ACADEMIC RECORD MANAGEMENT AND ASSISTANT SYSTEM

Project submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

BY

CH BHARADWAJ REDDY	(18C91A0514)
D KASIMBEE	(18C91A0519)
B VINOD KUMAR	(18C91A0506)

Under the Esteemed guidance of

Mrs. EENAJA M.Tech

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

HOLY MARY INSTITUTE OF TECHNOLOGY & SCIENCE

(COLLEGE OF ENGINEERING)

(Approved by AICTE New Delhi, Permanently Affiliated to JNTU Hyderabad, Accredited by NAAC with 'A' Grade)

Bogaram (V), Keesara (M), Medchal District -501 301. – 11

2021 - 2022

HOLY MARY INSTITUTE OF TECHNOLOGY & SCIENCE

(COLLEGE OF ENGINEERING)

(Approved by AICTE New Delhi, Permanently Affiliated to JNTU Hyderabad, Accredited by NAAC with 'A' Grade)

Bogaram (V), Keesara (M), Medchal Dist-501301.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the mini project entitled "ACADEMIC RECORD MANAGEMNET AND ASSISTANT SYSTEM" is being submitted by CH BHARADWAJ REDDY (18C91A0514), D KASIMBEE (18C91A0519), B VINOD KUMAR (18C91A0506), in Partial fulfillment of the academic requirements for the award of the degree of Bachelor of Technology in "COMPUTER SCIENCE AND ENGINEERING" HOLY MARY INSTITUTE OF TECHNOLOGY & SCIENCE, JNTU Hyderabad during the year 2021- 2022.

INTERNAL GUIDE	**HEAD OF THE DEPARTMENT**
Mrs.EENAJA (M.Tech)	DR .B.NARSIMHA M.Tech, Ph.D.
Assistant Professor	Professor & HoD
Dept. of Computer Science & Engineering.	Dept. of Computer Science & Engineering

EXTERNAL EXAMINER

<u>ACKNOWLEDGEMENT</u>

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, who's constant guidance and encouragement crowns all effort with success.

I take this opportunity to express my profound gratitude and deep regards to My Guide **Mrs.EENAJA , Assistant Professor**, Dept. of Computer Science & Engineering, Holy Mary Institute of Technology & Science for his / her exemplary guidance, monitoring and constant encouragement throughout the project work.

My special thanks to **Dr. B. Narsimha, Head of the Department**, Dept. of Computer Science & Engineering, Holy Mary Institute of Technology & Science who has given an immense support throughout the course of the project.

I also thank to **Dr. P. Bhaskara Reddy,** the **honorable Director** of my college Holy Mary Institute of Technology & Science for providing me the opportunity to carry out this work.

At the outset, I express my deep sense of gratitude to the beloved **Chairman A. Siddartha Reddy** of **Holy Mary Institute of Technology & Science**, for giving me the opportunity to complete my course of work

I am obliged to **staff members** of Holy Mary Institute of Technology & Science for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Last but not the least I thank **ALMIGHTY** and My **Parents**, and **Friends** for their constant encouragement without which this assignment would not be possible.

CH BHARADWAJ REDDY	**(18C91A0514)**
D KASIMBEE	**(18C91A0519)**
B VINOD KUMAR	**(18C91A0506)**

DECLARATION

This is to certify that the work reported in the present project titled **"ACADEMIC RECORD MANAGEMENT AND ASSISTANT SYSTEM"** is a record of work done by me in the Department of Computer Science & Engineering, Holy Mary Institute of Technology and Science.No part of the thesis is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text the reported are based on the project work done entirely by me not copied from any other source.

CH BHARADWAJ REDDY	(18C91A0514)
D KASIMBEE	(18C91A0519)
B VINOD KUMAR	(18C91A0506)

CONTENTS

LIST OF FIGURES

ABSTRACT

Academic Record Management and Assistant System targets specifically to meet the requirements of the organization/institution engaged in importing training. The idea is to set up an effective training candidate base management, which incorporates various day to day activities e.g., Advertisement of the courses to get new candidates for training, complete management of student data, batch details, faculty information, course details and examination details. Besides all these comments and suggestions of candidates taking training there will be maintained. The system will be under the command of In charge of the company.

1. INTRODUCTION

MOTIVATION

The objective is to develop an application which provides the details of a course related to institute. The purpose is to design an application which automates the processes involved and allows users to perform various operations.

PROBLEM DEFINITION

This project is mainly developed to provide easy approach to the individuals who opt to learn course from an organization/institution. This allows users to view their registered account and their performance at the tests conducted by the institution and to motivate themselves for better performance in future.

SCOPE OF PROJECT

One of the fastest growing industries now a day is mobile industry. There are many competitors in this area who are doing research and development on new platforms & user experience. One such technology is Android from Google which is supported for Google phones. These phones are described as next Generation mobiles [As described by Google]. This project is mainly developed for improving.

USER CLASSES AND CHARACTERISTICS

In this, users get registered. It helps to avail the service provided by the organization whenever there occurs a need for a particular requirement. The services can be availed only after register and login of user.

2. LITERATURE SURVEY

INTRODUCTION

In literature survey we look into the details about the existing system and we try to reduce the disadvantages of the existing system. We try to improve the performance and the efficiency of the new system.

EXISTING SYSTEM

The existing application has details of a User and Faculty through users, but it cannot retrieve details of the status accomplished successfully within limited amount of time. The majority of currently running application are web based, which requires heavy infrastructure and it is hard to work as per convenience. Currently working application require heavy maintenance cost and we can overcome with this issue.

PROPOSED SYSTEM

In developed system users can view the profile of user, faculty and they can also view the dates of the exams published. Once the need for any courses arises then the user can register though the application. This is the way people can be benefited and especially when they are interested to learn a course. This application focused more on user experience which is surveyed with many people. User does not need to go for login credential to look at the offered courses by the admin. Also overcome the issues authentication with faster and effectiveness.

3. ANALYSIS

INTRODUCTION

In this phase the requirements are gathered and analyzed. Users requirements are gathered in this phase. This phase is the focus of the administrators and registered accounts. Meetings with users and registered people are held to determine the requirements like: Who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirements gathering phase. After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied.

Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

SOFTWARE REQUIREMENT SPECIFICIFICATION

The software requirements specification specifies the functional requirements and non-functional requirements. Functional requirements refer to how the system is going to react according to the input provided and how it is going to behave in particular situations and nonfunctional requirements refers to Usability, Reliability, Availability, Performance, Security, Supportability, Interface.

USER INTERFACES:

This application includes GUI standards or product family style guides that are to be followed screen layout constraints, buttons and functions that will appear on every screen, error message display standards, and so on.

EXTERNAL INTERFACE REQUIREMENTS:

SOFTWARE REQUIREMENTS

Android SDK

Eclipse Ganymede IDE

Operating System can be either of these- Windows XP, Windows 7, Windows 8 with different versions, Windows 10, etc.

HARDWARE REQUIREMENTS

PROCESSOR	: (min) P4 processor
RAM	: 1GB
Hard Disk	: 20 GB
Keyboard	: Lenovo Keyboard
Mouse	: HP X1000

JAVA

Java is an object-oriented language and is very similar to C++.Java is simplified to eliminate language features that cause common programming errors. Java source code files are compiled into a format called bytecode, which can then be executed by a Java interpreter.

- **PLATFORM INDEPENDENT**

  The programs written on one platform can run on any platform provided the platform must have the JVM.

- **PORTABLE**

  The feature Write-once-run-anywhere makes the java language portable provided that the system must have interpreter for the JVM.

- **SIMPLE**

Programs are easy to write and debug because java does not use the pointers explicitly. It also has the automatic memory allocation and deallocation system.

- **MULTITHREADED**

Multithreading means a single program having different threads executing independently at the same time.

- **ROBUST**

Java has the strong memory allocation and automatic garbage collection mechanism. It provides the powerful exception handling and type checking mechanism as compare to other programming languages.

- **OBJECT ORIENTED**

To be an Object-Oriented language, any language must follow at least the four characteristics.

  - Inheritance
  - Encapsulation
  - Polymorphism
  - Dynamic binding

- **DISTRIBUTED**

The widely used protocols like HTTP and FTP are developed in java. Internet programmers can call functions on these protocols and can get access to the files from any remote machine on the internet rather than writing codes on their local system.

- **SECURE**

All the programs in java run under an area known as the sand box. Security manager determines the accessibility options of a class like reading and writing a file to the local disk.

