Online IET

Software Design Specification

(Session 2017 - 2018)

Guided By:	Submitted By:
Dr. Vaibhav Jain	Anuj Singh Koli (15C7111)
Assistant Professor	Devyani Lambhate (15C7117)
	Taikhoom Rajodwala (15C7157)

Department of Computer Engineering

Institute of Engineering & Technology

Devi Ahilya Vishwavidyalaya, Indore (M.P.)

(www.iet.dauniv.ac.in)

October 2018

TABLE OF CONTENTS

Chapter-1 Introduction

1.1 Overview and issues involved

The attendance system we are going to develop will allow teachers to easily maintain student's attendance records. It will be especially helpful in preparation of detention list while simultaneously allowing student to view their attendance in real time. Currently teachers are taking attendance on diaries provided by the college or on paper which have higher chances of being lost and are difficult to maintain also our system have search functionality which will allow teachers to view details of any student. Most of the other tasks like marks record keeping and feedback are also maintained manually with the help of pen and paper. Feedback was also initially taken on paper which was very tedious for faculties to sort out the information and evaluate themselves. Feedback System was developed by our seniors which solved the problem to large extent and it also inspired us to develop a similar system for attendance. Marks system is under development, in parallel to attendance system by other team. Since all these systems required faculty login, we have decided to develop a portal which will provide access to all systems under single faculty authentication. In addition, our system have search functionality which will allow faculties to view details of any student, maintain their scholar information and update their profile, displayed on IET website. Android App will have additional broadcast feature which will enable faculties to broadcast real time messages to students. Since all above systems are developed for maintaining student information, a student portal/app will enable students to view these information and be always updated.

1.2 Problem definition

Our college currently depends on manual paper based method for various tasks like marks management, attendance management, student feedback which require a lot of time and resources just to manage paperwork which can be eliminated with help of software based solution. A point interface that allow faculties to perform all above task on their mobile and pcs from a single login.

Currently attendance management in college is done manually with the help of pen and paper. Which has the following drawbacks

1. Prone to human error
2. Difficult to manage
3. Higher chances Of getting lost
4. Lacking transparency between faculty and students
5. Not environment friendly

1.3 Proposed solution

To tackle above problems a digitized solution is needed which should fulfill following objectives i.e. allow faculty to edit attendance in case of error, attendance record should be stored online and also readily available to view, backup for future reference, students should be able to view their attendance and finally should avoid unnecessary paperwork. Apart from attendance system, time-table management is another domain which required to be looked upon. Currently time-table preparation is also done manually on a trial and error basis which often create sub optimal TT which doesn't satisfy all the real life constraints and often imperfect from both student and faculty point of view.

The complete system will be based on client server architecture in which there are 3 kind of end user i.e. faculty, student and system administrator.

- Single Point Login: - It will allow faculty to login into marks, feedback and attendance system from single interface. Here we propose to send user credential to login script of corresponding system which create user session and logs in the user.
- Attendance System: - Given system will allow faculty to save attendance with the android app or web app which is centrally stored in a database on a LAMP server hosted in college intranet. Attendance of each student will be present in format of days which lecture is taken and if a given student was present or not which will allow us to create a comprehensive report of student attendance in each subject which students can view through student portal.
- Faculty Broadcast: - All information from faculties flow through Class Representative (CR) to students which causes delays and confusion. Our app will provide direct broadcast from faculties to students with the help of Google cloud messaging.
- Time table Scheduling: - Time table generation requires one to consider various constraints and it is often difficult to find optimal Time table. This problem can be solved with the application of genetic algorithms and heuristic based approach which allow to find most optimal time table based on various hard and soft constraints
- Student Details: - The android app will allow students to maintain profiles that will be stored in database which in turn allow faculty to search students and view their details for quick reference.
- Admin Panel: - a common admin panel will allow system administrator to maintain faculty data. All the system are simultaneously updated as soon as addition, deletion or updation of faculty take place. This common database will be helpful in student detail feature and for timetable generation.

Chapter-2 Literature Survey

2.1 Methodology

Initially Requirement analysis will be done, it will include the identification and gathering of all the software and hardware requirement, which will include a pc and an android phone with 4.0 version or higher with developers option activated. It will also require interfaces to work on like android Studio and NetBeans for PHP based work.

Prepare use case and flowcharts. Break each element into deliverable and lay down a strategy to go ahead with. Work on designing part (UI/UX) and prepare a design that delivers the best user experience. Test it on different devices. Ensure smooth navigation on the Mobile App.

While working on designing, Start implementing back-end Processes. Deploy each functionality one by one and keep testing it. Firstly the basic visual layout of the application will be developed on android studio using Java and XML. Later the application will be linked to server using php.

Test the functionalities and make necessary changes. Ask dummy users to use the app and make necessary changes according to their inputs.

Finally the mobile application will be launched on Google play.

2.2 Technologies and Tools

- PHP as our backend language for both mobile app and website.
- HTML, CSS, JavaScript as front end language for website. Bootstrap as a front end framework.
- Android Studio IDE for android app development.
- JSON for data transfer in mobile app.
- LAMP stack server.
- MySQL database.
- Google Cloud Messaging.
- LDAP server.

2.3 Existing Solutions

Till date our college is dependent on pen-paper works for keeping record of attendance and marks of students also we follow manual methods for time table generation.

Many faculties use excel sheet instead of pen paper work but that is also not very convenient as they need to maintain a different sheet for each class. Also use of excel sheet does not provide transparency to students. We came across an app developed by students of VIT, providing pretty much same functionality to faculties and students of VIT (except attendance), as mentioned in abstract above. So we decided to develop the same for our institute, incorporating the missing features of VIT app. Many apps are available which are helpful in maintaining attendance but they lack authentication and are not dependable. Also they were working on general concepts and layout for entering attendance was also not satisfactory. Our app addresses these problems as it requires faculty authentication and only the concerned faculty can update attendance and layout for entering attendance is very flexible and faculty can mark attendance randomly and in any order and also can mark all present and absent on single click. Also viewing of attendance is very comprehensible and various filters such as sorting on basis of present percent and date can be applied. Besides access to other systems is provided in an interactive way, acting as a one stop solution for faculties and students. Further scheduling of time table is a tedious and time taking work undertaken by every institute once or twice in an academic session. We are trying to develop a scheduler based on machine learning concepts and heuristics, which will generate time tables on a single click, based on information provided and certain constraints. Our attendance system can be used by any educational institute like school, college or university that require regularly to take attendance. Only thing needed is slight changes based on user requirements. also the given attendance system can be used by coaching institute and various business institution which require employee attendance management for financial purposes but it will require a slight changes as branches will be replaced with departments and classes with employee subdivision. It can work as an alternative to biometric based attendance system. While all the educational institute require timetable for schedule management. Which can be resolved with our time-table scheduler.

Chapter-3 Analysis

3.1 Software Requirements

- Php version > 5.5.9
- MySQL version > 5.5.58
- Apache version > 2.4.10
- Android OS Version >= 4.0 (Ice Cream Sandwich)

3.2 Hardware Requirements

The application is intended to be a standalone, single-user system. The application will run on an android mobile device or on android emulator. Physical server with minimum core i3 and 4 GB ram and 500 GB ssd rest depends upon scalability and uses.

