ISSN No:-2456-2165

M-Course Registration: A Mobile Students' Course Registration Platform

Fred Sangol Uche
Department of Computer Science
University of the Gambia
Kanifing, Serrekunda

Abstract:- In this paper, we propose an android mobile platform that provides students in colleges and universities the convenience of online course registration. It seeks to reduce the operating cost of tertiary institutions, since all course registration activities can be presented using cheap technology, through screens of electronic devices. Moreover, it is developed in such a way that the activity (course registration process) where students spend the most time, is made accessible in offline mode i.e. without the need for internet connectivity.

So, the goal of my research is twofold: first, to provide an understanding of the analysis and design of such systems and second, to create a new mobile course registration application based on the findings of this research, which provides a digital interaction environment between students and the course registration department at the University of The Gambia.

Keywords:- *Student, Mobile, Course, Registration, Connection.*

I. INTRODUCTION

One of the most special things about course registration through mobile devices is that it intends to bring course registration close to the students who are vastly great users of mobile devices. Mobile technologies are becoming more embedded, ubiquitous and networked with enhanced capabilities for rich social interactions, context awareness and internet connectivity [2]. "Smart" mobile phone seems to be an asset most individuals especially students possess and take almost everywhere with them; it is therefore a highly effective means of bringing information to them faster, easily and on the move [3]. Beyond time and space. The core of course registration through mobile devices is the flexibility, convenience and intuitiveness it provides for students to be able to enrol courses. The m-course registration is identified by different elements that are valuable to stakeholders that interact with it: [4]. The course registration procedure may be manual, web based or online and mobile based. This enables students to register their courses with convenience and without being required to be present on campus. While most educational institutions including the University of The Gambia have now moved from the manual based method to embrace the web based and online course registration method, the possibility of using a mobile application for educational purposes has not been explored fully especially in West Africa. As more and more lecturers in tertiary education experiment with technology, looking for new ways of enhancing their traditional ways of teaching, the need for flexible tools to support well planned and blended learning scenarios is emerging [1]. While the impact of web or online course registration system is appreciated, the advent of mobile communication technology is changing the face of information technology (IT).

Hence, the motivation of this paper is to research on best tools and methodologies to employ in developing mobile course registration platforms.

II. RELATED WORK

M-course registration is an aspect of learning through mobile devices (m-Learning). M-learning as discussed in this study [18] is an extension of E-Learning that came into existence with the emergence of wireless and mobile devices and technologies [19]. The study provides an analysis of principles and patterns of mobile design. It also provides tactics that solve common mobile development problems and based on these findings, an android mobile application is developed to support distance-learning and offer direct communication between students and their teachers through the internet. A similar work was also done in [17] were a Mobile Application Based Course Registration Platform (MABCRP) was designed and implemented to facilitate the process of student's course registration. In this approach, students are able to automatically register expected courses per semester, view all registered courses and also add or delete registered courses. So in our approach, we make use of similar design and implementation techniques but with added functionalities such as offline capability for the activity that takes most time in the course registration process. We also implement LED notifications that will alert students of information like when the course registration deadline is fast approaching. Moreover, we factor in our implementation cases when there is low internet bandwidth.

III. ANALYSIS

This section provides analysis of the mobile application under development. To Start with, it discusses the actors or stakeholders that will be interacting with the systems (only the mobile application for students in this case). It also discusses about the functional and the non-

functional requirements. It presents the use case diagrams and scenarios.

A. Stakeholders

In the requirement analysis there is need to specify who the main stakeholders of the system are. These people interact with the systems (mobile and web app) directly or indirectly. Below are the list of the stakeholders and a description of them is presented:

- **Primary Users**: Ones who are going to be using the application. In this case it's the Students. They will have access to add, edit and delete courses registered.
- **Mobile Application Developers**: Mobile application developers can make use of the mobile application as a reference to develop similar mobile applications for educational institutions.

In terms of this project, the above stakeholders have the same interest in the application, to use a fully-functional application. Thus, they can be defined as one stakeholder, the user.

B. System Requirements

In this section, the system requirements are presented. Requirements can be divided into Functional and Non-Functional requirements. Functional requirements define the functions that a system must be able to perform successfully. Non-Functional requirements define the qualities and criteria that can be used to judge the operation of a system. [9]

- *Functional Requirements*

Functional requirements are supported by non-functional requirements, which impose constraints on the design or implementation. Generally, they are expressed in the form "system must do <requirement>". [9]

The system's functional requirements are given below:

ID	Description	Explanation
FR01	Users must be able to register (sign up)	Users must be able to sign up by filling a form and enter their Mat number, fullname, email, major, password and retype password.
FR02	Users must be able to login to the system.	Users must be able to login to the system with their mat number and password.
FR03	Users must be able to logout of the system	Users must be able to logout of the system at any time by pressing on the logout button in application's menu.
FR04	The system must provide an error recognition message in case of error during the log in or registration process.	The system must show an error message to the user, highlighting the error, if any of the input details of the login/registration form are wrong.
FR05	Users must be able to change their Matriculation number (matno) or password, or any other info entered in process of registration after they are logged in.	Users must be able to change their matno, password or any other personal info after they are logged in, adding their old matno/ password details as well as the new.
FR06	The system must provide an error recognition message in case of error during the detail-changing process.	If old details are wrong or new ones are wrong (empty fields), a recognition message is displayed to the user.
FR06	Users must be able to validate their payment token.	Users must be able to enter a valid registration token after payment to allow them register for courses.
FR07	The system must provide an error recognition message if token is not valid	If the token entered by user is not valid the system should display a recognition message.
FR08	Users must be able to register for courses and send enrolled courses.	Users must be able to register for courses by first selecting a school, then a program and then a course to enrol.
FR09	The system should be able display an error recognition message if registration fails.	If during the registration of course a user does not see the course displayed in the list view, a recognition message should be displayed to the user.

Table 1: Functional Requirements

➢ *Non-functional Requirements*

Generally, non-functional requirement are described in the form "system shall be <requirement>". [9]

Below, the most important classes of the non-functional requirements are described: [10]

- **Usability**: This describes the ease with which a user is able to operate the system.
- **Extensibility**: The capacity of the system to be easily modifiability to meet new requirements.
- **Portability**: The ease in which the system can be migrated from one environment to another.
- **Flexibility**: The capacity of the system to seamlessly exchange data with the user.
- **Reliability**: Is the capacity of a system to perform its functions under expressed conditions for a particular time frame.
- **Security**: The ability of the system to provide authentication through login, password requirements.
- **Performance**: This relates to the operation of the system in terms of speed.
- **Total Cost**: The absolute expense of the project as far as cost and time is portrayed by this quality attribute.

Below, the system's related non-functional requirements are given:

ID	Description	Explanation
NFR01	The graphical user interface must be intuitive for any user to navigate through and screens must have similar look and feel.	The buttons, menus and layouts should be the same in all the screens of the application. The users will execute specific actions in a certain way. They will find the same options and menus in each screen, while pressing the navigation buttons.
NFR02	The system should show clear and detailed notification messages to the user.	The system should display messages in pop-up windows with details regarding the status of any operation. If something needs the attention of the user, the system will display the notification message for it. For example, the process of registering a course or validating token (numeric code used for validating payment).
NFR03	The system must have lack of bugs and inform the user of every wrong operation.	The system must be checked for any possible bugs in its operations before it is released to users. Also, it must provide log error messages (notification messages) to users to inform them of any wrong operation.
NFR04	The system will be able to run on all Android devices.	The system will be developed to run on all Android devices, such as mobiles, tablets or any other device that uses the Android operating system.
NFR05	The system will request a password for each user account.	The system will not display the available features to a user unless he logs in to his account.
NFR06	The system will have fast response time.	The system should provide all its operations very fast. The user must not wait for any operation for a long time. If something needs time to be executed, then a spinning loader will be displayed until the end of the operation.
NFR07	The system must be designed to be able to accept new operations and features.	The system must be designed in such a way that a developer can add new operations and features to the source code.

