IIT(BHU) - Management Portal

Project Documentation

Software Engineering - CSE381

Pankaj Kumar Jha
Roll 17074010
pankaj.krjha.cse17@itbhu.ac.in

Professor
Dr. AK Tripathi
Mentor
Miss Deepti

Department of Computer Science and Engineering
Indian Institute of Technology (BHU)
India
June 10, 2020

0 Contents

1 Introduction

The Objective of IIT BHU management system is to keep the allow management keep the records of Student and staffs, running courses,allocated courses.student should be able to register for courses online and admin should be able to allot courses to different teachers. Admin should be able to allot hostel and rooms to students.

1.1 Activities performed

The following activities were performed during the course of the project

- Requirement Collection
- System Design
- Coding
- Testing the Application

2 Requirement Collection

2.0.1 Identification of stakeholders

This software design is a difficult and complex decision where it involves people issues more than technological issues. Identification of stakeholders is an important step during the implementation, because if done improperly, it will lead to failure of the implementation project. The stakeholders are listed below,

Teachers
Teachers are one of the stakeholder for the software.They should be able to see the courses allotted to them. They should be able to see the students in the courses and other teachers co-teaching the courses. They should be able to mark attendance.

Students
Students are end users of the software. The attendance, internals marks uploaded by the teachers are viewed by them. It helps track their attendance. It helps them communicate with teachers. They should be able to view their results and make course registrations

Administrator
Admin has complete acess to database.He should be able to add, delete and update the teachers,students and staffs.He is also able to create more such staff accounts and manage the permissions in it.

2.1 Requirement Specification

2.1.1 Functional Requirements

Product Features

- Each teacher will be able to enter attendance and marks for their respective students.
- Each student will be able to view the attendance status for their respective courses.
- The teachers will be able to view the co-teaching staffs
- The students will be able to Communicate and provide feedback to their teachers.
- The warden can allot rooms and hostel to the students
- The administrator will be able to view and update information such as departments, classes, teachers, students, courses.

The student should have the following features:

- View the Attendance status of the courses to which they are enrolled.
- View the Marks of the courses to which they are enrolled.
- View the notification from the college administrator.
- Communicate or give feedback to their respective teachers.

The staff should have the following features:

- Access to the information of all students that attend their courses.
- Add and edit the Attendance status of those students.
- Add and edit the exam marks of those students.
- View co-teaching staffs

The administrator should have the following features:

- Add and update students, teachers and courses.
- Assign teachers and students to courses.
- Delete students, teachers and courses.

2.1.2 Non-functional requirements

Security requirements
The database should be secured. There should not be any sensitive information in client side.
Software Quality Attributes
Availability: The users must always be able to view their information so that they can keep track regularly.
Correctness: The information about attendance and marks must be correct to not feed wrong information to the users.
Responsiveness: The users access the system from various platforms such as desktops and mobile phones. The software must be portable to all platforms and the user experience must be optimal.

3 System Design

Various Design concepts and processes were applied to this project. Following concepts like separation of concerns, the software is divided into individual modules that are functionally independent and incorporate information hiding. The software is divided into 4 modules which are students, teachers,administrators, Hostel.

3.1 Teacher

The Teacher will have unique id.It will be related to department.Names, Address, Phone Number are stored and Appropriate permissions are set.It will have access to view the students courses and co-teaching members.

3.2 Administrator

The administrator will have access to all the information in the different tables in the database. They will access all the tables in a list form. They will be able to add an entry in any table and also edit them. The design of the view for the admin will provide a modular interface so that querying the tables will be optimized. They will be provided with search and filter features so that they can access data efficiently.

3.3 Hostel

Hostel will have a id for identification with field name,location,no of rooms. Each hostel will have warden. Warden is a instance of staff who is responsible for Management of Hostel

3.4 Student

Each student belongs to a class identified by semester and section and year. Each class belongs to a department and is assigned a set of courses. Therefore, these courses are common to all students of that class. The students are given a unique username and password to login. Each of them will have a different view. These views are described below.

Student information
Each student can view only their own personal information. Also, they can view the courses they are enrolled in and the attendance marks of each of those.

Attendance information
Attendance for each course will be displayed. This includes the number of attended classes and the attendance percentage. If the attendance percentage is below a specified threshold, say 75, It will be marked in red otherwise it will be in green. There will also be a day wise attendance view for each course which shows the date and status.

This will be presented in a calendar format.

Marks information

There will be 5 events and 1 semester end examination for each course. The marks for each of these will be provided in the system.

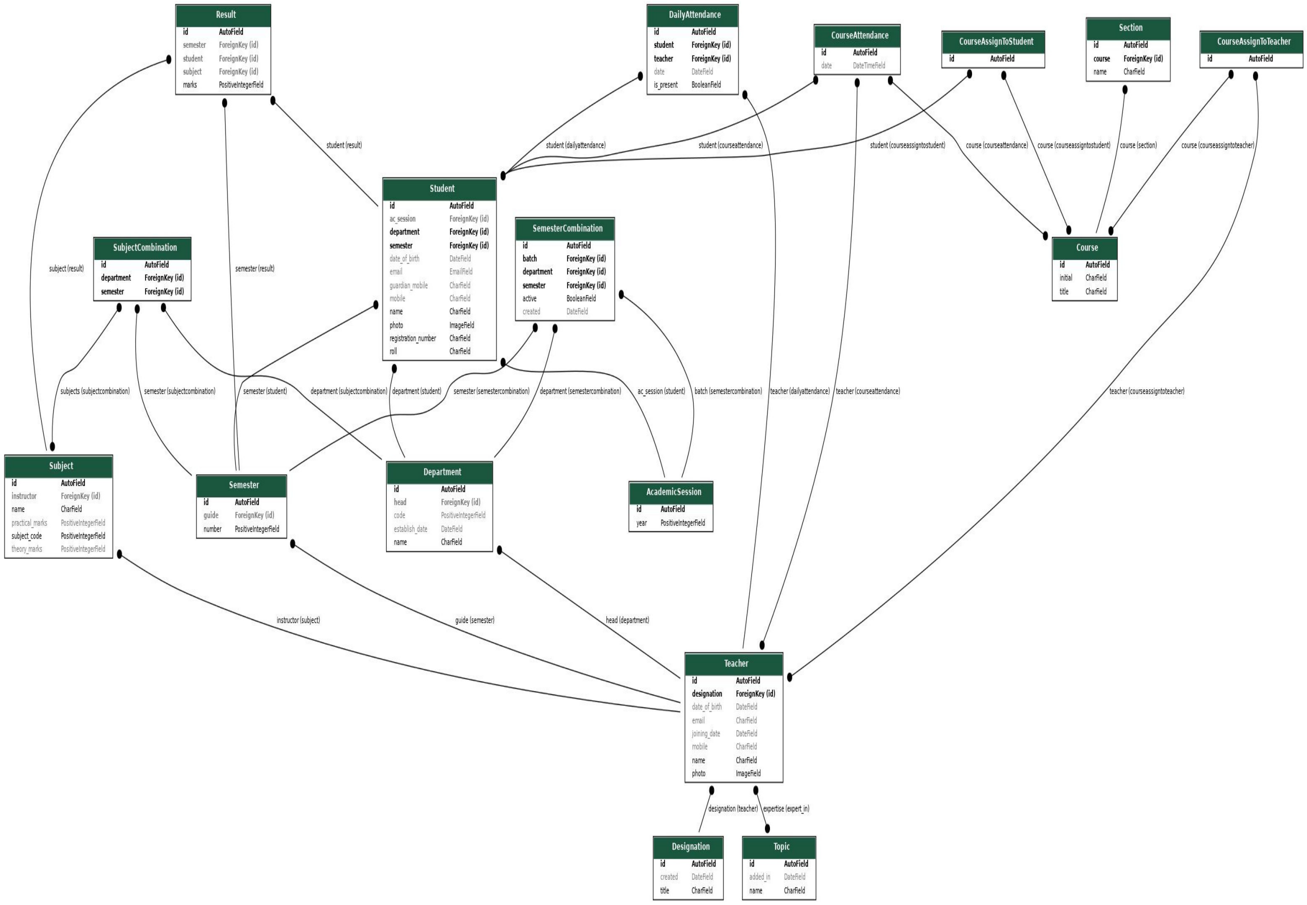
Notifications and events

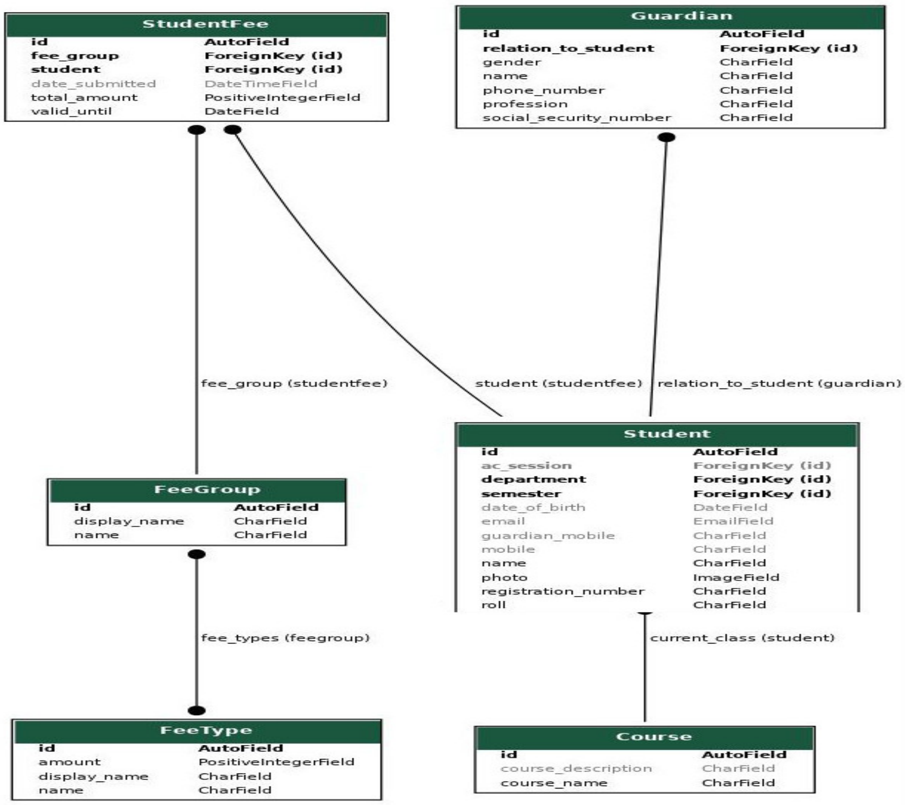
This section is common to all students. Notifications are messages from the admin such as declaration of holidays, test timetable etc. The events and their details are specified here.