- **HIGH PERFORMANCE**

In the beginning interpretation of bytecode resulted in slow performance but the advance version of JVM uses the adaptive and just in time compilation technique that improves the performance.

- **INTEGRATED**

Java is an interpreted language as well. Programs run directly from the source code.

PROJECT PERSPECTIVE

Android is a software stack for mobile devices that includes an operating system, middleware, and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

The Android SDK includes a comprehensive set of development tools. Requirements include Java Development Kit, the officially supported integrated development environment (IDE) is Eclipse (3.2 or later) using the Android Development Tools (ADT) Plug in, though developers may use any text editor to edit Java and XML files then use command line tools to create, build and debug Android applications. It would be more cost effective if we can use normal phone numbers for receiving data from customers via SMS. However, it is a very tedious process if we don’t have backend automated system to analyze the received data. We are proposing the below approach to make this process cost effective as well as efficient.

ANDROID

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

The Android SDK includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator (based on QEMU), documentation, sample code, and tutorials. Currently supported development platforms include x86-architecture computers running Linux (any modern desktop Linux distribution), Mac OS X 10.4.8 or later, Windows XP or Vista. The officially supported integrated development environment (IDE) is Eclipse (3.2 or later) using the Android Development Tools (ADT) Plugin, though developers may use any text editor to edit Java and XML files then use command line tools to create, build and debug Android applications.

ABOUT NATIVE CODE:

Libraries written in C and other languages can be compiled to ARM native code and installed, but the Native Development Kit is not yet officially supported by Google. Native classes can be called from Java code running under the Dalvik VM using the System. Load Library call, which is part of the standard Android Java classes.

USER DOCUMENTATION

In this user manual we are going to keep the information regarding our product, which can be understandable by a new person who is going to use it. If a new person is using it, online help will be provided in that. We are going to explain each and every step clearly about our product so that any user can easily understand it.

CREATING AN ANDROID PROJECT

The ADT plug-in provides a New Project Wizard that you can use to quickly create a new Android project (or a project from existing code). To create a new project:

- Select File > New > Project.

- Select Android > Android Project, and click Next.
- Select the contents for the project:
- Enter a Project Name. This will be the name of the folder where your project is created.
- Under Contents, select Create new project in workspace. Select your project workspace location.
- Under Target, select an Android target to be used as the project's Build Target. The Build Target specifies which Android platform you'd like your application built against.
- Unless you know that you'll be using new APIs introduced in the latest SDK, you should select a target with the lowest platform version possible, such as Android 1.1.
- Under Properties, fill in all necessary fields:

Enter an Application name. This is the human-readable title for your application — the name that will appear on the Android device.

- Enter a Package name. This is the package namespace (following the same rules as for packages in the Java programming language) where all your source code will reside.
- Select Create Activity (optional, of course, but common) and enter a name for your main Activity class.
- Enter a minimum SDK Version. This is an integer that indicates the minimum API Level required to properly run your application. Entering this here automatically sets the minimum SDK Version attribute in the <uses-sdk> of your Android Manifest file. If you're unsure of the appropriate API Level to use, copy the API Level listed for the Build Target you selected in the Target tab.
- Click Finish.

TO CREATE AN AVD WITH THE AVD MANAGER:

- Select Window > Android SDK and AVD Manager, or click the Android SDK and AVD Manager icon (a black device) in the Eclipse toolbar.
- In the Virtual Devices panel, you'll see a list of existing AVDs. Click New to create a new AVD.
- Fill in the details for the AVD.
- Give it a name, a platform target, an SD card image (optional), and a skin (HVGA is default).
- Click Create AVD.

When you first run a project as an Android Application, ADT will automatically create a run configuration. The default run configuration will launch the default project Activity and use automatic target mode for device selection (with no preferred AVD).

TO CREATE OR MODIFY A LAUNCH CONFIGURATION

Follow these steps as appropriate for your Eclipse version:

- Open the run configuration manager.
- In Eclipse 3.3 ,select Run > Open Run Dialog (or Open Debug Dialog)
- In Eclipse 3.4 (Ganymede), select Run > Run Configurations (or Debug Configurations)
- Expand the Android Application item and create a new configuration or open an existing one.

SQLITE

SQLite is an ACID-compliant embedded relational database management system contained in a relatively small C programming library. The source code for SQLite is in the public domain.

DESIGN

Unlike client-server database management systems, the SQLite engine is not a standalone process with which the application program communicates. Instead, the SQLite library is linked in and thus becomes an integral part of the application program. The library can also be called dynamically. The application program uses SQLite's functionality through simple SMS, which reduces latency in database access as function SMS within a single process are more efficient than inter-process communication.

FEATURES

SQLite implements most of the SQL-92 standard for SQL but it lacks some features. A stand-alone program called sqlite3 is provided which can be used to create a database, define tables within it, insert and change rows, run queries and manage a SQLite database file. SQLite is a popular choice for local/client SQL storage within a web browser and within a rich internet application framework. This may be because SQLite's dynamically typed storage matches the web browser's core languages of JavaScript and XML.

4. DESIGN

INTRODUCTION

Software design is the process by which an agent creates a specification of a software artifact, intended to accomplish goals, using a set of primitive components and subject to constraints. Software design may refer to either "all the activity involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems" or "the activity following requirements specification and before programming, as in a stylized software engineering process." Software design usually involves problem solving and planning a software solution. This includes both a low-level component design and a high-level, architecture design.

ARCHITECTURE DIAGRAM

Architecture diagram is a diagram of a system, in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation.

A user for who the application looks like an user interface actually consists of a database called as SQLite that comes along with Android SDK and need no other installation. This is the database that is used to store and retrieve information. This is an application that is developed in java and hence all its features apply here as well such as platform independence, data hiding,

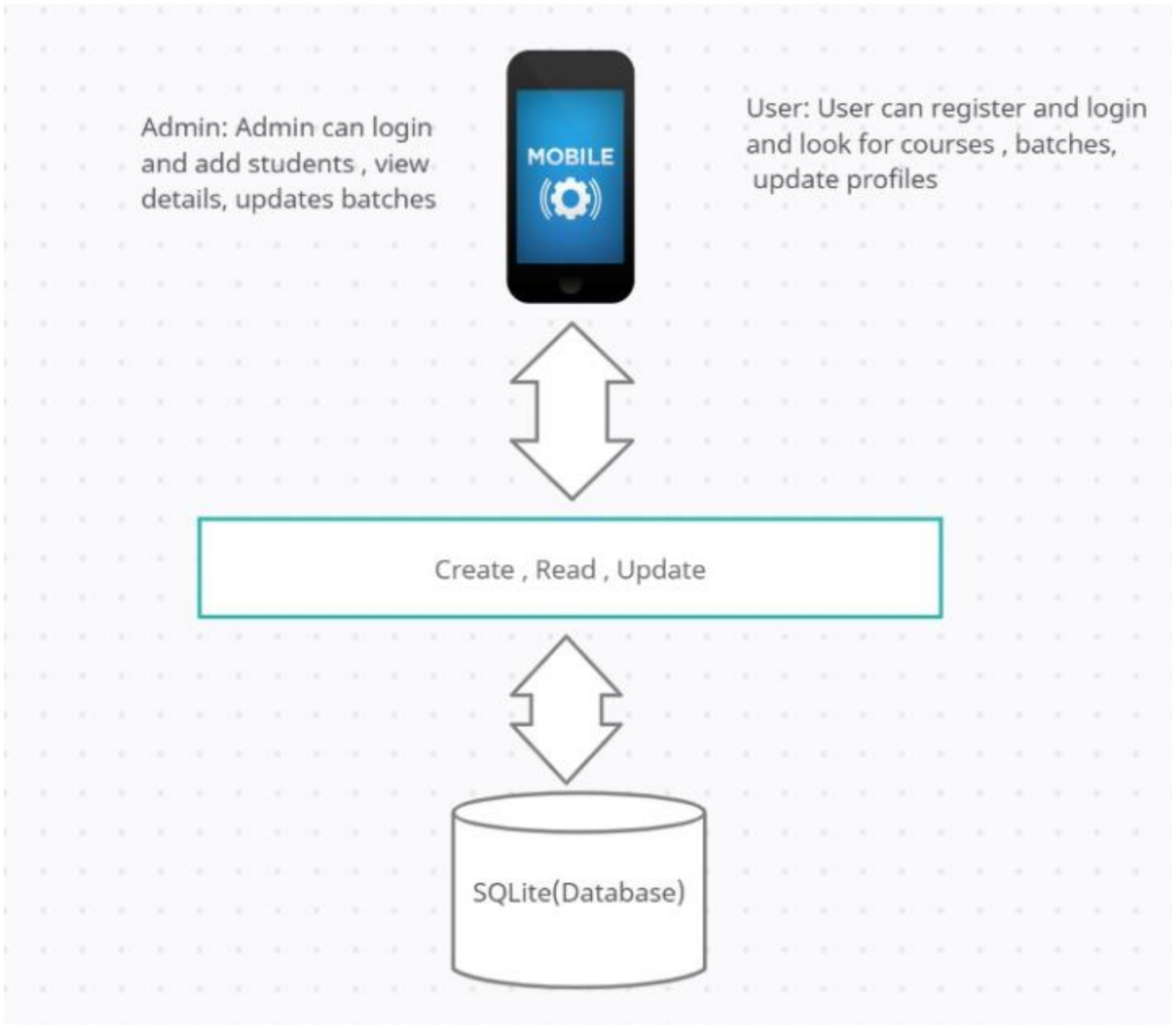

Admin: Admin can login and add students , view details, updates batches
MOBILE
User: User can register and login and look for courses , batches, update profiles
Create , Read , Update
SQLite(Database)