Table 2: Non-Functional Requirement

IV. USE CASES

A use case is a list of actions defining interactions between actors and a system, to achieve a goal. An actor is a person, organization or external system that plays a role in interacting with the system. [11]

A. Use Case Diagrams

In this project, the primary users from the stakeholders' analysis are the actors. Figure 1 presents the use case diagram of the system. As the diagram shows, there is only one actor – the student.

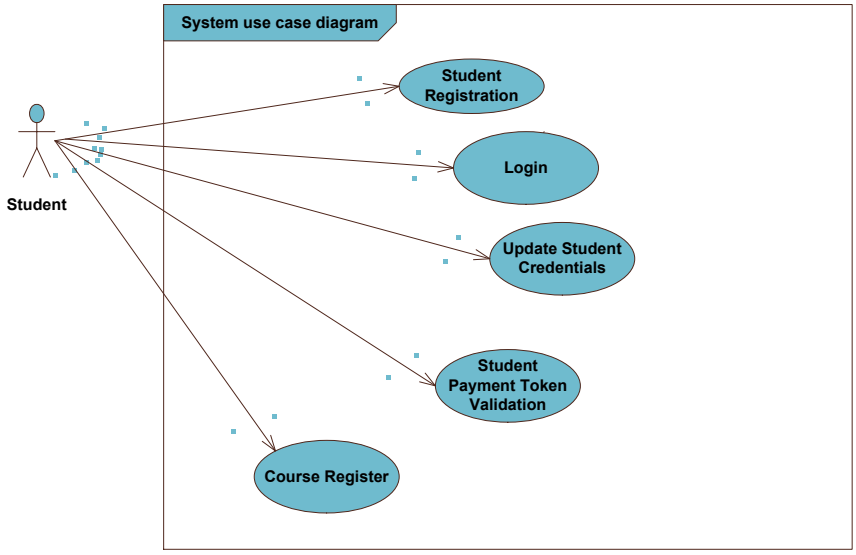

Figure 1: Use case Diagram

V. DESIGN

This section discusses the design of the mobile application. In the design of our platform we employ some key points for the proper design of the activities, supported by the mobile devices [5][6]. In addition, we discuss about the software's architectural design, followed by the design of the software model, web server design and the user interface design.

A. Architectural Design

This project uses the client-server architecture to establish communication between the mobile application and web server (in this case Apache), this is necessary for the exchange of data between the two entities. This communication process is similar to that of the HTTP protocol [12], used by web browsers. Figure 2 the component view of the system, it shows the different entities and processes involved to establish a successful communication between the mobile application and the web server.

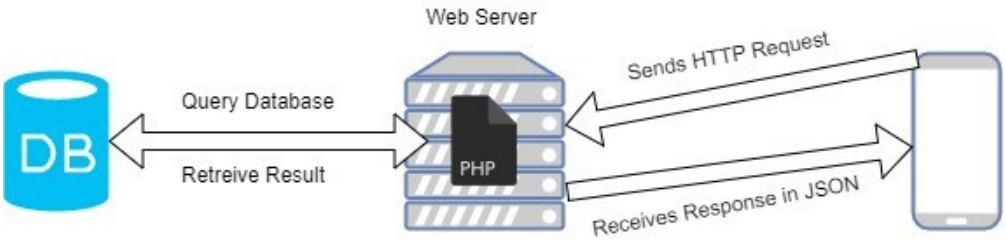

Figure 2: Component View of System

It basically uses HTTP POST/GET requests and responses [12], to retrieve data from web server database to the mobile application and to send data from the application to the web server. The web server consists of PHP files that contains written code to facilitate interaction between the mobile application and the database.

➢ *Technologies Used*

We make use of PHP [13] which is a server-side based language executed on computers with a PHP processor module. To run the PHP code, we make use of Apache web server. After the installation of Apache, the server is able to accept HTTP requests from any mobile device. MySQL Server [14] is also installed to provide a database for storing the data into tables.

Figure 2 shows the web server's response format, which is mostly in JSON or JavaScript Object Notation. Objects or arrays can be constructed where the object in terms of JSON is used for key-value pairs. [15].

B. Web Server Design

This section presents the design of the web server and its implementation. Firstly, we take a look at the structure of the web server, followed by the database structure.

➢ *Web Server Structure*

Figure 3 shows the web server structure. It consists of eight PHP files (dbConnector, CourseDetails, UpdateStudentCredentials, StudentLogin, StudentRegistration, ValidateStudent, Course Register and UtilFunctions). A description of each web server component is given below:

dbConnector: This file as the name implies is responsible for establishing connection to any other php file that needs to access to the database server. For example, the ValidateStudent php file does a HTTP POST to the server. It sends a request to validate the token provided by the student and then the web server sends a response back.

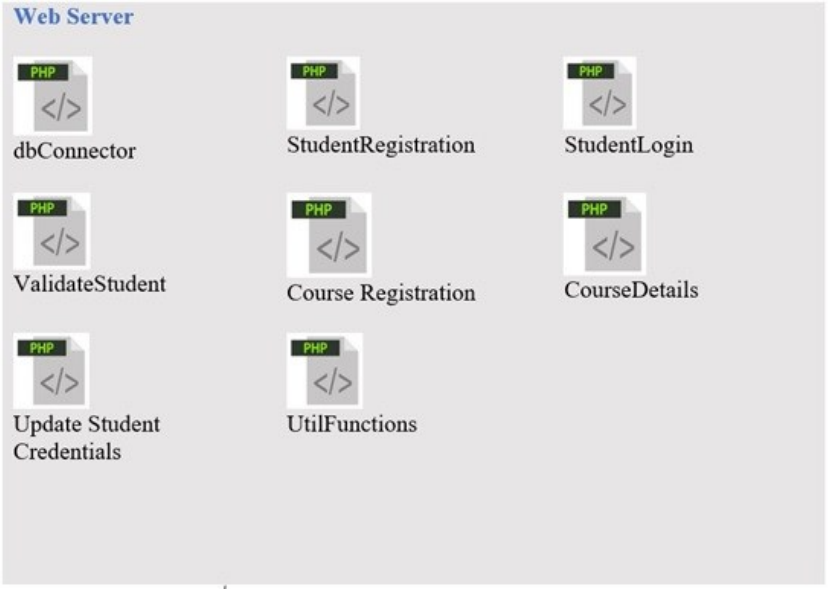

Figure 3: Web Server Structure

Course Details: This file is responsible for retrieving course related information from the database server. Data such as school and program names, courses, the credit hour of courses and lecturer names is downloaded seamlessly to the application at runtime and saved in their corresponding android SQLite tables. Again a valid connection must be established.

Student Registration: It is responsible for allowing students to signup before they access the mobile application. Students have to sign up by providing some necessary data which is sent via http to the web server and saved in the database server. A response message is sent back to the client if the student data is added successfully or if student matno already exist or registered.

