Mini-Project Report On

AUTOMATED SUBJECT ALLOCATION SYSTEM

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

By

George Jose (U2003084)
Joseph Tomy (U2003110)
Midhun Mohan K M (U2003134)
Naman Mathew George (U2003141)

Under the guidance of
Ms. Jisha Mary Jose

Department of Computer Science & Engineering
Rajagiri School of Engineering and Technology (Autonomous)
Rajagiri Valley, Kakkanad, Kochi, 682039

July 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)
RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039

CERTIFICATE

This is to certify that the mini-project report entitled ***AUTOMATED SUBJECT ALLOCATION SYSTEM*** *is a bonafide work done by* ***Mr. George Jose (U2003084), Mr. Joseph Tomy (U2003110), Mr. Midhun Mohan K M (U2003134), Mr. Naman Mathew George (U2003141)****, submitted to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2022-2023.*

Dr. Preetha K. G.	**Mr. Uday Babu P.**	**Ms. Jisha Mary Jose**
Head of Department	Mini-Project Coordinator	Mini-Project Guide
Dept. of CSE	Asst. Professor	Asst. Professor
RSET	Dept. of CSE	Dept. of CSE
	RSET	RSET

ACKNOWLEDGEMENTS

We wish to express our sincere gratitude towards **Dr. P. S. Sreejith**, Principal of RSET, and **Dr. Preetha K. G.**, Head of Department of Computer Science and Engineering for providing us with the opportunity to undertake our mini-project, AUTOMATED SUBJECT ALLOCATION SYSTEM.

We are highly indebted to our mini-project coordinators, **Ms.Tripti C**, Assistant Professor, Department of Computer Science and Engineering and **Mr. Uday Babu**, Assistant Professor, Department of Computer Science and Engineering for their valuable support.

It is indeed our pleasure and a moment of satisfaction for us to express our sincere gratitude to our mini-project guide **Ms. Jisha Mary Jose**, for her patience and all the priceless advice and wisdom she has shared with us.

Last but not the least, we would like to express our sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

George Jose

Joseph Tomy

Midhun Mohan K M

Naman Mathew George

ABSTRACT

Our project encompasses the development of a comprehensive subject allocation system designed to streamline the process for teachers and administrators. Built using React JS and Django, the system offers distinct user interfaces for administrators and teachers, each with separate login credentials.

The administrator interface empowers administrators to manage teacher, subject, and department details efficiently. Through this interface, administrators can insert, delete, and update information related to teachers, subjects, and departments. Specifically, in the subject allocation section, administrators have the authority to initiate or halt the subject selection process, as well as monitor its progress.

In the teacher interface, educators can participate actively in the subject selection process by indicating their preferences. They have the freedom to prioritize subjects according to their preferences, and the system displays their selected subjects accordingly. Once teachers have finalized their selections, they can review their opted subjects in order of priority.

Upon the conclusion of the teacher's selection submission, the backend triggers the allocation process, considering factors such as seniority and designation. Consequently, one or two of the most preferred subjects are assigned to each teacher.

Following the allocation process, teachers gain access to comprehensive information on all teachers and their allotted subjects for every academic year. This holistic view enables teachers to plan and manage their academic responsibilities efficiently.

In conclusion, our subject allocation system brings enhanced efficiency and transparency to the allocation process, benefiting both administrators and teachers alike. It optimizes the distribution of subjects based on individual preferences and criteria, facilitating a seamless educational experience.

Contents

Acknowledgements ii

Abstract iii

List of Figures vii

1 Introduction 1

1.1 Background 1

1.2 Existing System 2

1.3 Problem Statement 2

1.4 Objectives 3

1.5 Scope 3

2 Literature Review 5

3 System Analysis 7

3.1 Expected System Requirements 7

3.2 Feasibility Analysis 7

3.2.1 Technical Feasibility 7

3.2.2 Operational Feasibility 7

3.2.3 Economic Feasibility 7

3.3 Hardware Requirements 8

3.4 Software Requirements 8

3.4.1 Visual Studio for web development 8

3.4.2 React JS 8

3.4.3 Django 9

3.4.4 Python 3.10 9

4 Methodology **10**

4.1 Proposed Method 10

4.1.1 Requirement Gathering 10

4.1.2 Design and Planning 10

4.1.3 Database Design 11

4.1.4 System Development 11

4.1.5 Teacher input 11

4.1.6 Algorithm Implementation 11

4.1.7 Conflict Resolution 11

5 System Design **13**

5.1 Architecture Diagram 13

5.2 Use Case 14

5.3 Flow diagram 15

6 System Implementation **16**

6.1 Technology Stack 16

6.2 User interfaces 16

6.3 Administrator Interface 16

6.4 Teacher Interface 16

6.5 Subject Allocation Process 17

6.6 Data Processing in Back end 17

6.7 Viewing Allocated Subjects 17

7 Testing **18**

7.1 Unit Testing 18

7.2 Integration Testing 18

7.3 System Testing 18

7.4 User Interface Testing 19

8 Results **20**

9 Risks and Challenges **34**

10 Conclusion **35**

References **36**

Appendix A: Front End Sample Code **36**

Appendix B: Back End Sample Code **67**

Appendix C: CO-PSO Mapping **83**

List of Figures

5.1 Architecture diagram 13
5.2 use case diagram 14
5.3 use case diagram 15

8.1 Main Menu interface 20
8.2 Admin interface 21
8.3 admin dashboard 21
8.4 admin department view 22
8.5 admin department update 22
8.6 Teacher view 23
8.7 Teacher add 23
8.8 Teacher add 24
8.9 Teacher update 24
8.10 Subject view 25
8.11 Subject add/update 25
8.12 Subject view 26
8.13 26
8.14 starting Phase 1 27
8.15 Teacher Login interface 27
8.16 Interface for subject selection 28
8.17 certain restrictions while selecting subject 28
8.18 opting Subjects 29
8.19 viewing Opted subjects 29
8.20 Displaying subjects opted by other teachers 30
8.21 Starting Phase 2 30
8.22 admin view during subject selection in phase 2 31
8.23 Stopping Phase 2 32

8.24 Final view After allocation 33

Chapter 1

Introduction

1.1 Background

Using an automated subject allocation system for teachers in college offers numerous benefits and advantages. Firstly, the system can be customized to cater to the college's specific needs and policies, accommodating various allocation algorithms, subject assignment rules, and teacher preferences. It also excels in conflict resolution by automatically detecting and resolving issues like scheduling overlaps or teacher unavailability, ensuring subjects are assigned appropriately. Considering teacher preferences and qualifications, the system enhances overall teacher satisfaction by matching them with subjects that align with their expertise and interests. Moreover, the automated process reduces the workload for administrators, sparing them from manual subject management for each teacher. Integrating seamlessly with other college management systems, such as payroll, attendance, and student databases, fosters a cohesive and interconnected administrative ecosystem. Real-time updates allow teachers and administrators to stay informed about subject allocations, changes, and notifications, fostering better communication and transparency. Leveraging data analytics, the system facilitates improved resource planning, enabling administrators to assess subject demand and allocate resources effectively for future academic terms. Streamlined communication between teachers and administrators is facilitated by the automated system, empowering teachers to provide feedback or request changes to their subject assignments more efficiently. Ensuring compliance with college regulations, government policies, and accreditation standards, the system assures fair and consistent subject assignments. The system's long-term benefits are evident as it continuously accumulates data and insights, leading to ongoing improvements in the subject allocation process based on past allocations and evolving college needs. In terms of cost-effectiveness, implementing the automated system may involve an initial invest-

ment, but it results in long-term savings due to increased efficiency, minimized errors, and optimized resource allocation. Security is enhanced through robust measures protecting sensitive teacher and subject data, mitigating risks associated with manual handling and paper record storage. Additionally, the system caters to teachers with special needs, ensuring fair treatment and inclusivity in subject allocation. Lastly, it aids in tracking teacher performance in relation to subject assignments, allowing identification of areas for improvement or professional development. Overall, the automated subject allocation system significantly improves the efficiency, accuracy, and fairness of the subject allocation process in college.

1.2 Existing System

Manual seniority-based subject allocation is a traditional approach used in educational institutions to assign subjects to teachers based on their years of service within the organization. In this process, administrators take into account the seniority of teachers and their expressed preferences for specific subjects. Teachers are given the opportunity to rank their desired subjects, and administrators make allocation decisions considering these preferences along with subject availability and qualifications. Seniority holds a crucial role in determining the priority for subject allocation, giving more experienced teachers the advantage of choosing their preferred subjects. While this method relies on human decision-making and may require iterative adjustments, it aims to acknowledge the expertise and experience of senior teachers while ensuring a balanced distribution of subjects among the teaching staff.

1.3 Problem Statement

The need for an efficient and streamlined online subject allocation system for teachers in educational institutions. Current manual methods lead to inefficiencies, inconsistencies and potential biases.

People with disabilities face a lot of difficulties in coping with a fast changing modern society.This app would help them overcome their disabilities and avoid feeling isolated. In order to make them comfortable in the present world, this app can be an ideal option

for the senior citizens who suffer from cognitive disabilities.

The application addresses the social divide among common people and people with disabilities ,by helping the latter using technology.

1.4 Objectives

- **T**o develop a web application for subject allocation management.
- **A**utomate and streamline the subject allocation process.
- **S**ave time and manual labour for administrators and teachers.
- **E**liminate manual paperwork and tedious duties.
- **M**ake the subject allocation procedure more effective.

1.5 Scope

The scope of online seniority-based subject allocation for teachers is broad and can offer numerous advantages to educational institutions. Here are some key aspects that highlight the scope and benefits of implementing such a system:

1. Efficiency and Time Savings: An online subject allocation system streamlines the entire process, reducing the administrative burden and saving time for both teachers and administrators. Automation enables quick data processing, preference gathering, and allocation decisions, leading to a more efficient and timely subject assignment.

2. Transparency and Fairness: By moving the subject allocation process online, educational institutions can enhance transparency and fairness. Teachers can easily view the criteria and factors considered in the allocation process, promoting trust and confidence in the system. This transparency minimizes the likelihood of perceived biases and ensures equitable treatment for all teachers.

3. Customization and Flexibility: Online systems can be tailored to accommodate various allocation scenarios, such as different subject preferences, subject availability, and changing institutional needs. Administrators can configure the system to meet specific requirements, ensuring a flexible approach that adapts to the evolving dynamics of the institution.

4. Optimization of Subject Allocation: Online systems have the capability to optimize subject assignments based on multiple parameters, including seniority, qualifications, expertise, and teacher preferences. By considering these factors comprehensively, the system can make well-informed decisions, leading to improved subject-teacher alignment.

5. Data-Driven Decision Making: The online system provides a wealth of data that can be analyzed to gain insights into subject preferences, teacher expertise, and allocation patterns. This data-driven approach facilitates continuous improvement and data-based decision-making for future subject allocations.

6. Integration with Existing Systems: An online subject allocation system can be integrated with other educational management systems, such as teacher databases, timetable generation, and performance evaluation, creating a cohesive ecosystem for educational administration in the near future.

Chapter 2

Literature Review

Subject allocation for teachers is a crucial aspect of educational management, directly impacting the teaching-learning process and overall academic performance. Researchers have explored various methods and decision-making models to optimize subject allocation based on teachers' preferences and qualifications. Two relevant studies shed light on this area.

In the study by T. Matsuo and T. Fujimoto, an effective lecture/class allocation method for elective subjects is proposed. The researchers emphasize the significance of considering users' profiles, which in this context refers to teachers' preferences and qualifications, during the allocation process. By analyzing these profiles, the method aims to identify the most suitable subject assignments for teachers, thereby enhancing teacher satisfaction and overall teaching quality. Though the study focuses on elective subjects, its approach of using profiles for allocation serves as valuable inspiration for developing an efficient subject allocation system in educational institutions.

Similarly, P. Parthiban, K. Ganesh, S. Narayanan, and R. Dhanalakshmi present a Preferences Based Decision-Making Model (PDM) to address the faculty course assignment problem. The researchers propose a model that considers faculty preferences, qualifications, and expertise to optimize the course assignment process. The PDM aims to assign courses to faculty members in alignment with their preferences, thereby enhancing their motivation and engagement in the teaching process. This study highlights the importance of incorporating preferences in decision-making models for subject allocation, which can significantly improve teacher satisfaction and contribute to a more effective teaching-learning environment.

These studies emphasize the critical role of considering teacher preferences and qualifications in the subject allocation process. Leveraging insights from these research works, the project's online subject allocation system can implement a data-driven decision-

making model. By optimizing subject assignments based on teachers' seniority, designation, preferences, and expertise, the system will promote fairness, transparency, and efficiency in subject allocation, ultimately leading to higher teacher satisfaction and a more productive educational institution.

Chapter 3

System Analysis

3.1 Expected System Requirements

The system of user which is a smart phone is expected to have the following features:

- Requirement of stable Internet connection .
- A modern web browser like Google Chrome or Mozilla Firefox.
- A minimum Ram size of 8GB is required in the device.

3.2 Feasibility Analysis

3.2.1 Technical Feasibility

The project is technically feasible since all staff have access to stable internet connection inside of college premises.

3.2.2 Operational Feasibility

The operations are built in a simple and easy to use manner for staff and admin.

3.2.3 Economic Feasibility

Moving away from pen and paper to an automated system can contribute to reducing paper waste and promote a more environmentally friendly approach.

3.3 Hardware Requirements

The following are the system requirements to develop the Unify App.

- Processor: Intel Core i5
- RAM: Minimum 8GB

3.4 Software Requirements

The following are the software used in the development of the app.

- Visual Studio
- React JS
- Django
- Python 3.10

3.4.1 Visual Studio for web development

Visual Studio Code (VS Code) is a popular, lightweight, and open-source code editor developed by Microsoft. With extensive language support, intelligent code completion, syntax highlighting, debugging capabilities, and version control integration, it offers a feature-rich environment for developers. Its flexibility is enhanced through a vast range of extensions, allowing customization based on individual requirements. Whether working on web development, software engineering, data science, or any coding project, VS Code provides an efficient and user-friendly platform for coding tasks. Its seamless integration with various tools makes it a preferred choice for developers across diverse domains

3.4.2 React JS

React JS, also known as React, is a widely-used JavaScript library for creating user interfaces in web applications. Developed by Facebook, it has gained popularity due to its efficiency, flexibility, and component-based architecture. React allows developers to build reusable UI components, simplifying the construction of complex interfaces by combining and nesting these components. With a declarative approach, developers describe how

the UI should appear based on the application state, leaving React to handle DOM updates efficiently using its virtual DOM. It can be easily integrated into projects and is commonly used alongside other libraries like Redux and React Router. The active community and regular updates make React a favored choice for building modern and scalable web applications.

3.4.3 Django

Python Django is a high-level web framework that streamlines and accelerates web application development. It follows the "batteries-included" philosophy, providing pre-built components for common tasks. Django's robustness lies in its adherence to the Model-View-Template (MVT) architectural pattern, enhancing code maintainability. With its ORM system, developers interact with the database using Python objects, avoiding direct SQL queries. The framework's built-in administrative interface simplifies content management and allows customization. Django ensures robust security with various authentication methods and benefits from a vibrant open-source community, receiving regular updates and comprehensive support. It is an excellent choice for developers seeking scalable, secure, and feature-rich web applications.