Fig 4.2 Architecture

MODULES

- Module I: Student Registration
    i. In this module, student need to registered using the following input (Name, Email, Password, Mobile, Branch)
- Module II: Authentication Module
    i. Students, Admin will login from this module and can perform the operations
- Module III: Courses offered (Advertisement)
    i. List of courses added by admin will be displayed here to help students to view and understand the courses
- Module IV: Batch Details
    i. Students where are added batch by admin can be view here.

UNIFIED MODELING LANGUAGE (UML)

The unified modeling is a standard language for specifying, visualizing, constructing and documenting the system and its components is a graphical language which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure and control information about the systems.

Depending on the development culture, some of these artifacts are treated more or less formally than others. Such artifacts are not only the deliverables of a project; they are also critical in controlling, measuring, and communicating about a system during its development and after its deployment.

The UML addresses the documentation of a system's architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UML provides a language for modeling the activities of project planning and release management.

BUILINDING BLOCKS OF UML

The vocabulary of the UML encompasses three kinds of building blocks:

- Things
- Relationships
- Diagrams

Things are the abstractions that are first-class citizens in a model, relationships tie these things together and diagrams group interesting collections of things.

THINGS IN THE UML

There are four kinds of things in the UML:

Structural things

1. Behavioral things
2. Grouping things
3. Anotational things

- **STUCTURALTHINGS:** are the nouns of UML models. The structural things used in the project design are:

First, a **class** is a description of a set of objects that share the same attributes, operations, relationships and semantics.

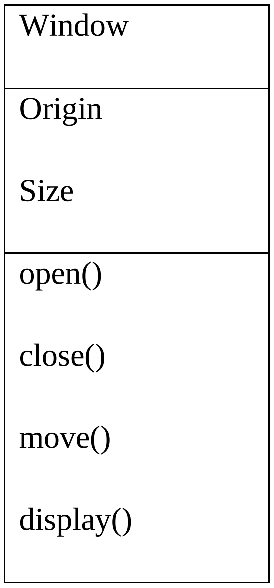


Fig: Classes

Second, a **use case** is a description of set of sequence of actions that a system performs that yields an observable result of value to particular actor.

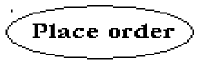


Fig: Use Cases

Third, a node is a physical element that exists at runtime and represents a computational resource, generally having at least some memory and often processing capability.

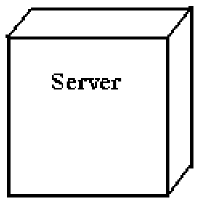


Fig: Nodes

2. BEHAVIORAL THINGS: are the dynamic parts of UML models. The behavioral thing used is:

INTERACTION:

An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves a number of other elements, including messages, action sequences (the behavior invoked by a message, and links (the connection between objects).

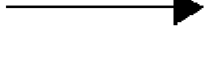


Fig: Messages

RELATIONAL IN THE UML

There are four kinds of relationships in the UML:

- Dependency
- Association
- Generalization
- Realization

A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).

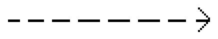


Fig: Dependencies

An **association** is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.

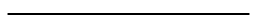


Fig: Association

A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element(the parent).

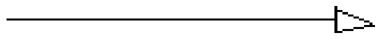
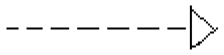


Fig: Generalization

A **realization** is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.



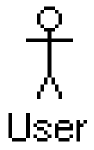
SEQUENCE DIAGRAMS

UML sequence diagrams are used to represent the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interactions of the header elements, which are displayed horizontally at the top of the diagram.

Sequence Diagrams are used primarily to design, document and validate the architecture, interfaces and logic of the system by describing the sequence of actions that need to be performed to complete a task or scenario. UML sequence diagrams are useful design tools because they provide a dynamic view of the system behavior which can be difficult to extract from static diagrams or specifications.

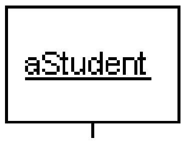
ACTOR

Represents an external person or entity that interacts with the system



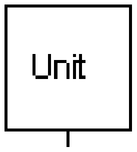
OBJECT

Represents an object in the system or one of its components



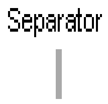
UNIT

Represents a subsystem, component, unit, or other logical entity in the system (may or may not be implemented by objects)



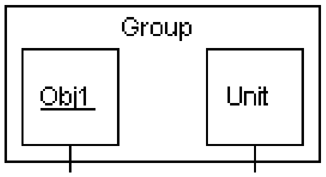
SEPERATOR

Represents an interface or boundary between subsystems, components or units (e.g., air interface, Internet, network)



GROUP

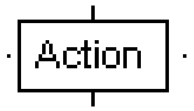
Groups related header elements into subsystems or components



SEQUENCE DIAGRAM BODY ELEMENTS

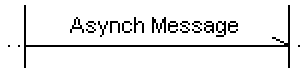
ACTION

Represents an action taken by an actor, object or unit



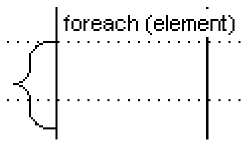
ASYNCHRONUS MESSAGE

An asynchronous message between header elements



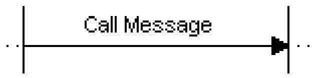
BLOCK

A block representing a loop or conditional for a particular header element



CALL MESSAGE

A call (procedure) message between header elements



CREATE MESSAGE

A "create" message that creates a header element (represented by lifeline going from dashed to solid pattern)

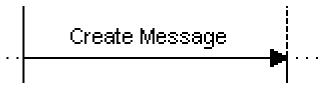
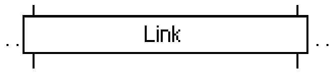
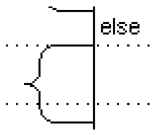


DIAGRAM LINK

Represents a portion of a diagram treated as a functional block. Similar to a procedure or function call that abstracts functionality or details not shown at this level and can be an optional link to another diagram for elaboration.

