

SE PROJECT

CLASSROOM ATTENDANCE MANAGEMENT SYSTEM

- KESHAV GARG
2018UCP1674
5th SEMSTER
CST311

INTRODUCTION

Classroom Attendance Management System is a Mobile Application developed for daily student attendance in classroom of schools, colleges and institutes. It facilitates to access the attendance information of a particular student in a particular class. The attendance information is managed by the admin , which will be provided by the teacher for a particular class. This application will also help in generating attendance report for a particular student and evaluating attendance eligibility criteria of a student in exams.

PURPOSE

The purpose of developing classroom attendance management system is to digitalize the tradition way of taking attendance. Another purpose for developing this application is to generate the report automatically at the end of the session or in the between of the session for a particular student for regular check basis.

SCOPE

The following project has much scope both in present as well as future. In present situation the system can be accessed on android mobile devices.

Any android device which has the access to it's server on which the project has been deployed, can use the application i.e. the project will work for a particular institution. But later on the project can be modified to operate for many institutions.

DEFINITIONS, ACRONYMS AND ABBREVIATIONS

CATS	Classroom Attendance management system
DBMS	Database management system
ERD	Entity Relationship Diagram
DFD	Data Flow Diagram
STD	State Transition Diagram
DOJ	Date of Joining

REFERENCES

1. Tutorials Point
2. YouTube
3. Stack Overflow
4. Software Engineering - Pressman

PROJECT OBJECTIVES

My project is classroom attendance management system and the objectives are:

1. Manual work for information retrieval on attendance becomes less as the work.
2. Easy access for students because they can view their attendance and make up for the shortage of attendance accordingly.
3. It is also time saving as manual work is less.
4. There is less chance of error.
5. It eliminates duplicate data entry in time and attendance entries.
6. Weekly or Monthly attendance reports can be generated for a particular student.
7. Easy maintenance of students and faculty data by admin.
8. Data redundancy can be decreased as data is now stored at one place.
9. Data is now much more secure.
10. Easy access for teachers as they can now view their student's attendance report easily.

DEFINATION OF THE PROBLEM

PROBLEM DEFINATION

This system developed will reduce the manual work and avoid redundant data. By maintaining the attendance manually, then efficient reports cannot be generated. The system can generate efficient weekly, consolidate report based on the attendance. As the attendances are maintained in registers it has been a tough task for admin and staff to maintain for long time. Instead the software can keep long and retrieve the information when needed.

MODULE DESCRIPTION

The system should be designed in such a way that only authorized people should be allowed to access some particular modules. The records should be modified by only administrators and no one else. The user should always be in control of the application and not the vice-versa. The user interface should be consistent so that the user can handle the application with ease and speed. The application should be visually, conceptually clear.

● ADMIN MODULE

Admin has access to all information related teachers, students, courses and attendance reports. Admin can **insert, update, delete** any details regarding courses,

students and teachers. Admin enrolls students in courses and decide which course will be taught by which teacher. Admin also manages attendance reports of all the students currently in institution.

● **TEACHER MODULE**

Teacher has access to all information of students enrolled under course taught by him. Teacher has the responsibility of taking digitized attendance of all students under him. Teacher can also update his profile.

Both the users should have a basic knowledge of using mobile application.

FEASIBILITY STUDY

WHAT IS FEASIBILITY STUDY?

A feasibility study includes an estimate of the level of expertise required for a project and who can provide it, quantitative and qualitative assessments of other essential resources and whether project is viable or not.

- **ECONOMIC FEASIBILITY?**

The system being developed is economic with respect to School or College's point of view. **It is cost effective in the sense that has eliminated the paper work completely.**

- **BEHAVIOURAL FEASIBILITY?**

The system working is quite easy to use and learn due to its simple but **attractive interface**. User requires no special training for operating the system.

- **TECHNICAL FEASIBILITY?**

The technical requirement for the system is economic and it does not use any other additional Hardware and software. Technical evaluation must also assess whether the existing systems can be upgraded to use the new technology and whether the organization has the expertise to use it.

REQUIREMENT ANALYSIS AND SPECIFICATION

Application developed follows the steps of SDLC under which the SRS model developed for the system is listed below:

The SRS model contains:

1. Functional Requirements
2. Non Functional Requirements

● **FUNCTIONAL REQUIREMENTS**

The functional requirement part of the system discuss the functional behavior that should be possessed by the system. Each requirement maps to a high level function that transforms the given set of input data into output data. They are:-

1. LOGIN

Admin and Faculty will login in into the system with username and password, if username and password are correct admin/faculty will be prompt to proceed option otherwise error will be shown

INPUT: username and password

OUTPUT: student detail information

2. UPDATE COURSE DETAILS

Admin can insert, update, delete course details.

INPUT: Course name

OUTPUT: Operation(insert/update/delete) successful

3. UPDATE STUDENT INFO AND COURSE ENROLLMENT

Admin can insert, update, delete student's info and enroll students in particular courses.

INPUT: name, Roll No., DOB, Course

OUTPUT: Operation(insert/update/delete) successful

4. UPDATE TEACHERS INFO AND ASSIGN COURSES

Admin can insert, update, delete teacher's info and assign a course to each teacher.

INPUT: Name, Address, Email, image, course, DOJ

OUTPUT: Operation(insert/update/delete) successful

5. MARK ATTENDANCE

Faculty can take attendance on a particular day over a student name by marking him present or absent.

INPUT: Absent/Present

OUTPUT: Confirmation message

6. WEEKLY/MONTHLY ATTENDANCE REPORT/CHART

Admin/Faculty can create student's attendance report/chart between any 2 dates

INPUT: starting date, ending date

OUTPUT: attendance report/chart

• **NON-FUNCTIONAL REQUIREMENTS**

❖ **EXTERNAL INTERFACE REQUIREMENTS**

1. USER INTERFACES

The user interface for system shall be compatible with Android 4.4 (KITKAT) version and above.

2. SOFTWARE INTERFACES (Tool- Android Studio):

Front-end: XML

Back-end: Java

DATABASE: MySQL, Firebase

SERVER: SQLite Server

3. HARDWARE INTERFACES

Monitor resolution: 1024X768

Processor: Intel i5 above

RAM: 4GB

Disk Space: 50GB

4. COMMUNICATION INTERFACES

The mobile device may have an active network connectivity, as first it will store the attendance data in the mobile device itself in SQL, then while syncing it will transfer to the main database in Firebase.

❖ **PERFORMANCE REQUIREMENTS**

1. Data in the database should be updated within 2 seconds.
2. Query results must return results within 5 seconds.
3. Load time of UI should not exceed 2 seconds.
4. Login validation should be done within 3 seconds.