3.4.4 Python 3.10

Python 3.10, released on October 4, 2021, is the latest major version of the Python programming language. This update introduces significant improvements, including pattern matching with the new ‘match‘ statement for more concise conditional branching. Type hinting has also been enhanced with new types like ‘Protocol‘ and ‘Literal‘. The ‘str‘ type gains additional methods for easier string manipulation. Python 3.10 brings performance optimizations, improved error messages, and updates to the standard library, enhancing the overall development experience. With its continued focus on readability, simplicity, and versatility, Python 3.10 remains a valuable tool for a wide range of applications, making it a recommended upgrade for developers looking to leverage the latest features and improvements.

Chapter 4

Methodology

4.1 Proposed Method

- Develop a web application that allocate subject based on teacher seniority and designation.
- We divide the web app into two parts one for admin and other for faculty
- Admin can add teacher details,subject details ,department details as input and it is stored onto the database.
- Application also contains features like mailing reminder system and a clash resolution mechanism

4.1.1 Requirement Gathering

This is the initial stage where the project team collects information and identifies the specific needs and expectations of the college concerning subject allocation. Explanation: During this phase, the team will communicate with college administrators, teachers, and other stakeholders to understand the key requirements, such as teacher qualifications, subject preferences, constraints, and any other factors that need to be considered for the allocation process.

4.1.2 Design and Planning

In this stage, the project team creates a detailed plan that outlines the structure and approach for developing the automated subject allocation system. The design and planning phase involves determining the algorithms, rules, and criteria that will be used for subject allocation. It also includes outlining the user interface design, database structure, and integration points with other existing college management systems.

4.1.3 Database Design

Database design involves creating a well-organized and efficient database to store all relevant information related to teachers, subjects, and allocation history.During this stage, the project team will design the database schema, defining tables, relationships, and constraints necessary to store teacher qualifications, subject details, preferences, and other essential data required for the allocation process.

4.1.4 System Development

System development refers to the actual coding and programming of the automated subject allocation system based on the design and planning.In this phase, the project team will write the code for the system using a suitable programming language (such as Python) and web development framework (like Django) to build the core functionality of the system

4.1.5 Teacher input

This step involves providing a way for teachers to input their subject preferences, availability, and other relevant information into the automated system.To make the subject allocation process fair and considerate of teachers' preferences, the system should allow teachers to enter their desired subjects, availability for teaching, and any other factors that may affect their allocation

4.1.6 Algorithm Implementation

Algorithm implementation refers to translating the subject allocation rules and criteria into executable code to determine the best allocation for each teacher.This stage involves programming the algorithms that analyze teachers' qualifications, preferences, and constraints to match them with appropriate subjects based on predefined rules.

4.1.7 Conflict Resolution

Conflict resolution involves handling and resolving conflicts that may arise during the subject allocation process, such as scheduling overlaps or teacher unavailability.The system should be equipped with mechanisms to automatically identify and address any conflicts

that occur during the allocation process, ensuring that teachers are assigned to subjects without any conflicting schedules.

Chapter 5

System Design

5.1 Architecture Diagram

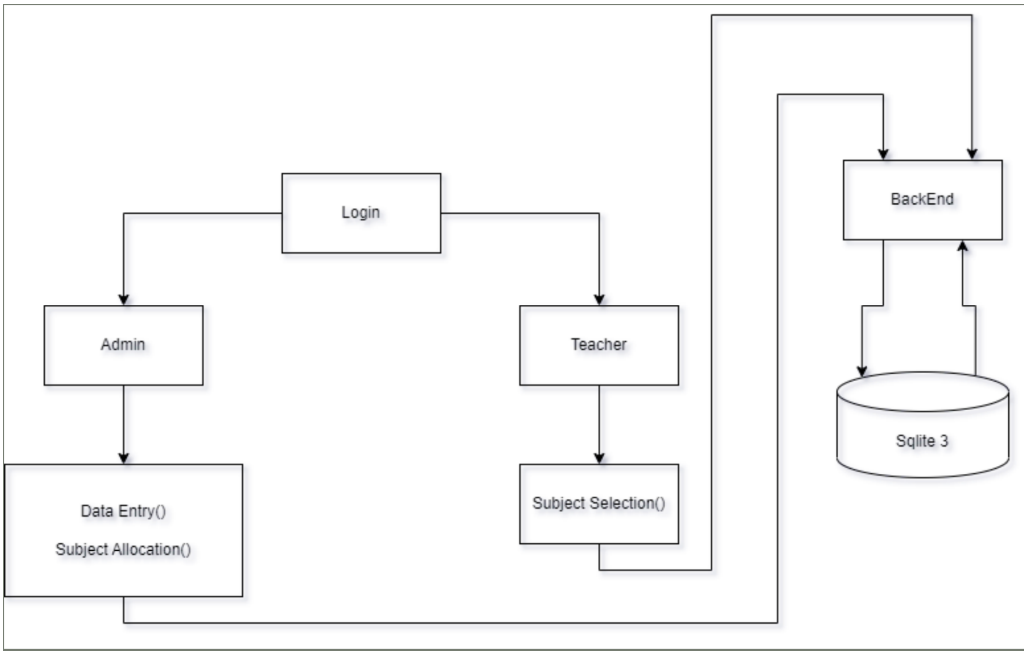


Figure 5.1: Architecture diagram

5.2 Use Case

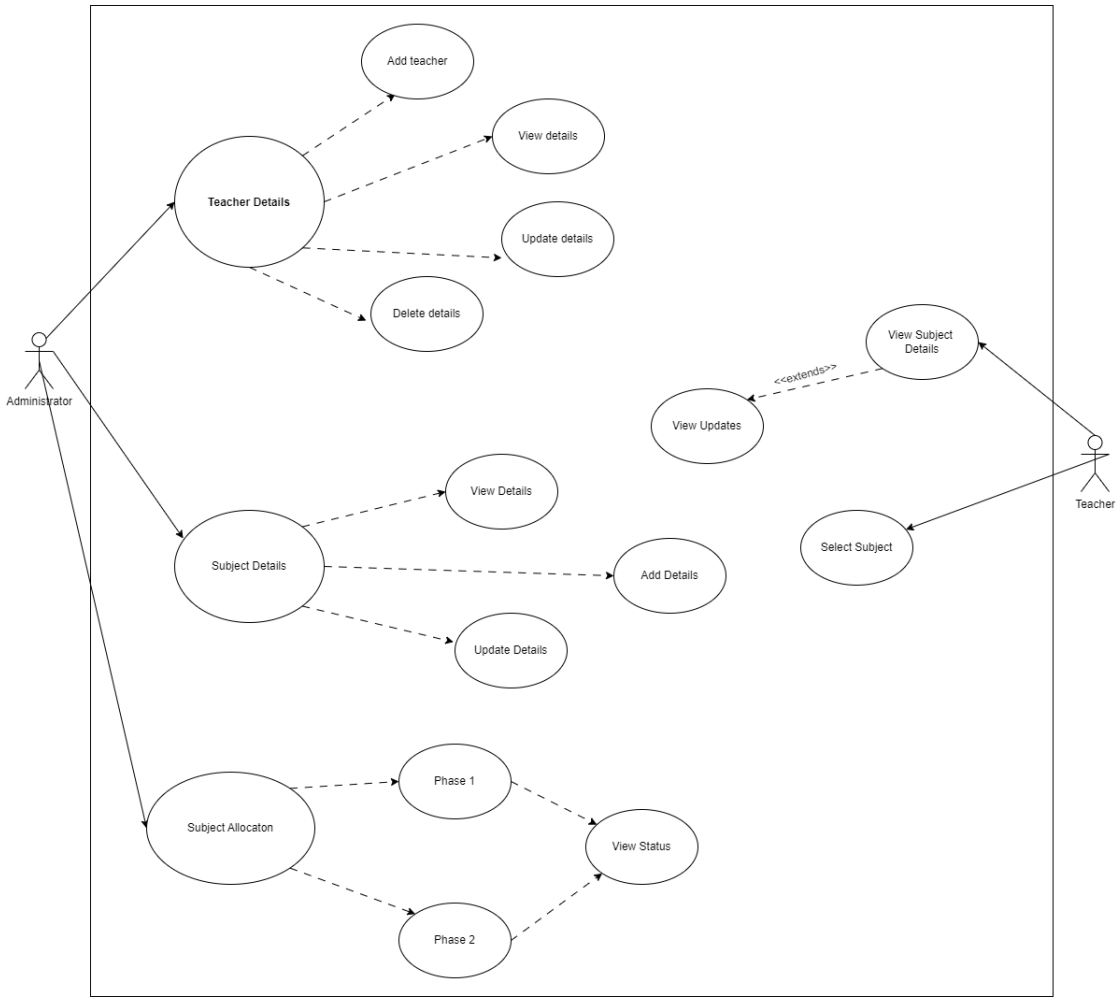


Figure 5.2: use case diagram

5.3 Flow diagram

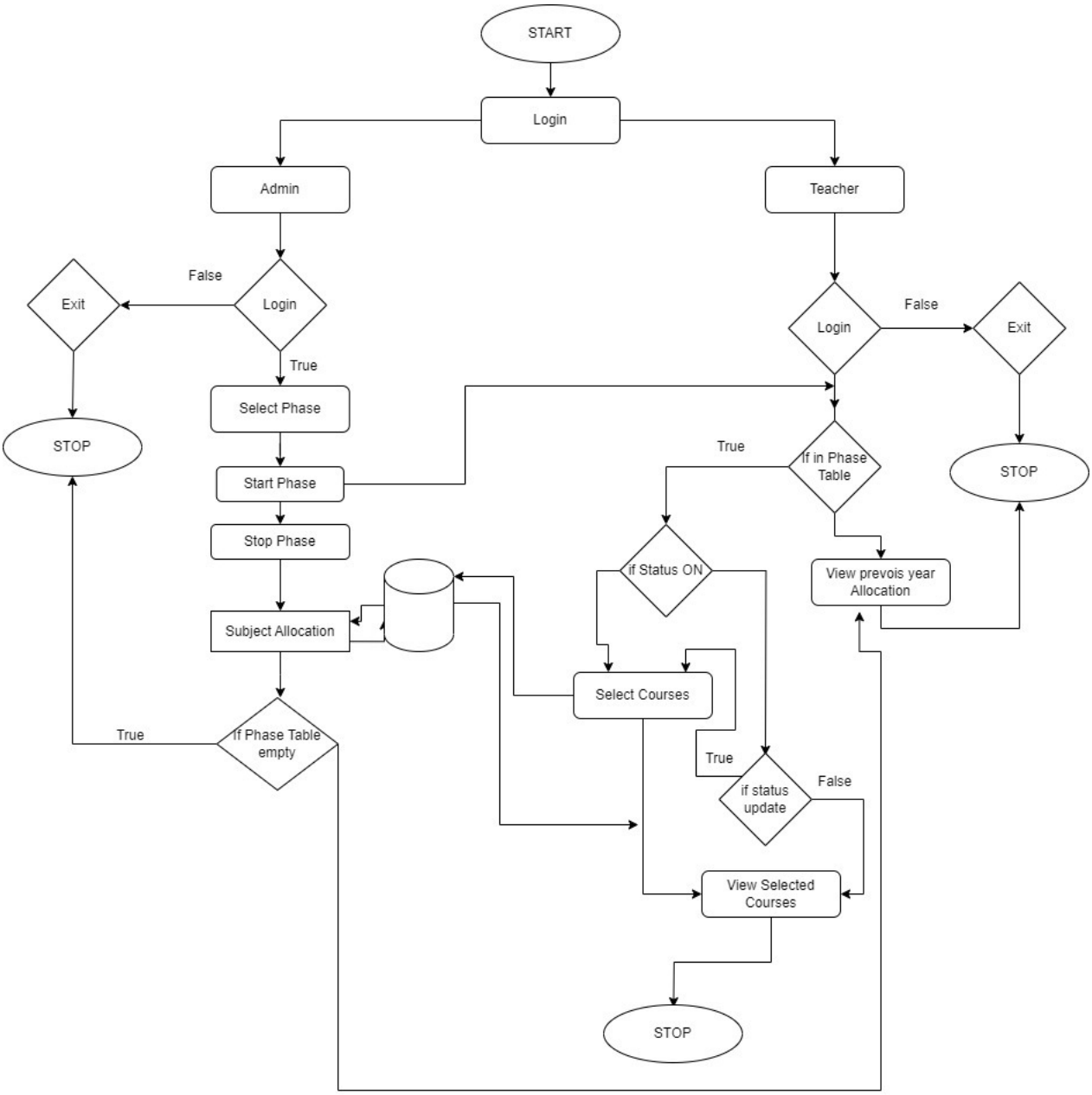


Figure 5.3: use case diagram

Chapter 6

System Implementation

6.1 Technology Stack

The website utilizes React JS for the frontend and Django for the backend. React JS enables the creation of dynamic and interactive user interfaces, while Django provides robust backend functionalities, including database management and API handling.

6.2 User interfaces

The website offers two distinct user interfaces: one for administrators and the other for teachers. Each interface requires separate login credentials to access the corresponding functionalities.

6.3 Administrator Interface

The administrator interface empowers administrators with complete control over teacher, subject, and department details. Administrators can perform CRUD (Create, Read, Update, Delete) operations to manage the data efficiently. Additionally, administrators have the authority to initiate or terminate the subject allocation process and monitor its progress in real-time.

6.4 Teacher Interface

In the teacher interface, teachers can access a personalized dashboard to select subjects based on their preferences and qualifications. The interface provides an intuitive display of selected subjects alongside their priorities. Once teachers submit their preferred subject list, they can review the opted subjects in the order of their choosing.

6.5 Subject Allocation Process

Upon submission of the teacher's preferred subjects, the back end commences the allocation process. The system considers seniority and designation to ensure that each teacher receives one or two of their most preferred subjects. This approach maximizes teacher satisfaction by aligning the allocation with their qualifications and experience.

6.6 Data Processing in Back end

In the back end, the received data is processed and combined with the generated Python script to create a self-contained script. The data is formatted and made ready for the training process.

6.7 Viewing Allocated Subjects

Once the allocation process concludes, teachers can access a comprehensive list displaying all teachers and their allotted subjects for each academic year. This feature provides teachers with clarity and transparency regarding their subject assignments.

Chapter 7

Testing

7.1 Unit Testing

Individual modules or components of the platform were tested in isolation to confirm their correctness and accuracy. Writing test cases for each unit was used to carry out this testing. Unit tests were also built for user login and signup using incorrect usernames, passwords, or both. These tests validated error-handling messages. To verify accurate data retrieval and updating, database interactions were also validated in the admin module, information area, and user login/signup. A priority-based subject allocation approach was carried out using unit tests. We were able to discover and correct any faults or problems inside the units by separating each component.

7.2 Integration Testing

To evaluate how well the platform's various components integrated, integration testing was done. This testing made sure that information moved between modules without any problems and that the functions of different parts worked together to provide the required results. We tested how well the various modules—including login/signup, admin, data entry, teacher, phase selection, and subject selection—worked together. Integration tests were crucial to find any potential problems caused by the interaction of various components.

7.3 System Testing

In order to verify that the system was in conformity with the project's requirements and goals, system testing required evaluating the system as a whole. To ensure that the platform allocates subjects appropriately and effectively, we conducted system testing.

To confirm that the platform met the project's performance objectives, we also assessed its effectiveness and response times in various scenarios.

7.4 User Interface Testing

User interface (UI) testing was done to verify the visual components, design, and usability of the platform. In the project's development, this test was crucial. The project's functionality is greatly influenced by the GUI that is supplied for choosing the algorithm and entering its hyperparameters. We put the user interface through tests to make sure it was responsive, easy to use, and intuitive. During this assessment, the platform's overall visual attractiveness as well as the functionality of the buttons, menus, and textboxes were assessed. We improved the interface through UI testing to enhance the user experience.