3.3 Use Case Model

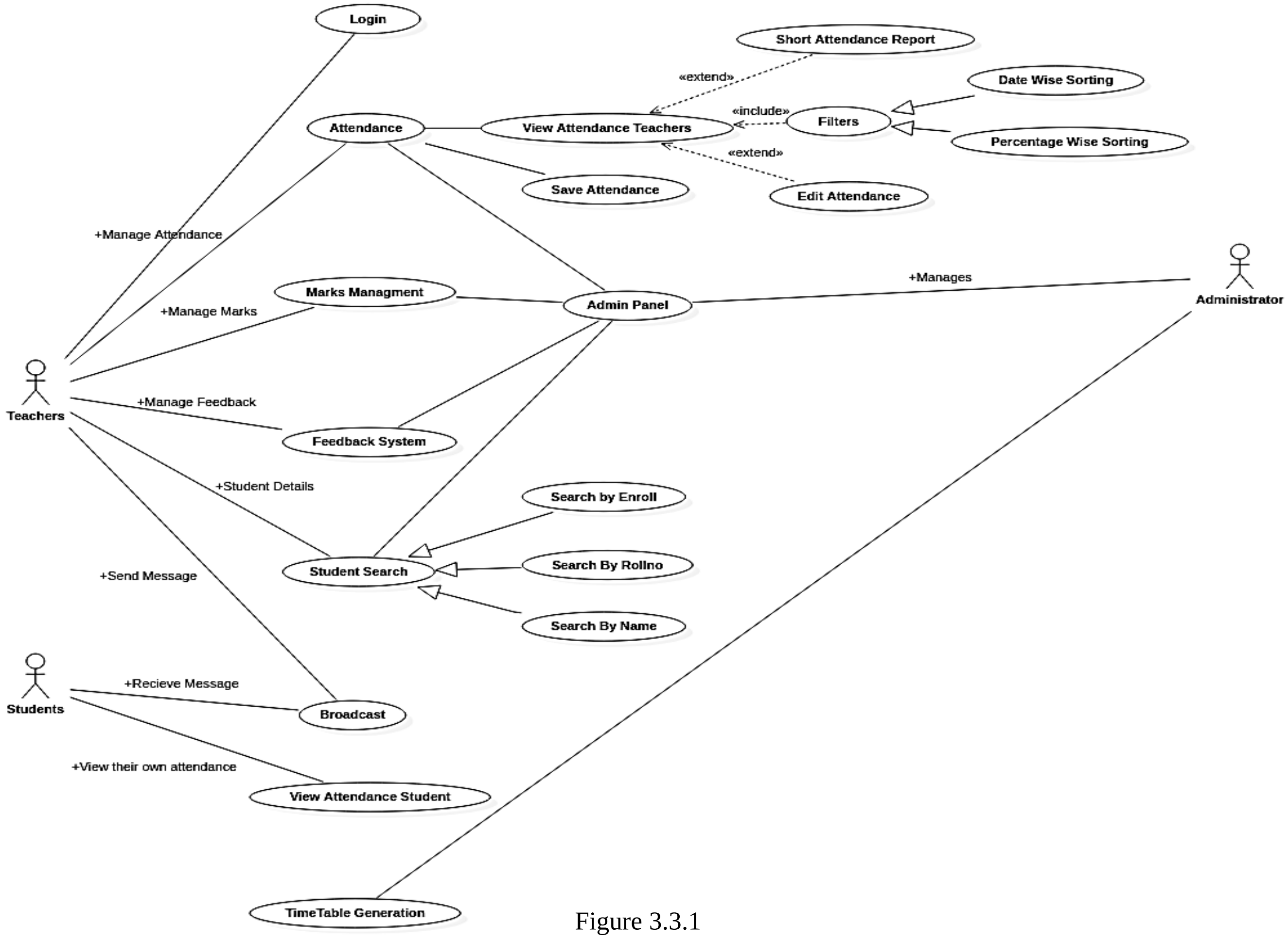


Figure 3.3.1

3.4 Use Case Description

Login

- Brief description - Faculties need to be login to perform any task for that they need to verify using their provided username & password.
- Users - Registered Faculty
- Basic flow
    1. User need to visit the faculty portal.
    2. After that user enter his credentials.
    3. Credentials are verified by system and he is redirected to faculty dashboard.
- Alternative flows
    1. If password/username is incorrect, the faculty is presented with a failure popup.
    2. If faculty is not registered, he/she has to contact admin for registration.

View attendance student

- Brief description - The user can be teacher or the student. Teacher can view attendance of all students he teaches and a student can view attendance of all class he attends.
- Users - A student with existing profile.
- Basic flow
    1. Student need to login first.
    2. Then select a subject to view its attendance.
    3. When student select one subject then complete information of his attendance will be shown to him.
- Alternative flows
    1. If details of student is incorrect he will get a message invalid username or password.
    2. If no subject is associated with that student then he will get a message no subjects to show.

View attendance faculty

- Brief description - The user can be teacher or the student. Teacher can view attendance of all students he teaches and a student can view attendance of all class he attends.
- Users - A faculty with existing profile.
- Basic flow
    1. Teacher need to login first.

2. Then select a class to view its attendance.
3. When teacher selects one class then complete information related to attendance of his class will be shown to him.

- Alternative flows
    1. If details of teacher is incorrect he will get a message invalid username or password.
    2. If no class is associated with that teacher then he will get a message no class to show.

Save attendance

- Brief description - The user can only be teacher here. Teacher can save attendance of the class he teaches.
- Users - User is teacher with existing profile.
- Basic flow
    1. Teacher need to login first.
    2. Then select a class for which he wants to save attendance.
    3. When teacher selects one class then name and roll no of that class will be shown to him.
    4. Teacher will click the roll no that are present and click on save.
    5. The attendance will be saved in database and message will be shown to teacher.
- Alternative flows
    1. If details of teacher is incorrect he will get a message invalid username or password.
    2. If no subject is associated with that student then he will get a message no subjects to show.
    3. If attendance is not saved due to some reason corresponding message will be shown to teacher and he can save the attendance again.

Filters

- Brief description - Teacher can view attendance according to filters i.e. date wise sorting, percentage wise sorting and short attendance report.
- Users - user is view with existing profile
- Basic flow
    1. Teacher need to login first.
    2. Then select a class for which he wants to view attendance.
    3. Attendance will be shown to him date wise by default.
    4. Teacher can sort the attendance percentage wise.
    5. Or can view the report of students who have short attendance.

- Alternative flows
- If details of teacher is incorrect he will get a message invalid username or password.
- If no class is associated with that teacher then he will get a message no class to show.

Edit attendance

- Brief description - The user can only be teacher here. Teacher can edit the attendance of the class he teaches and previously saved.
- Users - User is teacher with existing profile.
- Basic flow
  1. Teacher need to login first.
  2. Then select a class for which he wants to edit attendance.
  3. When teacher selects one class then name and roll no of that class and present/absent of all students will be shown to him.
  4. Teacher can change the attendance and save.
  5. The attendance will be saved in database and message will be shown to teacher.
- Alternative flows
  1. If details of teacher is incorrect he will get a message invalid username or password.
  2. If no subject is associated with that student then he will get a message no subjects to show.
  3. If attendance is not saved due to some reason corresponding message will be shown to teacher and he can save the attendance again.

Time table generation

- Basic Description - Admin can generate time table of all branches and years
- Users - Admin
- Basic flow-
  1. Admin will first Insert details about classes subject and faculties
  2. Then he will add some more details about teachers and student preferences.
  3. After that he will require to add configure some initial constraints to time table algorithm
  4. Finally he can generate the timetable using algorithm
  5. If he is not satisfied with the timetable he can reiterate from step 2.
- Alternative flow
  1. If initial parameters are not sufficient he can change the parameter till the fittest time-table is achieved.

2. Also previous year timetable can be used to generate a better timetable.

<u>Student Search</u>

- Brief description - Teacher can view student details such as their name, enrollment number, mobile number, email id, roll no.
- Users - Teachers
- Basic flow
  1. Faculty need to login first.
  2. Then go to search student option.
  3. Faculty enter the name or roll-no or enroll-no of student he want to search.
  4. Faculty is presented with student details.
- Alternative flows
  1. If details of teacher is incorrect he will get a message invalid username or password
  2. If student identifier doesn't match any student in database he will get a student no such student exists.

<u>Broadcast</u>

- Brief description - Teacher can broadcast messages to students. It is a one way form of communication.
- Users - Faculty & students
- Basic flow
  1. After login teacher is presented with their classes.
  2. They have to select the class they want to convey message to, it can be a particular class or message can be send to all the students.
  3. Teacher press send message button and the message is broadcasted to all the recipients.
- Alternative flows
  1. If message cannot be delivered the faculty is presented with a failure popup.

<u>Admin panel</u>

- Brief description - This is a centralized panel for uploading and editing all faculty and student related data from here faculty can be assigned to their particular classes similarly students can be assigned subjects also admin can have backup of a given semester from one click backup.

- User- Administrator
- Basic flow
    1. Admin logins into his account
    2. After login he will get a dashboard from which he can choose the system whose data need to be updated it can be anyone of attendance system, marks system, feedback system.
    3. Admin is presented with following options
        a. Classes - assign subject and students to classes
        b. Faculties - insert or update faculty details
        c. Subjects - insert or update subject/courses details
        d. Students - insert student details
        e. Batches - define separation roll-no for batch a & b
        f. Class coordinator - assign class coordinator for batches
    4. After selecting one option above admin can decide if he want to add details one by one or using batch mode in batch mode admin can bulk upload the data using csv or excel file
    5. Admin can also truncate the data if required in case of erroneous data.
    6. Admin can then save his work and logout.
- Alternative flows
    1. If details of admin is incorrect he will get a message invalid username or password
    2. If details can't be updated in batch mode he will be presented with error message regarding the cause of error.
    3. If admin mistakenly perform same action twice he will be presented with error.
    4. If admin try to add student without inserting details of faculty he will be presented with a prompt.