3.5 UML Diagrams

The class diagram states the different classes involved in the software. For each class, a set of attributes and methods are included. The relationship between the classes are also specified. For example, the teacher class has the attributes Id, name, phone no, address and methods such as marking attendance, declaring marks and preparing report cards. Each instance of the teacher class belongs to a department. This is specified by the relationship between Teacher and Department classes.

Please Note that The modules of django framework are also included in UML diagrams. Since the complete Diagram can't be accomodated in One part it is separated into 4 parts as per modules..





StudentFee	
id	**AutoField**
fee_group	**ForeignKey (id)**
student	**ForeignKey (id)**
date_submitted	DateTimeField
total_amount	PositiveIntegerField
valid_until	DateField

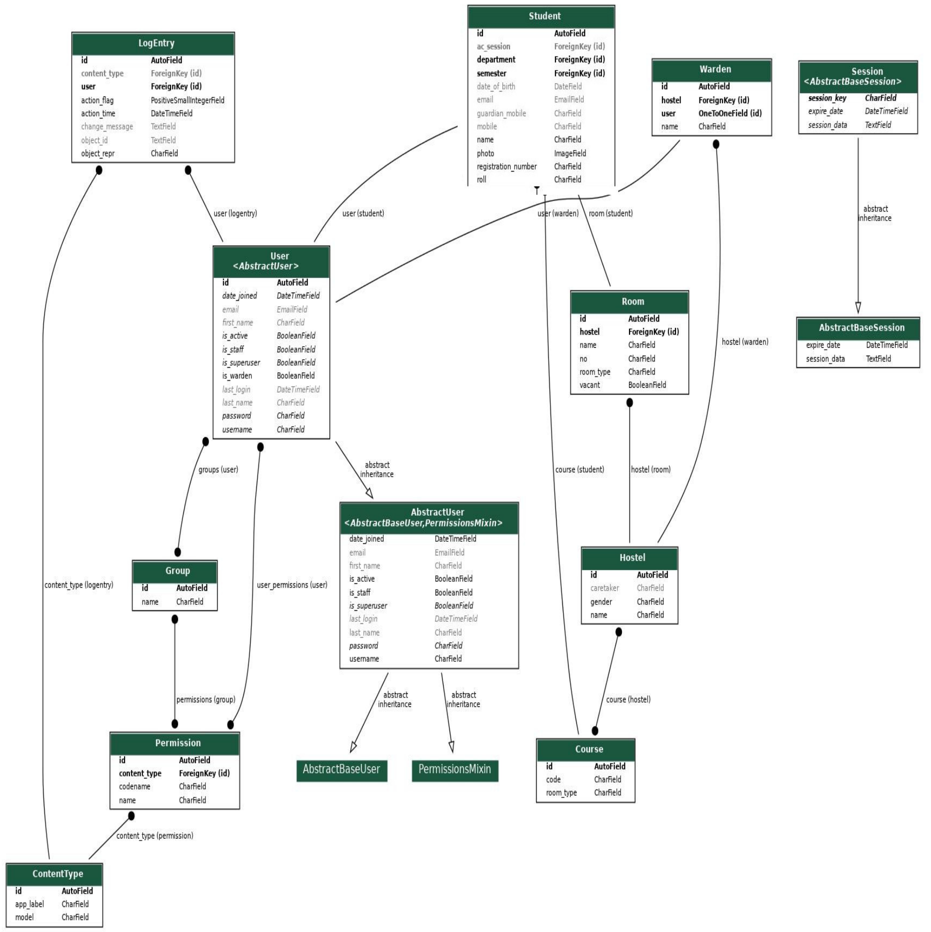
Guardian	
id	**AutoField**
relation_to_student	**ForeignKey (id)**
gender	CharField
name	CharField
phone_number	CharField
profession	CharField
social_security_number	CharField

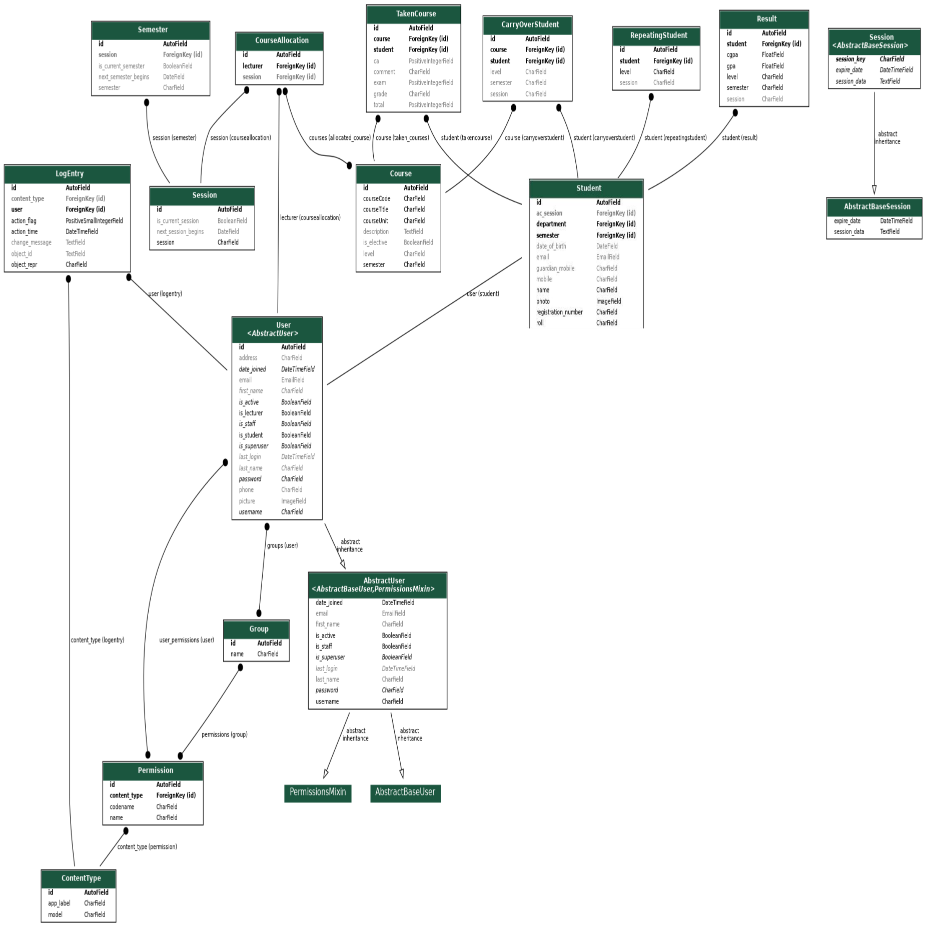
Student	
id	**AutoField**
ac_session	**ForeignKey (id)**
department	**ForeignKey (id)**
semester	**ForeignKey (id)**
date_of_birth	DateField
email	EmailField
guardian_mobile	CharField
mobile	CharField
name	CharField
photo	ImageField
registration_number	CharField
roll	CharField

FeeGroup	
id	**AutoField**
display_name	CharField
name	CharField

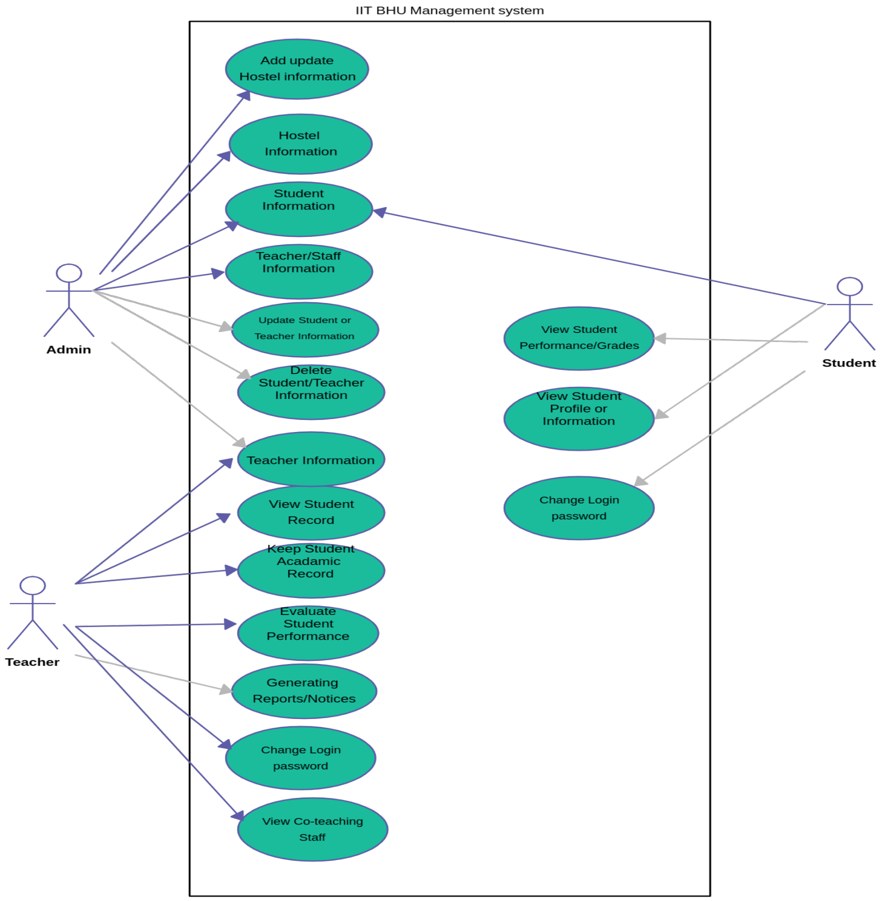
FeeType	
id	**AutoField**
amount	PositiveIntegerField
display_name	CharField
name	CharField