❖ **SECURITY REQUIREMENTS**

1. All external communications between the data's server and client must be encrypted.
2. Login Captcha must be used.
3. All data must be stored and protected.
4. Personal details should not be leaked to other users in any way. System should not communicate to each other for private data.

● **SAFETY REQUIREMENTS**

1. Database should be committed after every query to keep it up to date.
2. Under failure, system should be able to come back to working condition in less than an hour.

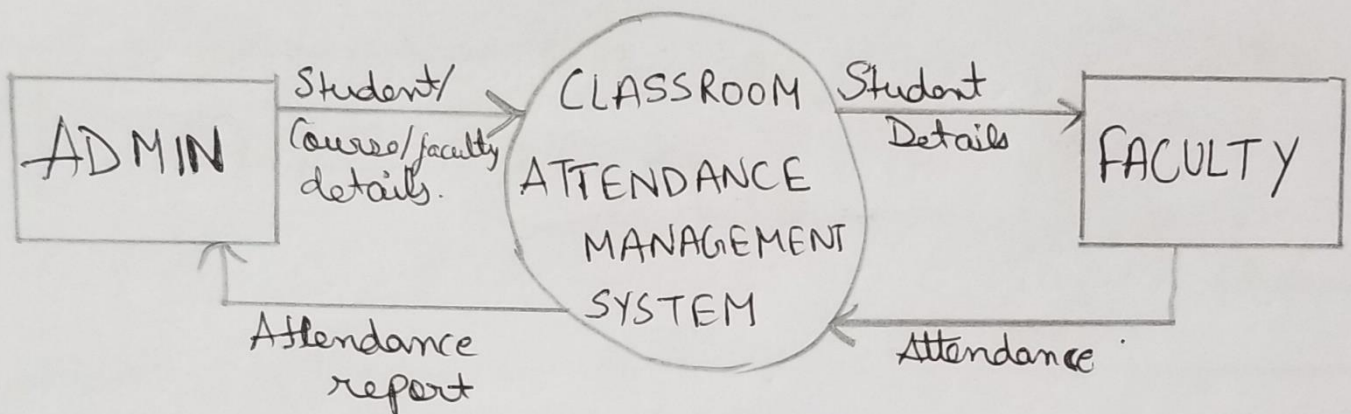
SYSTEM ATTRIBUTES

1. **Correctness:** This system should satisfy the normal Attendance Management operations to fulfil the end user objectives.
2. **Efficiency:** Enough resources to be implemented to achieve particular task efficiently.
3. **Flexibility:** System should be flexible to provide space to add new features and maintain the old ones. The new feature must be integrated with minimum effort.
4. **Integrity:** System should secure user information and not allow unauthorized access or updating to anyone.
5. **Portability:** The system should run on any android device.
6. **Usability:** The system should have enough information to allow user to know the use of the displayed screen. Any major changes made should be notified back to the user to allow them to know about the changes done by them.
7. **Testability:** The system should be tested according to the requirements and confirm proper working of the software.
8. **Maintainability:** Any errors found after the release of the system, the software must be maintainable to give required performance.

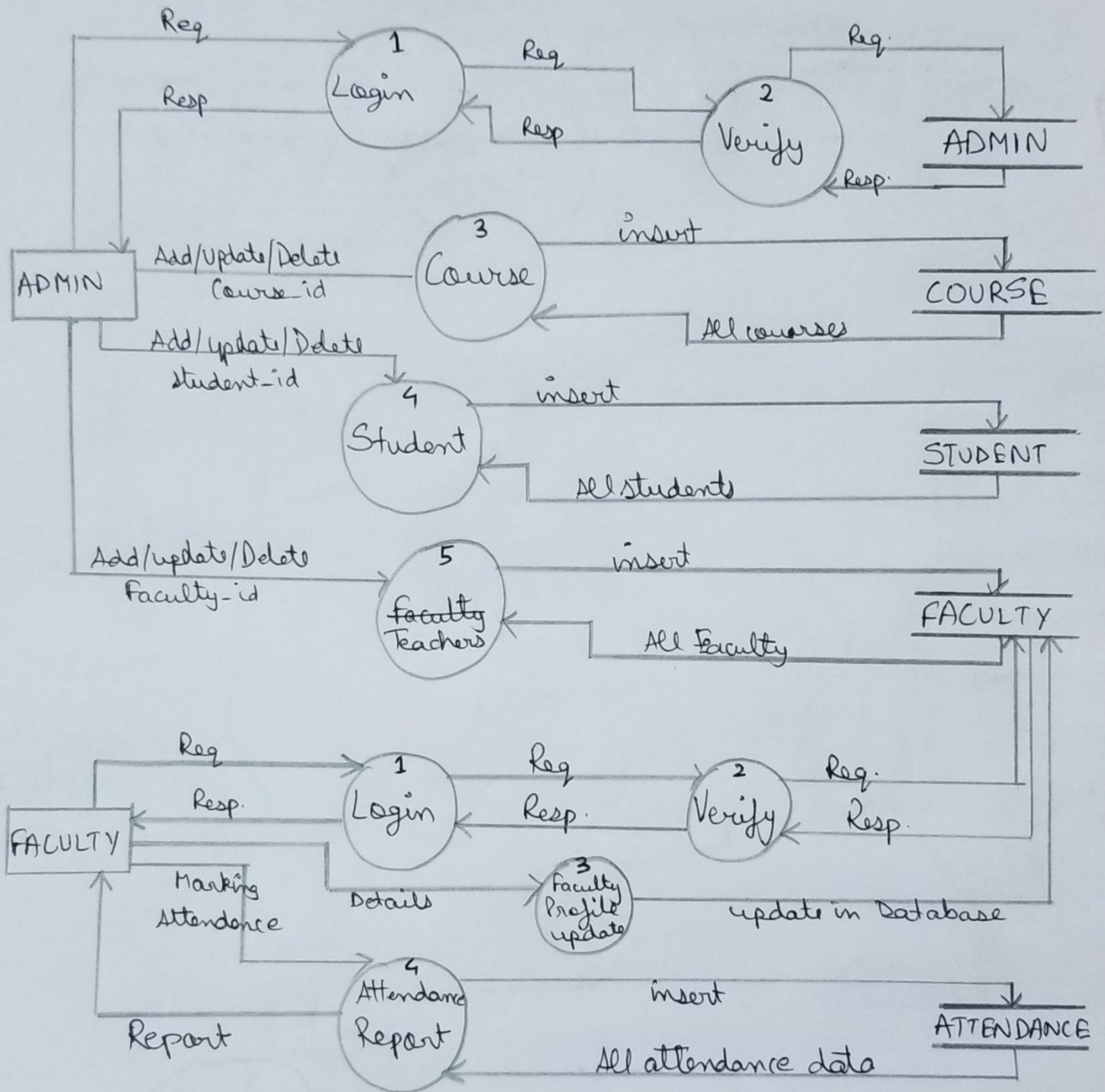
DESIGN PHASE

The design phase emphasizes on the transformation of customer requirements as defined in the SRS document, into a form that is suitable for coding.