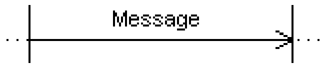


Else Block Represents an "else" block portion of a diagram block



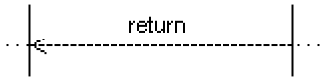
MESSAGE

A simple message between header elements



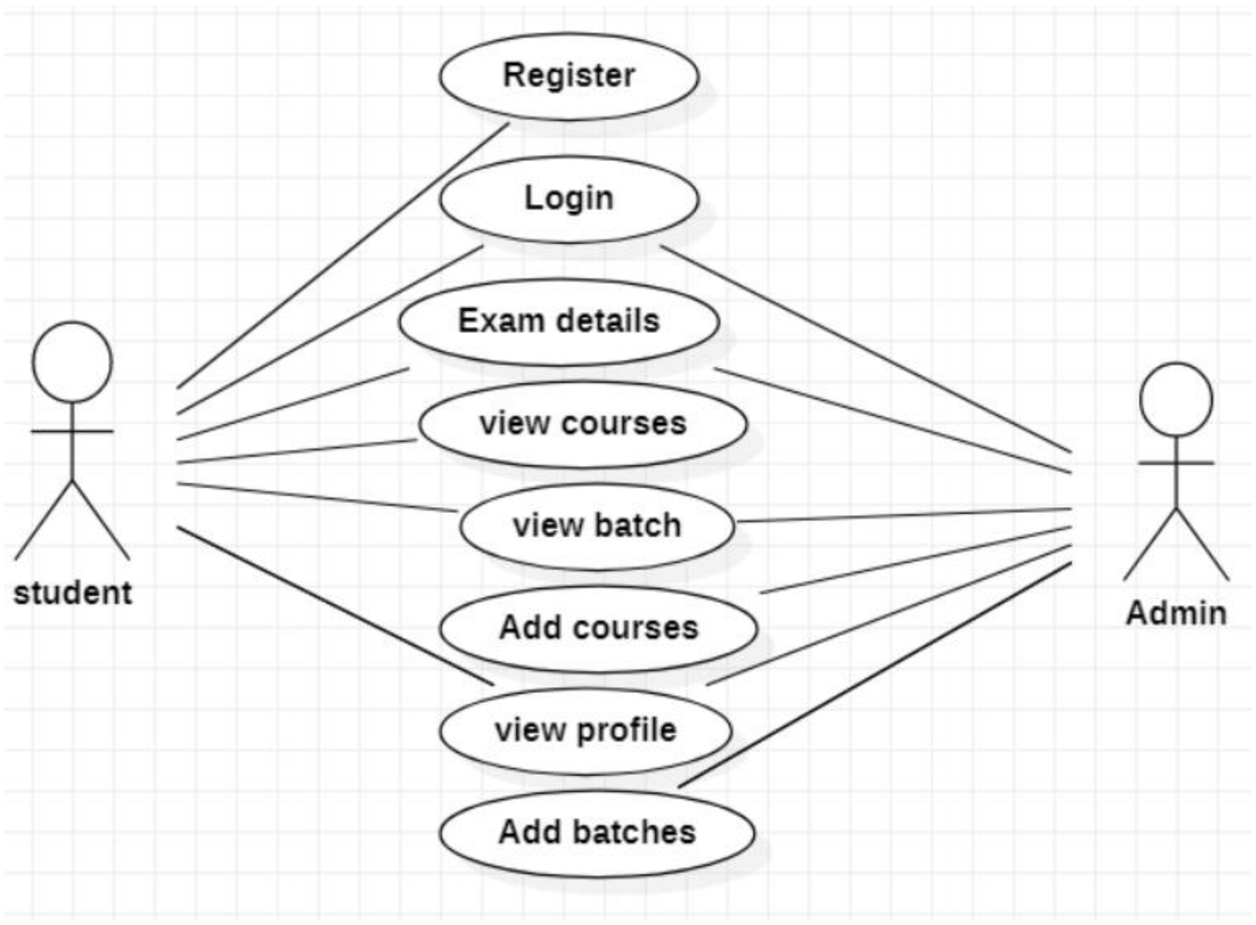
RETURN MESSAGE

A return message between header elements



UML DIAGRAMS

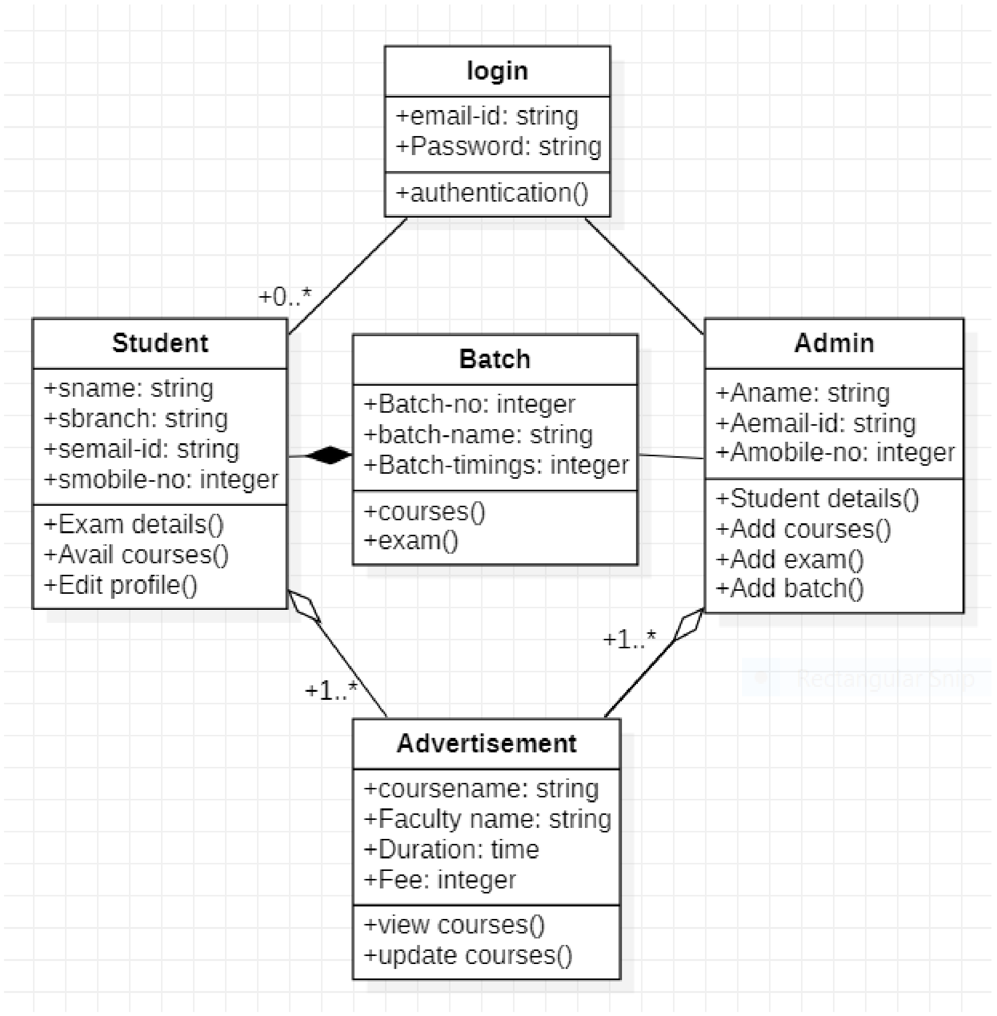
USE CASE DIAGRAM



Description:

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases(Add courses, View Courses, Add exam, View exam details), actors(Students, Admin), and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system

