NIMESH BHATTARAI

NEO-ATTENDANCE

TEXAS COLLEGE OF MANAGEMENT & IT

DEPARTMENT OF INFORMATION TECHNOLOGY

LINCOLN UNIVERSITY COLLEGE

September 09th, 2022

Submitted by:	Submitted to:
Nimesh Bhattarai	Jayaram Pudasaini
Faculty: BIT	
Year/Sem: 3rd/5th	
LCID: LC00017000791	

Table of Contents

List of Figures

ACKNOWLEDGEMENT

I am grateful to Texas College of Management & IT for providing me this opportunity. I am also glad to provide my academic capabilities to my college. This project had taught me a lot regarding my subject matter. It also had led me towards the professional project plan, management and handling lessons. I also would like to thank more to my project supervisor Mr. Jayaram Pudasaini for enduring my steep mistakes. I would finally like to thanks all my friends who had provided their images for completion of this project and special thanks to the supervisor and college for accepting my project.

Mr. Nimesh Bhattarai

Form Of Declaration

I confirm that the enclosed written work along with my application code is entirely my own. I also declare that wherever I had copied, paraphrased, summarized by providing full credit to the respective author. More than this if any other material, word, sentences found will be just a mistake. I had tried to be full open and honest.

ABSTRACT

Manual attendance is a more time-consuming task. Attendance sheet could be lost or misplaced by anyone. In this roll call system, present student can answer the roll call of absent students. There is another practice also where each student signs the individual attendance sheet, this practice allows present student in the class to sign physically absent students.

Rapid growing technology has widened the scope of attendance system. When the attendance system becomes digital it would reduce the burden for teachers, students. The digital attendance smoothens the lecture hours. Digital attendance is more accurate than a manual attendance system. This report also highlights the one kind of digital attendance system. The system named as "Neo-Attendance". It is the face recognition attendance system which automatically marks the attendance of the present students in the class. It is a modified attendance system compared to the manual attendance system. It uses artificial intelligence, machine learning and deep learning concepts. By integrating these concepts laborious tasks can be transformed to smart task.

This report would be beneficial for individuals and organization interested to work in artificial intelligence and web development. With the help of Python programming and its Django framework I will be introducing one suitable website which deals with regular taking attendance. Schools, Colleges, University, Offices, Banks who wants to develop a modern attendance system, this project can be taken as an example.

CHAPTER 1: SYSTEM INTRODUCTION

1.1 Introduction

Neo-Attendance is the face recognition attendance system which automatically marks the attendance of the present students. Face recognition attendance can be applied in many fields like education, banking, home and so on. Neo-Attendance is mainly focused on education. With this project many problems in classroom can be solved. Teacher can give fulltime in teaching. Student can focus on their studies by listening lectures. Concerned members can access student records at any time, any place.

Face recognition feature is applied on the student registration system in Neo-Attendance. At the beginning student records are required to use face recognition feature. Student registration is a must to use the face recognition. Student registration provides the images of the students which are used during face recognition. Finally, the system generates the spreadsheet file with student attendance recorded.

1.2 Proposed system for the attendance problem

The automated system helps in increasing the accuracy and speed of task execution in real-time. The aim of this proposed system is to capture the video of the students and mark the attendance. Each captured face in the image of the video is compared with the record present in the database and marks the attendance of the present student. The problem in attendance system done in school, colleges time-consuming, less accuracy rate of marking correct mark on the attendance sheet. Neo-Attendance mark the attendance based on student image records present in the system. Bias will not occur during attendance marking. This system can do laborious tasks in a smart and efficient way.

1.3 Project Scope And Objective

Attendance marking is a time-consuming task during the lecture hours. In a classroom, marking the attendance for a large number of students is challenging. With the development of the technology, attendance marking techniques are also changing. From manual marking to biometric techniques using fingerprint, frequency identification tags and so on. Thought stability has not been achieved in the attendance system. In the proposed Neo-Attendance project automated attendance marking is done by using face detection algorithm and recognition algorithm.

Scope of the project are as follows:

i. Input for the system will be video. Image processing is done in frames of video.
ii. Automate attendance of student can be done.
iii. System uses live face recognition to recognize each present student and mark their attendance automatically.

The proposed system automatically records the attendance of the students by using face recognition technology. System will recognize the face using face recognition algorithm. The processed image will be compared with the images stored in the database. According to the matched images attendance will be marked accordingly. This face recognition system reduces the workload of the teacher. Image capturing, face detection, face comparison and face recognition, updating of attendance in the database in the scope of the project.

The objective of this project is to make the attendance system efficient, time saving, simple and easy. To get this objective Neo-Attendance is categorized into 3 phases:

a. Login System

   Only the authenticated users can enter into the Neo-Attendance System.

b. Student Registration System

   Data of students are registered through the registration system. During registration images and other details of students are taken and stored in database. This image helps to recognize the student during attendance. Google form is used to acquire the image of the students which can be used during student registration.

c. Face Recognition System

Face recognition system performs tasks in component manners as image acquisition, face detection, face recognition and attendance marking.

a. Image Acquisition

The camera helps to acquire the image for the system. Image quality and quantity also affects the performance of the system. Highly blurred images effect the algorithm to detect the face of the students in the image frame. High quality takes huge time to process so it degrades the performance speed and efficiency might not be achieved.

b. Face Detection

Each face in the frame is detected using a modeling algorithm available in OpenCV.

c. Face Recognition

Face detected in each frame is compared with the faces stored in the database.

d. Attendance Marking

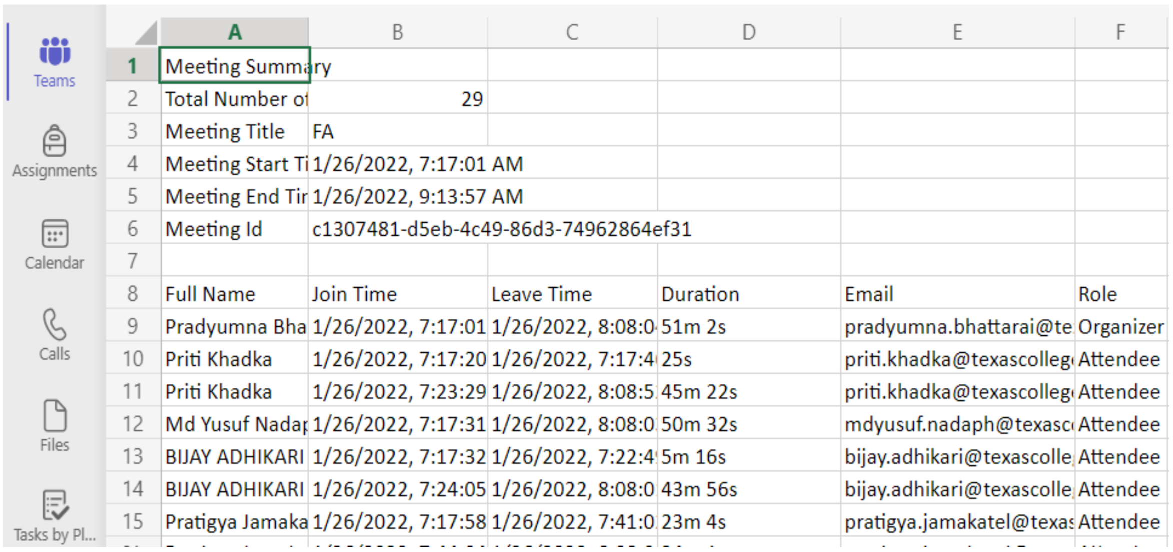
After recognizing the student's face, attendance is maintained.

CHAPTER 2: LITERATURE REVIEW

2.1 Background/ History

This project is about the development of artificial intelligence integrated attendance system called “Neo-Attendance”. This system registers the students and compare registered images in database with face recognition algorithm to check whether student is present during class hours. Though attendance system is the normal regular activities done in school, colleges and so on. Marking attendance of huge number of students is time consuming and full of errors. To reduce the time and increase the accuracy of the attendance system, shifts to the digital platform.

Microsoft Teams is the widely used and popular remote classroom application.



	A	B	C	D	E	F
1	Meeting Summary					
2	Total Number of	29				
3	Meeting Title	FA				
4	Meeting Start Ti	1/26/2022, 7:17:01 AM				
5	Meeting End Tir	1/26/2022, 9:13:57 AM				
6	Meeting Id	c1307481-d5eb-4c49-86d3-74962864ef31				
7						
8	Full Name	Join Time	Leave Time	Duration	Email	Role
9	Pradyumna Bha	1/26/2022, 7:17:01	1/26/2022, 8:08:0	51m 2s	pradyumna.bhattarai@te	Organizer
10	Priti Khadka	1/26/2022, 7:17:20	1/26/2022, 7:17:4	25s	priti.khadka@texascolleg	Attendee
11	Priti Khadka	1/26/2022, 7:23:29	1/26/2022, 8:08:5	45m 22s	priti.khadka@texascolleg	Attendee
12	Md Yusuf Nadap	1/26/2022, 7:17:31	1/26/2022, 8:08:0	50m 32s	mdyusuf.nadaph@texasc	Attendee
13	BIJAY ADHIKARI	1/26/2022, 7:17:32	1/26/2022, 7:22:4	5m 16s	bijay.adhikari@texascolle	Attendee
14	BIJAY ADHIKARI	1/26/2022, 7:24:05	1/26/2022, 8:08:0	43m 56s	bijay.adhikari@texascolle	Attendee
15	Pratigya Jamaka	1/26/2022, 7:17:58	1/26/2022, 7:41:0	23m 4s	pratigya.jamakatel@texas	Attendee

Figure 1: Microsoft Teams Attendance File

In this application organizer or teacher can take attendance of student during the session. There are many other education platforms where attendance is marked digitally.

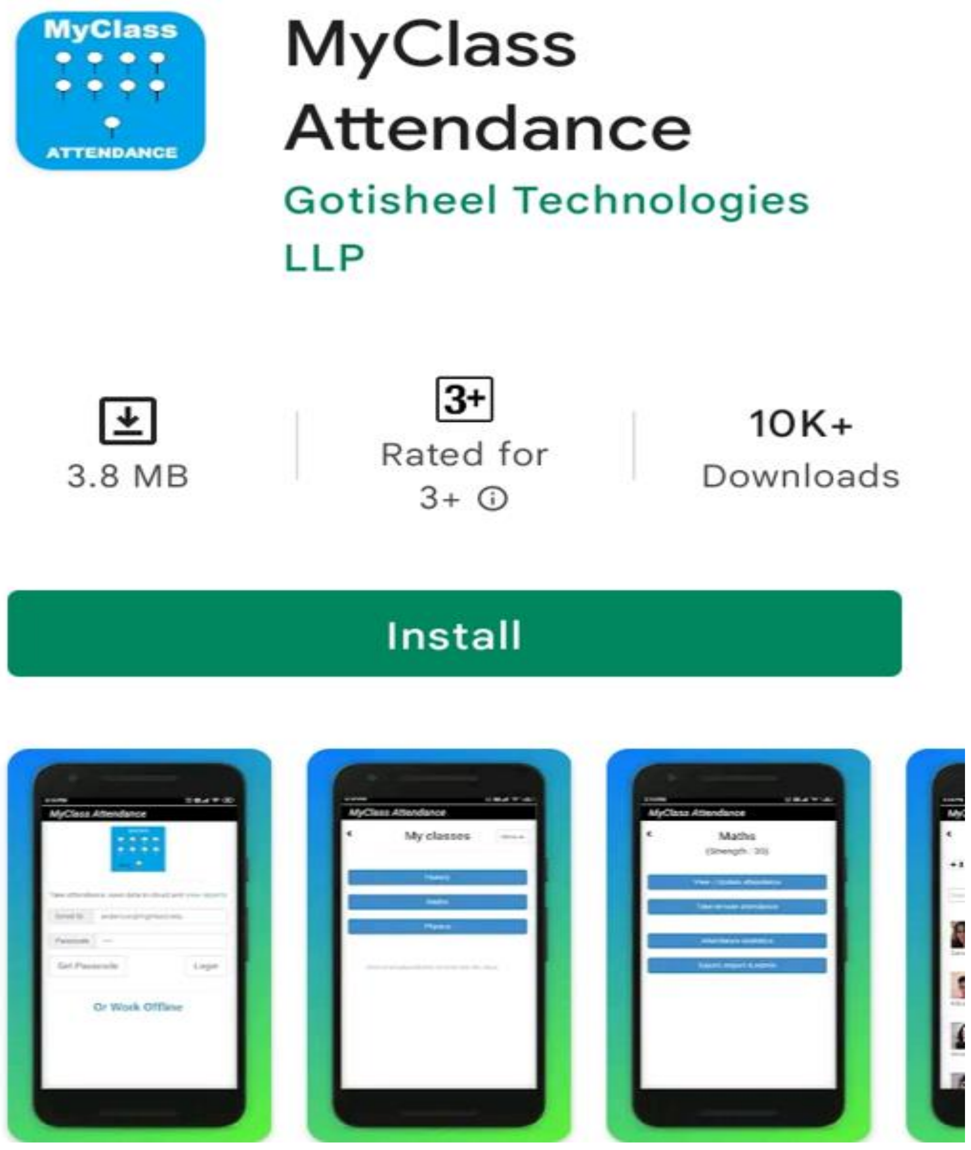


Figure 2: Existing attendance app

"MyClass Attendance" is the existing system available in "play store". This system is smart than manual physical system. Using this system teacher can take attendance of student while having physical class. This system also has cons. In the roll call system, present students can answer the roll call of absent students.

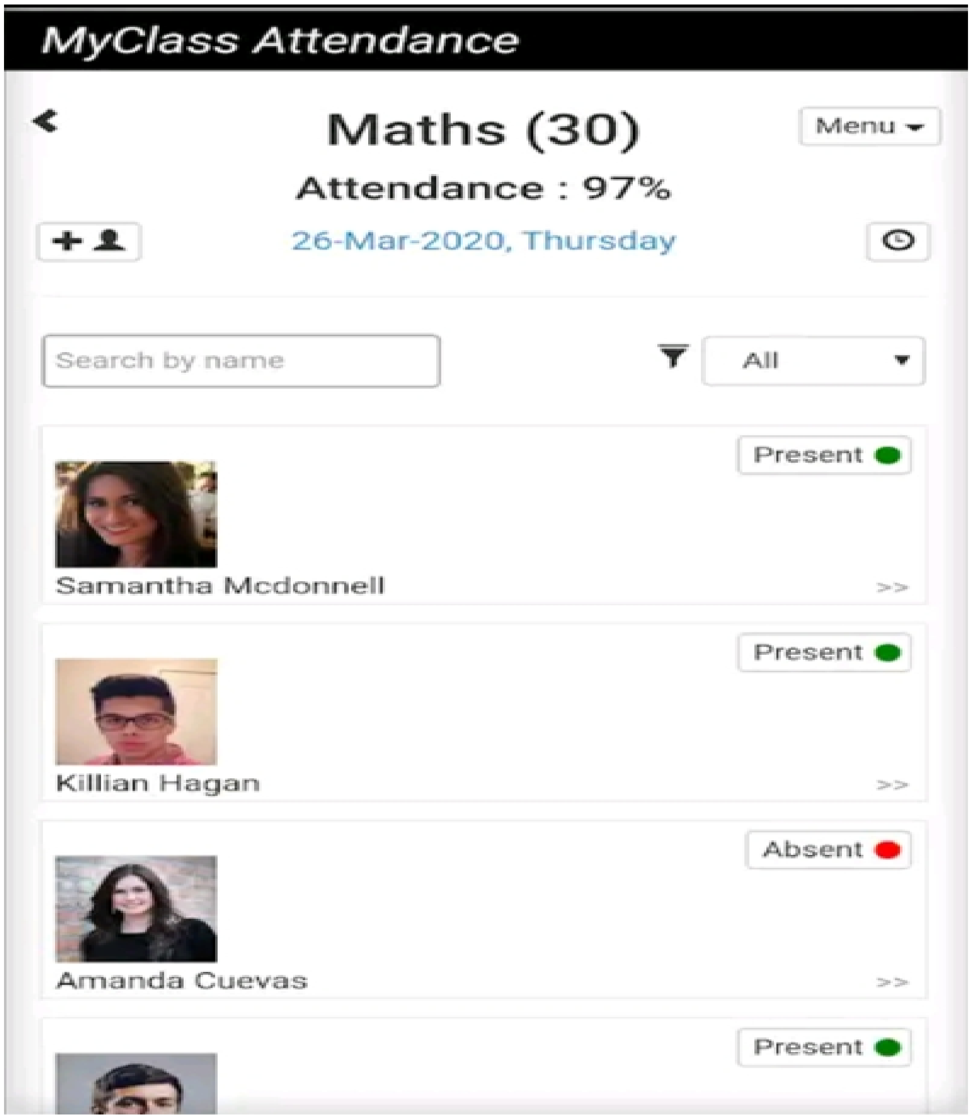


Figure 3: Online Attendance

Comparing to Neo-Attendance this system is not smarter. The main cons of "MyClass Attendance" can be resolved using Neo-Attendance. Attendance is taken using frames of video or image. The automated classroom attendance system helps in increasing the accuracy and speed of taking real-time attendance.

2.2 Development methodology to be implemented

The Iterative waterfall methodology is proposed for the development of Neo-Attendance. Phases in software development life cycle (SDLC):

1. Planning

   Image collection, resource allocation for the project is considered. Google form is used to collect student details.
2. Analysis

   Requirement analysis like hardware and software compatibility are analyzed.
3. Design

   As per requirement specification, login system design, registration form design is constructed.
4. Coding

   Face detection algorithm, face recognition algorithm and other required coding is done here.
5. Testing

   Unit test, integration test and system test are done during the testing phase.
6. Maintenance

   After development, time to time maintenance is required.