Course	
id	**AutoField**
course_description	CharField
course_name	CharField


LogEntry
id AutoField
content_type ForeignKey (id)
user ForeignKey (id)
action_flag PositiveSmallIntegerField
action_time DateTimeField
change_message TextField
object_id TextField
object_repr CharField
Student
id AutoField
ac_session ForeignKey (id)
department ForeignKey (id)
semester ForeignKey (id)
date_of_birth DateField
email EmailField
guardian_mobile CharField
mobile CharField
name CharField
photo ImageField
registration_number CharField
roll CharField
Warden
id AutoField
hostel ForeignKey (id)
user OneToOneField (id)
name CharField
Session
<AbstractBaseSession>
session_key CharField
expire_date DateTimeField
session_data TextField
user (logentry)
user (student)
user (warden)
room (student)
abstract
inheritance
User
<AbstractUser>
id AutoField
date_joined DateTimeField
email EmailField
first_name CharField
is_active BooleanField
is_staff BooleanField
is_superuser BooleanField
is_warden BooleanField
last_login DateTimeField
last_name CharField
password CharField
username CharField
Room
id AutoField
hostel ForeignKey (id)
name CharField
no CharField
room_type CharField
vacant BooleanField
hostel (warden)
AbstractBaseSession
expire_date DateTimeField
session_data TextField
groups (user)
abstract
inheritance
course (student)
hostel (room)
AbstractUser
<AbstractBaseUser,PermissionsMixin>
date_joined DateTimeField
email EmailField
first_name CharField
is_active BooleanField
is_staff BooleanField
is_superuser BooleanField
last_login DateTimeField
last_name CharField
password CharField
username CharField
content_type (logentry)
Group
id AutoField
name CharField
user_permissions (user)
Hostel
id AutoField
caretaker CharField
gender CharField
name CharField
permissions (group)
abstract
inheritance
abstract
inheritance
course (hostel)
Permission
id AutoField
content_type ForeignKey (id)
codename CharField
name CharField
AbstractBaseUser
PermissionsMixin
Course
id AutoField
code CharField
room_type CharField
content_type (permission)
ContentType
id AutoField
app_label CharField
model CharField

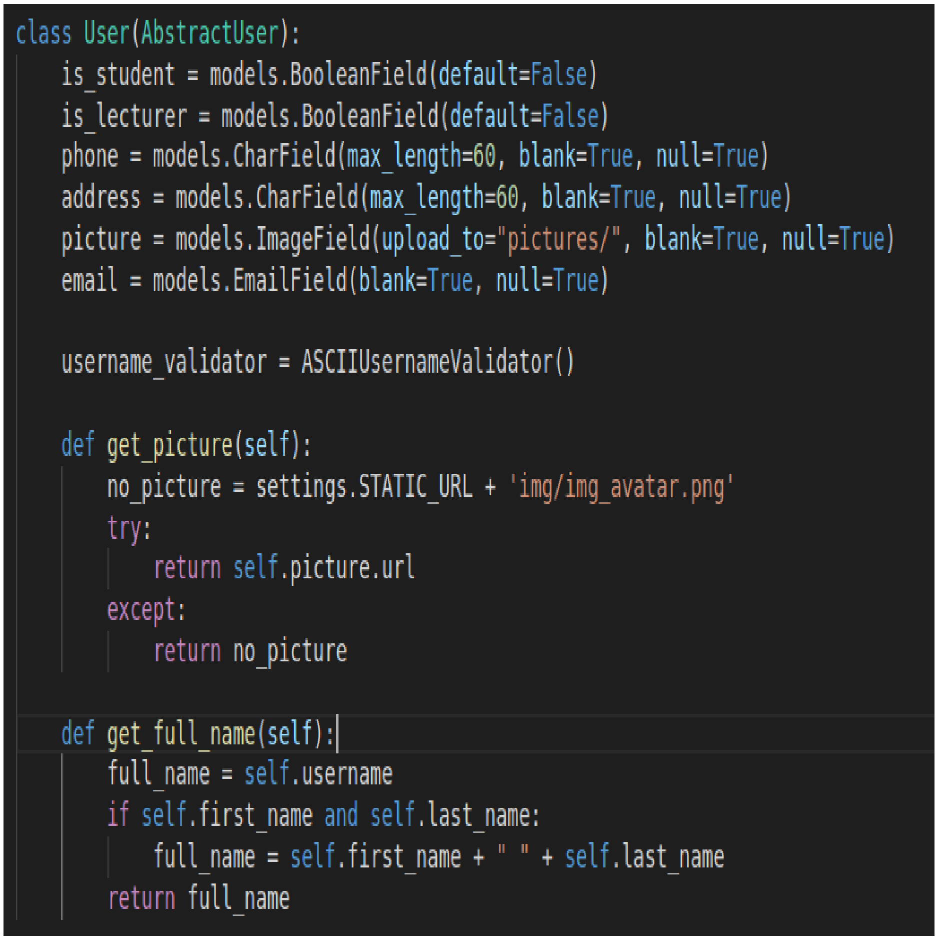

Semester
id AutoField
session ForeignKey (id)
is_current_semester BooleanField
next_semester_begins DateField
semester CharField
CourseAllocation
id AutoField
lecturer ForeignKey (id)
session ForeignKey (id)
TakenCourse
id AutoField
course ForeignKey (id)
student ForeignKey (id)
ca PositiveIntegerField
comment CharField
exam PositiveIntegerField
grade CharField
total PositiveIntegerField
CarryOverStudent
id AutoField
course ForeignKey (id)
student ForeignKey (id)
level CharField
semester CharField
session CharField
RepeatingStudent
id AutoField
student ForeignKey (id)
level CharField
session CharField
Result
id AutoField
student ForeignKey (id)
cgpa FloatField
gpa FloatField
level CharField
semester CharField
session CharField
Session
<AbstractBaseSession>
session_key CharField
expire_date DateTimeField
session_data TextField
abstract
inheritance
AbstractBaseSession
expire_date DateTimeField
session_data TextField
session (semester)
session (courseallocation)
courses (allocated_course)
course (taken_courses)
student (takencourse)
course (carryoverstudent)
student (carryoverstudent)
student (repeatingstudent)
student (result)
LogEntry
id AutoField
content_type ForeignKey (id)
user ForeignKey (id)
action_flag PositiveSmallIntegerField
action_time DateTimeField
change_message TextField
object_id TextField
object_repr CharField
Session
id AutoField
is_current_session BooleanField
next_session_begins DateField
session CharField
lecturer (courseallocation)
Course
id AutoField
courseCode CharField
courseTitle CharField
courseUnit CharField
description TextField
is_elective BooleanField
level CharField
semester CharField
Student
id AutoField
ac_session ForeignKey (id)
department ForeignKey (id)
semester ForeignKey (id)
date_of_birth DateField
email EmailField
guardian_mobile CharField
mobile CharField
name CharField
photo ImageField
registration_number CharField
roll CharField
user (logentry)
user (student)
User
<AbstractUser>
id AutoField
address CharField
date_joined DateTimeField
email EmailField
first_name CharField
is_active BooleanField
is_lecturer BooleanField
is_staff BooleanField
is_student BooleanField
is_superuser BooleanField
last_login DateTimeField
last_name CharField
password CharField
phone CharField
picture ImageField
username CharField
groups (user)
abstract
inheritance
AbstractUser
<AbstractBaseUser,PermissionsMixin>
date_joined DateTimeField
email EmailField
first_name CharField
is_active BooleanField
is_staff BooleanField
is_superuser BooleanField
last_login DateTimeField
last_name CharField
password CharField
username CharField
content_type (logentry)
user_permissions (user)
Group
id AutoField
name CharField
permissions (group)
abstract
inheritance
abstract
inheritance
PermissionsMixin
AbstractBaseUser
Permission
id AutoField
content_type ForeignKey (id)
codename CharField
name CharField
content_type (permission)
ContentType
id AutoField
app_label CharField
model CharField

3.6 Use Case Diagrams


IIT BHU Management system
Add update Hostel information
Hostel Information
Student Information
Teacher/Staff Information
Update Student or Teacher Information
Delete Student/Teacher Information
Teacher Information
View Student Record
Keep Student Acadamic Record
Evaluate Student Performance
Generating Reports/Notices
Change Login password
View Co-teaching Staff
View Student Performance/Grades
View Student Profile or Information
Change Login password
Admin
Teacher
Student

