

## 1. INTRODUCTION

### 1.1 ABOUT THE PROJECT

We have seen over the years that the process of notice boards, important notification about academics has been carried out manually almost across all educational institutions. The process is not only time consuming but also inefficient. This traditional system requires a manual work of writing notifications, taking printouts, displaying it on notice boards and also requires students to watch periodically. It uses a lot of paper work. Today, we need not to maintain paper based Notice boards. Following this thought, we have developed a system based on the concept of web services which is implemented on Android mobile application.

In recent years the Android Technology with web services has brought many drastic changes in mobile application development field. The creation and management of accurate, up-to-date information regarding a student's academic career is crucial for the colleges. This application provides a solution through a simple interface for maintenance of student information and also helps parents to get detailed information regarding their ward such as attendance, fees due, marks, important notice, event details, etc. It also contains query message option for parents so that parents can interact with the college faculty through this application. It also facilitates parents to gain all the notifications about the activities held in the college. Each individual parent will be provided with the details of his/her ward only.

### 1.2 PURPOSE

This application will make the job easier for parents to get the all information about their ward. This android application provides an interface between parents and faculty. If parents want to know about their ward they can also ask queries through this application.

### 1.3 OVERVIEW

This application contains the following modules. Mainly it contains the 3 modules

- . Admin
- . Faculty
- . Parents

#### **Admin**

Admin as contains all the rights for adding the faculty, removing the faculty, adding parents and removing the parents. And he provides the username and password to all the parents and faculty. The total control will be done by the admin.

#### **Faculty**

To interact with this application faculty needs authentication. Administrator provides the login details for the faculty. After successfully login into the system he has the rights to send all the information to the parents like attendance, notifications, marks etc.

#### **Parents**

Parents need to get the login details from the administrator to view the details concerning their ward. After successful login into the system he can receive the all the information about their ward like attendance, notifications, marks etc.

## 2. SYSTEM ANALSYS

### 2.1 EXISTING SYSTEM

In the existing system information to parents regarding their ward is provided through post cards, SMS etc. Here the mobile numbers will be taken by the students and sometimes they may give the wrong number or their own number. We didn't sure about the numbers and we can't able to send the marks and monthly attendance through the SMS. If we use the post cards these are very time consuming and lengthy process and sometimes students may give the wrong address.

### SMS SYSTEM ARCHITECTURE

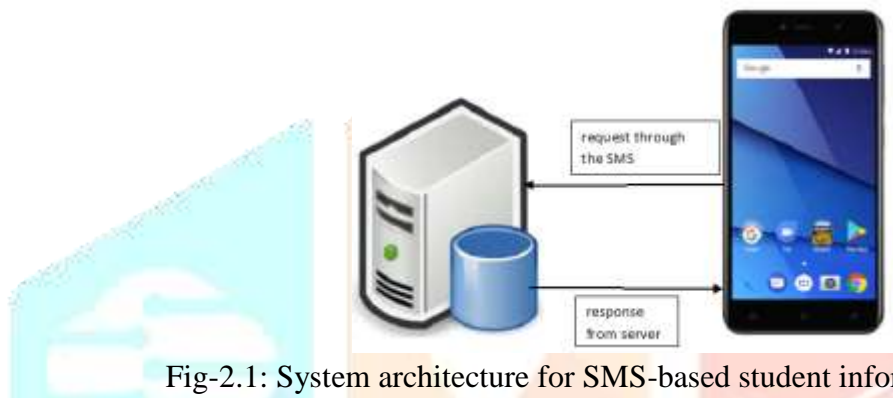


Fig-2.1: System architecture for SMS-based student information

This system is worked based on the following important steps those are, firstly parents can ask about the information about their child through the number. And then server maps that number with the students' name. Server is searches that number is belongs to which students and information about the students. And after finding the data about that particular student which will be sends to their parents.

#### 2.1.1 DISADVANTAGES

- sending information through post cards are very lengthy process
- students may give wrong numbers
- we can't send all the information to the parents like marks list
- sending details through post cards are time consuming processes

### 2.2 PROPOSED SYSTEM

It is a standalone Application and can be used in various institutions like schools, colleges, etc. There are three kinds of users who will be using our application admin and parents, faculty. Parents will be using the application through smart phone. Parents get all the information regarding ward through this application and also if they want any questions they can ask through this application Faculty will get screen to add details of student's attendance and fees structure and notifications etc. They can add all the information to their respective parents. In order to login to the system admin will provide the login and passwords to the faculty. Admin has the all rights for the application and he provides the login and passwords to the parents and faculty. And he has a right for adding and removing the faculty and parents in the database.

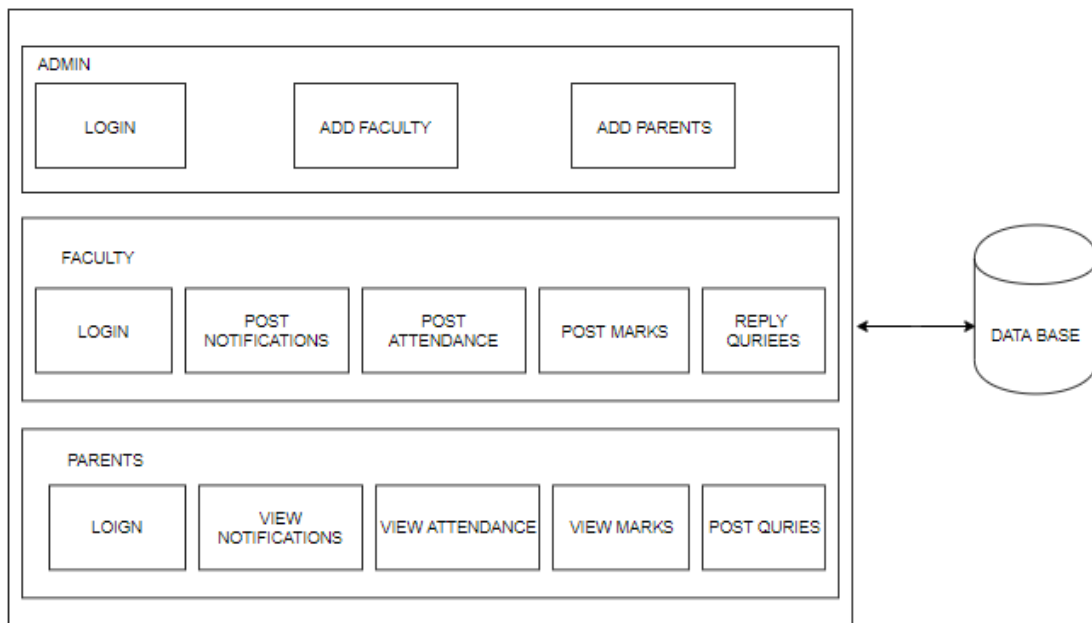


Fig-2.2: System Architecture of PCI

### The Proposed System is provides the following implements

- It is provides the online interface for the faculty and Parents
- Increases the capability of college record management
- Reduces the paper work
- Provides the more secure

#### 2.2.1 ADVANTAGES

- Parents can easily get information through this application
- This is less time consuming process
- Parents can able ask queries through this application
- Minimize the manual work load

### 2.3 SYSTEM STUDY

#### 2.3.1 FEASIBILITY STUDY

It is wise to think about the feasibility of any problem we take on. Feasibility is the study of impact, what happens in the organization by the development of a system. The impact can be either positive or negative. When the positive dominates the negative, then the system is considered feasible. Here the feasibility study can be performed in four ways such as economic feasibility, technical feasibility, operational feasibility, behavioural feasibility.

##### 2.3.1.1 OPERATIONAL FEASIBILITY

The proposed system is beneficial if and only if they are turned into a system that will meet the operating requirements. The best feasibility asks if the system will work when it is developed and installed. The

proposed of the operational feasibility study is to determine whether the new system is used if it is developed and implemented. Proposed system will be used frequently as it satisfies communication needs, hence operational feasibility is assured.

### *2.3.1.2 TECHNICAL FEASIBILITY*

According to feasibility analysis procedure the technical feasibility of the system is analyzed and the technical requirements such as software facilities, procedure, inputs are identified. It is also one of the important phases of the system development activities. All resources needed for the development of the software as well as the maintenance of the same is available. Here we are utilizing the resources, which are already available.

### *2.3.1.3. BEHAVIOURAL FEASIBILITY*

People are inherently resistant to change and computer has been known to facilitate changes. An estimate should be made of how strong the user is likely to move towards the development of computerized system. These are various levels of users in order to ensure proper authentication and authorization and security of sensitive data of the organization.

### *2.3.1.4 ECONOMIC AND FINANCIAL FEASIBILITY*

Economic analysis is most frequently used for evaluation of the effectiveness of the system. More commonly known as cost/benefit analysis the procedure is to determine the benefit and saving that are expected from a system and compare them with costs, decisions is made to design and implement the system. This part of feasibility study gives the top management the economic justification for the new system. Economic analysis that gives actual comparison of costs and benefits is much more meaningful in such cases. In the system, the organization is most satisfied by economic feasibility.

## **3. SYSTEM DESIGN**

### **3.1 PROBLEM DESCRIPTION**

We have seen from the past several years SMS based system. This is used mainly in the colleges and schools etc. In these system colleges sends the information to the number which is given by the parents. This system permits the parents for sending the SMS to the number which is before registered number for getting the information. After receives the SMS, this system automatically queries the database for the information of the student which is based on the mobile number. And receives all the information about a particular student and sends as reply that information.

### **3.2 PROPOSED SOLUTION**

It is a standalone application and can be used in various institutions like schools, colleges etc. There are two kinds of users who will be using this application parents and faculty. And admin is the main user who will have a authority of this application. Admin has all rights about this application. Admin adds the parents and faculty to these applications and provides the login details to them. And admin has the rights to remove the parents and faculty from the application. Faculty will add details of students like attendance and fee dues, marks, notification and day to day events which are conducted in the college. Parents receive all the

information which is sends by the faculty. And also if parents want to know about their ward they can ask through the quires.

### 3.3 SYSTEM ANALYSIS MODELS

Once the analysis stage is completed, the next stage is to determine in broad outline form how the problem might be solved. During system design, we are beginning to move from the logical to physical level.

System design involves architectural and detailed design of the system. Architectural design involves identifying software components, decomposing them into processing modules and conceptual data structures, and specifying the interconnections among components.

Detailed design is concerned with how to package processing modules and how to implement the processing algorithms, data structures and interconnections of standard algorithms, invention of new algorithms, and design of data representations and packaging of software products. Two kinds of approaches are available:

Top down approach

Bottom up approach

#### Bottom up Approach

Design being performed from smallest and lowest level modules one at a time. For each module in bottom up approach a short idea provided in order the needed approach so, that the module is asked to perform the way it will when embedded within the large system. When bottom level modules are tested attention turns to those on the next level that use the lower level once they are designed individually and then liked with the previously examined lower level modules.

#### Top down Approach

This type of design starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written.

### UML MODELLING

In the field of software engineering, the **Unified Modelling Language (UML)** is a standardized visual specification language for object modelling. UML is a general-purpose modelling language that includes a graphical notation used to create an abstract model of a system, referred to as a UML model The model also contains a "Semantic backplane" — documentation such as written use cases that drive the model elements and diagrams.

#### Importance of UML in Modelling

A modelling language is a language whose vocabulary and rules focus on the conceptual and physical representation of a system. A modelling language such as UML is thus a standard language for software blueprints. The UML is not a visual programming language, but its models can be directly connected to various programming languages. This means that it is possible to map from a model in the UML to a programming language Java, C++ or Visual Basic, or even to tables in relational database or the persistent store of an object oriented database. This mapping permits forward engineering: the generation of code from a UML model into a programming language.

The reverse is also possible you can reconstruct a model from an implementation back into UML. This is a programming language that is used for object-oriented software development.

To organize program code more efficiently, programmers often create "objects" that are sets of structured data within programs. UML, which has been standardized by the Object Management Group (OMG), was designed for this purpose. The language has gained enough support that it has become a standard language for visualizing and constructing software programs. The design gives the blueprint of project, so it should be

done perfectly without any error .On design only all the other phases of development are present. We use UML language for it.

These things are the basic object oriented building blocks of the UML. They are used to write well-formed models.