Chapter-4 Design

4.1 Technology Selection

- We are using PHP as our backend language for both mobile app and website as it is simple, open source, interpreted, faster and has a wide support.
- HTML, CSS, JavaScript as front end language for website. Bootstrap as a front end framework because it is easy to use and is lightweight.
- Android Studio IDE for android app development using java. Android Studio is official IDE from Google and supports smart development and automations.
- JSON for data transfer in mobile app.
- LAMP stack server in college to host our website for hosting php scripts and databases.
- MySQL database.
- Google Cloud Messaging for broadcasting messages.
- LDAP server for login authentication because of security reasons.

4.2 Sequence diagrams

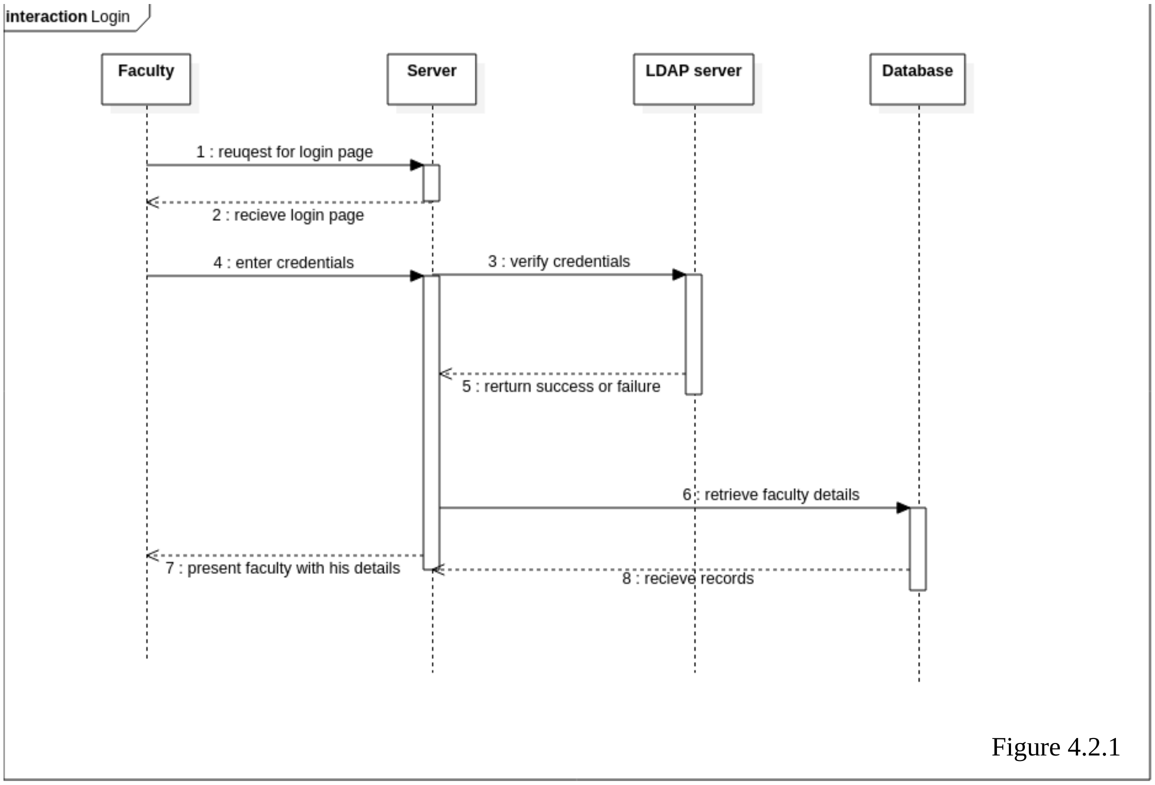


Figure 4.2.1

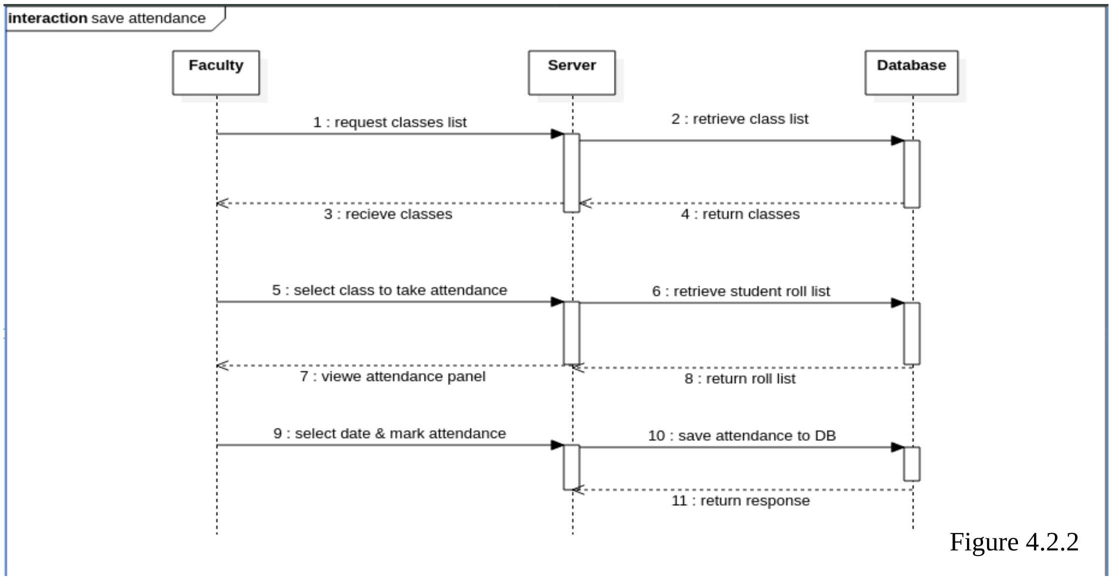

interaction save attendance
Faculty
Server
Database
1 : request classes list
2 : retrieve class list
3 : recieve classes
4 : return classes
5 : select class to take attendance
6 : retrieve student roll list
7 : viewe attendance panel
8 : return roll list
9 : select date & mark attendance
10 : save attendance to DB
11 : return response

Figure 4.2.2

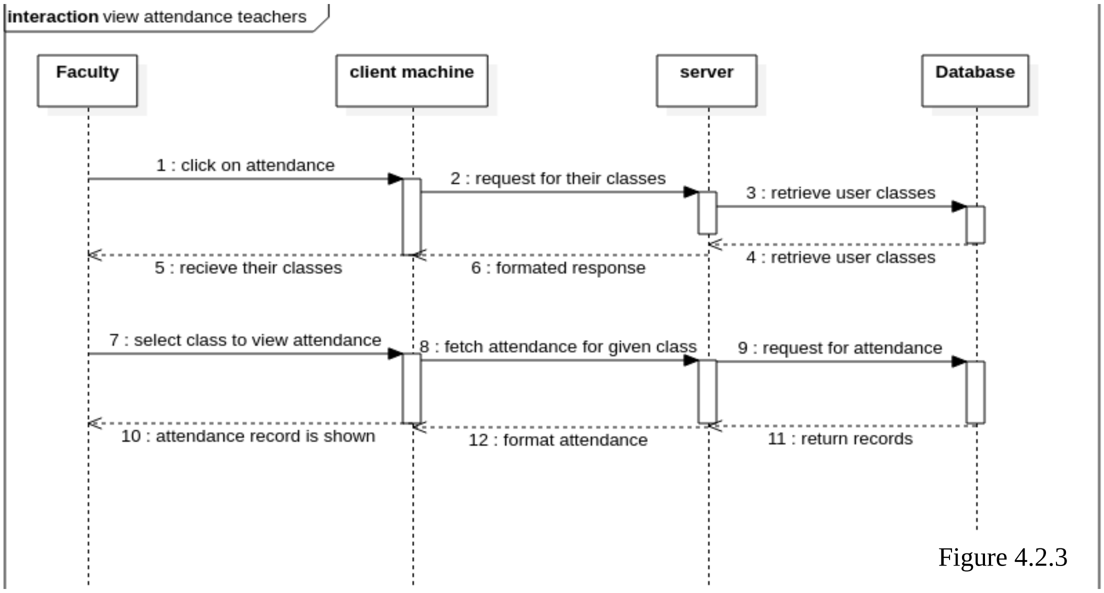

interaction view attendance teachers
Faculty
client machine
server
Database
1 : click on attendance
2 : request for their classes
3 : retrieve user classes
4 : retrieve user classes
5 : recieve their classes
6 : formated response
7 : select class to view attendance
8 : fetch attendance for given class
9 : request for attendance
10 : attendance record is shown
11 : return records
12 : format attendance