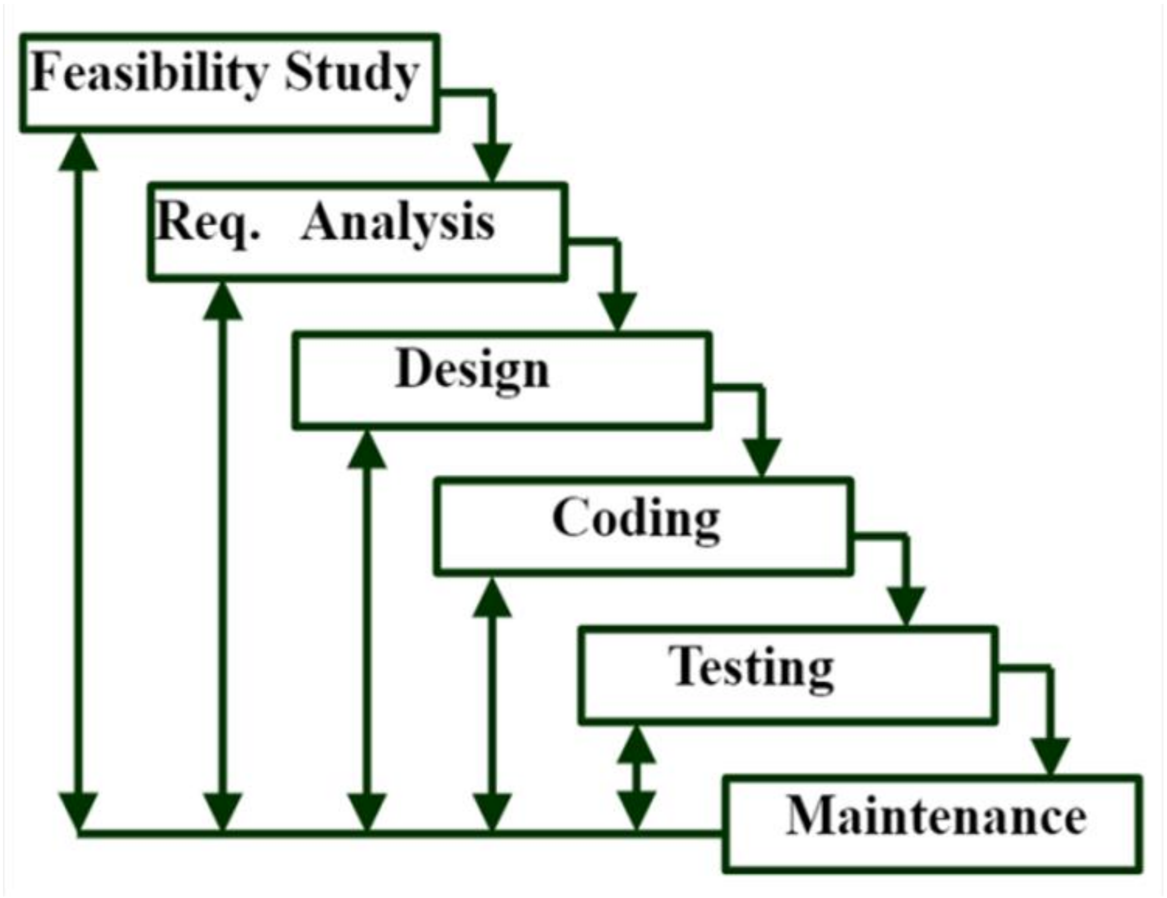


Figure 4: Iterative Waterfall Model

The iterative methodology is the extended form of waterfall model. The only difference is backtracking of phases on detection of errors at any stage is possible. Once a bug is detected changes can be made where it is introduced.

CHAPTER 3: SYSTEM REQUIREMENT SPECIFICATION AND ANALYSIS

3.1 Technical Feasibility Study

To develop this project current resources of both hardware and software are analyzed. Hardware and software used to develop Neo-Attendance are listed as:

a. Hardware

i. Laptop

Type	**Description**
Micro Processor	11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz
Machine OS	Windows 11 Home Single Language, 64-bit operating system, x64-based processor
Machine RAM	8.00 GB

ii. Camera

b. Software

i. Python - v3.10

ii. Django – v4

iii. Visual Studio Code - v 1.68

iv. PyCharm 2022.1.4

v. HTML5

vi. Bootstrap – v5

vii. CSS3

viii. PostgreSQL – v14

ix. pgAdmin4 – v6

3.2 Requirement Analysis Introduction

Requirement analysis is a most crucial phase of software development. This task of analysis begins after the completion of feasibility study. In this phase, task like fact-finding, current system overview and research overview are conducted in order to find out the actual facts or requirements for the purposed system.

3.3 Purpose of system requirement

System requirement specification is well documented which describes the requirement analysis. It covers the optimized form of requirements that has been gathered from the whole process of requirement analysis phase. This process of preparing system requirement specification makes easier for system designer to understand the actual requirements as the main purpose of this document.

The intended reader of this document is designer. Since this document covers the overall description and overview of the system, so in designer's perspective it is important for designer to develop the system according to the system requirement specification.

3.4 User classes and characteristics

Admin User

Admin of the Neo-Attendance can operate basically 3 tasks.

i. Add: Admin can add new user for the system, register new students to the system.
ii. Update: Admin can update the student records.
iii. Remove: Admin can remove the students details from the system.

Design and implementation (constraints)

i. The authentication shall be done with username and password.
ii. The user should have a valid username.
iii. The user must have logic and intelligence to use the features of the site.

System Features

This section of the specification refers to the detailed functional requirements of the proposed system.

The detailed functional requirements are listed below:

S.N.	Features	Use Case
1	Authentication System - Authentication of user.	Validates System User
2	Student Registration System - Create, update, delete and view the students.	Create Student Profile
3	Admin System	Django Admin Panel
4	System User Registration	Add User Profile

3.5 Interface Design Requirement

Site Login Panel

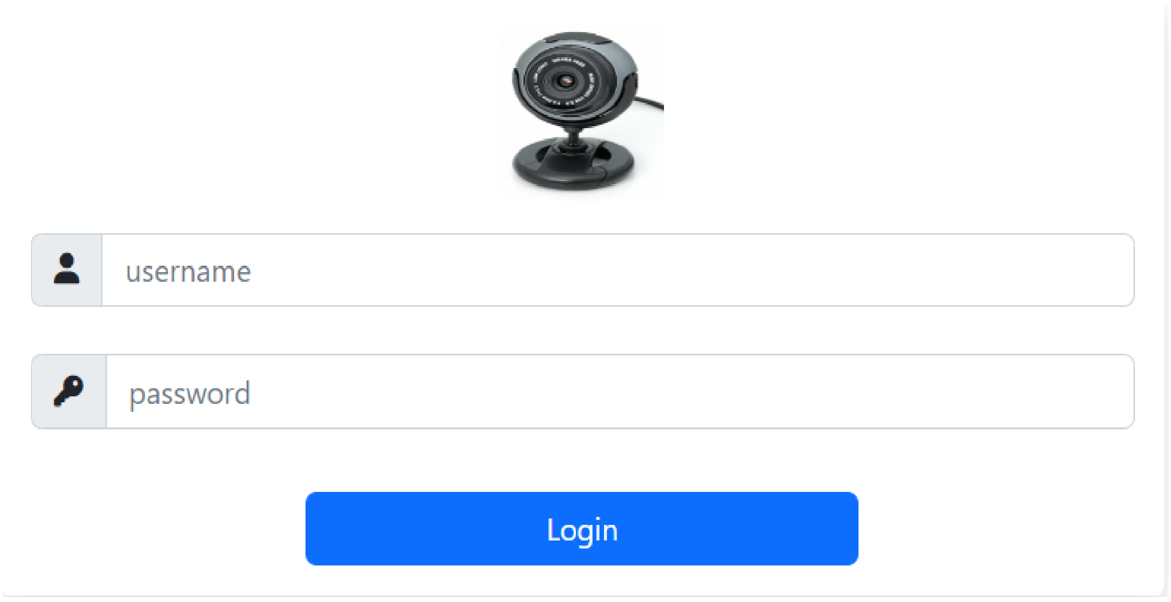
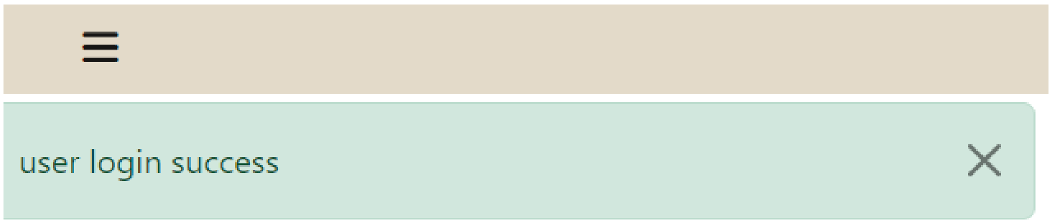


Figure 5: Login Page

Details Requirement

i. Username and password are mandatory for site login.

ii. If username or password is wrong then user can't access the system.



user login success

Figure 6: Login Success Message

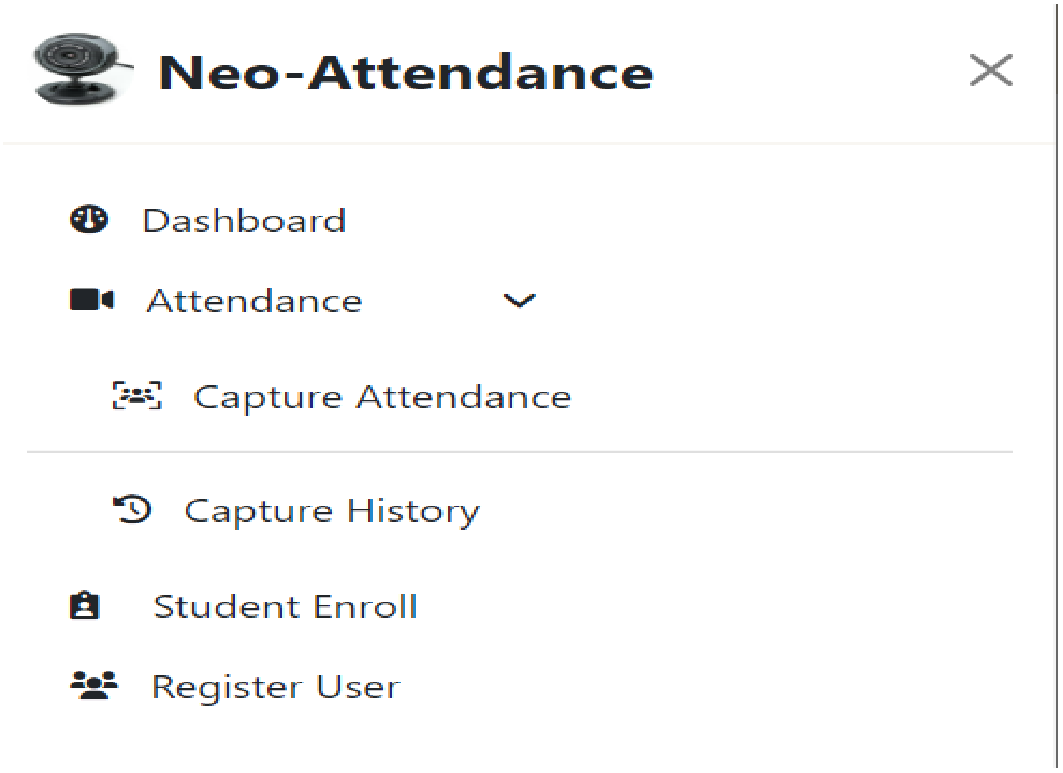


Figure 7: Sidebar After Successful Login

Either username or password is incorrect then it sends the authentication error message. Invalid username or password message is shown in case of login failure.

Student Registration Page

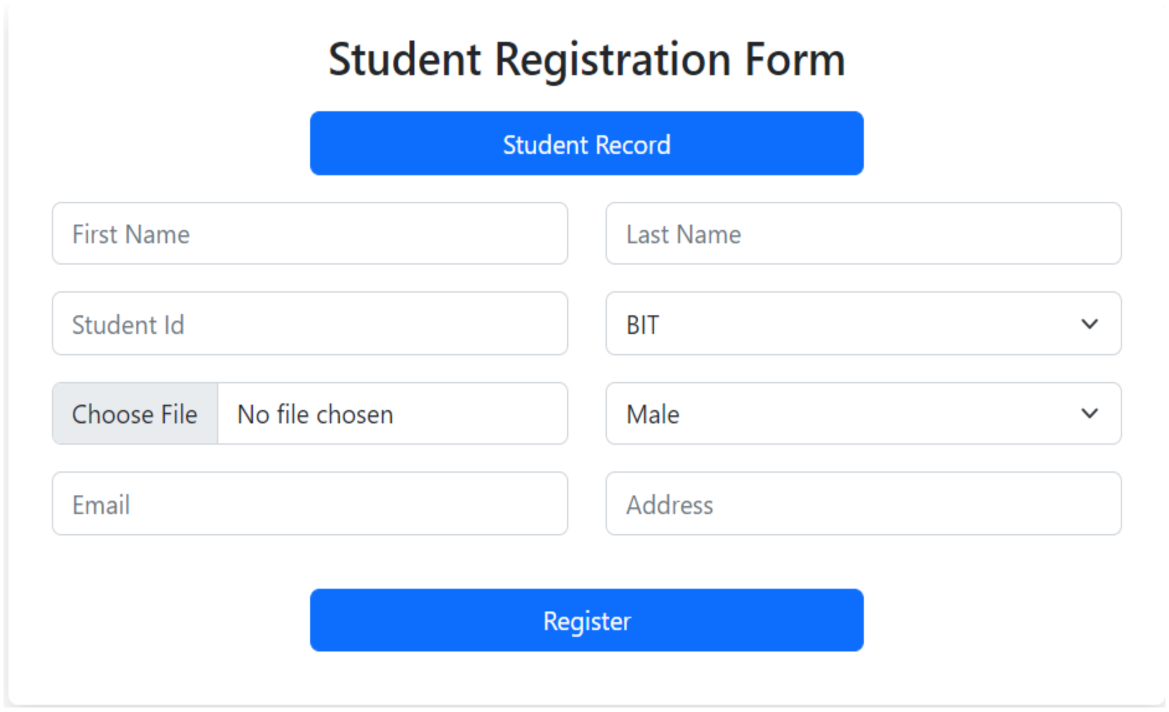
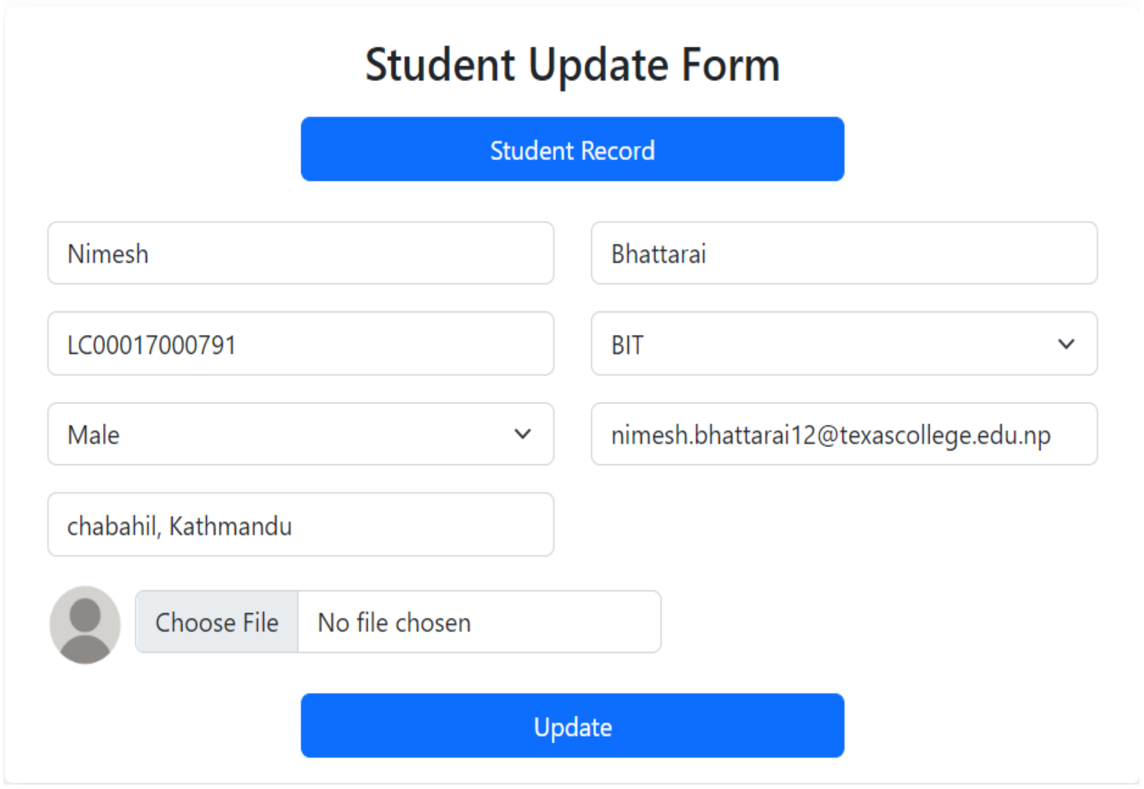


Figure 8: Student Registration Page Interface

Detailed Requirement

i. User can add student details like first name, last name, address, email.
ii. User can store the image of each students.
iii. After successful registration, user can view student record.

Update Student Record

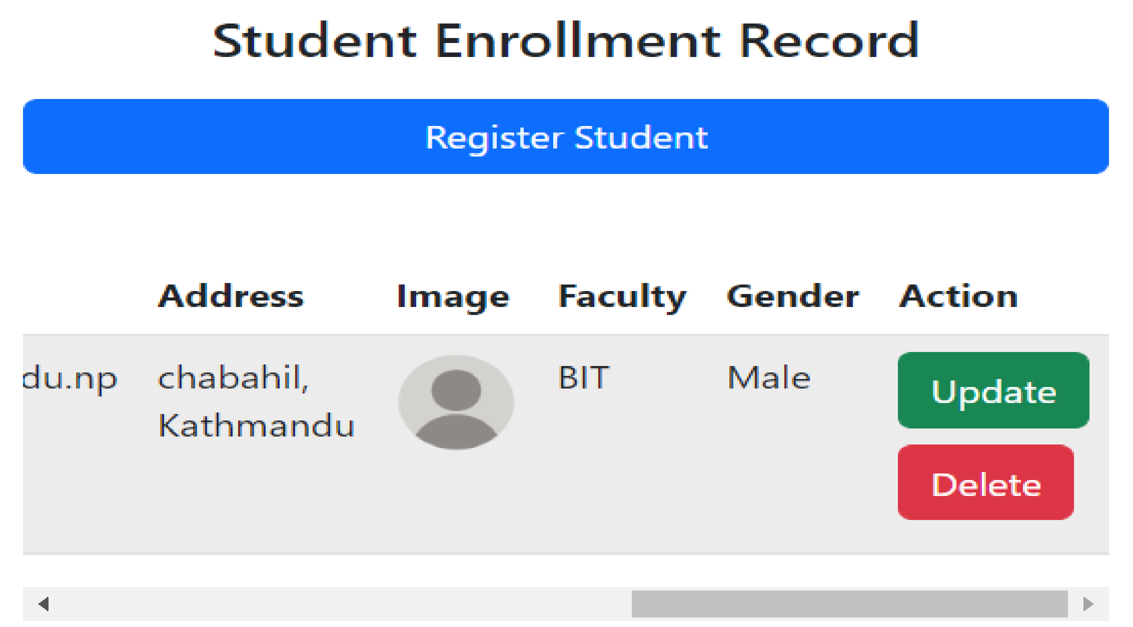

Student Update Form

Nimesh	Bhattarai
LC00017000791	BIT
Male	nimesh.bhattarai12@texascollege.edu.np
chabahil, Kathmandu	

Figure 9: Student Update Page Interface

From the update UI, user can update student record. After updating the record old record replaced with new record. It is not mandatory to re-write the data again. Only change the required field and update it.