### Diagrams in UML

Diagrams are graphical presentation of set of element. Diagrams project a system, or visualize a system from different angles and perspectives. The UML has nine diagrams these diagrams can be classified into the following groups.

#### Static

1. Class diagrams.
2. Object diagrams.
3. Component diagrams.
4. Deployment diagrams.

#### Dynamic

1. Use case diagram.
2. Sequence diagram.
3. Collaboration diagram.
4. State chart diagram.
5. Activity diagram.

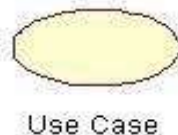
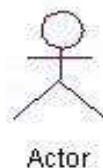
In design phase the system and the software design is prepared from the requirement specifications which were studied in the requirements specification phase. Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. In system design, a design constraint refers to some limitation on the conditions under which a system is developed, or on the requirements of the system.

The design constraint could be on the systems form, fit or function or could be in the technology to be used, materials to be incorporated, time taken to develop the system, overall budget, and so on.

### 3.3.1 USE CASE DIAGRAM

It shows a set of use cases and actors and their relationships. These diagrams illustrate the static use case view of a system and are important in organizing and modelling the behaviours of a system.

The two main components of a use case diagram are use cases and actors.



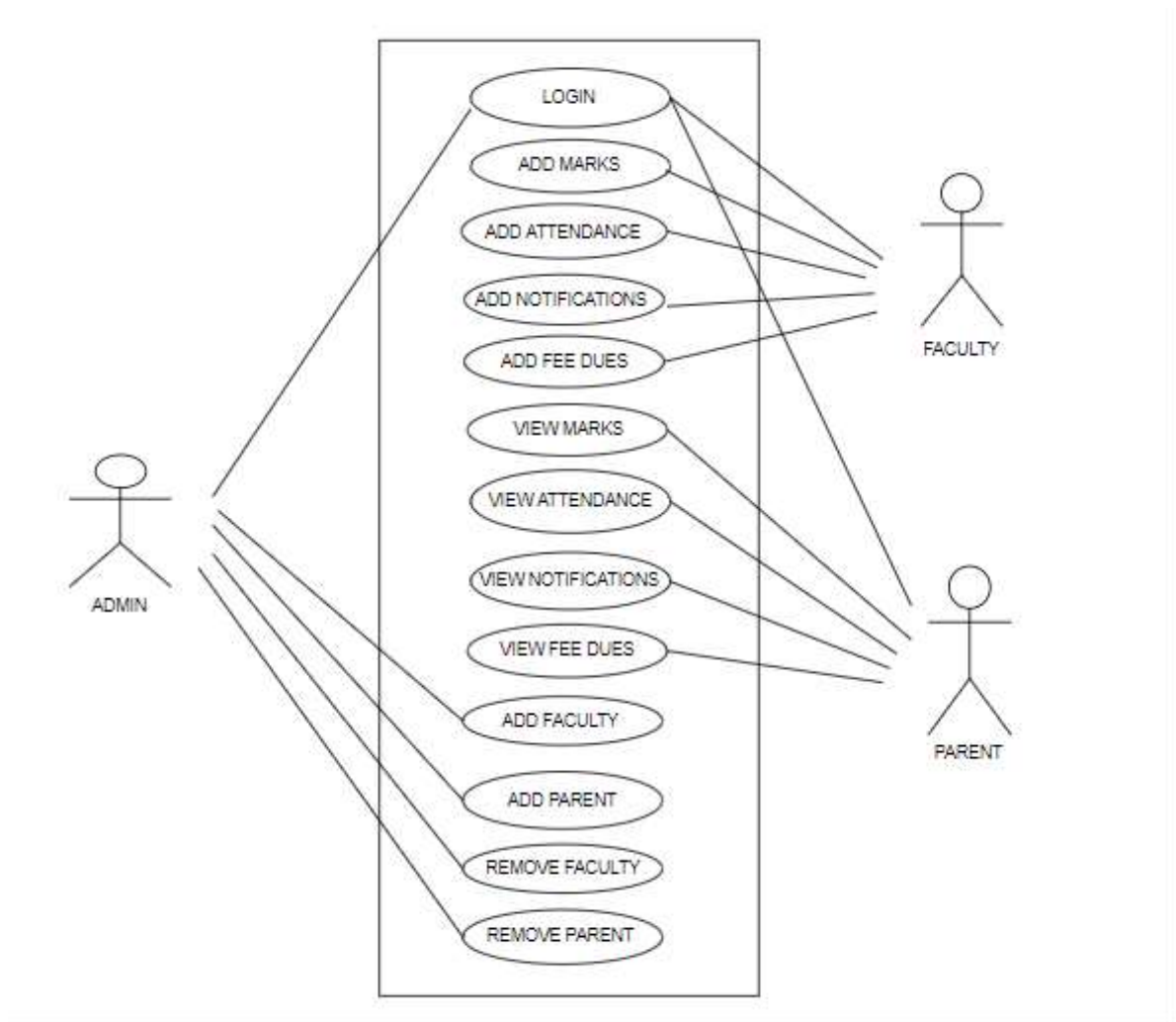


Fig-3.3.1: Use case diagram for PCI

**Description:**

The present use case diagram shows the use cases of the admin, parent and faculty. The admin gives the login details to the parent and faculty. And Faculty adds the marks, attendance, notifications, fee dues etc. And parents view the all information which is send by the faculty.

**3.3.2 ACTIVITY DIAGRAM**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

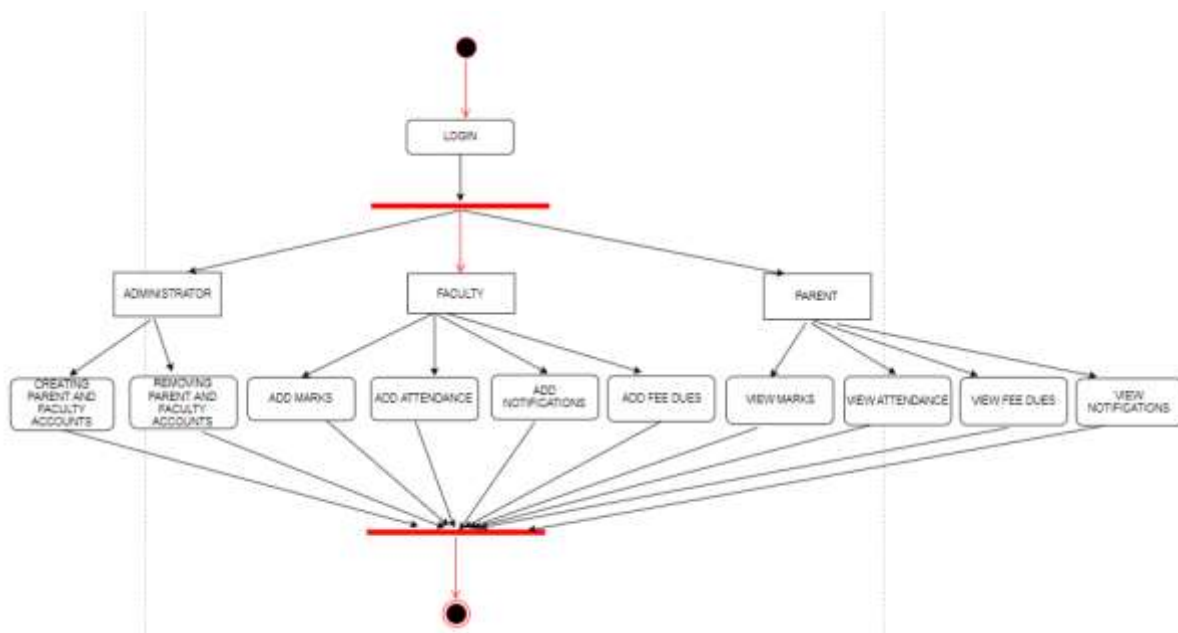


Fig-3.3.2: Activity Diagram for PCI

**Description:**

In this Activity diagram shows what actions performed by the admin, faculty and parent. Main actions are when faculty sends the information parents receive that information.

**3.4 SYSTEM REQUIREMENTS**

System Requirements Specification (SRS) the requirements work product that formally specifies the system-level requirements of a single system or an application. The System Requirements Specification identifies, defines and clarifies the requirements, that when satisfied through development meet the operational/functional need identified in the Project Concept Proposal, Project Business Case, and Project Charter. Approval of this document constitutes agreement that the developed system satisfying these requirements will be accepted.

**FUNCTIONAL REQUIREMENTS**

- Parents should receive the information
- Admin provide the login and password to the parents and faculty
- Faculty sends the details of the students to their parents

**NON-FUNCTIONAL REQUIREMENTS**

- Time and cost effective
- Portable and can be installed on any android platform
- Provide security through password authentication
- The system should provide the reliability

**3.4.1 SOFTWARE REQUIREMENTS**

- Language used : Java
- Front end : XML
- Database : DDMS
- Operating system : Windows 8.1
- Software : Eclipse



### 3.4.2 HARDWARE REQUIREMENTS

- Processor : I3 PROCESSOR
- RAM : 4 GB
- HDD : 250 GB

### 3.5 SYSTEM DESIGN

In design phase the system and the software design is prepared from the requirement specifications which were studied in the requirements specification phase. Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The main design of this project is to maintain different student's certification information at a single area and to assign the staff for counselling.

#### 3.5.1 DESIGN CONCEPTS

Design concept becomes the framework for all our design decisions.

Creating a Design Concept using

- Android UI Controls
- Activities
- Intents
- Fragments
- User interactions
- Layout
- Screen size

### DESIGN CONSTRAINTS

In system design, a design constraint refers to some limitation on the conditions under which a system is developed, or on the requirements of the system. The design constraint could be on the systems form, fit or function or could be in the technology to be used, materials to be incorporated, time taken to develop the system, overall budget, and so on.

### CONCEPTUAL DESIGN

Conceptual Design is an umbrella term given to all forms of non-aesthetic design management disciplines. It is an early phase of the design process, in which the broad outlines of function. It is the creation and exploration of new ideas. It is distinguished from conceptual art by closely relating to function.

### LOGICAL DESIGN

Logical design is a conceptual, abstract design. It deals only with defining the types of information that needed. The process of logical design involves arranging data into a series of logical relationships called entities and attributes.

Logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modelling using an over abstract sometimes graphical model of the actual system.

Logical design is a graphical representation of a system showing the system's processes and the flows of data into and out of the processes. Logical design is used to document the information systems because we can represent the logical nature of a system-what tasks the system is doing, without having to specify how, where or

by whom the tasks are being accomplished. To represent logical design of a system we can use different diagrams.

**Relationships in UML**

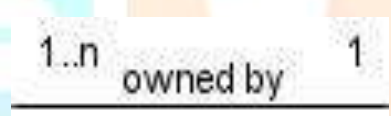
There are different types of relationships in the UML. They are:

1. Dependency
2. Association
3. Aggregation
4. Composition
5. Generalization
6. Realization

**1. Dependency:** This is relationship between two classes whenever one class is completely dependent on the other class. Graphically the dashed line represents it with arrow pointing to the class that it is being depended on.



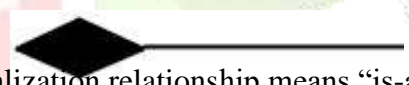
**2. Association:** Association is a generic relationship between two classes and is modeled by a line connecting the two classes. This line also shows the feature multiplicity.



**3. Aggregation:** Aggregation indicates the whole part-of relationship. It is represented by the symbol.



**4. Composition:** Composition relationship means the class is a member of another class. It cannot be present by itself. It is represented by



**5. Generalization:** Generalization relationship means “is-a” relationship. Generalization has a triangle pointing to the super class. It is represented by

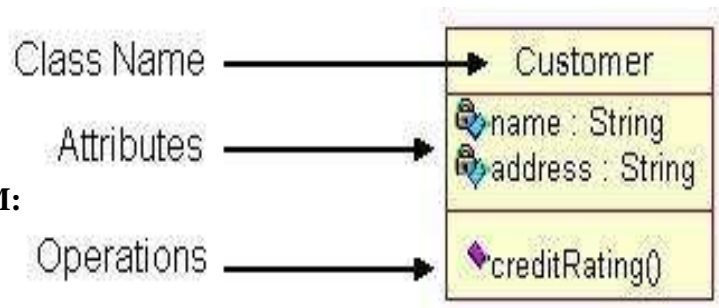


**3.5.2 CLASS DIAGRAM**

This shows a set of classes, interfaces, collaborations and their relationships. There are the most common diagrams in modelling the object oriented systems and are used to give the static view of a system.

