Content Learning Management
System for RAD and FLEX Students
at Phinma University of Iloilo

A Project presented to the
College of Information
Technology Education PHINMA
University of Iloilo
Rizal, Iloilo City

Submitted in partial fulfillment of the
requirements for the degree of Bachelor of
Science in Information Technology
(IT Project Management, Information Systems (including
Database Fundamentals), and Object-Oriented Programming)

[*MEMBERS NAME*]

Buenafe,Nelmar

Cabanig,John Lloyd

Sablon,Rona

III Suamen,Maximo

Villanueva,Alyssa Marie

Simpao Maessy

October 2024

I. Project Description

The project is about building a flexible and interactive study system for students at Phinma University of Iloilo who follow RAD and FLEX learning formats. The system will let students log in using their student ID and birthdate, track their academic progress, and take video lessons or documents to help them understand where they need improvement. It also allows teachers to monitor student performance and offer support when needed. The goal is to make learning easier and more effective, helping students stay engaged and better manage their studies across both in-person and online formats.

Project Objectives

General Objectives

Develop a comprehensive Content Learning Management System (CLMS) that facilitates the seamless organization, management, and access to video lessons for both teachers and students, enhancing the educational experience through efficient content delivery and user-friendly interaction.

Specific Objectives

Specifically, this project aims to develop the following:

1. Develop a user-friendly platform where teachers can upload, organize, manage video lessons and documents.
2. Create a video playlist where uploaded video lessons are stored securely and can be accessed and modified by teachers at any time.
3. Provide student access to video lessons through a dedicated portal that displays the relevant videos for the lessons they are enrolled in, ensuring easy navigation and playback of lessons.
4. Enable lesson update functionality so teachers can easily update or replace old video.

II. Scope and Delimitation

The project will create a study system for Phinma University of Iloilo's RAD and FLEX students, helping them track their progress and watch video lessons / documents. Students can log in using their ID and birthdate, while teachers can monitor how they're doing.

- The system will only work within the university and will focus on subjects taught at the school. It’s designed specifically for Phinma students and won’t include outside content or work outside the campus network.

- The learning materials within the system will be focused exclusively on the curriculum
  established by the university, ensuring alignment with the academic standards and
  requirements of Phinma UI.

III. Software Development Model

In developing the Content Learning Management System (CLMS), the team chose the Agile Development Model because it allows for flexibility and regular updates. Agile breaks the project into smaller tasks that can be worked on in stages, making it easier to build and improve features gradually. The teacher could first create the login feature and later add video uploading, student tracking, and document uploads. With each new feature, we get feedback and make changes as needed. This way, the system stays flexible and adapts to the needs of both teachers and students as the project grows.

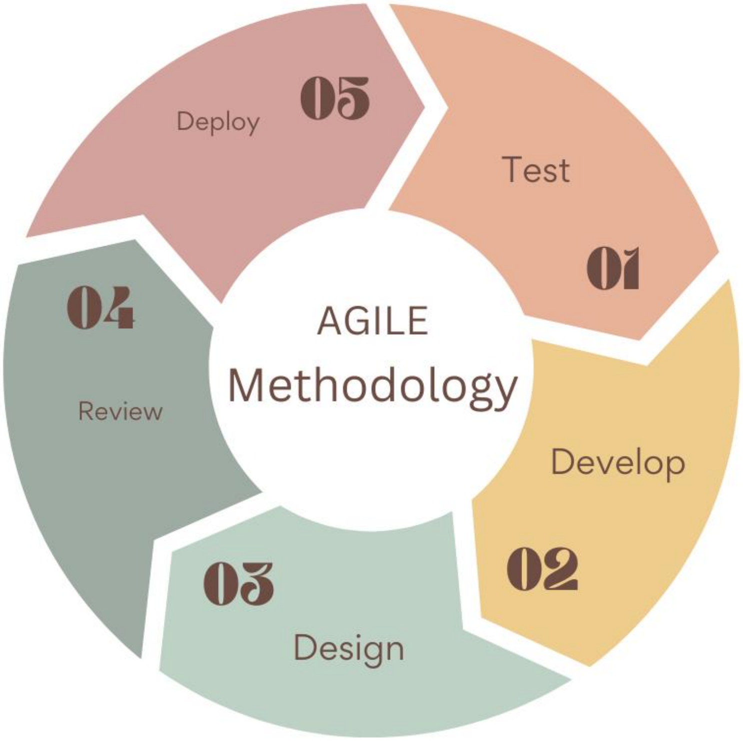


Figure 1: Agile Development Cycle

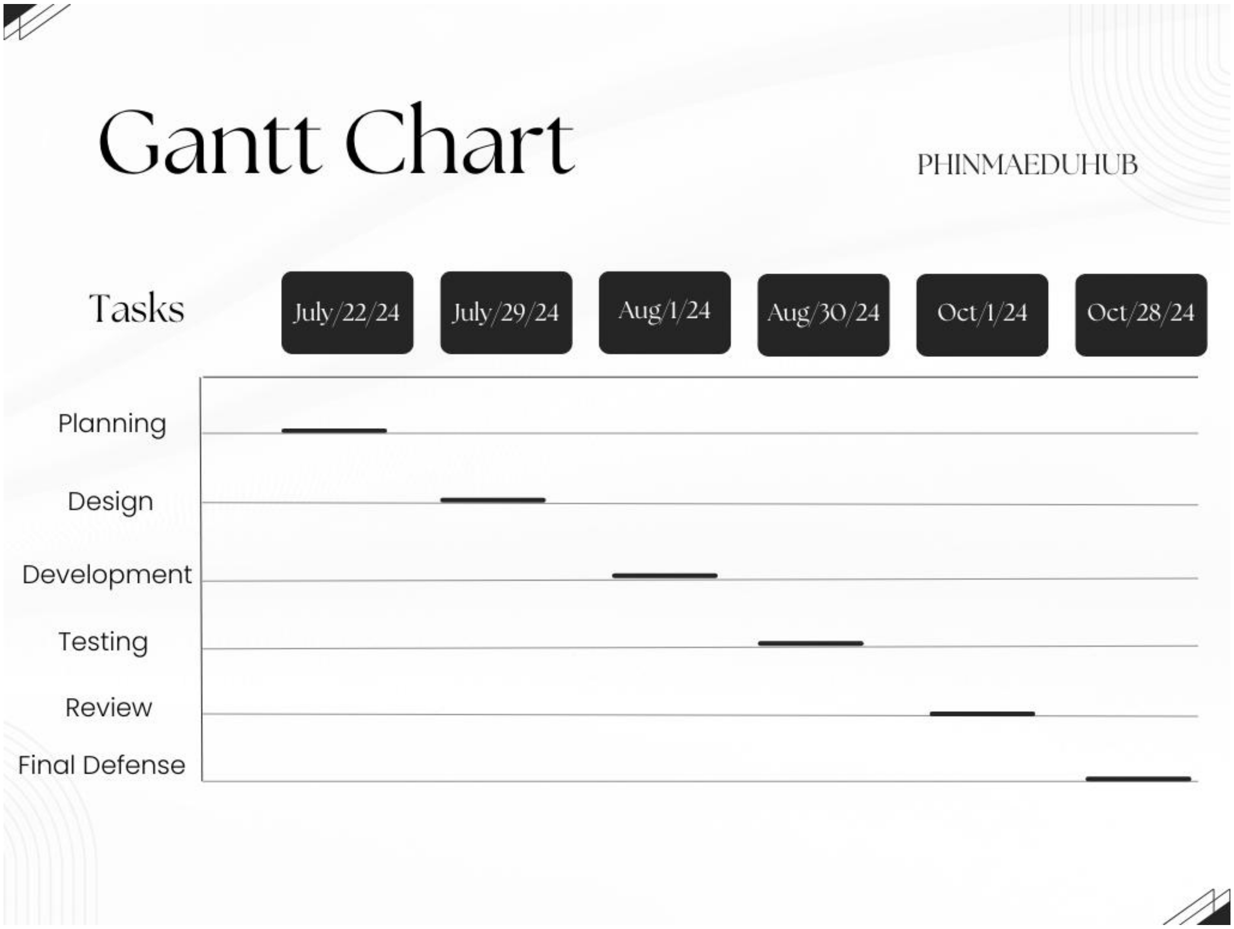
Explain the Agile part what is Planning, Deploy, Test, Maintenance, and Review

IV. Project Timeline

List the major project milestones and the required delivery dates. A ‘milestone’ is a significant event or stage to be completed. Explain why each milestone is critical to the project, as follows: *(Gantt Chart)*

Milestone	Date©	Description
Planning	22/7/24	We made a planning together with my members.
Design	29/7/24	Completed the design of the application, including layout and user interface.
Development	1/8/24	Started the development phase, coding the main functionalities.
Testing	30/8/24	Conducted thorough testing to ensure the system works as expected.
Review	1/10/24	Gathered feedback from users and made necessary improvements
Final Defense	28/10/24	Presented the project to panelist, answering questions, addressing feedback and demonstrating the application's functionality

Graph 1: Gantt Chart



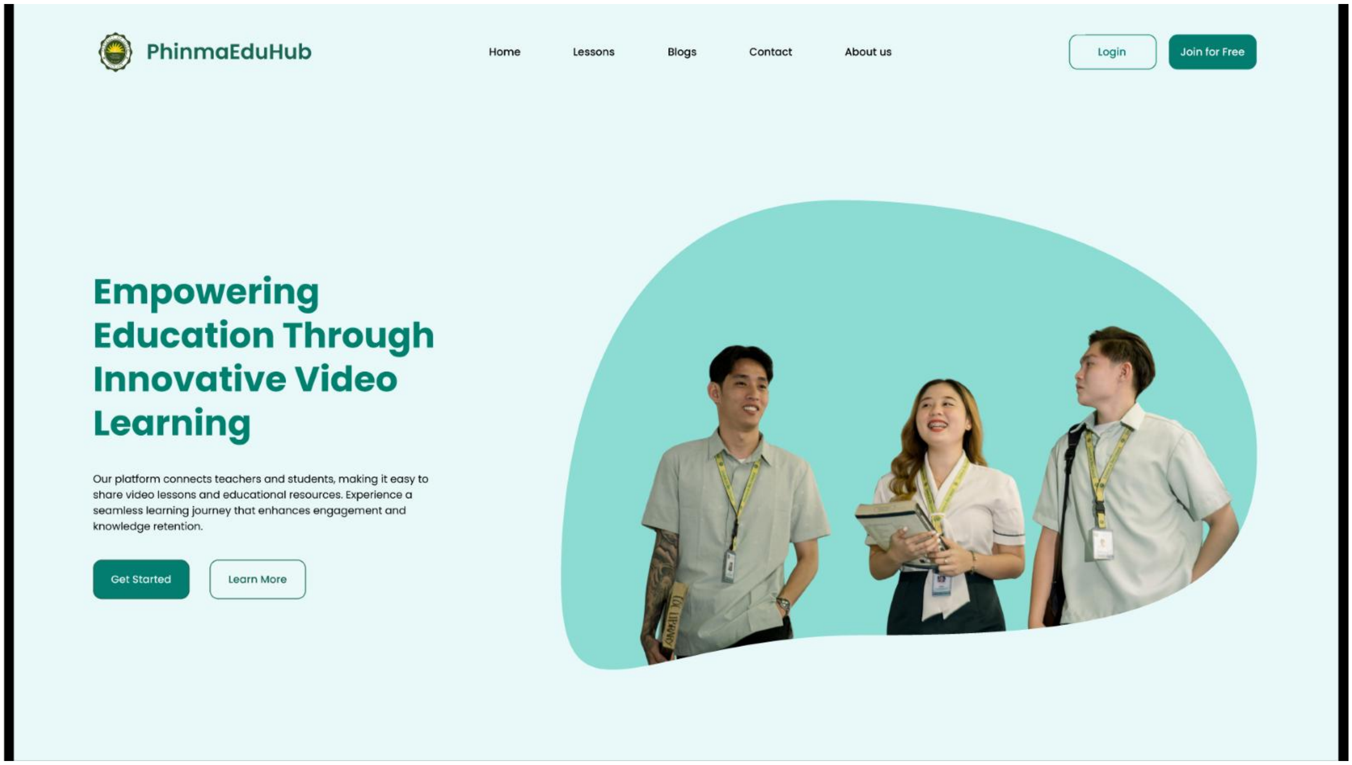
V. Project Team Roles and Responsibilities

This section lists the people involved in the process and their roles.

Name	Role	Responsibilities
Nelmar Buenafe	Project Manager	He will oversee the project, coordinating team efforts, managing the timeline, addressing issues, and communicating with stakeholders to ensure it stays on track and meets its goals.
John Lloyd Cabanig	Programmer	He is responsible for coding, developing application, debug error and testing the system.
Rona Sablon	Documentator	She will manage all aspects of the project life cycle, including meeting minutes, schedules, and user manuals, to ensure accurate and accessible record-keeping.
Maximo Suamen III	Designer	He will be responsible for the visual and user experience design of the application, including layout creation, color scheme selection, icon and graphics design, and ensuring aesthetic appeal.
Alyssa Marie Villanueva	Documentator	She will manage some of our documents same with other documentator.
Simpao Maessy	Researcher	She will research information about the project and share her findings with the team to help ensure that everything meets the project’s goals.

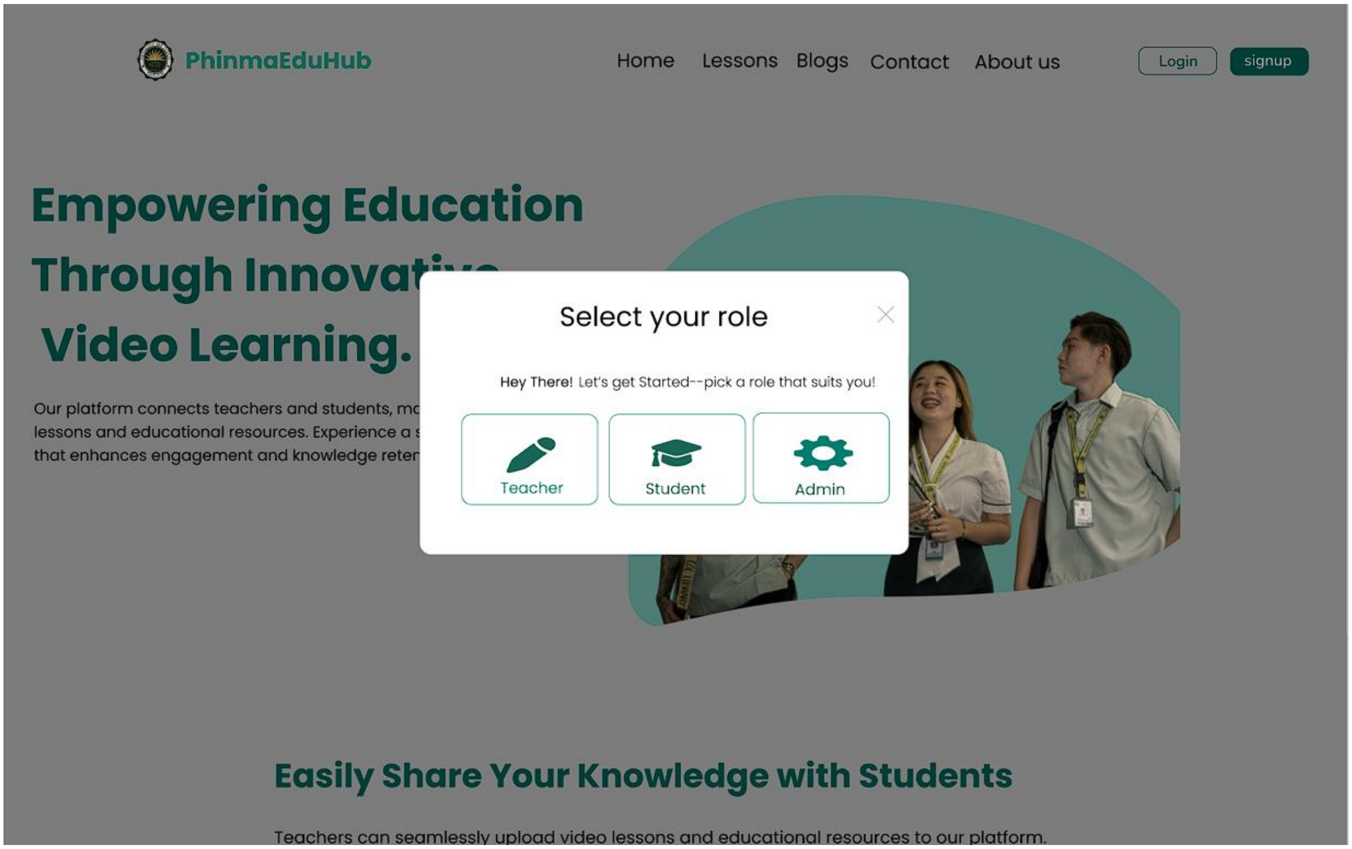
VI. Implementation

Wireframe/Figma



Empowering Education Through Innovative Video Learning

Our platform connects teachers and students, making it easy to share video lessons and educational resources. Experience a seamless learning journey that enhances engagement and knowledge retention.



Empowering Education Through Innovative Video Learning.