CLASS DIAGRAM

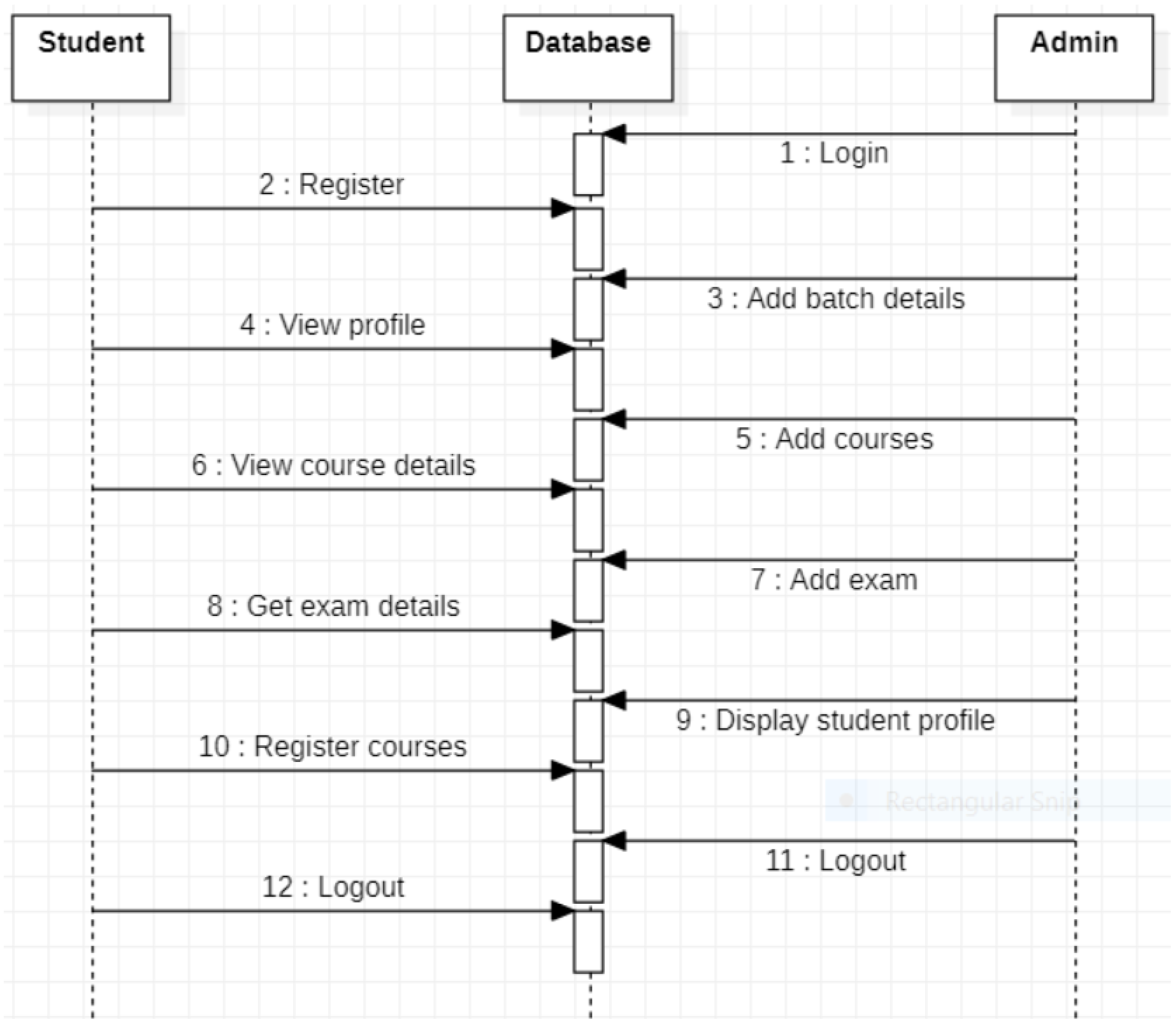


Description:

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of objectoriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

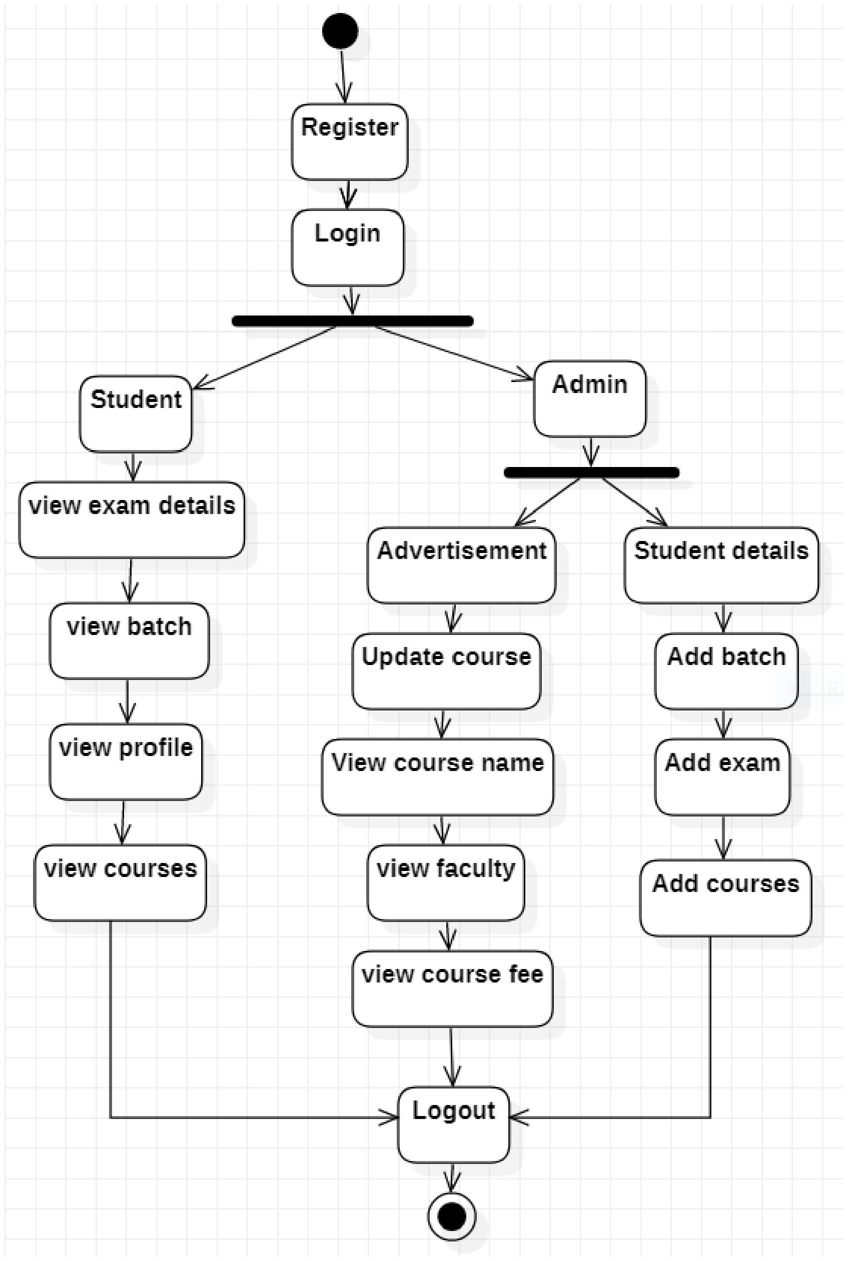
SEQUENCE DIAGRAM



Description:

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

ACTIVITY DIAGRAM



Description:

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. Firstly, student get registered from the registration module and after providing details in authentication, it get authenticated and student/admin can view courses, exam , profile

5. IMPLEMENTATION

INTRODUCTION

Implementation is the most crucial stage in achieving a successful system and giving the user's confidence that the new system is workable and effective. Implementation of the modified application to replace an existing one. This type of conversation is relatively easy to handle, provide there are no major changes in the system.

Each program is tested individually at the time of development using the data and has verified that this program linked together in the way specified in the programs specification. The computer system and its environment are tested to the satisfaction of the user. The system that has been developed is accepted and proved to be satisfactory for the user. And so the system is going to be implemented very soon. A simple operating procedure is included so that the user can understand the different functions clearly and quickly.

Initially as a first step the executable form of the application is to be created and loaded in the common server machine which is accessible to all the user and the server is to be connected to a network. The final stage is to document the entire system which provides components and the operating procedures of the system.

DESIGN AND IMPLEMENTATION CONSTRAINTS

All modules are coded thoroughly based on requirements from software organization. The software is designed in such a way that the user can easily interact with the screen. Software is designed in such a way that it can be extended to the real time business.

Pseudo Code

```
package com.academicplanner;
import android.os.AsyncTask;
import android.os.Bundle;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class LoginActivity extends Activity {
EditText EmailId, password;
Button a;
String u;
 String p;
 SQLiteDatabase db;



@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

        final GlobalClass globalvariabel = (GlobalClass)getApplicationContext();

        setContentView(R.layout.activity_login);

        EmailId = (EditText) findViewById(R.id.editText1);

        password = (EditText) findViewById(R.id.editText2);

        a=(Button) findViewById(R.id.button1);

        a.setOnClickListener(new OnClickListener() {
```

```
                    @Override
                    public void onClick(View v) {
                              // TODO Auto-generated method stub

     if(EmailId.getText().toString().equals("")||password.getText().toString().equals("")){


Toast.makeText(LoginActivity.this, "PLz enter the fields..!", Toast.LENGTH_LONG).show();
                              }else{
                                        u = EmailId.getText().toString();
                                        p = password.getText().toString();
                                        try{

db=openOrCreateDatabase("APLANNERDB",SQLiteDatabase.CREATE_IF_NECESSARY,
null);
                                             }
catch(Exception exception){
                                         exception.printStackTrace();
                                     }  try{


                                              Cursor cc = db.rawQuery("select * from studenttb where
EmailId = '"+u+"' and password = '"+p+"' ", null);
 if(EmailId.getText().toString().equals("admin")&&
password.getText().toString().equals("admin")){
Toast.makeText(LoginActivity.this, "Welcome To Admin Home Page " + u ,
Toast.LENGTH_LONG).show();
```

```
      globalvariabel.SetUserName(EmailId.getText().toString().trim());

      globalvariabel.SetPassword(password.getText().toString().trim());
      Intent i = new Intent(LoginActivity.this,AdminHome.class);
      startActivity(i);



       // User Login
            else if(cc.moveToFirst()){


      String temp="";
      if (cc != null) {
                              if(cc.getCount() > 0)
                                                          {
                              //return true;
                               scan g=new scan();
                              g.execute();


Toast.makeText(LoginActivity.this, "Welcome To student Home Page " + u , Toast.LENGTH_LONG).show();


      globalvariabel.SetUserName(EmailId.getText().toString().trim());


      globalvariabel.SetPassword(password.getText().toString().trim());
Intent i = new Intent(LoginActivity.this,StudentHome.class);
startActivity(i);
```

```
                }else{
      Toast.makeText(LoginActivity.this, "Login Fails..!", Toast.LENGTH_LONG).show();
                        }
                          }
                                      }


                                      }catch(Exception exception){
                                          exception.printStackTrace();
                                      }
                                          }
                        }
      });





}
public class scan extends AsyncTask<String, String, String>{


      private ProgressDialog pd;


      protected void onPreExecute() {
                  super.onPreExecute();
       pd = new ProgressDialog(LoginActivity.this);
       pd.setTitle("Please Wait");
```

```
            pd.setMessage("Logging....");
            pd.setMax(200);
            pd.show();
           }

           @Override
           protected String doInBackground(String... params) {
                   // TODO Auto-generated method stub
                   return null;
           }

       }

}
```