Classes are composed of three things: a name, attributes, and operations. Below is an example of a class.

**CLASS DIAGRAM:**



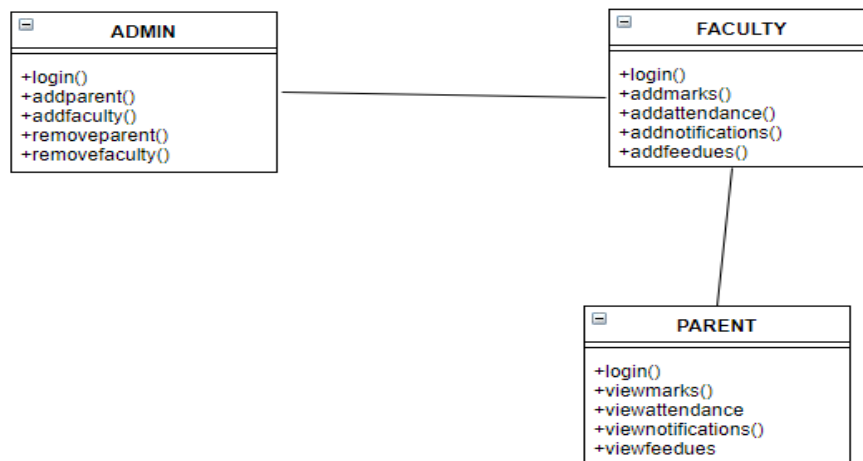


Fig-3.5.2: Class diagram of PCI

**Description:** This class diagram of Parent College Interaction shows the major classes, relationships, attributes and operations of each class. This classes identified in the system are admin, faculty and parents.

### 3.5.3 SEQUENCE DIAGRAM

Sequence diagram is an interaction diagram which focuses on the time ordering of messages it shows a set of objects and messages exchange between these objects. This diagram illustrates the dynamic view of a system.

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

**Description:** This sequence diagram focuses on the overall system. Here the flow of sequence among the admin, pci, faculty and parent. Here the sequence of actions such as enters the marks, notifications, fee dues, attendance.

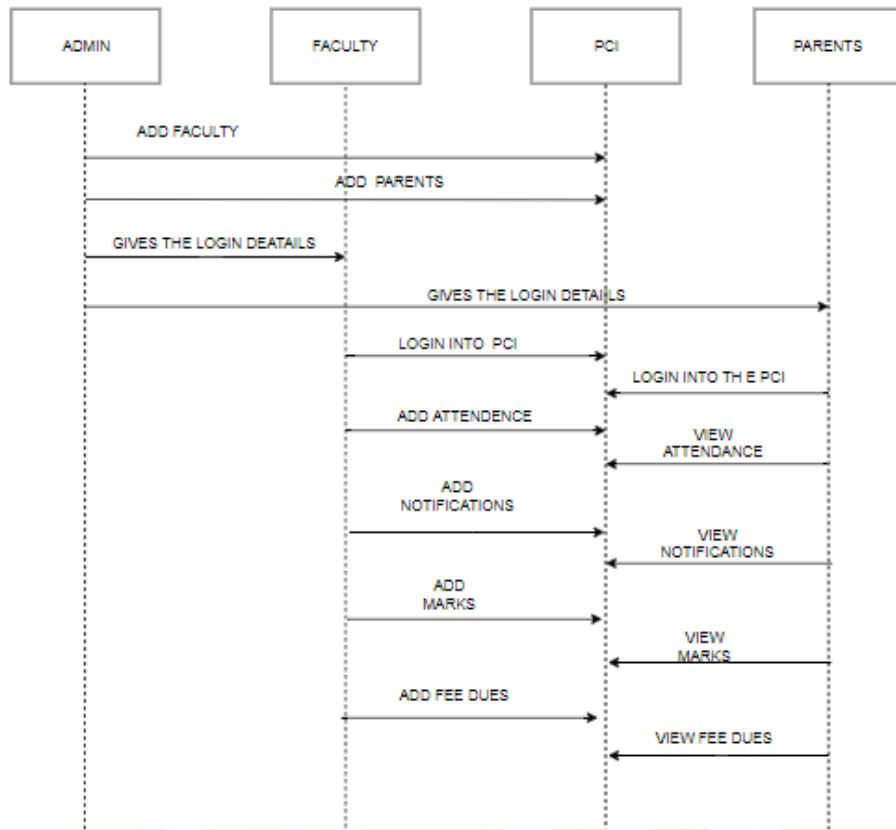


Fig-3.5.3: Sequence Diagram of PCI

### 3.6 E-R DIAGRAM

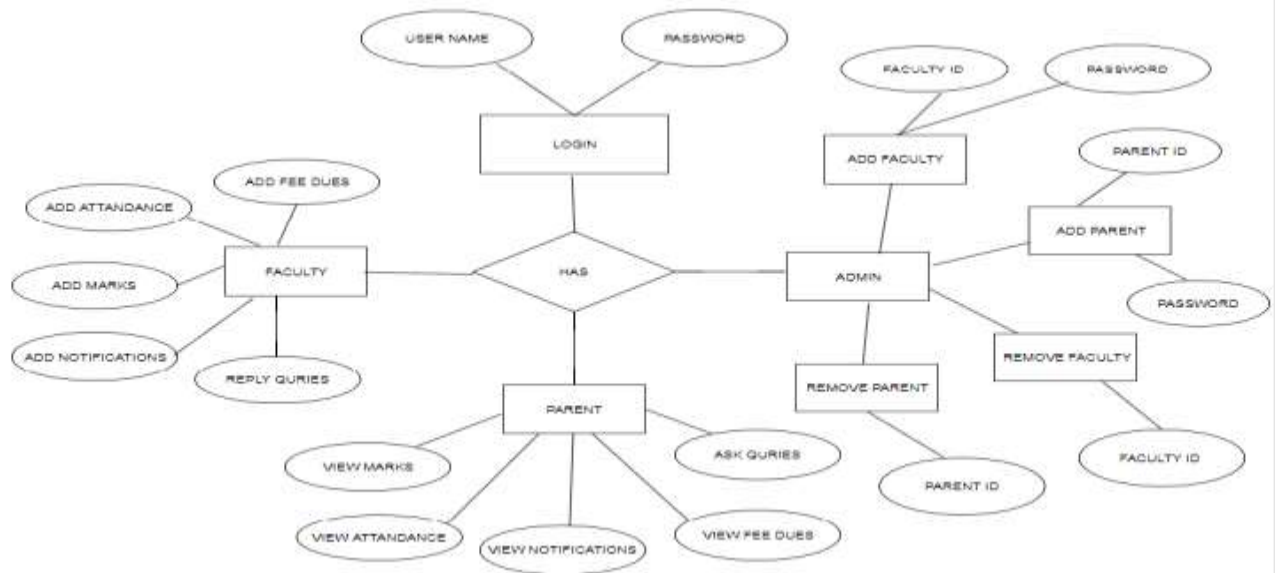


Fig-3.6: E-R Diagram

## 4. IMPLEMENTATION

### 4.1 TOOLS USED

The various technologies used in our smart menu application are as follows. They are:

- ❖ Java
- ❖ XML
- ❖ Software Development Kit(SDK)
- ❖ ADT plug-in

#### 4.1.1 JAVA

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan at Sun Microsystems Inc. in 1991. This language was initially called “Oak” but was renamed as “Java” in 1995. This is mostly used programming language because of its various features.

The various features of the java, which made java so popular, are as follows:

- ❖ Simple
- ❖ Secure
- ❖ Portable
- ❖ Object-oriented
- ❖ Robust
- ❖ Multithreaded
- ❖ Architecture-neutral
- ❖ Interpreted
- ❖ Distributed
- ❖ Dynamic

#### Java Environment

Java environment includes a large number of development tools and hundreds of classes and methods. The development tools are part of the system known as *Java Development Kit (JDK)*, the classes and methods are part of the *Application Programming Interface (API)*.

The word "Java", alone, usually refers to Java programming language that was designed for use with the Java platform. For the connectivity purpose that is to connect the frontend and backend, this application uses java technology. Java is a platform independent one. It is considered to be secure as it follows object oriented paradigm.

In this application we use Java Server Pages (JSP). It is a technology for developing web pages that support dynamic content which helps developers insert java code in HTML pages by making use of special JSP tags. A Java Server Pages component is a type of Java servile that is designed to fulfil the role of a user interface for a Java web application.

JSP is used to collect input from users through web forms, present records from a database or another source. JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages and sharing information between requests, pages etc.

In this application authentication is required for users. They specify the user id and password through forms. JSP enables to pass the user id and password to the database and authenticate them and retrieve the result from the database. Similarly when the operator enters student information, JSP enables to store the data in database. Whenever we need to view some data we can view it which is stored in the database.

### 4.1.2 XML

Extensible Mark-up Language (XML) is the predominant markup language for web pages. XML is designed to transport and store the data. XML tags are not predefined. We can define your own tags. With XML data can be stored in separate XML files. This makes it much easier to create data that can be shared by different applications.

XML data is stored in Text format. This makes it easier to expand or upgrade to new operating system, new applications, or new browsers without losing the data. The syntax of XML is very simple and logical. An element can contain text, attributes, other elements, or mixture of all of the above.

**XML simplifies data interchange:** Because different organizations (or even different parts of the same organization) rarely standardize on a single set of tools, it can take a significant amount of work for applications to communicate. Using XML, each group creates a single utility that transforms their internal data formats into XML and vice versa. Best of all, there's a good chance that their software vendors already provide tools to transform their database records (or LDAP directories, or purchase orders, and so forth) to and from XML.

**XML enables smart code:** Because XML documents can be structured to identify every important piece of information (as well as the relationships between the pieces), it's possible to write code that can process those XML documents without human intervention. The fact that software vendors have spent massive amounts of time and money building XML development tools means writing that code is a relatively simple process.

**XML enables smart searches:** Although search engines have improved steadily over the years, it's still quite common to get erroneous results from a search. If you're searching HTML pages for someone named "Chip," you might also find pages on chocolate chips, computer chips, wood chips, and lots of other useless matches. Searching XML documents for <first-name> elements that contained the text Chip would give you a much better set of results.

The XML specification requires a parser to reject any XML document that doesn't follow the basic rules. Most HTML parsers will accept sloppy markup, making a guess as to what the writer of the document intended. To avoid the loosely structured mess found in the average HTML document, the creators of XML decided to enforce document structure from the beginning.

There are two rules for attributes in XML documents:

- Attributes must have values
- Those values must be enclosed within quotation marks

### 4.1.3 ECLIPSE

Eclipse is an integrated development environment (IDE) used in computer programming and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including C , C++, java, .net NATURAL, Perl, Ruby. It can also be used to develop documents (via a Ellipse plug-in) and packages for the software Matimatica.

The initial codebase originated from IBM Visual Age the Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by

installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Since the introduction of the OSGi implementation in version 3 of Eclipse, plug-ins can be plugged-stopped dynamically and are termed (OSGI) bundle.

Eclipse was inspired by the Smalltalk based Visual Age family of Integrated Development Environment (IDE) products. Although fairly successful, a major drawback of the Visual Age products was that developed code was not in a Computer Based software engineering model. Instead, all code for a project was held in a compressed lump (somewhat like a zip file but in a proprietary format called .data). Individual classes could not be easily accessed, certainly not outside the tool. A team primarily at the IBM Cary NC lab developed the new product as a Java-based replacement. In November 2001, a consortium was formed with a board of stewards to further the development of Eclipse as open source software. It is estimated that IBM had already invested nearly \$40 million by that time.

## 4.2 COMPONENT DIAGRAM

It Shows a set of components and their relationships and are used to illustrate the static implementation view of a system. They are related to class diagrams where in components map to one or more classes, interfaces or collaborations.