Student Login: This is an essential php file. It is responsible for authenticating a student based on the provided matno and password. If login is successful user is redirected to the token validation screen otherwise an error recognition message is displayed.

Validate Student: It is responsible to validate the payment token given to students after payment of tuition fee. It is the only means through which students' can register for courses. Students can only get to the course registration activity if the token for a given semester is valid.

Course Register: This php file gets invoked when student presses on the "Send Data" button after course registration. This sends the data of courses registered from the SQLite local storage DB to the online MySQL DB Server. The data available in the MySQL DB is visible to the administrator of the system via a web application.

Update Student Credentials: This file is responsible for updating the student information such as matno, password and major etc. The data entered by student is sent to the web server and changes are affected and committed to the database.

Util Functions: This is called the function file. It will contain all functions that are de-fined to be used by other php files or classes. The getStudentId and getCourseId functions defined in this file are used during the process of inserting data into the course register table. This is done by making use of the course registration activity in the mobile app.

ISSN No:-2456-2165

➢ *Database Structure*

Figure 4 below shows the entity relation diagram of the MySQL database. We provide a description of the role of each table:

- **Users**: Stores the info of users that will access the web application e.g the faculty officer or registrar.
- **Courses**: Stores all the courses that have been created online via a web application by the faculty officer. The table schools is used to store the school info a student is enrolled in.
- **Programs**: Stores all programs under a specific school.
- **Lecturers**: Stores information related to lecturers.
- **Students**: Stores students' login details.
- **Student Course Registers**: Stores courses registered by students via the mobile application.
- **Academic Semesters**: Stores data regarding academic years and semesters.
- **Courses Available Per Semester**: Stores only courses that are available in a given semester for registration.
- **Student Payment Validations**: Stores all valid payment tokens that are associated to each student for any given semester, after tuition fees are paid. The faculty officer stores data in these tables through the different use cases available in the web application system, except for the student_course_registers table where data gets stored via the mobile app.

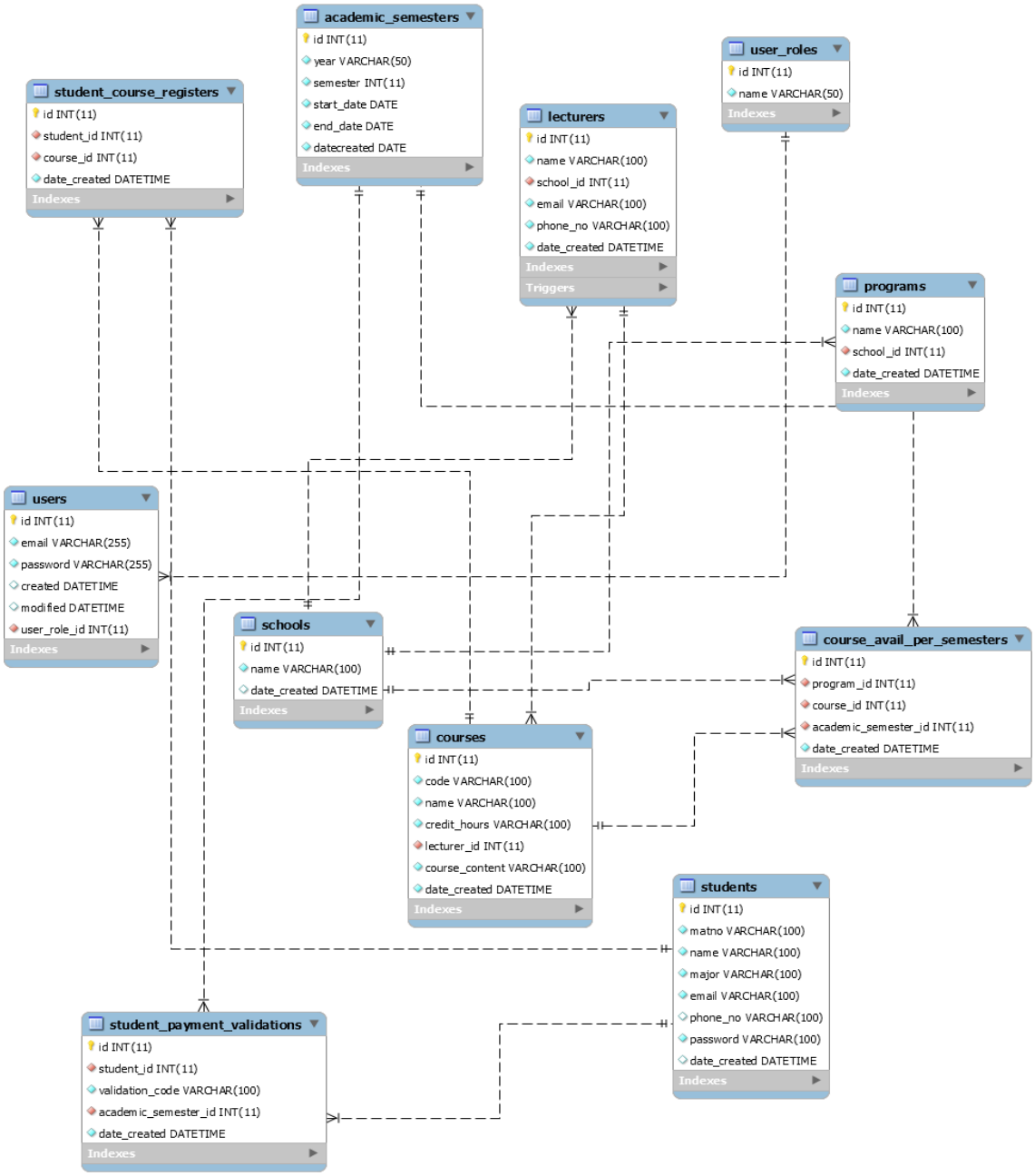

Figure 4: The entity-relation diagram of database

C. User Interface Design

For the application to fulfil the requirements and maximize the ease of use, much accentuation of the design is put on the UI. The standards and guidelines that are referenced in [7] can be identified in the sections below:

➢ *Layout Structure*

The application's screens must follow some standards. Consequently, three layouts are utilized for the application's interface, to support all device sizes and orientations. Every application screen utilizes either the LinearLayout or RelativeLayout, wrapped by ScrollView to provide the above feature. Figure 5 shows the Student login screen displayed in both orientations i.e Portrait and landscape views.

➢ *Themes and Icons*

Same theme is utilized for all screens. The blend of the (blue, white, light grey) colours gives a user a friendly environment. The buttons have either a blue or a grey colour, and colour changes when pressed, to let the user know an action has happened. An Icon is utilized for the main screen window (the login activity). Figure 5 shows the theme of the application and the icons that have been utilized in the student login screen.

