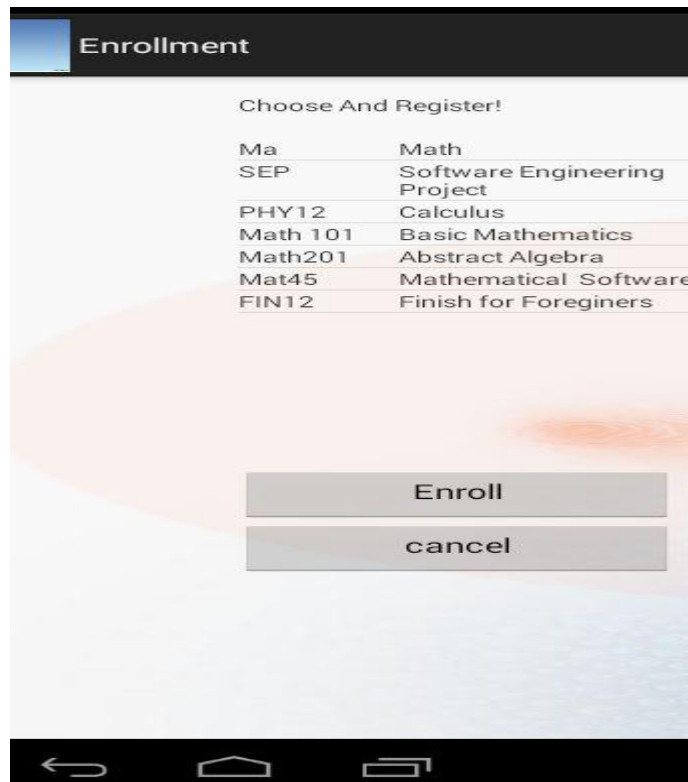


Figure 30: Enroll to course test case.

Figure 30 illustrates the situation when Enroll to courses button is pressed .The student chooses the desired course and clicks Enroll button in order to register into a course.

7 IMPLEMENTATION

This section describes the implementation of the application. An introduction and a description are given to the different resource files that make up the application. Due to the complexity of student enrollment Android application, a description is given only for the major functions and classes that are written by using android programming.

7.1 User Login

In order to use these application users should first login to access the main features of the application. When the user clicks the login button in order to login, the first thing that happens during the login process is that the required necessary fields are checked. If the required fields are empty that will be checked without sending the parameters to the server. If the user enters the username and password, then it will be checked, if the user is already registered in the database. The following code snippet is taken from the student login page which was designed by xml .In the diagram it is shown that this graphical interface consists of two buttons (login and cancel), two text views (Username and Password) and also two edit text views (for username and password).

```

<TextView
    android:id="@+id/userteac"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="25dip"
    android:textColor="#000000"
    android:text="User Name" />

    <EditText
        android:id="@+id/edteacherun"
        android:layout_width="263dp"
        android:layout_height="wrap_content"
        android:inputType="text" />

<TextView
    android:id="@+id/passtu"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="25dip"
    android:textColor="#000000"
    android:text="Password" />

    <EditText
        android:id="@+id/edteacherpw"
        android:layout_width="263dp"
        android:layout_height="wrap_content"
        android:inputType="textPassword" />

<Button
    android:id="@+id/btnLogin"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="25dip"
    android:text="Login" />

<Button
    android:id="@+id/btnLogout"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="25dip"
    android:text="Cancel" />

```

Code Snippet 1: Student login xml

The user inputs the right username and password and clicks the login button. The following code snippet 2 is taken from the main student activity class. It shows that the btnlogin from the student login xml is declared as loginBtn and assigned by calling `setOnClickListener(View.OnClickListener)`.

The `setOnClickListener` will attach the login button on the activity using the `OnClickListener` interface that responds to a call back method `onClick` when clicked.

```
loginBtn = (Button) findViewById(R.id.btnlogin);  
btnCancel = (Button) findViewById(R.id.btnCancel);  
  
loginBtn.setOnClickListener(this);  
btnCancel.setOnClickListener(this);
```