Chapter 8

Results

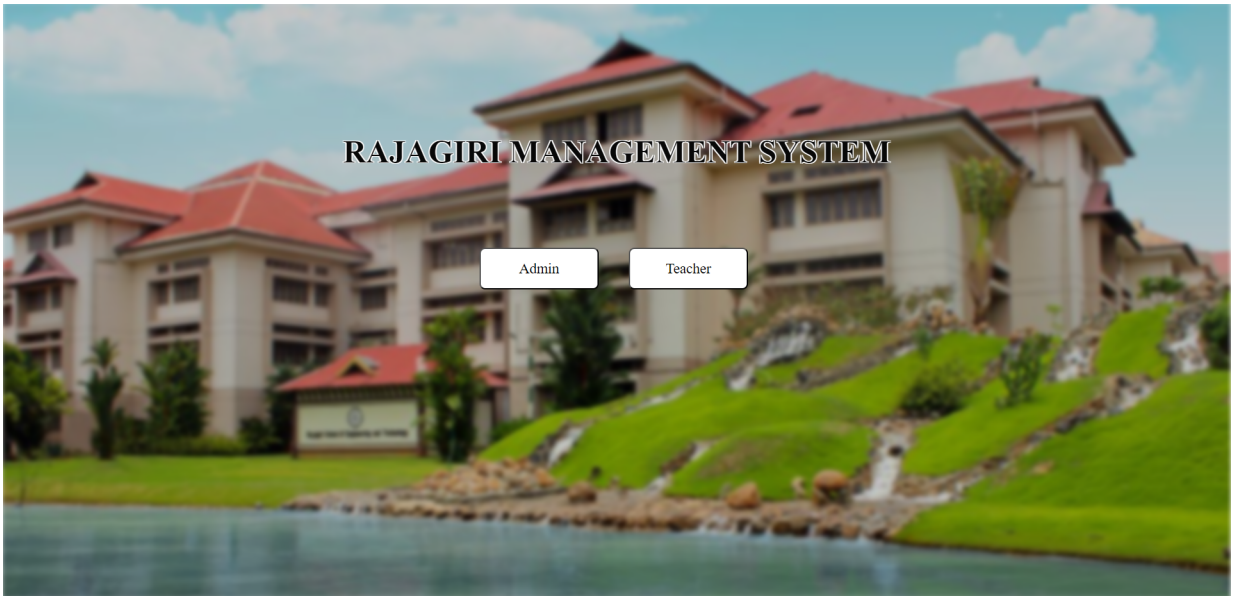


Figure 8.1: Main Menu interface

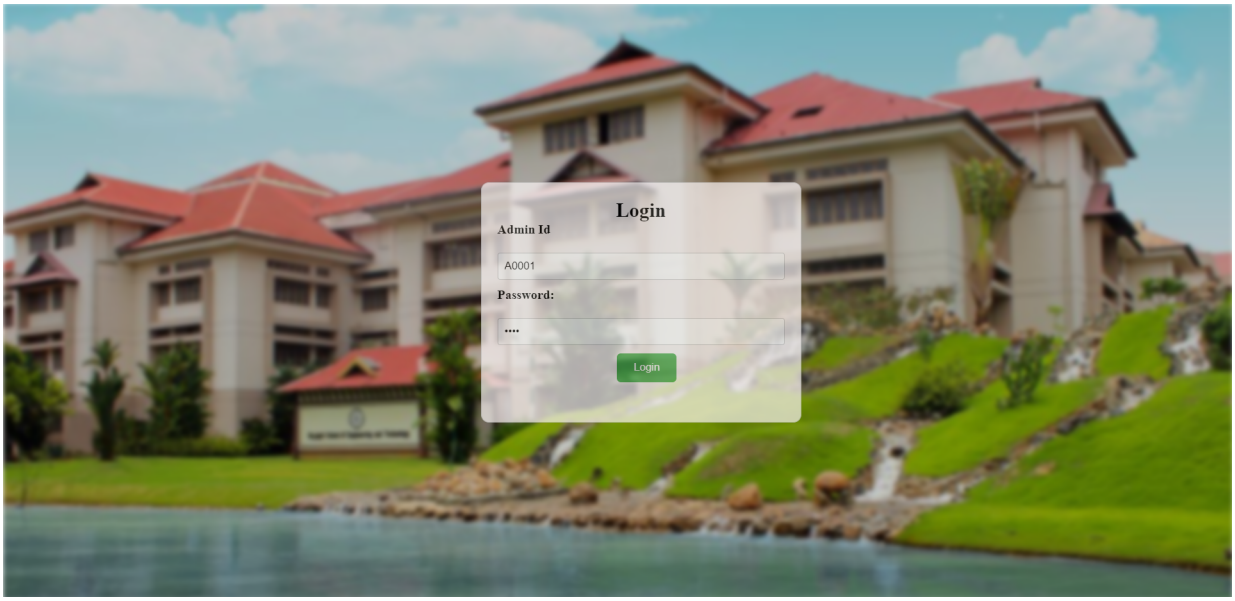


Figure 8.2: Admin interface

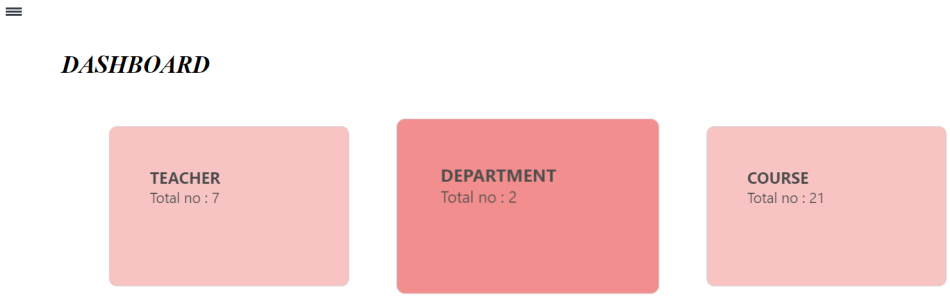


Figure 8.3: admin dashboard

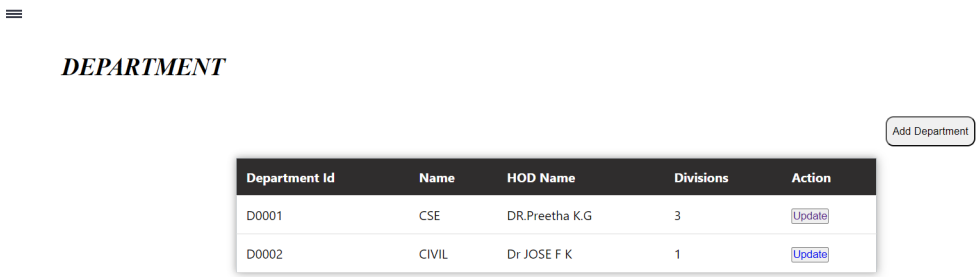


Figure 8.4: admin department view

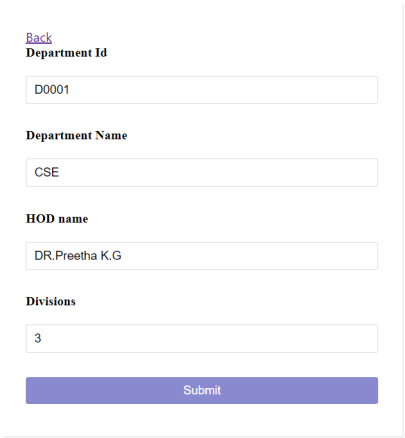


Figure 8.5: admin department update

TEACHER

Add new Teacher

Name	Email	Gender	Id	Designation	Date Of Joining	Dept Id	Action
Uday Babu P	udaybp@rajagiri.edu.in	Male	T0001	Asst proff	2016-06-01	CSE	Update Delete
Dr. Saritha S	saritha_s@rajagiritech.edu.in	FEMALE	T0002	Asso proff	2004-07-01	CSE	Update Delete
Amitha Mathew	amitham@rajagiritech.edu.in	FEMALE	T0004	Asst proff	2012-06-05	CSE	Update Delete
Dr. Jisha G.	jishag@rajagiritech.edu.in	FEMALE	T0006	Asst proff	2010-01-18	CSE	Update Delete
Dr.VarghesePual	varghesep@rajagiritech.edu.in	MALE	T0003	Proff	2016-08-01	CSE	Update Delete
Dr. Uma Narayan	uman@rajagiritech.edu.in	FEMALE	T0007	Asst proff	2021-12-01	CSE	Update Delete
Ms. Tripti C	triptic@rajagiritech.edu.in	FEMALE	T0008	Asst proff	2008-07-01	CSE	Update Delete