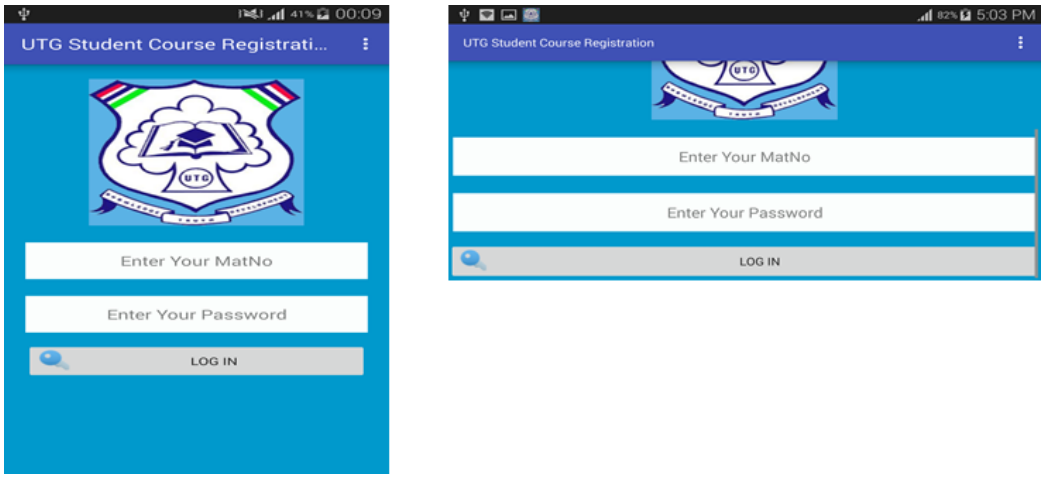

Figure 5: Student login Screen in both orientations with scrolling in landscape

➢ *Navigation and Controls*

Each screen has navigation to other screens. The Login, Register and Course register screens (activities) use option menus to navigate to the main screen. Also, the course registration screen provides a search feature as described in [8] to allow users to easily search for courses. Figure 6 shows an example of the application's controls and navigation in the Validate Student screen and the Course Register screen. From the Course register screen the user can navigate to the login or student register screen. Controls can provide a user with the features of logout, redirect to home screen (Main) or refresh the data in a list.

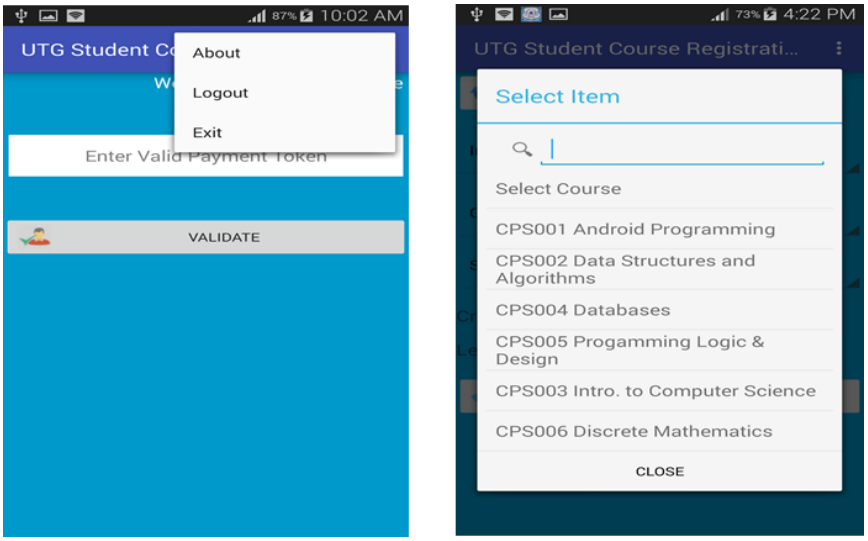

Figure 6: Application's Navigation and Controls

➢ *Feedback and Alert Dialogs*

Feedback and alert dialogs exist in the application. Alert dialog is used to show the progress of an operation. We took a step further in the display of feedback messages by making use of toasts instead of displaying alert dialogs that hide the content of the application. Figure 7 shows a feedback message (in the form of a toast message) when a student attempts to login with a wrong credential i.e an incorrect matno or password. It also shows a display of an alert dialog when a student attempts to delete an already registered course. Additionally, the mobile application has the functionality to display led notification messages, accompanied with a beep sound in this case to alert students on the days left before course registration deadline.

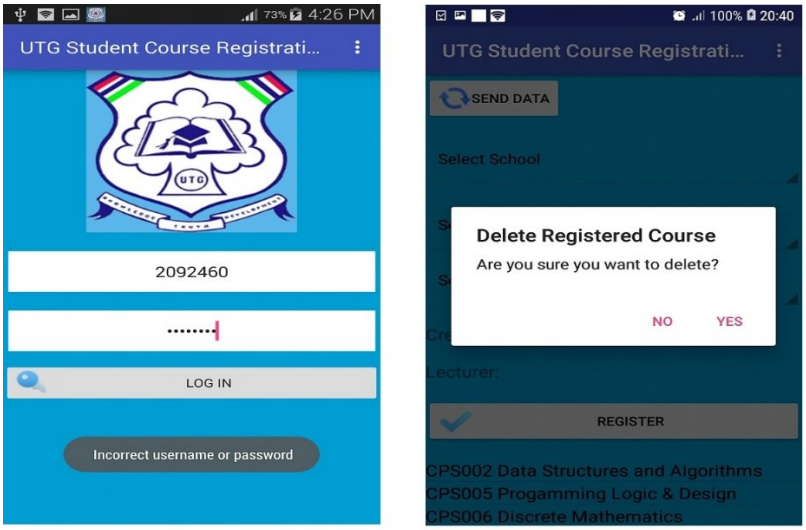

Figure 7: Applications Feedback and Alert Dialogs

VI. IMPLEMENTATION

This section presents the implementation of the mobile app and web server design presented in the **DESIGN** section. It first presents the development of the application and then the web server development. The tools that are been utilized are also discussed.

➢ *Application Development*

This section presents the implementation of the Android application. The written code and techniques of implementation presented here are based on the discussions in the **DESIGN** section.

- *Tools and Libraries Used*

The implementation of the Android application was done completely utilizing the android studio, plugins and Android SDK. All the software tools necessary to design, develop and implement the application are included in this platform [20].

- *Connection Manager Implementation*

The implementation of the manager class is required to keep data running during the application and to check and provide information about status of connection.

Figure 8: Connection Manager Class Diagram

```
package com.example.sangol.myapplication;
import android.content.Context;
import android.net.ConnectivityManager;

public class ConnectionManager {
   String urlPath = "http://192.168.43.187:85/utg/";
   public boolean isNetworkAvailable(Context context) {
```

```
    Runtime runtime = Runtime.getRuntime();
try {

    Process ipProcess = runtime.exec("/system/bin/ping -c 1 192.168.43.187");
    int exitValue = ipProcess.waitFor();
    return (exitValue == 0);

} catch (IOException e)          { e.printStackTrace(); }
catch (InterruptedException e) { e.printStackTrace(); }

return false;

}
```

Listing 1: Connection Manager

- *Data Service Implementation*

The service package provides communication between the application's activities and the web server. This class primarily provides the service of sending data to the web server for insertion into the MySQL database.

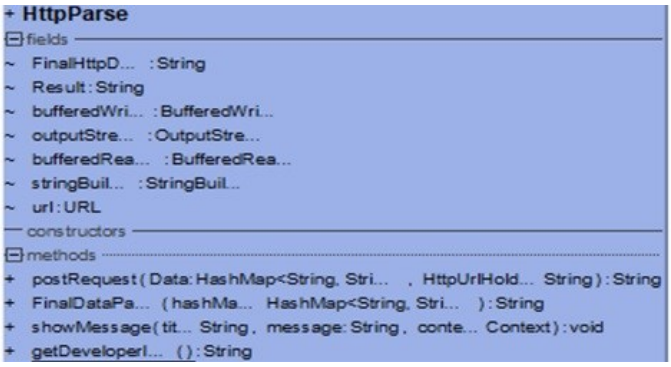

Figure 9: HttpParse service class diagram

- *GUI Implementation*

This section presents how the GUI is implemented, as it is presented in the User Interface **DESIGN** section. Utilizing Android's XML vocabulary, it is easy to design UI layouts and the screen elements they contain, just like you would create web pages in HTML. Each layout file has to contain one root component (ScrollView, RelativeLayout, LinearLayout). Each layout should be save with the .xml extension in res/layout/ directory of the Android's project, so it will compile properly.