Code Snippet 2: Declaring and assigning buttons

The following code snippet shows that if the user types only the username and clicks the login button then the application will display an error message which states that “Password field is required!”. In addition, it will display the error message for the other fields as well as shown clearly inside the code. If the user enters the right username and password then the `loginProcess` class is called for the login process.

```

case R.id.btnlogin:
    if(new Student_declarations().isNetworkAvailable(this)){
        if ( ( !inputusername.getText().toString().equals("") ) &&
            ( !inputpassword.getText().toString().equals("") ) )
        {
            Log.d("Network", "Available");
            new LoginProcess().execute();
        }
        else if ( ( !inputusername.getText().toString().equals("") ) )
        {
            Toast.makeText(getApplicationContext(),
                "Password field is required !", Toast.LENGTH_SHORT).show();
        }
        else if ( ( !inputpassword.getText().toString().equals("") ) )
        {
            Toast.makeText(getApplicationContext(),
                "User Name field is required !", Toast.LENGTH_SHORT).show();
        }
        else
        {
            Toast.makeText(getApplicationContext(),
                "User Name and Password field are required !", Toast.LENGTH_SHORT).show();
        }
    }
    else{
        Toast.makeText(getApplicationContext(),
            "Network Not Available !", Toast.LENGTH_SHORT).show();

        new Student_declarations().showErrorDialog(this, StudentMainActivity.this);
    }
    break;

```

Code Snippet 3: User login checking credentials

In the login process class the parameters are built to get the user details by making an HTTP request. The PHP code which stored in online server interacts with the database in order to implement the validation. The script returns '1' if the username and password entered by the user from the mobile phone are the same as the one in the Users table in the database. The following code snippet shows what method is done when this login process class is called.

```

class LoginProcess extends AsyncTask<String, String, String> {

    boolean failure = false;
    protected void onPreExecute() {
        pDialog = new ProgressDialog(StudentMainActivity.this);
        pDialog.setMessage("Logging in...");
        pDialog.show();
    }

    protected String doInBackground(String... args) {
        int success;
        String username = inputusername.getText().toString();
        String password = inputPassword.getText().toString();

        try {

            List<NameValuePair> params = new ArrayList<NameValuePair>();
            params.add(new BasicNameValuePair(Student_declarations.TAG_USERNAME, username));
            params.add(new BasicNameValuePair(Student_declarations.TAG_PASSWORD, password));

            JSONObject json = jsonParser.makeHttpRequest(LOGIN_URL, "POST",
                params);

            Log.d("Login process", json.toString());

            success = json.getInt(Student_declarations.TAG_SUCCESS);

            if (success == 1) {
                Log.d("Login Successful!", json.toString());
            }
        }
    }
}

```

Code Snippet 4: Login process class

On the other hand, if the user enters wrong password or username, the PHP script returns '0' and the GUI display, "Incorrect Username or Password". The user is then offered another chance to re-enter the password and the username.

7.2 View Registered Courses Activity Class

This class belongs to the student course package. It displays to the student the list of enrolled courses. The following code snippet is taken from the code. Firstly the parameters are built with the right student id and then it gets the course details by

making an HTTP request to JSON. Later if the JSON response is successful, then it loops through all the courses and stores each JSON item in variable, then it will add them to the array list which can be displayed in the GUI.

```
protected String doInBackground(String... args) {
    try {
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair(
            Student_declarations.TAG_STUDENT_ID, String
                .valueOf(student_id)));

        JSONObject json = jsonParser.makeHttpRequest(COURSE_LIST_URL,
            "POST", params);
        Log.d("All Courses", json.toString());

        if (json.getInt(Student_declarations.TAG_SUCCESS) == 1) {
            Log.d("Success", json.toString());

            JSONArray courseArray = json
                .getJSONArray(Student_declarations.TAG_COURSE);
            for (int i = 0; i < courseArray.length(); i++) {
                JSONObject c = courseArray.getJSONObject(i);
                String courseName = c
                    .getString(Student_declarations.TAG_COURSE_NAME);
                String courseCode = c
                    .getString(Student_declarations.TAG_COURSE_CODE);
                String courseDate = c
                    .getString(Student_declarations.TAG_COURSE_DATE);
                String courseStatus = c
                    .getString(Student_declarations.TAG_COURSE_STATUS);
                int courseId = c.getInt(Student_declarations.TAG_COURSE_ID);

                HashMap<String, String> courseInfo = new HashMap<String, String>();
                courseInfo.put(Student_declarations.TAG_COURSE_NAME,
                    courseName);
                courseInfo.put(Student_declarations.TAG_COURSE_CODE,
                    courseCode);
                courseInfo.put(Student_declarations.TAG_COURSE_DATE,
                    courseDate);
                courseInfo.put(Student_declarations.TAG_COURSE_STATUS,
                    courseStatus);
                courseInfo.put(Student_declarations.TAG_COURSE_ID,
                    String.valueOf(courseId));
                courseList.add(courseInfo);
            }
        }
    }
}
```

Code Snippet 5: View registered courses.