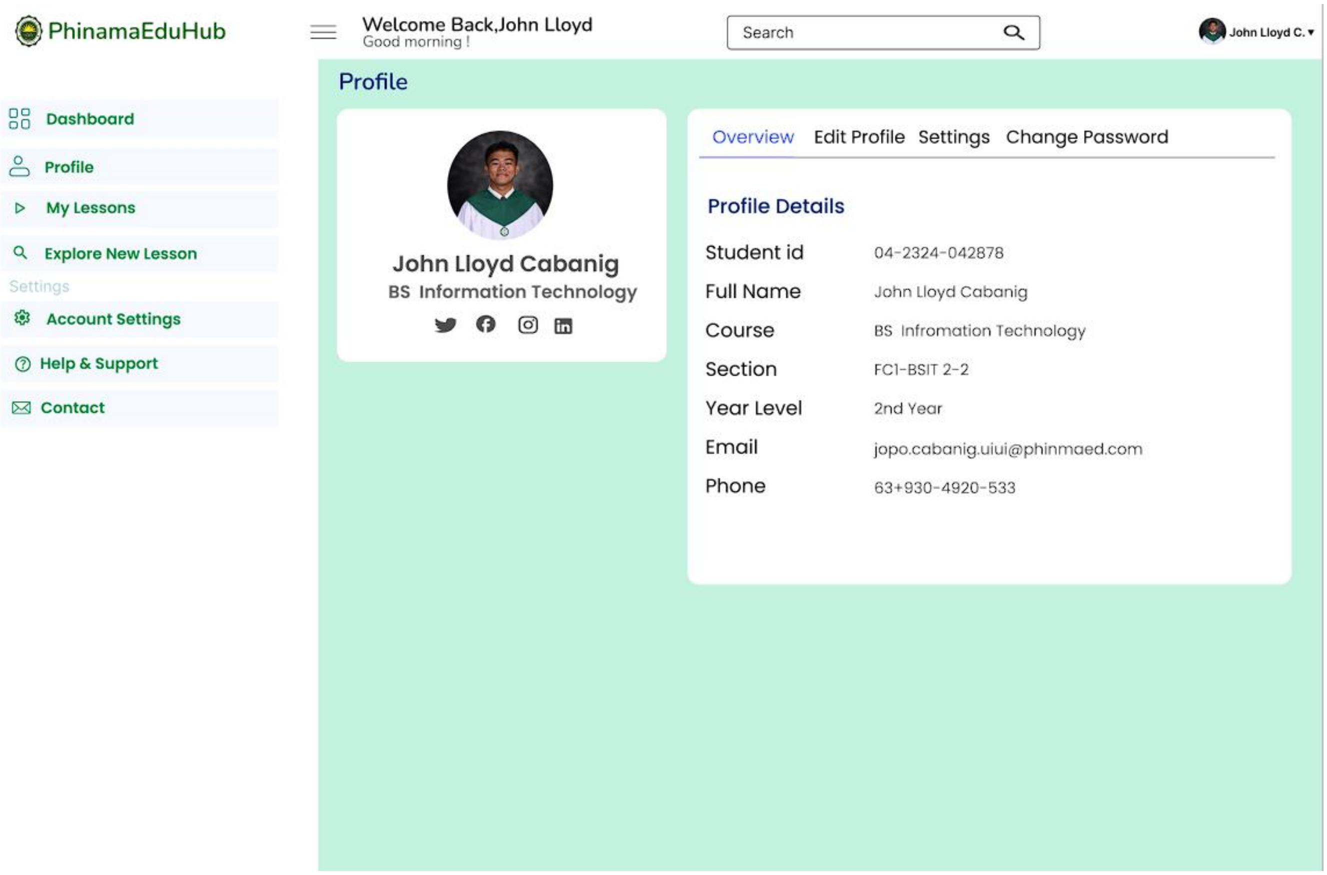
Our platform connects teachers and students, ma lessons and educational resources. Experience a s that enhances engagement and knowledge reter

Select your role

Hey There! Let's get Started--pick a role that suits you!

Easily Share Your Knowledge with Students

Teachers can seamlessly upload video lessons and educational resources to our platform.



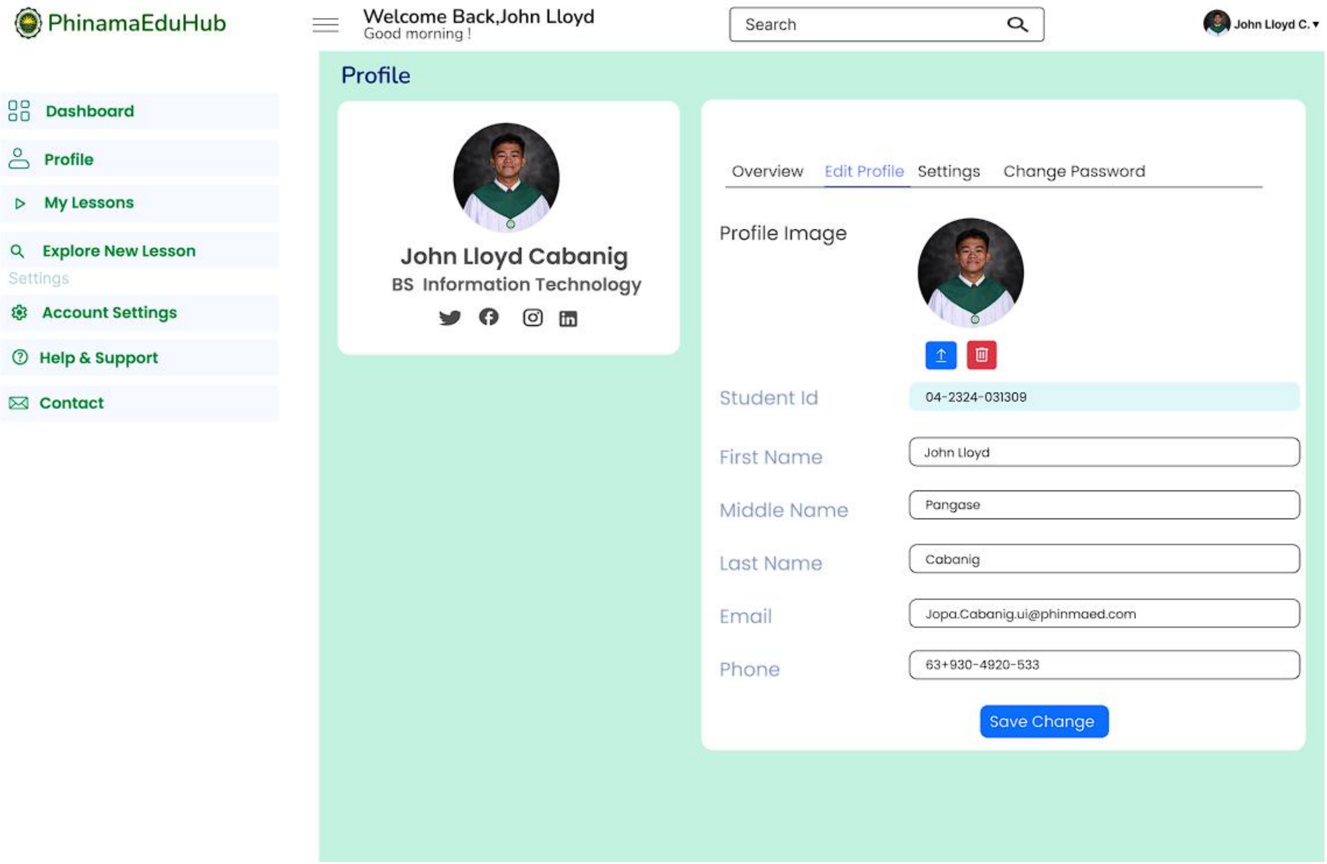
Welcome Back,John Lloyd
Good morning !

Profile

John Lloyd Cabanig
BS Information Technology

Profile Details

Student id	04-2324-042878
Full Name	John Lloyd Cabanig
Course	BS Infromation Technology
Section	FC1-BSIT 2-2
Year Level	2nd Year
Email	jopo.cabanig.uiui@phinmaed.com
Phone	63+930-4920-533



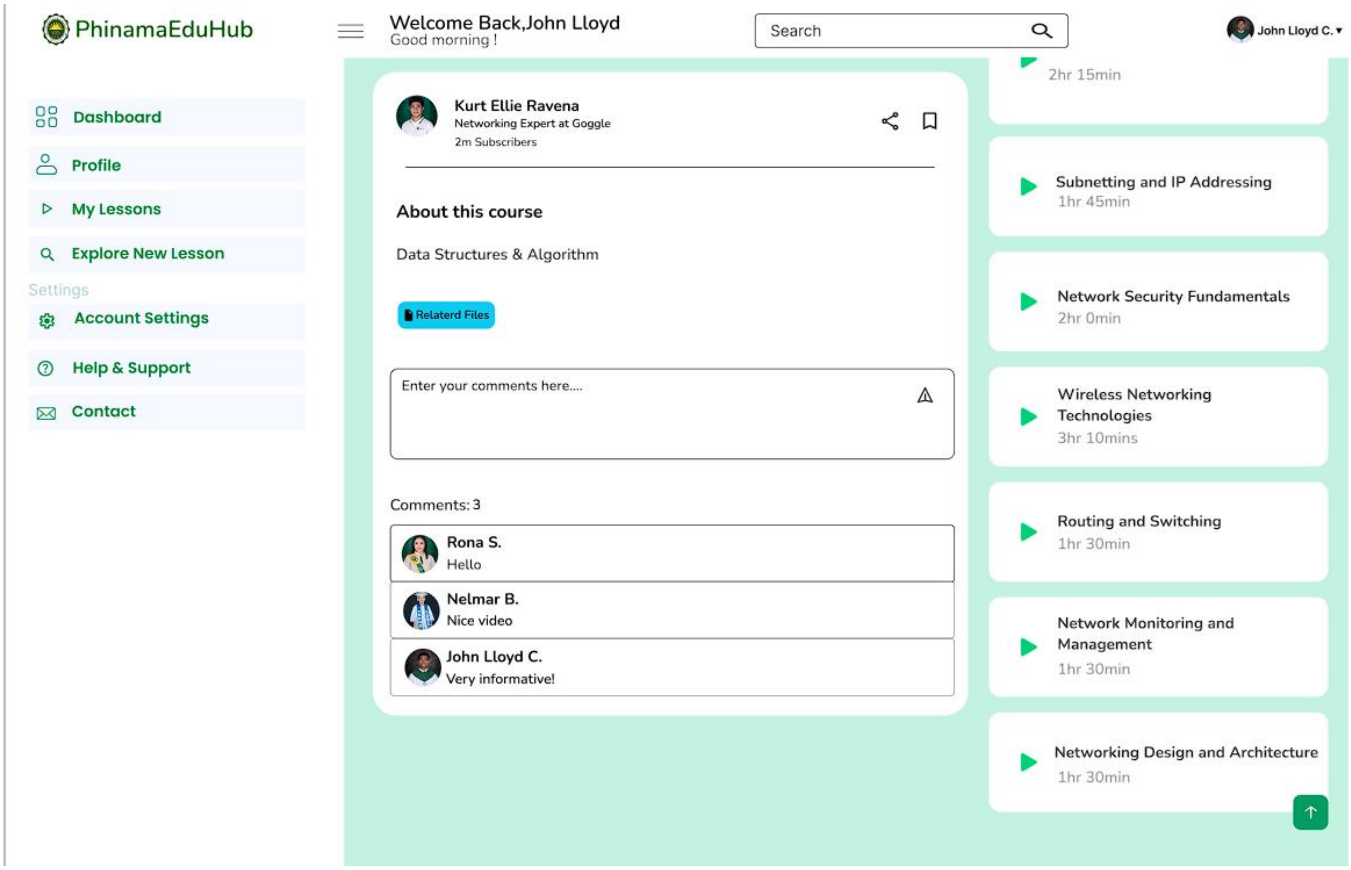
Welcome Back,John Lloyd
Good morning !

Profile

John Lloyd Cabanig
BS Information Technology

Profile Image

Student Id	04-2324-031309
First Name	John Lloyd
Middle Name	Pangase
Last Name	Cabanig
Email	Jopa.Cabanig.ui@phinmaed.com
Phone	63+930-4920-533



Welcome Back,John Lloyd
Good morning !

Kurt Ellie Ravena
Networking Expert at Goggle
2m Subscribers

About this course

Data Structures & Algorithm

Enter your comments here....

Comments: 3

Rona S.
Hello

Nelmar B.
Nice video

John Lloyd C.
Very informative!

2hr 15min

Subnetting and IP Addressing
1hr 45min

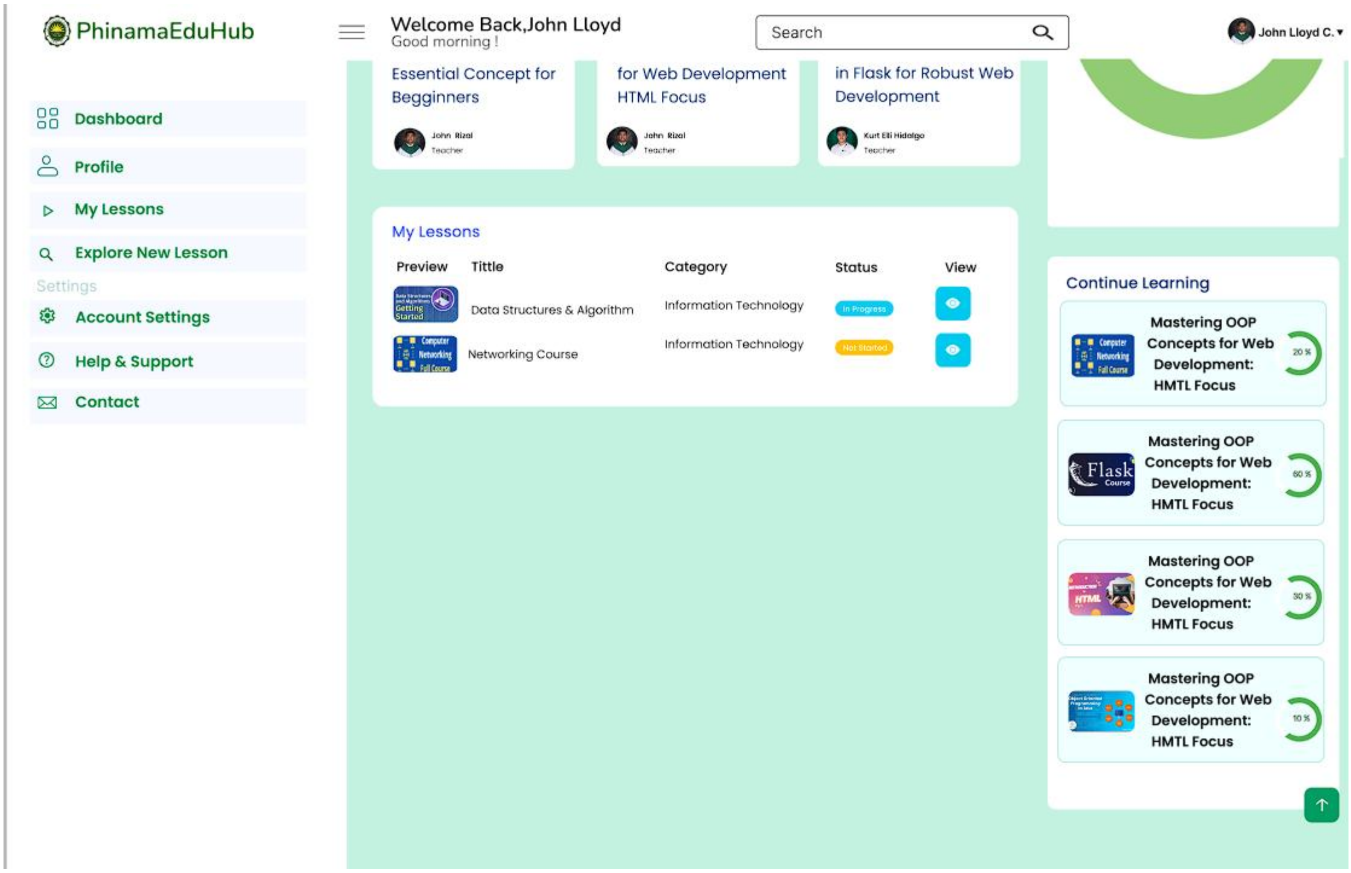
Network Security Fundamentals
2hr 0min

Wireless Networking Technologies
3hr 10mins

Routing and Switching
1hr 30min

Network Monitoring and Management
1hr 30min

Networking Design and Architecture
1hr 30min



Welcome Back,John Lloyd
Good morning !