Figure 8.6: Teacher view

back

Name

Dr. Uma Narayan

E-mail

uman@rajagiritech.edu.in

Gender

Male

Id

T0007

Password

1234

Date Of Joining

dd-mm-yyyy

Figure 8.7: Teacher add

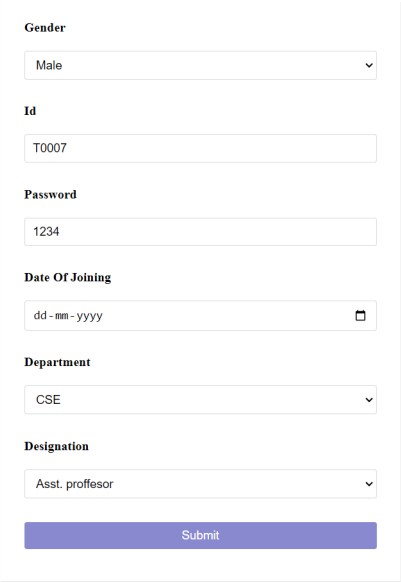


Figure 8.8: Teacher add

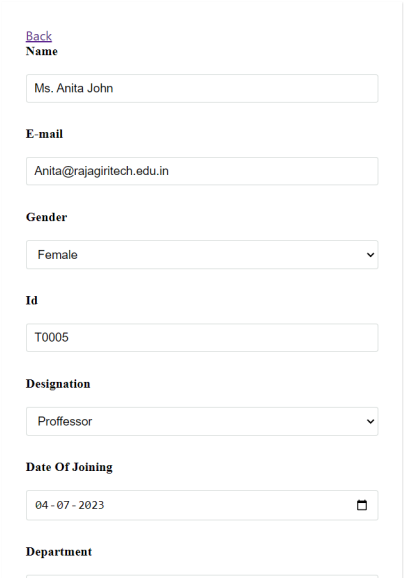


Figure 8.9: Teacher update

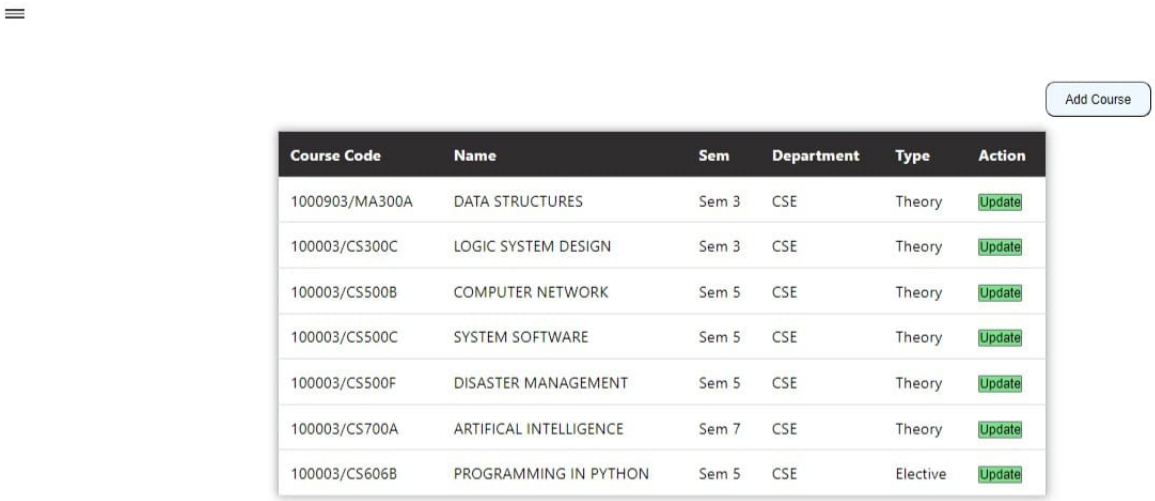


Figure 8.10: Subject view

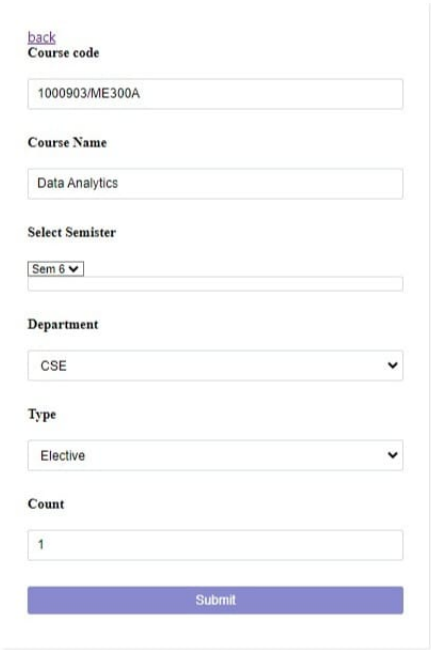


Figure 8.11: Subject add/update

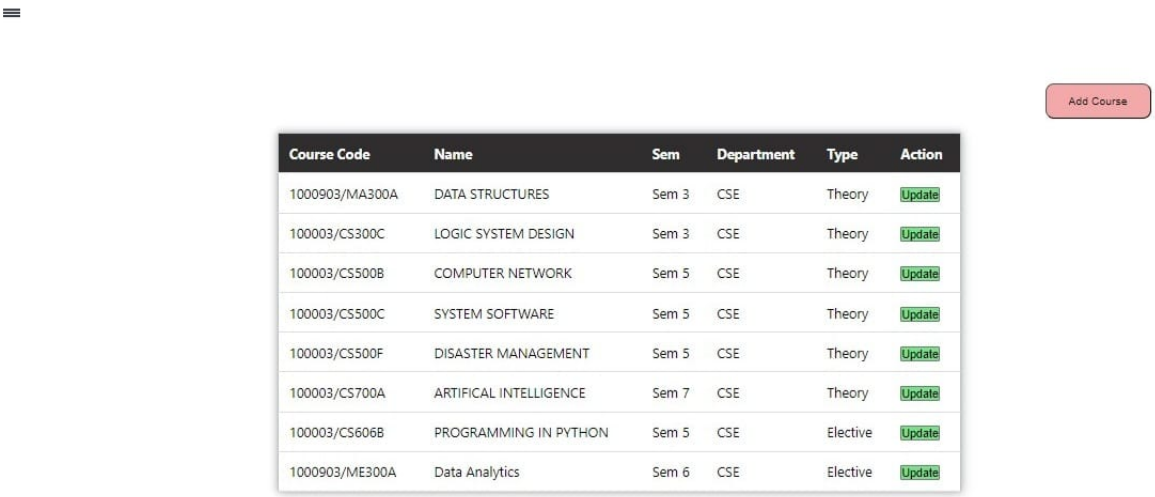


Figure 8.12: Subject view

Figure 8.13

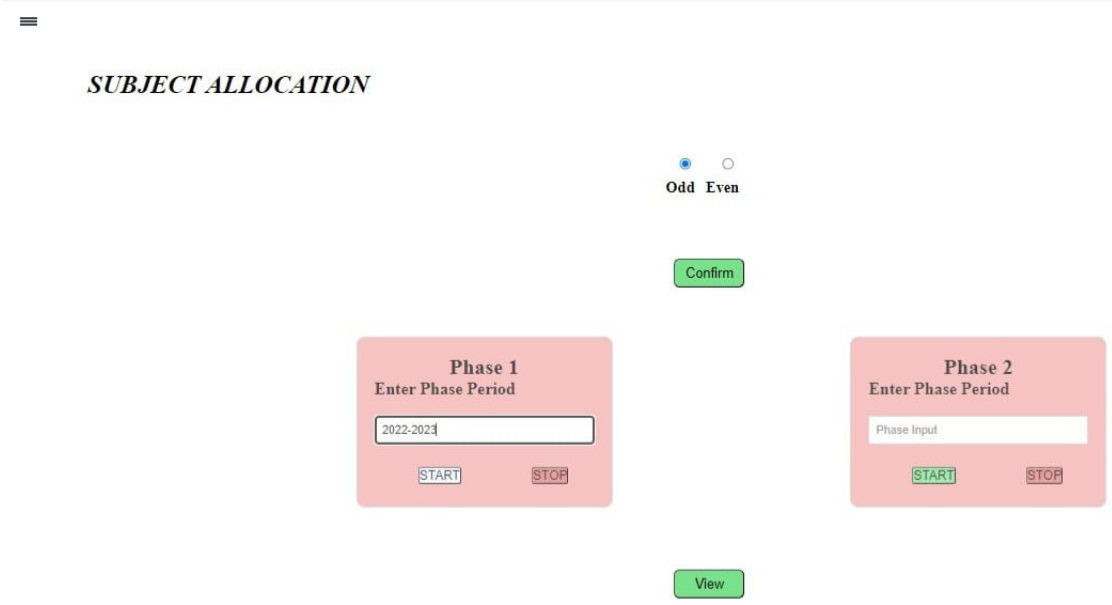


Figure 8.14: starting Phase 1

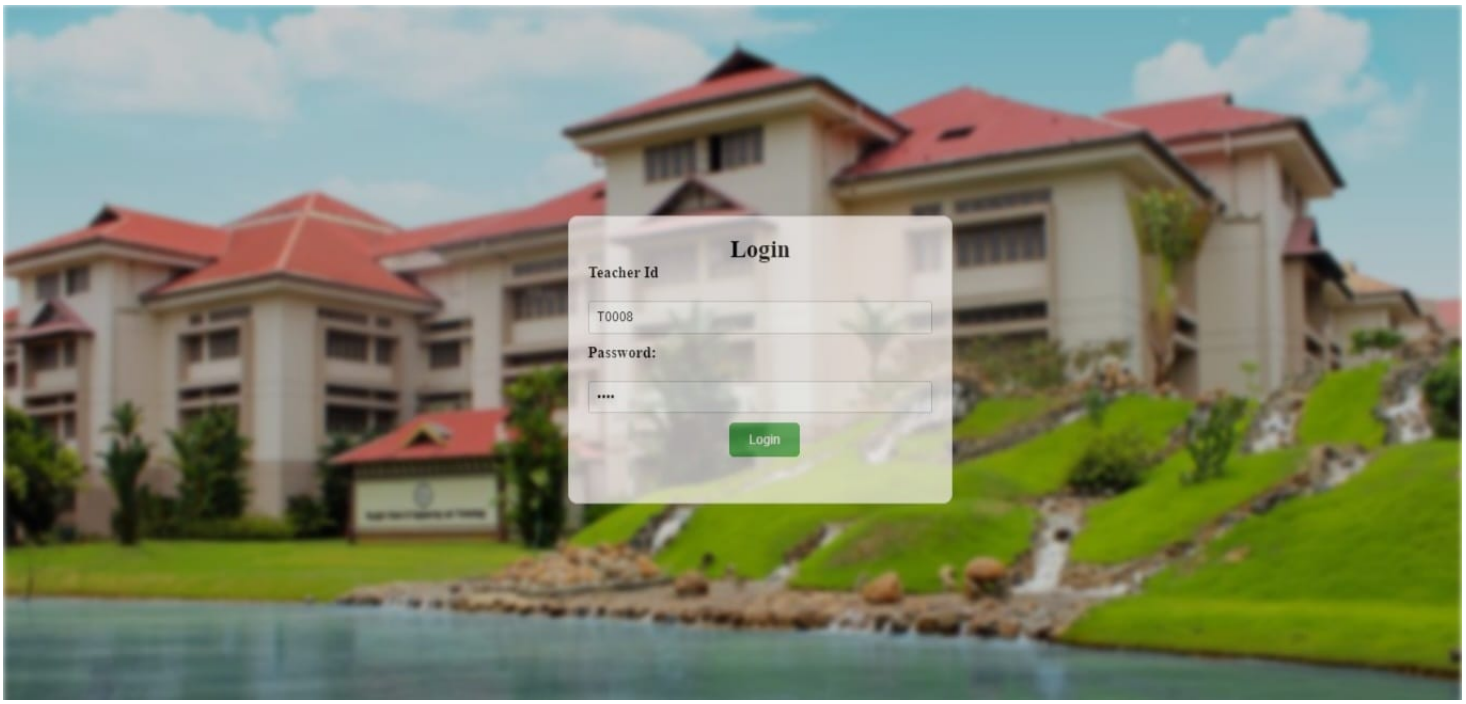


Figure 8.15: Teacher Login interface

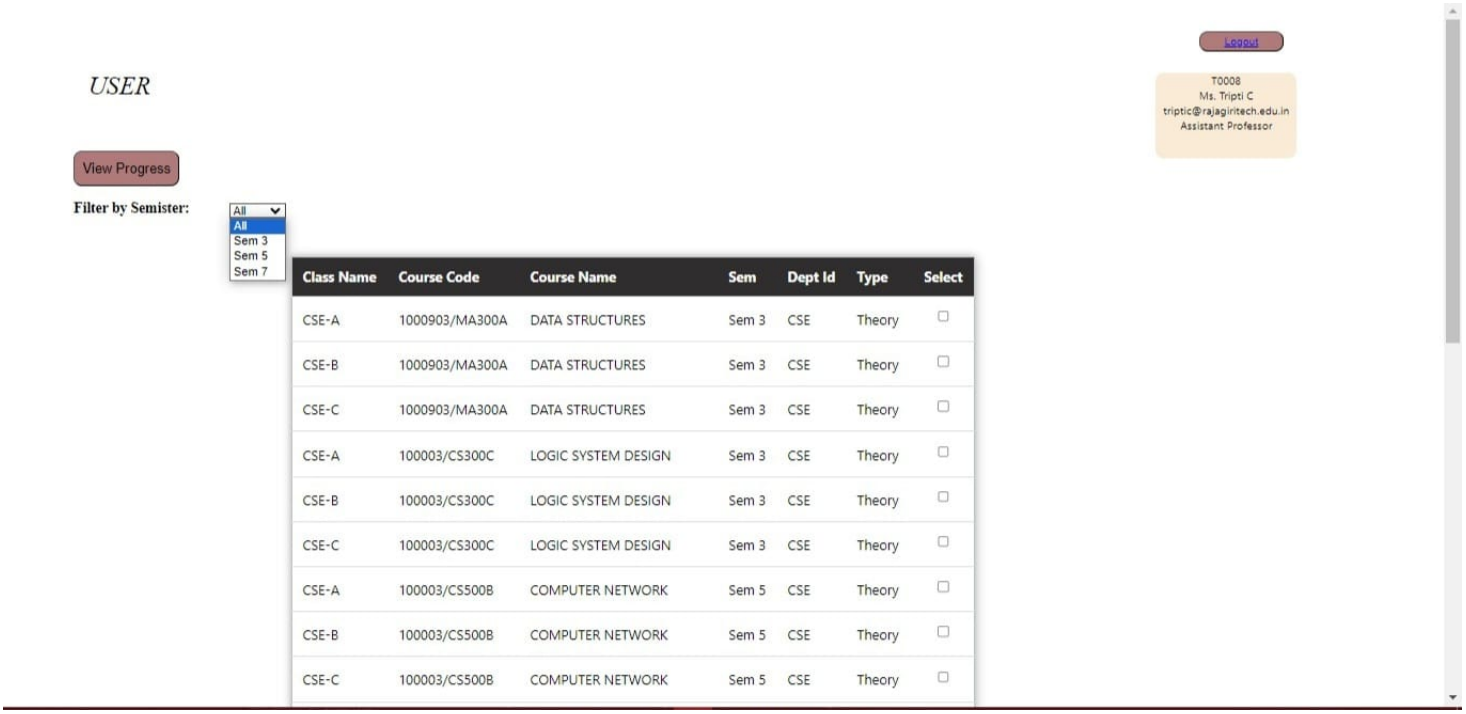


Figure 8.16: Interface for subject selection

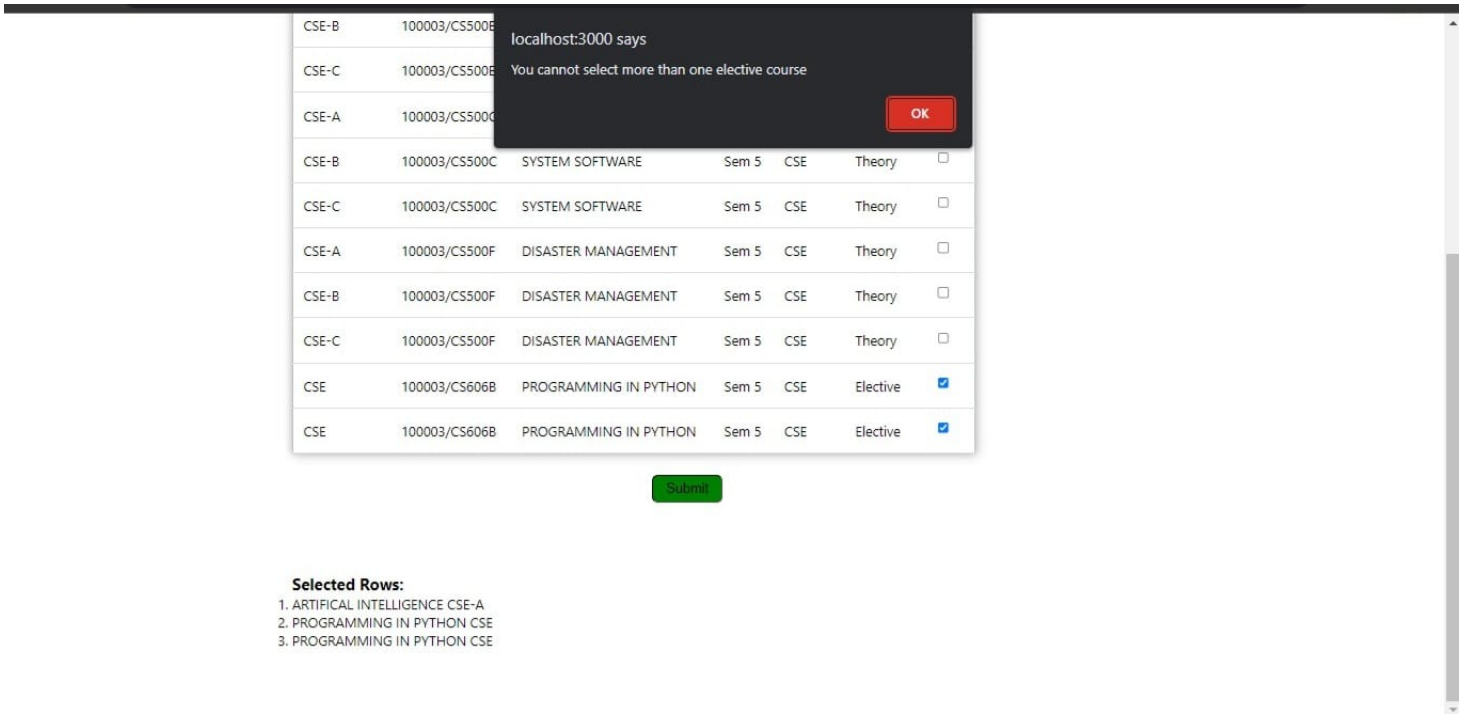


Figure 8.17: certain restrictions while selecting subject

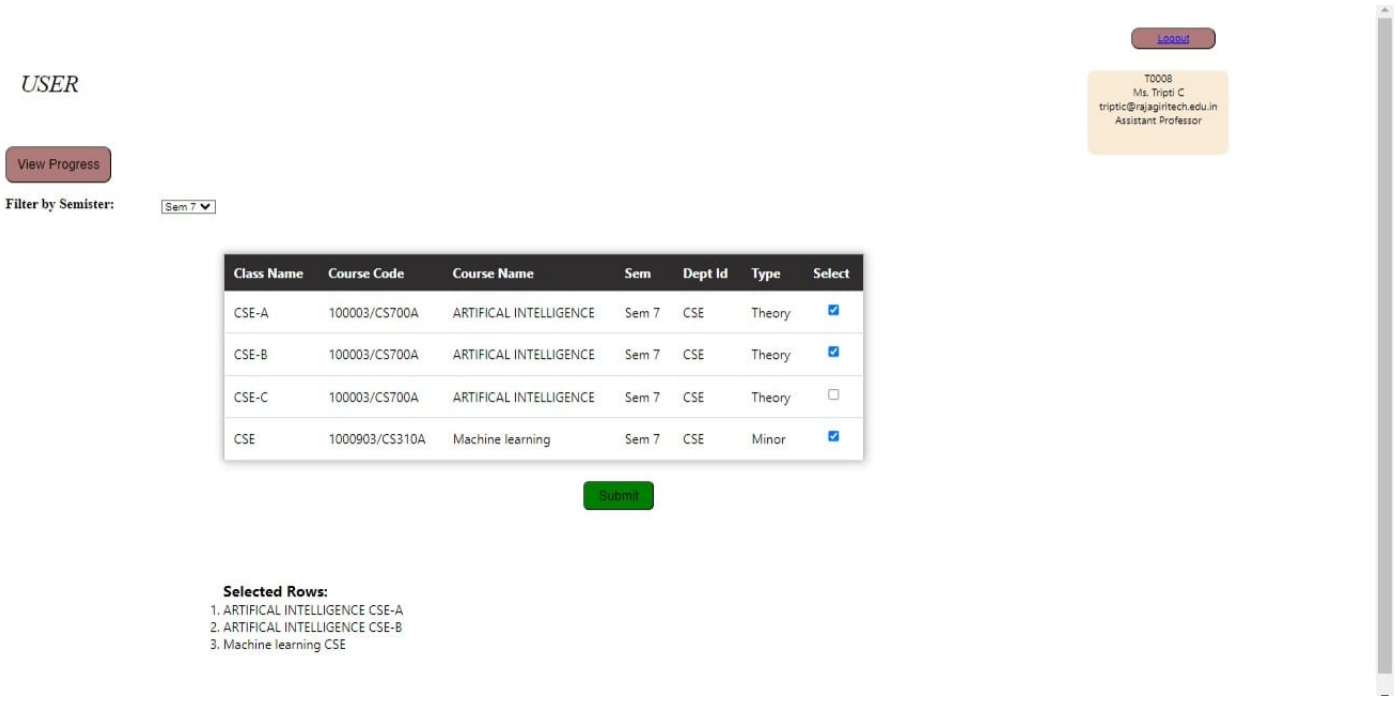


Figure 8.18: opting Subjects

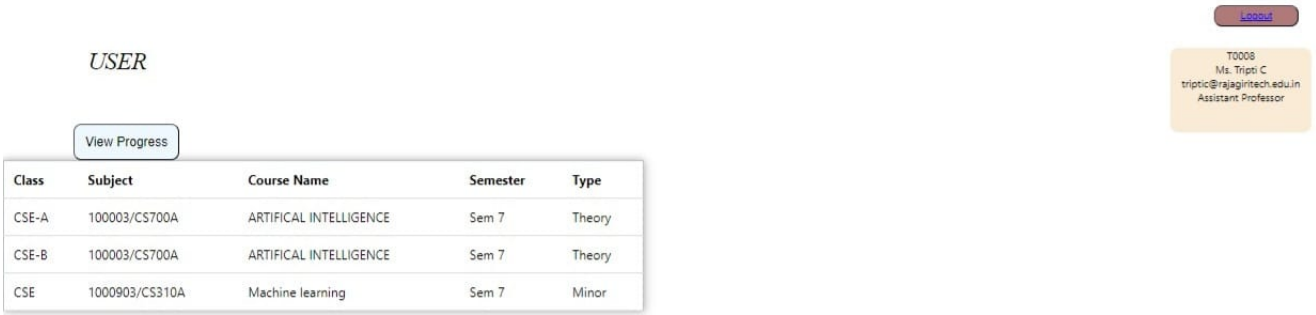


Figure 8.19: viewing Opted subjects

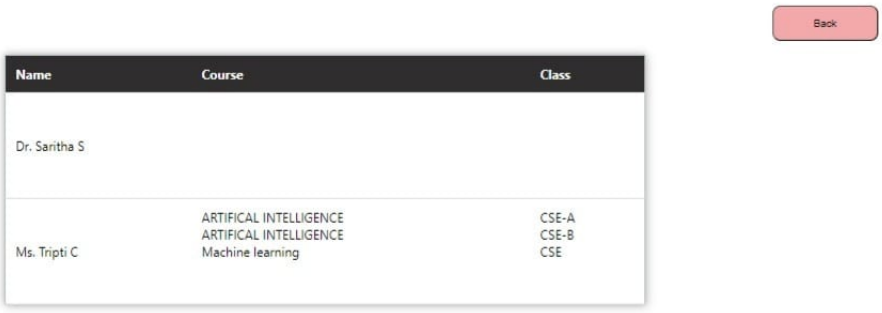


Figure 8.20: Displaying subjects opted by other teachers