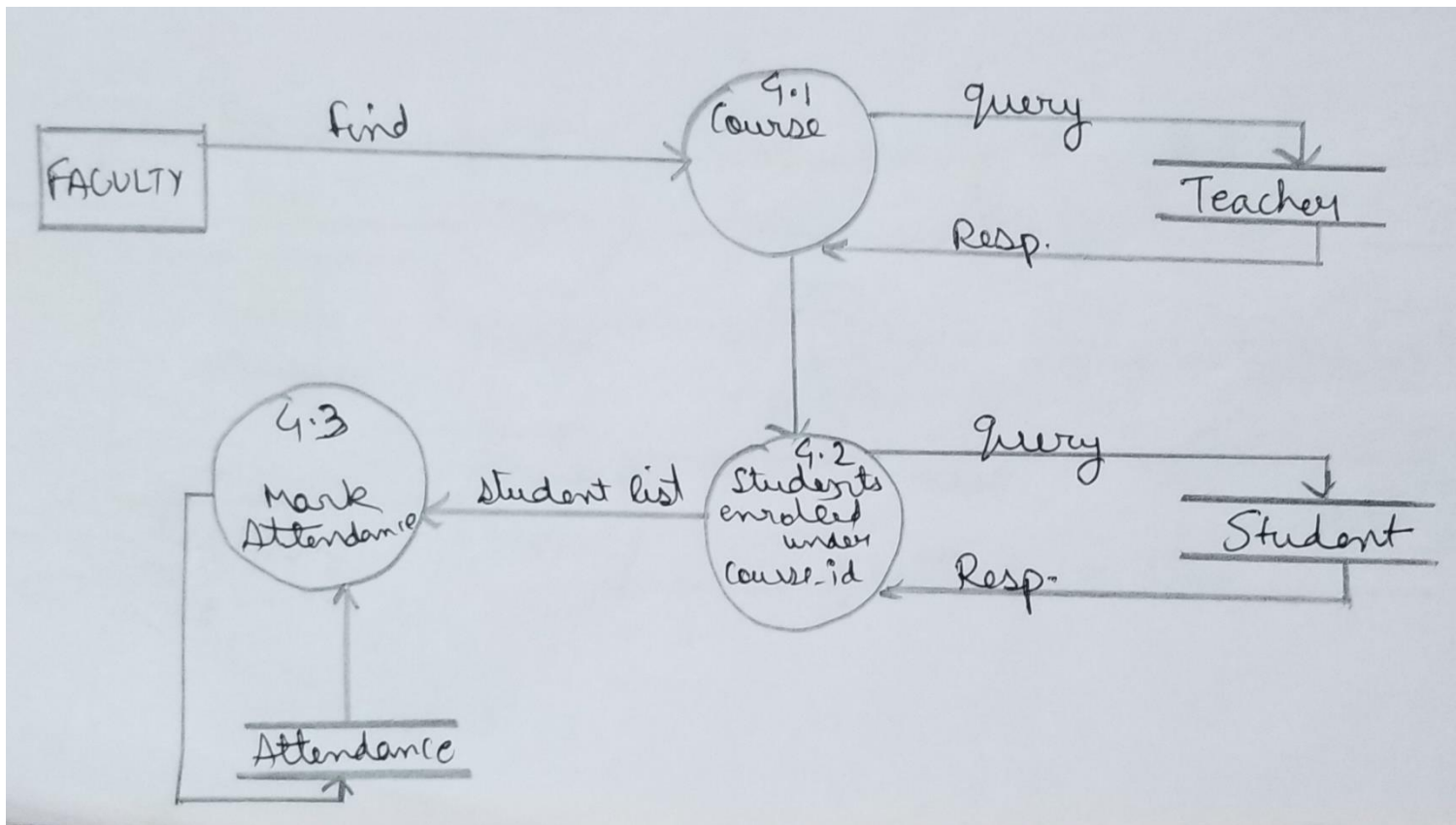
DFD LEVEL 0



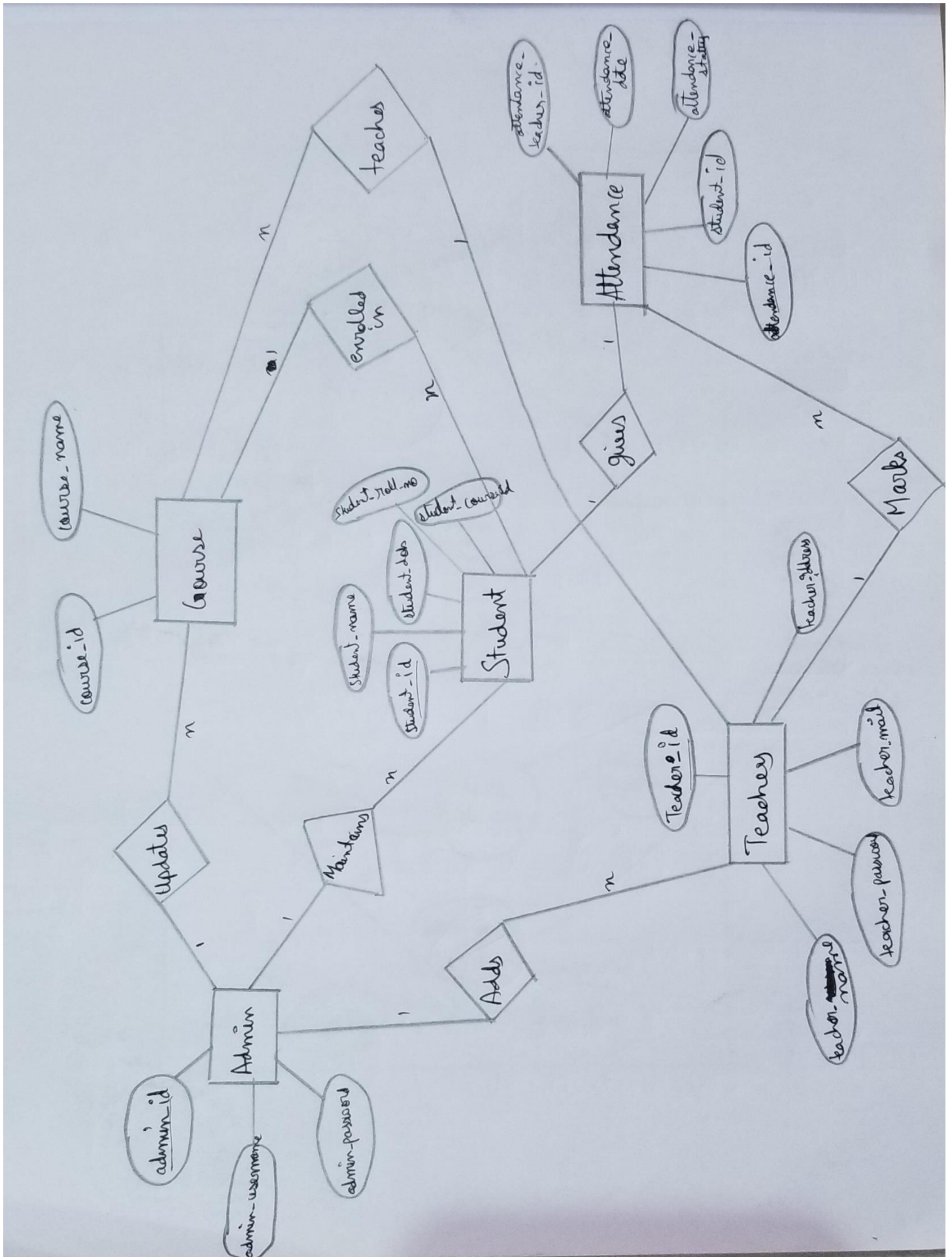
DFD LEVEL 1



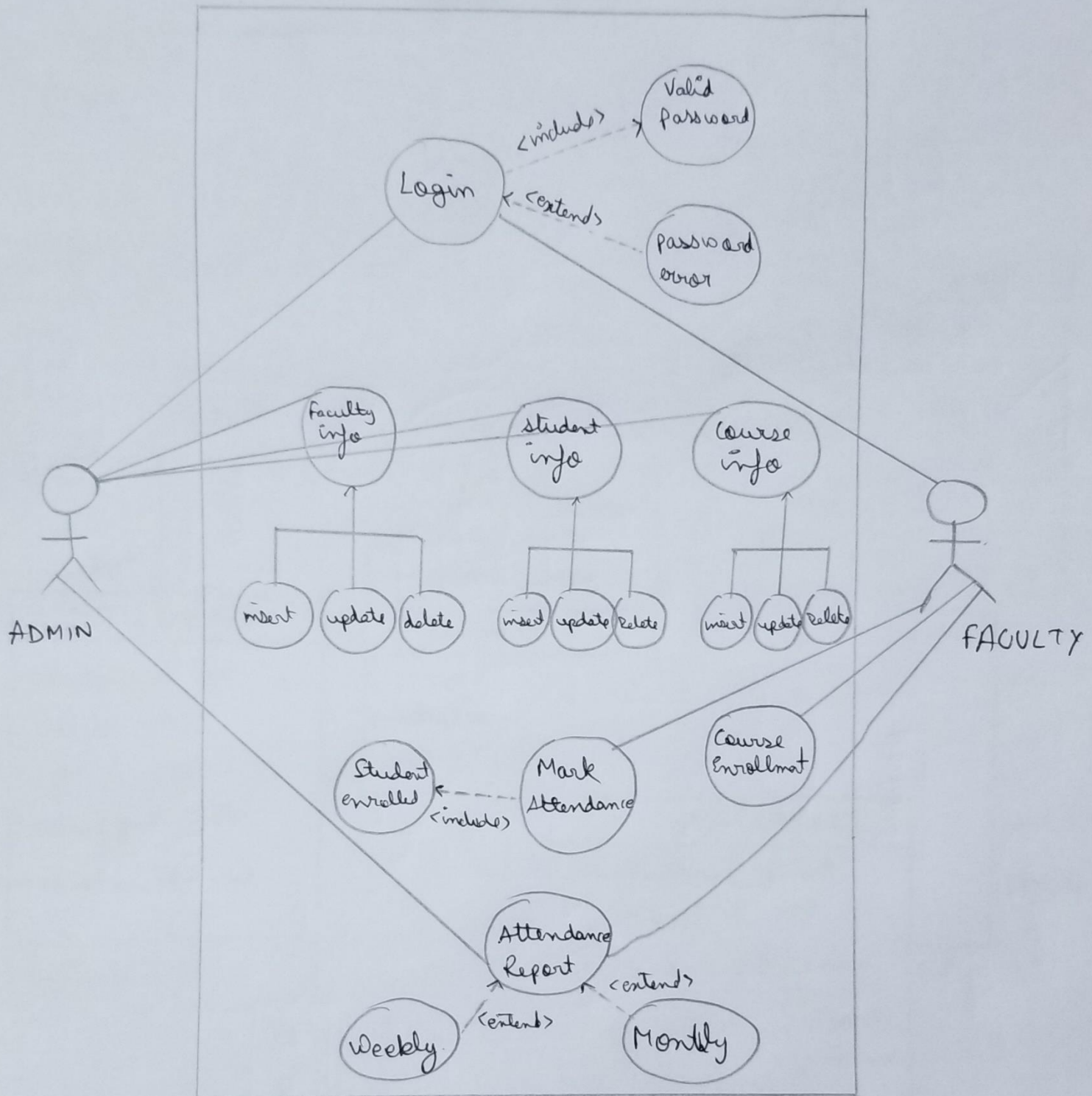
DFD LEVEL 2



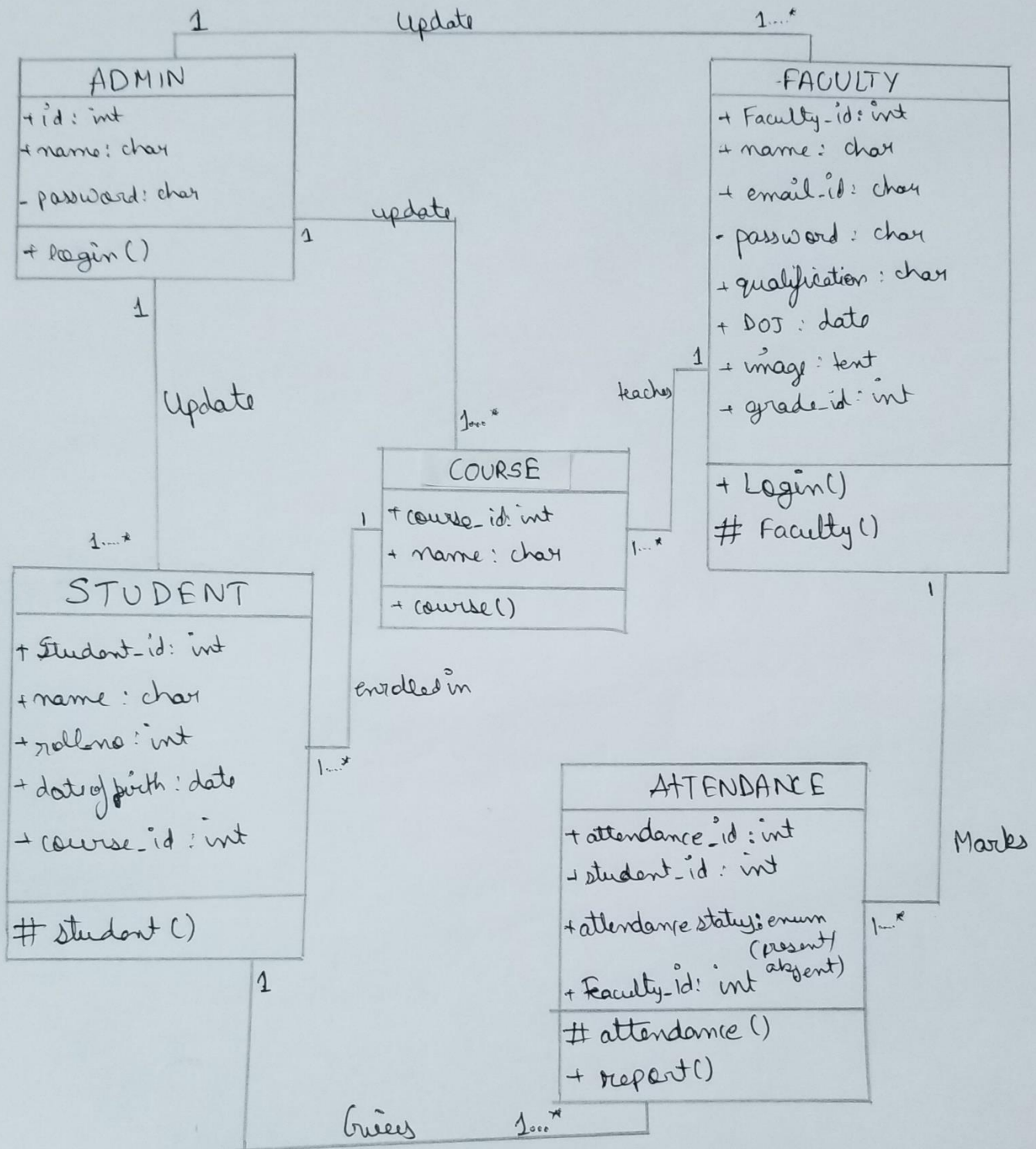
ER DIAGRAM