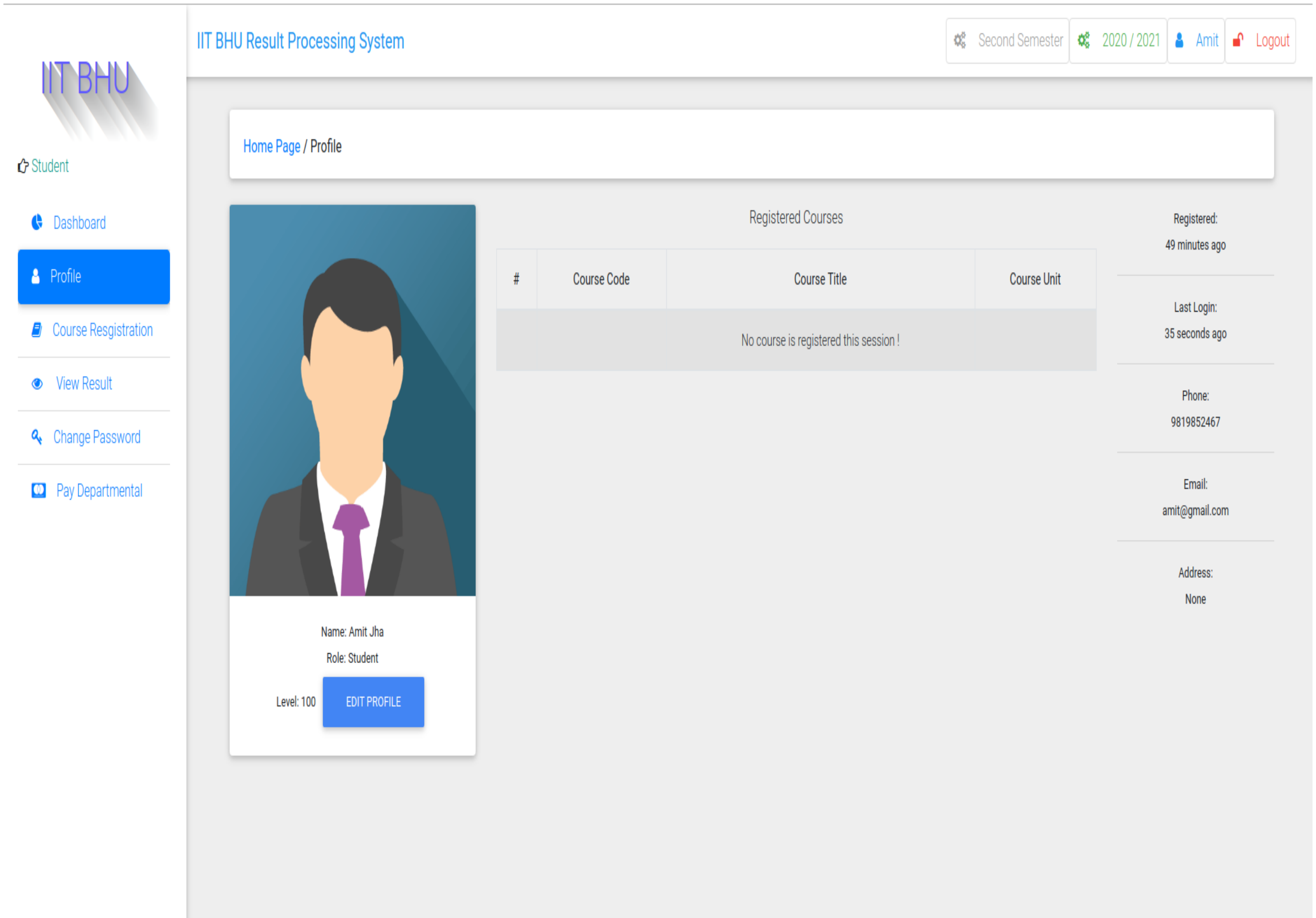
4 System Implementation

I have used django framework for the implementation and Coding purpose.Note that the default user model in django has been extended to create all other users.



```
class User(AbstractUser):
    is_student = models.BooleanField(default=False)
    is_lecturer = models.BooleanField(default=False)
    phone = models.CharField(max_length=60, blank=True, null=True)
    address = models.CharField(max_length=60, blank=True, null=True)
    picture = models.ImageField(upload_to="pictures/", blank=True, null=True)
    email = models.EmailField(blank=True, null=True)

    username_validator = ASCIIUsernameValidator()

    def get_picture(self):
        no_picture = settings.STATIC_URL + 'img/img_avatar.png'
        try:
            return self.picture.url
        except:
            return no_picture

    def get_full_name(self):
        full_name = self.username
        if self.first_name and self.last_name:
            full_name = self.first_name + " " + self.last_name
        return full_name
```

4.1 Student



IIT BHU Result Processing System

Home Page / Profile

Name: Amit Jha

Role: Student

Level: 100

Registered Courses

#	Course Code	Course Title	Course Unit
No course is registered this session !			

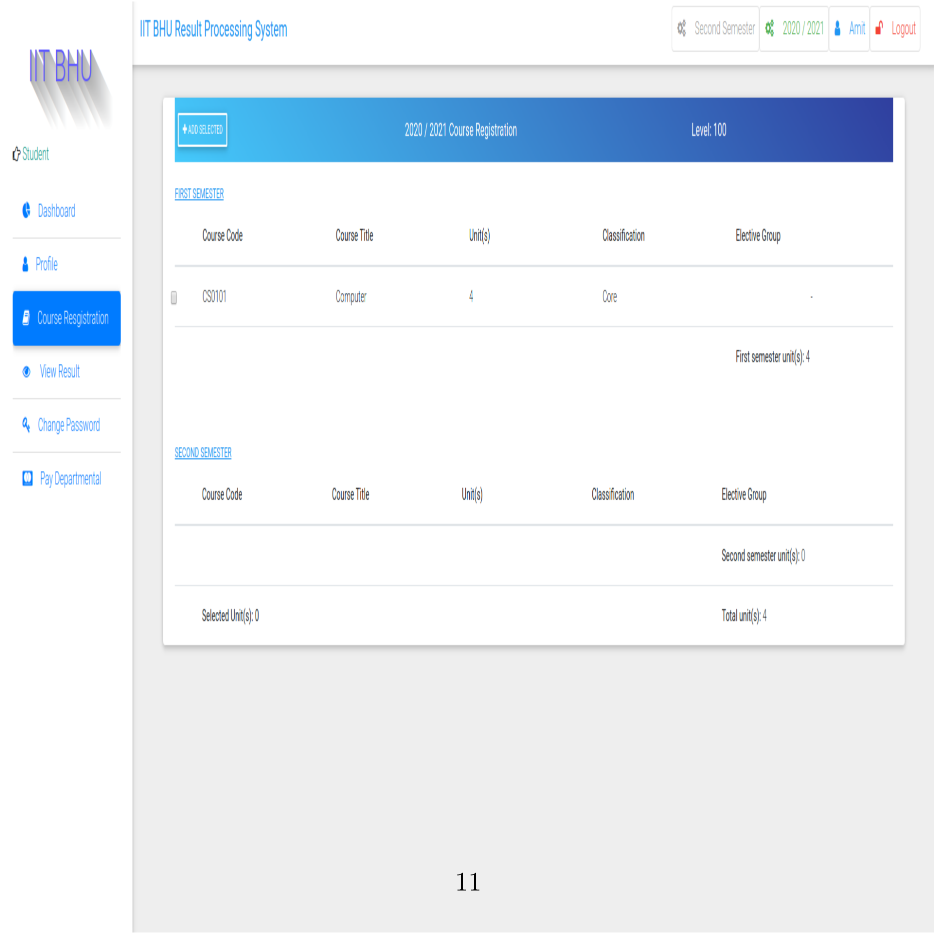
Registered:
49 minutes ago

Last Login:
35 seconds ago

Phone:
9819852467

Email:
amit@gmail.com

Address:
None



IIT BHU Result Processing System

2020 / 2021 Course Registration

Level: 100

FIRST SEMESTER

	Course Code	Course Title	Unit(s)	Classification	Elective Group
□	CS0101	Computer	4	Core	-