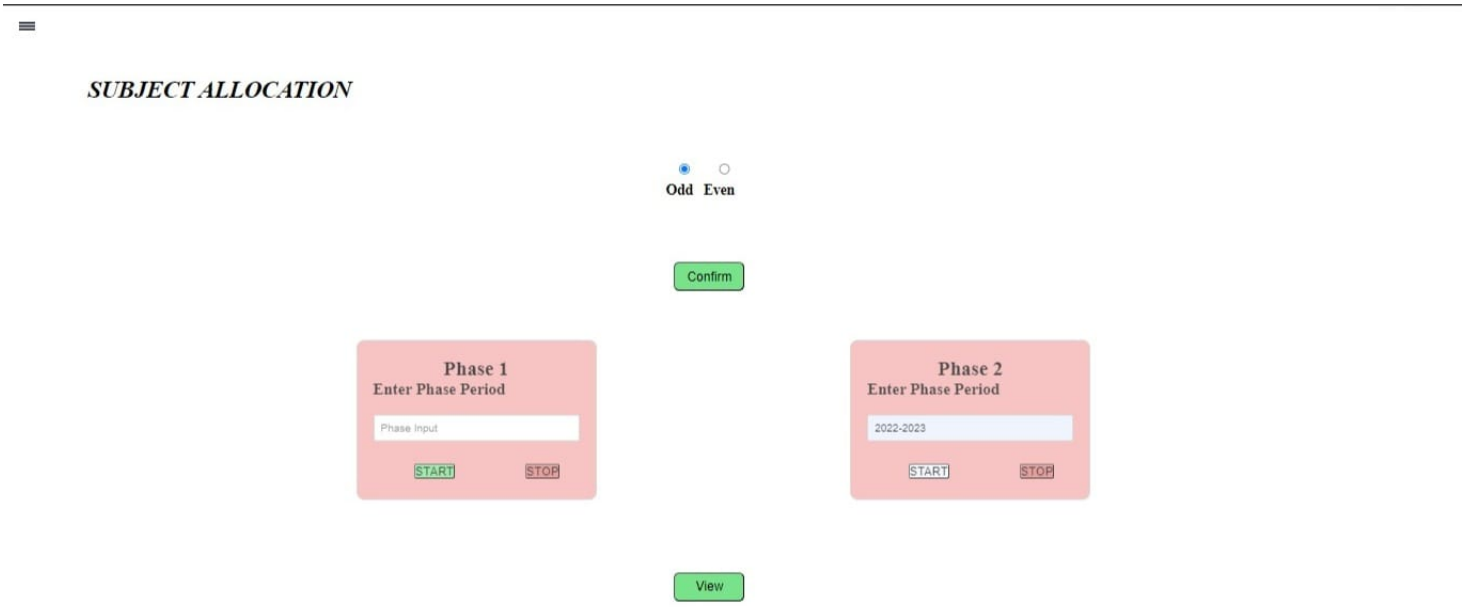


Figure 8.21: Starting Phase 2

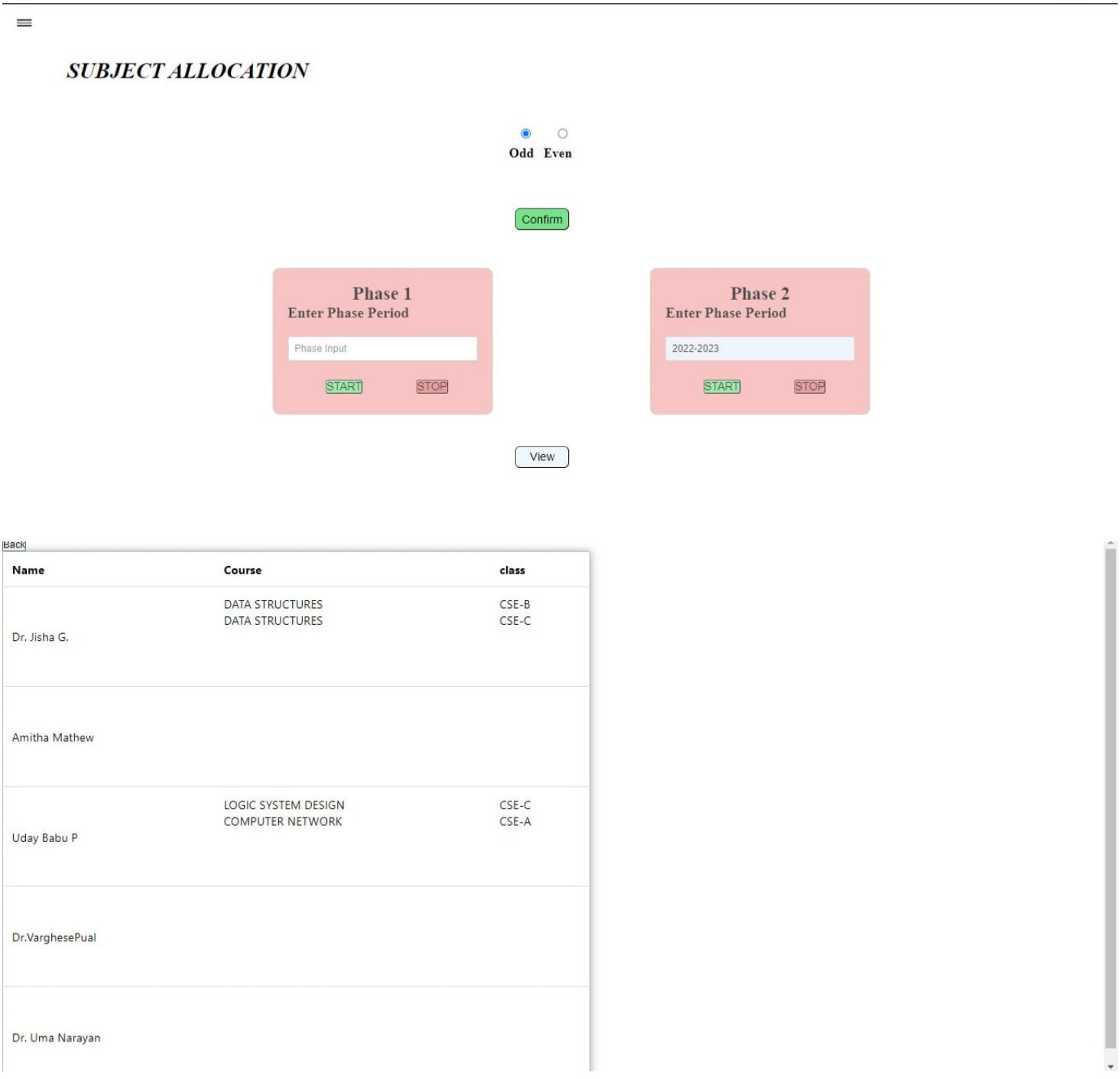


Figure 8.22: admin view during subject selection in phase 2

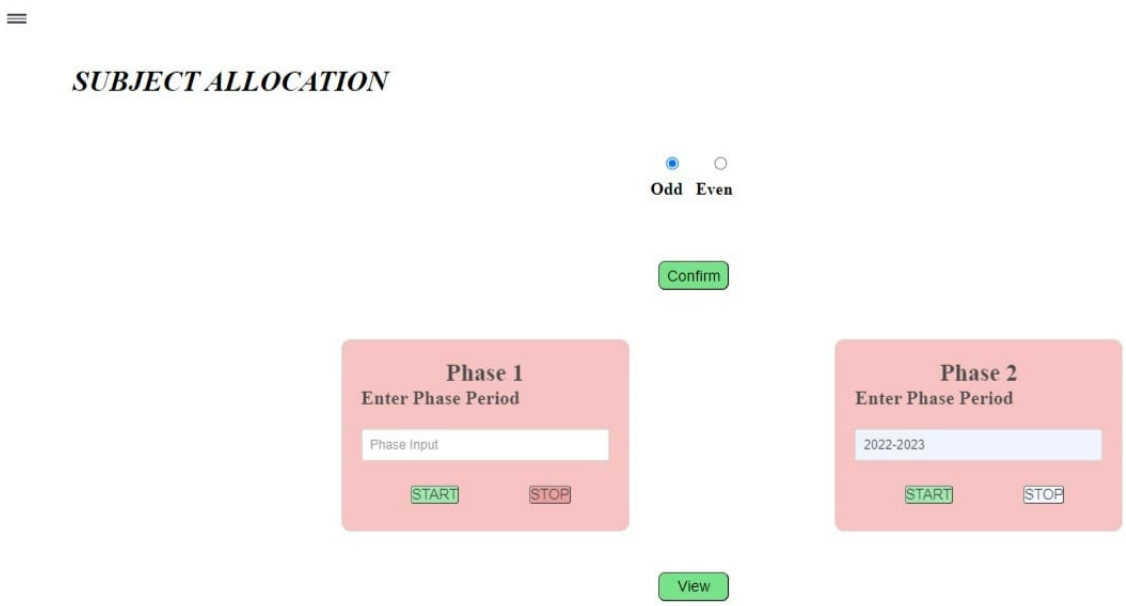
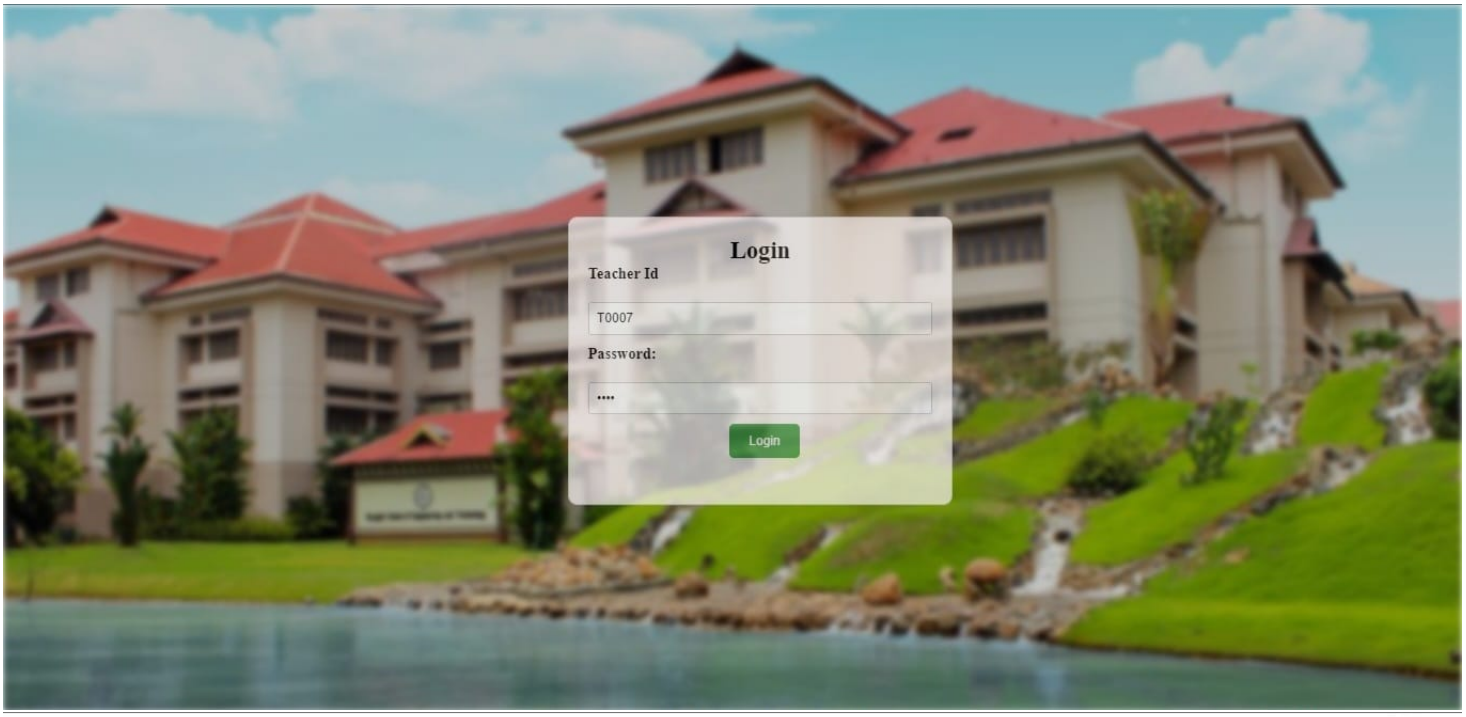


Figure 8.23: Stopping Phase 2



USER

Logout

T0007
Dr. Uma Narayan
uman@rajagiritech.edu.in
Assistant Professor

Select Academic Year

All
All
2022-2023

	Course	class	Academic Year
Dr. Saritha S	DATA STRUCTURES	CSE-A	2022-2023
Ms. Tripti C	ARTIFICAL INTELLIGENCE ARTIFICAL INTELLIGENCE	CSE-A CSE-B	2022-2023
Dr. Jisha G.	DATA STRUCTURES DATA STRUCTURES	CSE-B CSE-C	2022-2023
Amitha Mathew	LOGIC SYSTEM DESIGN LOGIC SYSTEM DESIGN	CSE-A CSE-B	2022-2023
Uday Babu P	LOGIC SYSTEM DESIGN COMPUTER NETWORK	CSE-C CSE-A	2022-2023
Dr.VarghesePual	COMPUTER NETWORK	CSE-B	2022-2023
Dr. Uma Narayan	Machine learning ARTIFICAL INTELLIGENCE	CSE CSE-C	2022-2023

Figure 8.24: Final view After allocation

Chapter 9

Risks and Challenges

1. System Security: As the system handles sensitive teacher and subject data, maintaining robust security measures is essential to protect against potential breaches or unauthorized access.