7.3 Add New Course Activity Class

This class belongs to the teacher course package and consists of methods which add courses to the system. The values that the teacher inserts from the GUI are assigned to the variables shown in Code snippet 7 and later used to build the parameters.

```
@Override
protected String doInBackground(String... args) {
    // TODO Auto-generated method stub
    String courseName = nameTxt.getText().toString();
    String courseCode = courseCodeTxt.getText().toString();
    String courseDateFinish = courseFinishDate.getText().toString();
    Date currentDate = new Date();
    SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
    String date = sdf.format(currentDate);
}
```

Code Snippet 6: Assign the course main details

After the course details are assigned, and then it adds the course details by making an HTTP request using the post method. Then if the JSON response is successful, then a course is successfully created.


```

@Override
protected String doInBackground(String... args) {
    try {
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair(Teacher_declarations.TAG_COURSE_ID,
            String.valueOf(course_id)));

        JSONObject json = jsonParser.makeHttpRequest(COURSE_DETAIL_URL,
            "POST", params);
        Log.d("Course", json.toString());
        success = json.getInt(Teacher_declarations.TAG_SUCCESS);
        if (success == 1) {
            Log.d("Success", json.toString());

            String courseCode = json
                .getString(Teacher_declarations.TAG_COURSE_CODE);
            String courseName = json
                .getString(Teacher_declarations.TAG_COURSE_NAME);

            String courseStatus = json.getString(Teacher_declarations.TAG_COURSE_STATUS);
            int courseId = json.getInt(Teacher_declarations.TAG_COURSE_ID);

            courseDetailMap = new HashMap<String, String>();
            courseDetailMap.put(Teacher_declarations.TAG_COURSE_NAME,
                courseName);
            courseDetailMap.put(Teacher_declarations.TAG_COURSE_CODE,
                courseCode);

            courseDetailMap.put(Teacher_declarations.TAG_COURSE_STATUS,
                courseStatus);
            courseDetailMap.put(Teacher_declarations.TAG_COURSE_ID,
                String.valueOf(courseId));

        } else {
        }
    }
}

```

Code Snippet 7: Adding the course main details

7.4 Enroll To Course Activity

The following codes are some functions that were used in this activity that is shown in the Code snippet. It shows the task to enroll students to the course by taking their student ids as parameter of the PHP service.

```

class EnrollCourse extends AsyncTask<String, String, String> {
    private int success = 0;

    @Override
    protected void onPreExecute() {
        progressDialog = new ProgressDialog(EnrollToCourseActivity.this);
        progressDialog.setMessage("Loading...");
        progressDialog.show();
    }

    @Override
    protected String doInBackground(String... args) {

        for (Integer i : addedList) {
            List<NameValuePair> params = new ArrayList<NameValuePair>();
            params.add(new BasicNameValuePair(Student_declarations.TAG_STUDENT_ID,
                String.valueOf(student_id)));
            params.add(new BasicNameValuePair(
                Student_declarations.TAG_COURSE_ID, String.valueOf(i)));

            JSONObject json = jsonParser.makeHttpRequest(
                ENROL_STUDENT_TO_COURSE_URL, "POST", params);
            Log.d("AddedCourseList", json.toString());
            try {
                success = json.getInt(Student_declarations.TAG_SUCCESS);
            } catch (JSONException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        return null;
    }
}

```

Code snippet 8: Enroll Course Class in Enroll Course Activity

The following code snippet is also a part of the Enroll course activity which is implemented for the following function. First, it will get the list of courses which are not enrolled courses by looping in the course array, and then later adds the new course to the course list.

```

class GetCourseList extends AsyncTask<String, String, String> {

    @Override
    protected void onPreExecute() {
        progressDialog = new ProgressDialog(EnrollToCourseActivity.this);
        progressDialog.setMessage("Loading data...");
        progressDialog.show();
    }

    @Override
    protected String doInBackground(String... args) {
        try {
            List<NameValuePair> params = new ArrayList<NameValuePair>();
            params.add(new BasicNameValuePair(
                Student_declarations.TAG_STUDENT_ID, String
                    .valueOf(student_id)));

            JSONObject json = jsonParser.makeHttpRequest(
                ENROL_COURSE_LIST_URL, "POST", params);
            Log.d("Course List", json.toString());

            if (json.getInt(Student_declarations.TAG_SUCCESS) == 1) {
                JSONArray courseArray = json
                    .getJSONArray(Student_declarations.TAG_COURSE);
                Log.d("CourseList array", "" + courseArray.length());

                for (int i = 0; i < courseArray.length(); i++) {
                    JSONObject c = courseArray.getJSONObject(i);

                    int courseId = c.getInt(Student_declarations.TAG_COURSE_ID);
                }
            }
        } catch (JSONException e) {
            Log.e("CourseList", "JSONException: " + e.getMessage());
        }
    }
}