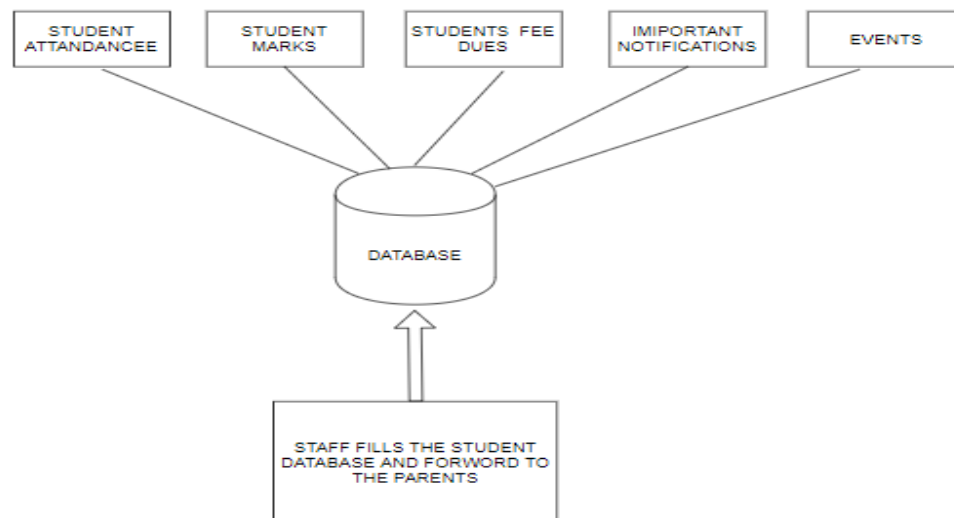


Fig-4.2: student information

## 4.3 PSEDU CODE

### AddParMainActivity.java:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_addpar_main);
    loginDataBaseAdapter= new LoginDataBaseAdapter(this, "",null, 1);
    Button btn1 = (Button) findViewById(R.id.button1);
    e1=(EditText)findViewById(R.id.editText1);
    e2=(EditText)findViewById(R.id.editText2);
```

```

btn1.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        String userName=e1.getText().toString();
        String password=e2.getText().toString();

        // check if any of the fields are vaccant
        if(userName.equals("")||password.equals("") )
        {
            Toast.makeText(getApplicationContext(), "Field   Vaccant",
Toast.LENGTH_LONG).show();
        }
        return;
    }
else{
        loginDataBaseAdapter.insert_parent(userName, password);
        Toast.makeText(getApplicationContext(), "Parent Account Successfully
Created ", Toast.LENGTH_LONG).show();
    }
});
}
}

```

This is the code for the parent login page. When user enters the username and password the values are verified. The login is valid only when the user name and the associate password both are matched. If the either of the fields of username and password are spaces the message is displayed as Field Vacant.

### AdminHomeActivity.java

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_admin_home);

    Button btn1 = (Button) findViewById(R.id.button1); //addfac
    Button btn2 = (Button) findViewById(R.id.button2);//add parent
    Button btn3 = (Button) findViewById(R.id.button3);//rem fac
}

```



```
Button btn4 = (Button) findViewById(R.id.button4);//rem par
```

```
btn1.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View arg0) {
```

```
        // TODO Auto-generated method stub
```

```
        Intent i1 = new Intent(getApplicationContext(), FacMainActivity.class);
        startActivity(i1);
```

```
    }
```

```
});
```

```
btn2.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View arg0) {
```

```
        // TODO Auto-generated method stub
```

```
        Intent i1 = new Intent(getApplicationContext(),
```

```
AddparMainActivity.class);
```

```
        startActivity(i1);
```

```
    }
```

```
});
```

```
btn3.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View arg0) {
```

```
        // TODO Auto-generated method stub
```

```
        Intent i1 = new Intent(getApplicationContext(), RemfacActivity.class);
```

```

        startActivity(i1);
    }

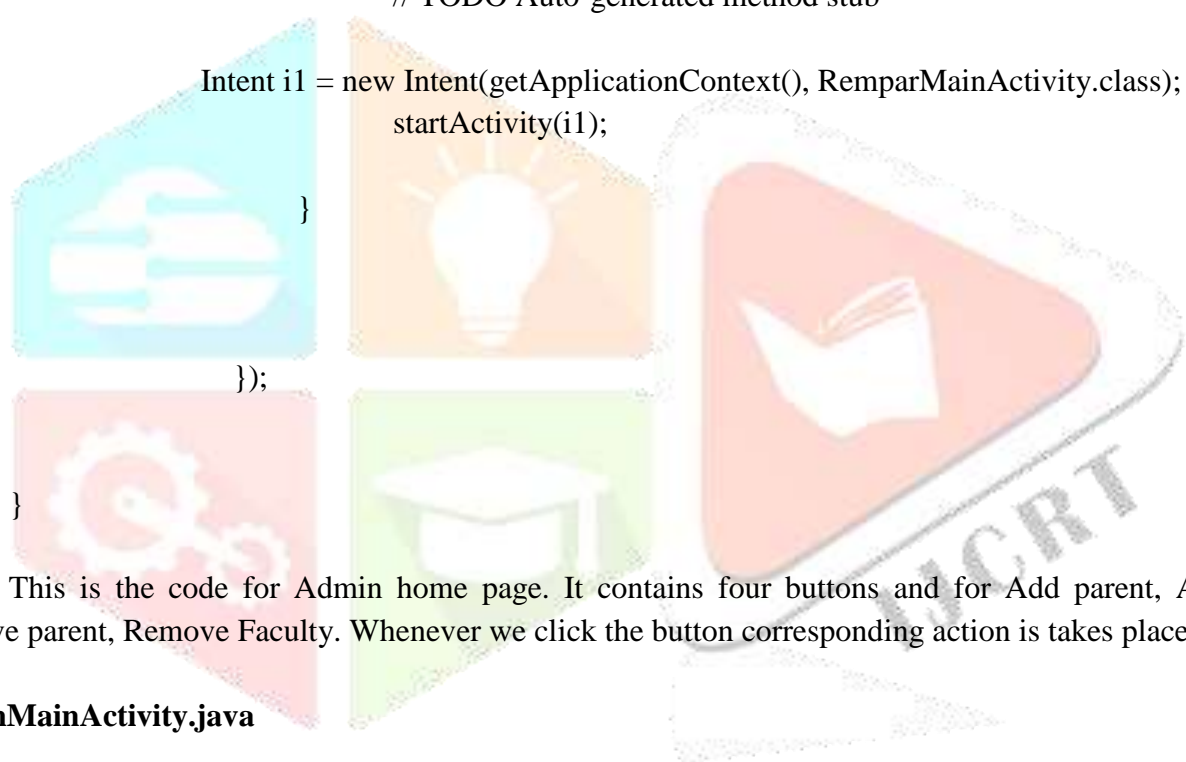
});

btn4.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View arg0) {
        // TODO Auto-generated method stub

        Intent i1 = new Intent(getApplicationContext(), RemparMainActivity.class);
        startActivity(i1);
    }
});
}

```



This is the code for Admin home page. It contains four buttons and for Add parent, Add Faculty, Remove parent, Remove Faculty. Whenever we click the button corresponding action is takes place.

### **AdminMainActivity.java**

```

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_admin_main);

    e1=(EditText)findViewById(R.id.editText1);
    e2=(EditText)findViewById(R.id.editText2);
}

```

```
Button btn1 = (Button) findViewById(R.id.button1); //ADMIN
```

```

btn1.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View arg0) {
        // TODO Auto-generated method stub

        String userName=e1.getText().toString();
        String password=e2.getText().toString();
        if(userName.equals("")||password.equals("")) )
        {
            Toast.makeText(getApplicationContext(), "Field
Vaccant", Toast.LENGTH_LONG).show();
            return;
        }
        if((userName.equals("admin"))&&(password.equals("admin")) ){
            Toast.makeText(getApplicationContext(), "Login Sucess", Toast.LENGTH_LONG).show();
            Intent i1 = new Intent(getApplicationContext(),
AdminHomeActivity.class);
startActivity(i1);
        }
        else{
            Toast.makeText(getApplicationContext(), "Invalid login
details", Toast.LENGTH_LONG).show();
            return;
        }
    }
});
}

```

This is code for Admin login page. When the username and password are matched then the login credentials are validated. And message is displayed as “Login Success”. If either of fields are empty then message is “Fields are Vacant”. If either of the fields are not matched with username and password then message printed as “Invalid Login Details”.

### AttMainActivity.java:

```
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_att_main);

    Bundle _bundle = getIntent().getExtras();
    s = _bundle.getString("name");
    loginDataBaseAdapter= new LoginDataBaseAdapter(this, "",null, 1);

    e1=(EditText)findViewById(R.id.editText1);//y
    e2=(EditText)findViewById(R.id.editText2);//s
    e4=(EditText)findViewById(R.id.editText4);//id

    Button btn1 = (Button) findViewById(R.id.button1); //addnoti

    btn1.setOnClickListener(new View.OnClickListener()
    {
        @Override

    public void onClick(View arg0) {
        // TODO Auto-generated method stub

        String M=e1.getText().toString();
        String A=e2.getText().toString();
        String id=e4.getText().toString();

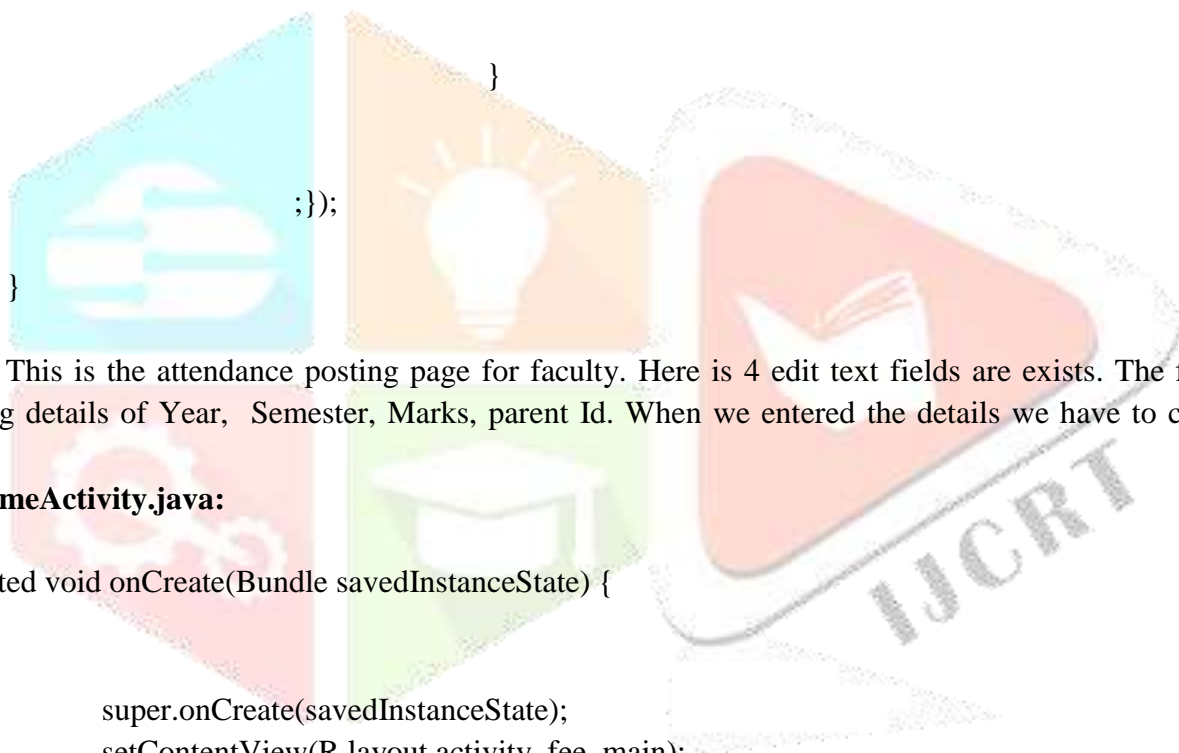
        if(M.equals("")||A.equals("")||id.equals(""))
        {
            Toast.makeText(getApplicationContext(), "Field
Vaccant", Toast.LENGTH_LONG).show();
            return;
        }
    }
}
```

```
else
{
loginDataBaseAdapter.insert_att(s,M,A,id);

Toast.makeText(getApplicationContext(), "Attendance Posted ",
Toast.LENGTH_LONG).show();

}

}
```



This is the attendance posting page for faculty. Here is 4 edit text fields are exists. The fields are for entering details of Year, Semester, Marks, parent Id. When we entered the details we have to click the post button.