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/activity_user_login"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@android:color/holo_blue_dark"
   android:padding="20dp"
   android:orientation="vertical"
   android:layout_gravity="center"
   tools:context="com.example.sangol.myapplication.StudentLoginActivity"
android:overScrollMode="never">


<ScrollView xmlns:android="http://schemas.android.com/apk/res/
android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
```

```
   android:layout_height="match_parent"
   android:layout_weight="1"
   android:fillViewport="true"
   android:overScrollMode="never">

   <LinearLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@android:color/holo_blue_dark"
   android:orientation="vertical"
   android:layout_gravity="center"
      android:overScrollMode="never">

   <ImageView
      android:id="@+id/imageViewLogo"
      android:layout_width="wrap_content"
      android:layout_height="200dp"
      android:layout_centerHorizontal="true"
      android:layout_gravity="center"
      app:srcCompat="@drawable/utg_logo" />



</LinearLayout>
   </ScrollView>
</LinearLayout>
```

Listing 2: Extract of Layout

Listing 2 shows an excerpt of the layout code which is utilized to design the screens. It declares ScrollView as a root component which gives the scroll feature on the screen. ScrollView contains the background property which sets the background of the screen. Inside the root component, a RelativeLayout (or LinearLayout) object, which holds an image object (ImageView) is defined. The image object is the header logo found on the login screen. The @drawable/utg_logo denotes that the image file with the name utg_logo exists in the project folder called drawable. The Match_parent is what makes the view as large as its parent. Wrap_contents is to cause the view to enclose its content.

Listing 10 shows the result of using the objects of Listing 3 in a layout file to create course registration screen. In the course register screen, the user first selects a school, then a program under that school, and then selects a course or courses under that program. After course selection, the course code and lecturer name is displayed on screen. The user can then click on the "Register" button to add a course. The final process is for the user to click on the "Send Data" button to send data from Local SQLite storage DB to the online MySQL DB. Once the sending data process begins, the progress dialog is seen in motion.

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_weight="1"
   android:fillViewport="true"
   android:overScrollMode="never"
   android:layout_alignParentTop="true"
   android:layout_alignParentLeft="true"
   android:layout_alignParentStart="true">

   <RelativeLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_gravity="center"
      android:background="@android:color/holo_blue_dark"
      android:orientation="vertical"
      android:overScrollMode="never">
```

ISSN No:-2456-2165

```
<EditText
        android:id="@+id/matNoView"
        style="@android:style/Widget.Material.EditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/pass"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginBottom="64dp"
        android:layout_marginTop="20dp"
        android:ems="10"
        android:hint="Mat No."
        android:inputType="textPersonName" />


    <ListView
        android:id="@+id/listViewCourses"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/register"
        android:layout_marginTop="24dp"
        android:elevation="1dp" />

    <Button
        android:id="@+id/register"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/lecturerName"
        android:layout_marginTop="18dp"
        android:drawableLeft="@drawable/register"
        android:onClick="registerCourse"
        android:text="Register" />


    <TextView
        android:id="@+id/lecturerName"
        style="@style/TextColor"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/textViewLecturer"
        android:layout_alignBottom="@+id/textViewLecturer"
        android:layout_alignLeft="@+id/creditHours"
        android:layout_alignStart="@+id/creditHours"
        android:elevation="2dp"
        android:gravity="left"
        android:text="Lecturer"
        android:textColor="@android:color/holo_orange_dark"
        android:textSize="18sp"
        android:textStyle="normal|bold"
        android:typeface="monospace" />

    <Spinner
        android:id="@+id/spinnerProgram"
        style="@android:style/Widget.Holo.Light.Spinner"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

ISSN No:-2456-2165

```
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/spinnerSchool"
        android:layout_marginTop="12dp"
        android:spinnerMode="dialog" />

    <com.toptoche.searchablespinnerlibrary.SearchableSpinner
        android:id="@+id/spinnerCourse"
        style="@android:style/Widget.Holo.Light.Spinner"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/spinnerProgram"
        android:spinnerMode="dialog" />
  </RelativeLayout>
</ScrollView>
```

Listing 3: Extract of Objects used in Layouts

For each screen layout, we added menus to provide additional functions. Listing 4 shows the XML code of a menu with about, logout, Update Credentials and exit menu items. For each layout, a menu XML file is created in the Android project's res/menu/ directory, so that it properly compiles.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto">
  <item
    android:id="@+id/action_about"
    android:orderInCategory="100"
    android:title="@string/action_about"
    android:icon="@drawable/icon"
    app:showAsAction="never"
    />

  <item
    android:id="@+id/action_logout"
    android:orderInCategory="100"
    android:title="@string/action_logout"
    android:icon="@drawable/icon"
    app:showAsAction="never"
    />

  <item
    android:id="@+id/action_updateStudentCredentials"
    android:orderInCategory="100"
    android:title="Update Credentials"
    android:icon="@drawable/icon"
    app:showAsAction="never"
    />

  <item
    android:id="@+id/action_exit"
    android:orderInCategory="100"
    android:title="@string/action_exit"
    android:icon="@drawable/icon"
    app:showAsAction="never"
    />
</menu>
```

Listing 4: Application menu example

ISSN No:-2456-2165

In each screen layout we added menus to provide extra functions. Figure 11 shows an example of a menu created in the Course Register screen which can be used by a student to navigate to the Update Student Credentials screen or logout from the application.