Figure 4.2.3

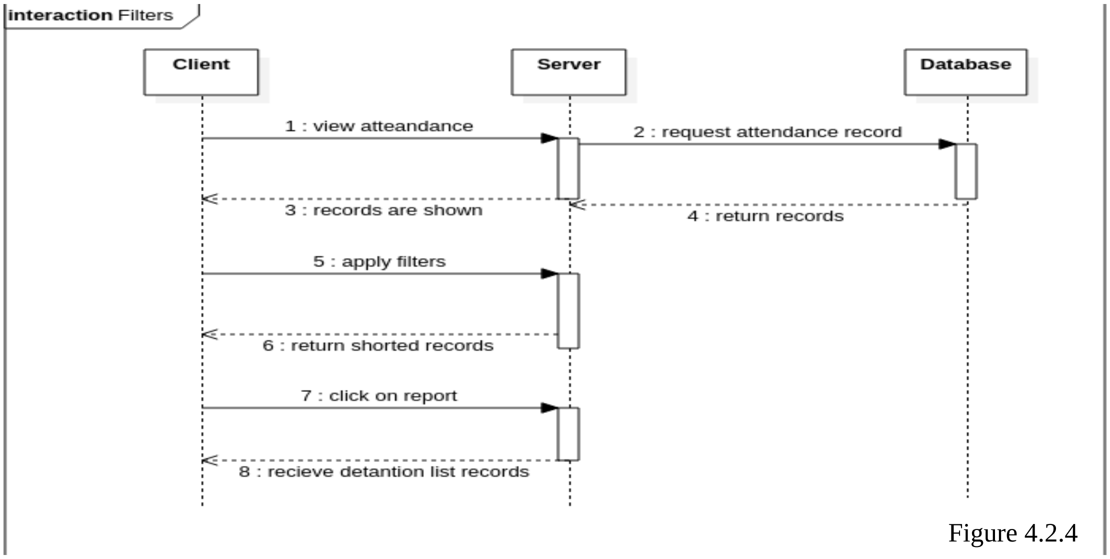


Figure 4.2.4

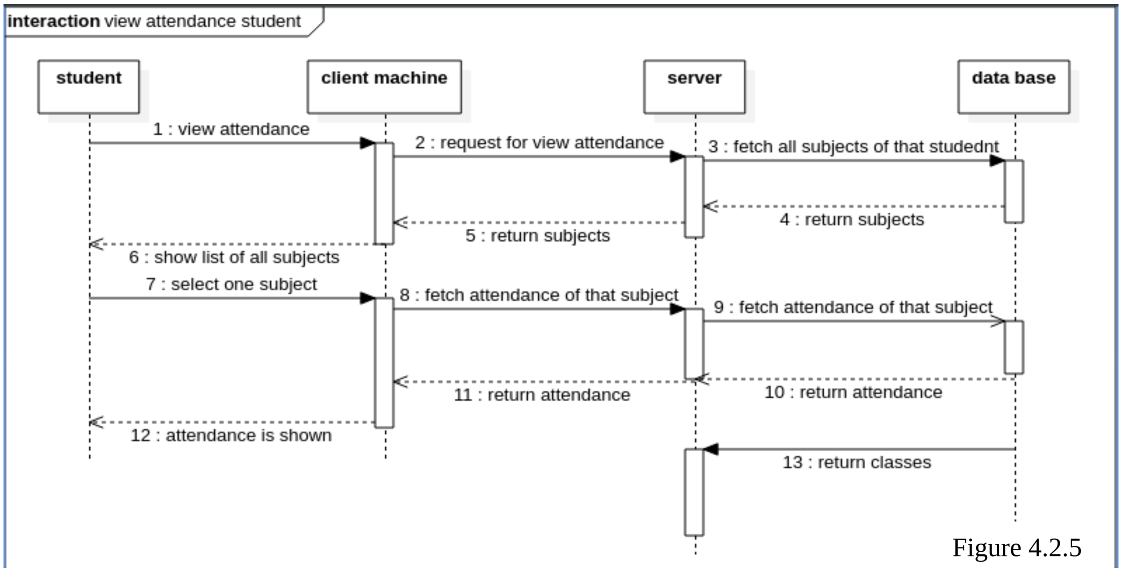

interaction view attendance student
student
client machine
server
data base
1 : view attendance
2 : request for view attendance
3 : fetch all subjects of that studednt
4 : return subjects
5 : return subjects
6 : show list of all subjects
7 : select one subject
8 : fetch attendance of that subject
9 : fetch attendance of that subject
10 : return attendance
11 : return attendance
12 : attendance is shown
13 : return classes

Figure 4.2.5

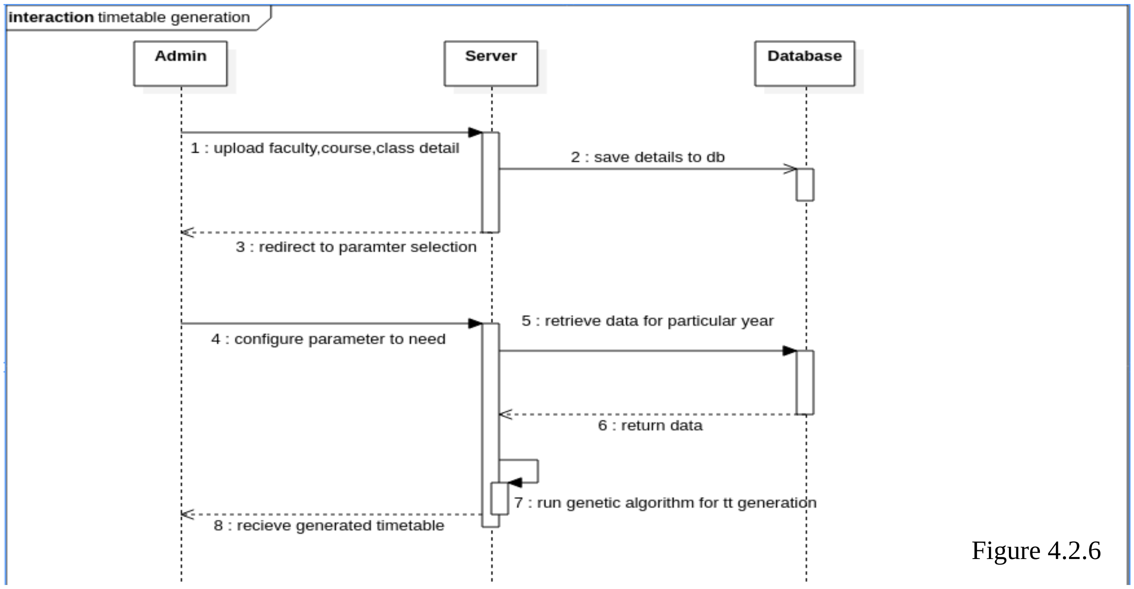

interaction timetable generation
Admin
Server
Database
1 : upload faculty,course,class detail
2 : save details to db
3 : redirect to paramter selection
4 : configure parameter to need
5 : retrieve data for particular year
6 : return data
7 : run genetic algorithm for tt generation
8 : recieve generated timetable

Figure 4.2.6

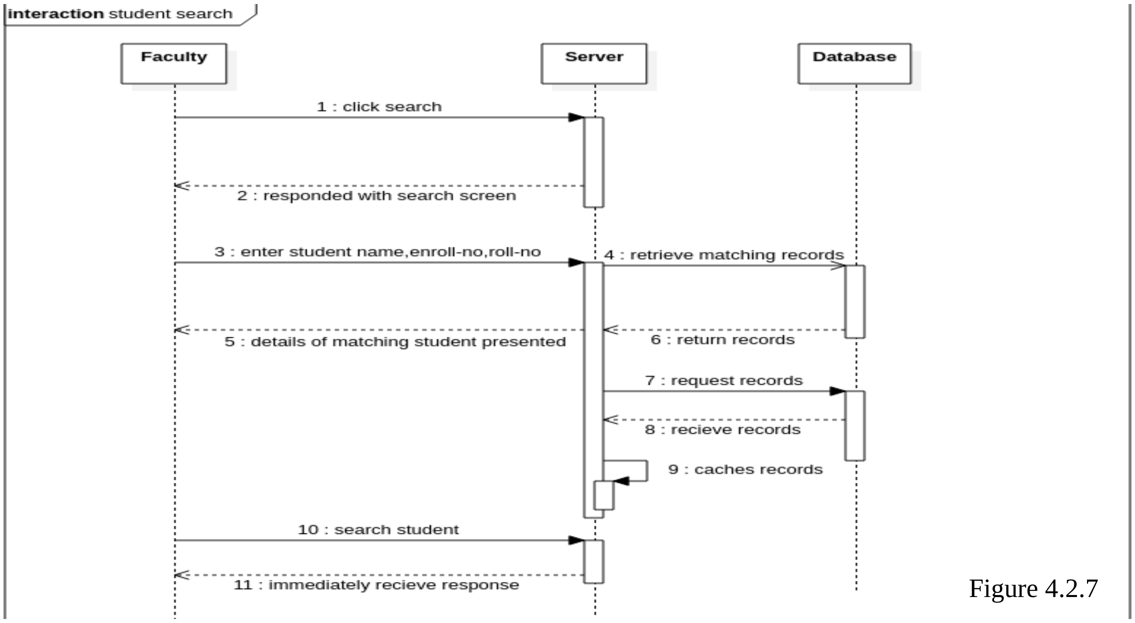

interaction student search
Faculty
Server
Database
1 : click search
2 : responded with search screen
3 : enter student name,enroll-no,roll-no
4 : retrieve matching records
5 : details of matching student presented
6 : return records
7 : request records
8 : recieve records
9 : caches records
10 : search student
11 : immediately recieve response