#### **FeeHomeActivity.java:**

```
protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);
setContentView(R.layout.activity_fee_main);
Bundle _bundle = getIntent().getExtras();
s = _bundle.getString("name");
loginDataBaseAdapter= new LoginDataBaseAdapter(this, "", null, 1);

e1=(EditText)findViewById(R.id.editText1);//y
e2=(EditText)findViewById(R.id.editText4);//s

Button btn1 = (Button) findViewById(R.id.button1); //addnoti

btn1.setOnClickListener(new View.OnClickListener() {
```

**FacHomeMainActivity.java:**

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_fac_home_main);
    Bundle _bundle = getIntent().getExtras();
    s = _bundle.getString("name");
    Button btn1 = (Button) findViewById(R.id.button1); //addmarks
    Button btn2 = (Button) findViewById(R.id.button2);//add attendance
    Button btn3 = (Button) findViewById(R.id.button3);//add noti
    Button btn4 = (Button) findViewById(R.id.button4);//add fees
```

```

    btn1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {
            // TODO Auto-generated method stub

            Intent i1 = new Intent(getApplicationContext(),
MarksMainActivity.class);
            i1.putExtra("name", s);
            startActivity(i1);
        }
    });

```

```
btn2.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View arg0) {
```

```
        // TODO Auto-generated method stub
```

```
        Intent i1 = new Intent(getApplicationContext(), AttMainActivity.class);
```

```
        i1.putExtra("name", s);
```

```
        startActivity(i1);
```

```
    }
```

```
});
```

```

btn3.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View arg0) {
        // TODO Auto-generated method stub

        Intent i1 = new Intent(getApplicationContext(),NotiMainActivity.class);
        i1.putExtra("name", s);
        startActivity(i1);
    }
});

btn4.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View arg0) {
        // TODO Auto-generated method stub

        Intent i1 = new Intent(getApplicationContext(),FeeMainActivity.class);
        i1.putExtra("name", s);
        startActivity(i1);
    }
});
}

```

This is the code for faculty home page. Here is the buttons for add fees dues and add attendance, adding notifications and adding marks.

### **ParHomeMainActivity.java:**

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_par_home_main);
}

```

```

Button btn1 = (Button) findViewById(R.id.button1); //MARKS
Button btn2 = (Button) findViewById(R.id.button2);//ATT
Button btn3 = (Button) findViewById(R.id.button3);//NOTI
Button btn4 = (Button) findViewById(R.id.button4);//FEE
Bundle _bundle = getIntent().getExtras();
    s = _bundle.getString("name");

```

```

btn1.setOnClickListener(new View.OnClickListener() {

```

```

    @Override
    public void onClick(View arg0) {
        // TODO Auto-generated method stub

```

```

Intent i1 = new Intent(getApplicationContext(), MarksParMainActivity.class);
i1.putExtra("name", s);

```

```

startActivity(i1);

```

```

    }
});

```

```

btn2.setOnClickListener(new View.OnClickListener() {

```

```

    @Override
    public void onClick(View arg0) {
        // TODO Auto-generated method stub

```

```

Intent i1 = new Intent(getApplicationContext(),
ParAttMainActivity.class);

```

```

i1.putExtra("name", s);
startActivity(i1);

```

```

    }

```

```

});

```

```

btn3.setOnClickListener(new View.OnClickListener() {

```



```

@Override
public void onClick(View arg0) {
    // TODO Auto-generated method stub

    Intent i1 = new Intent(getApplicationContext(), ParNotiActivity.class);
    i1.putExtra("name", s);
    startActivity(i1);
}

```

```
});
```

```
btn4.setOnClickListener(new View.OnClickListener() {
```

```

@Override
public void onClick(View arg0) {
    // TODO Auto-generated method stub

```

```
Intent i1 = new Intent(getApplicationContext(), PArFeeMainActivity.class);
```

```

i1.putExtra("name", s);
startActivity(i1);
}

```

```
});
```

```
}
```

Here is the code for viewing the details posted by the faculty. These are the buttons for viewing the attendance, marks, notifications and fees dues. When we click the buttons the corresponding information is displayed.

#### **ParntMainActivity.java :**

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

```

```

setContentview(R.layout.activity_parnt_main);
Button btn1 = (Button) findViewById(R.id.button1);
e1=(EditText)findViewById(R.id.editText1);
e2=(EditText)findViewById(R.id.editText2);
loginDataBaseAdapter= new LoginDataBaseAdapter(this, "",null, 1);

btn1.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        String userName=e1.getText().toString();
        String password=e2.getText().toString();

        // check if any of the fields are vaccant
        if(userName.equals("")||password.equals("") )
        {
            Toast.makeText(getApplicationContext(), "Field Vaccant",
Toast.LENGTH_LONG).show();
            return;
        }
        else{

            int i= loginDataBaseAdapter.check2(userName, password);
            if(i==0)

                Toast.makeText(getApplicationContext(), "Invalid username/password",
Toast.LENGTH_LONG).show();

            else
            {
                Intent i1 = new Intent(getApplicationContext(),
ParHomeMainActivity.class);
                i1.putExtra("name", userName);
                startActivity(i1);
                //Toast.makeText(getApplicationContext(), "Success",
Toast.LENGTH_LONG).show();
            }
        }
    }
}

```

```

    }
    });
}

```

This is the code for parent login page. If once login is validated then the button is redirected to the parent home page. Then he become enable to perform required action. If login details are not validated then message is displayed as “Invalid Login”. If there is any empty details are there then message displayed as “Field vacant”.

### NotiMainActivity.java :

```

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_noti_main);

    Bundle _bundle = getIntent().getExtras();
    s = _bundle.getString("name");
    loginDataBaseAdapter= new LoginDataBaseAdapter(this, "",null, 1);

    e1=(EditText)findViewById(R.id.editText1);

    Button btn1 = (Button) findViewById(R.id.button1); //addnoti

    btn1.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View arg0) {
            // TODO Auto-generated method stub

            String noti=e1.getText().toString();

            if(noti.equals(""))
            {
                Toast.makeText(getApplicationContext(), "Field
                Vaccant", Toast.LENGTH_LONG).show();
            }
            return;
        }
    }
}

```

```

        else
        {
            loginDataBaseAdapter.insert_noti(s,noti);
            Toast.makeText(getApplicationContext(), "Notification Posted ",
Toast.LENGTH_LONG).show();
        }
    }
});
}
}

```

This is the code for posting notifications. Before performing this activity the faculty should login. After posting this notification the message is displayed as “Notification is posted”.

#### **ParNotiActivity.java :**

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_par_noti);
    v1=(TextView)findViewById(R.id.textView1);
    Bundle _bundle = getIntent().getExtras();
    s = _bundle.getString("name");
    loginDataBaseAdapter= new LoginDataBaseAdapter(this, "",null, 1);
    loginDataBaseAdapter.List_all_Students4(v1);
}

```

Here the notification posted by the faculty is displayed by this file.

#### **ParAttMainActivity :**

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_par_att_main);
    v1=(TextView)findViewById(R.id.textView1);
    Bundle _bundle = getIntent().getExtras();
    s = _bundle.getString("name");
    loginDataBaseAdapter= new LoginDataBaseAdapter(this, "",null, 1);
    loginDataBaseAdapter.List_all_Students(v1,s);
}

```

```
}
```

After login and clicking the button marks the marks of the student is displayed.

### **LogInDatabaseAdapter:**

```
public class LoginDataBaseAdapter extends SQLiteOpenHelper
```

```
{
```

```
    public LoginDataBaseAdapter(Context context, String name,  
        SQLiteDatabase.CursorFactory factory, int version) {  
        super(context, "PCI2.db", null, version);  
        // TODO Auto-generated constructor stub
```

```
}
```

```
public void insert_faculty(String u,String p)
```

```
{
```

```
    ContentValues contentvalues=new ContentValues();  
    contentvalues.put("USERNAME", u);  
    contentvalues.put("PASSWORD", p);  
    this.getWritableDatabase().insertOrThrow("faculty", "", contentvalues);
```

```
}
```

```
public void insert_feedue(String u,String fee,String sid)
```

```
{
```

```
    ContentValues contentvalues=new ContentValues();  
    contentvalues.put("USERNAME", u);  
    contentvalues.put("fee", fee);  
    contentvalues.put("sid", sid);  
    this.getWritableDatabase().insertOrThrow("feedue", "", contentvalues);
```

```
}
```

```
public void insert_marks(String u,String y,String s,String m,String id1)
```

```
{
```

```
    ContentValues contentvalues=new ContentValues();  
    contentvalues.put("fname", u);  
    contentvalues.put("year1", y);  
    contentvalues.put("sem", s);  
    contentvalues.put("marks", m);  
    contentvalues.put("sid", id1);  
    this.getWritableDatabase().insertOrThrow("marks", "", contentvalues);
```

```
}
public void insert_att(String u,String M,String A,String id1)
{
ContentValues contentvalues=new ContentValues();
contentvalues.put("fname", u);
contentvalues.put("mon", M);
contentvalues.put("atten", A);

contentvalues.put("sid", id1);
this.getWritableDatabase().insertOrThrow("att", "", contentvalues);

}

public void insert_parent(String u,String p)
{
ContentValues contentvalues=new ContentValues();
contentvalues.put("USERNAME", u);
contentvalues.put("PASSWORD", p);
this.getWritableDatabase().insertOrThrow("parent", "", contentvalues);
}
public void deleteRow(String value)
{
SQLiteDatabase db = this.getWritableDatabase();
db.execSQL("DELETE FROM faculty WHERE username='"+value+"'");
db.close();
}

public void deleteRow1(String value)
{
SQLiteDatabase db = this.getWritableDatabase();
db.execSQL("DELETE FROM parent WHERE username='"+value+"'");
db.close();
}
public void insert_noti(String u,String n)
{
ContentValues contentvalues=new ContentValues();
contentvalues.put("USERNAME", u);
contentvalues.put("noti", n);
this.getWritableDatabase().insertOrThrow("notification1", "", contentvalues);
```

```
}  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        // TODO Auto-generated method stub  
        db.execSQL("create table faculty(ID integer primary key autoincrement,USERNAME  
text,PASSWORD text);");  
  
        db.execSQL("create table notification1(ID integer primary key  
autoincrement,USERNAME text,noti text);");  
  
        db.execSQL("create table marks(ID integer primary key autoincrement,fname text,year1  
text,sem text,marks text,sid text);");  
  
        db.execSQL("create table feedue(ID integer primary key autoincrement,username  
text,fee text,sid text);");  
  
        db.execSQL("create table att(ID integer primary key autoincrement,fname text, mon  
text,atten text,sid text);");  
  
        db.execSQL("create table parent(ID integer primary key autoincrement,USERNAME  
text,PASSWORD text);");  
        //db.execSQL("create table feedback(ID integer primary key  
autoincrement,USERNAME text,fd text);");  
    }  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
        // TODO Auto-generated method stub  
        db.execSQL("drop table if exists faculty;");  
        db.execSQL("drop table if exists att;");  
        db.execSQL("drop table if exists parent;");  
        db.execSQL("drop table if exists notification1;");  
        db.execSQL("drop table if exists feedue;");  
        db.execSQL("drop table if exists marks;");  
        //db.execSQL("drop table if exists feedback;");  
        onCreate(db);  
    }  
  
    public void List_all_Students(Textview v,String u)  
    {
```

```

Cursor c=this.getReadableDatabase().rawQuery("select * from att where sid=?",new
String[]{u});
v.setText("");
v.append("Month\tAttendance\n");
while(c.moveToNext())
{
    v.append(c.getString(2)+"\t"+c.getString(3)+"\n");
}
}
}
public void List_all_Students2(TextView v,String u)
{
Cursor c=this.getReadableDatabase().rawQuery("select * from feedue where sid=?",new
String[]{u});
v.setText("");
v.append("Due\n");
while(c.moveToNext())
{
    v.append(c.getString(2)+"\n");
}
}
}
public void List_all_Students3(TextView v,String u)
{
Cursor c=this.getReadableDatabase().rawQuery("select * from marks where sid=?",new
String[]{u});
v.setText("");
v.append("year\t\tsem\t\tmarks\n");
while(c.moveToNext())
{
    v.append(c.getString(2)+"\t"+c.getString(3)+"\t"+c.getString(4)+"\n");
}
}
}
public void List_all_Students4(TextView v)
{
Cursor c=this.getReadableDatabase().rawQuery("select * from notification1",null);
v.setText("");
v.append("Notification\n");
while(c.moveToNext())
{
    v.append(c.getString(2)+"\n");
}
}
}
public void Get_Students(TextView v,String u)

```