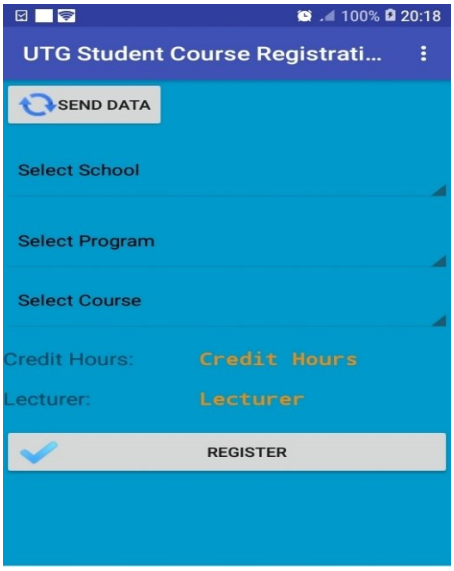

Figure 10: Course Registration screen

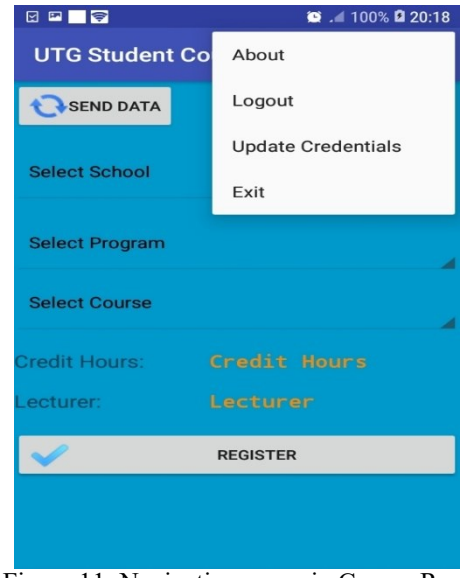

Figure 11: Navigation menu in Course Register screen

- *Using layout Objects in Activities*

This section presents how the activities are using some of layout objects in Listing 3. The first three lines of Listing 5 show how the layout objects are loaded to variables. A setOnClickListener must be used in order to add a click listener to a button. Thus, when a user presses on the button (in this case the "register"button), the code inside the onClick() method gets executed.

```
password = (EditText)findViewById(R.id.editTextPassword);

register = (Button)findViewById(R.id.Submit);
log_in = (Button)findViewById(R.id.Login);

//Adding Click Listener on button.
register.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {

      // Checking whether EditText is Empty or Not
      CheckEditTextIsEmpty();

 if(CheckEditText){

   // If EditText is not empty and CheckEditText = True then this block will execute.
  if(httpParseObj.checkPasswordMatch(password.getText(),reTypePassHolder)) {
      StudentRegisterFunction(matnoHolder, fullnameHolder, majorHolder, emailHolder, PasswordHolder);
   }
else { Toast.makeText(MainActivityRegister.this, "Please confirm passwords are the same", Toast.LENGTH_LONG).show();
}

}

else {


// If EditText is empty then this block will execute .
        Toast.makeText(MainActivityRegister.this, "Please fill all form fields.", Toast.LENGTH_LONG).show();
```

```
    }
```

Listing 5: Loading Layout Objects and Set onClickListener()

Listing 6 shows how data retrieved from the SQLite db is used to create a ListView to display each course registered. Thus, in our application, the data is retrieved in a data variable (typeList<Map<String, String») and then using a SimpleAdapter or ArrayAdapter, they are displayed (list.setAdapter (adapter)) in a list. Figure 12 shows the display of this data on the course register screen.

.

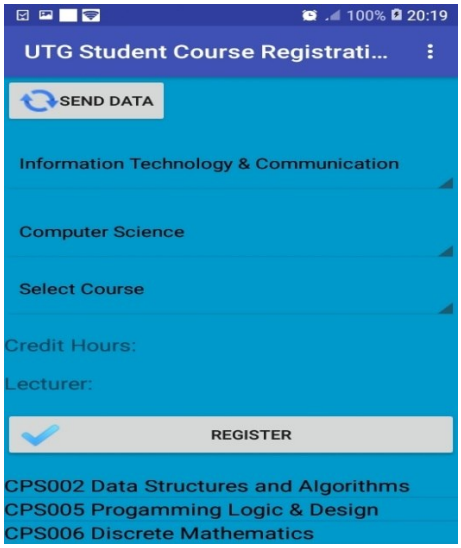

Figure 12: List of Registered Courses

```
//displaying list of registered courses in listview
ArrayList<String> arrayGet = getCourseList();
listView = (ListView) findViewById(R.id.listView1);
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, R.layout.listcourses,
 arrayGet);
listView.setAdapter(adapter);

//filling course spinner dropdownlist
Spinner spinnerSelect;
ArrayList<String> courseList = new ArrayList<String>();
courseList.add("Select Course");
spinnerSelect = (Spinner) findViewById(R.id.spinnerCourse);
courseList.add(json.getString(CourseNameArray));
spinnerSelect.setAdapter(new ArrayAdapter<String>(CourseRegister.this,
android.R.layout.simple_spinner_dropdown_item,  courseList ));
```

Listing 6: Populating ListView with data using Array Adapter

- *Activities Using other Activities and Libraries*

So far, we have mentioned what has been implemented in the GUI, service and activities side. Below we present how all these implementations are combine and run together as one application. First, we define how other activities (Listing 7) can start other activities for a user to view other application screens. The listing below shows how an activity like the CourseRegisterActivity starts.

```
startActivity(new Intent(getApplicationContext(), CourseRegisterActivity.class))
```

Listing 7: Starting an Activity

Secondly, we need to define how the activities can have access to the web server. Listing 8 shows how an activity tries to access the web server and sends course register data. The ConnectionManager and the ProgressDialog object is created. The ProgressDialog or ProgressBar object is used to display a loading dialog to inform the user that the activity is processing data. A background thread which runs in code is created to retrieve data from the web server. ProgressBar stops spinning when the job being executed in the background is done processing. The ConnectionManager object checks if there is network connectivity, if this is true, the ServiceFunction object calls the onPostExecute method to send the data asynchronously to the web server in HashMap data structure.

Figure 13 shows ProgressDialog (Loading message) when the activity tries to send course register data to the web server. Listing 9 shows how the course register activity retrieves schools, programs and courses related data from database via the web server. The queried data is then serialized in a JSON object with help of the json_encode php function in Listing 15. When the response is sent by the web server via http, the JSON object is deserialized by been converted to a JSON Array. After deserializing, the list of schools, programs and courses are populated in their respective spinners (dropdown lists). As mentioned in the design chapter, a JSONParser method which is developed using the Android Volley API [16] handles this. The Android Volley API acts as a service that facilitates the sending of data in the form of a request to the web server, and the retrieving of data in the form of a JSON response.

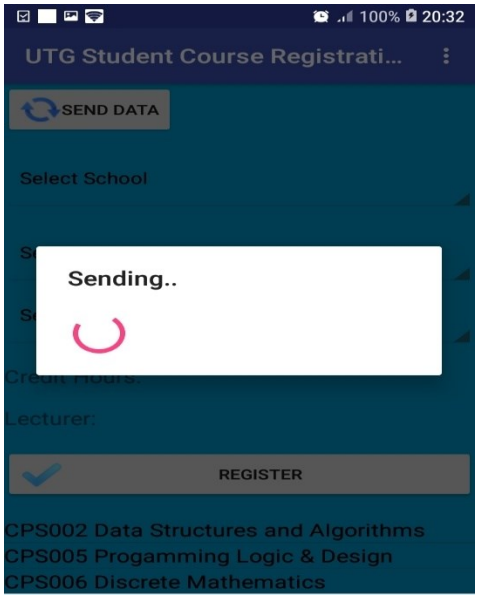

Figure 13: ProgressDialog while sending data to web server

```
public void StudentCourseRegister(final String matno, final String[]
courseCode){

    class StudentCourseRegisterClass extends AsyncTask<String,Void,String> {
        //final ProgressBar Bar = (ProgressBar)findViewById(R.id.progressBar1);
        protected void onPreExecute() {
            super.onPreExecute();

            progressDialog = ProgressDialog.show(CourseRegister.this,
"Sending..",null,true,true);
        }

        protected void onPostExecute(String httpResponseMsg) {

            super.onPostExecute(httpResponseMsg);
            progressDialog.dismiss();

            Toast.makeText(CourseRegister.this,httpResponseMsg.toString(),
Toast.LENGTH_LONG).show();

        }
protected String doInBackground(String... params) {

            for (int i=0;i< courseCode.length;i++) {
                hashMap.put("matno", matno);
                hashMap.put("courseCode", courseCode[i]);

            finalResult = httpParseObj.postRequest(hashMap, HttpURL);

            }
            return finalResult;
        }
    }

    StudentCourseRegisterClass studentCourseRegisterObj =
    new StudentCourseRegisterClass();

    studentCourseRegisterClass.execute();
}
}
```