Delete Student Record

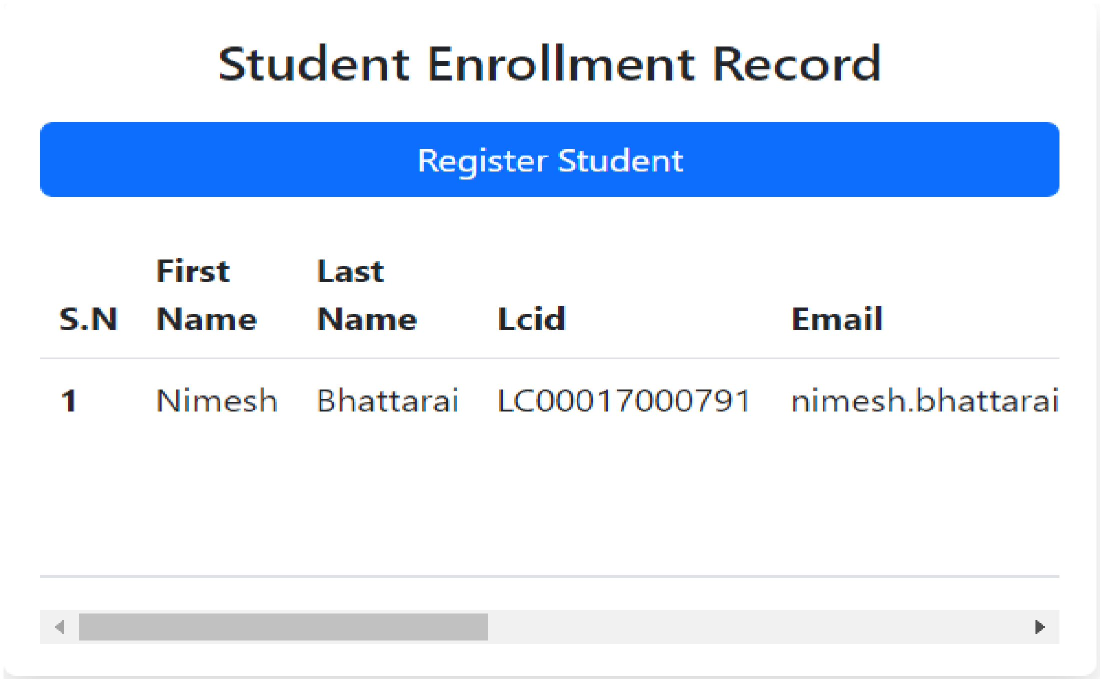

Student Enrollment Record

	Address	Image	Faculty	Gender	Action
du.np	chabahil, Kathmandu		BIT	Male	Update Delete

Figure 10: Student Delete Interface

User can delete the student record by clicking the delete button. The record also gets deleted from database.

View Student Record



Student Enrollment Record

S.N	First Name	Last Name	Lcid	Email
1	Nimesh	Bhattarai	LC00017000791	nimesh.bhattarai

Figure 11: Student Record Table Interface

User can view the stored data in database through the table. The stored data can be shown in table.

Conclusion

In this chapter, system requirement specification and analysis for the system requirement is prepared with a detail view of the requirements providing a brief textual description.

CHAPTER 4: SYSTEM DESIGN

4.1 Introduction

System design is the crucial phase of software development. This task begins after the completion of system requirement specification and system analysis. In this phase user requirements are transformed into suitable form. The output of this phase is the input for the development or coding phase. This phase provides the correct programming paradigm. System design ensures the minimal semantic errors. Design phase increases interactions between user and the system.

4.2 Purpose

The purpose of this design specification is to provide the diagrammatic representation of the proposed Neo-Attendance system. It shows how the system flows and how the overall system works on the web application. This also targets the developer to help them understand the scenario and develop the intended system. It will present the reader with various diagrams such as Unified Modeling Language (UML) and Entity-Relationship Diagram (ER diagram)

4.3 System Data Model Analysis

Scenario

Use Case Diagram is used to represent the scenario of the system. The system is divided into 4 distinct modules so as to make easier representation of the system. The modules are:

i. Student Registration Module (Create, update, delete and view the students)
ii. User Registration Module (Add new user for the system)
iii. Authentication Module (Validation for user and user login)
iv. Admin Module (Django Admin Panel)

A. Use Case Diagram For Neo-Attendance

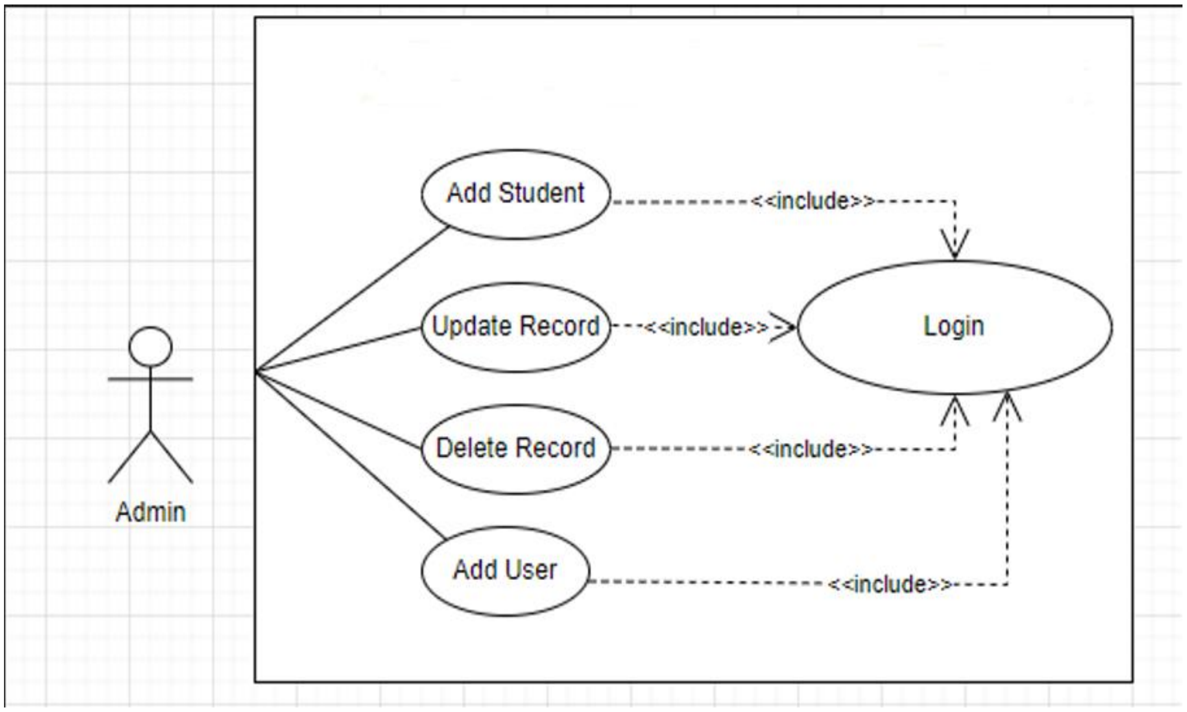


Figure 12: Use Case Diagram

Use Case Diagram Description

Use Case	Use Case Description
Add Student	Admin or user can register new student.
Update Record	User can modify student records.
Delete Record	User can delete student records.
Add User	User can add new users.
Login	Every function includes login.

B. Sequence Diagram For Neo-Attendance

Sequence diagram represents object interactions and arranged in time sequence. Sequence diagram shows the events occurs in Neo-Attendance.

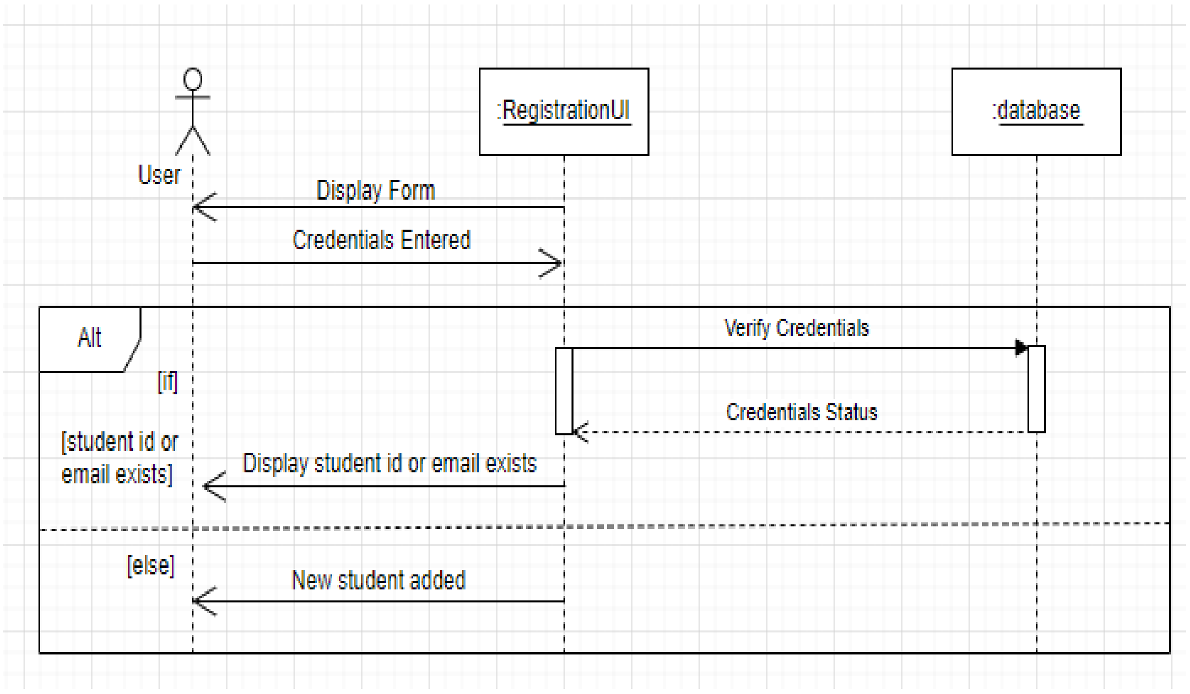


Figure 13: Sequence Diagram For Add Student

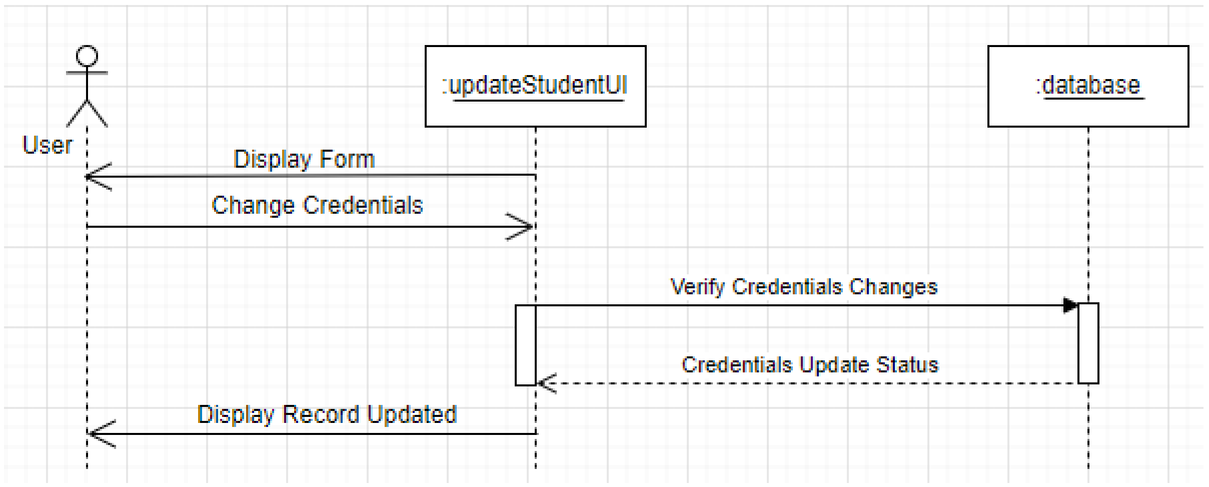


Figure 14: Sequence Diagram For Update Student Record

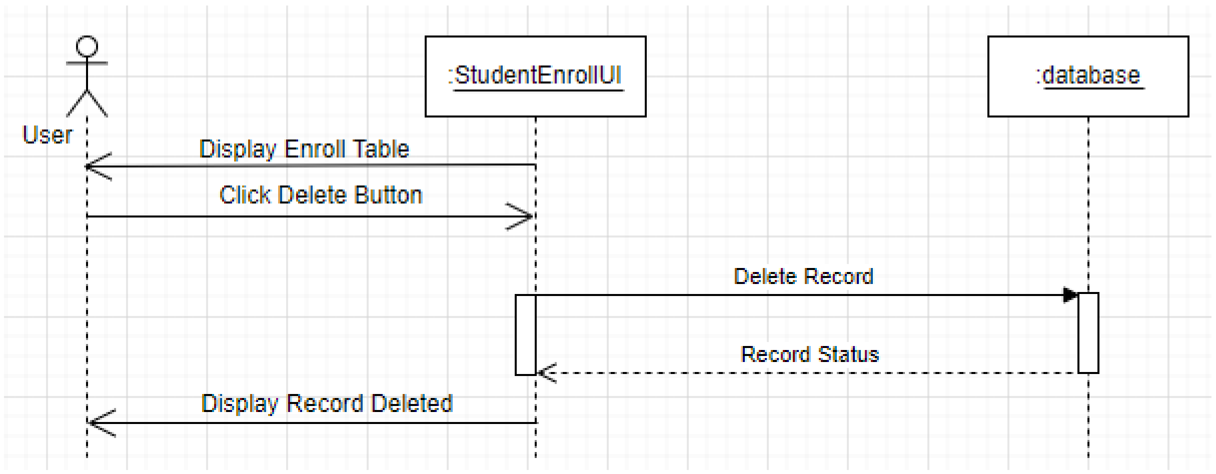


Figure 15: Sequence Diagram For Delete Student Record

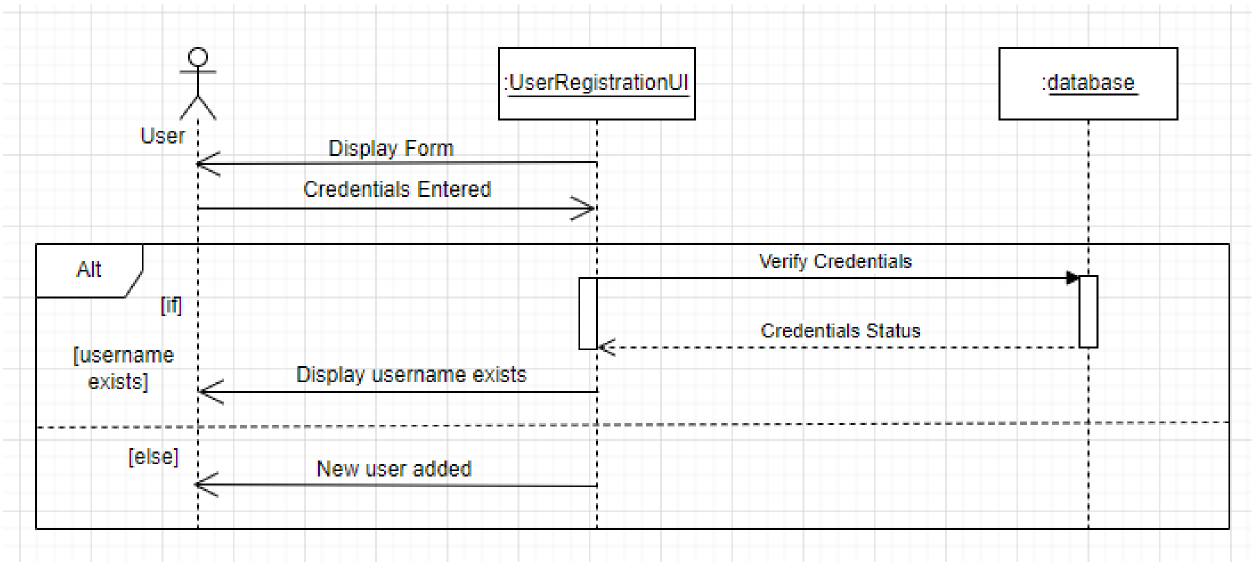


Figure 16: Sequence Diagram For Add User

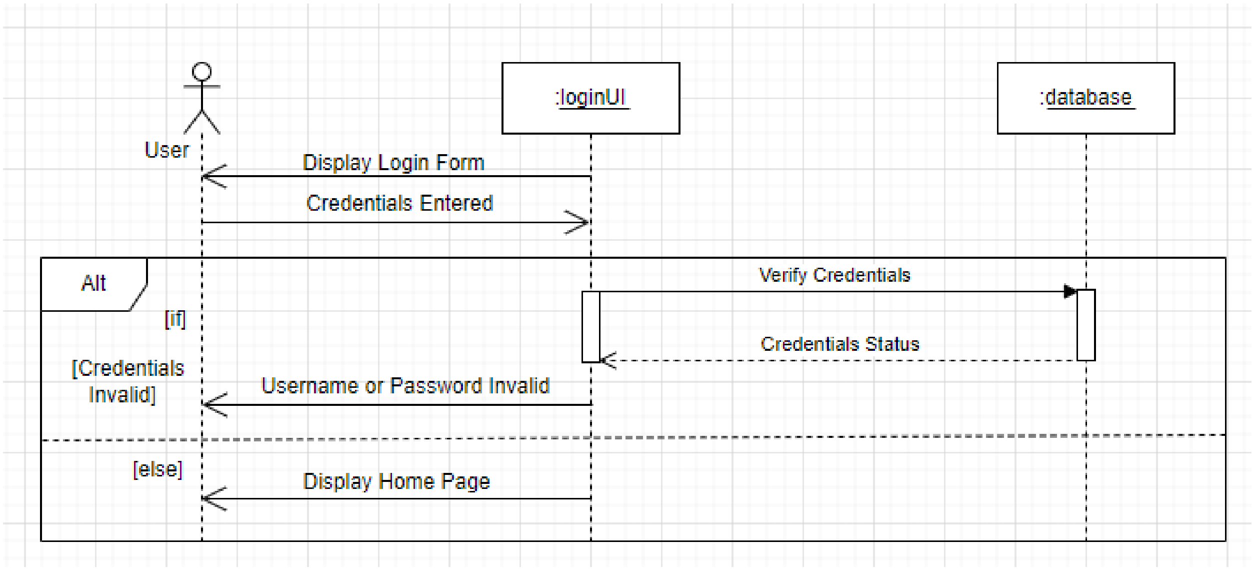


Figure 17: Sequence Diagram For User Login

C. State Diagram For Neo-Attendance

State diagram track the data and behavior of the object throughout its lifetime.