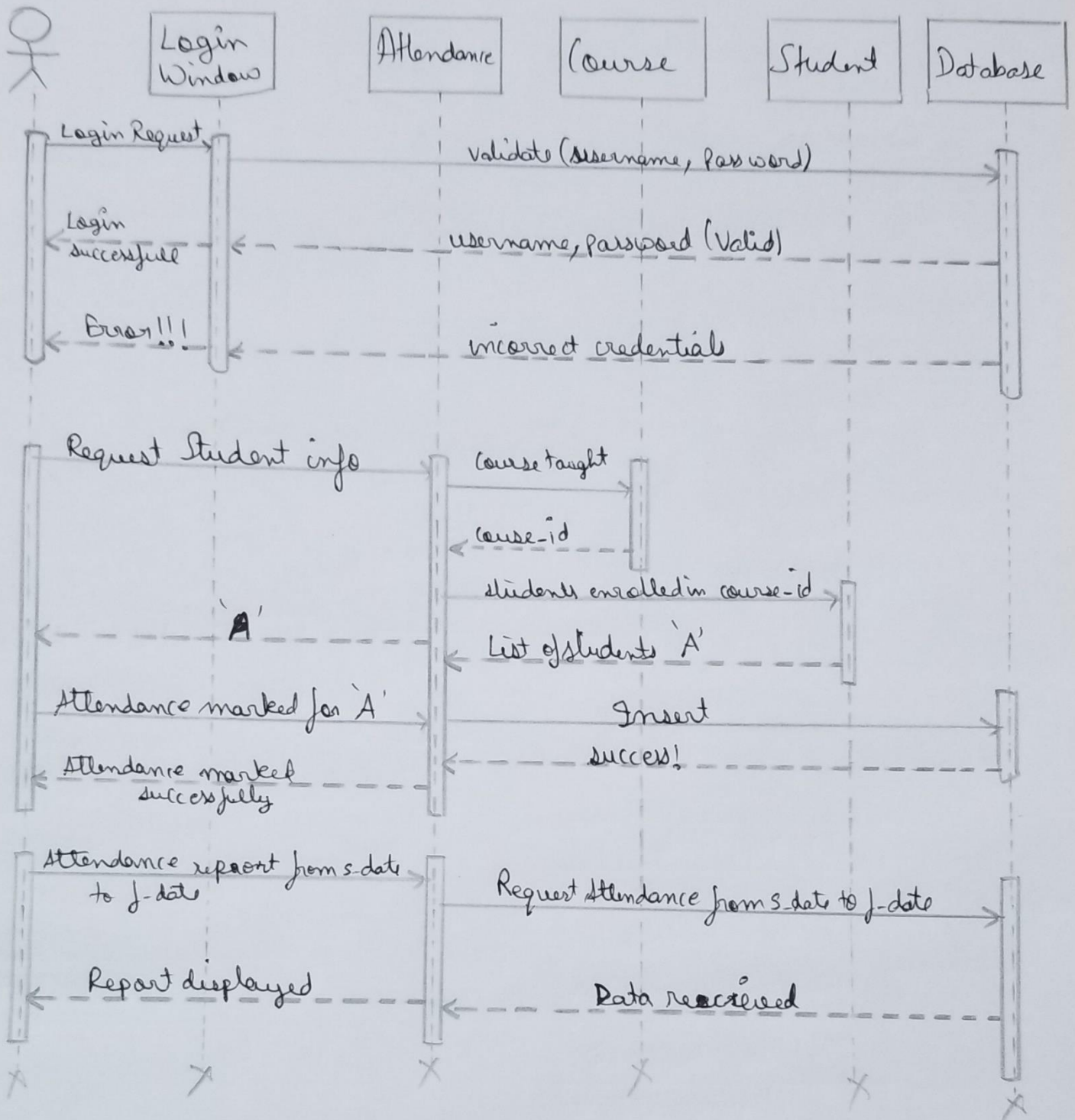
USE CASE DIAGRAM



CLASS DIAGRAM



SEQUENCE DIAGRAM



ALGORITHMS

1. LOGIN

```
Login()
{
    username    = get username(text)
    password    = get password(text) (encrypted)
    if(username in database)
    {
        If(password matches with password in database)
        {
            Display: Login Successful
            Goto: Main Page
        }
    }
    else
    {
        Display: Incorrect password
    }
}
else
{
    Display: Invalid username
}
}
```