First semester unit(s): 4

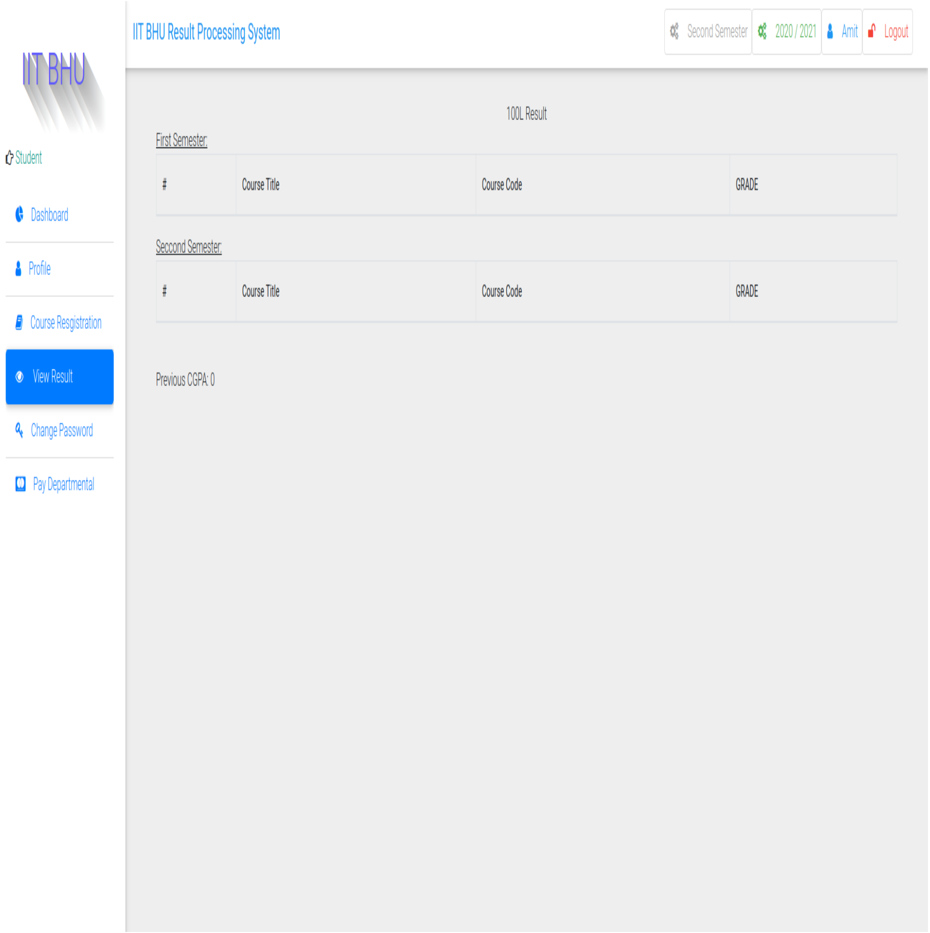
SECOND SEMESTER

Course Code	Course Title	Unit(s)	Classification	Elective Group

Second semester unit(s): 0

Selected Unit(s): 0

Total unit(s): 4



100L Result

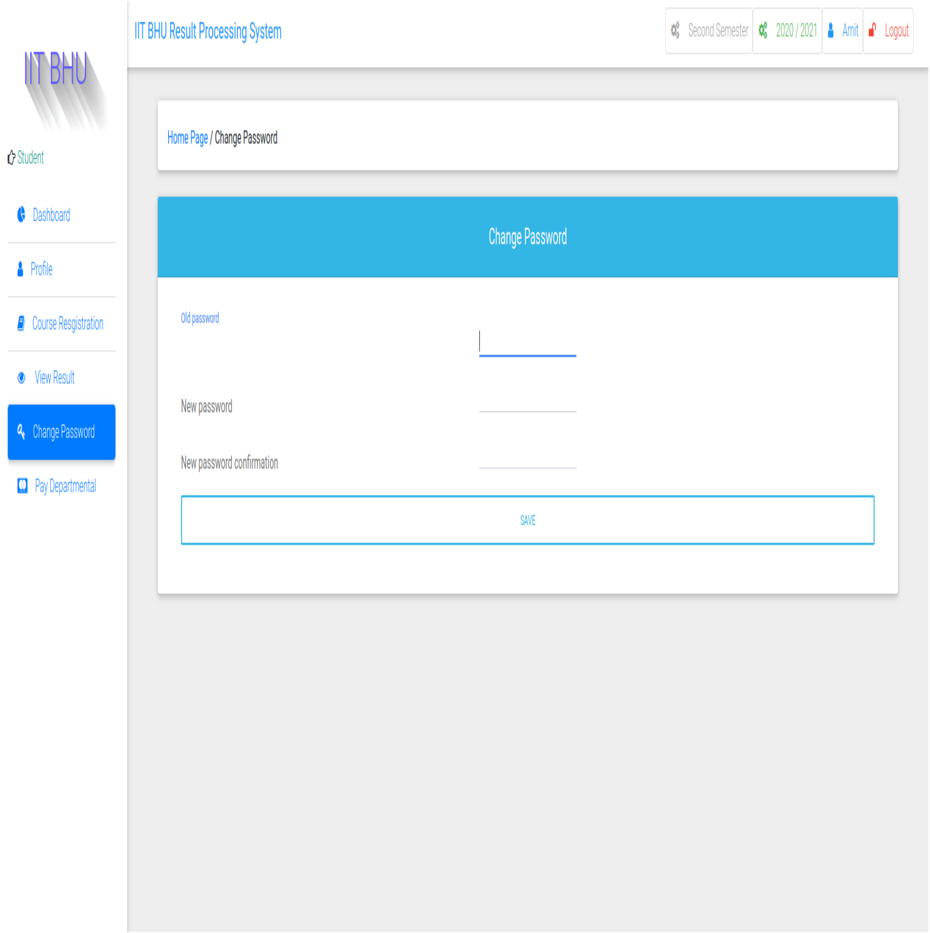
First Semester:

#	Course Title	Course Code	GRADE

Seccond Semester:

#	Course Title	Course Code	GRADE

Previous CGPA: 0



IIT BHU Result Processing System

Home Page / Change Password

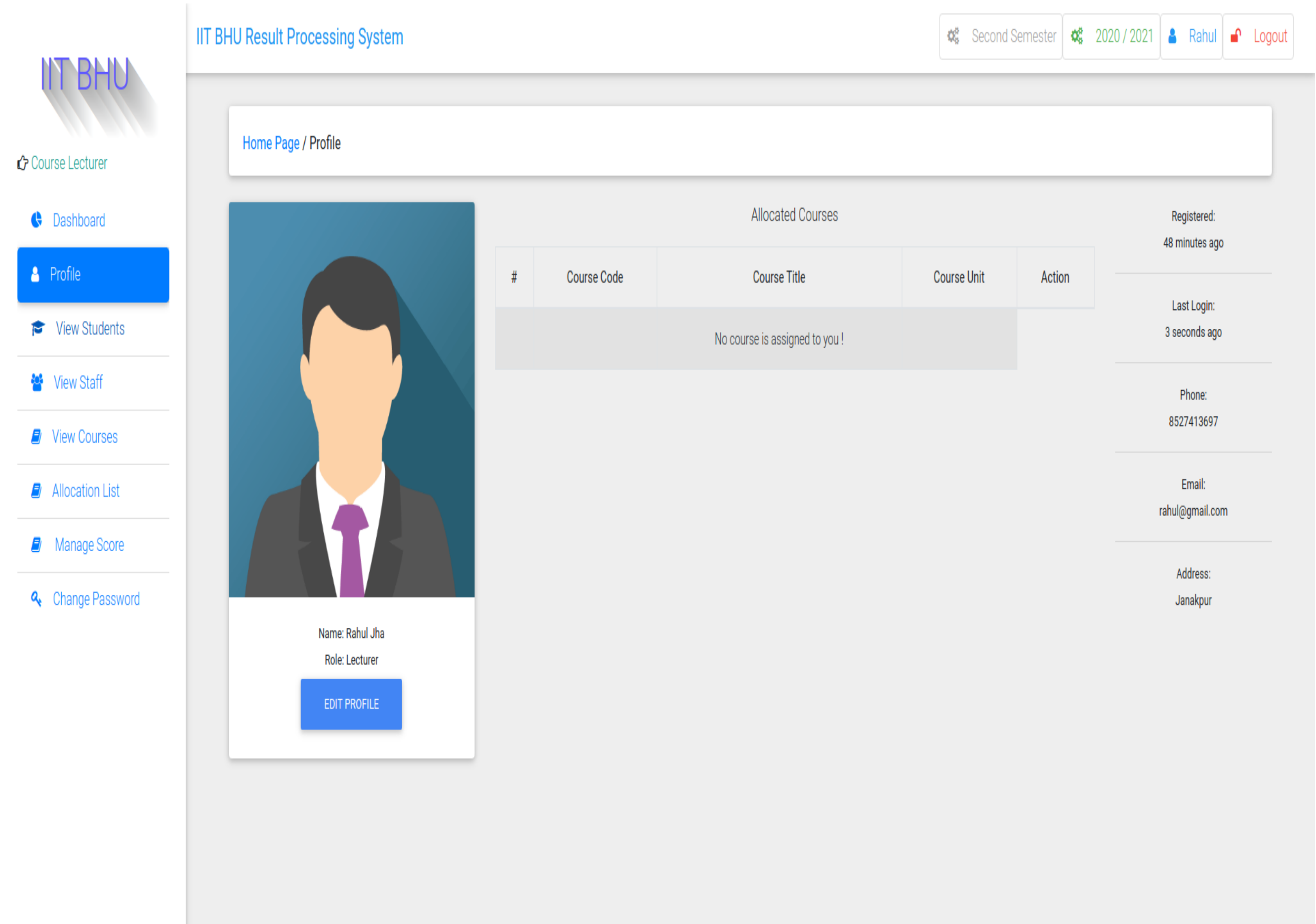
Change Password

Old password

New password

New password confirmation

4.2 Teachers



Allocated Courses

#	Course Code	Course Title	Course Unit	Action
No course is assigned to you !				

Name: Rahul Jha
Role: Lecturer

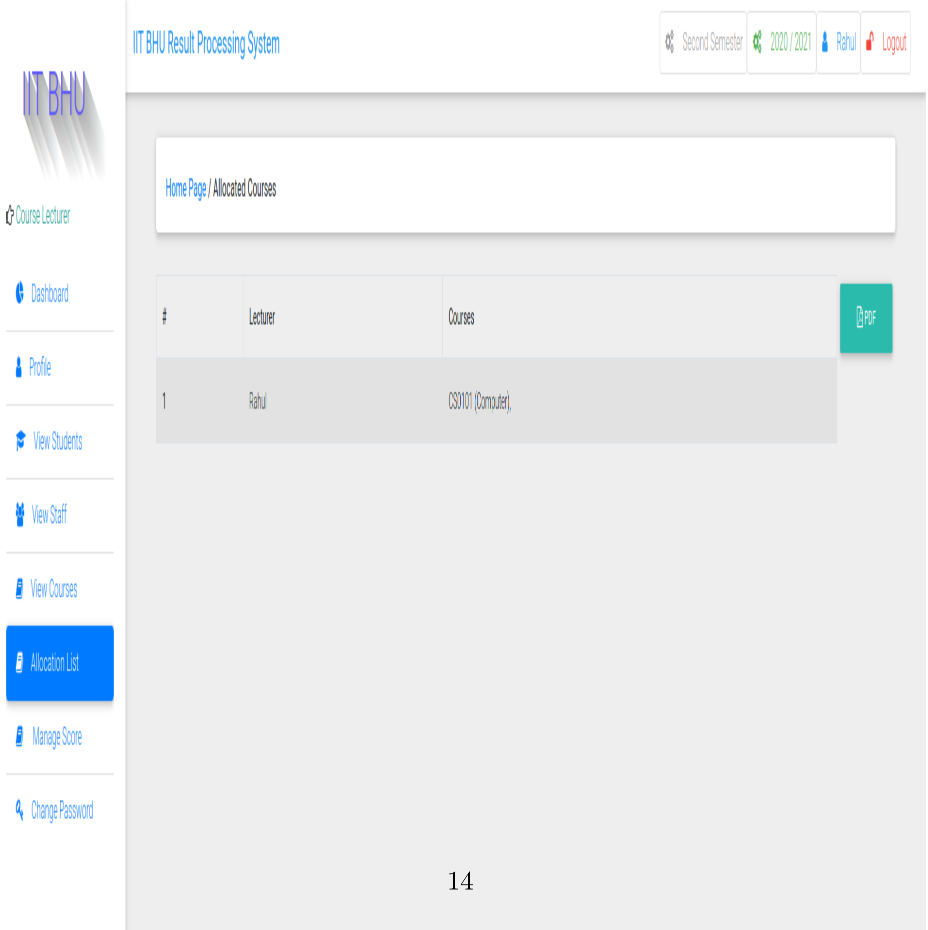
Registered:
48 minutes ago

Last Login:
3 seconds ago

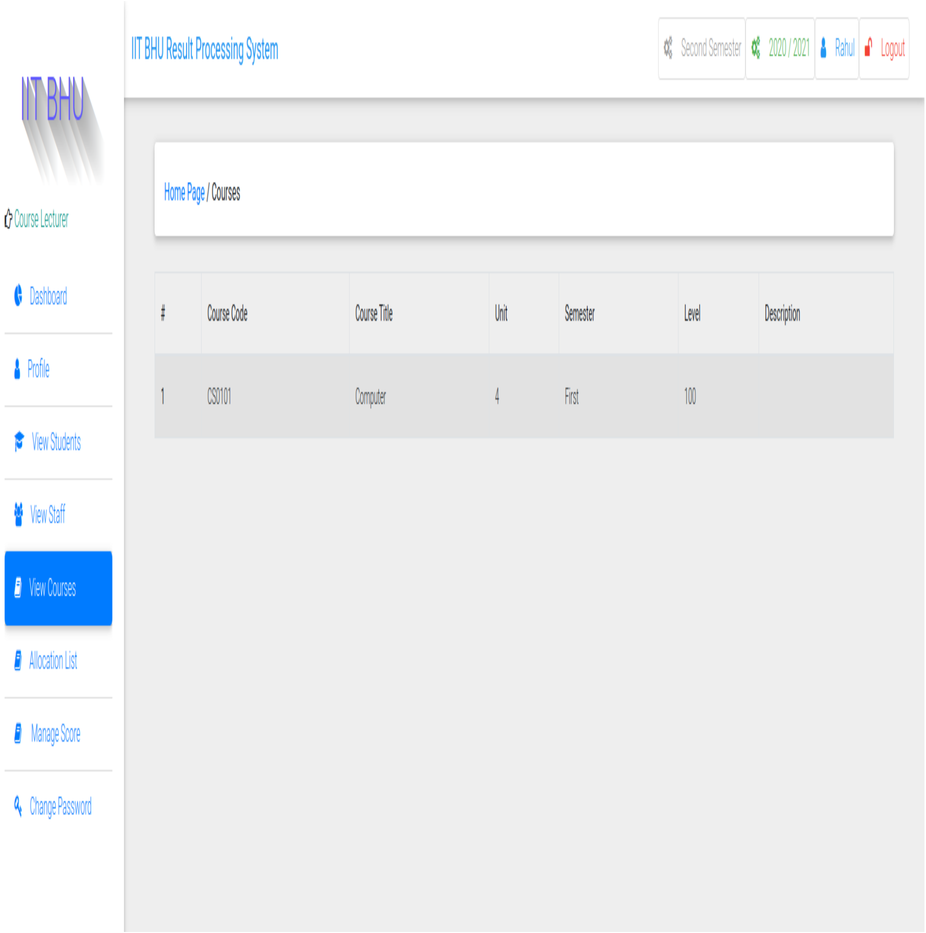
Phone:
8527413697

Email:
rahul@gmail.com

Address:
Janakpur



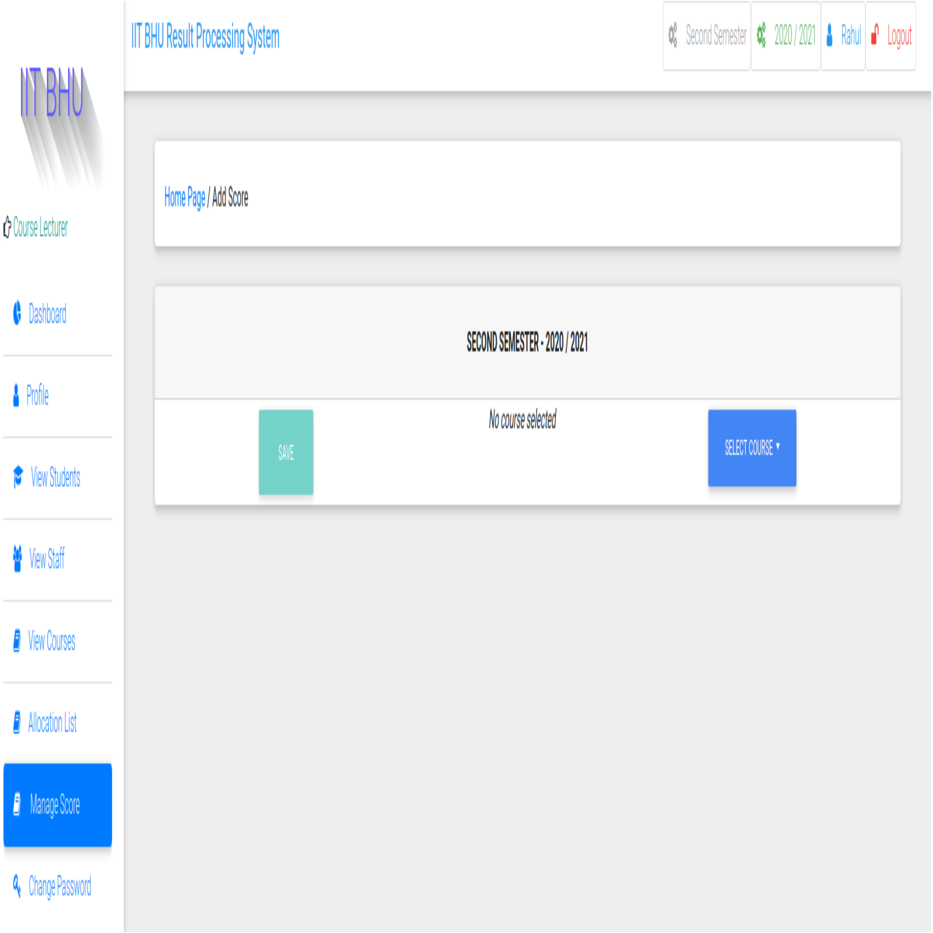
#	Lecturer	Courses
1	Rahul	CS0101 (Computer),