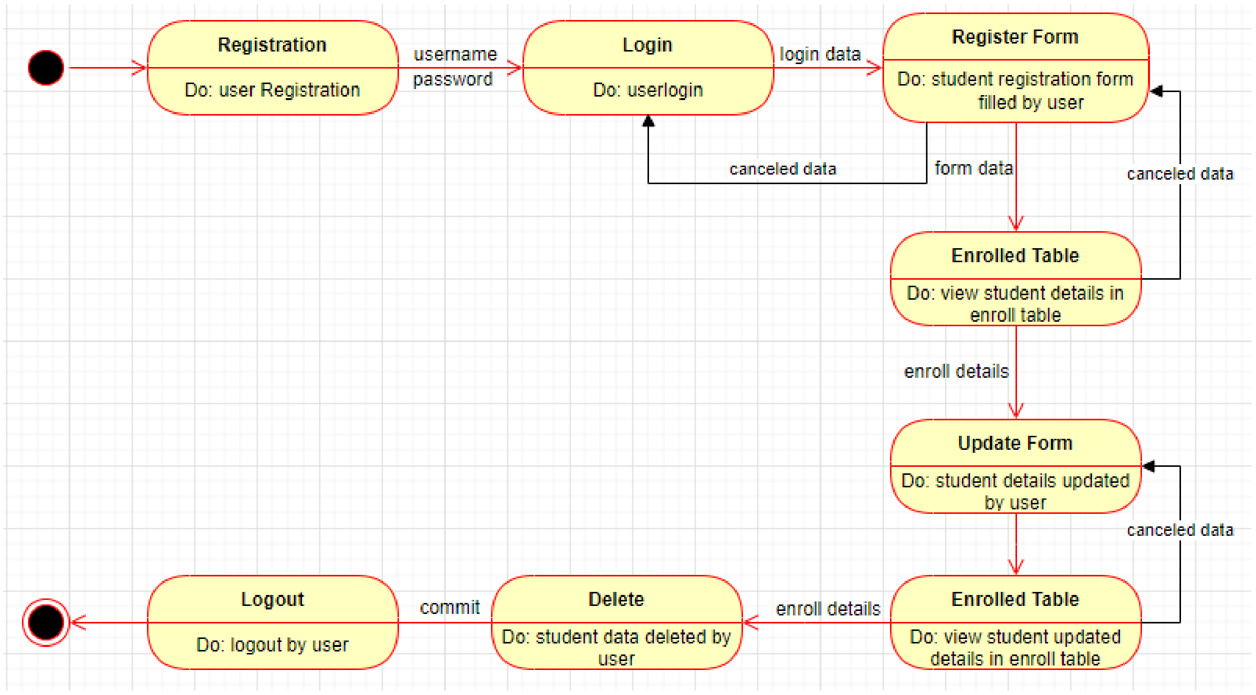


Figure 18: State Diagram For Neo-Attendance

Data model gives the structure of the data stored in the database. In relational database data are stored in rows and columns. Row store the values of the columns.

4.4 Tables and Fields in Database

PostgresSQL is used as database for this project. pgAdmin is the management tool for the PostgresSQL.

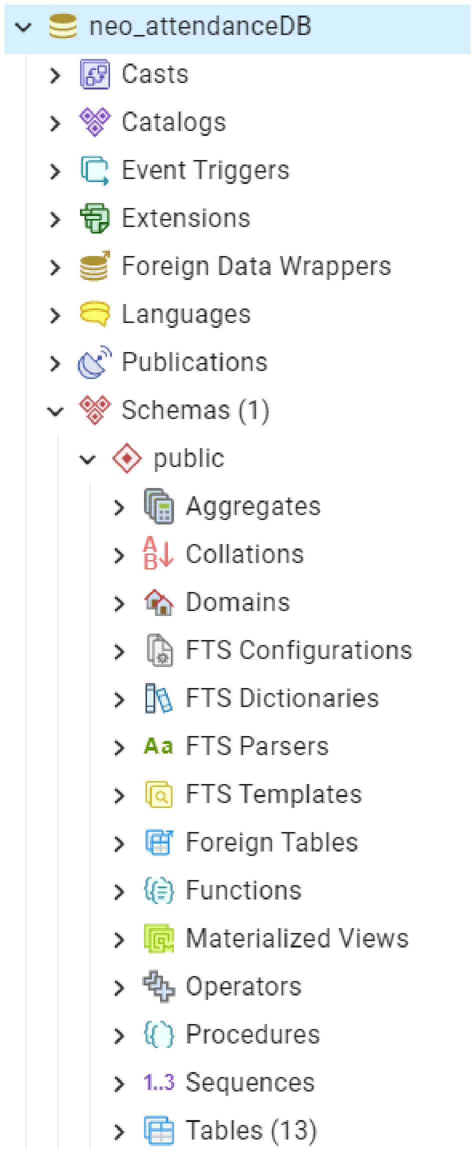
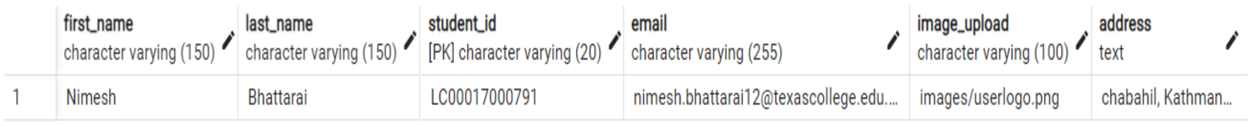


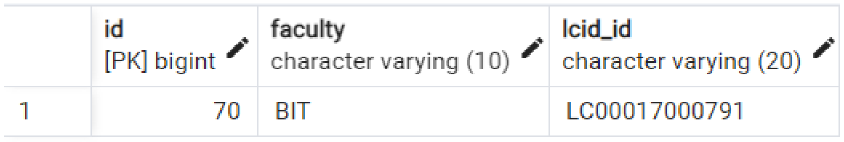
Figure 19: Neo-Attendance Database

Name of the database for the project is neo_attendanceDB.



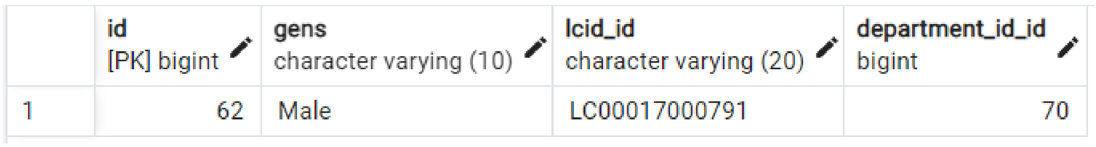
	first_name character varying (150)	last_name character varying (150)	student_id [PK] character varying (20)	email character varying (255)	image_upload character varying (100)	address text
1	Nimesh	Bhattarai	LC00017000791	nimesh.bhattarai12@texascollege.edu....	images/userlogo.png	chabahil, Kathman...

Figure 20: Student_Registration Table



	id [PK] bigint	faculty character varying (10)	lcid_id character varying (20)
1	70	BIT	LC00017000791

Figure 21: Department Table



	id [PK] bigint	gens character varying (10)	lcid_id character varying (20)	department_id_id bigint
1	62	Male	LC00017000791	70

Figure 22: Gender Table

4.5 Data Modelling In Relational Database Management System

Types of Data Models in RDBMS (Relational Database Management System)

There are three types of data models as Conceptual Data Model, Logical Data Model and Physical Data Model.

i. Conceptual Data Model

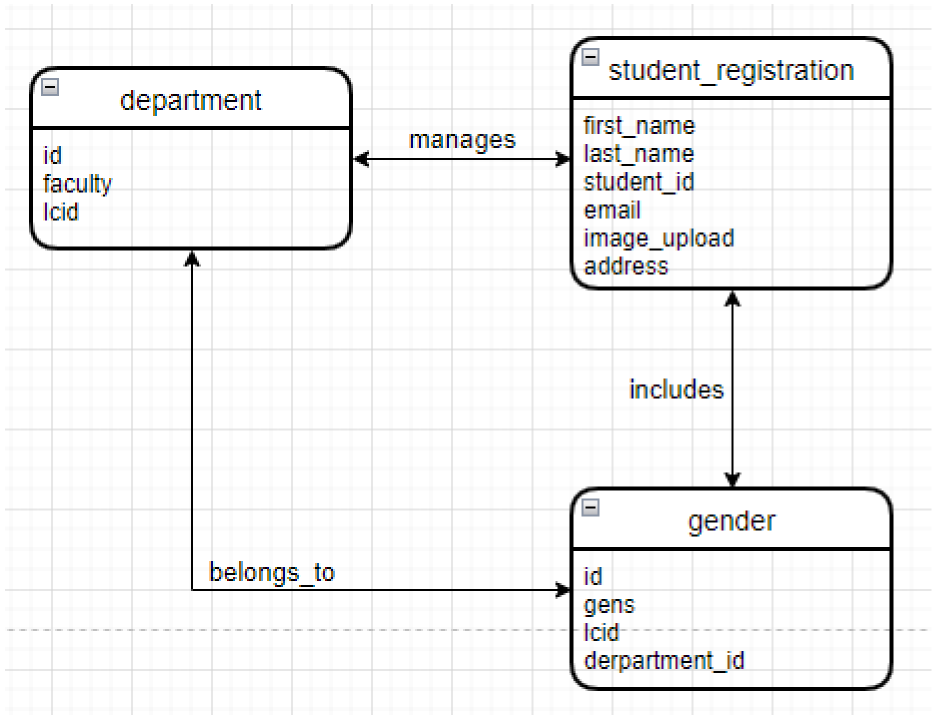


Figure 23: Conceptual Data Model

ii. Logical Data Model

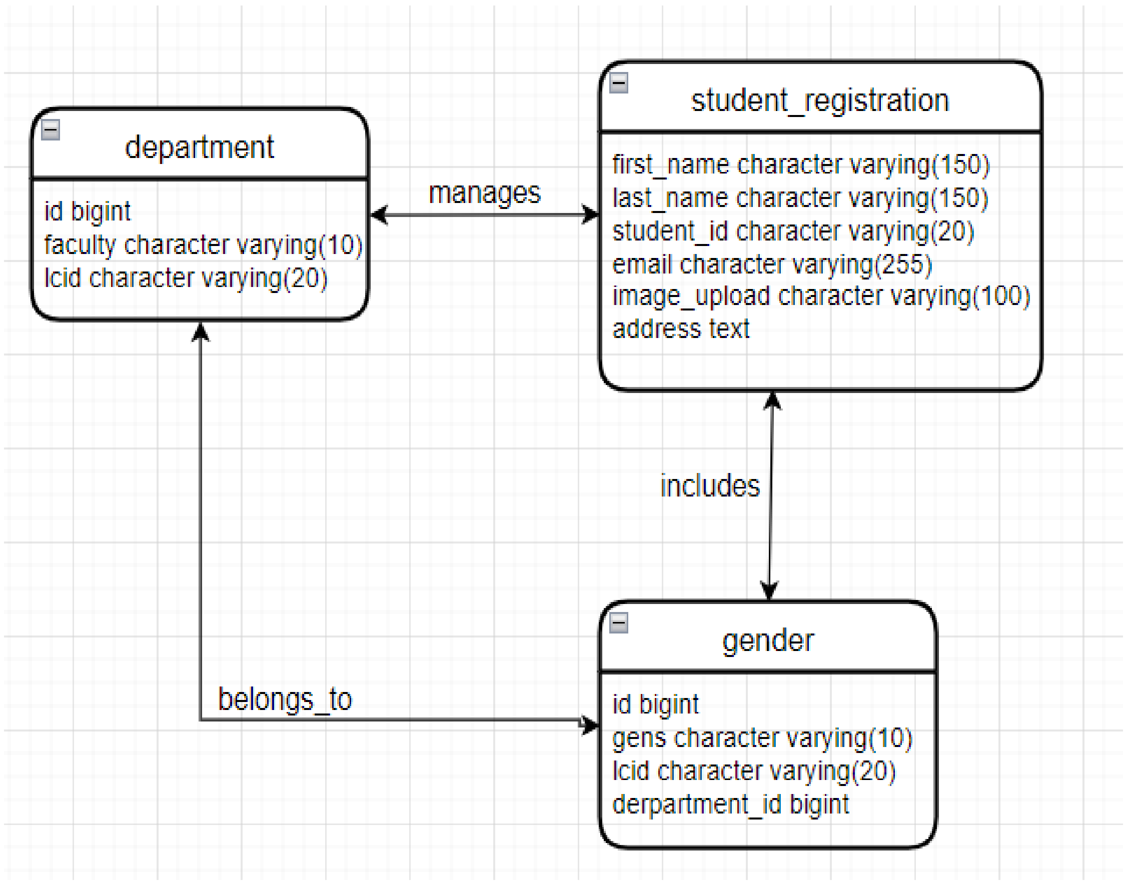

department
id bigint
faculty character varying(10)
lcid character varying(20)
manages
student_registration
first_name character varying(150)
last_name character varying(150)
student_id character varying(20)
email character varying(255)
image_upload character varying(100)
address text
includes
belongs_to
gender
id bigint
gens character varying(10)
lcid character varying(20)
derpartment_id bigint

Figure 24: Logical Data Model

iii. Physical Data Model

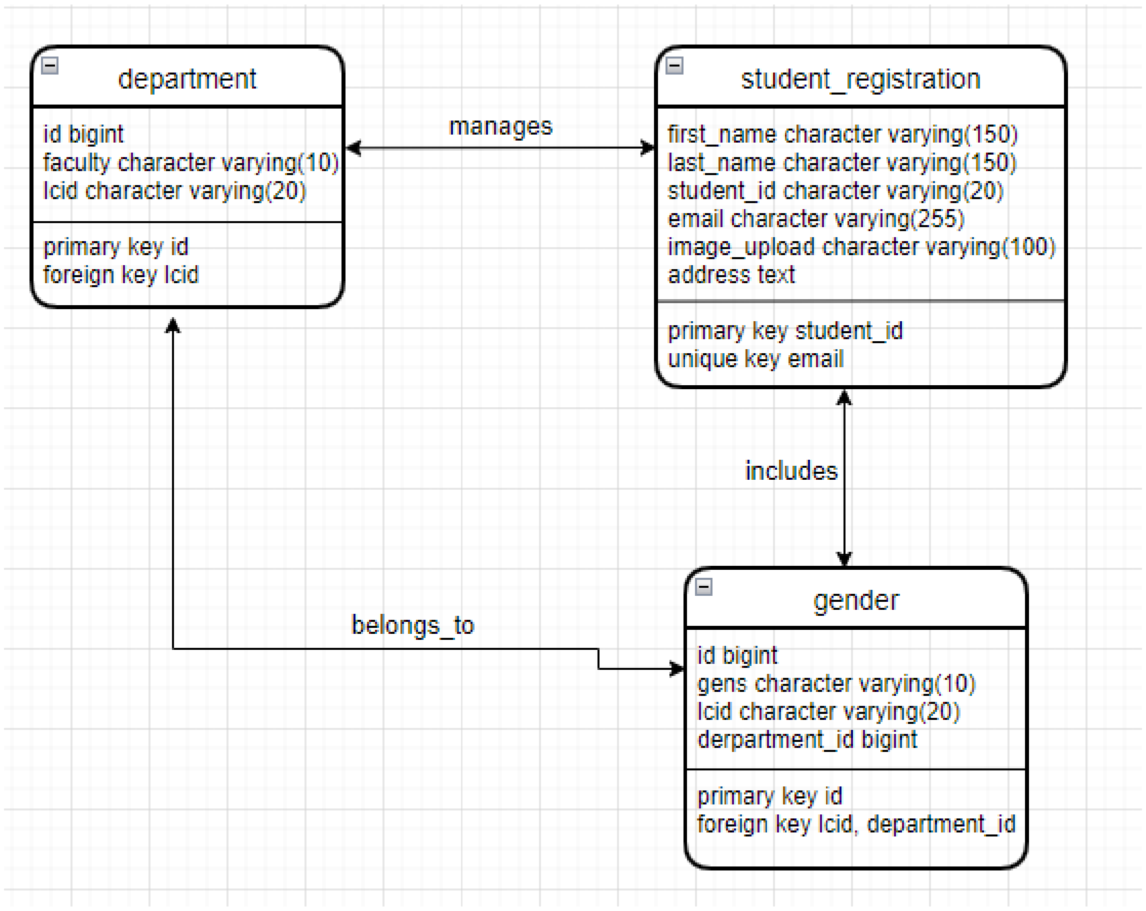

department
id bigint
faculty character varying(10)
lcid character varying(20)
primary key id
foreign key lcid
manages
student_registration
first_name character varying(150)
last_name character varying(150)
student_id character varying(20)
email character varying(255)
image_upload character varying(100)
address text
primary key student_id
unique key email
includes
belongs_to
gender
id bigint
gens character varying(10)
lcid character varying(20)
derpartment_id bigint
primary key id
foreign key lcid, department_id

Figure 25: Physical Data Model

Entity-Relationship Diagram For Neo-Attendance

ER diagram is the graphical representation of the data model which shows the relationships between entity sets stored in a database.