Figure 4.2.7

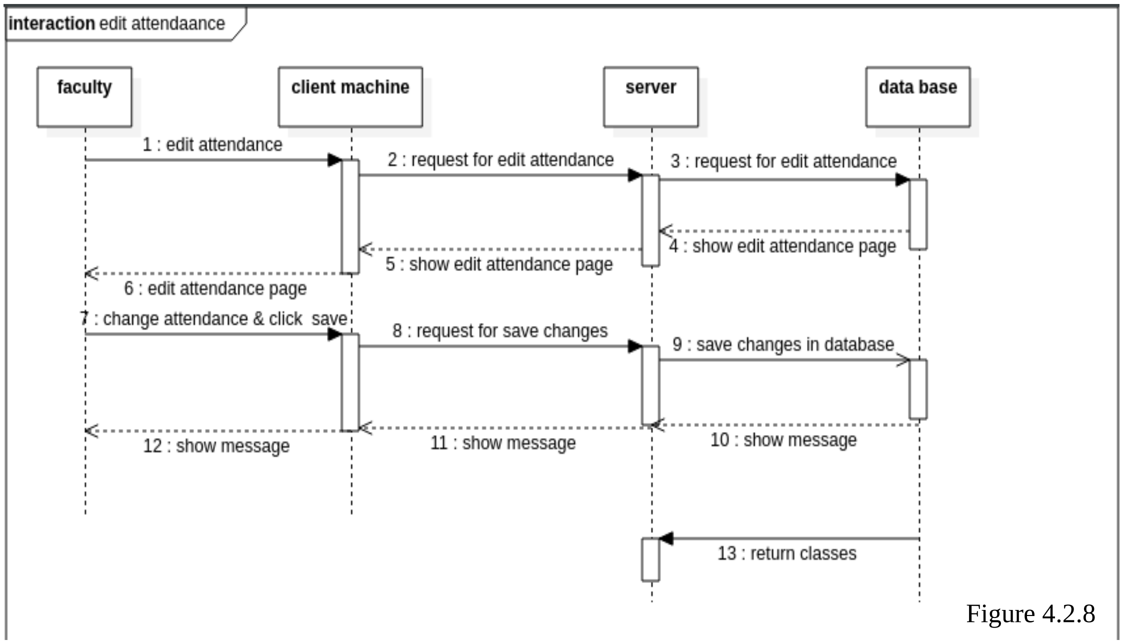

interaction edit attendaance
faculty
client machine
server
data base
1 : edit attendance
2 : request for edit attendance
3 : request for edit attendance
4 : show edit attendance page
5 : show edit attendance page
6 : edit attendance page
7 : change attendance & click save
8 : request for save changes
9 : save changes in database
10 : show message
11 : show message
12 : show message
13 : return classes

Figure 4.2.8

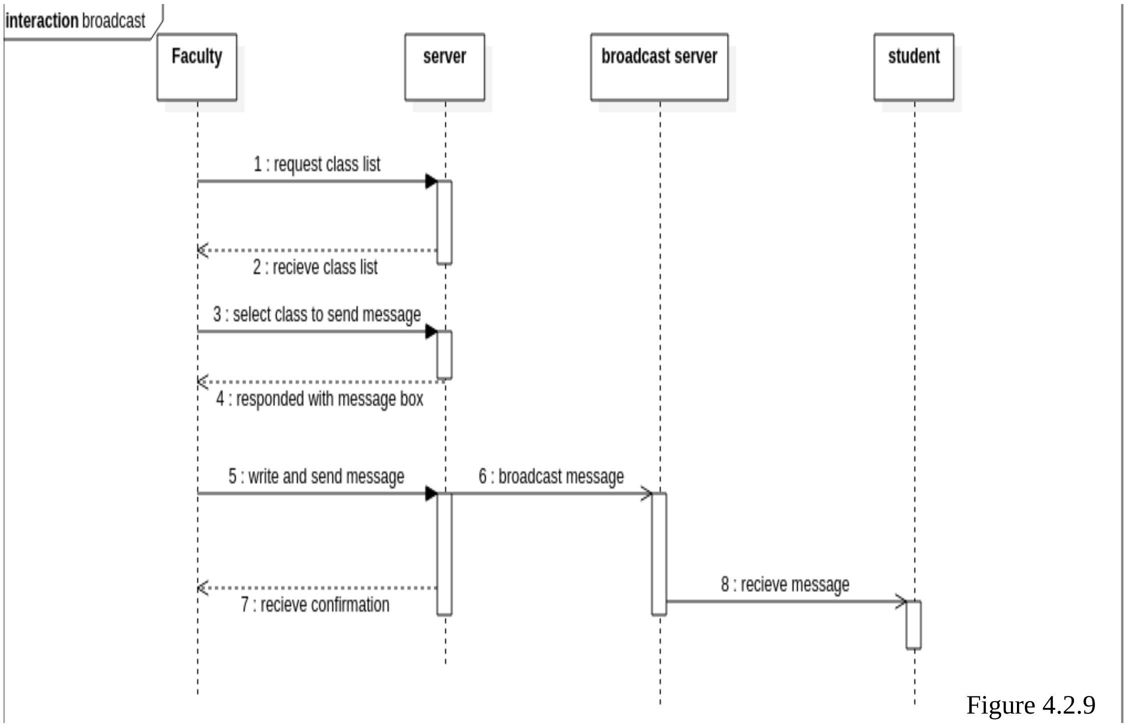
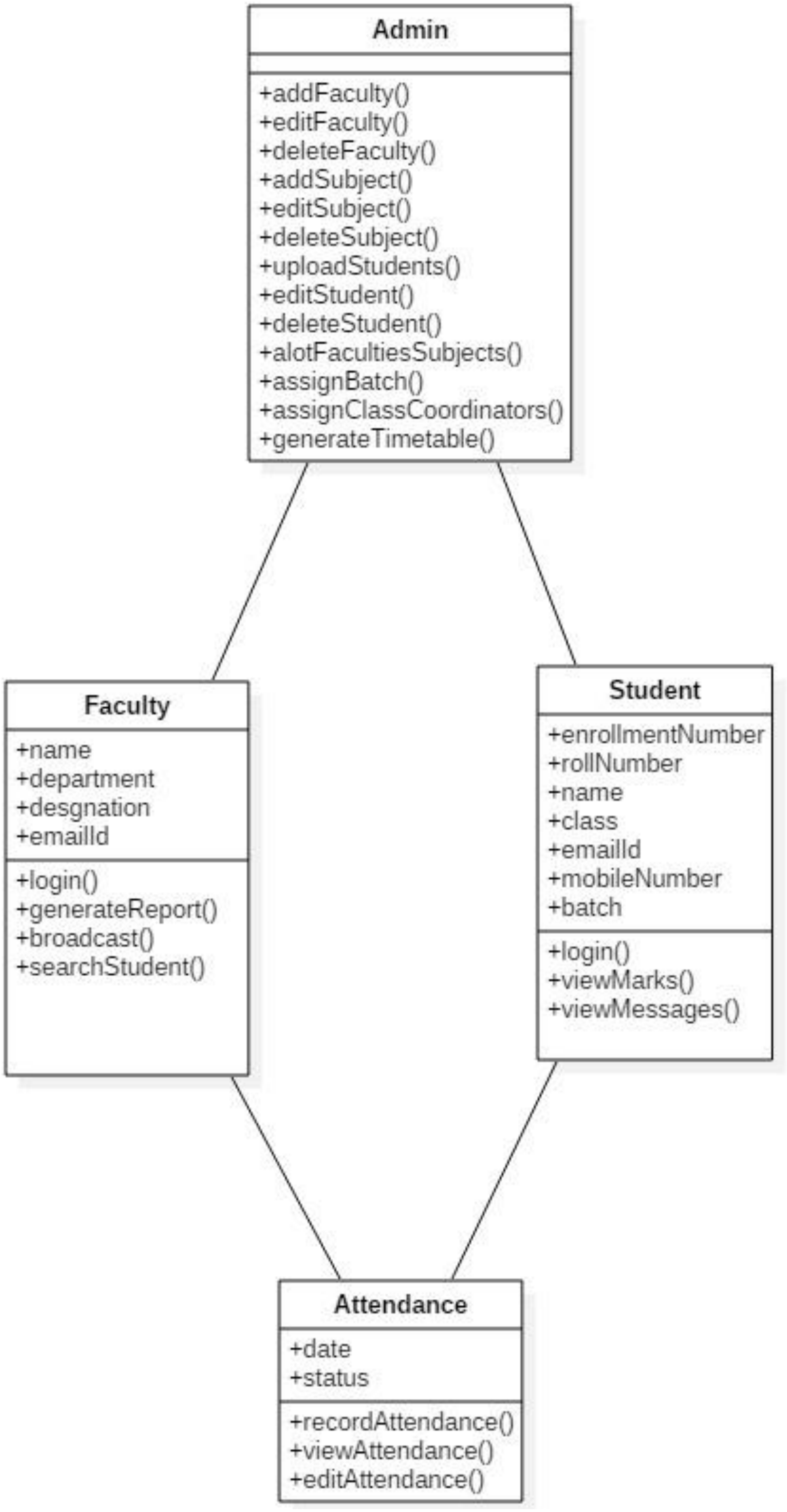