IIT BHU Result Processing System

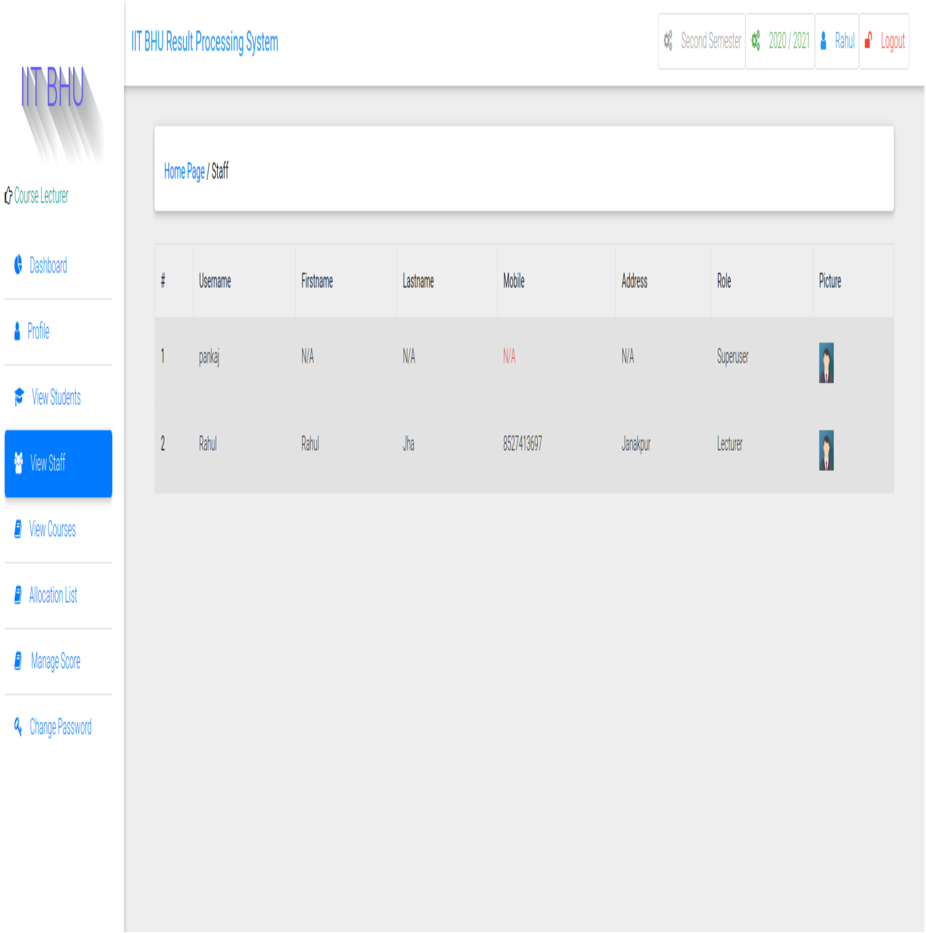
Home Page / Courses

#	Course Code	Course Title	Unit	Semester	Level	Description
1	CS0101	Computer	4	First	100	



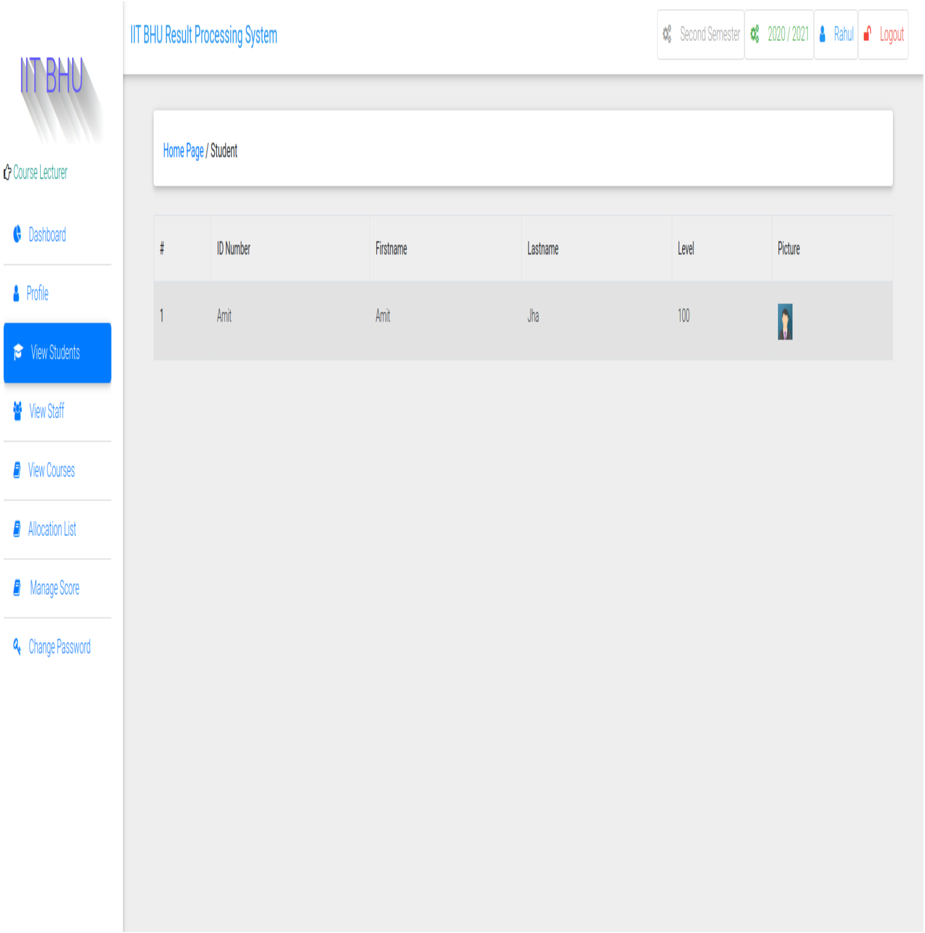
SECOND SEMESTER - 2020 / 2021

No course selected



Home Page / Staff

#	Username	Firstname	Lastname	Mobile	Address	Role	Picture
1	pankaj	N/A	N/A	N/A	N/A	Superuser	
2	Rahul	Rahul	Jha	8527413697	Janakpur	Lecturer	



#	ID Number	Firstname	Lastname	Level	Picture
1	Amit	Amit	Jha	100	

4.3 Administrator

The Adminstration has two different admin pages. One is default which is auto generated by django framework which has different tables. One More Interface has been created with which admin will be able to add , delete update courses, allocations, teachers , staffs, students.

Figure 1: Admin Home page

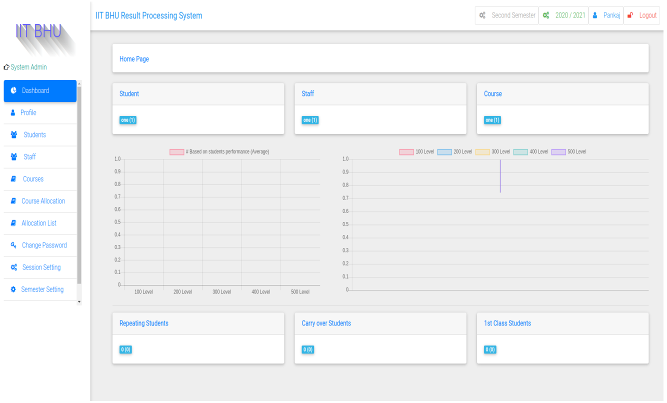
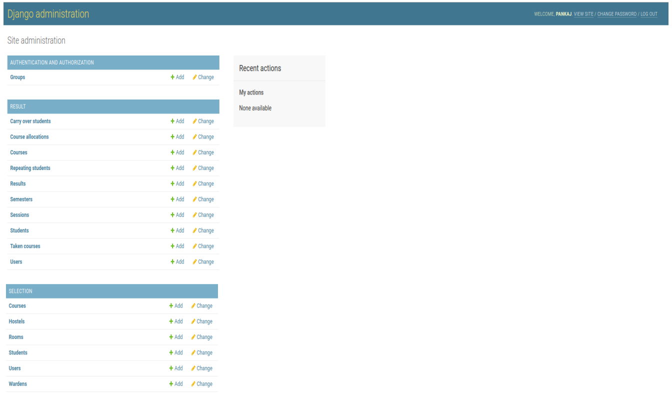


Figure 2: Django Admin



5 System Testing

The system is complete only after it has been thoroughly tested. Though this gives a feel the project is completed, there cannot be any project without going through this stage. Hence in this stage it is decided whether the project can undergo the real time environment execution without any break downs, therefore a package can be rejected even at this stage.

5.1 Testing Methods

Software testing methods are divided into black box testing and white box testing. These two approaches are used to describe the point of view that a testing engineer takes when designing test cases.

5.1.1 White Box testing

White box testing, by contrast to black box testing, is when the tester has access to the internal data structures and algorithms. White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested.

This project is implemented using Python with the Django framework. The code consists of models and views which can be tested. Models define the tables stored in SQL and the relationship between the different tables using foreign keys. A view function, or "view" for short, is simply a Python function that takes a web request and returns a web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image, etc.

Django also provides a file called test.py where we can write unit tests for the models and views. This automates the testing and we no longer have to manually test every page after there were any changes.

5.1.2 Black Box testing

Black box testing treats the software as a "black box," without any knowledge of internal implementation / code. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.
I performed black box testing on the teacher page to make sure every page was working as desired. I took into consideration various test cases and noted down the results. Below I have recorded various test cases and their respective results

- **Request the attendance page for a teacher with no assigned classes.**

  The web page loaded with the message "Teacher has no classes assigned".

- **Request the attendance page for a teacher with 1 assigned class.**

  The web page displayed the assigned class and options to enter attendance and view the students

- **Request to enter the attendance for an assigned class with one test student**

  The web page displays the student with his/her details and an option to mark present or absent. On marking absent, it can be viewed by the student.

- **Request to edit the attendance for an assigned class with one test student**

  The student is listed with his/her details and is initially marked as absent from the previous test. On marking present, the attendance for that student can be viewed by the student.

- **Request to enter the marks for an assigned class with one student**

  Initially, a list of tests is displayed such as internals 1, end semester examination etc. On selecting one of internals 1, the teacher can enter the marks for the student out of 20. On submitting, the status for that test turns green denoting that it has been successfully entered.

- **Request to edit the marks for an assigned class with one student**

  For each class, there is a list of tests such as internals 1, SEE etc. As the marks for internals 1 was already entered in the previous test, it is marked green and there is an option to edit. When editing, the marks already stored are displayed and appropriate changes can be made and saved.

- **Request to view the student information for an assigned class with no students**

  The requested page is display with no content and a message stating ”This class has no students assigned”

- **Request to view the student information for an assigned class with 1 student**

  The web page is the form of a table with entries for student name, username and their attendance percentage, marks in each test including 3 internals, 2 events and 1 end semester. If the attendance status is below 75 percent, it is marked in red.

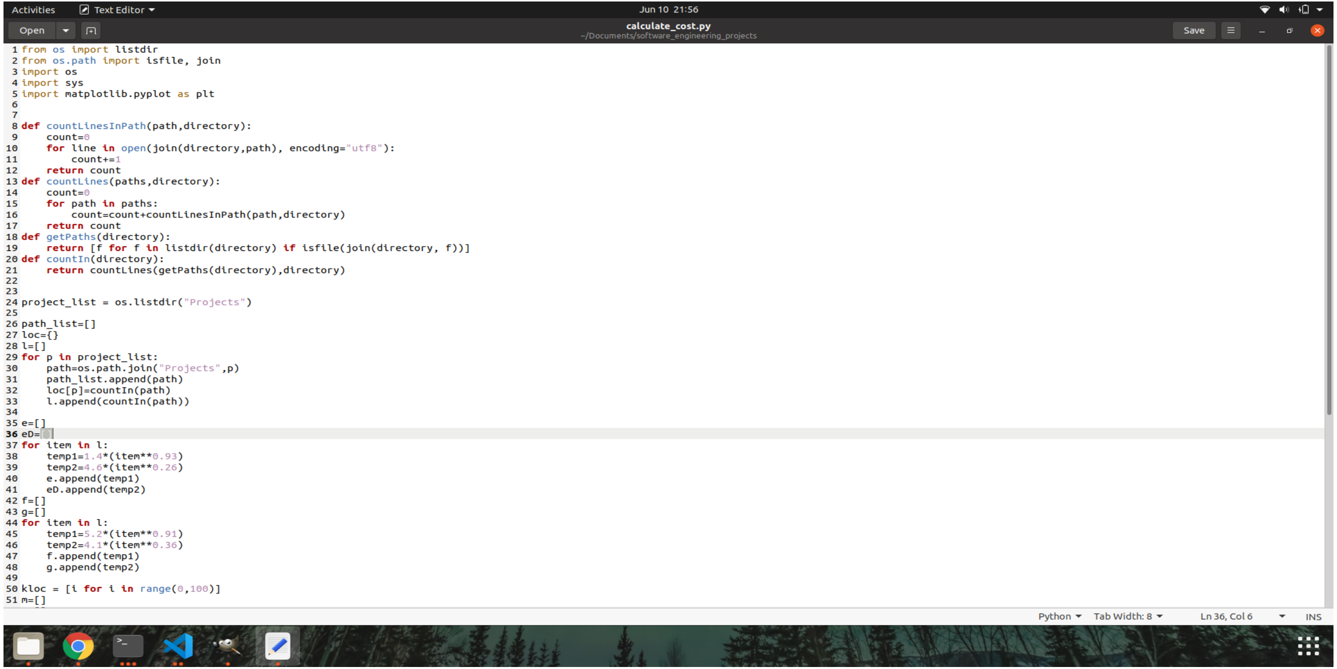
5.1.3 Acceptance Testing

Acceptance testing performed by the customer is known as user acceptance testing (UAT). Since our project is on a university management system, the teachers are a key stakeholder. Hence, it was important that I allow the teachers to test the software and get their approval as they intend to use the software the most. Therefore, I gave a demonstration of the project to our mentor Dr. Amit Biswas. I showed all the features and functionality of the website. He went through all the different web pages.