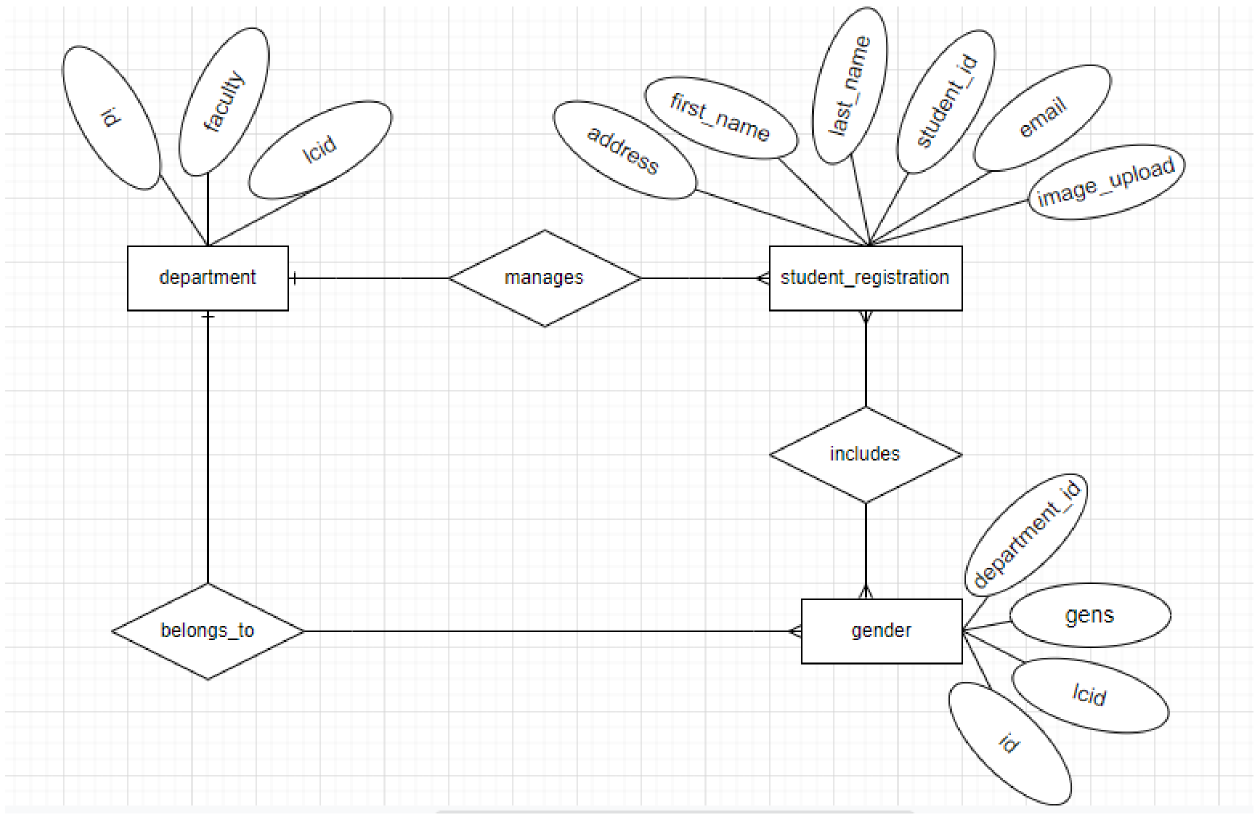
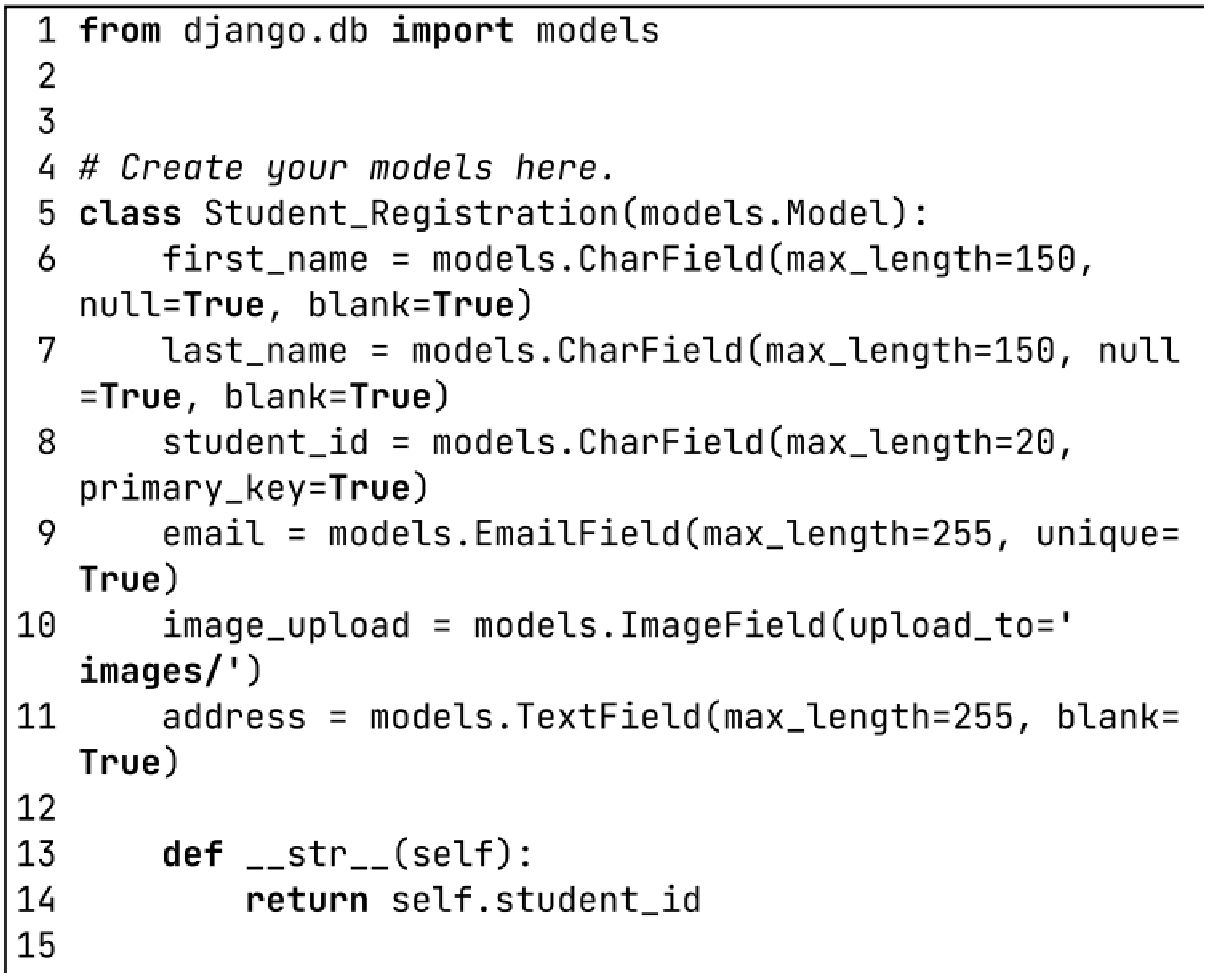


Figure 26: ER Diagram For Student Registration

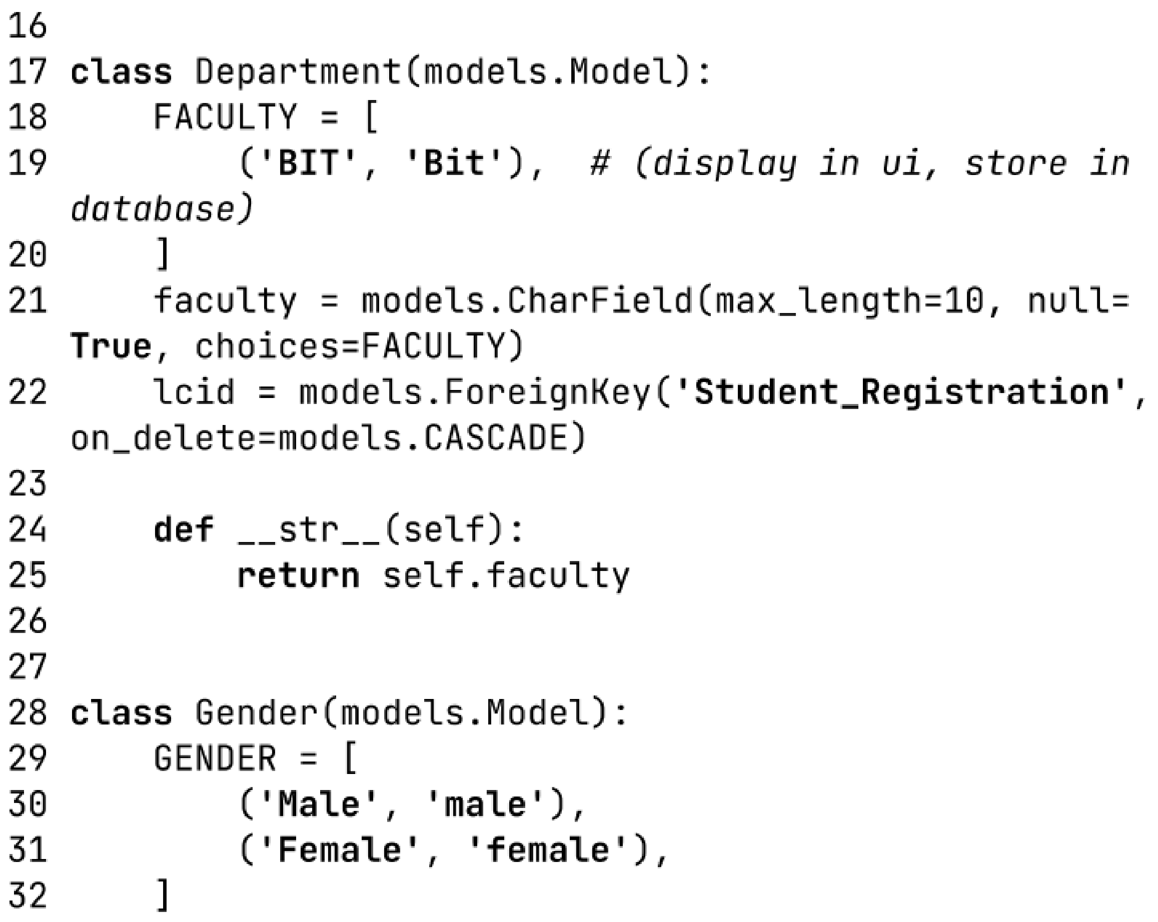
CHAPTER 5: SOURCE CODE

5.1 Models.py



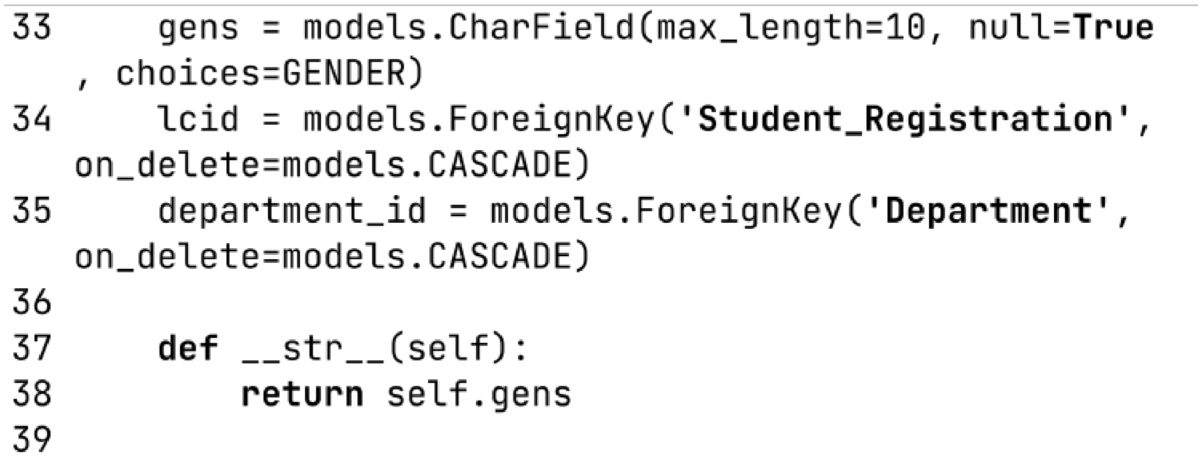
```
from django.db import models


# Create your models here.
class Student_Registration(models.Model):
    first_name = models.CharField(max_length=150,
null=True, blank=True)
    last_name = models.CharField(max_length=150, null
=True, blank=True)
    student_id = models.CharField(max_length=20,
primary_key=True)
    email = models.EmailField(max_length=255, unique=
True)
    image_upload = models.ImageField(upload_to='
images/')
    address = models.TextField(max_length=255, blank=
True)

    def __str__(self):
        return self.student_id

```

Figure 27: Models1



```python

class Department(models.Model):
    FACULTY = [
        ('BIT', 'Bit'),  # (display in ui, store in database)
    ]
    faculty = models.CharField(max_length=10, null=True, choices=FACULTY)
    lcid = models.ForeignKey('Student_Registration', on_delete=models.CASCADE)

    def __str__(self):
        return self.faculty


class Gender(models.Model):
    GENDER = [
        ('Male', 'male'),
        ('Female', 'female'),
    ]
```

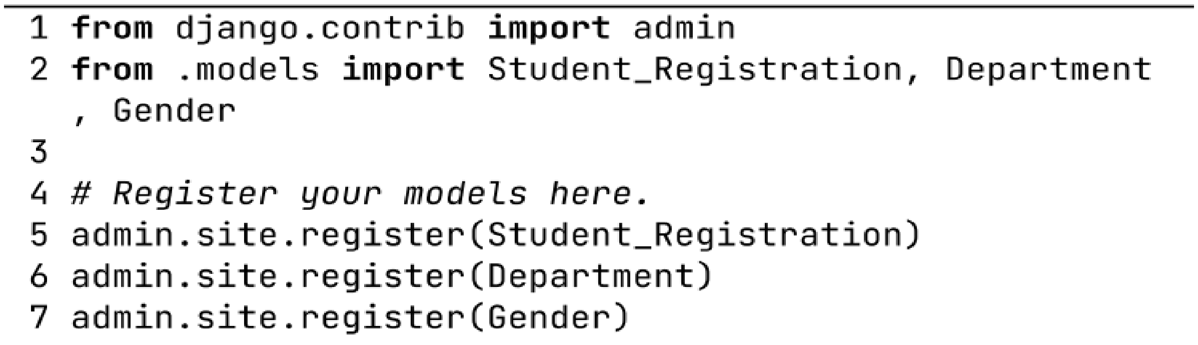
Figure 28: Models2



```python
    gens = models.CharField(max_length=10, null=True, choices=GENDER)
    lcid = models.ForeignKey('Student_Registration', on_delete=models.CASCADE)
    department_id = models.ForeignKey('Department', on_delete=models.CASCADE)

    def __str__(self):
        return self.gens

```

Figure 29: Models3

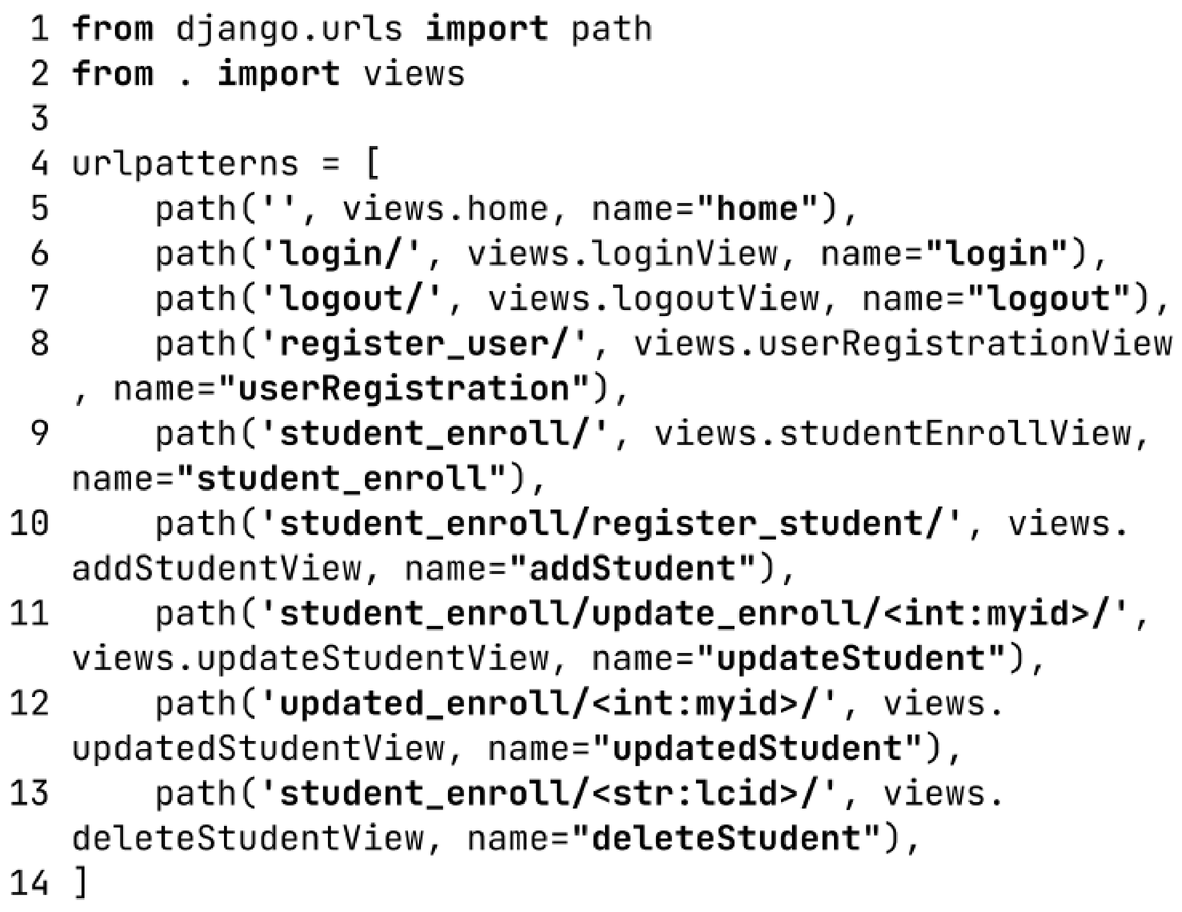
5.2 Admin.py



```python
from django.contrib import admin
from .models import Student_Registration, Department, Gender

# Register your models here.
admin.site.register(Student_Registration)
admin.site.register(Department)
admin.site.register(Gender)
```