2. Scalability: As the college grows or subject allocations become more complex, the system should be able to handle a larger number of teachers and subjects without compromising performance.

3. Integration with Existing Systems: Integrating the automated subject allocation system with other college management systems may pose integration challenges, requiring compatibility and seamless data exchange.

Chapter 10

Conclusion

In conclusion, the Seniority-based Subject Selection Python program is a powerful tool that enables teachers to make informed decisions about subject selection, taking into account their expertise, interests, and institutional requirements. By considering seniority, the program recognizes and rewards teachers' years of service, fostering their career development and boosting motivation.

Through this program, teachers can deliver high-quality instruction by selecting subjects aligned with their expertise. This leads to a more engaging and effective learning environment, benefiting students' educational experiences and outcomes.

Additionally, the program strikes a balance between individual teacher preferences and institutional needs. It ensures that the curriculum remains well-rounded and meets educational objectives while allowing teachers to teach subjects they are passionate about.

By investing in the Seniority-based Subject Selection Python program, educational institutions can enhance the overall educational experience. Students benefit from instruction delivered by knowledgeable and enthusiastic teachers, promoting their academic growth and personal development. This program empowers teachers, supports their professional advancement, and ultimately contributes to educational excellence.

References

[1] . Matsuo and T. Fujimoto, "An effective lecture/class allocation method based on users' profiles in elective subjects, " IRI -2005 IEEE International Conference on Information Reuse and Integration, Conf, 2005., Las Vegas, NV, USA, 2005, pp. 160-165, doi: 10.1109/IRI-05.2005.1506467.

[2] . Parthiban, K. Ganesh, S. Narayanan and R. Dhanalakshmi, "Preferences based decision-making model (PDM) for faculty course assignment problem," 2004 IEEE International Engineering Management Conference (IEEE Cat. No.04CH37574), Singapore, 2004, pp. 1338-1341 Vol.3, doi: 10.1109/IEMC.2004.1408912

Appendix A: Front End

Login

```
import React from "react";
import Home from "./Home";
import Subject from "./Subject";
import Addstaff from "./Addstaff";
import ViewTeacher from "./ViewTeacher";
import { Routes, Route, Link } from "react-router-dom";
import Update from "./Update";
import ALogin from "./ALogin";
import ViewSubject from "./ViewSubject";
import Login1 from "./Login1";
import ViewDept from "./ViewDept";
import Adddept from "./Adddept";
import Sub from "./sub";
import Phase from "./Phase";
import Updatesub from "./Updatesub";
import TLogin from "./TLogin";
import Teacherpage from "./Teacherpage";
import Updatedept from "./Updatedept";
import Progress from "./Progress";
import Admprog from "./Admprog";
function Login() {

   return(
      <div>


         <div>
         <Routes>
            <Route path="/" element={<Login1 />}></Route>
      <Route path="/login" element={<Login1 />}></Route>
      <Route path="/login/admin" element={<ALogin />}></Route>
      <Route path="/login/teacher" element={<TLogin />}></Route>

      <Route path="/home" element={<Home />}></Route>
      <Route path="/teacher/:id" element={<Teacherpage />}></Route>
      <Route path="/teacher/progress/:id" element={<Progress />}></Route>
      <Route path="/home/viewteacher" element={<ViewTeacher />}></Route>

      <Route path="/home/sub" element={<Sub />}></Route>

      <Route path="/home/sub/phase" element={<Phase />}></Route>
      <Route path="/home/sub/phase/prog" element={<Admprog />}></Route>
```

```
        <Route path="/home/viewsubject" element={<ViewSubject />}></Route>
        <Route path="/home/viewdepartment" element={<ViewDept />}></Route>
        <Route path="/home/subject" element={<Subject />}></Route>
        <Route path="/home/department" element={<Adddept />}></Route>
        <Route path="/home/addstaff" element={<Addstaff />}></Route>
        <Route path="/update/:tid" element={<Update />}></Route>
        <Route path="/updatesub/:subid" element={<Updatesub />}></Route>
        <Route path="/updatedept/:depid" element={<Updatedept />}></Route>
</Routes>
          </div>

        </div>
    );
}
export default Login;
```

Login interface

```
import React from 'react'
import "./login1.css"
import "./but.css"
import { useNavigate } from 'react-router-dom'

const Login1 = () => {
  let navigate=useNavigate();
  function handleAdminButtonClick(){
    navigate("/login/admin");
  }
  function handleTeacherButtonClick(){
    navigate("/login/teacher");
  }
  return (
    <div>
        <div className='image'></div>
        {/* <img className='logo' src='./images/rsetlogo.jpg'/> */}
        <h2 id='heading'>RAJAGIRI MANAGEMENT SYSTEM</h2>
        <button className='admbutt' onClick={handleAdminButtonClick}>Admin</button>
        <button className='teacbutt' type="button"
onClick={handleTeacherButtonClick}>Teacher</button>
    </div>
  )
}
```

```
export default Login1
```

Css for Login Interface

```
.image{
    background-image: url('abc.jpg');
    background-position: center;
    background-repeat: no-repeat;

}

.logo{
    position: relative;
    z-index: 100;
    height: 15vh;
    width: 7vw;
    margin-top: 2vh;
    margin-left: 2vh;
}


#heading{
    position: relative;
    font-family: 'Poppins';
    font-size: 40px;
    text-align: center;
    top: 22vh;
    font-style:bold;
    font-weight: 4000;
    text-shadow: -1px -1px 0 #ffffff, 1px -1px 0 #ffffff, -1px 1px 0 #ffffff, 1px 1px 0 #ffffff;
    color: rgb(8, 8, 8);
}

.admbutt{
    position: relative;
    width: 20vh;
    height: 7vh;
    margin-top: 35vh;
    margin-left: 80vh;
    font-family: 'Monospace';
    font-size: large;
    border-radius: 7px;
```

```
    border: -20px;
    border-color: black;
    background-color: white;
    transition: all 300ms ease-in-out;
}
.admbutt:hover{
    position: relative;
    width: 20vh;
    height: 7vh;
    margin-top: 35vh;
    margin-left: 80vh;
    font-family: 'Monospace';
    font-size: large;
    border-radius: 7px;
    border: -20px;
    background-color: black;
    color:white;
}
.teacbutt{
    position: relative;
    width: 20vh;
    height: 7vh;
    margin-top: 35vh;
    margin-left: 5vh;
    font-family: 'Monospace';
    font-size: large;
    border-radius: 7px;
    border: -20px;
    border-color: black;
    background-color: white;
    transition: all 300ms ease-in-out;
}
.teacbutt:hover{
    position: relative;
    width: 20vh;
    height: 7vh;
    margin-top: 35vh;
    margin-left: 5vh;
    font-family: 'Monospace';
    font-size: large;
    border-radius: 7px;
    border: -20px;
    border-color: black;
    background-color: black;
```

```
    color: white;
}
```

TeacherLogin Interface

```
import React from "react";
import { useState } from "react";
import { useNavigate } from "react-router-dom";
function TLogin(){
    const [values, setValues] =useState({Tid: '',password: ''});
 const navigate=useNavigate();

 const submitForm = (e)=> {
    e.preventDefault();

    const endpoint = 'http://127.0.0.1:8000/home/teacherlogin'
        e.preventDefault()
        const payload = {
          Tid: values.Tid,
          password: values.password
        }
        console.log(payload)

        fetch(endpoint,
            {
                method: 'POST',
                headers: {
                    'Accept': 'application/json',
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(payload)
            })
            .then(response => response.json())
            .then(data => {
              if(data==="SUCCESS")
              {
                  const tid=values.Tid;
                navigate('/teacher/'+tid);

              }
              else{
```

```
                alert("Please check your login")
              }

            })



}
   return(
      <div className='back'>
         <div className="image"></div>
         <div className="box1">
         <h2>Login</h2>
         <form  onSubmit={submitForm}>
         <label >Teacher Id</label>
         <input className='' type="text" placeholder="teacher Id"  value={values.Tid}
onChange={e => setValues({...values,Tid: e.target.value})}></input>

         <label >Password: </label>
         <input className=''  type="password" placeholder="Password"
value={values.password} onChange={e => setValues({...values,password:
e.target.value})}></input>

         <button className='submit'>Login</button>

      </form>
      </div> </div>
   );

}
export default TLogin
```

Css For Teacher/Admin Login Interface

```
.image{
  background-image: url('abc.jpg');
  filter:brightness(0.5) blur(3px) ;
  -webkit-filter: blur(3px);
  height: 100vh;
  width: 100vw;
  background-repeat: no-repeat;
  background-size: cover;
  background-position: center;
```

```
  z-index: -1;
  position: fixed;

}


.box1 {
  position: fixed;
  margin-top: 30vh;
  width: 400px;
  height:300px;
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
  margin-left: 80vh;
  background-color: #f9f4f4;
  /*background-image:
url('https://images.pexels.com/photos/7130560/pexels-photo-7130560.jpeg?cs=srgb&dl=pexels-
codioful-%28formerly-gradienta%29-7130560.jpg&fm=jpg');*/
  padding: 20px;
  opacity: 0.7;
  border: 1px solid #ccc;
  border-radius: 10px;

}

h2 {
  text-align:center;
  font-family: 'Caprasimo';
  font-size: 25px;
}

label {
  display: block;
  font-weight: bold;
  font-family: 'PT Sans Narrow';
  margin-bottom: 20px;
}

input {
  width: 100%;
  padding: 8px;
  margin-bottom: 10px;
  border: 1px solid #ccc;
```

```
  border-radius: 3px;
}

.submit{
  width: 10vh;
  margin-left: 20vh;
  background-color: #4CAF50;
  color: #fff;
  padding: 10px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

.submit:hover {
  background-color: #45a049;
}
```