```

    {
        SQLiteDatabase db=this.getReadableDatabase();
        String amt="0";
        Cursor c=db.rawQuery("SELECT * FROM userreg WHERE USERNAME=?",new
String[] {u});
        //Cursor c=this.getReadableDatabase().rawQuery("select * from userreg where
USERNAME="+u+"", null);
        v.setText("");
        //v.append("name password busno sstop estop passtype\n");
        while(c.moveToNext())
        {
            v.append("name: "+c.getString(1)+"\n"+"password: "+c.getString(2)+"\nbusno:
"+c.getString(3)+"\ns_stop: "+c.getString(4)+"\ne_stop: "+c.getString(5)+"\nAmt: "+c.getString(6)+"\n");

            if(c.getString(3).equals("10"))
                amt="1000";
            if(c.getString(3).equals("20"))
                amt="2000";
            if(c.getString(3).equals("30"))
                amt="3000";
            v.append("Renewal Amt"+amt);
        }
    }
    public int check1(String U,String p)
    {
        SQLiteDatabase db=this.getReadableDatabase();
        Cursor cursor=db.rawQuery("SELECT * FROM faculty WHERE USERNAME=? AND
password=?",new String[] {U,p});
        if(cursor.getCount()>
0)

            return 1;

        else
            return 0;
    }

    public int check2(String U,String p)
    {

```

```

        SQLiteDatabase db=this.getReadableDatabase();
        Cursor cursor=db.rawQuery("SELECT * FROM parent WHERE USERNAME=? AND
password=?",new String[]{U,p});
        if(cursor.getCount()>
0)

        return 1;