2. INSERT/UPDATE/DELETE COURSE DETAILS

```
Course()
{
    If(Add or Update)
    {
        Course_name = get (text);
        If(Course_name is empty)
        {
            Message: Course should have a name
            Return;
        }
        else
        {
            Database_Update();
            Display: Course updated successfully.
        }
    }
    else if(delete)
    {
        Course_id = get Cid;
        Delete    from    database    whose    course_id    is
        Course_id (SQL query to delete that course)
        Display: Course deleted successfully
        Update page;
    }
}
```

3. INSERT, UPDATE, DELETE STUDENT INFORMATION AND ENROLL STUDENTS IN COURSES

```
Student()
{
    If(Add/Update)
    {
        name = get text;
        Roll_no = get number;
        DOB = get date;
        Course = get option selected;
        If( (name, Roll_no, DOB, course is empty)
            {
                Display: No field should be empty.
                Return;
            }
        else if(Roll_no in database)
        {
            Display: Every student should have unique Roll_no.
            Return;
        }
        else
        {
            Database_update()
            Display: Student info updated successfully.
        }
    }
}
```

```

else if(delete)
{
    Roll_no. = get s_id;
    Delete from database whose s_id is Roll_no.
    Display: Student info deleted successfully.
    Update page;
}
}

```

4. INSERT, UPDATE, DELETE TEACHER' S INFORMATION AND ASSIGN COURSE TO A TEACHER

```

Teacher()
{
    If(Add /Update)
    {
        name = get text;
        address = get text;
        email = get address;
        pwd = get password;
        Course = get option selected;
        DOJ = get date;
        Image = get image;
        If((name, address, email, pwd, Course, DOJ, Image) is empty)
        {
            Display: No filed should be empty.
            Return;
        }
    }
}

```



```

        else
        {
            Databse_update();
            Dsiplay: Teacher info updated successfully.
        }
    }
    else if(delete)
    {
        Teacher_id = get Tid;
        Delete from database whose Tid is Teacher_id.
        Display: Teacher info deleted successfully.
        Update page;
    }
}

```

5. MARK ATTENDANCE

```

Attendance()
{
    date = getDate();
    student_id = getsid();
    Status = get selected option;

    if(status is empty)
    {
        Display: error;
        Return;
    }
}

```

```

    }
    else
    {
        Insert in database student_id's attendance status on date.
        Display: Attendance marked successfully.
    }
}

```

6. VIEW ATTENDANCE REPORT/CHART

```

Report()
{
    Startdate = get date;
    End_date = get date;
    If(Start_date or End_date is empty)
    {
        Display: Error; Return;
    }
    else
    {
        If(Option is Report)
        {
            Show pdf of Attendance report of selected student from
            Start_date to End_date;
        }
    }
}

```

```
    else
    {
        Show chart representation of Attendance report of
        selected student from Startdate to End_date;
    }
}
}
```

TESTING

Once source code has been generated, software must be tested to uncover (and correct) as many errors as possible before delivery to customer

BLACK BOX TESTING

Software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

1. Equivalence class partitioning

Software testing technique that divides the input data of a software unit into partitions of equivalent data

from which test cases can be derived. In principle, test cases are designed to cover each partition at least once.

- Username : $0 < \text{length} < 45$, **no space allowed, Char allowed**
- Password : $0 < \text{length} < 12$, **1 special character is must**
- Student_roll_no : $0 < \text{length} < 11$
- Student_name : $0 < \text{length} < 100$, **Char is allowed**
- Teacher_name : $0 < \text{length} < 100$, **Char is allowed**
- Course_name : $0 < \text{length} < 15$, **Char is allowed**
- DOB : maximum date = present date
- Mobile : length=10, **digits allowed**

Any other values input by user is considered invalid.

2. Decision table testing

A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

Username	Invalid	Invalid	Valid	Valid
Password	Invalid	Valid	Invalid	Valid
Output	Error	Error	Error	Successful login