Figure 4.2.9

4.3 Packaging (Using Class Diagram)



4.4 Database Design

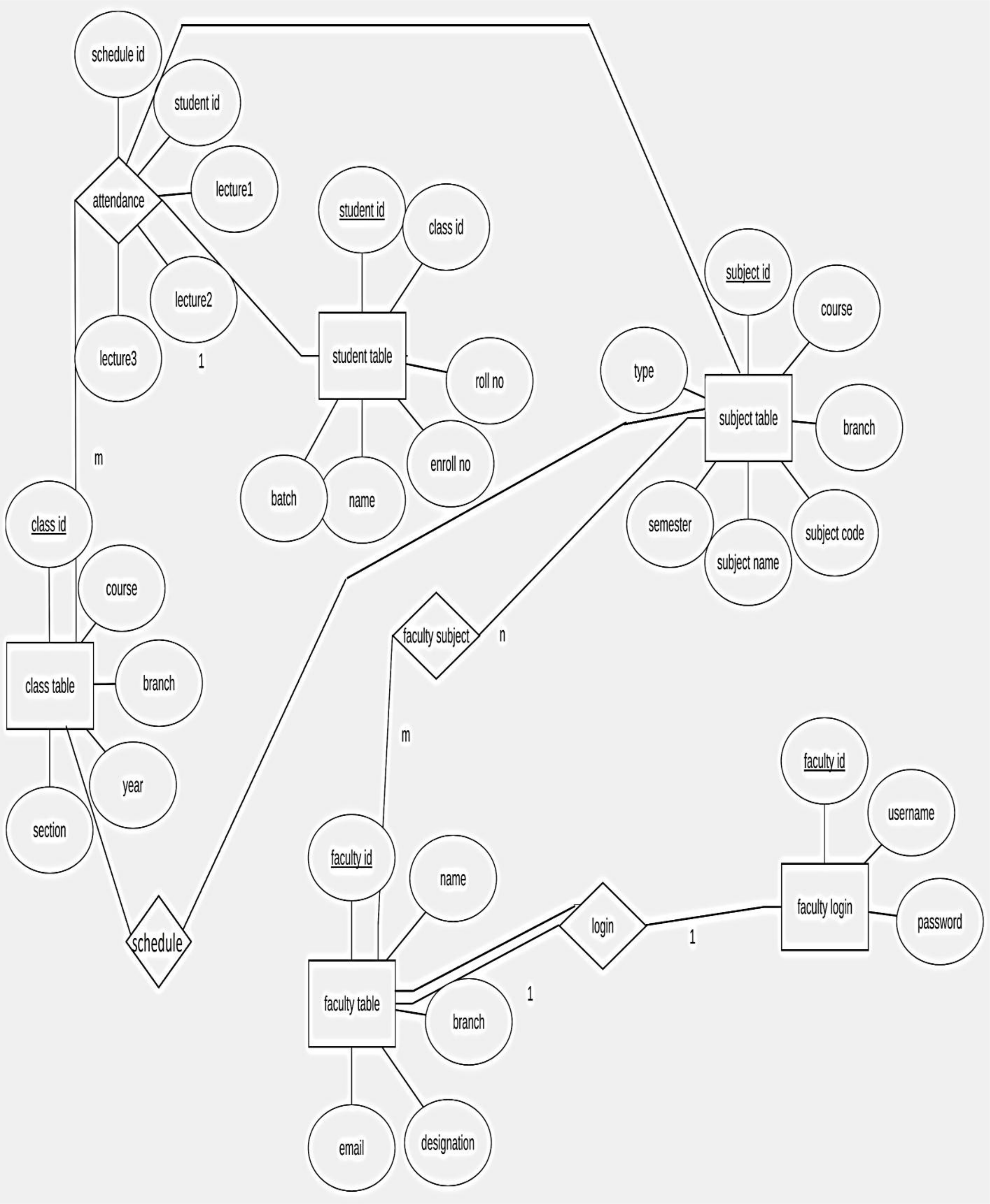
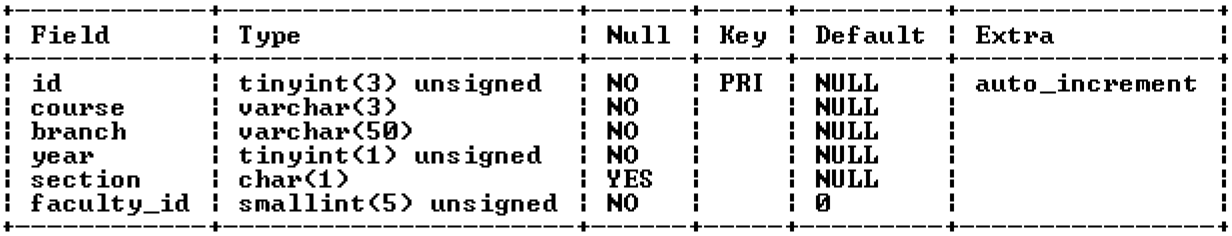


Figure 4.4.1

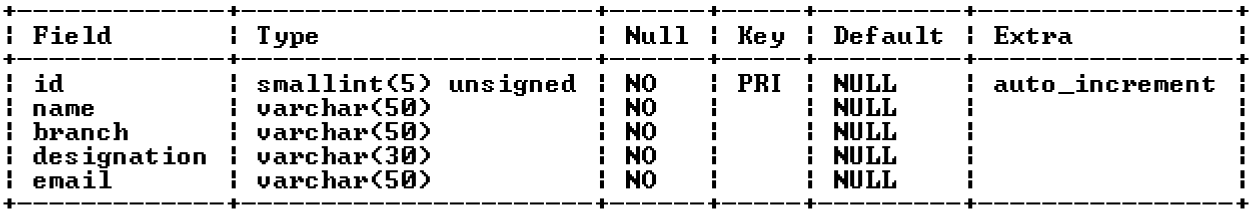
Based on this Entity relationship diagram following tables must be constructed in database:

Class Table - It will store the details of all classes with id as primary key.



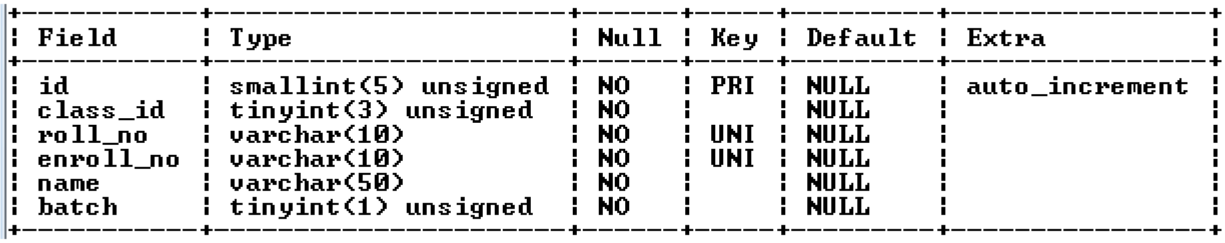
```
+------------+----------------------+------+-----+---------+----------------+
| Field      | Type                 | Null | Key | Default | Extra          |
+------------+----------------------+------+-----+---------+----------------+
| id         | tinyint(3) unsigned  | NO   | PRI | NULL    | auto_increment |
| course     | varchar(3)           | NO   |     | NULL    |                |
| branch     | varchar(50)          | NO   |     | NULL    |                |
| year       | tinyint(1) unsigned  | NO   |     | NULL    |                |
| section    | char(1)              | YES  |     | NULL    |                |
| faculty_id | smallint(5) unsigned | NO   |     | 0       |                |
+------------+----------------------+------+-----+---------+----------------+
```

Faculty Table - It will store the details about faculty.