else
return 0;
    }
    public void List_all_feedback(Textview v)
    {
        Cursor c=this.getReadableDatabase().rawQuery("select * from feedback", null);
        v.setText("");
        v.append("name \tfeedback\n");
        while(c.moveToNext())
        {
            v.append(c.getString(1)+"\t"+c.getString(2)+"\n");
        }
    }
}

```

This is the code for implementing all functionalities. Once the admin login in insert\_faculty method the faculty name is added to database. The admin has the other functionalities like delete faculty, delete parent, insert parent. For inserting fees dues the method insert\_feesdue is added. For adding the faculty first login and then enter the amount of feesdue.

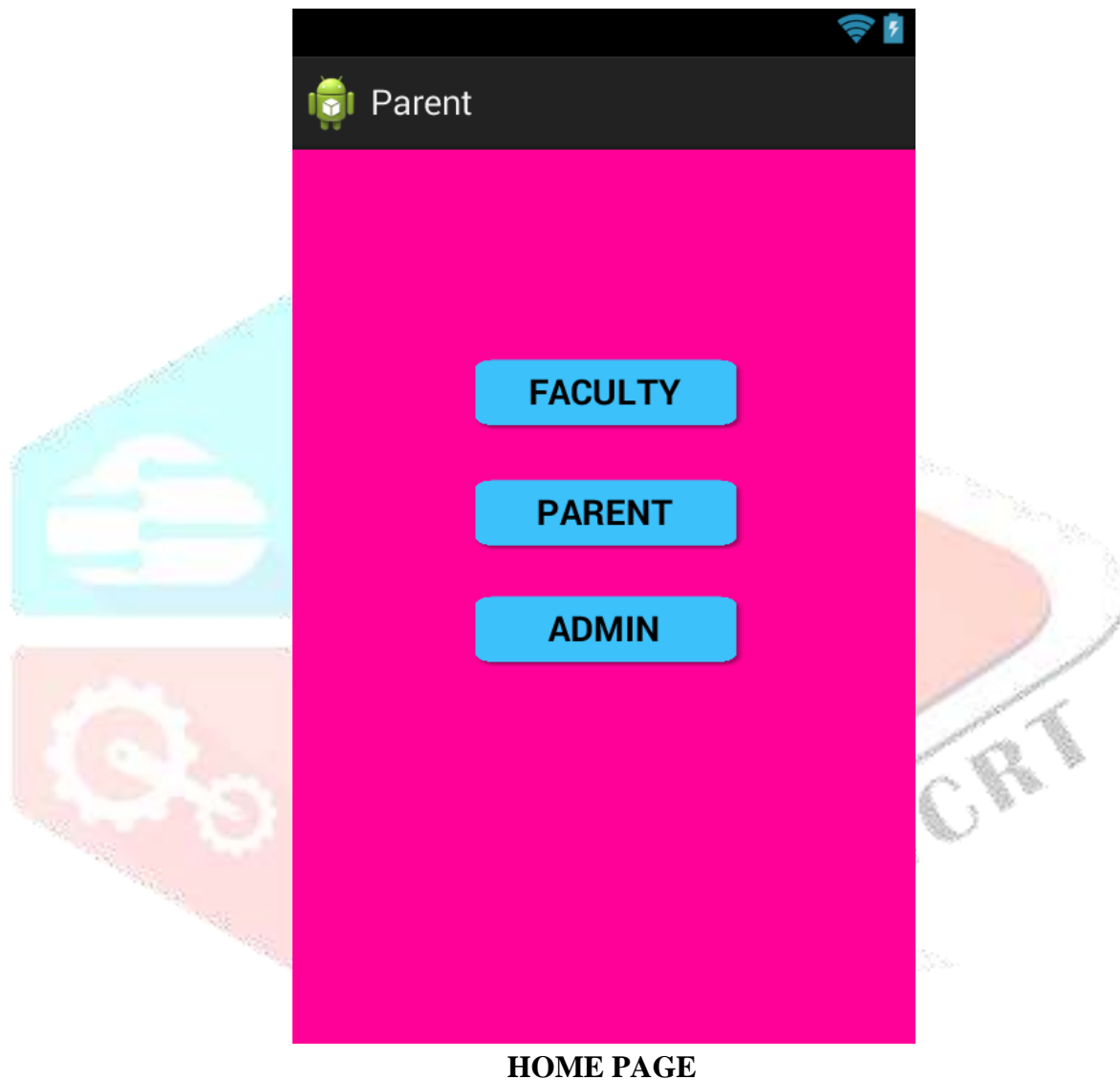
For inserting marks of the student the faculty has to login first and then the faculty has to add the following details.

1. Year
2. semester
3. marks
4. parent username

Similarly for adding attendance the faculty should login and next faculty has to enter details like month, attendance percentage of that month and the parent username.

For inserting notification also the faculty must login. Later the faculty has to add the corresponding notification.

#### 4.4 SCREENS

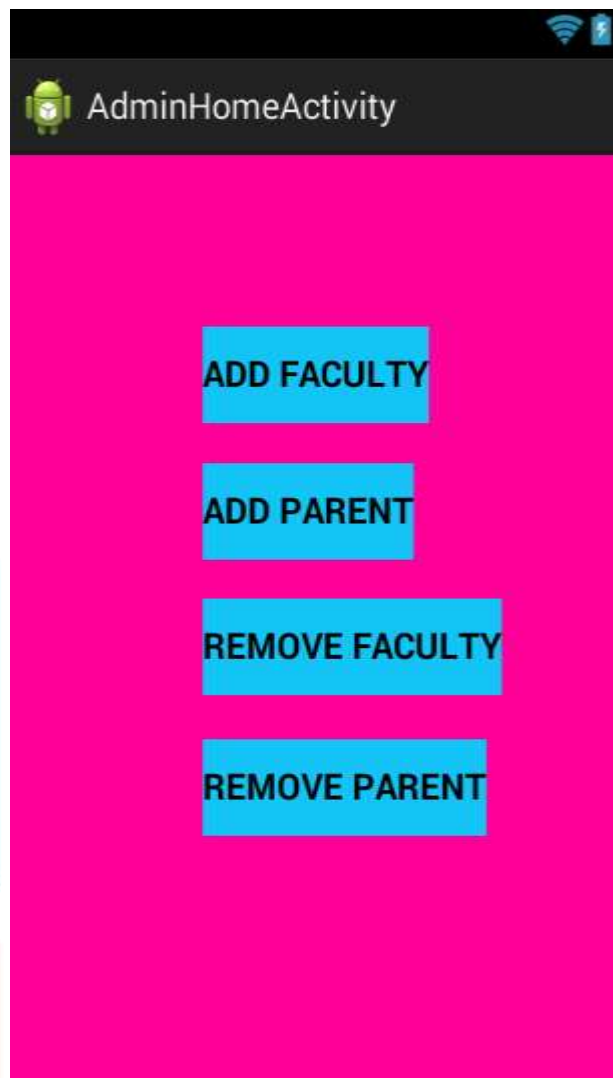


**DESCRIPTION:** Home page for the parent college interaction. This is the starting page for the android application.



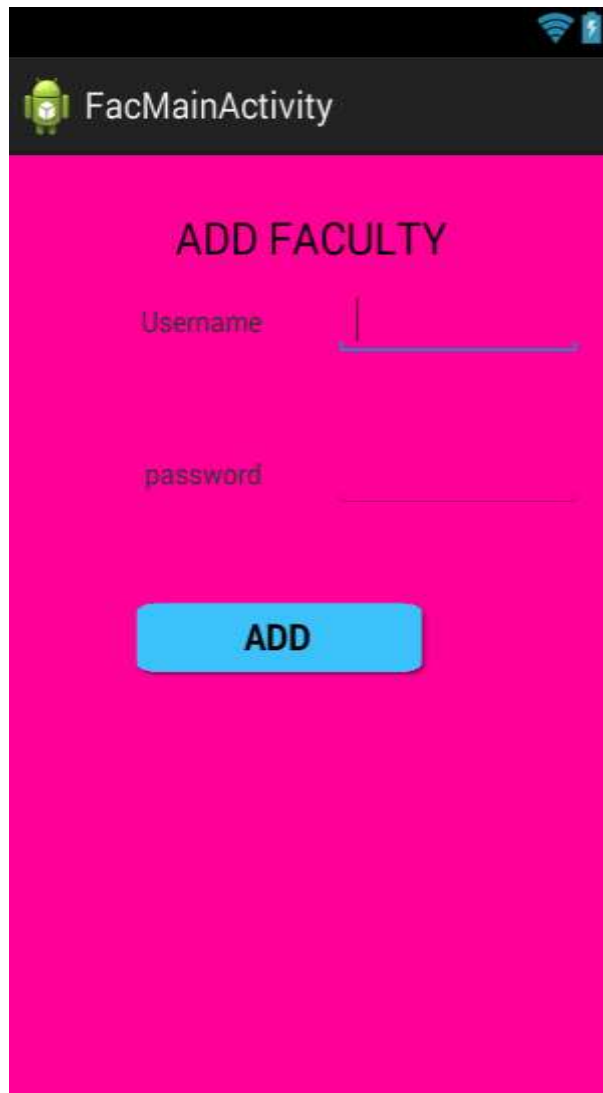
**ADMIN LOGIN**

**DESCRIPTION:** Admin login into the application for performing the actions. Admin provides the username and password to all the parents and faculty. The total control will be done by the admin.



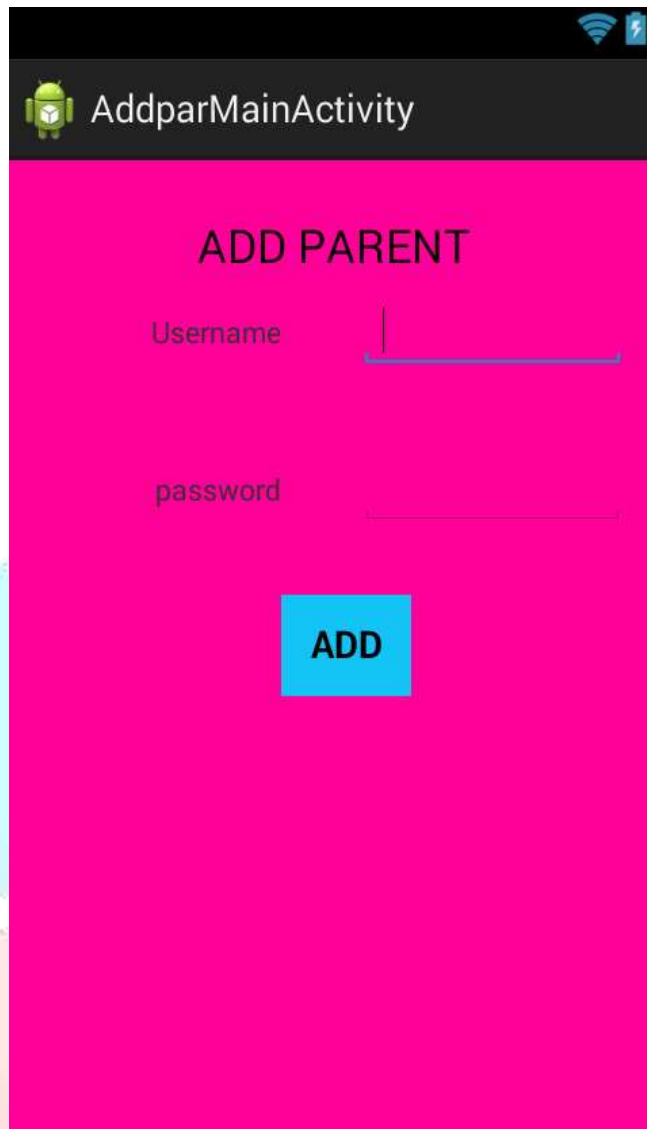
**ADMIN HOME PAGE**

**DESCRIPTION:** This is the home page for Admin. Admin contains all the rights for adding parents and faculty and removing the parents and faculty.



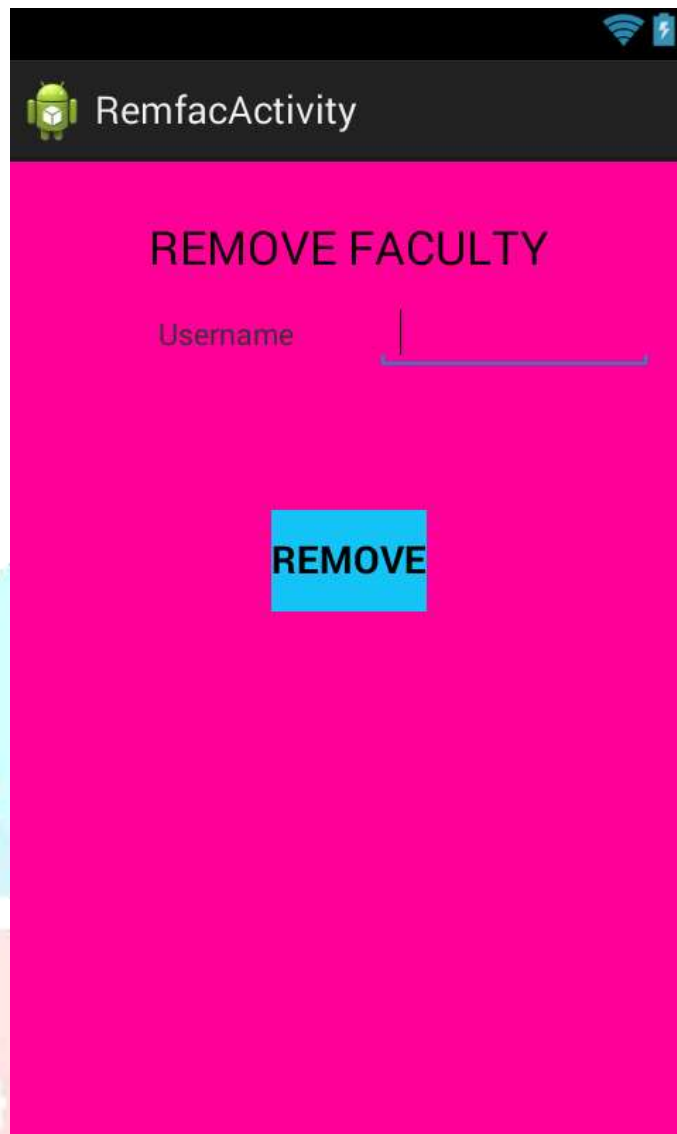
**ADD FACULTY**

**DESCRIPTION:** In this screen admin adds the faculty to the database.



**ADD PARENT**

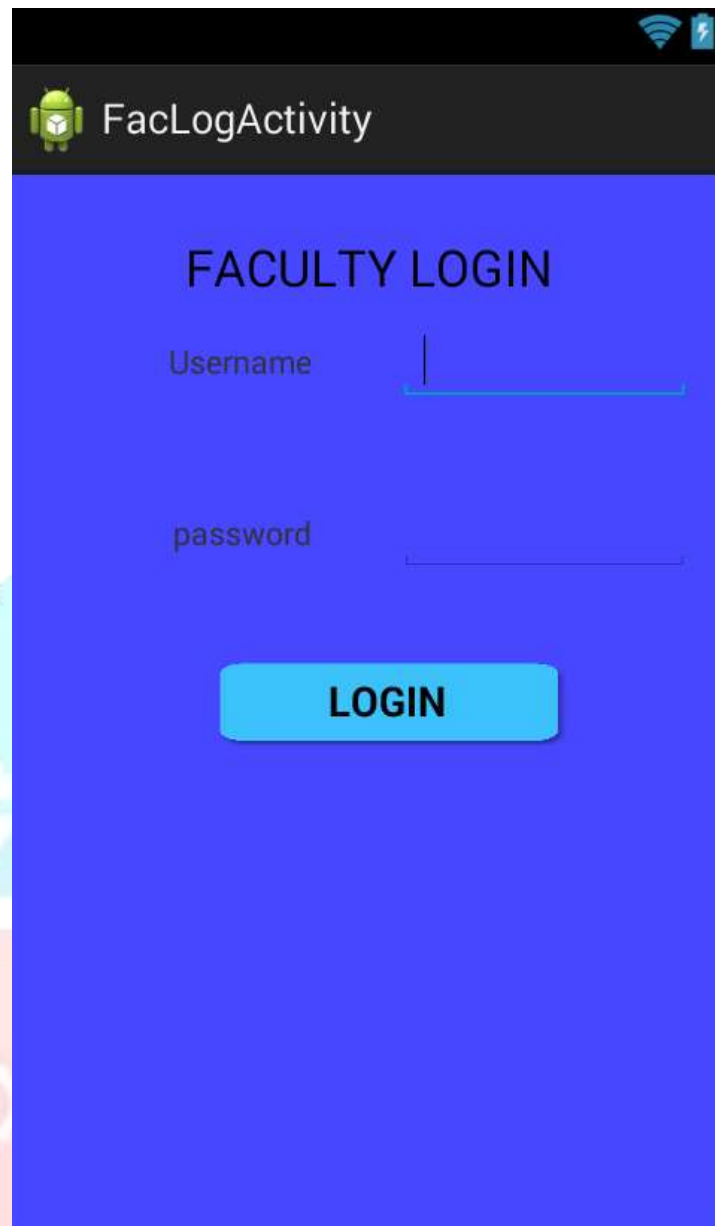
**DESCRIPTION:** In this screen admin adds the parent to the database.



**REMOVE FACULTY**

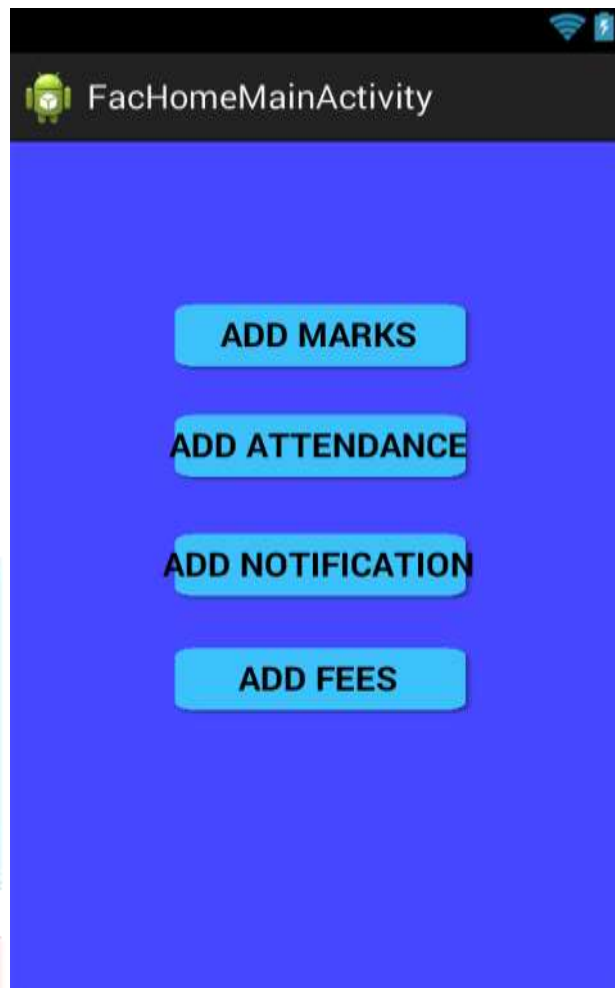
**DESCRIPTION:** In this screen admin removes the faculty from the database.





**FACULTY LOGIN**

**DESCRIPTION:** In this screen faculty login to the application. For this he needs to enter the username and password. These are given by the admin.

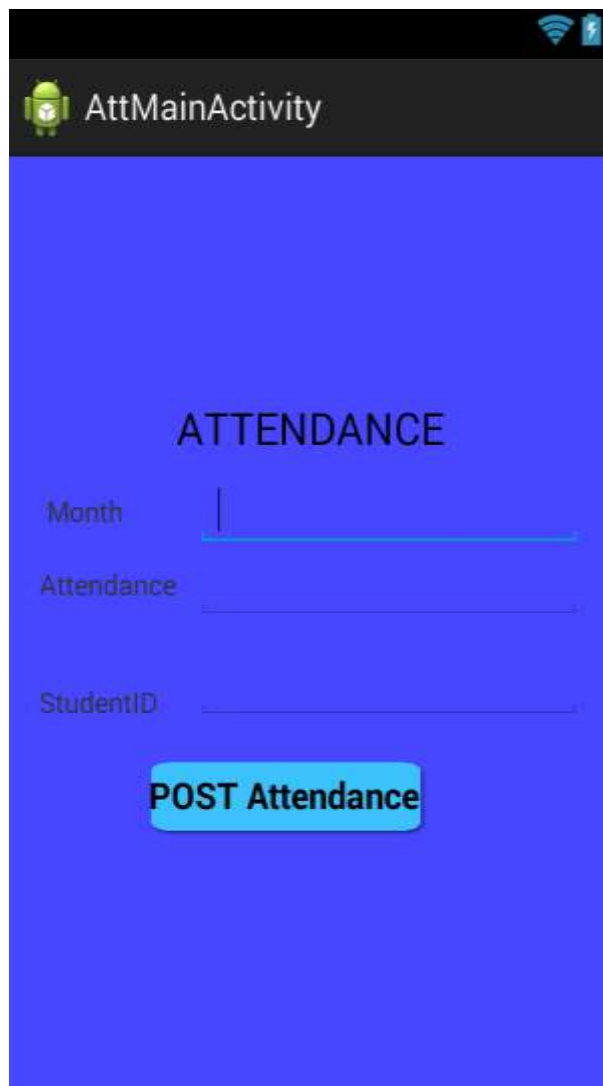


**FACULTY HOME PAGE**

**DESCRIPTION:** This is home page for the faculty. Faculty adds marks, attendance, notifications and fee dues to parents.

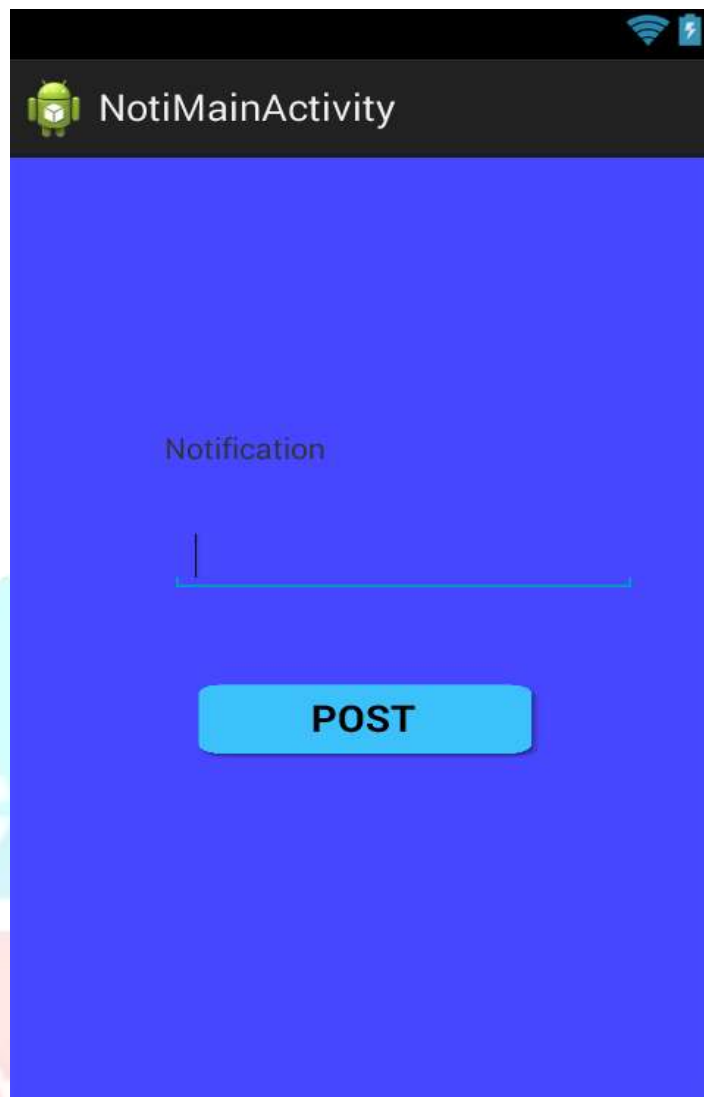


**DESCRIPTION:** Faculty adds the marks to the parents. Here faculty adds the marks according to the semester wise.



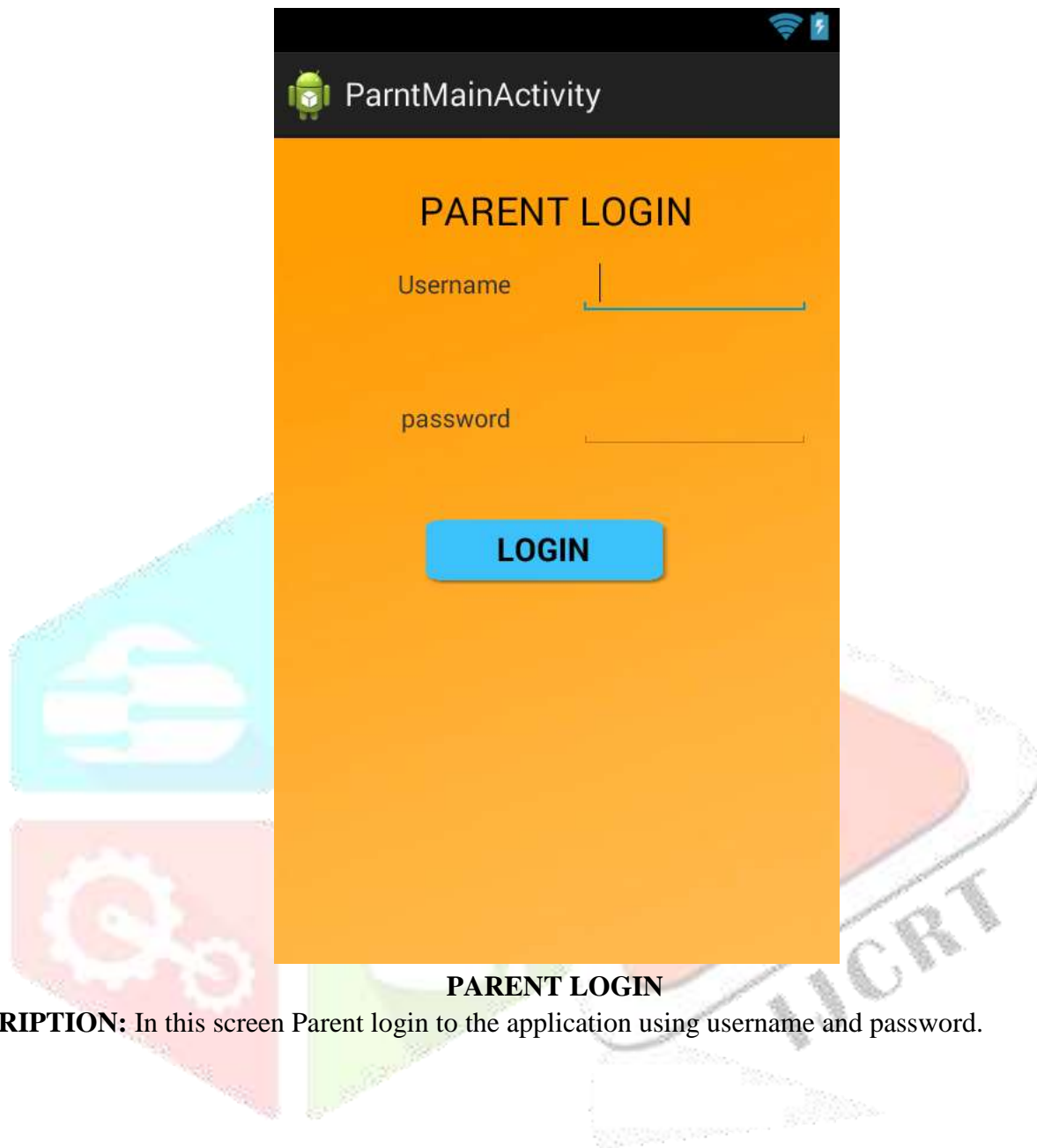
**ATTENDANCE**

**DESCRIPTION:** Faculty post the attendance of a particular student for every month.



### ADD NOTOIFICATIONS

**DESCRIPTION:** This screen enables faculty to post the notifications regarding the events that are going to take place.



**PARENT LOGIN**

**DESCRIPTION:** In this screen Parent login to the application using username and password.



**PARENT HOME PAGE**

**DESCRIPTION:** This is the Home page for Parents. In this page the parents has to select a particular details about their ward.



**FEE DUES**

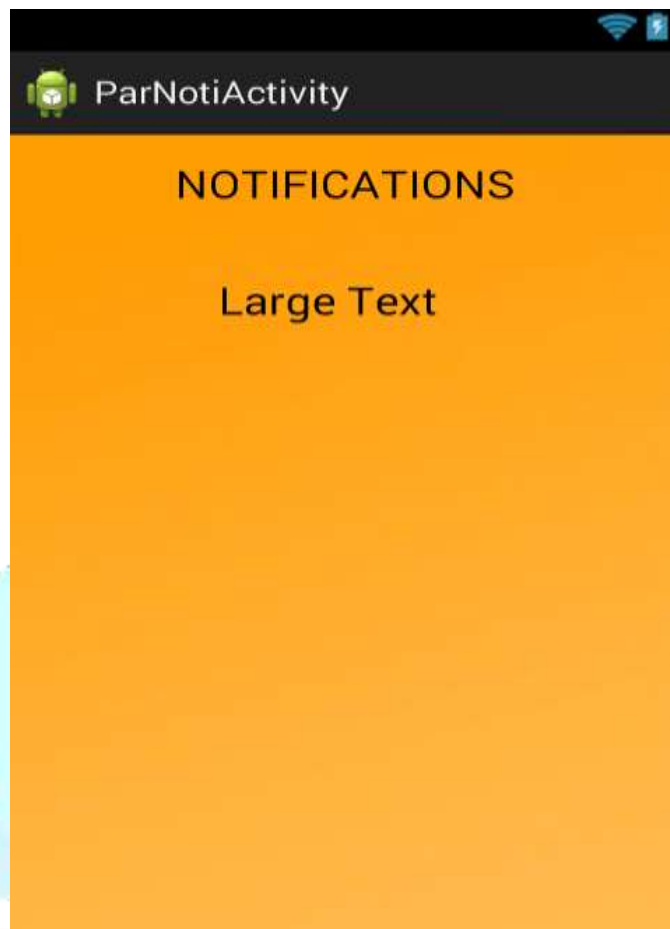
**DESCRIPTION:** This screen provides the information if there is any fee dues of the students.





**MARKS**

**DESCRIPTION:** This screen gives the information about the overall marks of the student.



### NOTIFICATIONS

**DESCRIPTION:** Here, the parent able to see all the notifications posted by the faculty.

## 5. TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### 5.1 TYPES OF TESTING

#### 5.1.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive.

Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### **5.1.2 INTEGRATION TESTING**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