SCREEN SHOTS

HOME SCREEN

This is the home screen of our project which contains students registration form , authentication and advertisement (where students can see the courses offered)Also student can view the batches which they are added in by admin.

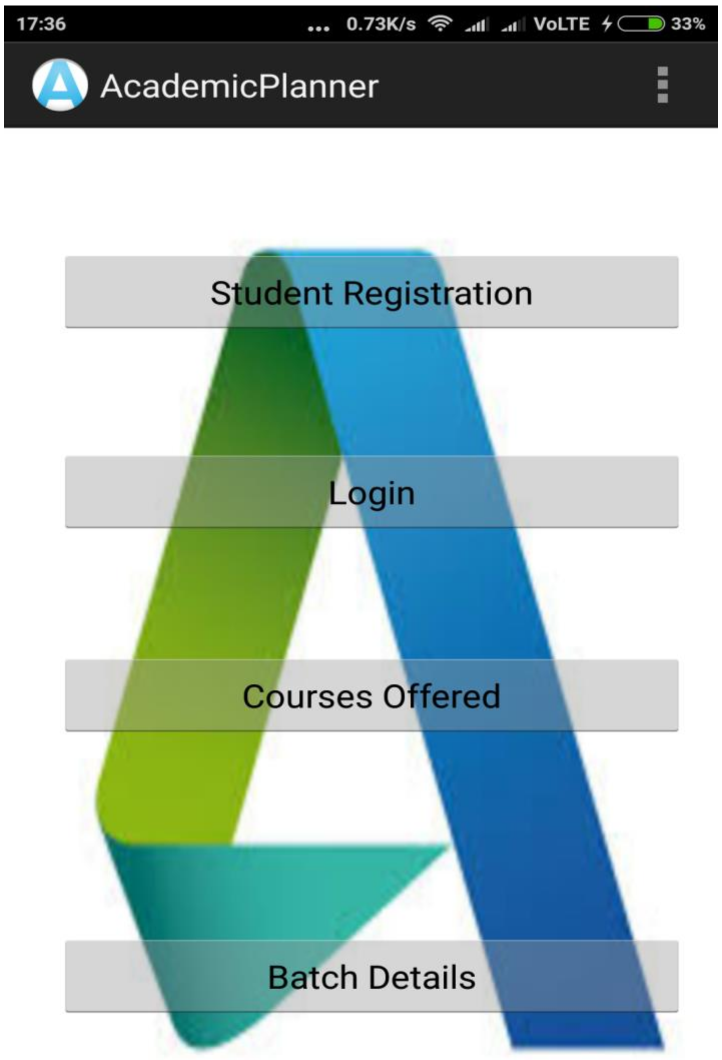
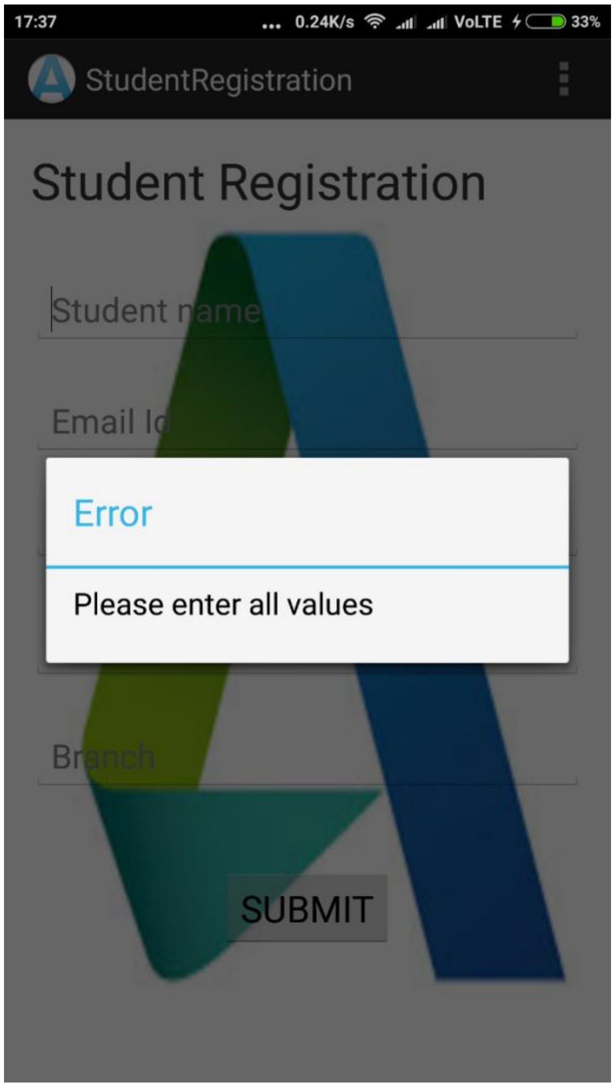


Fig:5.1

STUDENT REGISTRATION

In this screen user i.e. Student has to register with proper details which is require for academic purpose. If any text field is incomplete it will show the error as (fig: 5.4.2)



Error

Please enter all values

Fig:5.2

Authentication screen

Student and admin has to login from this screen. As student registered by filling details in student registration form. After providing authenticated details student can see the modification done by admins

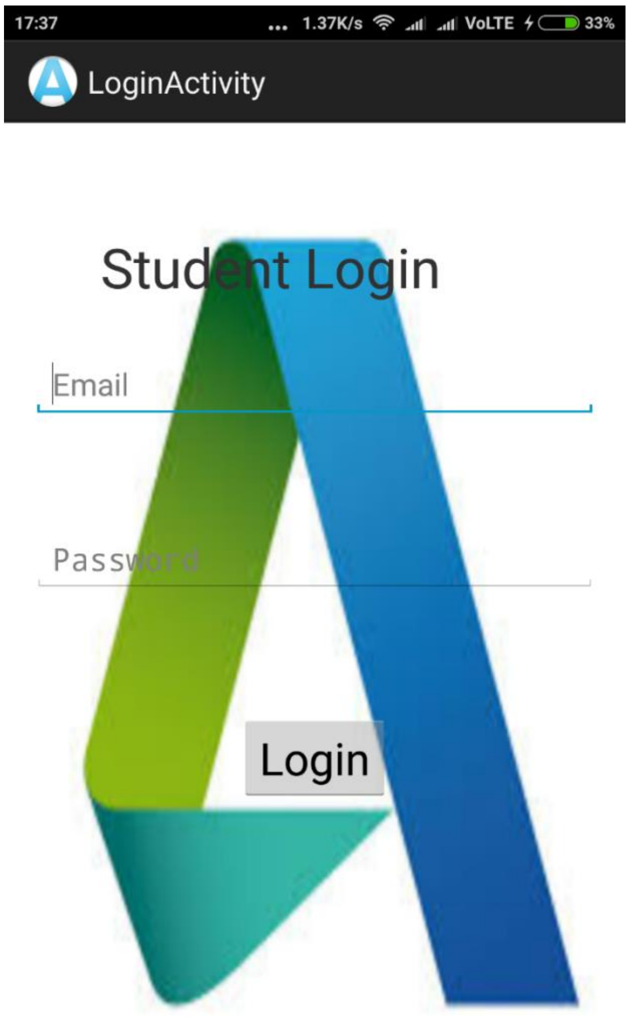
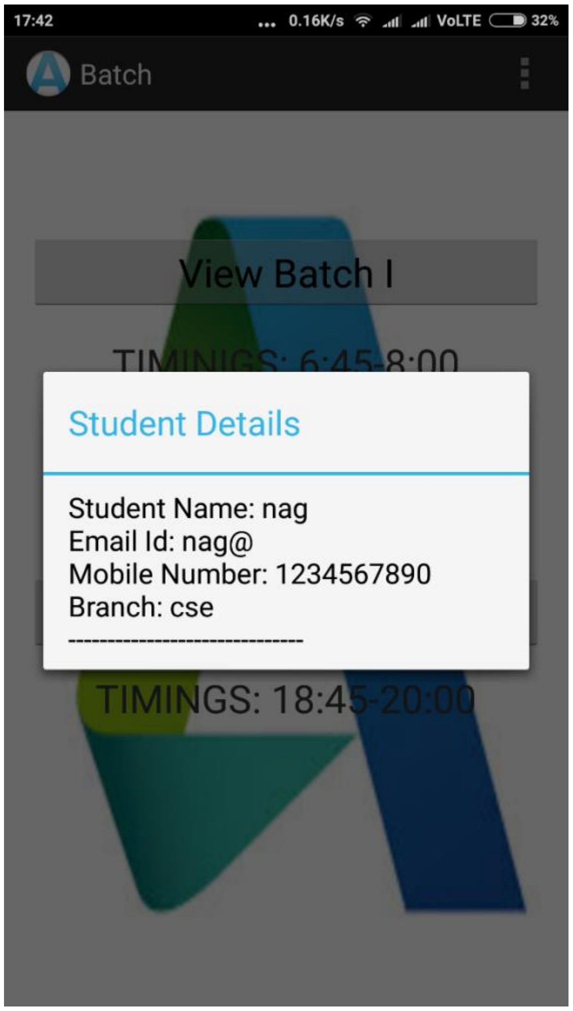