Essential Concept for Begginners
John Rizal
Teacher

for Web Development HTML Focus
John Rizal
Teacher

in Flask for Robust Web Development
Kurt Elli Hidalgo
Teacher

My Lessons

Preview	Tittle	Category	Status	View
	Data Structures & Algorithm	Information Technology	In Progress	
	Networking Course	Information Technology	Not Started	

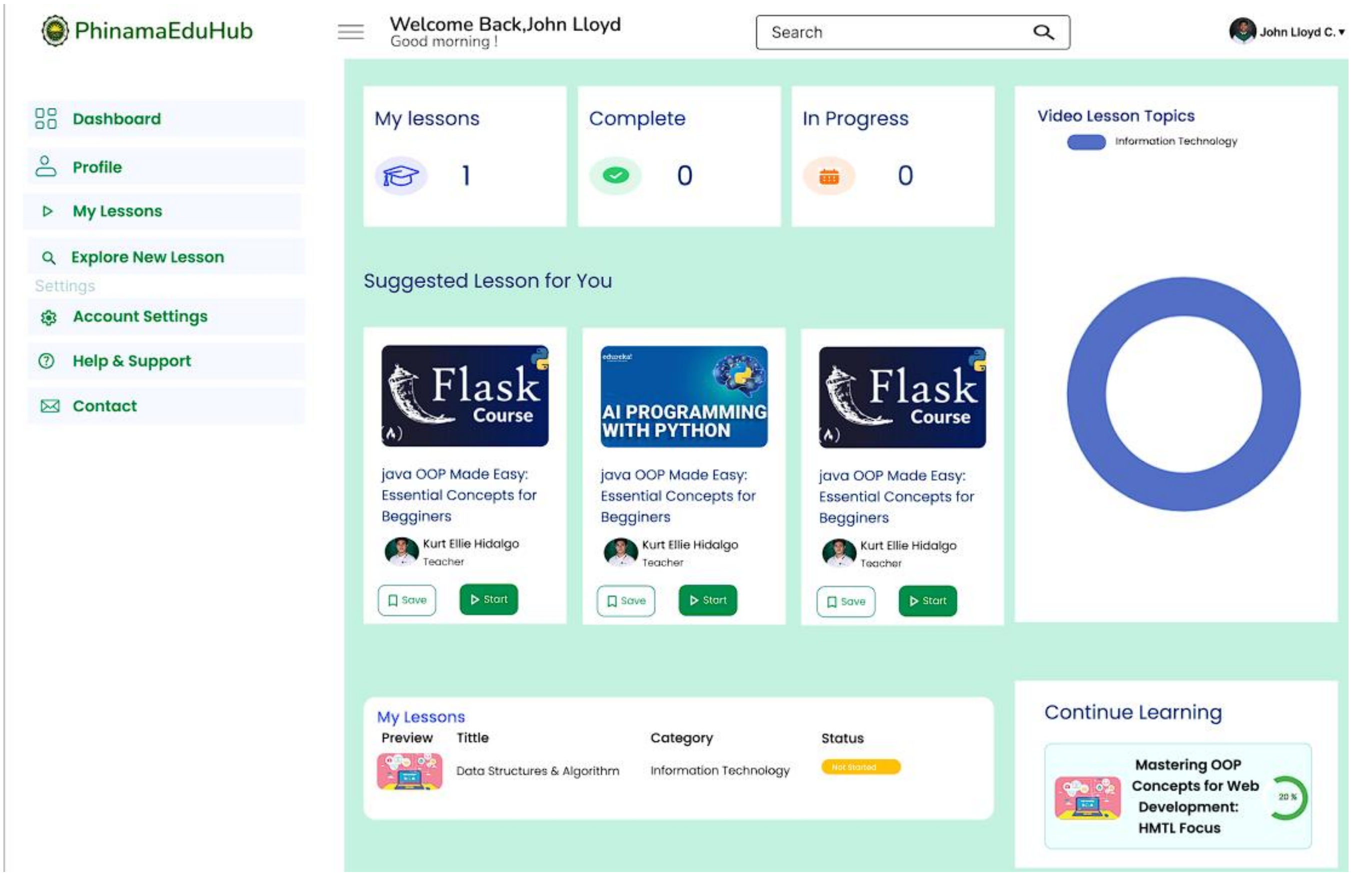
Continue Learning

Mastering OOP Concepts for Web Development: HMTL Focus

Mastering OOP Concepts for Web Development: HMTL Focus

Mastering OOP Concepts for Web Development: HMTL Focus

Mastering OOP Concepts for Web Development: HMTL Focus



PhinamaEduHub

Welcome Back,John Lloyd
Good morning !

Search

John Lloyd C. ▾

- Dashboard
- Profile
- My Lessons
- Explore New Lesson

Settings

- Account Settings
- Help & Support
- Contact

My lessons 1

Complete 0

In Progress 0

Video Lesson Topics

● Information Technology

Suggested Lesson for You

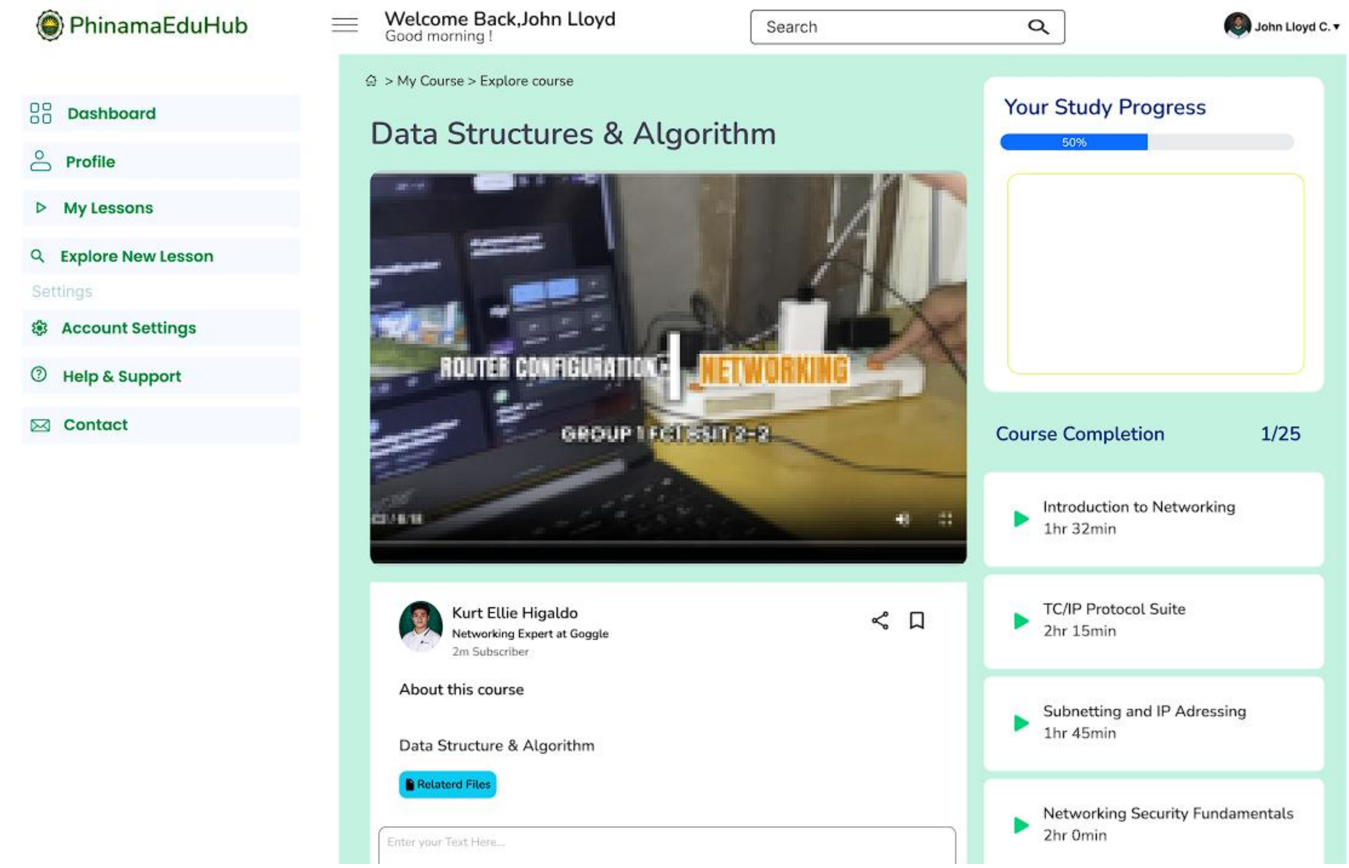
Flask Course	AI PROGRAMMING WITH PYTHON	Flask Course
java OOP Made Easy: Essential Concepts for Begginers	java OOP Made Easy: Essential Concepts for Begginers	java OOP Made Easy: Essential Concepts for Begginers
Kurt Ellie Hidalgo Teacher	Kurt Ellie Hidalgo Teacher	Kurt Ellie Hidalgo Teacher
Save · Start	Save · Start	Save · Start

My Lessons

Preview	Tittle	Category	Status
	Data Structures & Algorithm	Information Technology	Not Started

Continue Learning

Mastering OOP Concepts for Web Development: HMTL Focus — 20 %



PhinamaEduHub

Welcome Back,John Lloyd
Good morning !

Search

John Lloyd C. ▾

- Dashboard
- Profile
- My Lessons
- Explore New Lesson

Settings

- Account Settings
- Help & Support
- Contact

⌂ > My Course > Explore course

Data Structures & Algorithm

Kurt Ellie Higaldo
Networking Expert at Goggle
2m Subscriber

About this course

Data Structure & Algorithm

Relaterd Files

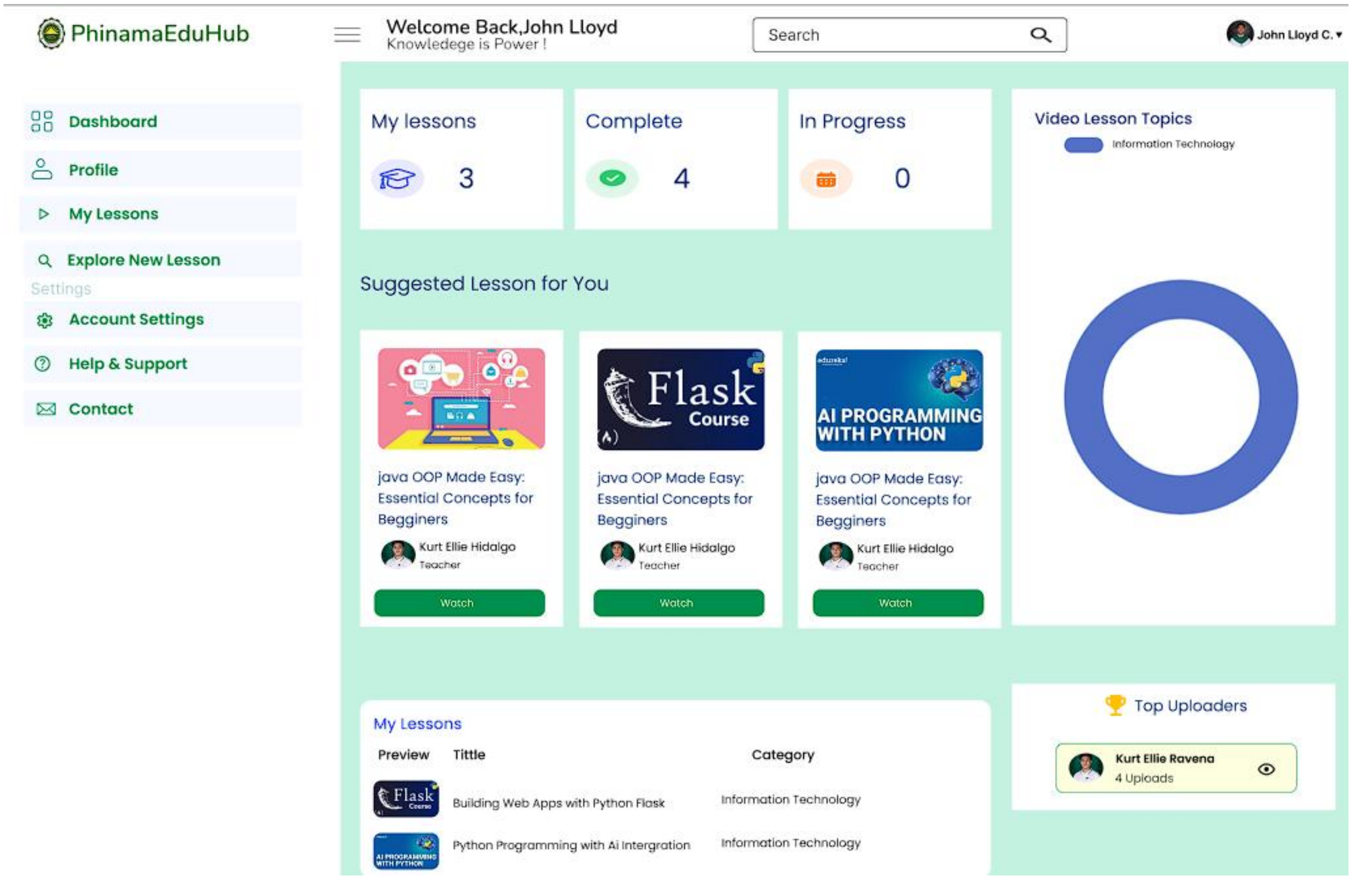
Enter your Text Here...

Your Study Progress

50%

Course Completion 1/25

- ▶ Introduction to Networking 1hr 32min
- ▶ TC/IP Protocol Suite 2hr 15min
- ▶ Subnetting and IP Adressing 1hr 45min
- ▶ Networking Security Fundamentals 2hr 0min



PhinamaEduHub

Welcome Back,John Lloyd
Knowledege is Power !

Search

John Lloyd C. ▾

- Dashboard
- Profile
- My Lessons
- Explore New Lesson

Settings

- Account Settings
- Help & Support
- Contact

My lessons	Complete	In Progress
3	4	0

Video Lesson Topics

🟦 Information Technology

Suggested Lesson for You

java OOP Made Easy: Essential Concepts for Begginers
Kurt Ellie Hidalgo
Teacher
Watch

java OOP Made Easy: Essential Concepts for Begginers
Kurt Ellie Hidalgo
Teacher
Watch

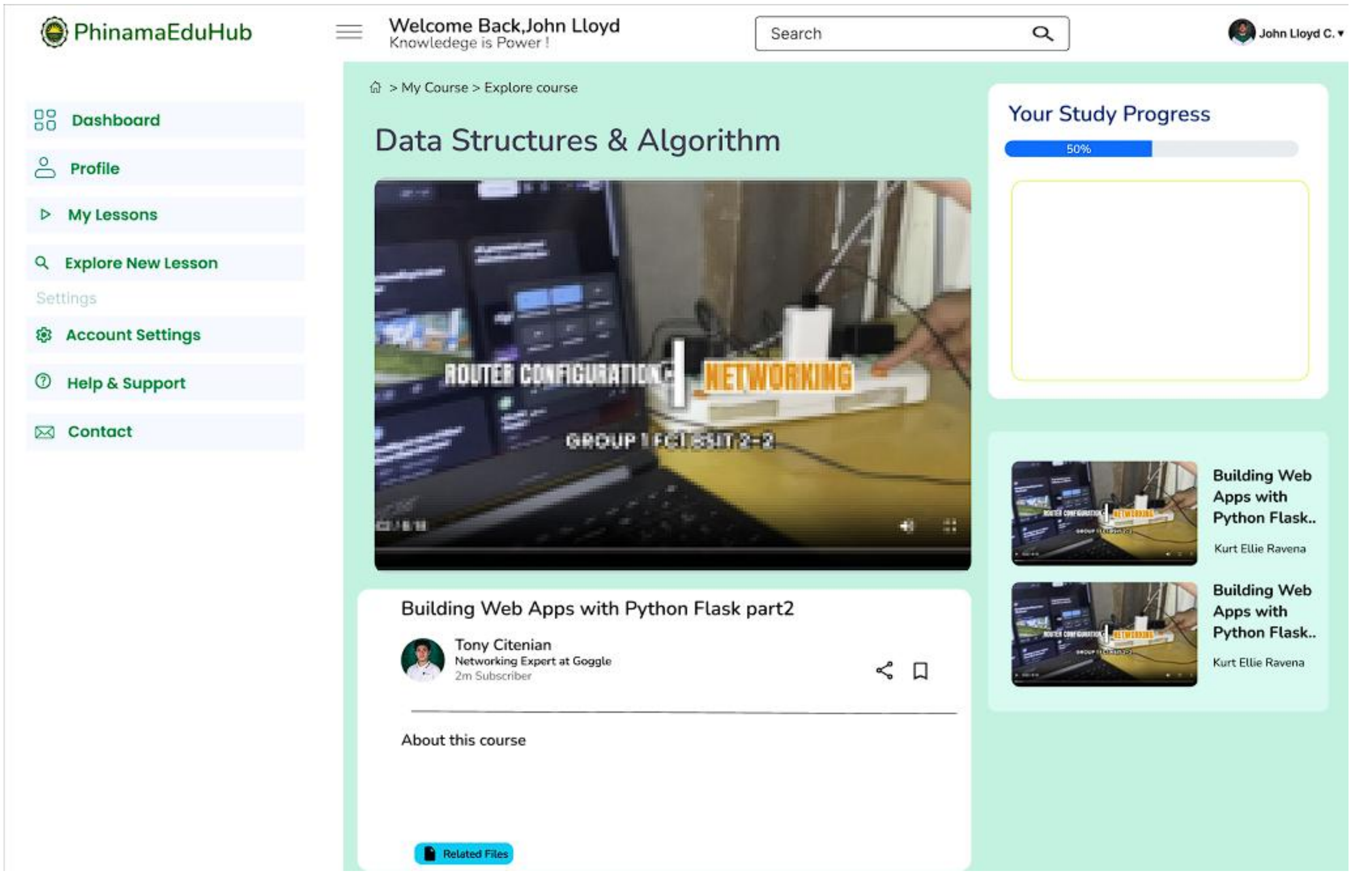
java OOP Made Easy: Essential Concepts for Begginers
Kurt Ellie Hidalgo
Teacher
Watch

My Lessons

Preview	Tittle	Category
Flask Course	Building Web Apps with Python Flask	Information Technology
AI PROGRAMMING WITH PYTHON	Python Programming with Ai Intergration	Information Technology

Top Uploaders

Kurt Ellie Ravena
4 Uploads



PhinamaEduHub

Welcome Back,John Lloyd
Knowledege is Power !