Figure 30: Admin

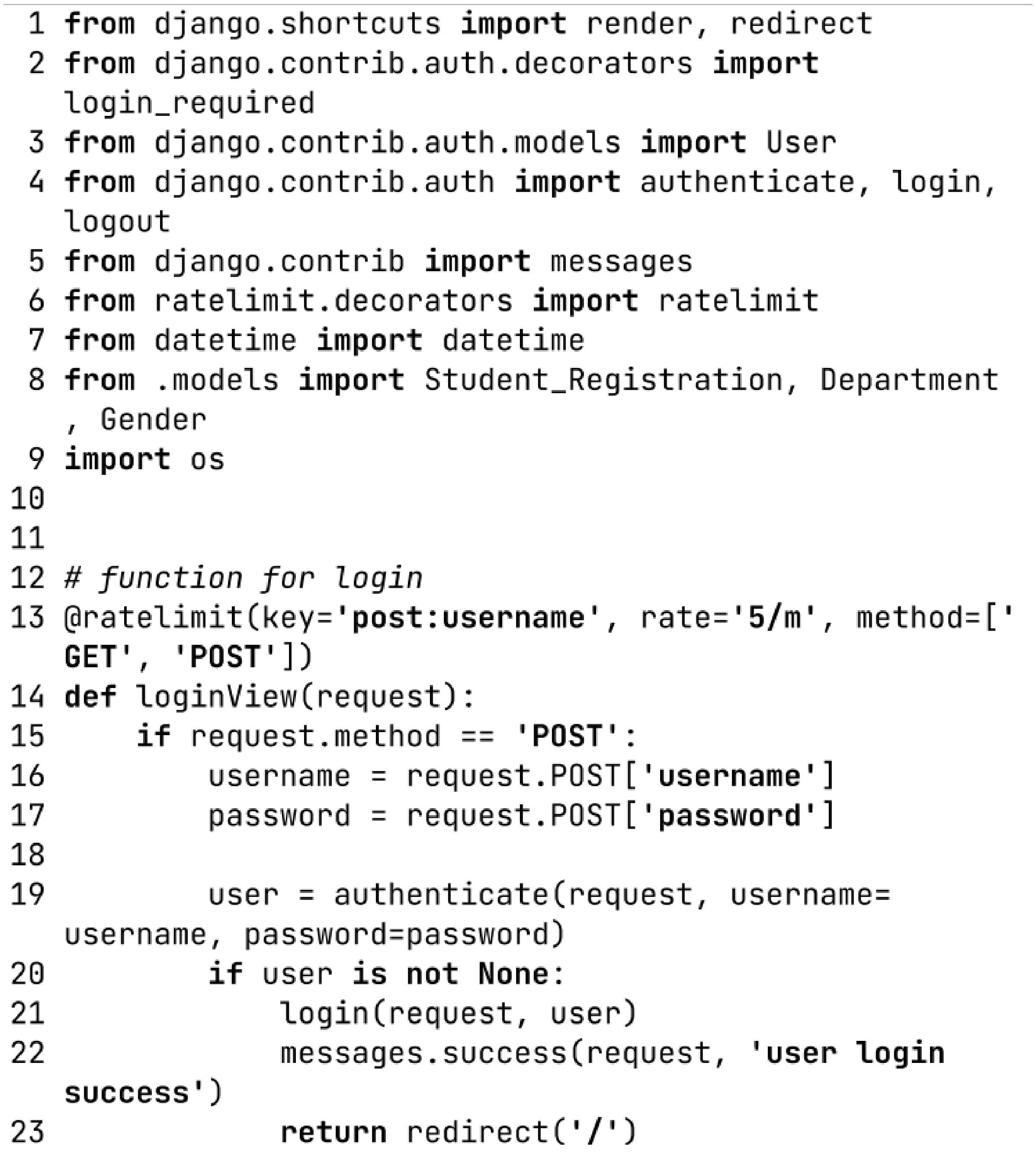
5.3 AppUrls.py



```python
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name="home"),
    path('login/', views.loginView, name="login"),
    path('logout/', views.logoutView, name="logout"),
    path('register_user/', views.userRegistrationView, name="userRegistration"),
    path('student_enroll/', views.studentEnrollView, name="student_enroll"),
    path('student_enroll/register_student/', views.addStudentView, name="addStudent"),
    path('student_enroll/update_enroll/<int:myid>/', views.updateStudentView, name="updateStudent"),
    path('updated_enroll/<int:myid>/', views.updatedStudentView, name="updatedStudent"),
    path('student_enroll/<str:lcid>/', views.deleteStudentView, name="deleteStudent"),
]
```

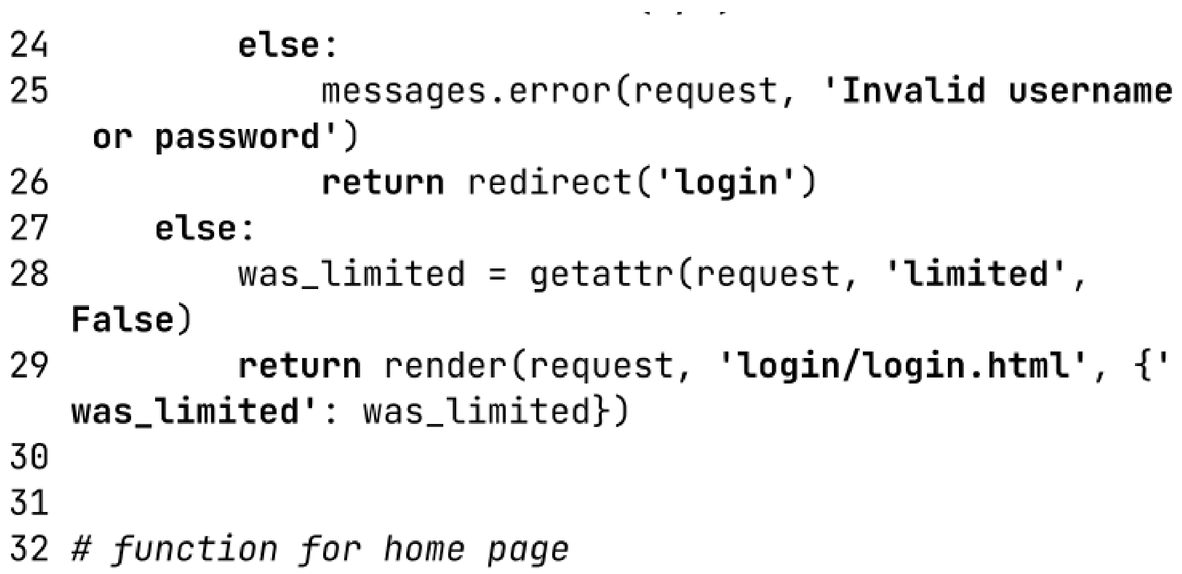
Figure 31: App Urls

5.4 Views.py



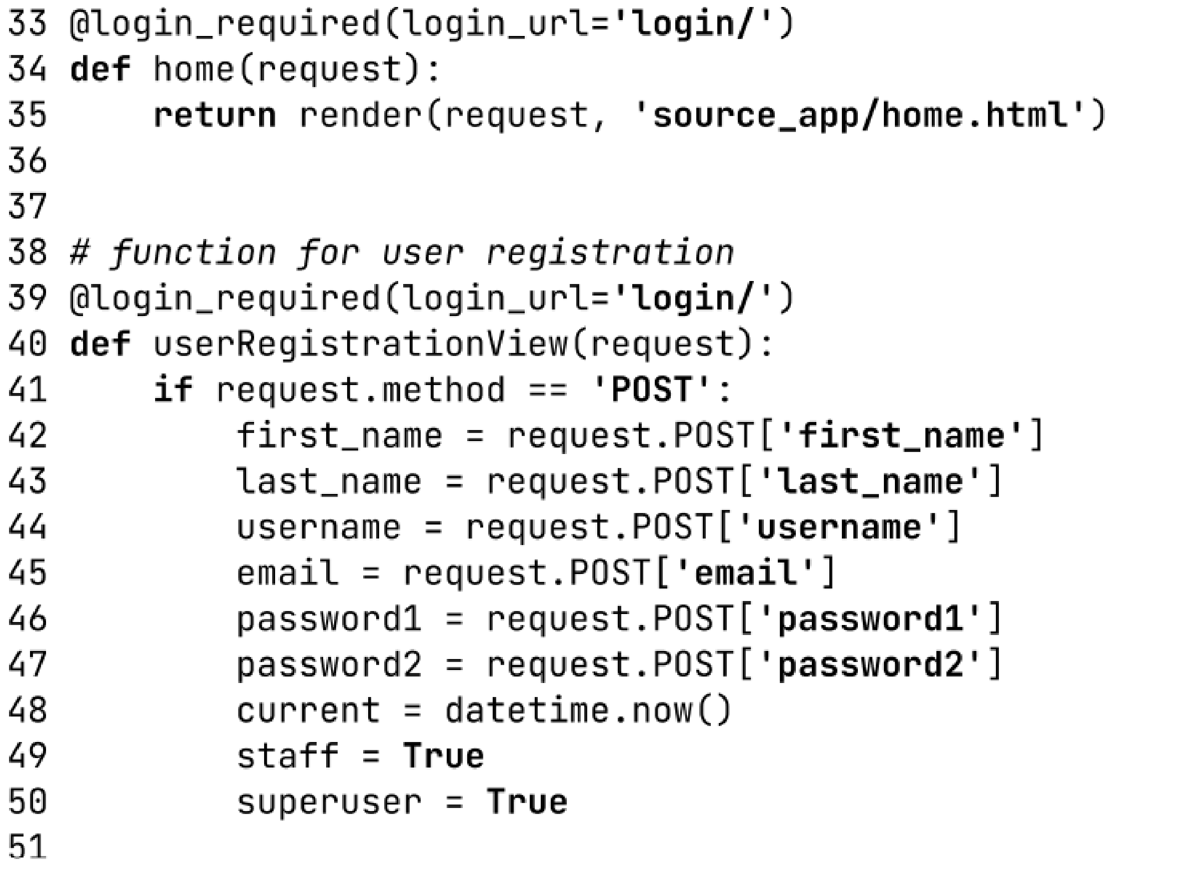
```python
from django.shortcuts import render, redirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages
from ratelimit.decorators import ratelimit
from datetime import datetime
from .models import Student_Registration, Department, Gender
import os


# function for login
@ratelimit(key='post:username', rate='5/m', method=['GET', 'POST'])
def loginView(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            messages.success(request, 'user login success')
            return redirect('/')
```

Figure 32: Views1



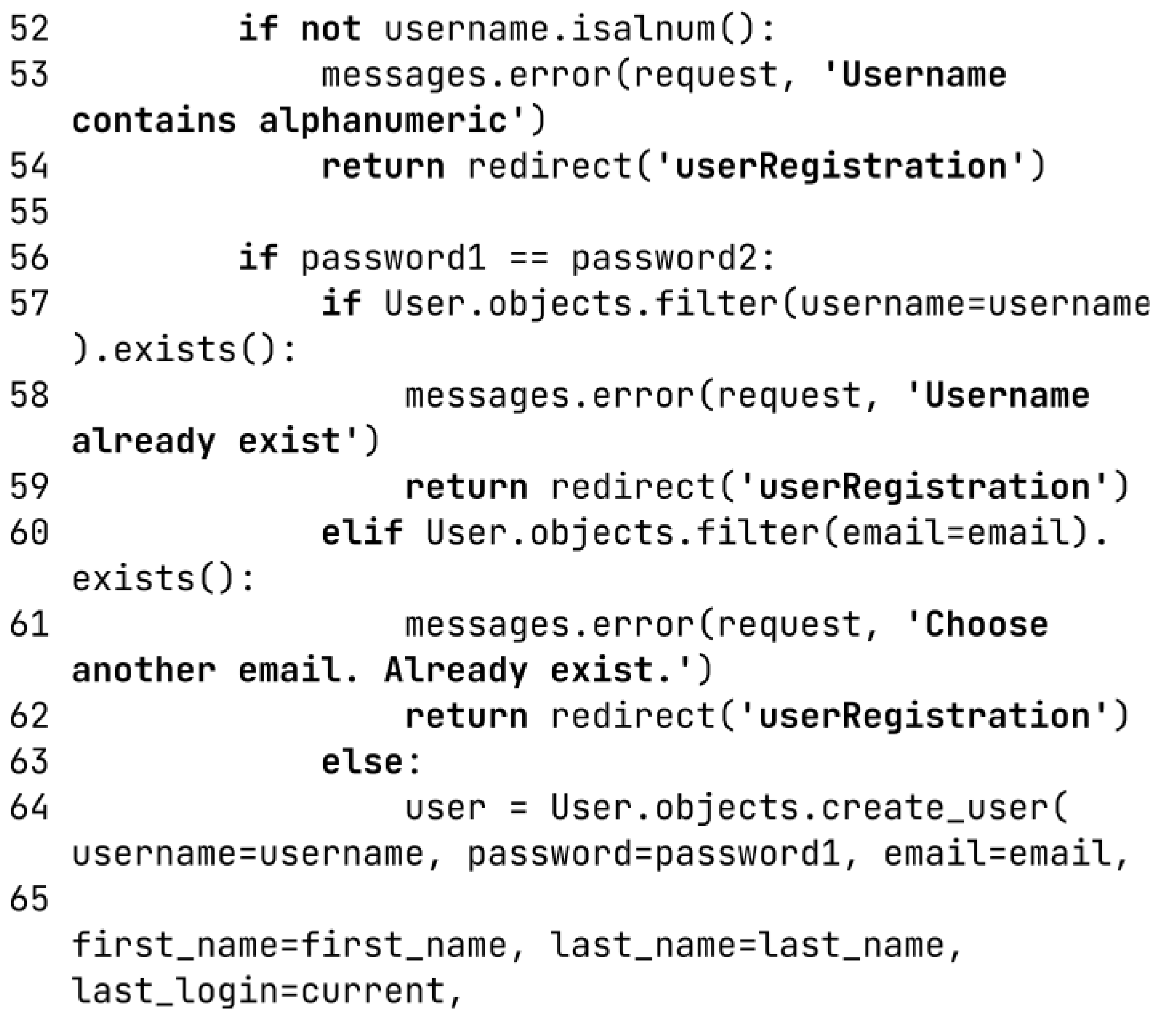
```python
        else:
            messages.error(request, 'Invalid username or password')
            return redirect('login')
    else:
        was_limited = getattr(request, 'limited', False)
        return render(request, 'login/login.html', {'was_limited': was_limited})


# function for home page
```

Figure 33: Views2



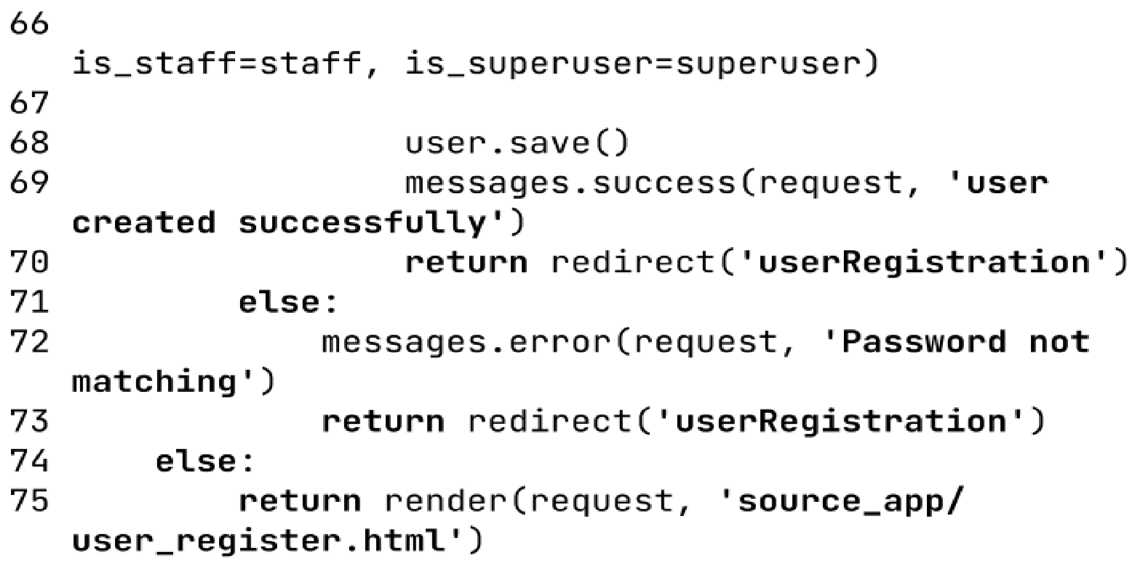
```python
@login_required(login_url='login/')
def home(request):
    return render(request, 'source_app/home.html')


# function for user registration
@login_required(login_url='login/')
def userRegistrationView(request):
    if request.method == 'POST':
        first_name = request.POST['first_name']
        last_name = request.POST['last_name']
        username = request.POST['username']
        email = request.POST['email']
        password1 = request.POST['password1']
        password2 = request.POST['password2']
        current = datetime.now()
        staff = True
        superuser = True