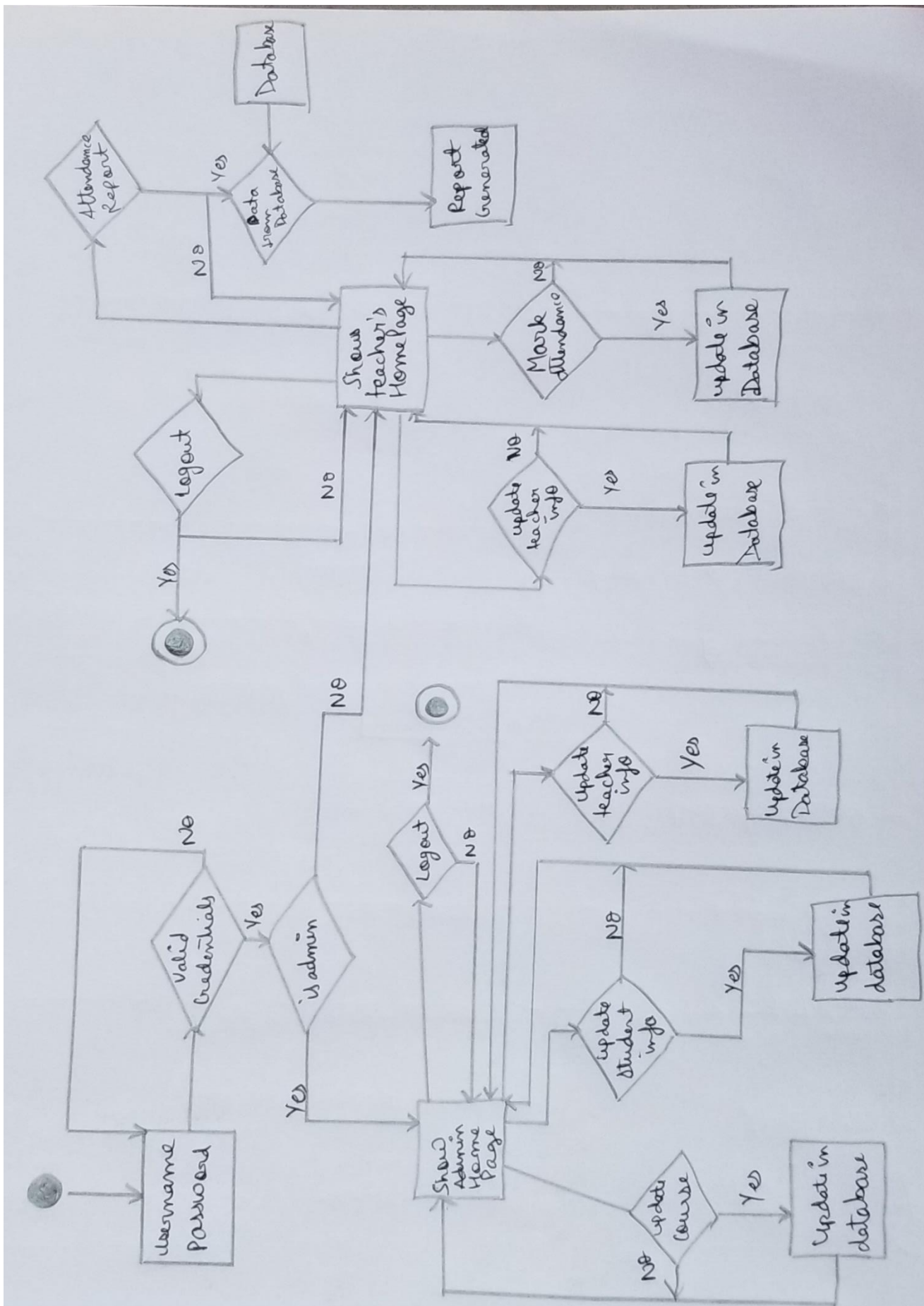
WHITE BOX TESTING

Method in which the internal structure /design /implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs.

Statement coverage testing

1. Test case 1 : invalid user, returnBack
2. Test case 2 : valid user, is admin, true, update course, true
3. Test case 3 : valid user , is admin, true, update student info, true
4. Test case 4 : valid user , is admin , true , update teacher info, true
5. Test case 5 : valid user , is admin , true, logout, true
6. Test case 6 : valid user , is admin, false, update profile, true
7. Test case 7 : valid user, is admin, false, mark attendance, true
8. Test case 8 : valid user, is admin, false, attendance report, true
9. Test case 10 : valid user, is admin, false, logout, true

See the next image of class diagram for reference.