Search

John Lloyd C. ▾

- Dashboard
- Profile
- My Lessons
- Explore New Lesson

Settings

- Account Settings
- Help & Support
- Contact

> My Course > Explore course

Data Structures & Algorithm

Building Web Apps with Python Flask part2

Tony Citenian
Networking Expert at Goggle
2m Subscriber

About this course

Related Files

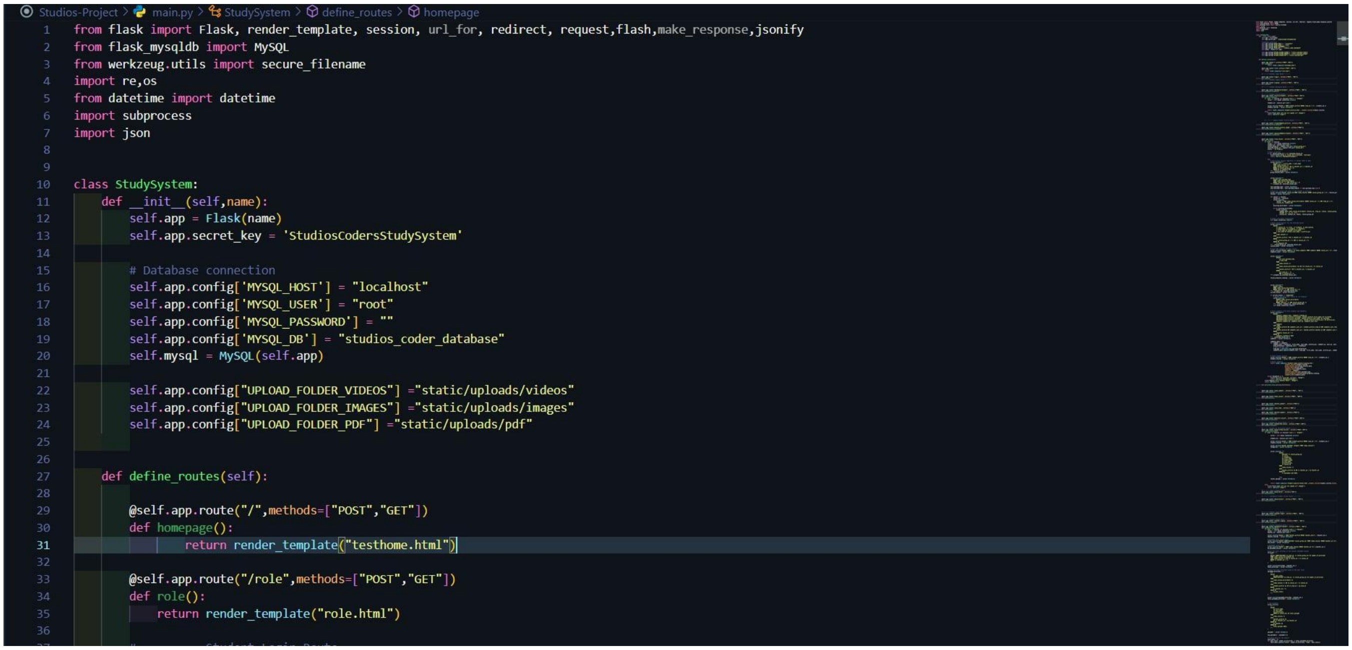
Your Study Progress

50%

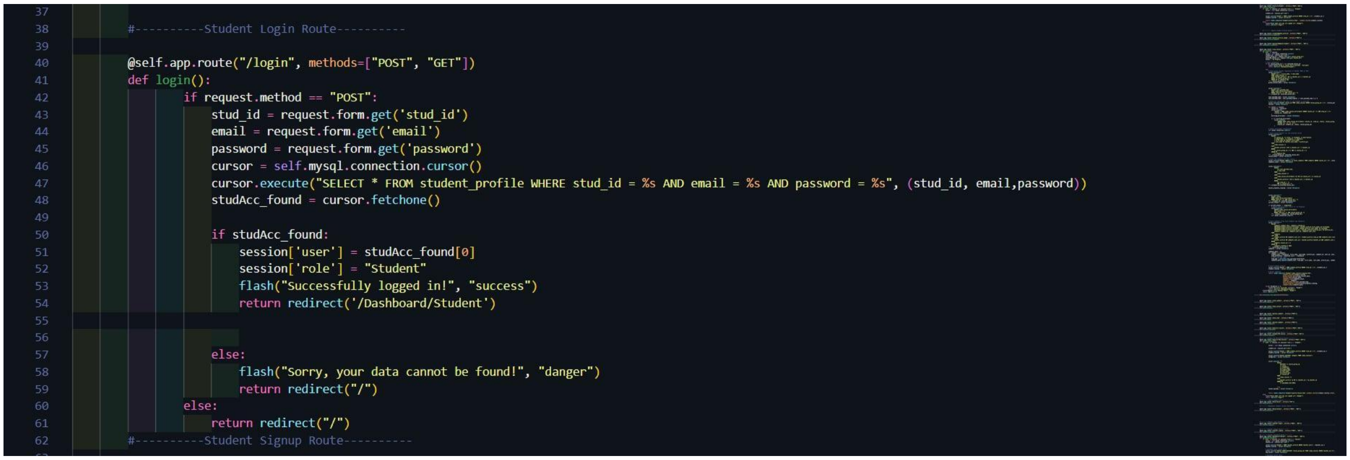
Building Web Apps with Python Flask..
Kurt Ellie Ravena

Building Web Apps with Python Flask..
Kurt Ellie Ravena

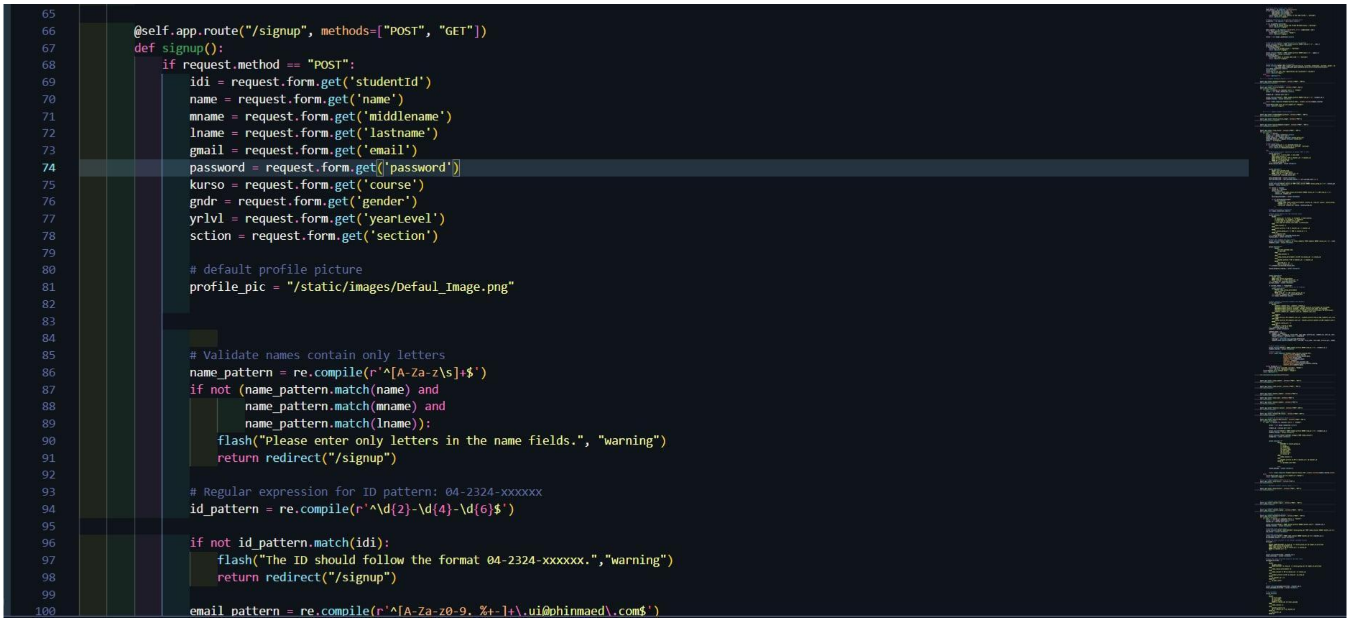
Python Flask code


Studios-Project > main.py > StudySystem > define_routes > homepage

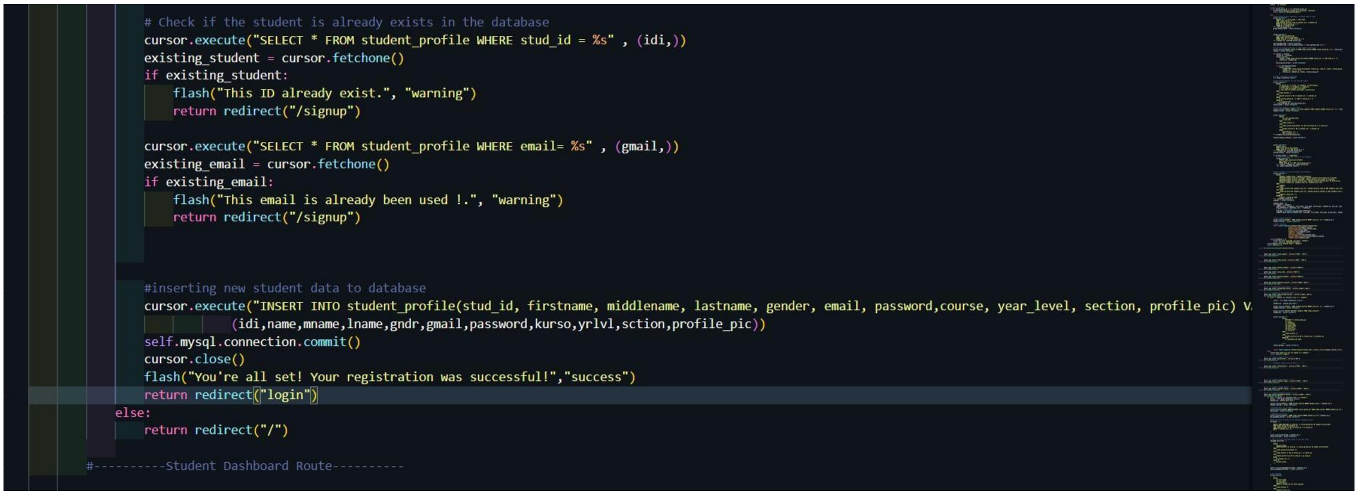
```python
from flask import Flask, render_template, session, url_for, redirect, request,flash,make_response,jsonify
from flask_mysqldb import MySQL
from werkzeug.utils import secure_filename
import re,os
from datetime import datetime
import subprocess
import json


class StudySystem:
    def __init__(self,name):
        self.app = Flask(name)
        self.app.secret_key = 'StudiosCodersStudySystem'

        # Database connection
        self.app.config['MYSQL_HOST'] = "localhost"
        self.app.config['MYSQL_USER'] = "root"
        self.app.config['MYSQL_PASSWORD'] = ""
        self.app.config['MYSQL_DB'] = "studios_coder_database"
        self.mysql = MySQL(self.app)

        self.app.config["UPLOAD_FOLDER_VIDEOS"] ="static/uploads/videos"
        self.app.config["UPLOAD_FOLDER_IMAGES"] ="static/uploads/images"
        self.app.config["UPLOAD_FOLDER_PDF"] ="static/uploads/pdf"


    def define_routes(self):

        @self.app.route("/",methods=["POST","GET"])
        def homepage():
                return render_template("testhome.html")

        @self.app.route("/role",methods=["POST","GET"])
        def role():
            return render_template("role.html")

```



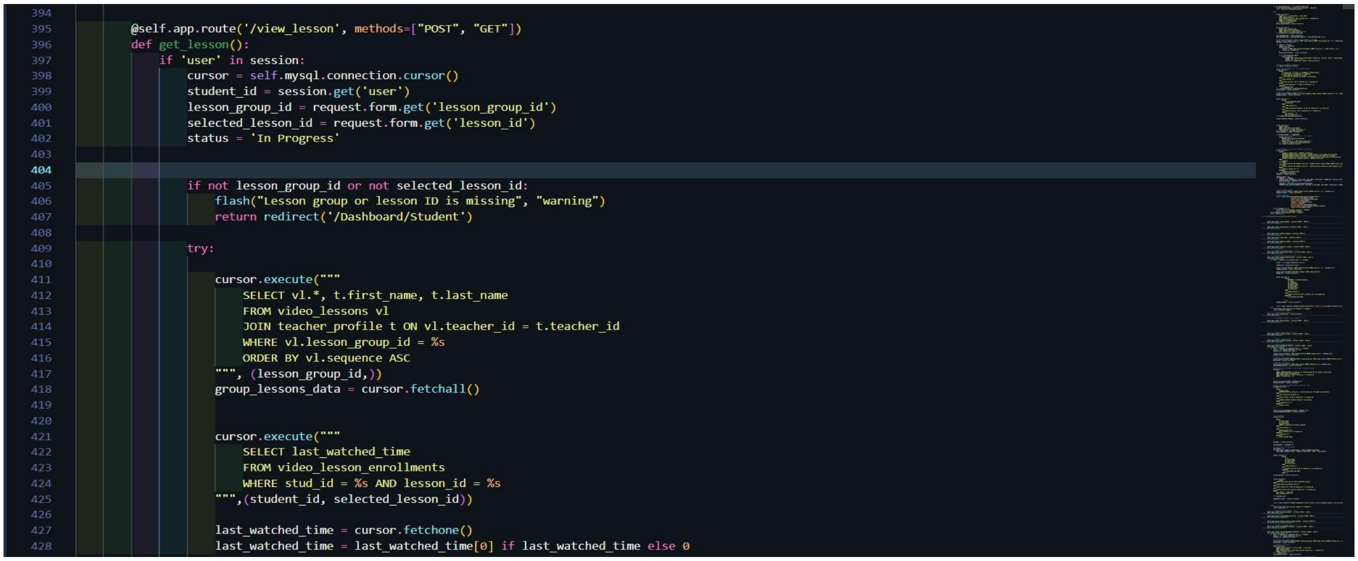
```python

        #---------Student Login Route---------

        @self.app.route("/login", methods=["POST", "GET"])
        def login():
                if request.method == "POST":
                    stud_id = request.form.get('stud_id')
                    email = request.form.get('email')
                    password = request.form.get('password')
                    cursor = self.mysql.connection.cursor()
                    cursor.execute("SELECT * FROM student_profile WHERE stud_id = %s AND email = %s AND password = %s", (stud_id, email,password))
                    studAcc_found = cursor.fetchone()

                    if studAcc_found:
                        session['user'] = studAcc_found[0]
                        session['role'] = "Student"
                        flash("Successfully logged in!", "success")
                        return redirect('/Dashboard/Student')


                    else:
                        flash("Sorry, your data cannot be found!", "danger")
                        return redirect("/")
                else:
                    return redirect("/")
        #---------Student Signup Route---------
```