Teacherpage

```
import React from 'react'
import { useNavigate, useParams } from 'react-router-dom'
import { Link } from 'react-router-dom';
import { useEffect } from 'react';
import { useState } from 'react';
import Prog from './prog';
import "./teacherstyle.css"
function Teacherpage () {

    const {id} =useParams();//obtain id of the logged in teacher
    console.log(id);

    const [editUser, setEditUser]=useState({ tname: "",tmail: "",gender: "",tid: "",pos:"",password:
"",year: "",depid: ""});//store logged in teacher info
    const [phase, setPhase]=useState({});//store phaseview
    const [view, setView]=useState([]);//subject selected
     const [temclass, setTemclass]=useState([]);
     const [semtype, setSemtype]=useState([]);
    const [subsel, setSubsel]=useState([]);
    const [depsel, setDepsel]=useState([]);
```

```
const [final, setFinal]=useState([]);
const [tea, setTea]=useState([]);
const [selectedRows, setSelectedRows] = useState([]);

useEffect(()=>{//fetching required datas from API
     const getUser= async()=>{
         const reqData= await fetch('http://127.0.0.1:8000/home/teacherupdation/'+id);
         const resData= reqData.json();
         setEditUser(await resData);
     }
     getUser()

     const fetchStat = () =>{
       fetch('http://127.0.0.1:8000/home/phaseview')
          .then((response) => response.json())
          .then(data => setPhase(data))
   }
    fetchStat()

    const fetchtea = () =>{
     fetch('http://127.0.0.1:8000/home/viewteacher/')
         .then((response) => response.json())
         .then(data => setTea(data))
 }
  fetchtea()

    const fetchtype = () =>{
     fetch('http://127.0.0.1:8000/home/semtype')
         .then((response) => response.json())
         .then(data => setSemtype(data))
 }
  fetchtype()

  const fetchsub = () =>{
   fetch('http://127.0.0.1:8000/home/subject/')
       .then((response) => response.json())
       .then(data => setSubsel(data))
}
 fetchsub()
  const fetchdept = () =>{
     fetch('http://127.0.0.1:8000/home/department')
         .then((response) => response.json())
         .then(data => setDepsel(data))
 }
```

```
    fetchdept()

   const fetchfinal = () =>{
    fetch('http://127.0.0.1:8000/home/teachersubs')
        .then((response) => response.json())
        .then(data => setFinal(data))
 }
  fetchfinal()

  },[]);
  console.log(final)
  const [classtab, setClasstab]=useState([])
  const [data, setData] = useState([]);
  const [isLoading, setIsLoading] = useState(true);
  const [filter, setFilter] = useState('');
  const [selectedItems, setSelectedItems] = useState([]);
  useEffect(() => {
    fetchData();
  }, []);

  const fetchData = async () => {
    try {
      //fetching data from class division table
      const response = await fetch('http://127.0.0.1:8000/home/teacherlogin/division');
      const jsonData = await response.json();
      setTemclass(jsonData);
      setIsLoading(false);
    } catch (error) {
      console.error('Error fetching data:', error);
      setIsLoading(false);
    }
  };

//mapping

 const classtab1 = temclass.map(item1 => {
  const matchingItem = subsel.find(item2 => item2.subid === item1.subject);
  return { ...item1, ...matchingItem };
  });

  let sem;
  if(semtype.length>0){
    sem=semtype[0].sem
  }
```

```
  console.log(sem);


  const classtab2 = classtab1.map(item1 => {
    const matchingItem = depsel.find(item2 => item2.depid === item1.depid);
    return { ...item1, ...matchingItem };
    });
  console.log(classtab1)

            //odd or even
            let icd=-1; //indes for classtab
      if(sem==="odd")
      {
          for(let i=0; i<classtab2.length; i++)
          {
            if(classtab2[i].sem==="Sem 1" ||classtab2[i].sem==="Sem 3" ||
            classtab2[i].sem==="Sem 5" ||
            classtab2[i].sem==="Sem 7")
            {
              icd=icd+1;
              classtab[icd]=classtab2[i];
            }
          }
      }
      else{
        for(let i=0; i<classtab2.length; i++)
          {
            if(classtab2[i].sem==="Sem 2" ||classtab2[i].sem==="Sem 4" ||
            classtab2[i].sem==="Sem 6" ||
            classtab2[i].sem==="Sem 8")
            {
              icd=icd+1;
              classtab[icd]=classtab2[i];
            }
          }
      }
console.log(classtab)

// progress
const navigate=useNavigate();
       const handleprogress=()=>{
        let demo="1"
        Prog(demo);
        navigate("/teacher/progress/"+id);
```

```
    }
let x;
  const sendSelectedValues = async () => {//sending selected course to Api on buttom press
    try {
      let itindex=-1,hcount=0,mcount=0,ecount=0;
      for(let i=0; i<selectedRows.length; i++)
      {
        itindex++;
        selectedItems[itindex]=selectedRows[i].classid;
        console.log(selectedRows[i].subtype)
        if(selectedRows[i].subtype==="E")
        {
          ecount=ecount+1;
        }
        if(selectedRows[i].subtype=="H")
        {
          hcount=hcount+1;
        }
        if(selectedRows[i].subtype==="M")
        {
          mcount=mcount+1;
        }
      }
      console.log(hcount)
      //sending subjects to api
      if(ecount >1 || hcount>1 || mcount>1)
      {
         if(ecount>1)
         {
          alert("You cannot select more than one elective course");
         }
         else if(hcount>1)
         {
          alert("You cannot select more than one honour course");
         }
         else if(mcount>1)
         {
          alert("You cannot select more than one minor course");
         }
      }
      else{
        for(let i=x; i<6; i++)
      {
        selectedItems[i]="";
```

```
          }
        const payload = {
          no:x,
          tid:editUser.tid,
          sub1:selectedItems[0],
          sub2:selectedItems[1],
          sub3:selectedItems[2],
          sub4:selectedItems[3],
          sub5:selectedItems[4],
          sub6:selectedItems[5]
      }
      console.log(payload)

        // Make API call with selectedRows
        await fetch('http://127.0.0.1:8000/home/subselect', {
          method: 'POST',
          body: JSON.stringify(payload),
          headers: {
            'Content-Type': 'application/json'
          }
        });

        }
      } catch (error) {
        console.log(error);
      }
       window.location.reload();
    };

    let available=0;
    for(let i=0; i<phase.length; i++)  //checking whether teacher available in current phase
    {
      if(editUser.tid==phase[i].tid)
      {
        available=1;
        break;
      }
    }
    console.log(available);

if(available===1) //if teacher is able to choose course
{
    let min=editUser.pos=="0"?2:(editUser.pos=="1"?1:1);
    console.log(min);
```

```
  var temp=[];
  for(let i=0; i<phase.length; i++)
  {
    if(editUser.tid==phase[i].tid)
    {
      temp[0]=phase[i];
      break;
    }
  }
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
let ifd=-1 //index for data array
for( let i=0; i< classtab.length; i++)//filtering out subjects opted by previos phase
{
  if(classtab[i].classalloc!=="1" && classtab[i].depid===editUser.depid)
  {
    ifd=ifd+1;
    data[ifd]=classtab[i];
  }

}

console.log(data)

console.log(classtab)
//filtering
const handleFilterChange = (event) => {
setFilter(event.target.value);
};

const filteredData = data.filter((item) =>
filter === '' ? true : item.sem === filter
);

if (isLoading) {
return <div>Loading...</div>;
}

let hcount;
const handleCheckboxChange = (event, row) => {   //checkbox
  if (event.target.checked) {
    setSelectedRows([...selectedRows, row]);
```

```
    } else {
      const updatedRows = selectedRows.filter((selectedRow) => selectedRow.classid !==
row.classid);
      setSelectedRows(updatedRows);
    }
  };

  x=selectedRows.length;
  var temp1=[];
  console.log(temp[0].sub1) //temp1 saves only subject allocated by this teacher
  if(temp[0].status=="OFF")
  {
  temp1[0]=temp[0].sub1;
  temp1[1]=temp[0].sub2;
  temp1[2]=temp[0].sub3;
  temp1[3]=temp[0].sub4;
  temp1[4]=temp[0].sub5;
  temp1[5]=temp[0].sub6;
  let k=-1;
  for(let i=0; i<5; i++)
  {
  for(let j=0; j<classtab.length; j++)
  {
    if(temp1[i]==classtab[j].classid)
    {
      k++;
      view[k]=classtab[j];
      break;
    }
  }
  }
  }

  console.log(view);

  if(temp[0].status=="OFF")
  {
  return (
    <div>
      <div className='imagebackground-app'></div>
        <div className='dash'>USER</div>
      <div>
      <button className='logout-butt'><Link to="/login" >Logout</Link></button>
      </div>
```

```
      <div className='teachpage-cont1'>
  <ul>
    <li className='contmargin1'>{editUser.tid}</li>
    <li className='list123'>{editUser.tname}</li>
    <li className='list123'>{editUser.tmail}</li>
    <li className='list123'>{editUser.pos=="0"?"Assistant
Professor":(editUser.pos=="1"?"Associate proffesor":"Proffessor")}</li>
  </ul>
</div>

          <div>
            <button className='progressbutt' onClick={handleprogress}> View Progress</button>
          </div>
    <table>
      <thead>
        <tr>
          <th>Class</th>
          <th>Subject</th>
          <th>Course Name</th>
          <th>Semester</th>
          <th>Type</th>
        </tr>
      </thead>
      <tbody>
        {view.map((item) => (
          <tr key={item.classid}>
            <td>{item.classname}</td>
            <td>{item.subject}</td>
            <td>{item.subname}</td>
            <td>{item.sem}</td>

<td>{item.subtype==="T"?"Theory":(item.subtype=="P"?"Practicals":(item.subtype=="E"?"Electiv
e":(item.subtype=="M"?"Minor":"Honours")))}</td>
          </tr>
        ))}
      </tbody>
    </table>
    </div>
  );
  }
  else
  {
  return (
    <div>
```

```
    <div className='imagebackground-app'></div>
      <div className='dash'>USER</div>
    <div>
    <button className='logout-butt'><Link to="/login" >Logout</Link></button>
    </div>
    <div className='teachpage-cont1'>
  <ul>
    <li className='contmargin1'>{editUser.tid}</li>
    <li className='list123'>{editUser.tname}</li>
    <li className='list123'>{editUser.tmail}</li>
    <li className='list123'>{editUser.pos=="0"?"Assistant
Professor":(editUser.pos=="1"?"Associate proffesor":"Proffessor")}</li>
  </ul>
</div>

        <div>
          <button className='progressbutt' onClick={handleprogress}> View Progress</button>
        </div>

    <div className='title-margin'>
    <label  htmlFor="filter">Filter by Semister:</label>
    </div>

    <div className='teach-sem-drop'>
    <select id="filter" value={filter} onChange={handleFilterChange}>
      <option className='teach-sem-drop' value="">All</option>
      {Array.from(new Set(data.map((item) => item.sem))).map((gender) => (
        <option key={gender} value={gender}>
          {gender}
        </option>
      ))}
    </select>
    </div>

    <table className='view-teach-select-container'>
      <thead>
        <tr className='header'>
                <th>Class Name</th>
                <th>Course Code</th>
                <th>Course Name</th>
                <th>Sem</th>
                <th>Dept Id</th>
                <th>Type</th>
                <th>Select</th>
```

```
          </tr>
        </thead>
        <tbody>
          {filteredData.map((row) => (
            <tr key={row.classid}>

                      <td>{row.classname}</td>
                      <td>{row.subject}</td>
                      <td>{row.subname}</td>
                      <td>{row.sem}</td>
                      <td>{row.depname}</td>

<td>{row.subtype==="T"?"Theory":(row.subtype=="P"?"Practicals":(row.subtype=="E"?"Elective"
:(row.subtype=="M"?"Minor":"Honours")))}</td>
                        <td>
                    <input
                      type="checkbox"
                      checked={selectedRows.some((selectedRow) => selectedRow.classid ===
row.classid)}
                      onChange={(event) => handleCheckboxChange(event, row)}
                      disabled={(selectedRows.length >= 6 && !selectedRows.some((selectedRow) =>
selectedRow.classid === row.classid) )}
                    />
                  </td>
            </tr>
          ))}
        </tbody>
      </table>

      <button className="updatebuttsubmit" onClick={sendSelectedValues}
disabled={selectedRows.length < min}>Submit</button>
    <div>

    </div>

   <div className='select-row-cont'>
    <h3>Selected Rows:</h3>
      {selectedRows.length > 0 ? (
        <ol type='1'>
          {selectedRows.map(idd => (
            <li  >{idd.subname} {idd.classname}</li>
          ))}
        </ol>
      ) : (
```

```
          <p>No course selected</p>
        )}
        </div>

    </div>
  );
  }

}
else{//if subject selection is not enabled for teacher
  const display = final.map((dictionary) => {
    return { ...dictionary, subname1: "",subname2: "",classname1: "",classname2: "",tname: "" };
  });

 for(let i=0;i< display.length; i++)
 {
   for(let j=0; j<tea.length; j++)
   {
    if(display[i].tid===tea[j].tid)
    {
      display[i].tname=tea[j].tname;
    }
   }
   for( let k=0; k<classtab.length; k++)
   {
    if(display[i].sub1===classtab[k].classid)
    {
      display[i].subname1=classtab[k].subname;
      display[i].classname1=classtab[k].classname;
    }
    if(display[i].sub2===classtab[k].classid)
    {
      display[i].subname2=classtab[k].subname;
      display[i].classname2=classtab[k].classname;
    }

   }
 }
 console.log(display);


 const handleFilterChange = (event) => {
  setFilter(event.target.value);
  };
```

```
 const filteredData = display.filter((item) =>
 filter === '' ? true : item.year === filter
 );



 return(
 <div>
 <div className='imagebackground-app'></div>
      <div className='dash'>USER</div>
    <div>
    <button className='logout-butt'><Link to="/login" >Logout</Link></button>
    </div>
    <div className='teachpage-cont1'>
 <ul>
   <li className='contmargin1'>{editUser.tid}</li>
   <li className='list123'>{editUser.tname}</li>
   <li className='list123'>{editUser.tmail}</li>
   <li className='list123'>{editUser.pos=="0"?"Assistant
Professor":(editUser.pos=="1"?"Associate proffesor":"Proffessor")}</li>
 </ul>
</div>


 <div>
 <label htmlFor="filter">Select Academic Year</label>
    <select id="filter" value={filter} onChange={handleFilterChange}>
      <option value="">All</option>
      {Array.from(new Set(display.map((item) => item.year))).map((gender) => (
        <option key={gender} value={gender}>
          {gender}
        </option>
      ))}
    </select>
 <table>
    <thead>
      <tr>
        <th>Name</th>
        <th>Course</th>
        <th>class</th>
        <th>Academic Year</th>
      </tr>
    </thead>
```

```
      <tbody>
       {filteredData.map((item) => (
         <tr key={item.id}>
           <td>{item.tname}</td>
           <td>{item.subname1 }<br/>
           {item.subname2 }<br/>
           </td>
           <td>{item.classname1}<br/>
           {item.classname2}<br/>
           </td>
           <td>{item.year}</td>
         </tr>
       ))}
      </tbody>
    </table>
  </div>
  </div>
 )
}
}

export default Teacherpage
```

Css for Teacherpage

```
.logout-butt{
    position: relative;
    margin-left: 83%;
    top: 0vh;
    height: 3vh;
    width: 12vh;
    text-decoration: none;
    border-radius: 10px;
    font-size: small;
    transition: all 300ms ease-in-out;
    background-color: rgb(176, 122, 122);
}

.logout-butt:hover{
    background-color:aliceblue;
}
```

```
.teachpage-cont1{
    position: relative;
    width: 20vh;
    height: 12vh;
    border-radius: 10px;
    background-color: antiquewhite;
    margin-top: 1vh;
    margin-left: 80%;
    text-align: center;
    border-color: black;
    font-size: small;
}

.contmargin1{
    margin-top: 3vh;
    list-style: none;
}

.list123{
    list-style: none;
}

.progressbutt{
    margin-top: -15vh;
    margin-left: 10vh;
    position: relative;
    height: 5vh;
    width: 15vh;
    text-decoration: none;
    border-radius: 10px;
    font-size: medium;
    transition: all 300ms ease-in-out;
    background-color: rgb(176, 122, 122);
}

.progressbutt:hover{
    background-color:aliceblue;
}

.title-margin{
    margin-left: 10vh ;
    margin-top: 2vh;
    width: 25vh;
    height: 5vh;
```

```
    font-size: 17px;
}

.teach-sem-drop{
    margin-left: 32vh ;
    margin-top: -5vh;
    width: 20vh;
    height: 5vh;

}

.view-teach-select-container{
    position: relative;
    top: 50%;
    left: 20%;
    margin-top: 3vh;

}

.updatebuttsubmit{
    margin-left: 45%;
    margin-top: 3vh;
    font-size: medium;
    border-radius: 7px;
    width: 10vh;
    height: 4vh;
    background-color: green;
    color: black;
}

.updatebuttsubmit::after{
    background-color: aliceblue;
    color: black;
}

.select-row-cont{
    position: relative;
    top: 50%;
    margin-top: 10vh;
    align-items: center;
    margin-bottom: 10vh;
    margin-left: 20%;
}
```

View Progress of other teachers during subject selection

```
import React, { useState } from 'react'
import { useEffect } from 'react'
import { useNavigate } from 'react-router-dom';
import { useParams } from 'react-router-dom';
function Progress(){

const {id} =useParams();
const navigate=useNavigate();

const [phase, setPhase]=useState([]);
const [depsel, setDepsel]=useState([]);
const [tea, setTea]=useState([]);
const [temclass, setTemclass]=useState([]);
const [subsel, setSubsel]=useState([]);

useEffect(()=>{//fetching required datas from API
  const getUser= async()=>{
       const reqData= await fetch('http://127.0.0.1:8000/home/phaseview');
       const resData= reqData.json();
       setPhase(await resData);
  }
  getUser()

 const fetchtea = () =>{
  fetch('http://127.0.0.1:8000/home/viewteacher/')
      .then((response) => response.json())
      .then(data => setTea(data))
}
fetchtea()


const fetchsub = () =>{
fetch('http://127.0.0.1:8000/home/subject/')
    .then((response) => response.json())
    .then(data => setSubsel(data))
}
fetchsub()
const fetchdept = () =>{
  fetch('http://127.0.0.1:8000/home/department')
      .then((response) => response.json())
      .then(data => setDepsel(data))
}
```

```
fetchdept()

const fetchclass = () =>{
  fetch('http://127.0.0.1:8000/home/teacherlogin/division')
      .then((response) => response.json())
      .then(data => setTemclass(data))
}
fetchclass()

},[]);

console.log(temclass)

const classtab1 = temclass.map(item1 => {
  const matchingItem = subsel.find(item2 => item2.subid === item1.subject);
  return { ...item1, ...matchingItem };
  });

  const classtab2 = classtab1.map(item1 => {
    const matchingItem = depsel.find(item2 => item2.depid === item1.depid);
    return { ...item1, ...matchingItem };
    });
  console.log(classtab2)



  const opt = phase.map((dictionary) => {
    return { ...dictionary, subname1: "",subname2: "",subname3: "",subname4: "",subname5:
"",subname6: "",classname1: "",classname2: "",classname3: "",classname4: "",classname5:
"",classname6: "",tname: "" };
  });




console.log(opt)

for(let i=0;i< opt.length; i++)
 {
   for(let j=0; j<tea.length; j++)
   {
    if(opt[i].tid===tea[j].tid)
    {
```

```
      opt[i].tname=tea[j].tname;
    }
   }
   for( let k=0; k<classtab2.length; k++)
   {
    if(opt[i].sub1===classtab2[k].classid)
    {
      opt[i].subname1=classtab2[k].subname;
      opt[i].classname1=classtab2[k].classname;
    }
    if(opt[i].sub2===classtab2[k].classid)
    {
      opt[i].subname2=classtab2[k].subname;
      opt[i].classname2=classtab2[k].classname;
    }
    if(opt[i].sub3===classtab2[k].classid)
    {
      opt[i].subname3=classtab2[k].subname;
      opt[i].classname3=classtab2[k].classname;
    }
    if(opt[i].sub4===classtab2[k].classid)
    {
      opt[i].subname4=classtab2[k].subname;
      opt[i].classname4=classtab2[k].classname;
    }
    if(opt[i].sub5===classtab2[k].classid)
    {
      opt[i].subname5=classtab2[k].subname;
      opt[i].classname5=classtab2[k].classname;
    }
    if(opt[i].sub6===classtab2[k].classid)
    {
      opt[i].subname6=classtab2[k].subname;
      opt[i].classname6=classtab2[k].classname;
    }

   }
  }

const handleclick =(e) =>{
    navigate("/teacher/"+id)
}

console.log(opt);
```

```
return(
<div>
  <div>
    <button className="adddeptbutt" onClick={handleclick}>Back</button>
  </div>
  <div>
  <table className='view-dep-container'>
      <thead>
        <tr className='header'>
          <th>Name</th>
          <th>Course</th>
          <th>Class</th>
        </tr>
      </thead>
      <tbody>
        {opt.map((item) => (
          <tr key={item.id}>
            <td>{item.tname}</td>
            <td>{item.subname1 }<br/>
            {item.subname2 }<br/>
            {item.subname3 }<br/>
            {item.subname4 }<br/>
            {item.subname5 }<br/>
            {item.subname6 }

            </td>
            <td>{item.classname1}<br/>
            {item.classname2}<br/>
            {item.classname3}<br/>
            {item.classname4}<br/>
            {item.classname5}<br/>
            {item.classname6}
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>

 </div>

)
```

```
}

export default Progress
```