Fig:5.3

Batches Screen

In this screen admin added students to list of students to batches after login from Authentication screen. User and admin both can access the batches added by admin. Students details requires are Student names, email ID, phone, and branches



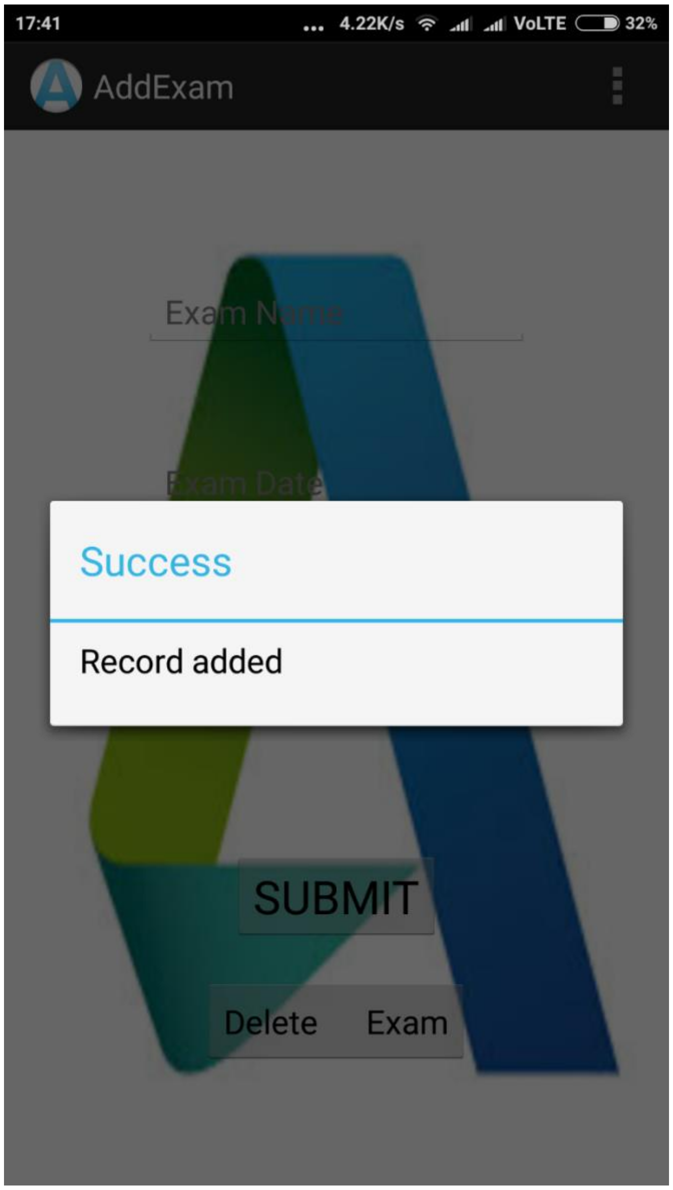
Student Details

Student Name: nag
Email Id: nag@
Mobile Number: 1234567890
Branch: cse

Fig:5.4

EXAM DETAILS SCREEN

In these screen, admin will access after authentication from authentication and display the exam details. Admin can also delete the exam details.



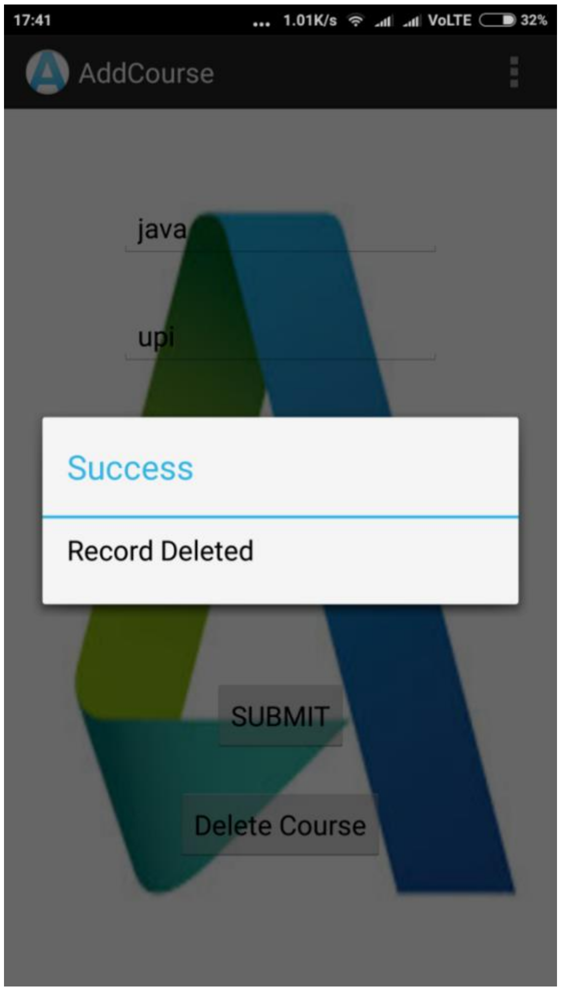
Success

Record added

Fig:5.5

ADD COURSE SCREEN

In these screen, admin will access after authentication from authentication page and display the Course details. Admin can also delete the Courses details.



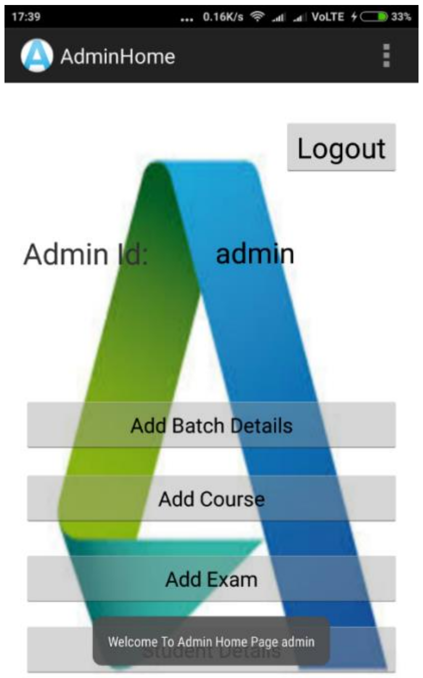
Success

Record Deleted

Fig:5.6

ADMIN SCREEN

In this screen, after authentication from authentication page, admin can add students to batches by providing student details, can also add courses, can add exam details, and also access to student details



Welcome To Admin Home Page admin

Fig:5.7

STUDENT SCREEN

In this screen, Student after registration from registration page, after registration providing authentications details , student can access to course registration, profile, and provide exam details, and a courses.