### **5.1.3 FUNCTIONAL TESTING**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

### **5.1.4 SYSTEM TESTING**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### **5.1.5 WHITE BOX TESTING**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

### **5.1.6 BLACK BOX TESTING**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

## **5.2 TEST CASES**

<b>Test Case ID</b>	<b>Test Case</b>	<b>Expecting Behaviour</b>	<b>Exhibiting behaviour</b>	<b>Result</b>

1	Enter the wrong Username and Password in receipt sign in page.	Invalid username and password is displayed.	Error message will be displayed.	Pass
2	Enter the correct user id and password for receipt	Enters into the login screen.	Enters into the login screen.	Pass
3	Admin adds the faculty into the database	Added successfully	Added successfully	Pass
4	Admin adds the parent	Added successfully	Added successfully	Pass
5	Admin removes the parents	Removed	Removed	Pass
6	Faculty adds the notifications	Posted successfully	Posted successfully	Pass
7	Faculty adds the marks to the parents	Posted marks successfully	Posted marks successfully	Pass
8	Faculty adds the attendance	Posted attendance successfully	Posted attendance successfully	Pass
9	Faculty adds the fee dues	Posted fee dues successfully	Posted fee dues successfully	Pass

**Table 5.2 Testing Table**

## 6. CONCLUSION

This application is automating the existing manual system. This is a paperless work. It reduces man power required. It will provide accurate information always. Entire information can be saved and can be accessed using application at any time. Administrator, parents, faculty can get the required information without delay. As this application is only made for the general purpose it can be generalized to big scale use such as universities and even distance education can be benefited with this application as this app can provide all related information about the ward without directly contacting the staff. In future, we can also add the feature like paying the fees of the ward.

## BIBLIOGRAPHY

### REFERENCES:

- [1] AmitaDhale, MadhavMistry, TusharZore, "A Survey on Smart Connect" an Android and web based application for college management system" Department of C.E, B.D.C.E, Sewagram, Wardha, 11 November 2014.
- [2] Jianye Liu & Jiankun Yu "Research on Development of Android Application, "School of Information Yunnann University of Finance and Economics KunMing, China, IEEE research papers, 2011.
- [3] Mihal Brumbulli, Blerina Topicu, Arbora Dalaci, "SMIS: A Web-Based School Management Information System," Department of Computer Engineering, Faculty of Information Technoloies, Polytechnic University of Tirana, ALbania, 2008.
- [4] Uduak A. Umoh, M.Sc.1, Enoch O. Nwachukwu, Phd.D, Eyoh, M.Sc., "Object Oriented Database Management System: A UML Design Approach," University Of Uyo, Nigeria, Nov 2009.
- [5] S. R. Bharamagoudar, Geeta, S.G. Totad, "Web Based Student Information Management System" Assistant Professor, Dept. of Electronics & Communication Engg, Basaveshwar Egg College Bagalkot Karnataka Associate Professor, Department of IT, GMR Institute of Technology, RAJAM, Andhra Pradesh Professor, June 2013.
- [6] Bongani T. Mabunda and Johnson O. Dehinbo "Enhancing University Class Management System with Instant email feedback Alert", Oct 2012.
- [7] Wei-Meng Lee, "Beginning Android Application Development", Crosspoint Boulevard : Wiley Publishing, Inc. 10475, 2011.
- [8] Mark L. Murphy, "Android Programming Tutorials:", USA : CommonsWare, 2011

## USER MANUAL

- In this application first admin adds the faculty and parents. And after adding the faculty and parents admin provides the login and password to the parents.
- Admin has rights for removing the faculty and parents.
- Faculty needs to login into the application in order to entering. After login they will has a chance for adding the attendance, marks, fee dues, notifications.
- When faculty adds the notifications will be sends to all the users. And when faculty adds the attendance, marks, fee dues these will be sends to the particular parents.
- In order to entering into the application for parents they needs to login. After login they can able to see all the attendance, marks, fee dues and notifications.