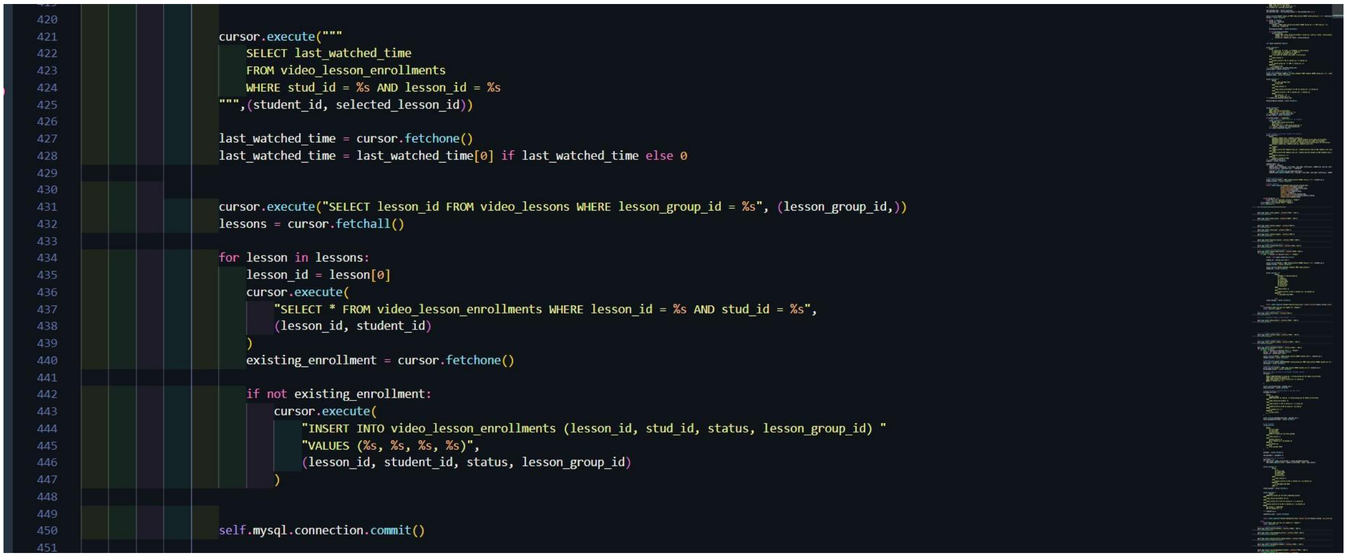
```python

        @self.app.route("/signup", methods=["POST", "GET"])
        def signup():
            if request.method == "POST":
                idi = request.form.get('studentId')
                name = request.form.get('name')
                mname = request.form.get('middlename')
                lname = request.form.get('lastname')
                gmail = request.form.get('email')
                password = request.form.get('password')
                kurso = request.form.get('course')
                gndr = request.form.get('gender')
                yrlvl = request.form.get('yearLevel')
                sction = request.form.get('section')

                # default profile picture
                profile_pic = "/static/images/Defaul_Image.png"



                # Validate names contain only letters
                name_pattern = re.compile(r'^[A-Za-z\s]+$')
                if not (name_pattern.match(name) and
                        name_pattern.match(mname) and
                        name_pattern.match(lname)):
                    flash("Please enter only letters in the name fields.", "warning")
                    return redirect("/signup")

                # Regular expression for ID pattern: 04-2324-xxxxxx
                id_pattern = re.compile(r'^\d{2}-\d{4}-\d{6}$')

                if not id_pattern.match(idi):
                    flash("The ID should follow the format 04-2324-xxxxxx.","warning")
                    return redirect("/signup")

                email_pattern = re.compile(r'^[A-Za-z0-9._%+-]+\.ui@phinmaed\.com$')
```



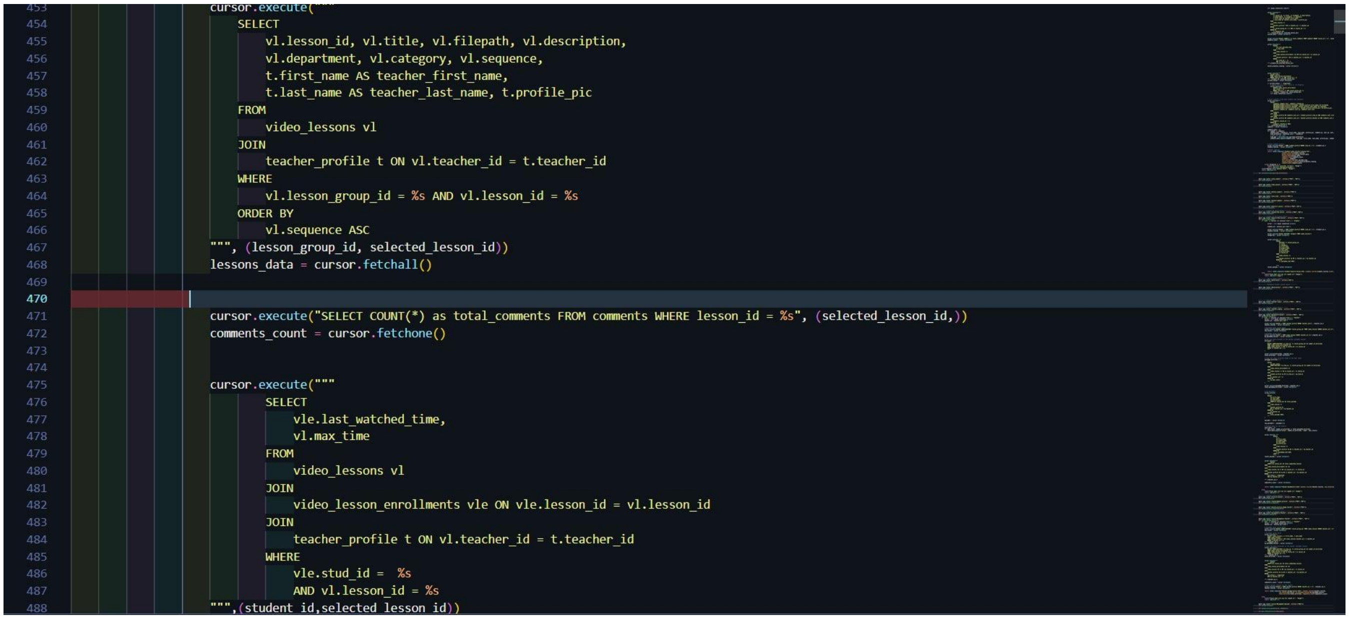
```
                # Check if the student is already exists in the database
                cursor.execute("SELECT * FROM student_profile WHERE stud_id = %s" , (idi,))
                existing_student = cursor.fetchone()
                if existing_student:
                    flash("This ID already exist.", "warning")
                    return redirect("/signup")

                cursor.execute("SELECT * FROM student_profile WHERE email= %s" , (gmail,))
                existing_email = cursor.fetchone()
                if existing_email:
                    flash("This email is already been used !.", "warning")
                    return redirect("/signup")



                #inserting new student data to database
                cursor.execute("INSERT INTO student_profile(stud_id, firstname, middlename, lastname, gender, email, password,course, year_level, section, profile_pic) V
                            (idi,name,mname,lname,gndr,gmail,password,kurso,yrlvl,sction,profile_pic))
                self.mysql.connection.commit()
                cursor.close()
                flash("You're all set! Your registration was successful!","success")
                return redirect("login")
            else:
                return redirect("/")

        #---------Student Dashboard Route----------
```



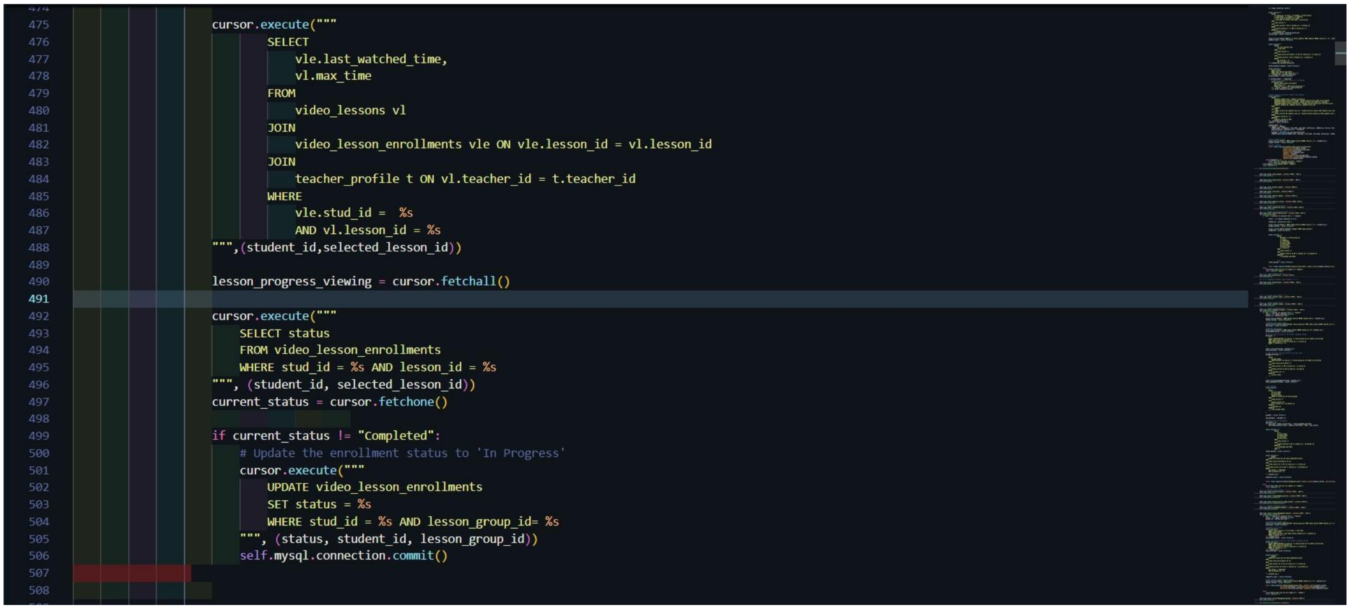
```

        @self.app.route('/view_lesson', methods=["POST", "GET"])
        def get_lesson():
            if 'user' in session:
                cursor = self.mysql.connection.cursor()
                student_id = session.get('user')
                lesson_group_id = request.form.get('lesson_group_id')
                selected_lesson_id = request.form.get('lesson_id')
                status = 'In Progress'


                if not lesson_group_id or not selected_lesson_id:
                    flash("Lesson group or lesson ID is missing", "warning")
                    return redirect('/Dashboard/Student')

                try:

                    cursor.execute("""
                        SELECT vl.*, t.first_name, t.last_name
                        FROM video_lessons vl
                        JOIN teacher_profile t ON vl.teacher_id = t.teacher_id
                        WHERE vl.lesson_group_id = %s
                        ORDER BY vl.sequence ASC
                    """, (lesson_group_id,))
                    group_lessons_data = cursor.fetchall()


                    cursor.execute("""
                        SELECT last_watched_time
                        FROM video_lesson_enrollments
                        WHERE stud_id = %s AND lesson_id = %s
                    """,(student_id, selected_lesson_id))

                    last_watched_time = cursor.fetchone()
                    last_watched_time = last_watched_time[0] if last_watched_time else 0
```



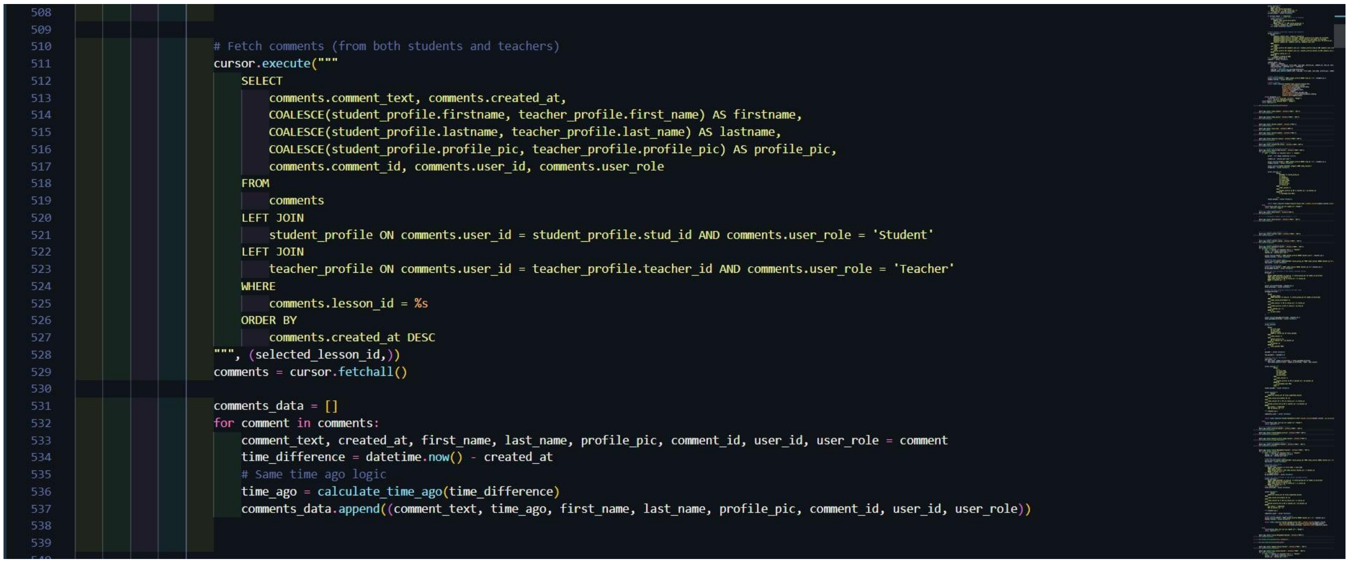
```

                    cursor.execute("""
                        SELECT last_watched_time
                        FROM video_lesson_enrollments
                        WHERE stud_id = %s AND lesson_id = %s
                    """,(student_id, selected_lesson_id))

                    last_watched_time = cursor.fetchone()
                    last_watched_time = last_watched_time[0] if last_watched_time else 0


                    cursor.execute("SELECT lesson_id FROM video_lessons WHERE lesson_group_id = %s", (lesson_group_id,))
                    lessons = cursor.fetchall()

                    for lesson in lessons:
                        lesson_id = lesson[0]
                        cursor.execute(
                            "SELECT * FROM video_lesson_enrollments WHERE lesson_id = %s AND stud_id = %s",
                            (lesson_id, student_id)
                        )
                        existing_enrollment = cursor.fetchone()

                        if not existing_enrollment:
                            cursor.execute(
                                "INSERT INTO video_lesson_enrollments (lesson_id, stud_id, status, lesson_group_id) "
                                "VALUES (%s, %s, %s, %s)",
                                (lesson_id, student_id, status, lesson_group_id)
                            )


                    self.mysql.connection.commit()

```



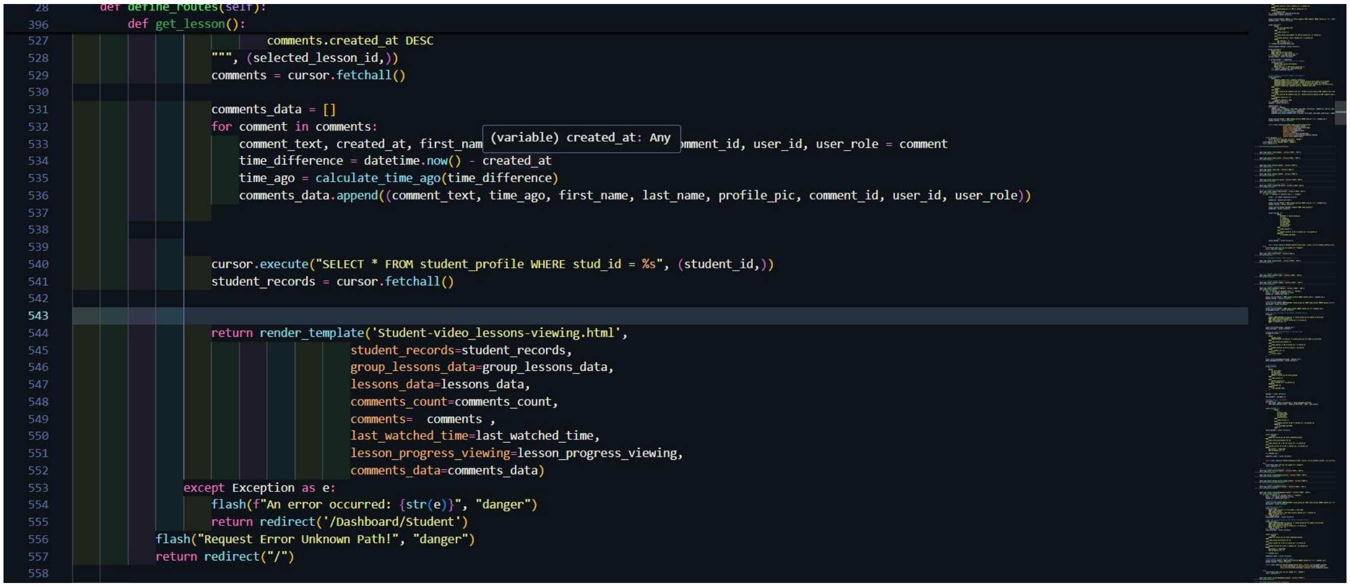
```
cursor.execute("""
    SELECT
        vl.lesson_id, vl.title, vl.filepath, vl.description,
        vl.department, vl.category, vl.sequence,
        t.first_name AS teacher_first_name,
        t.last_name AS teacher_last_name, t.profile_pic
    FROM
        video_lessons vl
    JOIN
        teacher_profile t ON vl.teacher_id = t.teacher_id
    WHERE
        vl.lesson_group_id = %s AND vl.lesson_id = %s
    ORDER BY
        vl.sequence ASC
""", (lesson_group_id, selected_lesson_id))
lessons_data = cursor.fetchall()


cursor.execute("SELECT COUNT(*) as total_comments FROM comments WHERE lesson_id = %s", (selected_lesson_id,))
comments_count = cursor.fetchone()


cursor.execute("""
        SELECT
            vle.last_watched_time,
            vl.max_time
        FROM
            video_lessons vl
        JOIN
            video_lesson_enrollments vle ON vle.lesson_id = vl.lesson_id
        JOIN
            teacher_profile t ON vl.teacher_id = t.teacher_id
        WHERE
            vle.stud_id =  %s
            AND vl.lesson_id = %s
""",(student_id,selected_lesson_id))
```