```

Figure 34: Views3



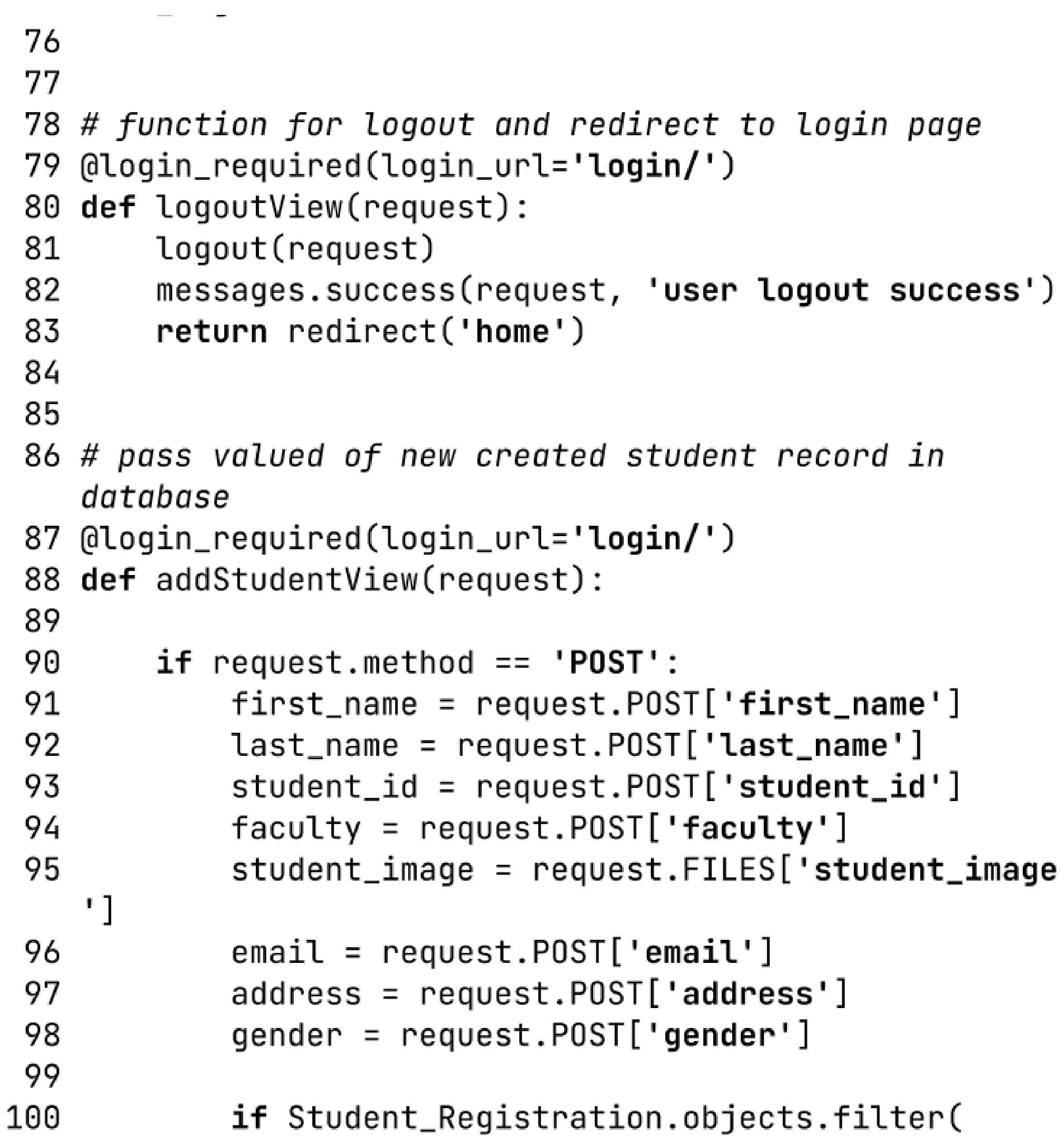
```python
        if not username.isalnum():
            messages.error(request, 'Username contains alphanumeric')
            return redirect('userRegistration')

        if password1 == password2:
            if User.objects.filter(username=username).exists():
                messages.error(request, 'Username already exist')
                return redirect('userRegistration')
            elif User.objects.filter(email=email).exists():
                messages.error(request, 'Choose another email. Already exist.')
                return redirect('userRegistration')
            else:
                user = User.objects.create_user(username=username, password=password1, email=email,
                    first_name=first_name, last_name=last_name, last_login=current,
```

Figure 35: Views4



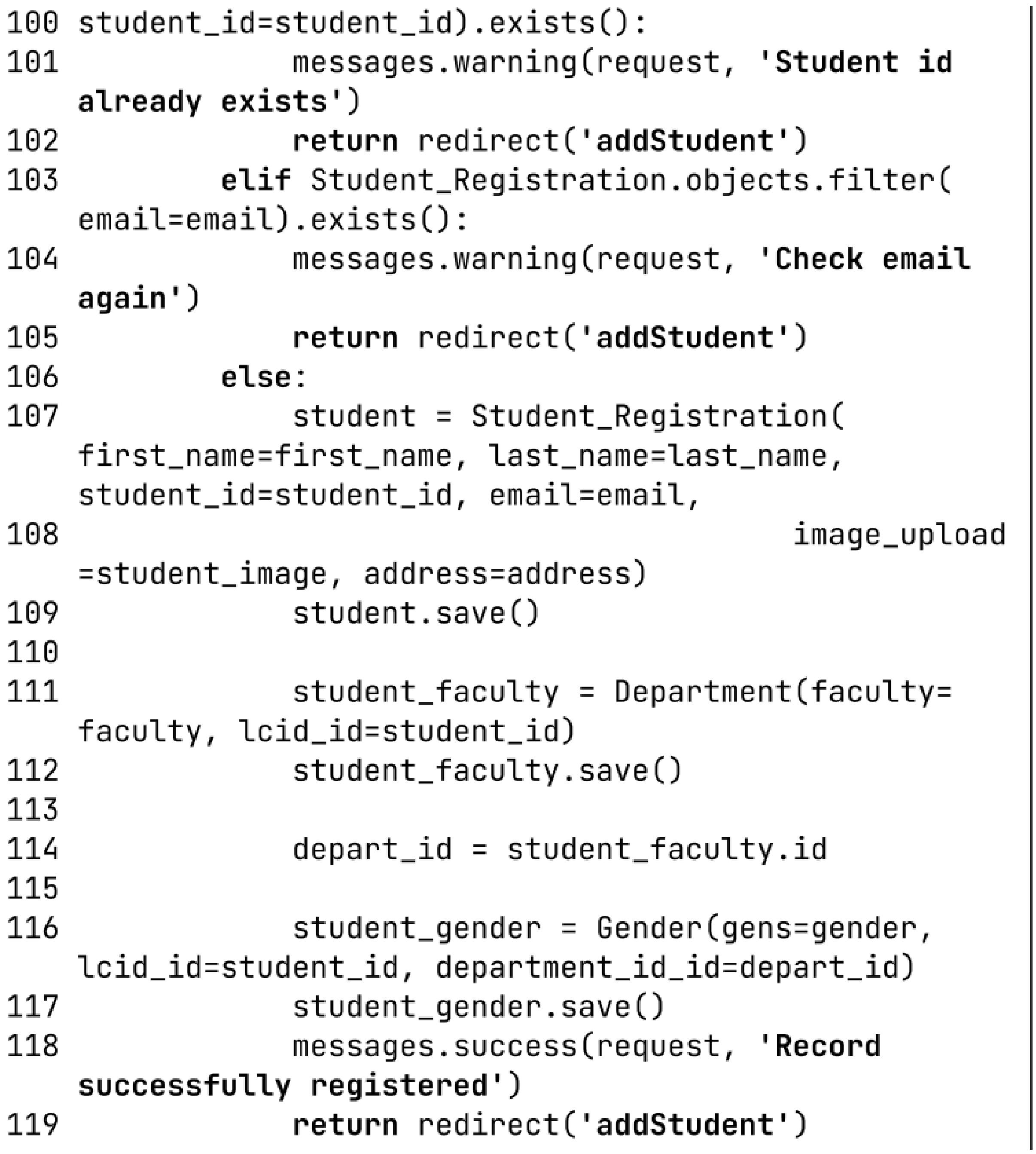
```python
                    is_staff=staff, is_superuser=superuser)

                user.save()
                messages.success(request, 'user created successfully')
                return redirect('userRegistration')
        else:
            messages.error(request, 'Password not matching')
            return redirect('userRegistration')
    else:
        return render(request, 'source_app/user_register.html')
```

Figure 36: Views5



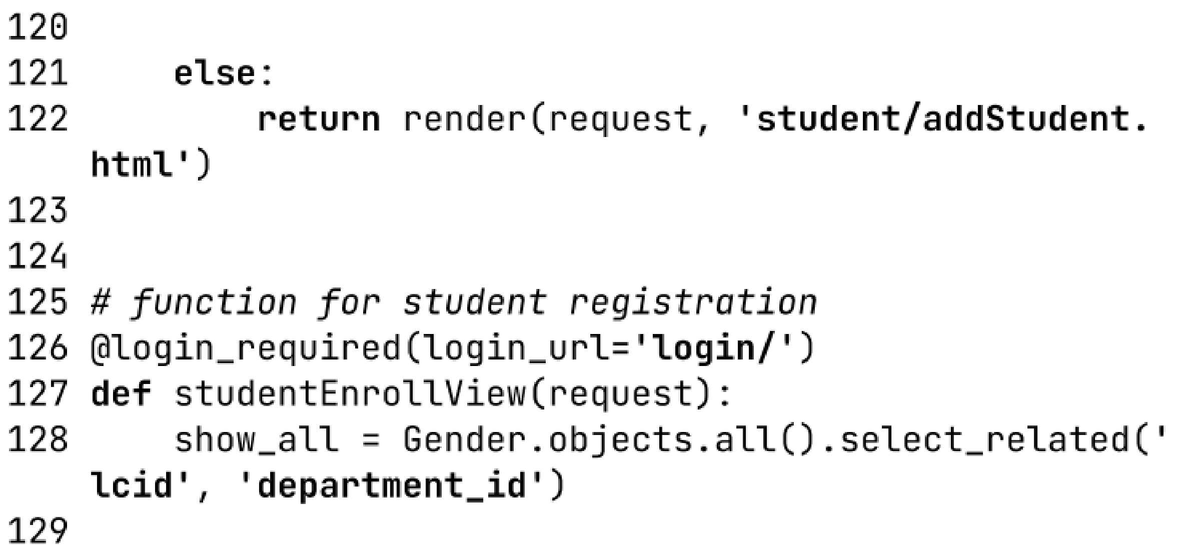
```python


# function for logout and redirect to login page
@login_required(login_url='login/')
def logoutView(request):
    logout(request)
    messages.success(request, 'user logout success')
    return redirect('home')


# pass valued of new created student record in database
@login_required(login_url='login/')
def addStudentView(request):

    if request.method == 'POST':
        first_name = request.POST['first_name']
        last_name = request.POST['last_name']
        student_id = request.POST['student_id']
        faculty = request.POST['faculty']
        student_image = request.FILES['student_image']
        email = request.POST['email']
        address = request.POST['address']
        gender = request.POST['gender']

        if Student_Registration.objects.filter(
```

Figure 37: Views6



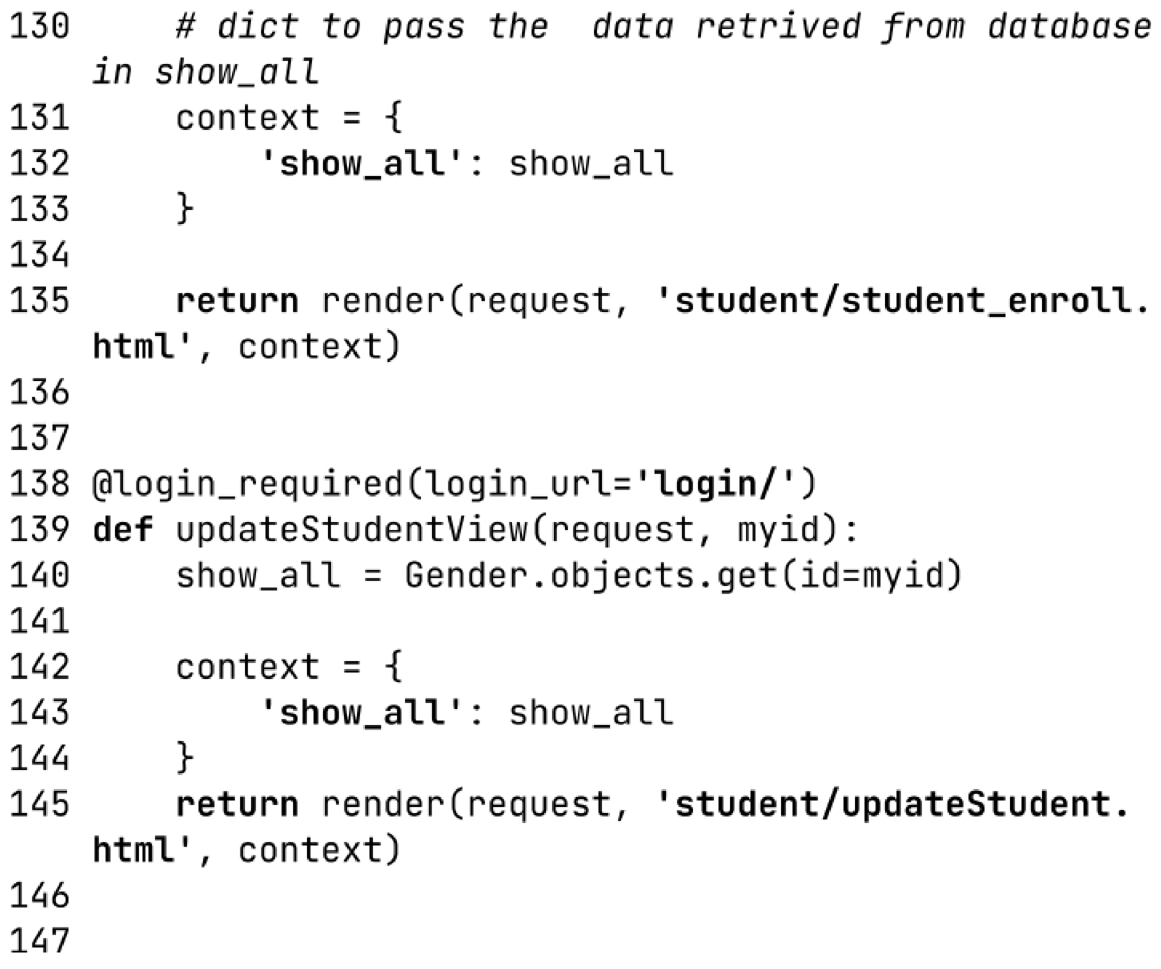
```python
student_id=student_id).exists():
            messages.warning(request, 'Student id already exists')
            return redirect('addStudent')
        elif Student_Registration.objects.filter(email=email).exists():
            messages.warning(request, 'Check email again')
            return redirect('addStudent')
        else:
            student = Student_Registration(first_name=first_name, last_name=last_name, student_id=student_id, email=email,
                                        image_upload=student_image, address=address)
            student.save()

            student_faculty = Department(faculty=faculty, lcid_id=student_id)
            student_faculty.save()

            depart_id = student_faculty.id

            student_gender = Gender(gens=gender, lcid_id=student_id, department_id_id=depart_id)
            student_gender.save()
            messages.success(request, 'Record successfully registered')
            return redirect('addStudent')
```

Figure 38: Views7



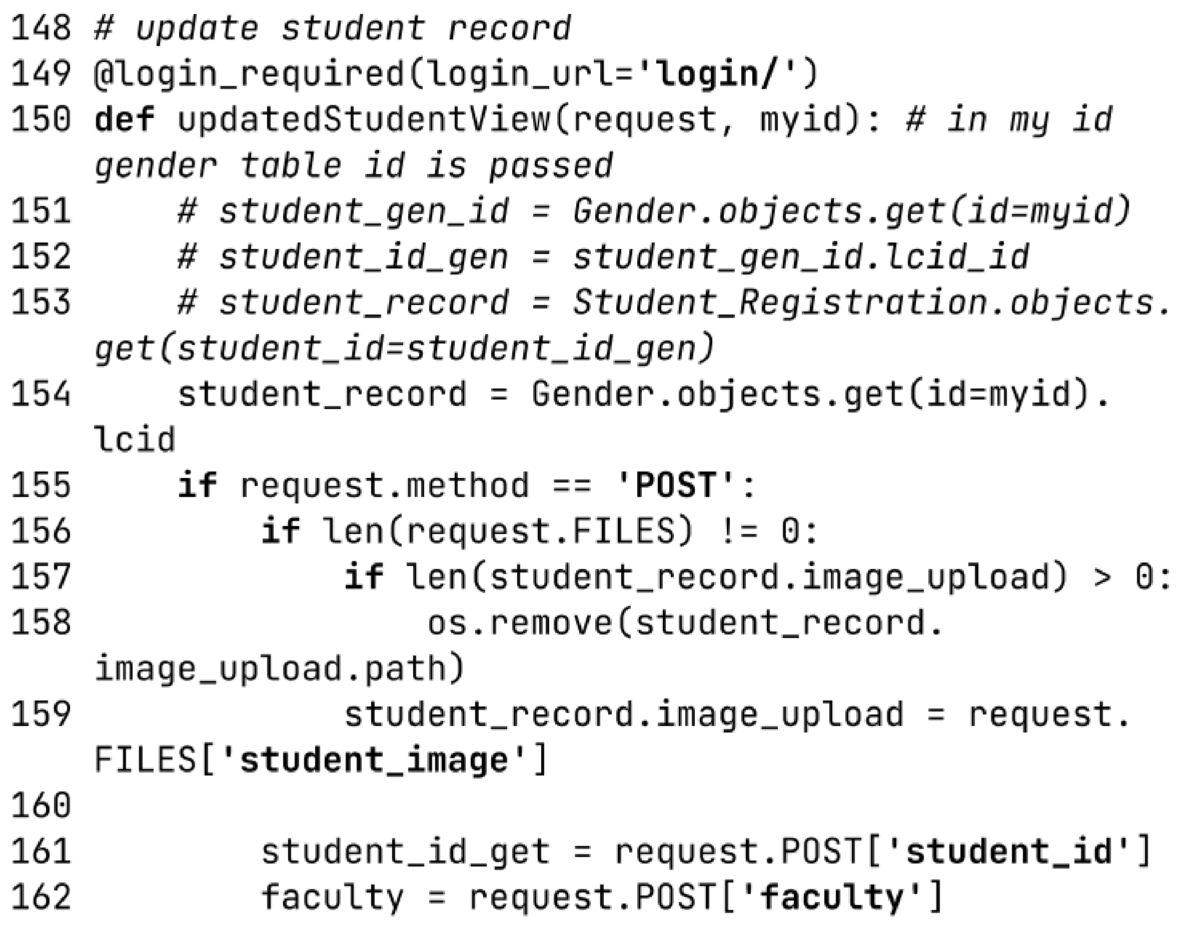
```python

    else:
        return render(request, 'student/addStudent.html')


# function for student registration
@login_required(login_url='login/')
def studentEnrollView(request):
    show_all = Gender.objects.all().select_related('lcid', 'department_id')

```

Figure 39: Views8



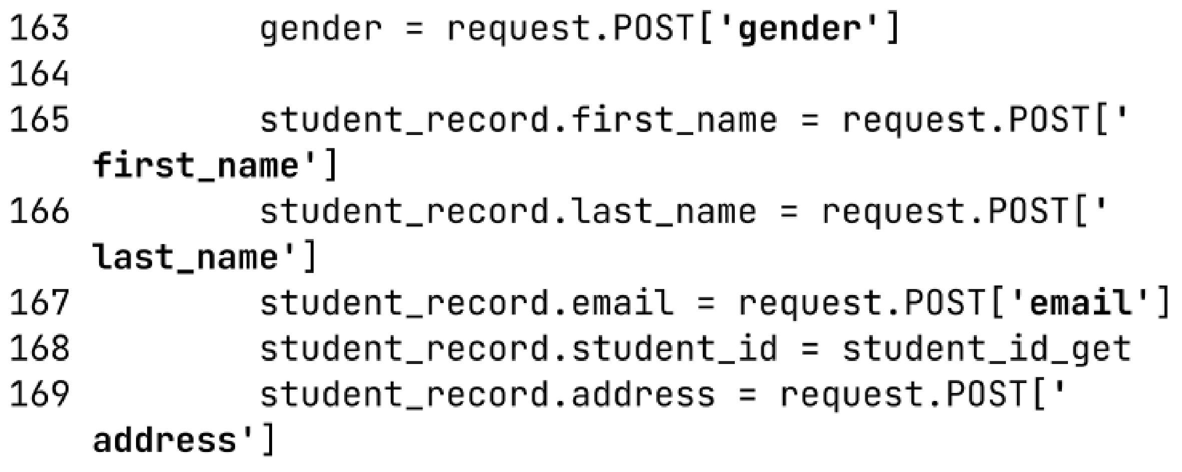
```python
    # dict to pass the  data retrived from database in show_all
    context = {
        'show_all': show_all
    }

    return render(request, 'student/student_enroll.html', context)


@login_required(login_url='login/')
def updateStudentView(request, myid):
    show_all = Gender.objects.get(id=myid)

    context = {
        'show_all': show_all
    }
    return render(request, 'student/updateStudent.html', context)