Listing 8: Activity sends data to Web Server

```
private void getCourseData() {
   StringRequest stringRequest = new StringRequest(
Request.Method.POST,conManager.urlPath+"CourseDetails.php",
         new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
               JSONObject j = null;
               try {
                  j = new JSONObject(response);
                  result = j.getJSONArray(JSON_ARRAY); //allows you to get Json object data by key
                  httpParseObj.showMessage("ResponseCourses:",
response.toString(),CourseRegister.this);
                  //showMessage("Result", result.toString());


                  insertIntoCourses(result);
               } catch (Exception e) {
                  httpParseObj.showMessage("ExceptionCourses", e.toString(),CourseRegister.this);
                  e.printStackTrace();
               }
            }
         },
         new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
               httpParseObj.showMessage("ExceptionVolley", error.toString(),
CourseRegister.this);
            }
         }){
      @Override
      protected Map<String, String> getParams() {
         Map<String, String> params = new HashMap<>();
         params.put("schoolName", "courseSchool");
         params.put("programName", "programSchool");

         return params;
      }

   };
   RequestQueue requestQueue = Volley.newRequestQueue(this);
   requestQueue.add(stringRequest);
}
```

Listing 9: Activity Retrieves data from Web Server

➢ *Web Server Implementation*

The implementation of the web server is presented in this section. Based on the design described in the **DESIGN** section, the code written is presented. In the *Web Server Design* section, it is discussed that the web server consists of eight PHP files. So, in this section we present the implementation. Listing 10 shows an excerpt of the dbConnector PHP file. In this file the host name, database name, host user and password are defined. All other php files include this file for authorization in order to send data to or retrieve data from the web server.

```
<?php

//Define your host here.
$HostName = "localhost";

//Define your database username here.
```

```
$HostUser = "root";

//Define your database password here.
$HostPass = "xxxx";

//Define your database name here.
$DatabaseName = "student_course_reg";

$con= new mysqli($HostName, $HostUser, $HostPass, $DatabaseName);

if ($con->connect_error) {

   die("Connection failed: " . $con->connect_error);
}

?>
```

Listing 10: Extract of dbConnector PHP file

Listing 11 shows the extract of the StudentRegistration PHP file. Once there is an established connection to the DB server, the data entered during student registration using the mobile app is transferred to the web server. Once it is received we first check whether student is already registered in the database. If student is not registered then students' registration information is saved in the DB.

```
<?php
if($_SERVER['REQUEST_METHOD']=='POST'){
include 'dbConnector.php';

 $matno = $_POST['matno'];
 $fullname = $_POST['fullname'];
 $major = $_POST['major'];
 $email = $_POST['email'];
 $password = MD5($_POST['password']);

 $CheckSQL = "SELECT * FROM students WHERE matno='$matno'";

 $check = mysqli_fetch_array(mysqli_query($con,$CheckSQL));

 if(isset($check)){

 echo 'Student Already Registered';

 }
else{
$Sql_Query = "INSERT INTO students (matno,name,major,email,password,
date_created) values ('$matno','$fullname','$major','$email','$password',now())";

 if(mysqli_query($con,$Sql_Query))
{
 echo 'Registration Successful';
}
else
{
 echo 'Something went wrong';
 }
 }
}
 mysqli_close($con);
?>
```

Listing 11: Extract of StudentRegistration php file

Listing 12 shows the extract of the StudentLogin PHP file. Once there is an established connection to the database server the information entered during the login process is transferred to the web server. When a user enters a matno and password and presses on the login button, the matno and password values are sent to this file located in the web server via HTTP POST. Then a select query with a where clause is executed that checks for a match based on matno and password provided. If there is a match, a string message is returned otherwise a login error message.

```
<?php

if($_SERVER['REQUEST_METHOD']=='POST'){

include ('dbConnector.php');

 $matno = $_POST['matno'];
 $password = MD5($_POST['password']);
 $Sql_Query = "select * from students where matno = '$matno' and password = '$password' ";

 $r = mysqli_query($con,$Sql_Query);

 while($row = mysqli_fetch_array($r))
{
  $result = $row['name'];  //returns student name if login credentials is valid else an empty string
}
echo $result;
}
mysqli_close($con);
?>
```

Listing 12: Extract of StudentLogin PHP file

Listing 13 shows the extract of the ValidateStudent PHP file. Once there is an established connection to the DB server, the data entered by the student during token using the mobile app is sent with the help of this PHP file located in the web server. A select query with a where clause is executed which checks for a match based on token provided, taking into consideration the current semester. If there is a match, a string message is returned otherwise a validate error message.

```
<?php

if($_SERVER['REQUEST_METHOD']=='POST'){

include 'dbConnector.php';

$matno = $_POST['matno'];
$code = $_POST['code'];

$Sql_Query= "select matno from students,student_payment_validations,
academic_semesters where matno = '$matno' and validation_code = '$code' and
students.id = student_id and student_payment_validations.academic_semester_id = academic_semesters.id
and CURDATE() between academic_semesters.start_date and academic_semesters.end_date";

$check = mysqli_fetch_array(mysqli_query($con,$Sql_Query));

if(isset($check)){

   echo "Data Matched";
}
else{
   echo "Invalid Token Please Try Again";
}
```

```
}else{
   echo "Check Again";
}
mysqli_close($con);

?>
```

Listing 13: Extract of ValidateStudent php file

Listing 14 shows the extract of the Course Registration PHP file. Once a connection to the database server is established, the information entered is first stored locally in an SQLite database. When it's time to send the data, the "Send Data" button is pressed and the data is transferred from the local database to the web server via HTTP POST request. The Course registration file contains an insert query that allows data to be stored in database.

```
<?php
include 'dbConnector.php';

$matNo = $_POST['matno'];

$studentId = getStudentIds($matNo)

$courseCode = $_POST['courseCode'];

$courseId = getCourseIds($courseCode);

//add a column for in StudentCourseRegister table

$sql = "insert into student_course_registers (student_id,course_id,date_created) values
('$studentId','$courseId',now())";
if(mysqli_query($con,$sql)){
   echo 'success';
}
mysqli_close($con);


?>
```

Listing 14: Extract of CourseRegistration PHP file

Listing 15 shows the extract of the CourseDetails PHP file. Once a connection to the database server is established, the role of the SQL query defined in this file is to select the schools, programs and course names under those programs and saved them in JSON format. These data gets populated in the course registration school, program and course spinners. It retrieves this data by performing a select query and storing the data in JSON format in the form of key and value. When the JSON data reaches the spinner controls, with the help of the android Volley API, the data is deserialized and attached to the respective spinners.

```
<?php

include 'dbConnector.php';

$schoolName= $_POST['schoolName'];
$programName= $_POST['programName'];

// select list of schools
if ($schoolName=="school") {

   $sql = "select id, name from schools";

   $r = mysqli_query($con,$sql);
   $result = array();
   while($row = mysqli_fetch_array($r)){
      array_push($result,array('id'=>$row['id'],'name'=>$row['name']
```

```
        ));
    }

}
// select list of programs. This data is invoked and then stored in local SQLite DB
if (($schoolName=="school") && ($programName=="program")) {

    $sql = "select id,name,school_id from programs";

    $r = mysqli_query($con,$sql);
    $result = array();
    while($row = mysqli_fetch_array($r)){
        array_push($result,array('id'=>$row['id'],'name'=>$row['name'],'school_id'=>$row['school_id']
        ));
    }

}

// select list of courses and programs available for current semester
if (($schoolName=="courseSchool") && ($programName=="programSchool")) {

$sql = "select course_avail_per_semesters.id as id, programs.name as  program,courses.code as code, courses.name as
cName,courses.credit_hours as creditHours,lecturers.name as lecturer, academic_semesters.end_date as end_date from courses,
lecturers, programs, course_avail_per_semesters,academic_semesters where courses.id =course_avail_per_semesters.course_id
and course_avail_per_semesters.program_id = programs.id and lecturers.id = courses.lecturer_id and academic_semesters.id =
course_avail_per_semesters.academic_semester_id and CURDATE() between academic_semesters.start_date  and
academic_semesters.end_date";

    $r = mysqli_query($con,$sql);
    $result = array();
    while($row = mysqli_fetch_array($r)){
        array_push($result,array('id'=>$row['id'],
            'program'=>$row['program'],'code'=>$row['code'],
            'cName'=>$row['cName'],'creditHours'=>$row['creditHours'],
            'lecturer'=>$row['lecturer'],'end_date'=>$row['end_date']
        ));
    }

}

echo json_encode(array('result'=>$result));
mysqli_close($con);?>
```