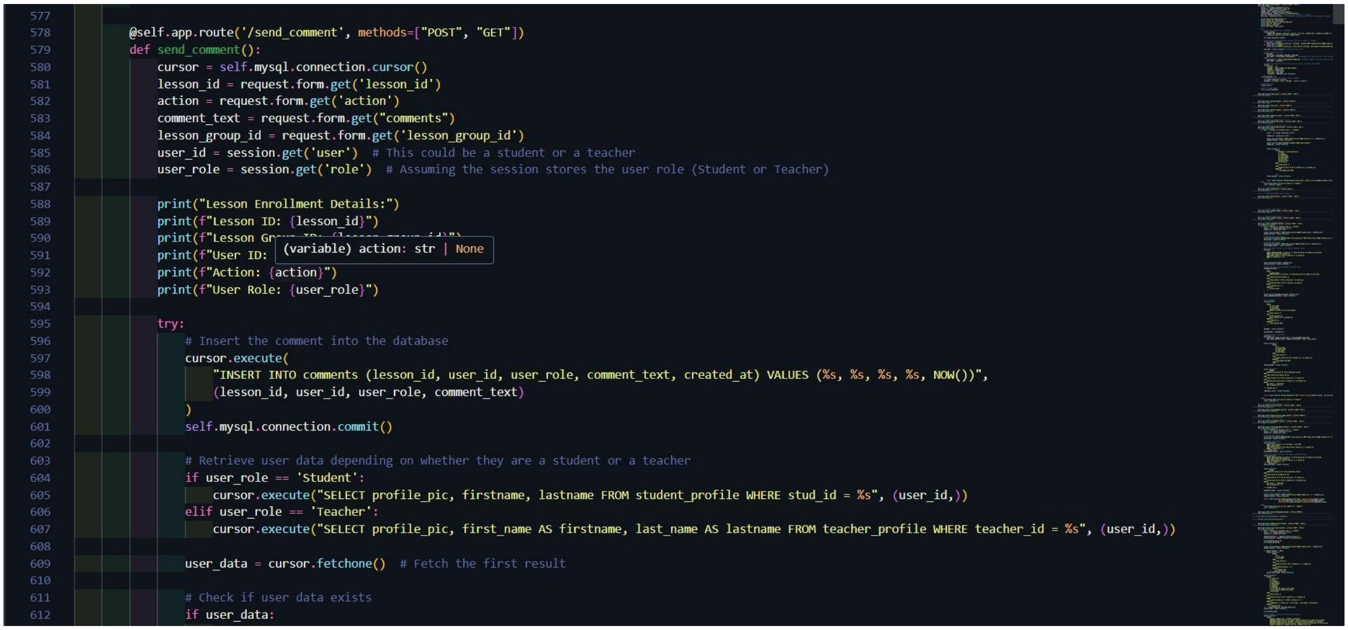
```
cursor.execute("""
        SELECT
            vle.last_watched_time,
            vl.max_time
        FROM
            video_lessons vl
        JOIN
            video_lesson_enrollments vle ON vle.lesson_id = vl.lesson_id
        JOIN
            teacher_profile t ON vl.teacher_id = t.teacher_id
        WHERE
            vle.stud_id =  %s
            AND vl.lesson_id = %s
""",(student_id,selected_lesson_id))

lesson_progress_viewing = cursor.fetchall()

cursor.execute("""
    SELECT status
    FROM video_lesson_enrollments
    WHERE stud_id = %s AND lesson_id = %s
""", (student_id, selected_lesson_id))
current_status = cursor.fetchone()

if current_status != "Completed":
    # Update the enrollment status to 'In Progress'
    cursor.execute("""
        UPDATE video_lesson_enrollments
        SET status = %s
        WHERE stud_id = %s AND lesson_group_id= %s
    """, (status, student_id, lesson_group_id))
    self.mysql.connection.commit()
```



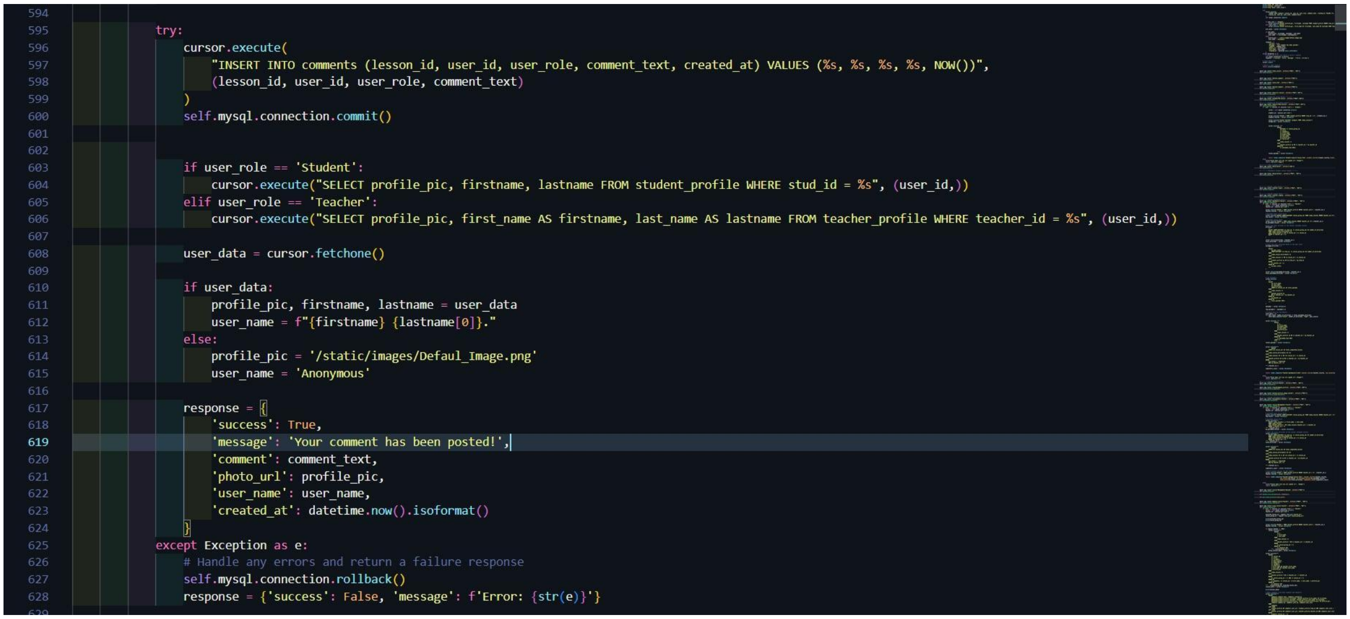
```
# Fetch comments (from both students and teachers)
cursor.execute("""
    SELECT
        comments.comment_text, comments.created_at,
        COALESCE(student_profile.firstname, teacher_profile.first_name) AS firstname,
        COALESCE(student_profile.lastname, teacher_profile.last_name) AS lastname,
        COALESCE(student_profile.profile_pic, teacher_profile.profile_pic) AS profile_pic,
        comments.comment_id, comments.user_id, comments.user_role
    FROM
        comments
    LEFT JOIN
        student_profile ON comments.user_id = student_profile.stud_id AND comments.user_role = 'Student'
    LEFT JOIN
        teacher_profile ON comments.user_id = teacher_profile.teacher_id AND comments.user_role = 'Teacher'
    WHERE
        comments.lesson_id = %s
    ORDER BY
        comments.created_at DESC
""", (selected_lesson_id,))
comments = cursor.fetchall()

comments_data = []
for comment in comments:
    comment_text, created_at, first_name, last_name, profile_pic, comment_id, user_id, user_role = comment
    time_difference = datetime.now() - created_at
    # Same time ago logic
    time_ago = calculate_time_ago(time_difference)
    comments_data.append((comment_text, time_ago, first_name, last_name, profile_pic, comment_id, user_id, user_role))
```



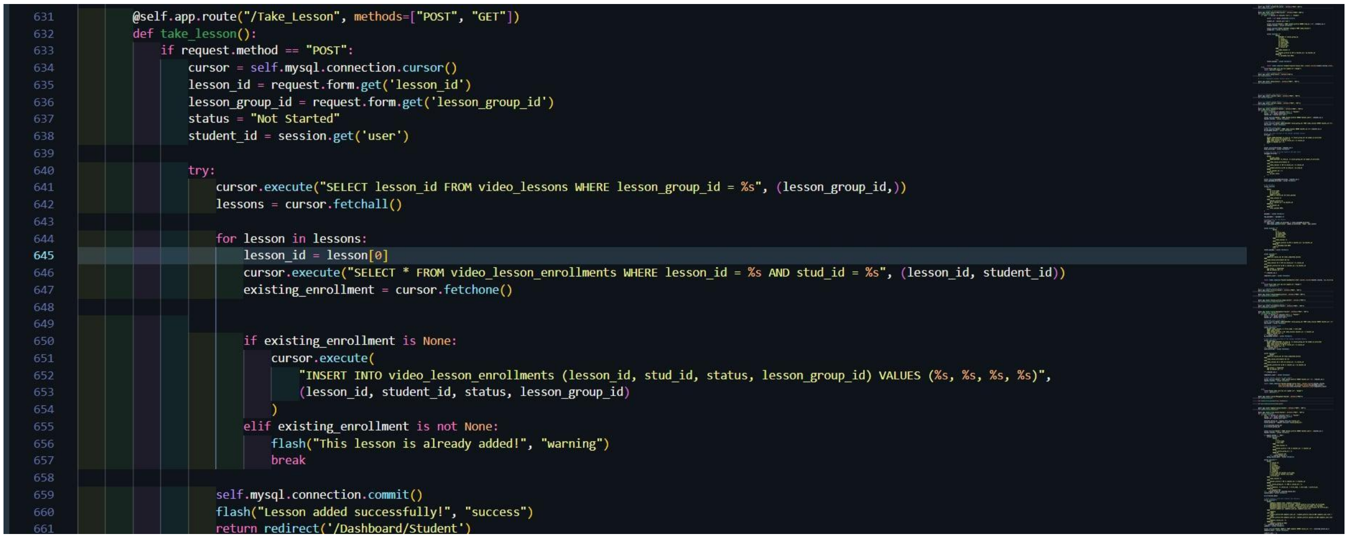
```
    def define_routes(self):
        def get_lesson():
                            comments.created_at DESC
                    """, (selected_lesson_id,))
                    comments = cursor.fetchall()

                    comments_data = []
                    for comment in comments:
                        comment_text, created_at, first_nam                            mment_id, user_id, user_role = comment
                        time_difference = datetime.now() - created_at
                        time_ago = calculate_time_ago(time_difference)
                        comments_data.append((comment_text, time_ago, first_name, last_name, profile_pic, comment_id, user_id, user_role))



                    cursor.execute("SELECT * FROM student_profile WHERE stud_id = %s", (student_id,))
                    student_records = cursor.fetchall()


                    return render_template('Student-video_lessons-viewing.html',
                                        student_records=student_records,
                                        group_lessons_data=group_lessons_data,
                                        lessons_data=lessons_data,
                                        comments_count=comments_count,
                                        comments=  comments ,
                                        last_watched_time=last_watched_time,
                                        lesson_progress_viewing=lesson_progress_viewing,
                                        comments_data=comments_data)
                except Exception as e:
                    flash(f"An error occurred: {str(e)}", "danger")
                    return redirect('/Dashboard/Student')
            flash("Request Error Unknown Path!", "danger")
            return redirect("/")

```



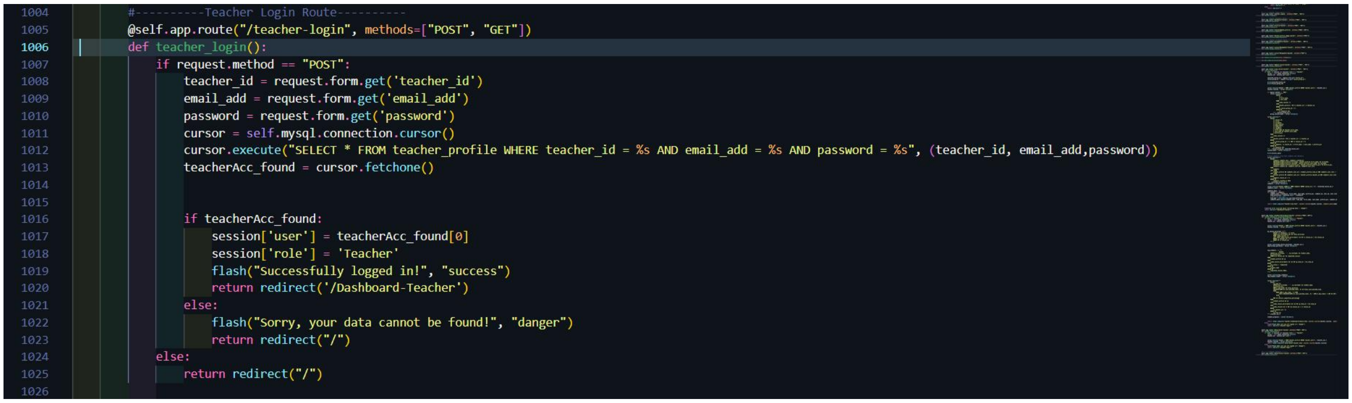
```

        @self.app.route('/send_comment', methods=["POST", "GET"])
        def send_comment():
            cursor = self.mysql.connection.cursor()
            lesson_id = request.form.get('lesson_id')
            action = request.form.get('action')
            comment_text = request.form.get("comments")
            lesson_group_id = request.form.get('lesson_group_id')
            user_id = session.get('user')  # This could be a student or a teacher
            user_role = session.get('role')  # Assuming the session stores the user role (Student or Teacher)

            print("Lesson Enrollment Details:")
            print(f"Lesson ID: {lesson_id}")
            print(f"Lesson Gr
            print(f"User ID:
            print(f"Action: {action}")
            print(f"User Role: {user_role}")

            try:
                # Insert the comment into the database
                cursor.execute(
                    "INSERT INTO comments (lesson_id, user_id, user_role, comment_text, created_at) VALUES (%s, %s, %s, %s, NOW())",
                    (lesson_id, user_id, user_role, comment_text)
                )
                self.mysql.connection.commit()

                # Retrieve user data depending on whether they are a student or a teacher
                if user_role == 'Student':
                    cursor.execute("SELECT profile_pic, firstname, lastname FROM student_profile WHERE stud_id = %s", (user_id,))
                elif user_role == 'Teacher':
                    cursor.execute("SELECT profile_pic, first_name AS firstname, last_name AS lastname FROM teacher_profile WHERE teacher_id = %s", (user_id,))

                user_data = cursor.fetchone()  # Fetch the first result

                # Check if user data exists
                if user_data:
```



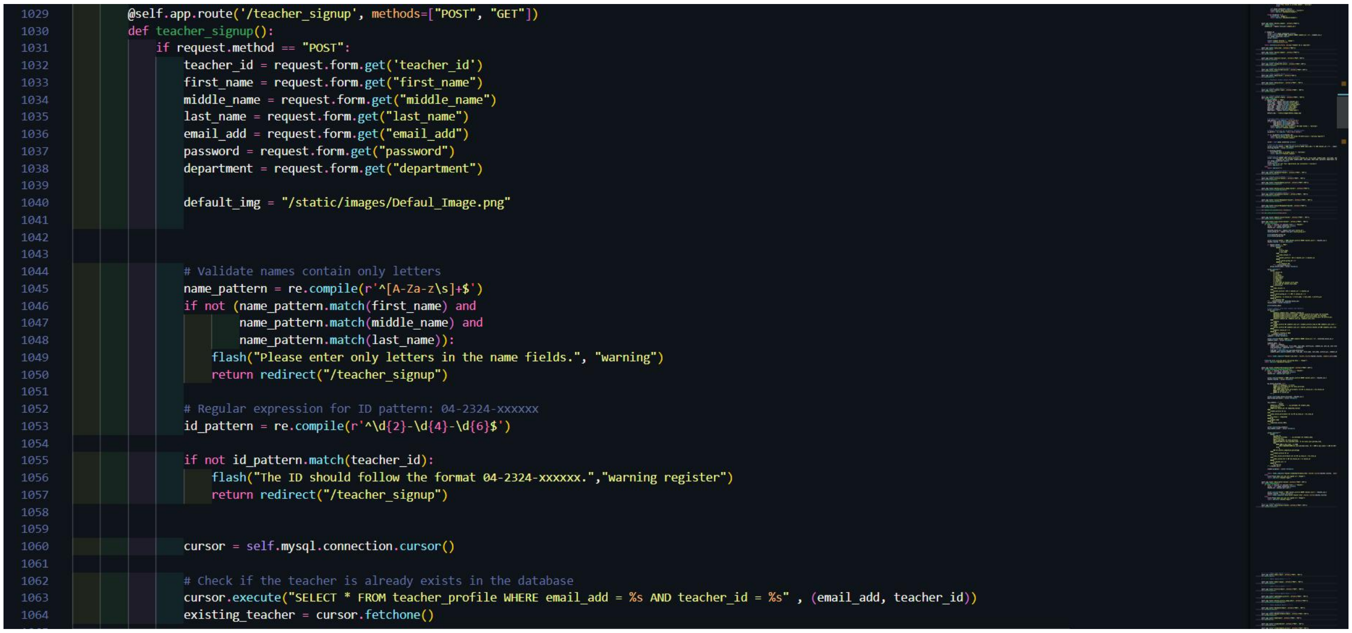
```

            try:
                cursor.execute(
                    "INSERT INTO comments (lesson_id, user_id, user_role, comment_text, created_at) VALUES (%s, %s, %s, %s, NOW())",
                    (lesson_id, user_id, user_role, comment_text)
                )
                self.mysql.connection.commit()


                if user_role == 'Student':
                    cursor.execute("SELECT profile_pic, firstname, lastname FROM student_profile WHERE stud_id = %s", (user_id,))
                elif user_role == 'Teacher':
                    cursor.execute("SELECT profile_pic, first_name AS firstname, last_name AS lastname FROM teacher_profile WHERE teacher_id = %s", (user_id,))

                user_data = cursor.fetchone()

                if user_data:
                    profile_pic, firstname, lastname = user_data
                    user_name = f"{firstname} {lastname[0]}."
                else:
                    profile_pic = '/static/images/Defaul_Image.png'
                    user_name = 'Anonymous'

                response = {
                    'success': True,
                    'message': 'Your comment has been posted!',
                    'comment': comment_text,
                    'photo_url': profile_pic,
                    'user_name': user_name,
                    'created_at': datetime.now().isoformat()
                }
            except Exception as e:
                # Handle any errors and return a failure response
                self.mysql.connection.rollback()
                response = {'success': False, 'message': f'Error: {str(e)}'}
```