```

Code Snippet 9: Get course list class in enroll Course Activity

7.5 Android Manifest

The manifest file contains all the necessary information about the Android application which includes permissions used by the Android application, all the activities which will be used by the Android application. In this application there are 21 activities included in the manifest xml so that the application will function properly. The following code snippet shows how activities are included inside the Android manifest.


```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.seproject"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="8" />

    <application
        android:configChanges="keyboardHidden|orientation"
        android:icon="@drawable/bird"
        android:label="Winha System" >

        <activity
            android:name="com.StudentEnrollmentApplication.MainActivity"
            android:label="Winha Application "
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- All Courses Activity -->
        <activity
            android:name="com.StudentEnrollmentApplication.ViewAllCourses"
            android:label="All Courses" >
        </activity>

        <!-- Add Course Activity -->
        <activity
            android:name="com.StudentEnrollmentApplication.AddCourse"
            android:label="Add New Course" >
        </activity>

```

Code snippet 10: Android manifest xml

8 CONCLUSION

The student enrollment application meets nearly all of the planned software requirements. Yet, there is still a huge space for possible improvements in the application even in this limited scope of functionality (compared to the main launch application). Since the project was done by using an, own server there is undoubtedly an important data migration to be done when using the school right server which is planned to do it as future work.

Students and teachers can use Student Enrollment Mobile Application with its full functionality more frequently rather than using a computer. By pursuing the Mobile Application users can easily track the course history at any time using a mobile or tablet.

This application will help other mobile applications to be made in the future. VAMK is one of the schools where several future IT professionals study. The school does not have a single mobile application at the moment but since we are living in Application world, VAMK will also have several apps in the near future.

There is not much sense in a relatively small application (as in the current form of student enrollment application) because the greatest improvement could be observed when transitioning between different screens (Activities). A bigger improved application of this API with a lot more of activities and possible changes should therefore be much more beneficial. Moreover, this Android application supports many screen densities and orientations so not only can it be launched on many devices but it also has different graphic layouts, customized for a variety of screen resolutions and dimensions.

Android by itself is not an easy operating system for writing complex applications if one wants to make a full use of its functionality. So far, this application has been developed by only one developer (author of this thesis). However, the launch application (Winha) is definitely too big of a project to be handled by such a small group of developers.

REFERENCES

- /1/ American Dialect Society. Last Accessed December 12, 2014
<http://www.americandialect.org/app-voted-2010-word-of-the-year-by-the-american-dialect-society-updated>
- /2/ A History of Pre-cupcake Code Names. last Accessed February 5,2015
<http://www.androidpolice.com/2012/09/17/a-history-of-pre-cupcake-android-codenames>
- /3/ Shane Conder and Lauren Darcey (2010). Android Mobile
(Application Development From A to Z). Developer eBook
- /4/ The JavaScript Object Notation (JSON) data interchange format.
Last Accessed 2.04.2015.
<http://json.org/>
- /5/ PHP Essentials Neil Smyth, First edition (December 19, 2010). Accessed
16.05.2014
<http://freecomputerbooks.com/PHP-Essentials.html#downloadLinks>
- /6/ Installing the Android SDK. Accessed 2.04.2014
<https://developer.android.com/sdk/index.html>
- /7/ Installing the Android ADT. Last Accessed 2.04.2014
<https://developer.android.com/tools/help/adt.html>
- /8/ Android Emulator. Last Accessed 4.04.2014.
<http://developer.android.com/tools/help/emulator.html>
- /9/ Android Open Source Project. Application Fundamentals – App
Components [online].Google; May 2014. Accessed December 11, 2014
<http://developer.android.com/guide/components/fundamentals.html>
- /10/ Absolute Classes. Last Accessed 3.05.2014
http://www.tutorialspoint.com/uml/uml_class_diagram.htm