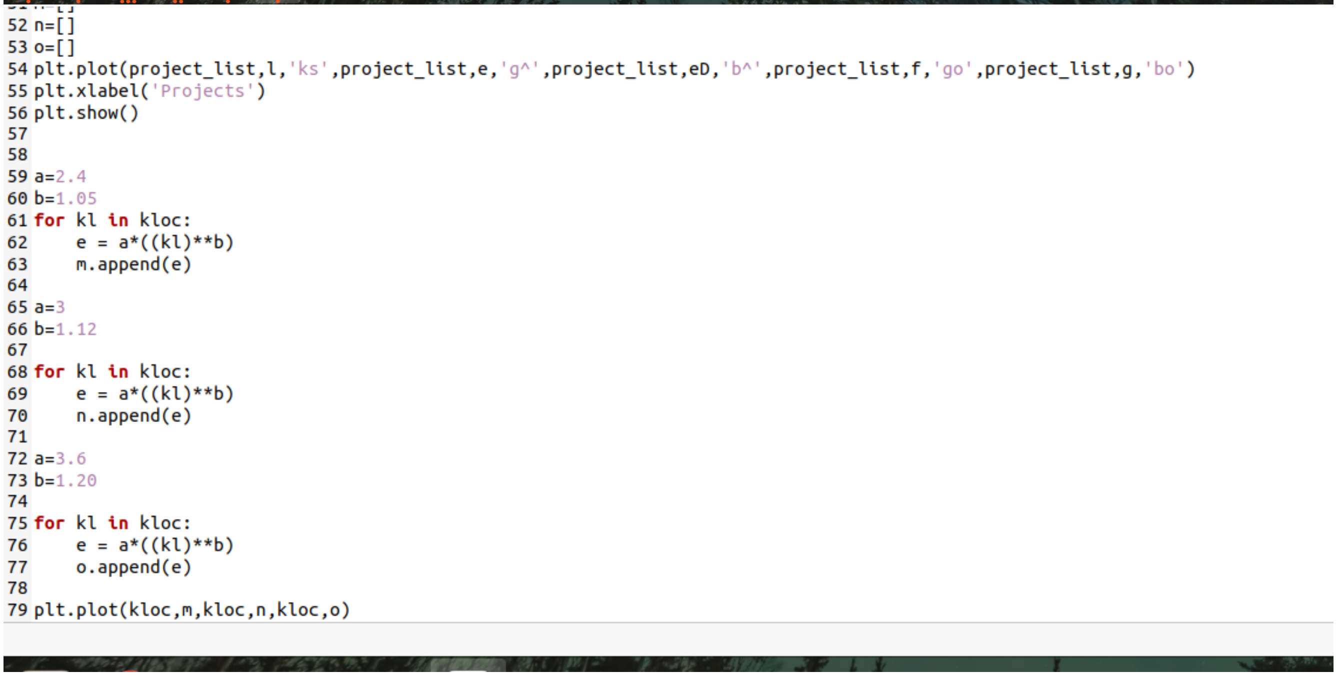
6 Cost Estimation

We had implemented a separate model for cost estimation which estimates the cost of project based on the lines of Code.Python programming Language was used to implement it.

Code

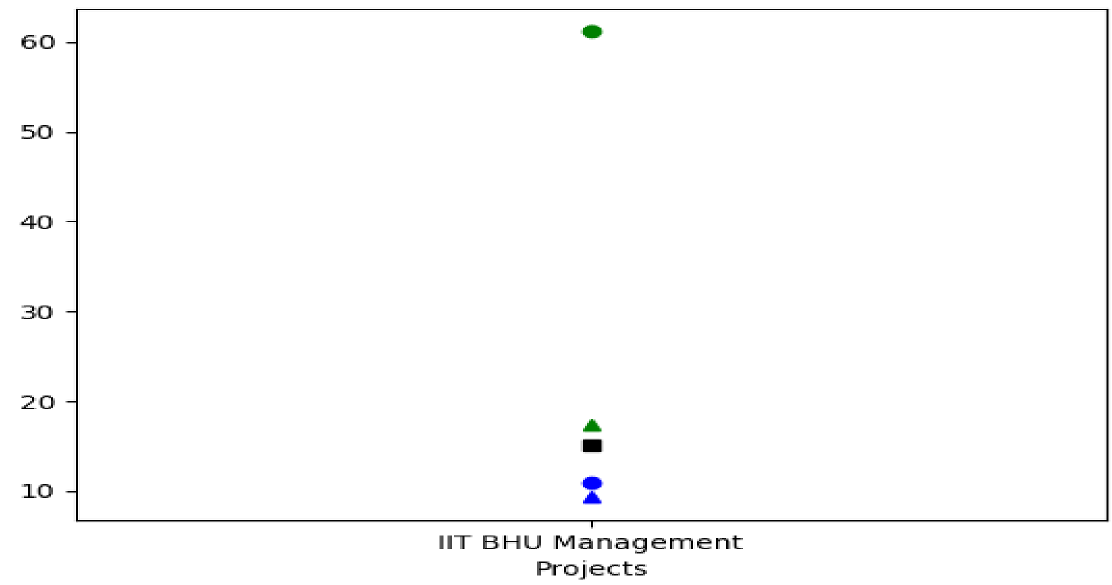


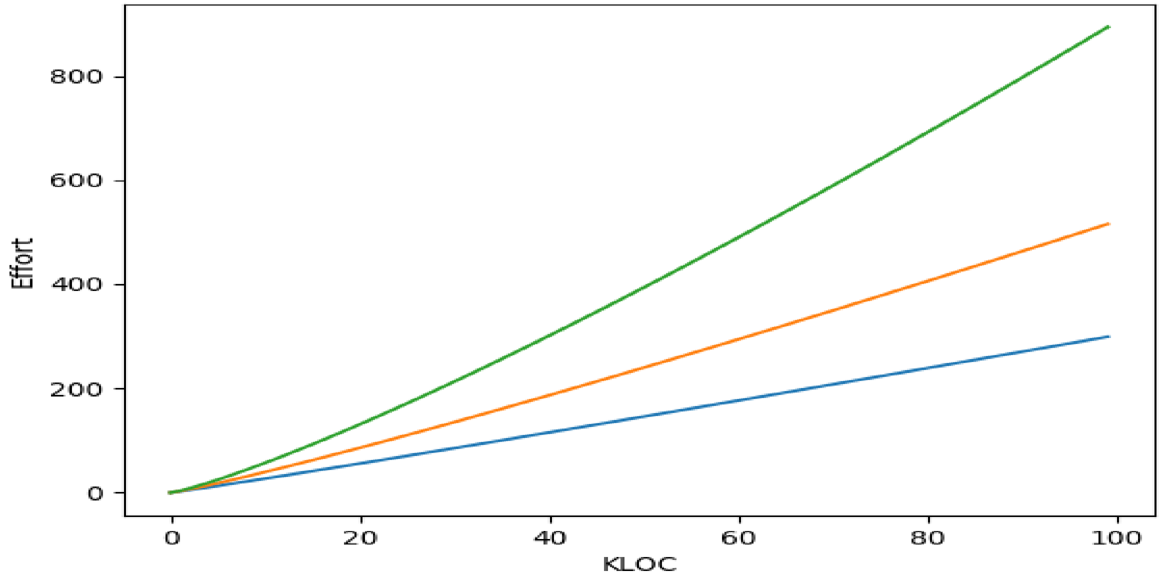
```python
from os import listdir
from os.path import isfile, join
import os
import sys
import matplotlib.pyplot as plt


def countLinesInPath(path,directory):
    count=0
    for line in open(join(directory,path), encoding="utf8"):
        count+=1
    return count
def countLines(paths,directory):
    count=0
    for path in paths:
        count=count+countLinesInPath(path,directory)
    return count
def getPaths(directory):
    return [f for f in listdir(directory) if isfile(join(directory, f))]
def countIn(directory):
    return countLines(getPaths(directory),directory)


project_list = os.listdir("Projects")

path_list=[]
loc={}
l=[]
for p in project_list:
    path=os.path.join("Projects",p)
    path_list.append(path)
    loc[p]=countIn(path)
    l.append(countIn(path))

e=[]
eD=[]
for item in l:
    temp1=1.4*(item**0.93)
    temp2=4.6*(item**0.26)
    e.append(temp1)
    eD.append(temp2)
f=[]
g=[]
for item in l:
    temp1=5.2*(item**0.91)
    temp2=4.1*(item**0.36)
    f.append(temp1)
    g.append(temp2)

kloc = [i for i in range(0,100)]
m=[]
```



```python
n=[]
o=[]
plt.plot(project_list,l,'ks',project_list,e,'g^',project_list,eD,'b^',project_list,f,'go',project_list,g,'bo')
plt.xlabel('Projects')
plt.show()


a=2.4
b=1.05
for kl in kloc:
    e = a*((kl)**b)
    m.append(e)

a=3
b=1.12

for kl in kloc:
    e = a*((kl)**b)
    n.append(e)

a=3.6
b=1.20

for kl in kloc:
    e = a*((kl)**b)
    o.append(e)

plt.plot(kloc,m,kloc,n,kloc,o)
```

Cost Estimation Plots


60
50
40
30
20
10
IIT BHU Management
Projects


800
600
400
200
0
Effort
0
20
40
60
80
100
KLOC

7 Conclusion

I have build the web application that can certainly ease the process of registration of students , allocation of hostel and management of staff. I have tried my best to meet the requirements discussed in the specification. No manual Labour will be required in the process.However there is still space left to optimize the software using techniques like database normalization.