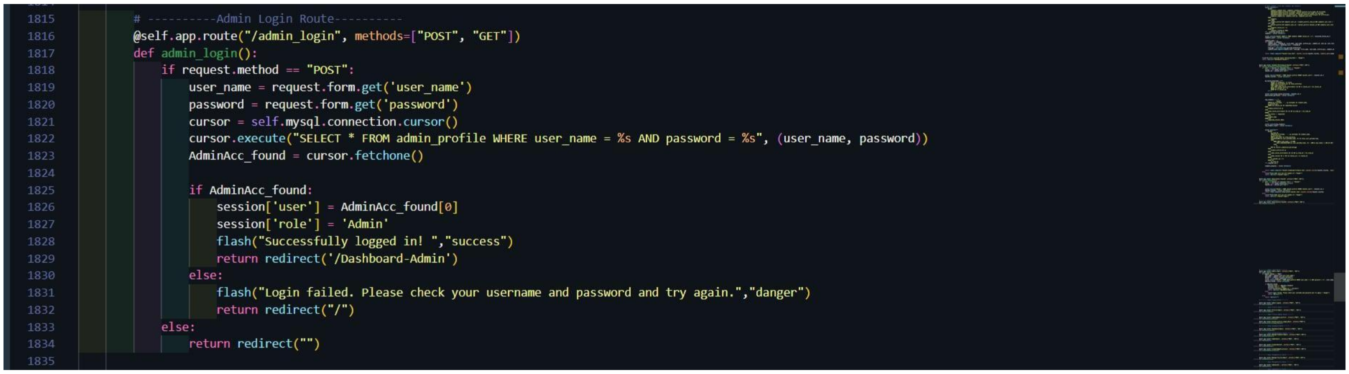
```python
    @self.app.route("/Take_Lesson", methods=["POST", "GET"])
    def take_lesson():
        if request.method == "POST":
            cursor = self.mysql.connection.cursor()
            lesson_id = request.form.get('lesson_id')
            lesson_group_id = request.form.get('lesson_group_id')
            status = "Not Started"
            student_id = session.get('user')

            try:
                cursor.execute("SELECT lesson_id FROM video_lessons WHERE lesson_group_id = %s", (lesson_group_id,))
                lessons = cursor.fetchall()

                for lesson in lessons:
                    lesson_id = lesson[0]
                    cursor.execute("SELECT * FROM video_lesson_enrollments WHERE lesson_id = %s AND stud_id = %s", (lesson_id, student_id))
                    existing_enrollment = cursor.fetchone()


                    if existing_enrollment is None:
                        cursor.execute(
                            "INSERT INTO video_lesson_enrollments (lesson_id, stud_id, status, lesson_group_id) VALUES (%s, %s, %s, %s)",
                            (lesson_id, student_id, status, lesson_group_id)
                        )
                    elif existing_enrollment is not None:
                        flash("This lesson is already added!", "warning")
                        break

                self.mysql.connection.commit()
                flash("Lesson added successfully!", "success")
                return redirect('/Dashboard/Student')
```



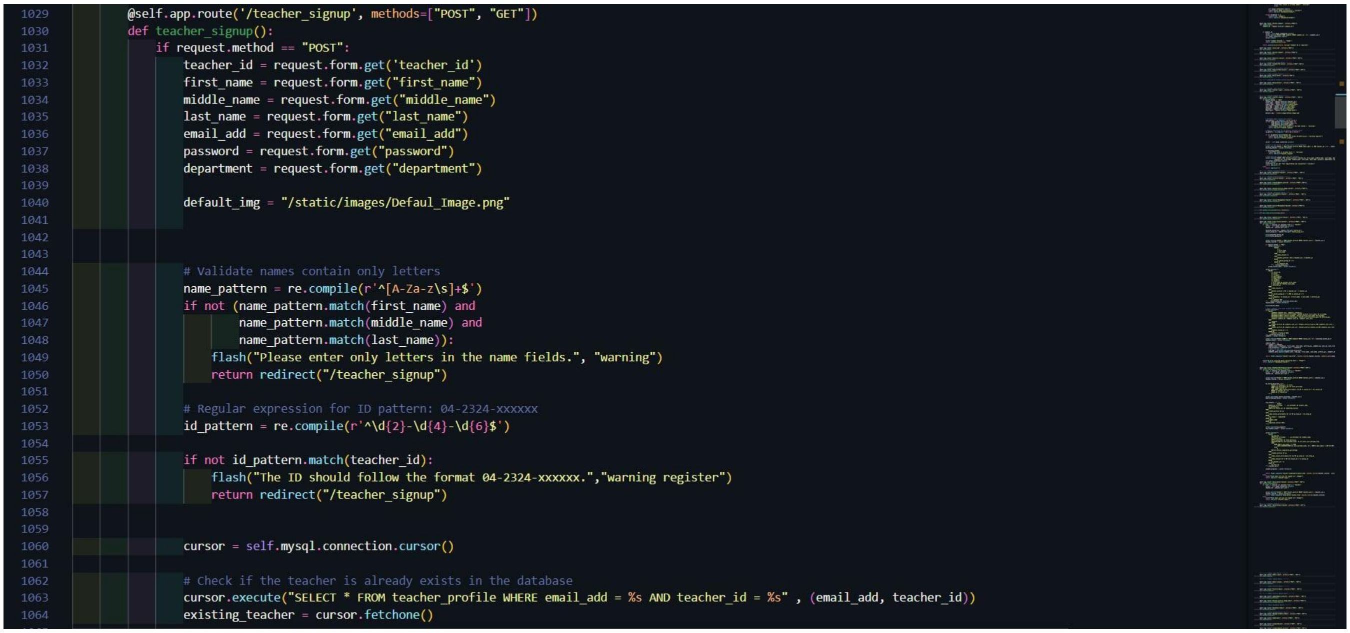
```python
    #----------Teacher Login Route----------
    @self.app.route("/teacher-login", methods=["POST", "GET"])
    def teacher_login():
        if request.method == "POST":
            teacher_id = request.form.get('teacher_id')
            email_add = request.form.get('email_add')
            password = request.form.get('password')
            cursor = self.mysql.connection.cursor()
            cursor.execute("SELECT * FROM teacher_profile WHERE teacher_id = %s AND email_add = %s AND password = %s", (teacher_id, email_add,password))
            teacherAcc_found = cursor.fetchone()


            if teacherAcc_found:
                session['user'] = teacherAcc_found[0]
                session['role'] = 'Teacher'
                flash("Successfully logged in!", "success")
                return redirect('/Dashboard-Teacher')
            else:
                flash("Sorry, your data cannot be found!", "danger")
                return redirect("/")
        else:
            return redirect("/")

```



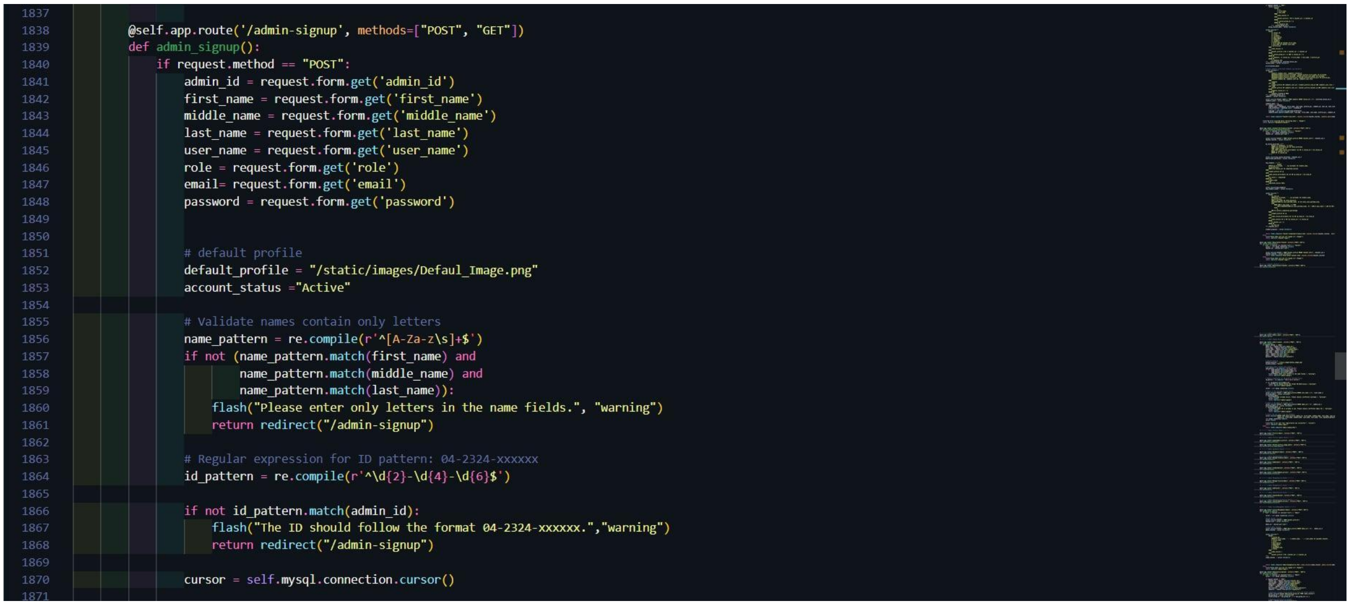
```python
    @self.app.route('/teacher_signup', methods=["POST", "GET"])
    def teacher_signup():
        if request.method == "POST":
            teacher_id = request.form.get('teacher_id')
            first_name = request.form.get("first_name")
            middle_name = request.form.get("middle_name")
            last_name = request.form.get("last_name")
            email_add = request.form.get("email_add")
            password = request.form.get("password")
            department = request.form.get("department")

            default_img = "/static/images/Defaul_Image.png"



            # Validate names contain only letters
            name_pattern = re.compile(r'^[A-Za-z\s]+$')
            if not (name_pattern.match(first_name) and
                    name_pattern.match(middle_name) and
                    name_pattern.match(last_name)):
                flash("Please enter only letters in the name fields.", "warning")
                return redirect("/teacher_signup")

            # Regular expression for ID pattern: 04-2324-xxxxxx
            id_pattern = re.compile(r'^\d{2}-\d{4}-\d{6}$')

            if not id_pattern.match(teacher_id):
                flash("The ID should follow the format 04-2324-xxxxxx.","warning register")
                return redirect("/teacher_signup")


            cursor = self.mysql.connection.cursor()

            # Check if the teacher is already exists in the database
            cursor.execute("SELECT * FROM teacher_profile WHERE email_add = %s AND teacher_id = %s" , (email_add, teacher_id))
            existing_teacher = cursor.fetchone()
```



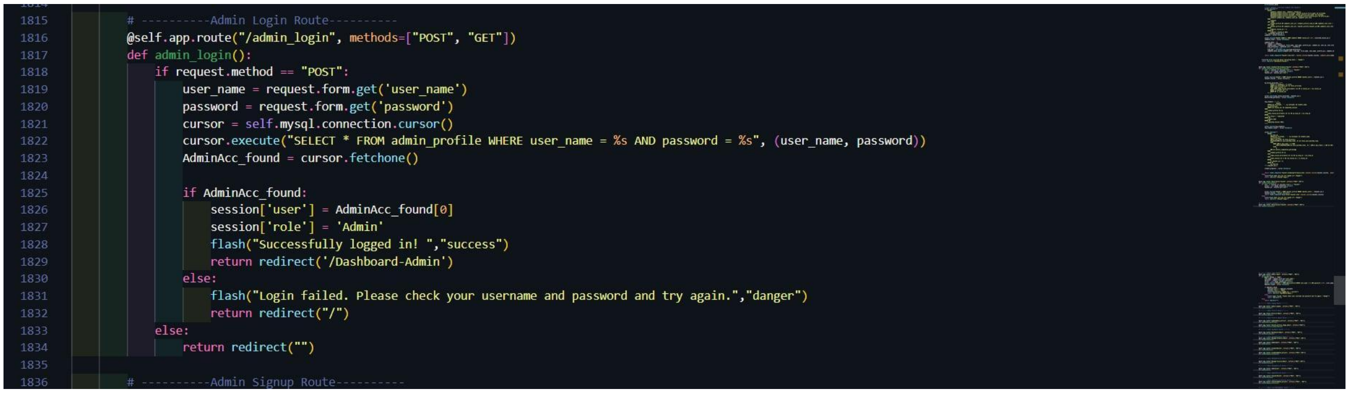
```python
        # ----------Admin Login Route----------
        @self.app.route("/admin_login", methods=["POST", "GET"])
        def admin_login():
            if request.method == "POST":
                user_name = request.form.get('user_name')
                password = request.form.get('password')
                cursor = self.mysql.connection.cursor()
                cursor.execute("SELECT * FROM admin_profile WHERE user_name = %s AND password = %s", (user_name, password))
                AdminAcc_found = cursor.fetchone()

                if AdminAcc_found:
                    session['user'] = AdminAcc_found[0]
                    session['role'] = 'Admin'
                    flash("Successfully logged in! ","success")
                    return redirect('/Dashboard-Admin')
                else:
                    flash("Login failed. Please check your username and password and try again.","danger")
                    return redirect("/")
            else:
                return redirect("")

```



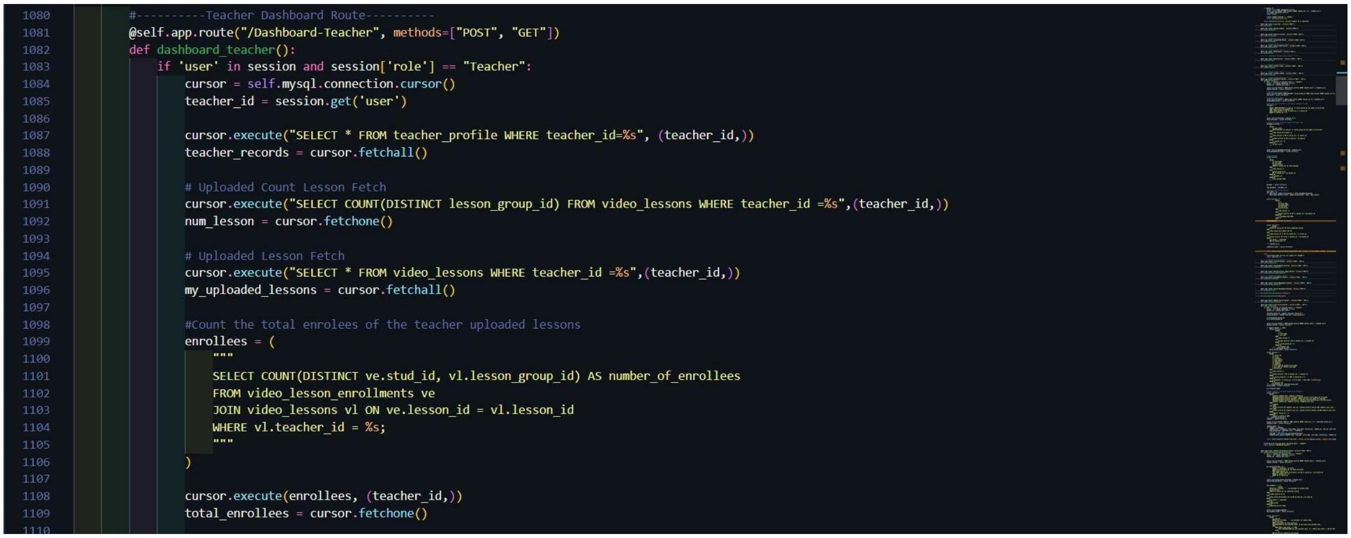
```python
        @self.app.route('/teacher_signup', methods=["POST", "GET"])
        def teacher_signup():
            if request.method == "POST":
                teacher_id = request.form.get('teacher_id')
                first_name = request.form.get("first_name")
                middle_name = request.form.get("middle_name")
                last_name = request.form.get("last_name")
                email_add = request.form.get("email_add")
                password = request.form.get("password")
                department = request.form.get("department")

                default_img = "/static/images/Defaul_Image.png"



                # Validate names contain only letters
                name_pattern = re.compile(r'^[A-Za-z\s]+$')
                if not (name_pattern.match(first_name) and
                        name_pattern.match(middle_name) and
                        name_pattern.match(last_name)):
                    flash("Please enter only letters in the name fields.", "warning")
                    return redirect("/teacher_signup")

                # Regular expression for ID pattern: 04-2324-xxxxxx
                id_pattern = re.compile(r'^\d{2}-\d{4}-\d{6}$')

                if not id_pattern.match(teacher_id):
                    flash("The ID should follow the format 04-2324-xxxxxx.","warning register")
                    return redirect("/teacher_signup")


                cursor = self.mysql.connection.cursor()

                # Check if the teacher is already exists in the database
                cursor.execute("SELECT * FROM teacher_profile WHERE email_add = %s AND teacher_id = %s" , (email_add, teacher_id))
                existing_teacher = cursor.fetchone()
```



```python

        @self.app.route('/admin-signup', methods=["POST", "GET"])
        def admin_signup():
            if request.method == "POST":
                admin_id = request.form.get('admin_id')
                first_name = request.form.get('first_name')
                middle_name = request.form.get('middle_name')
                last_name = request.form.get('last_name')
                user_name = request.form.get('user_name')
                role = request.form.get('role')
                email= request.form.get('email')
                password = request.form.get('password')


                # default profile
                default_profile = "/static/images/Defaul_Image.png"
                account_status ="Active"

                # Validate names contain only letters
                name_pattern = re.compile(r'^[A-Za-z\s]+$')
                if not (name_pattern.match(first_name) and
                        name_pattern.match(middle_name) and
                        name_pattern.match(last_name)):
                    flash("Please enter only letters in the name fields.", "warning")
                    return redirect("/admin-signup")

                # Regular expression for ID pattern: 04-2324-xxxxxx
                id_pattern = re.compile(r'^\d{2}-\d{4}-\d{6}$')

                if not id_pattern.match(admin_id):
                    flash("The ID should follow the format 04-2324-xxxxxx.","warning")
                    return redirect("/admin-signup")

                cursor = self.mysql.connection.cursor()