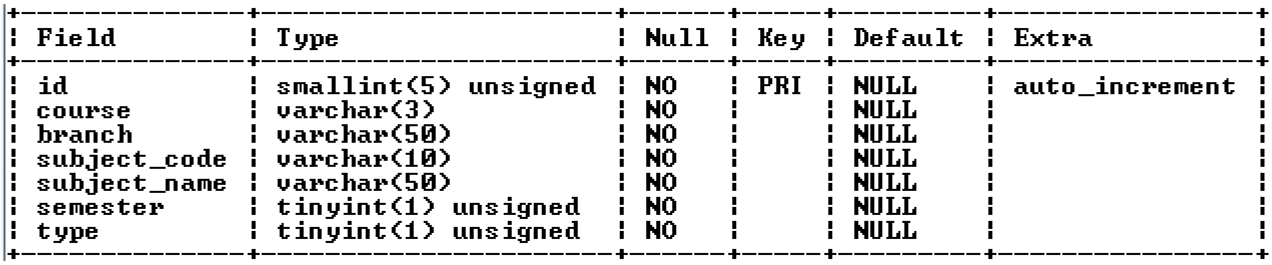
```
+-------------+----------------------+------+-----+---------+----------------+
| Field       | Type                 | Null | Key | Default | Extra          |
+-------------+----------------------+------+-----+---------+----------------+
| id          | smallint(5) unsigned | NO   | PRI | NULL    | auto_increment |
| name        | varchar(50)          | NO   |     | NULL    |                |
| branch      | varchar(50)          | NO   |     | NULL    |                |
| designation | varchar(30)          | NO   |     | NULL    |                |
| email       | varchar(50)          | NO   |     | NULL    |                |
+-------------+----------------------+------+-----+---------+----------------+
```

Student Table - It will store all the details of a student with id as primary key and enroll_no as unique key.



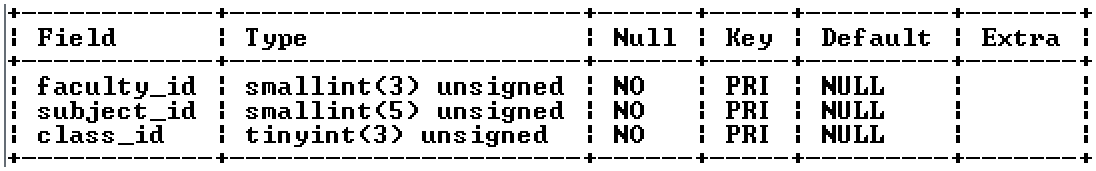
```
+-----------+----------------------+------+-----+---------+----------------+
| Field     | Type                 | Null | Key | Default | Extra          |
+-----------+----------------------+------+-----+---------+----------------+
| id        | smallint(5) unsigned | NO   | PRI | NULL    | auto_increment |
| class_id  | tinyint(3) unsigned  | NO   |     | NULL    |                |
| roll_no   | varchar(10)          | NO   | UNI | NULL    |                |
| enroll_no | varchar(10)          | NO   | UNI | NULL    |                |
| name      | varchar(50)          | NO   |     | NULL    |                |
| batch     | tinyint(1) unsigned  | NO   |     | NULL    |                |
+-----------+----------------------+------+-----+---------+----------------+
```

Subject Table - It stores details of all the courses taught in various semesters with id as primary key.



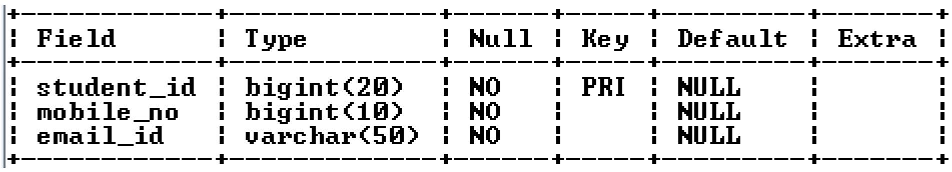
```
+--------------+----------------------+------+-----+---------+----------------+
| Field        | Type                 | Null | Key | Default | Extra          |
+--------------+----------------------+------+-----+---------+----------------+
| id           | smallint(5) unsigned | NO   | PRI | NULL    | auto_increment |
| course       | varchar(3)           | NO   |     | NULL    |                |
| branch       | varchar(50)          | NO   |     | NULL    |                |
| subject_code | varchar(10)          | NO   |     | NULL    |                |
| subject_name | varchar(50)          | NO   |     | NULL    |                |
| semester     | tinyint(1) unsigned  | NO   |     | NULL    |                |
| type         | tinyint(1) unsigned  | NO   |     | NULL    |                |
+--------------+----------------------+------+-----+---------+----------------+
```

Faculty Subject Table - It will store all the subject assigned to a particular teacher.



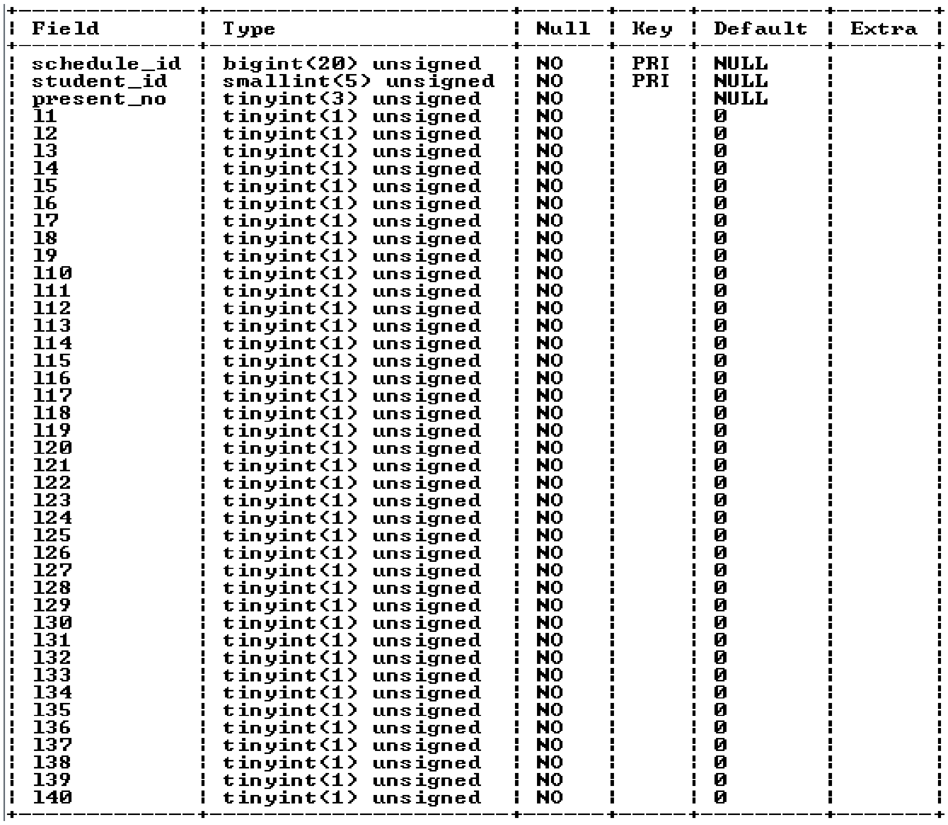
```
+------------+----------------------+------+-----+---------+-------+
| Field      | Type                 | Null | Key | Default | Extra |
+------------+----------------------+------+-----+---------+-------+
| faculty_id | smallint(3) unsigned | NO   | PRI | NULL    |       |
| subject_id | smallint(5) unsigned | NO   | PRI | NULL    |       |
| class_id   | tinyint(3) unsigned  | NO   | PRI | NULL    |       |
+------------+----------------------+------+-----+---------+-------+
```

Student Profile - It will store the student identity information.