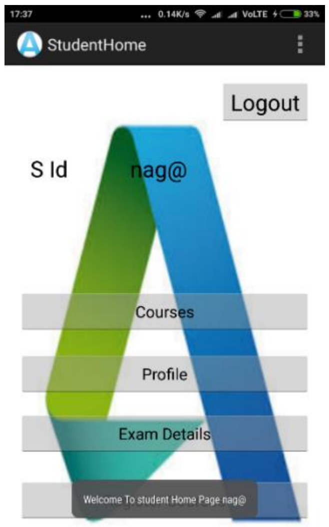


Fig:5.8

COURSE REGISTRATION SCREEN

After viewing courses from the advertisement page, student can register into courses after authentication by providing the require details

COURSE REGISTRATION SCREEN

After viewing courses from the advertisement page, student can register into courses after authentication by providing the require details

COURSE REGISTRATION SCREEN

After viewing courses from the advertisement page, student can register into courses after authentication by providing the require details

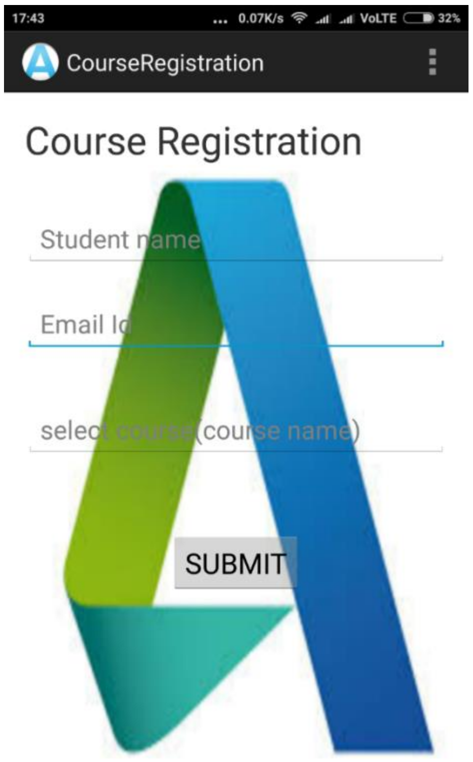


Fig:5.9

6. TESTING

SOFTWARE TESTING

Software testing is a critical element of software quality and assurance and represents ultimate review of specifications, design and coding. Testing is an exposure of the system to trial input to see whether it produces correct output.

TESTING PHASES

Software testing includes the following:

- Test activities are determined and test data selected
- The test is conducted and test results are compared with the expected results.

TESTING ACTIVITIES

1. **INSPECTING COMPONENTS**: This finds faults in the individual component through the manual inspection of its source code.
2. **UNIT TESTING**: This finds faults by isolating an individual component using test stubs and drivers and by exercising the components using a test case.
3. **INTEGRATION TESTING**: This finds faults by integrating several components together. System testing, which focuses on the complete system, its functional and non-functional requirements and its target environment.

UNIT TESTING

Unit testing focuses on the building blocks of the software system, that is, objects and subsystems. They are three motivations behind focusing on components. First, unit testing reduces the complexity of the overall test activities, allowing us to focus on smaller units of the system. Unit testing makes it easier to pinpoint and correct faults given that few computers are involved in this test. Unit testing allows parallelism in the testing activities; that is each component can be tested independently of one another

The specific candidates for unit testing are chosen from the object model and the system decomposition of the system. In principle, all the objects developed during the development process should be tested. Which is often not feasible because of time and budget?

EQUIALENCE TESTING

It is a black box testing technique that minimizes the number of test cases. The possible inputs are partitioned into equivalence classes, and a test case is selected for each class. The assumption of equivalence testing is that the system usually behaves in similar ways for all members of a class. To test the behavior associated with an equivalence class, we only need to test one member of the class. Equivalence testing consists of two steps: identification of the equivalence classes and selection of the test inputs.

The following criteria are used for the equivalence testing:

- **COVERAGE**:
  Every possible input belongs to one of the equivalent classes.
- **DISJOINTEDNESS**:
  No input belongs to one of the equivalent classes.
- **REPRESENTATION**:
  If the execution demonstrates an error when a particular member of an equivalence class is used as input, then the same error can be detected by using any other member of the class as input.

BOUNDARY TESTING

Boundary testing is a special case of equivalence testing and focuses on the conditions at the boundary of the equivalence classes. Rather than selecting any element in the equivalence class, boundary testing requires that the element be selected from the “edges” of the equivalence class.

A disadvantage of equivalence class and boundary testing is that these techniques do not explore combination of test input data.

PATH TESTING

Path testing is a white box testing technique that identifies faults in the implementation of the component. The assumption behind path is that, by exercising all possible paths through the code at least once, most faults will trigger failures. The identification of paths requires knowledge of the source code and data structures.

The path testing technique was developed for imperative languages. Object oriented language introduce several difficulties when using path testing.

POLYMORPHISM:

Polymorphism enables messages to be bound to different methods bases on the class of the target. Although this enables developers to reuse code across a large number of classes, it is also introduce more cases to test.

- **SHORTER METHODS:**

Methods in object-oriented language have the tendency to be shorter than procedures and functions in imperative languages. This decreases the likelihood of control flow faults, which can be uncovered using the path testing technique.

STATE BASED TESTING

Object oriented languages introduce the opportunity for new types of faults in object-oriented systems.

State based testing is a recent testing technique, which focuses on object-oriented systems. Most testing technique which focuses on selecting a number of test inputs for a given state of the system, exercising a component or a system, and comparing the observed outputs with java. State based testing focuses on comparing the resulting state of the system with the expected state. In the context of a class, state-based testing consists of deriving test cases from the UML state chart diagram for the class.

INTEGRATION TESTING

It detects faults that have not been detected during unit testing, by focusing on small group of components.

TEST CASE DESIGN

The design of tests for software and other engineering products can be as challenging as the initial design of the product. Test case methods provide the developer with a systematic approach to testing. Moreover, these methods provide a mechanism that can help to ensure the completeness of tests and provide the highest like hood for uncovering errors in software.

Any Engineered product can be tested in either of the two ways:

- Knowing the specified function that a product has been designed to perform, tests can be conducted. These tests demonstrate whether each function is fully operational and at the same time searches for errors in each function.
- Knowing the internal workings of a product, tests can be conducted to ensure that internal operations are performed according to specifications and all internal components hence been adequately exercised.

Test case design methods are divided into two types:

  - White-box testing
  - Black-box testing

WHITE-BOX TESTING

White –box testing, sometimes called glass-box testing is a test, case designed method that uses the control structure of the procedural design to derive test cases. Using white-box testing methods, the s/w engineer can derive test cases that guarantee that all independent paths within a module have been exercised at least once. Exercise all logical decisions on their true and false sides. Execute all loops at their boundaries and within their operational bounds. Exercise internal data structures to ensure their validity.

Basis path testing is a white-box testing technique. The basis path method enables the test case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basis set are guaranteed to exercise every statement in the program at least one-time during testing.

BLACK-BOX TESTING

Black-box testing, also called behavioral testing, focuses on the functional requirements of the s/w. Black-box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements of a program. It is a complementary approach that is likely to uncover a different class of errors that white-box methods could not.

Black-box testing attempted to find errors in the following categories.

- Incorrect or missing functions.
- Interface errors.
- Errors in data structures or external data base access.
- Behavior or performance errors.
- Initialization and termination errors.

Black-box testing purposely disregards control structure; attention is focused on information domain. By applying black-box techniques, we derive a set of cases that satisfies the criteria test cases that reduce, by a count that is greater than one, the number of additional test cases that must be designed to achieve reasonable testing. Test cases that tell us something about the presence or absence of classes of errors, rather than an error associated only with the specified.

CONCLUSION

At the end of this application it is a sophisticated approach for users to have a best selection and gives better performance for Administrator so that he can easily add, update and view institution details and students information. This initiative of academic record management and assistant system made more easy to communicate with an institute. This is convenient to the students to view and register courses easily. There is also a facility for the customer to check the status of registered courses and obtained marks

BIBILOGRAPHY

- Herbert Schildt.2008,” Java Complete Reference”, Tata McGraw-Hill , 7th Edition, pp. 177-180.
- Grady Brooch, James Rambaugh.1998, “Unified Modeling Language User Guide”, Addison Wesley Publishing, chapter 8-31.

WEBSITES:	**REFERRED URLS:**
www.android.com	http://developer.android.com/index.html
www.google.com	http://en.wikipedia.org/wiki/SQLite