```



```
        # ----------Admin Login Route----------
        @self.app.route("/admin_login", methods=["POST", "GET"])
        def admin_login():
            if request.method == "POST":
                user_name = request.form.get('user_name')
                password = request.form.get('password')
                cursor = self.mysql.connection.cursor()
                cursor.execute("SELECT * FROM admin_profile WHERE user_name = %s AND password = %s", (user_name, password))
                AdminAcc_found = cursor.fetchone()

                if AdminAcc_found:
                    session['user'] = AdminAcc_found[0]
                    session['role'] = 'Admin'
                    flash("Successfully logged in! ","success")
                    return redirect('/Dashboard-Admin')
                else:
                    flash("Login failed. Please check your username and password and try again.","danger")
                    return redirect("/")
            else:
                return redirect("")

        # ---------Admin Signup Route---------
```



```
        #----------Teacher Dashboard Route----------
        @self.app.route("/Dashboard-Teacher", methods=["POST", "GET"])
        def dashboard_teacher():
            if 'user' in session and session['role'] == "Teacher":
                cursor = self.mysql.connection.cursor()
                teacher_id = session.get('user')

                cursor.execute("SELECT * FROM teacher_profile WHERE teacher_id=%s", (teacher_id,))
                teacher_records = cursor.fetchall()

                # Uploaded Count Lesson Fetch
                cursor.execute("SELECT COUNT(DISTINCT lesson_group_id) FROM video_lessons WHERE teacher_id =%s",(teacher_id,))
                num_lesson = cursor.fetchone()

                # Uploaded Lesson Fetch
                cursor.execute("SELECT * FROM video_lessons WHERE teacher_id =%s",(teacher_id,))
                my_uploaded_lessons = cursor.fetchall()

                #Count the total enrolees of the teacher uploaded lessons
                enrollees = (
                    """
                    SELECT COUNT(DISTINCT ve.stud_id, vl.lesson_group_id) AS number_of_enrollees
                    FROM video_lesson_enrollments ve
                    JOIN video_lessons vl ON ve.lesson_id = vl.lesson_id
                    WHERE vl.teacher_id = %s;
                    """
                )

                cursor.execute(enrollees, (teacher_id,))
                total_enrollees = cursor.fetchone()
```

Data Flow Diagram:

Level 1

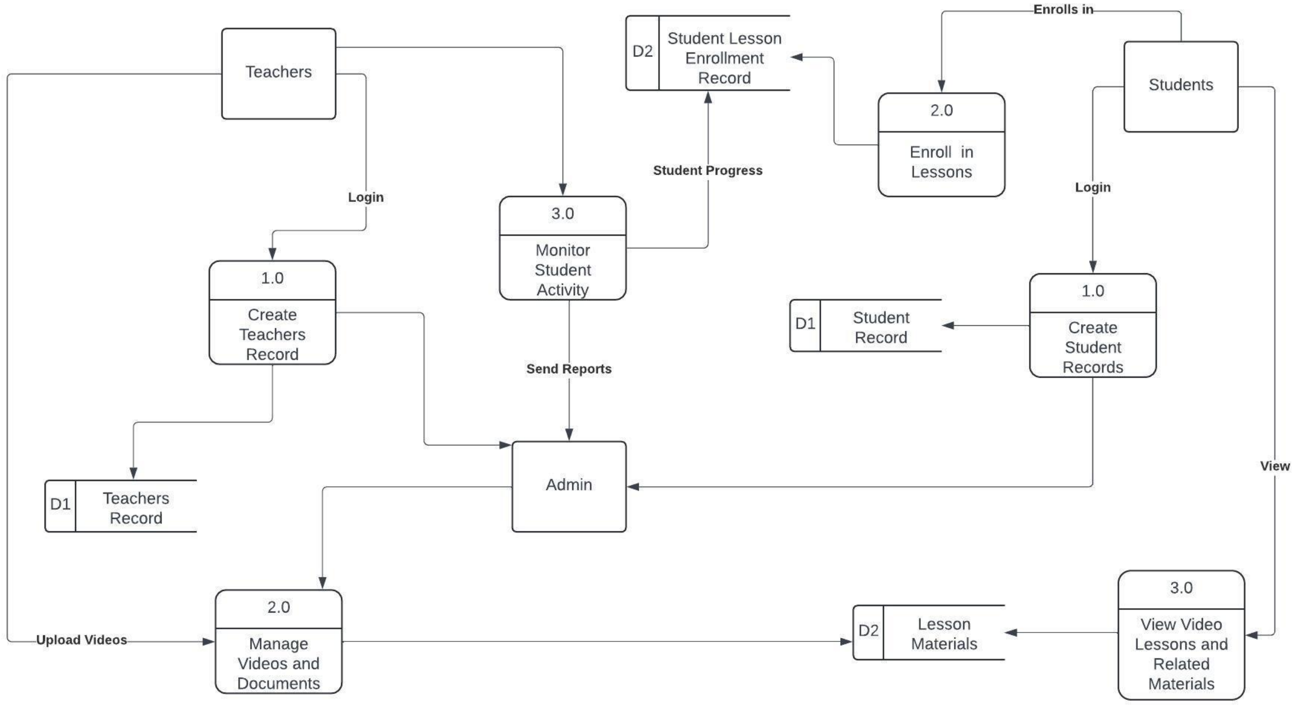

Teachers
D2 Student Lesson Enrollment Record
Enrolls in
Students
2.0 Enroll in Lessons
Student Progress
Login
3.0 Monitor Student Activity
Login
1.0 Create Teachers Record
D1 Student Record
1.0 Create Student Records
Send Reports
Admin
View
D1 Teachers Record
2.0 Manage Videos and Documents
Upload Videos
D2 Lesson Materials
3.0 View Video Lessons and Related Materials

Image 1: Level 1 Data Flow Diagram

Class Diagram UML:

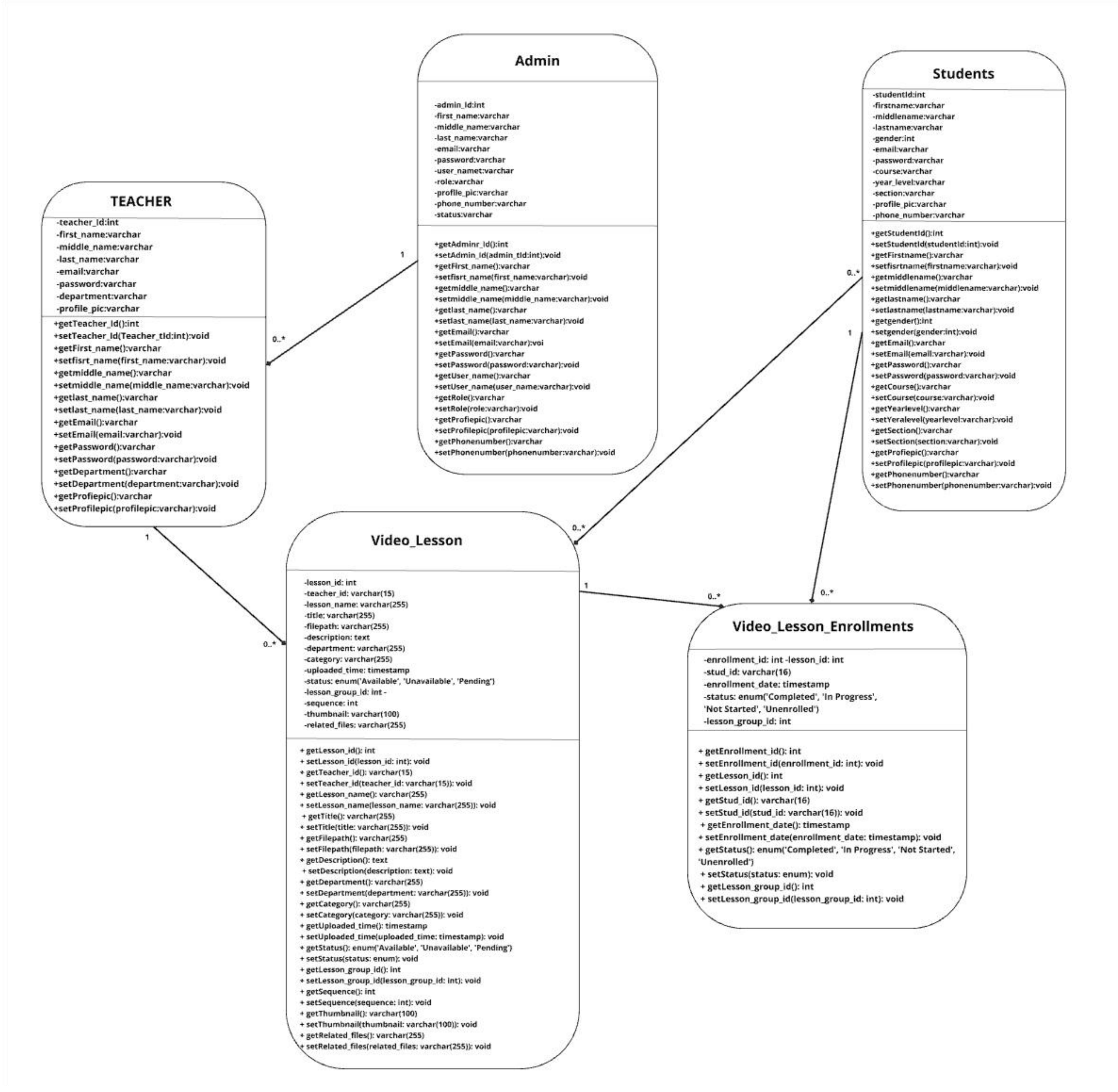

Admin
-admin_Id:int
-first_name:varchar
-middle_name:varchar
-last_name:varchar
-email:varchar
-password:varchar
-user_namet:varchar
-role:varchar
-profile_pic:varchar
-phone_number:varchar
-status:varchar
+getAdminr_Id():int
+setAdmin_Id(admin_tId:int):void
+getFirst_name():varchar
+setfisrt_name(first_name:varchar):void
+getmiddle_name():varchar
+setmiddle_name(middle_name:varchar):void
+getlast_name():varchar
+setlast_name(last_name:varchar):void
+getEmail():varchar
+setEmail(email:varchar):voi
+getPassword():varchar
+setPassword(password:varchar):void
+getUser_name():varchar
+setUser_name(user_name:varchar):void
+getRole():varchar
+setRole(role:varchar):void
+getProfiepic():varchar
+setProfilepic(profilepic:varchar):void
+getPhonenumber():varchar
+setPhonenumber(phonenumber:varchar):void
Students
-studentId:int
-firstname:varchar
-middlename:varchar
-lastname:varchar
-gender:int
-email:varchar
-password:varchar
-course:varchar
-year_level:varchar
-section:varchar
-profile_pic:varchar
-phone_number:varchar
+getStudentId():int
+setStudentId(studentId:int):void
+getFirstname():varchar
+setfisrtname(firstname:varchar):void
+getmiddlename():varchar
+setmiddlename(middlename:varchar):void
+getlastname():varchar
+setlastname(lastname:varchar):void
+getgender():int
+setgender(gender:int):void
+getEmail():varchar
+setEmail(email:varchar):void
+getPassword():varchar
+setPassword(password:varchar):void
+getCourse():varchar
+setCourse(course:varchar):void
+getYearlevel():varchar
+setYeralevel(yearlevel:varchar):void
+getSection():varchar
+setSection(section:varchar):void
+getProfiepic():varchar
+setProfilepic(profilepic:varchar):void
+getPhonenumber():varchar
+setPhonenumber(phonenumber:varchar):void
TEACHER
-teacher_Id:int
-first_name:varchar
-middle_name:varchar
-last_name:varchar
-email:varchar
-password:varchar
-department:varchar
-profile_pic:varchar
+getTeacher_Id():int
+setTeacher_Id(Teacher_tId:int):void
+getFirst_name():varchar
+setfisrt_name(first_name:varchar):void
+getmiddle_name():varchar
+setmiddle_name(middle_name:varchar):void
+getlast_name():varchar
+setlast_name(last_name:varchar):void
+getEmail():varchar
+setEmail(email:varchar):void
+getPassword():varchar
+setPassword(password:varchar):void
+getDepartment():varchar
+setDepartment(department:varchar):void
+getProfiepic():varchar
+setProfilepic(profilepic:varchar):void
1
0..*
0..*
1
1
0..*
Video_Lesson
-lesson_id: int
-teacher_id: varchar(15)
-lesson_name: varchar(255)
-title: varchar(255)
-filepath: varchar(255)
-description: text
-department: varchar(255)
-category: varchar(255)
-uploaded_time: timestamp
-status: enum('Available', 'Unavailable', 'Pending')
-lesson_group_id: int -
-sequence: int
-thumbnail: varchar(100)
-related_files: varchar(255)
+ getLesson_id(): int
+ setLesson_id(lesson_id: int): void
+ getTeacher_id(): varchar(15)
+ setTeacher_id(teacher_id: varchar(15)): void
+ getLesson_name(): varchar(255)
+ setLesson_name(lesson_name: varchar(255)): void
+ getTitle(): varchar(255)
+ setTitle(title: varchar(255)): void
+ getFilepath(): varchar(255)
+ setFilepath(filepath: varchar(255)): void
+ getDescription(): text
+ setDescription(description: text): void
+ getDepartment(): varchar(255)
+ setDepartment(department: varchar(255)): void
+ getCategory(): varchar(255)
+ setCategory(category: varchar(255)): void
+ getUploaded_time(): timestamp
+ setUploaded_time(uploaded_time: timestamp): void
+ getStatus(): enum('Available', 'Unavailable', 'Pending')
+ setStatus(status: enum): void
+ getLesson_group_id(): int
+ setLesson_group_id(lesson_group_id: int): void
+ getSequence(): int
+ setSequence(sequence: int): void
+ getThumbnail(): varchar(100)
+ setThumbnail(thumbnail: varchar(100)): void
+ getRelated_files(): varchar(255)
+ setRelated_files(related_files: varchar(255)): void
0..*
1
0..*
0..*
0..*
Video_Lesson_Enrollments
-enrollment_id: int -lesson_id: int
-stud_id: varchar(16)
-enrollment_date: timestamp
-status: enum('Completed', 'In Progress',
'Not Started', 'Unenrolled')
-lesson_group_id: int
+ getEnrollment_id(): int
+ setEnrollment_id(enrollment_id: int): void
+ getLesson_id(): int
+ setLesson_id(lesson_id: int): void
+ getStud_id(): varchar(16)
+ setStud_id(stud_id: varchar(16)): void
+ getEnrollment_date(): timestamp
+ setEnrollment_date(enrollment_date: timestamp): void
+ getStatus(): enum('Completed', 'In Progress', 'Not Started',
'Unenrolled')
+ setStatus(status: enum): void
+ getLesson_group_id(): int
+ setLesson_group_id(lesson_group_id: int): void

Image 2: UML Class Diagram