BRANCH COVERAGE TESTING

1. Test case 1 : invalid user, loopback

Routes visited : 1, 2

2. Test case 2 : valid user, is admin, true, update course, true

Routes visited: 1,3,4,7,9,6

3. Test case 3 : valid user, is admin, true, update student info, true

Routes visited: 1,3,4,8,11,10

4. Test case 4 : valid user, is admin, true, update teacher info, true

Routes visited: 1,3,4,12,13,14

5. Test case 5 : valid user, is admin, true, logout, true

Routes visited: 1,3,4,16,17

6. Test case 6 : valid user, is admin, false, update profile, true

Routes visited: 1,3,5,18,19,20

7. Test case 7 : valid user, is admin, false, mark attendance, true

Routes visited: 1,3,5,21,22,23

8. Test case 8 : valid user, is admin, false, attendance report, true

Routes visited: 1,3,5,26,27,29

9. Test case 10 : valid user, is admin, false, log out, true

Routes visited: 1,3,5,30,32

BASIC PATH TESTING

Nodes = 24

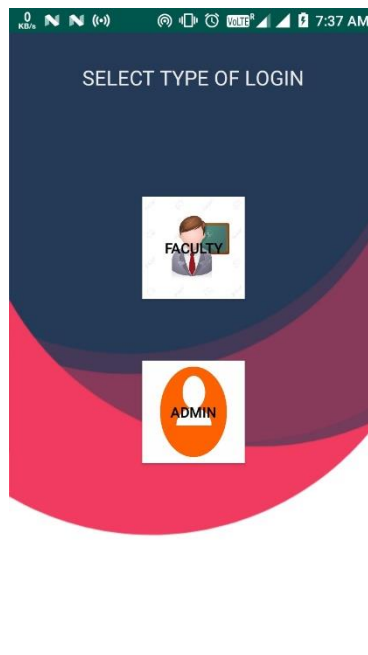
Edges = 32

Cyclomatic complexity = $32 - 24 + 2 = 10$

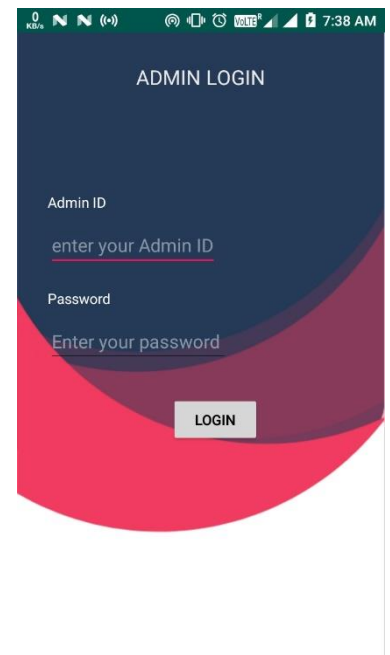
Some Screenshots of the Application



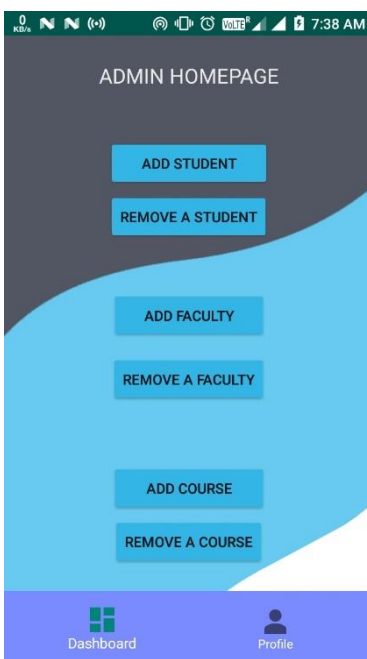
Welcome Screen



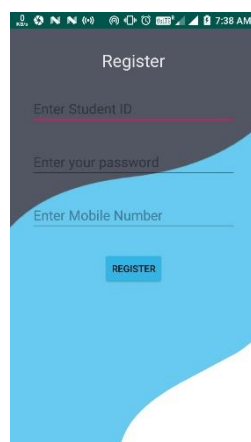
Login Screen



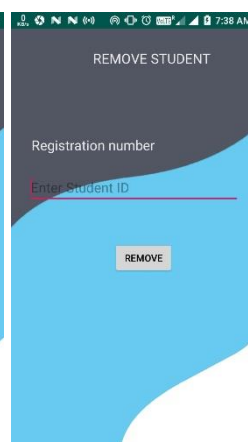
Admin Login



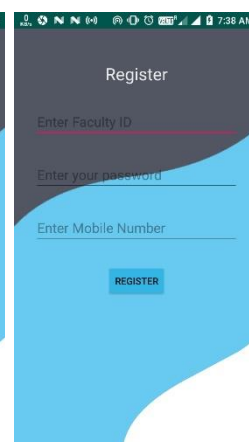
Admin Homepage



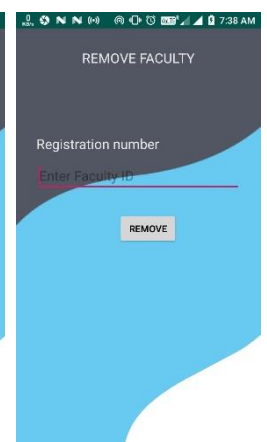
Register Student



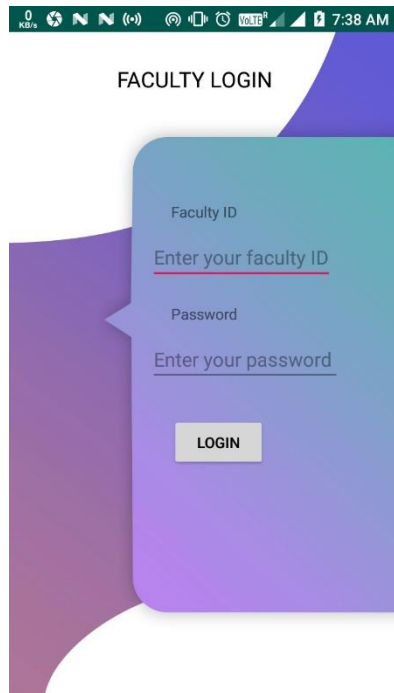
Remove Student



Register Faculty



Remove Faculty



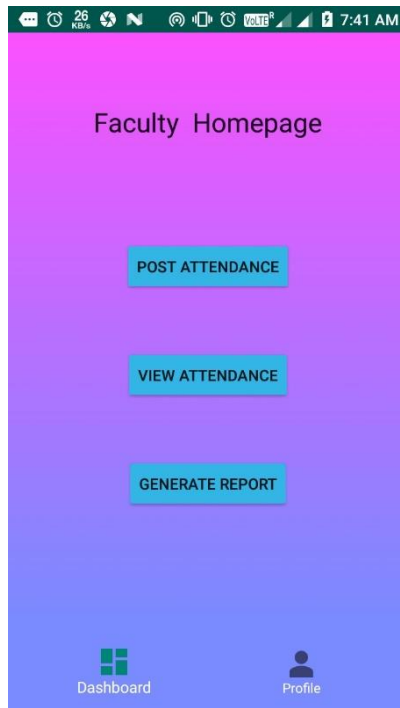
FACULTY LOGIN

Faculty ID
Enter your faculty ID

Password
Enter your password

LOGIN

Faculty Login



Faculty Homepage

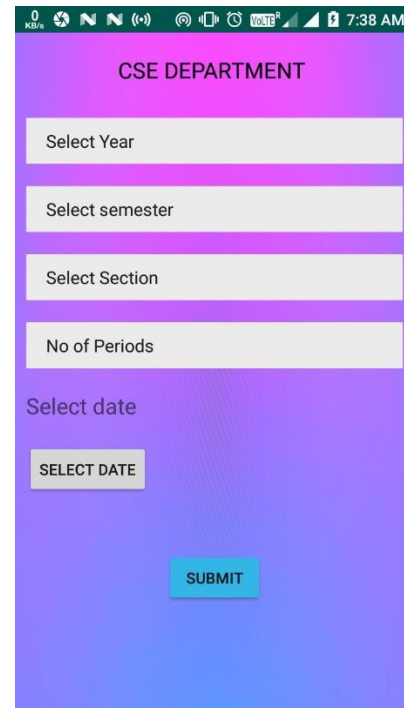
POST ATTENDANCE

VIEW ATTENDANCE

GENERATE REPORT

Dashboard Profile

Faculty Homepage



CSE DEPARTMENT

Select Year

Select semester

Select Section

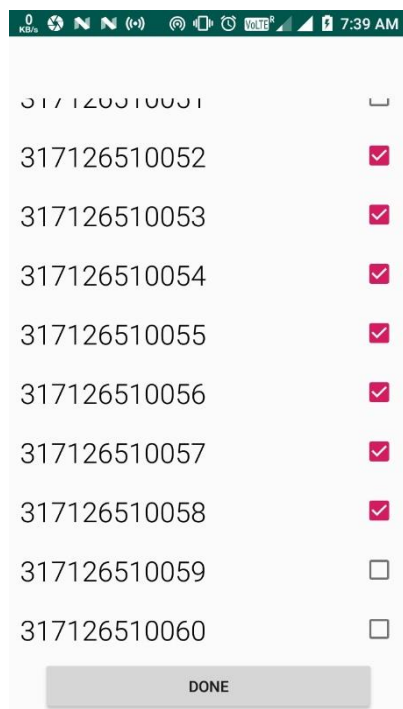
No of Periods

Select date

SELECT DATE

SUBMIT

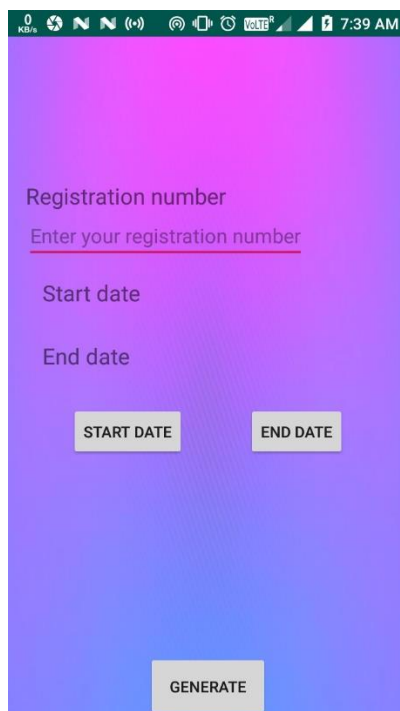
Post Attendance



317126510051	<input type="checkbox"/>
317126510052	<input checked="" type="checkbox"/>
317126510053	<input checked="" type="checkbox"/>
317126510054	<input checked="" type="checkbox"/>
317126510055	<input checked="" type="checkbox"/>
317126510056	<input checked="" type="checkbox"/>
317126510057	<input checked="" type="checkbox"/>
317126510058	<input checked="" type="checkbox"/>
317126510059	<input type="checkbox"/>
317126510060	<input type="checkbox"/>

DONE

Attendance List



Registration number
Enter your registration number

Start date

End date

START DATE END DATE

GENERATE

Generate Report

THANK YOU