```

Figure 40: Views9



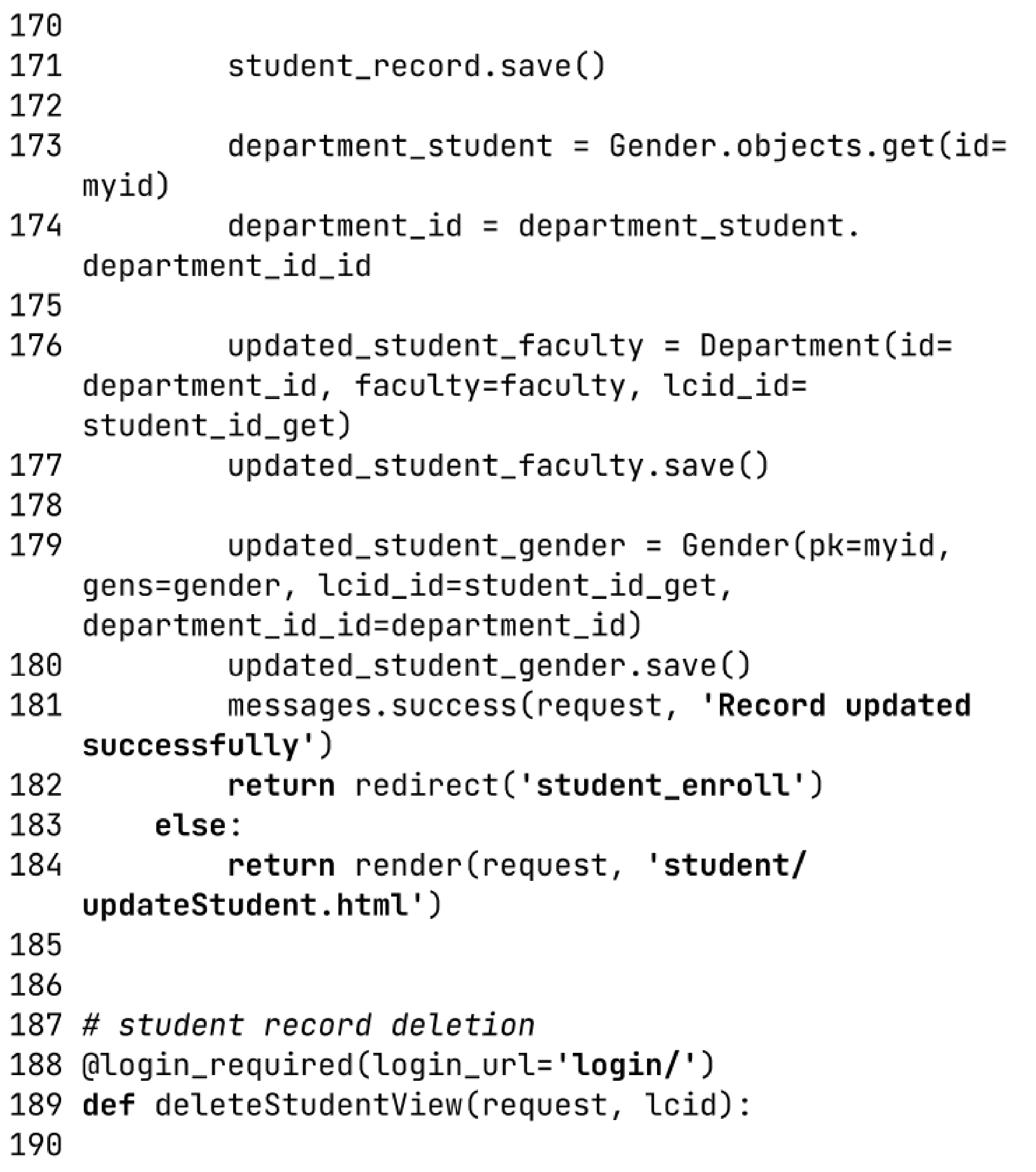
```python
# update student record
@login_required(login_url='login/')
def updatedStudentView(request, myid): # in my id gender table id is passed
    # student_gen_id = Gender.objects.get(id=myid)
    # student_id_gen = student_gen_id.lcid_id
    # student_record = Student_Registration.objects.get(student_id=student_id_gen)
    student_record = Gender.objects.get(id=myid).lcid
    if request.method == 'POST':
        if len(request.FILES) != 0:
            if len(student_record.image_upload) > 0:
                os.remove(student_record.image_upload.path)
            student_record.image_upload = request.FILES['student_image']

        student_id_get = request.POST['student_id']
        faculty = request.POST['faculty']
```

Figure 41: Views10



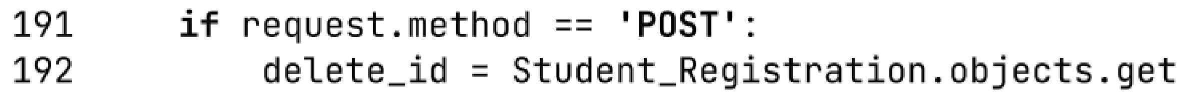
```python
        gender = request.POST['gender']

        student_record.first_name = request.POST['first_name']
        student_record.last_name = request.POST['last_name']
        student_record.email = request.POST['email']
        student_record.student_id = student_id_get
        student_record.address = request.POST['address']
```

Figure 42: Views11



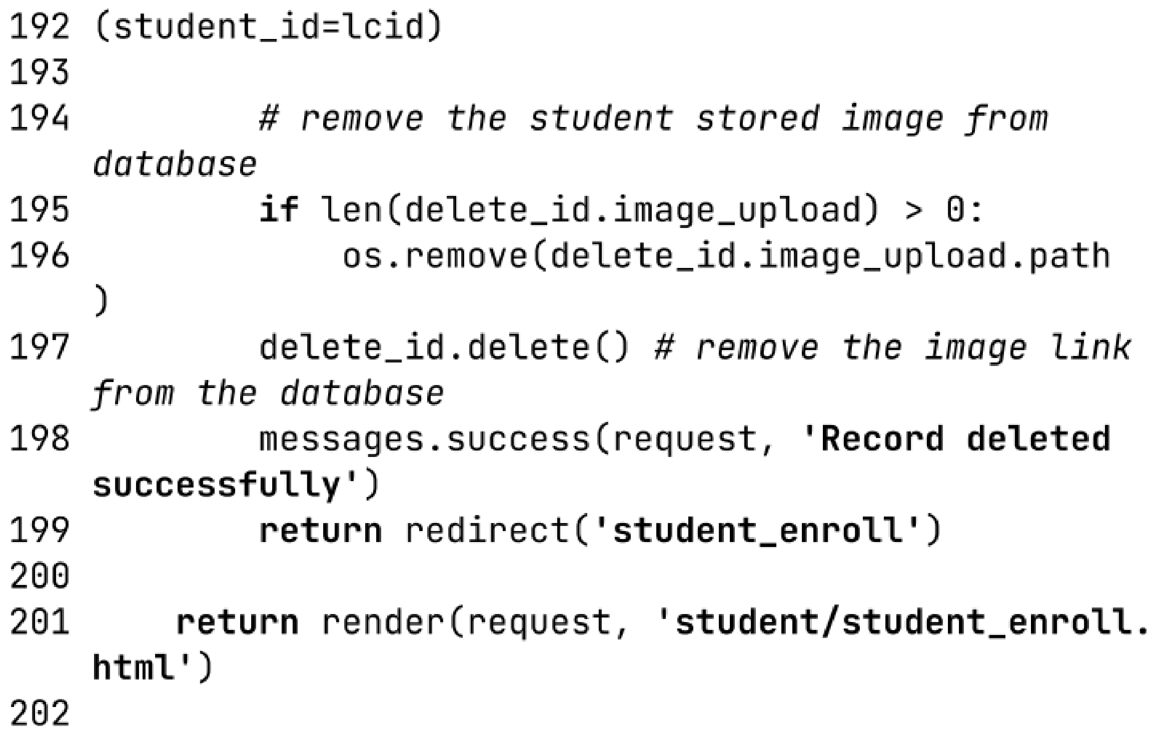
```
        student_record.save()

        department_student = Gender.objects.get(id=myid)
        department_id = department_student.department_id_id

        updated_student_faculty = Department(id=department_id, faculty=faculty, lcid_id=student_id_get)
        updated_student_faculty.save()

        updated_student_gender = Gender(pk=myid, gens=gender, lcid_id=student_id_get, department_id_id=department_id)
        updated_student_gender.save()
        messages.success(request, 'Record updated successfully')
        return redirect('student_enroll')
    else:
        return render(request, 'student/updateStudent.html')


# student record deletion
@login_required(login_url='login/')
def deleteStudentView(request, lcid):

```

Figure 43: Views12



```
    if request.method == 'POST':
        delete_id = Student_Registration.objects.get
```

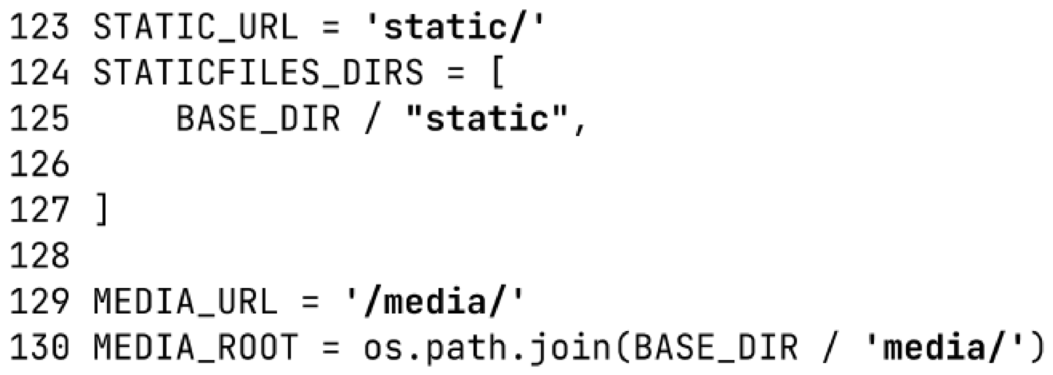
Figure 44: Views13



```
(student_id=lcid)

        # remove the student stored image from database
        if len(delete_id.image_upload) > 0:
            os.remove(delete_id.image_upload.path)
        delete_id.delete() # remove the image link from the database
        messages.success(request, 'Record deleted successfully')
        return redirect('student_enroll')

    return render(request, 'student/student_enroll.html')

```

Figure 45: Views14

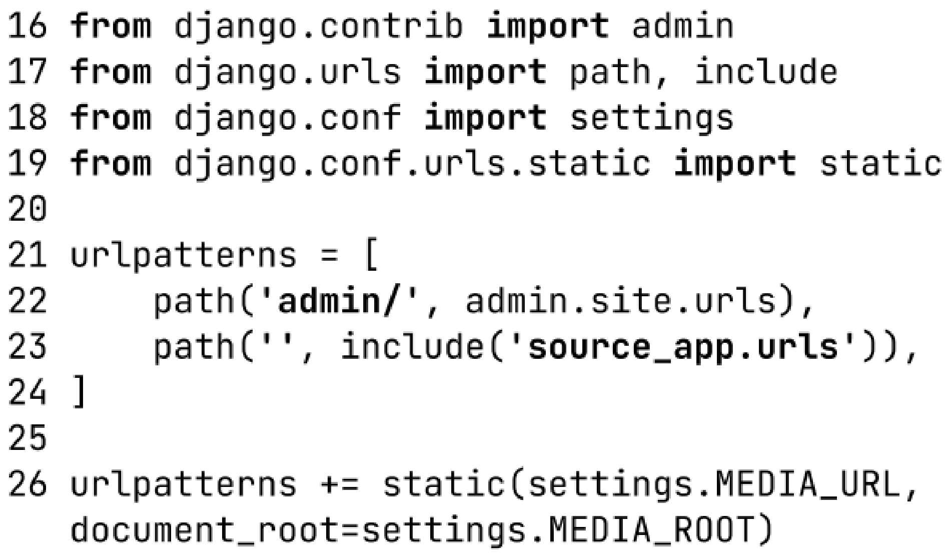
5.5 Settings.py



```
STATIC_URL = 'static/'
STATICFILES_DIRS = [
    BASE_DIR / "static",

]

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR / 'media/')
```

Figure 46: Settings

5.6 Urls.py



```python
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('source_app.urls')),
]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Figure 47: Urls

CHAPTER 6: REFERENCES

1. MyClass Attendance - Free Attendance App Last Accessed 07/25/2022
2. PPT - Life Cycle Models (Lecture 2) PowerPoint Presentation, free download - ID:202288 (slideserve.com) Last Accessed 07/29/2022
3. [PDF] Student Smart Attendance Through Face Recognition using Machine Learning Algorithm | Semantic Scholar Last Accessed 08/02/2022
4. Attendance Management system using Face Recognition – IJERT Last Accessed 08/10/2022
5. Face Recognition based Attendance System using Machine Learning (ijtsrd.com) Last Accessed 08/12/2022
6. What is Data Modelling? Types (Conceptual, Logical, Physical) (guru99.com) Last Accessed 09/03/2022
7. Django documentation | Django documentation | Django (djangoproject.com) Last Accessed 09/01/2022
8. Diagram Software and Flowchart Maker Last Accessed 09/03/2022
9. UML Sequence Diagram for beginner with Solved Example in Hindi | SOOAD Series - YouTube Use Case Diagram Video Last Accessed 09/03/2022
10. Sequence Diagram - Step by Step Guide with Example - YouTube Sequential Diagram Video Last Accessed 09/04/2022
11. UML State chart Diagram with solved Example(HINDI) || IGNOU || MCS-032 - YouTube State Chart Diagram Video Last Accessed in 09/05/2022

Revised Gantt Chart

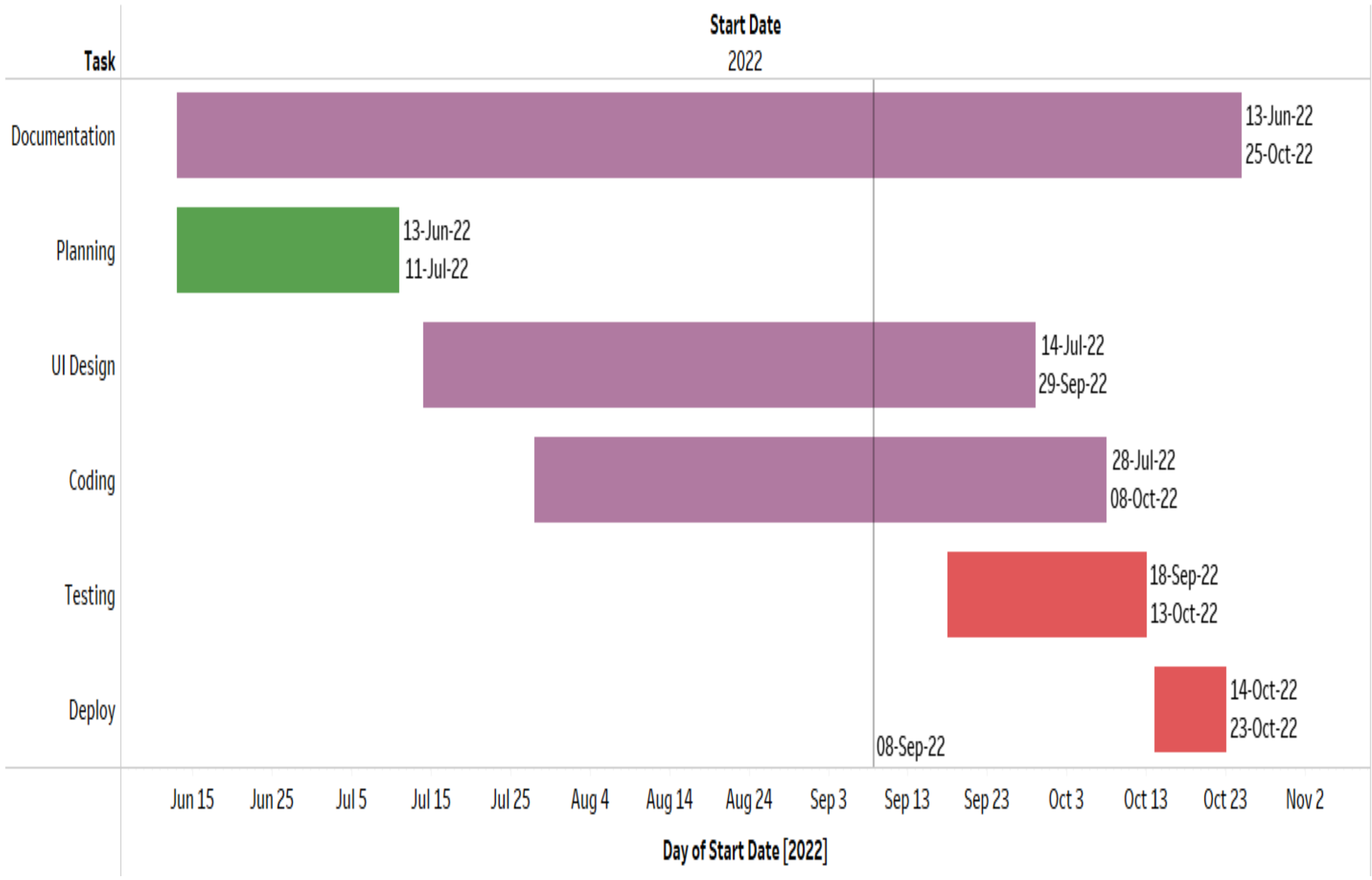


Figure 48: Revised Gantt Chart