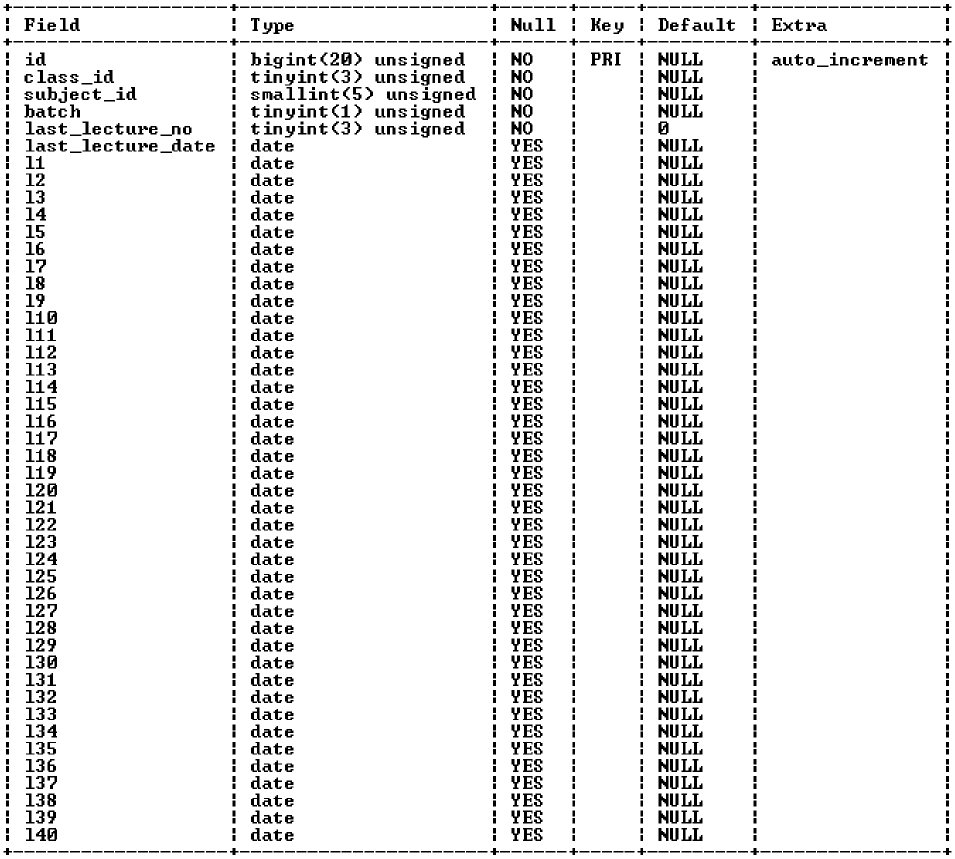
```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| student_id | bigint(20)  | NO   | PRI | NULL    |       |
| mobile_no  | bigint(10)  | NO   |     | NULL    |       |
| email_id   | varchar(50) | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

Attendance Table - It will store the attendance record of every student.



```
+--------------+----------------------+------+-----+---------+-------+
| Field        | Type                 | Null | Key | Default | Extra |
+--------------+----------------------+------+-----+---------+-------+
| schedule_id  | bigint(20) unsigned  | NO   | PRI | NULL    |       |
| student_id   | smallint(5) unsigned | NO   | PRI | NULL    |       |
| present_no   | tinyint(3) unsigned  | NO   |     | NULL    |       |
| l1           | tinyint(1) unsigned  | NO   |     | 0       |       |
| l2           | tinyint(1) unsigned  | NO   |     | 0       |       |
| l3           | tinyint(1) unsigned  | NO   |     | 0       |       |
| l4           | tinyint(1) unsigned  | NO   |     | 0       |       |
| l5           | tinyint(1) unsigned  | NO   |     | 0       |       |
| l6           | tinyint(1) unsigned  | NO   |     | 0       |       |
| l7           | tinyint(1) unsigned  | NO   |     | 0       |       |
| l8           | tinyint(1) unsigned  | NO   |     | 0       |       |
| l9           | tinyint(1) unsigned  | NO   |     | 0       |       |
| l10          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l11          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l12          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l13          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l14          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l15          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l16          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l17          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l18          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l19          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l20          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l21          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l22          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l23          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l24          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l25          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l26          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l27          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l28          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l29          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l30          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l31          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l32          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l33          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l34          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l35          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l36          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l37          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l38          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l39          | tinyint(1) unsigned  | NO   |     | 0       |       |
| l40          | tinyint(1) unsigned  | NO   |     | 0       |       |
+--------------+----------------------+------+-----+---------+-------+
```

Schedule Table - It will store the lecture date corresponding to each lecture.



```
+-------------------+----------------------+------+-----+---------+----------------+
| Field             | Type                 | Null | Key | Default | Extra          |
+-------------------+----------------------+------+-----+---------+----------------+
| id                | bigint(20) unsigned  | NO   | PRI | NULL    | auto_increment |
| class_id          | tinyint(3) unsigned  | NO   |     | NULL    |                |
| subject_id        | smallint(5) unsigned | NO   |     | NULL    |                |
| batch             | tinyint(1) unsigned  | NO   |     | NULL    |                |
| last_lecture_no   | tinyint(3) unsigned  | NO   |     | 0       |                |
| last_lecture_date | date                 | YES  |     | NULL    |                |
| l1                | date                 | YES  |     | NULL    |                |
| l2                | date                 | YES  |     | NULL    |                |
| l3                | date                 | YES  |     | NULL    |                |
| l4                | date                 | YES  |     | NULL    |                |
| l5                | date                 | YES  |     | NULL    |                |
| l6                | date                 | YES  |     | NULL    |                |
| l7                | date                 | YES  |     | NULL    |                |
| l8                | date                 | YES  |     | NULL    |                |
| l9                | date                 | YES  |     | NULL    |                |
| l10               | date                 | YES  |     | NULL    |                |
| l11               | date                 | YES  |     | NULL    |                |
| l12               | date                 | YES  |     | NULL    |                |
| l13               | date                 | YES  |     | NULL    |                |
| l14               | date                 | YES  |     | NULL    |                |
| l15               | date                 | YES  |     | NULL    |                |
| l16               | date                 | YES  |     | NULL    |                |
| l17               | date                 | YES  |     | NULL    |                |
| l18               | date                 | YES  |     | NULL    |                |
| l19               | date                 | YES  |     | NULL    |                |
| l20               | date                 | YES  |     | NULL    |                |
| l21               | date                 | YES  |     | NULL    |                |
| l22               | date                 | YES  |     | NULL    |                |
| l23               | date                 | YES  |     | NULL    |                |
| l24               | date                 | YES  |     | NULL    |                |
| l25               | date                 | YES  |     | NULL    |                |
| l26               | date                 | YES  |     | NULL    |                |
| l27               | date                 | YES  |     | NULL    |                |
| l28               | date                 | YES  |     | NULL    |                |
| l29               | date                 | YES  |     | NULL    |                |
| l30               | date                 | YES  |     | NULL    |                |
| l31               | date                 | YES  |     | NULL    |                |
| l32               | date                 | YES  |     | NULL    |                |
| l33               | date                 | YES  |     | NULL    |                |
| l34               | date                 | YES  |     | NULL    |                |
| l35               | date                 | YES  |     | NULL    |                |
| l36               | date                 | YES  |     | NULL    |                |
| l37               | date                 | YES  |     | NULL    |                |
| l38               | date                 | YES  |     | NULL    |                |
| l39               | date                 | YES  |     | NULL    |                |
| l40               | date                 | YES  |     | NULL    |                |
+-------------------+----------------------+------+-----+---------+----------------+
```

Chapter – 5 Conclusion

The main aim of our project is to provide:

- An attendance interface for faculty.
- An attendance interface for students.
- An integrated attendance, marks and feedback system.
- A time table generator based on constraints defined by using genetic algorithm.
- An attendance detention list generator for faculties.
- An admin portal to manage faculty details and provide this details to various other systems.

Bibliography

Application programming interface(API)	*A set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service.*
Android os version 4.0 or above	*Includes version Ice Cream Sandwich and above*
Database	*Collection of all the information acquired by the application*
Software Design Specification	*In the context of software, Design Specification is usually a design document that describes all data, architectural, interface and component-level design for the software. A design specification provides explicit information about the requirements for a product and how the product is to be put together.*
SQL	*Structured Query Language.* ***SQL*** *is used to communicate with a database.*

Appendix

[1]. V Shehu, A Dika ,”Using real time computer vision algorithms in automatic attendance management systems ,”International Journal of Advanced Research in Computer and Communication Engineering,Vol. 2, Issue 6, June 2013

[2]. G.Satyanarayana Reddy,Rallabandi Srinivasu,Srikanth Reddy Rikkula,Vuda Sreenivasa Rao,” Management Information System To Help Managers For Providing Decision Making In An Organization,” International Journal of Reviews in Computing, ISSN: 2076-3328, EISSN: 2076-3336

[3]. V.Ramesh, P.Parkavi and K.Ramar, “Predicting student performance: A statistical and data mining approach,” International Journal of Computer Applications (IJCA), vol. 63-No.8,February 2013

[4]. P.Meena Kumari, SK.Abdul Nabi and Puppal Priyanka,”Educational data mining and its role in educational field,” International Journal of Computer Science and Information Technologies (IJCSIT),vol. 5(2),2014