Listing 15: Extract of CourseDetails PHP file

Listing 16 shows the extract of the UpdateStudentCredentials PHP file. Again, once there is an established connection to the database server. The data entered using the mobile application is retrieved and sent to the web server via HTTP by making use of the data service implementation.

```php
<?php
if($_SERVER['REQUEST_METHOD']=='POST'){

   include 'dbConnector.php';

   $matnoHolder = $_POST['matnoHolder'];
   $matno = $_POST['matno'];
   $major = $_POST['major'];
   $pass = MD5($_POST['pass']);

  $Sql_Query = "UPDATE students SET matno= '$matno', major = '$major', password = '$pass' WHERE matno = $matnoHolder";

   if(mysqli_query($con,$Sql_Query))
   {
      echo 'Record Updated Successfully';
   }
   else
   {
      echo 'Something went wrong';
   }
}
mysqli_close($con);
?>
```

Listing 16: Extract of UpdateStudentCredentials PHP file

Listing 17 shows the extract of the UtilFunctions PHP file. In order for the functions defined in this file to be executed, they will have to be called from other php files such as the course registration. As always, a connection to the server has to be established first then the SQL queries in this functions gets executed and the return values send back to the file or class that made the function call.

```php
<?php

include('dbConnector.php');


function getCourseIds($courseCode) {
   $Sql_Query = "select id from courses where code = '$courseCode'";
   $r = mysqli_query($con,$Sql_Query);

   while($row = mysqli_fetch_array($r)){
      $result = $row['id'];
   }
   return $result;
}


function getStudentIds($matno) {
   $Sql_Query = "select id from students where matno = '$matno'";
   $r = mysqli_query($con,$Sql_Query);

   while($row = mysqli_fetch_array($r)){
      $result = $row['id'];
   }
   return $result;
}

?>
```

Listing 17: Extract of UtilFunctions PHP file

VII. CONCLUSION

Having discussed both the theoretical and practical aspects of the development of the mobile course registration platform, as well as its benefits concerning course registration in universities and tertiary institutions, we provide a general overview of our research. To begin with, we delved into understanding the analysis and design of such systems. Having taken these factors into consideration, we put them to work by developing an android based mobile course registration application, which is meant to be used by students anywhere (ubiquitous) at any given time, provided that the deadline for course registration is taken into consideration. In the development process, we employ the agile software model which is one of the most used software development methodologies because of its flexibility.

In conclusion, the developed tool will be very useful for other universities and tertiary institutions to acquire and adapt, as it will make the process of student course registration more effective and efficient.

- *Future Work*

The main purpose of this research paper is to provide insight and understanding about the best tools and methodologies to use in the implementation of an android mobile course registration application. This will provide developers with a first-hand knowledge in the implementation of such platforms. So in subsequent iterations, students will be able to pay their tuition fee from the platform, this will also include the automatic sending of emails to students with a valid payment token to be used in the course registration process. There is also the need for the functionality that allows students to view their transcripts after every semester. Additionally, since most courses have prerequisite courses, there is need for the functionality that allows students to only register courses provided they have already done the corresponding prerequisite courses.

ACKNOWLEDGEMENT

Special thanks go to Prof. Cheikh Ba — Lecturer School of Applied Sciences and Technologies at Gaston Berger University for taking time out of his busy schedule to guide me through process of writing this paper and making sure that it conforms to academic standards.

REFERENCES

[1]. Katerina Georgouli, Ilias Skalkidis, Pedro Guerreiro. "A Framework for Adopting LMS to Introduce e-Learning in a Traditional Course". International Forum of Educational Technology & Society, Vol. 11, No 2, 2008. pp. 227-240.
[2]. Laura Naismith, Peter Lonsdale, GiasemiVavoula and Mike Sharples. "Literature Review in Mobile Technologies and Learning", Future Lab Series Report 11, submitted to the University of Birmingham, 2008, pp. 1-48.
[3]. Emmanuel Rotimi Adagunodo, Oludele Awodele and Sunday Idowu. "SMS User Inter-face Result Checking System", Issues in Informing Science and Information Technology, Vol. 6, 2009, pp. 101-112.
[4]. H. Kynäslahti, "In Search of Elements of Mobility in the Context of Education," in Mobile Learning, Helsinki, 2003, pp. 41-48.
[5]. M. Milrad, "How should learning activities using mobile technologies be designed to support innovative educational practices?," Nottingham, UK, 2006.
[6]. J. Stark. (2012, March 15). Principles of Mobile Interface Design [Online]. Available: https://www.slideshare.net/jonathanstark/principles-of-mobile-interface-design
[7]. Reza B'Far, Mobile Computing Principles: Designing and Developing Applications with UML and XML, Press Syndicate of the University of Cambridge, Cambridge, United Kingdom, 2005.
[8]. T. Neil, Mobile Design Pattern Gallery: UI Patterns for Mobile Applications, O'Reilly Media, 2012.
[9]. Ian Sommerville, Software Engineering, 8th ed. Essex, England: Pearson Education Limited, 2007.
[10]. A. B. Cremers, Sascha Alda, Non-functional Requirements [Online]. Available: http://www.iai.uni-bonn.de/slides/10_Interactive
[11]. Ian F. Alexander, Neil Maiden, Scenarios, Stories, Use Cases: Through the Systems Develop-ment Life Cycle, John Wiley & Sons Ltd, The Atrium, Southern Gate,Chichester, West Sussex, England, 2004.
[12]. Hypertext Transfer Protocol, RFC 2616. [Online]. Available: http://tools.ietf.org/html/rfc2616/
[13]. PHP. (2020). [Online]. Available: http://www.php.net/
[14]. MySQL. (2020). [Online]. Available: http://www.mysql.com/
[15]. JSON (JavaScript Object Notation). (2020). [Online]. Available: http://www.json.com/
[16]. Android Volley API. (2020). [Online]. Available: https://developer.android.com/training/volley/index.html
[17]. Adigun Abimbola Adebisi, Akande Noah Oluwatobi, Ajagbe Oluwafemi Adeola (2015) Design and Implementation of a Mobile Students' Course Registration Platform, pp. 25-29
[18]. Konstantinos Semertzidis (2015) Mobile application development to enhance higher education lectures, pp. 10-78
[19]. T. Georgiev, E. Georgieva, and A. Smrikarov, "M-Learning – a new Stage of E-Learning," in International Conference on Computer Systems and Technologies, 2004.
[20]. Android. (2020). [Online]. Available: http://www.android.com/