CSS for Table

```
*{
    margin:0;
    padding: 0;
   }

   .header{
    background-color: #2f2d2d;
   color: #fff;
   }

.view-teach-container{
    position: relative;
    top:50%;
    left: 20%;
    margin-top: 15vh;
}

.view-sub-container{
    position: relative;
    top: 50%;
    left: 20%;
    margin-top: 15vh;

}

.view-dep-container{
    position: relative;
    top: 50%;
    left: 20%;
    margin-top: 15vh;

}
```

```
table{
 border-collapse: collapse;
 width: 800px;
 height: 100px;
 border: 1px solid #bdc3c7;
 box-shadow: 2px 2px 12px rgba(0,0,0,0.2), -1px -1px 8px rgba(0,0,0,0.2);
}

tr{
 transition: all .2s ease-in;
 cursor: pointer;
}

th,td{
 padding: 12px;
 text-align: left;
 border-bottom: 1px solid #ddd;
}
```

Appendix B: Back End

MODELS.PY

```
from django.db import models
# Create your models here.
class Department(models.Model):

depid=models.CharField(max_length=15,primary_key=True)

depname=models.CharField(max_length=10,default=None)

HODname=models.CharField(max_length=25,default=None)
    division=models.IntegerField(default=0)

class Subject(models.Model):

subid=models.CharField(max_length=25,primary_key=True)

subname=models.CharField(max_length=45)
    sem=models.CharField(max_length=10)

year=models.CharField(max_length=10,default='2022')
    depid=models.ForeignKey(Department,on_delete=models.CASCADE,default='D0001')

subtype=models.CharField(max_length=5,default='T')

class otherSubject(models.Model):

subid=models.CharField(max_length=25,primary_key=True)

subname=models.CharField(max_length=45)
    sem=models.CharField(max_length=10)
    year=models.CharField(max_length=10)
    depid=models.ForeignKey(Department,on_delete=models.CASCADE,default='D0001')

subtype=models.CharField(max_length=5,default='T')
    count=models.IntegerField()

class Teacher(models.Model):
    tname=models.CharField(max_length=15)

tmail=models.CharField(max_length=35,default=None)
    gender=models.CharField(max_length=10)

tid=models.CharField(max_length=10,primary_key=True)

#qualifications=models.CharField(max_length=20)

password=models.CharField(max_length=10)
    year=models.DateField()
    depid=models.ForeignKey(Department,on_delete=models.CASCADE,default='D0001')
    pos=models.IntegerField(default=0)

class AdminLogin(models.Model):

aid=models.CharField(max_length=20,primary_key=True)

password=models.CharField(max_length=10)

class TeacherLogin(models.Model):
    Tid=models.CharField(max_length=20)

password=models.CharField(max_length=20)

class TeacherSelection(models.Model):
    tid=models.ForeignKey(Teacher,on_delete=models.CASCADE)
    sub1=models.CharField(max_length=20)
    sub2=models.CharField(max_length=20)
    count=models.IntegerField()

selectionid=models.CharField(max_length=20,primary_key=True)
    year=models.CharField(max_length=20)



class ClassDivisions(models.Model):

classid=models.CharField(max_length=20,primary_key=True)
```

```
classname=models.CharField(max_length=20)

subject=models.ForeignKey(Subject,on_delete
=models.CASCADE)

depid=models.ForeignKey(Department,on_del
ete=models.CASCADE)

classalloc=models.CharField(max_length=10)
    exp=models.IntegerField()
    sem=models.CharField(max_length=20)
    subtype=models.CharField(max_length=20)



class Phase(models.Model):
    no=models.IntegerField()#no of subject
selected

pid=models.CharField(max_length=20,primary
_key=True)

tid=models.ForeignKey(Teacher,on_delete=mo
dels.CASCADE)
    alloc=models.IntegerField(default=0)

status=models.CharField(max_length=20,defa
ult='ON')#selection done
    exp=models.IntegerField()
    sub1=models.CharField(max_length=20)
    sub2=models.CharField(max_length=20)
    sub3=models.CharField(max_length=20)
    sub4=models.CharField(max_length=20)
    sub5=models.CharField(max_length=20)
    sub6=models.CharField(max_length=20)

academicyear=models.CharField(max_length=
20,default='2023')#phase year
    mail=models.CharField(max_length=20)

class phaseno(models.Model):

no=models.CharField(max_length=20,primary
_key=True)
    active=models.CharField(max_length=20)

class Final(models.Model):
    val=models.CharField(max_length=20)


class clash(models.Model):
    clashid=models.CharField(max_length=20)

class semtype(models.Model):
    sem=models.CharField(max_length=20)

class Phaseteacher(models.Model):
    tname=models.CharField(max_length=20)
    sub1=models.CharField(max_length=20)
    sub2=models.CharField(max_length=20)
    sub3=models.CharField(max_length=20)
    sub4=models.CharField(max_length=20)
    sub5=models.CharField(max_length=20)
    sub6=models.CharField(max_length=20)

classname1=models.CharField(max_length=20
,default="")

classname2=models.CharField(max_length=20
,default="")

classname3=models.CharField(max_length=20
,default="")

classname4=models.CharField(max_length=20
,default="")

classname5=models.CharField(max_length=20
,default="")

classname6=models.CharField(max_length=20
,default="")

class phaseget(models.Model):
    val=models.CharField(max_length=20)

class resolve(models.Model):
    rid=models.IntegerField()

SERILAIZER.PY
from rest_framework import serializers
from .models import
Teacher,Subject,AdminLogin,TeacherLogin,Clas
sDivisions,otherSubject
```

```
from .models import
Department,phaseno,Phase,TeacherSelection,
Final,clash,semtype,phaseget,Phaseteacher
class
Teacherserializer(serializers.ModelSerializer):
   class Meta:
      model=Teacher

fields=['tname','tmail','gender','tid','password'
,'year','depid','pos']

class
Departmentserializer(serializers.ModelSerializ
er):

   class Meta:
      model=Department

fields=['depid','depname','HODname','division
']



class
Subjectserializer(serializers.ModelSerializer):
   class Meta:
      model=Subject

fields=['subid','subname','sem','depid','subtyp
e']
class
AdminLoginserializer(serializers.ModelSerializ
er):
   class Meta:
      model=AdminLogin
      fields=['aid','password']

class
TeacherLoginserializer(serializers.ModelSeriali
zer):
   class Meta:
      model=TeacherLogin
      fields=['Tid','password']

class
Finalserializer(serializers.ModelSerializer):
   class Meta:
      model=Final
      fields=['val']
class
TeacherSelectionserializer(serializers.ModelSe
rializer):
   class Meta:
      model=TeacherSelection

fields=['tid','sub1','sub2','count','selectionid','y
ear']

class
ClassDivisionsserializer(serializers.ModelSeriali
zer):
    class Meta:
      model=ClassDivisions

fields=['classid','classname','subject','depid','cl
assalloc','exp','sem','subtype']



class
Phaseserializer(serializers.ModelSerializer):
   class Meta:
      model=Phase

fields=['no','tid','alloc','status','sub1','sub2','su
b3','sub4','sub5','sub6','academicyear','mail']



class
Phasenoserializer(serializers.ModelSerializer):
   class Meta:
      model=phaseno
      fields=['no','active']

class
clashserializer(serializers.ModelSerializer):
   class Meta:
      model=clash
      fields=['clashid']



class
Addclassserializer(serializers.ModelSerializer):
   class Meta:
      model=Phase
```

```
fields=['no','tid','sub1','sub2','sub3','sub4','sub
5','sub6']

class
otherSubjectserializer(serializers.ModelSeriali
zer):
   class Meta:
      model=otherSubject

fields=['subid','subname','sem','depid','subtyp
e','count']

class
Semtypeserializer(serializers.ModelSerializer):
   class Meta:
      model=semtype
      fields=['sem']

class
phasegetserializer(serializers.ModelSerializer):

   class Meta:
      model=phaseget
      fields=['val']

class
phaseteacherserializer(serializers.ModelSeriali
zer):

   class Meta:
      model=Phaseteacher

fields=['tname','sub1','classname1','sub2','clas
sname2','sub3','classname3',

'sub4','classname4','sub5','classname5','sub6',
'classname6']
```

URLS.PY

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [

path('viewteacher/',views.TeacherView.as_vie
w()),

path('subject/',views.SubjectView.as_view()),

path('department',views.DepartmentView.as_
view()),

path('adminlogin/',views.AdminLoginView.as_
view()),

path('teacherlogin',views.TeacherLoginView.as
_view()),

path('selectionfinal',views.Finalview.as_view()
),

path('teacherlogin/division',views.ClassDivisio
nsview.as_view()),

path('teacherdeletion/<str:pk>',views.Teacher
Deletion.as_view()),

path('teacherupdation/<str:pk>',views.Teache
rUpdation.as_view()),

path('subjectupdation/<str:pk>',views.Subject
Updation.as_view()),

path('departmentupdation/<str:pk>',views.De
partmentUpdation.as_view()),
   path('phase1',views.phase1view.as_view()),
   path('phase2',views.phase2view.as_view()),

path('phasestatus',views.phasestatusview.as_
view()),
   path('split',views.split.as_view()),

path('othersplit',views.othersplitview.as_view(
)),
```

```
path('phaseupdate/<str:pk>',views.Phaseupdate.as_view()),
   path('subselect',views.subselect.as_view()),

path('phaseview',views.phaseview.as_view()),

path('teachersubs',views.TeacherSelectionview.as_view()),
   path('clash',views.clashview.as_view()),

path('semtype',views.semtypeview.as_view()),

path('getview',views.phasegetview.as_view()),

path('tableview',views.Phaseteacherview.as_view()),
]
```

ADMIN.PY

```
from django.contrib import admin
from .models import
Teacher,Subject,Department,AdminLogin,TeacherLogin,phaseget
from .models import
phaseno,ClassDivisions,Phase,TeacherSelection,Final,semtype
from .models import
Phaseteacher,clash,resolve

admin.site.register(Teacher)
admin.site.register(Department)
admin.site.register(Subject)
admin.site.register(AdminLogin)
admin.site.register(TeacherLogin)
admin.site.register(TeacherSelection)
admin.site.register(ClassDivisions)
admin.site.register(Phase)
admin.site.register(phaseno)
admin.site.register(Final)
admin.site.register(semtype)
admin.site.register(phaseget)
admin.site.register(Phaseteacher)
admin.site.register(clash)
admin.site.register(resolve)
# Register your models here.
```

VIEW.PY

```
from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse,
QueryDict
from rest_framework import generics
from .models import (
   Teacher,
   Subject,
   AdminLogin,
   TeacherLogin,
   TeacherSelection,
   ClassDivisions,
   Final,
   otherSubject,
   semtype,
   phaseget,
   Phaseteacher,
)
from .models import Department, phaseno,
Phase,clash,resolve
from .serializers import (
   Teacherserializer,
   Subjectserializer,
   AdminLoginserializer,
   TeacherLoginserializer,
   Finalserializer,
   otherSubjectserializer,
   phasegetserializer,
   phaseteacherserializer,
)
from .serializers import (
   TeacherSelectionserializer,
   Departmentserializer,
   Phaseserializer,
   clashserializer,
   Semtypeserializer
)
from .serializers import Phasenoserializer,
ClassDivisionsserializer, Addclassserializer
from rest_framework.response import
Response

# from rest_framework import APIView
from rest_framework import status
from urllib.parse import urlparse
```

```
from urllib.parse import parse_qsl
import json
import random
from datetime import date


class
TeacherView(generics.ListCreateAPIView):
   queryset = Teacher.objects.all()
   serializer_class = Teacherserializer


class
SubjectView(generics.ListCreateAPIView):
   queryset = Subject.objects.all()
   serializer_class = Subjectserializer


class
DepartmentView(generics.ListCreateAPIView):
   queryset = Department.objects.all()
   serializer_class = Departmentserializer


class phaseview(generics.ListCreateAPIView):
   queryset = Phase.objects.all()
   serializer_class = Phaseserializer


class
DepartmentUpdation(generics.RetrieveUpdat
eDestroyAPIView):
   queryset = Department.objects.all()
   serializer_class = Departmentserializer


class
AdminLoginView(generics.CreateAPIView):
   serializer_class = AdminLoginserializer

   def post(self, request, *args, **kwargs):
      requestbody = dict(request.data)

      word = requestbody["password"]
      if word == "1234":
         return Response("SUCCESS",
status=status.HTTP_200_OK)
      else:
         return Response("NO",
status=status.HTTP_200_OK)


class Phaseview(generics.ListCreateAPIView):
   queryset = Phase.objects.all()
   serializer_class = Phaseserializer


class
Phaseupdate(generics.RetrieveUpdateDestroy
APIView):
   queryset = Phase.objects.all()
   serializer_class = Phaseserializer


class
TeacherLoginView(generics.CreateAPIView):
   serializer_class = TeacherLoginserializer

   def post(self, request, *args, **kwargs):
      requestbody = dict(request.data)
      title = requestbody["Tid"]
      word = requestbody["password"]
      obj = Teacher.objects.get(tid=title)
      a1 = obj.password
      a2 = obj.tid

      if word == a1:
         return Response("SUCCESS",
status=status.HTTP_200_OK)
      else:
         return Response("NO",
status=status.HTTP_200_OK)


class
TeacherDeletion(generics.RetrieveUpdateDest
royAPIView):
   serializer_class = Teacherserializer
   queryset = Teacher.objects.all()


class
TeacherUpdation(generics.RetrieveUpdateDes
troyAPIView):
   serializer_class = Teacherserializer
   queryset = Teacher.objects.all()
```

```
class
SubjectUpdation(generics.RetrieveUpdateDest
royAPIView):
    serializer_class = Subjectserializer
    queryset = Subject.objects.all()



class
ClassDivisionsview(generics.ListCreateAPIView
):
    queryset = ClassDivisions.objects.all()
    serializer_class = ClassDivisionsserializer


class split(generics.CreateAPIView):
    # queryset = Subject.objects.all()
    serializer_class = Subjectserializer

    def post(self, request, *args, **kwargs):
        requestbody = dict(request.data)
        print(requestbody)
        sid = requestbody["subid"]
        subname = requestbody["subname"]
        sem = requestbody["sem"]
        did = requestbody["depid"]
        subtype = requestbody["subtype"]
        user = Department.objects.get(depid=did)
        sub = Subject.objects.get(subid=sid)
        i = 0
        count = user.division

        name = user.depname

        while i < count:
            if subtype == "T":
                no = random.randint(10000, 99999)
                no = "C" + str(no)
                print(no)
                new_entry = ClassDivisions(
                    classid=no,
                    classname=name + "-" + chr(i +
ord("A")),
                    subject=sub,
                    depid=user,
                    classalloc=0,
                    exp=0,
                    sem=sem,
                    subtype=subtype,
                )
                new_entry.save()

            elif subtype == "L":
                k = 0
                while k < 3:
                    no = random.randint(10000,
99999)
                    no = "C" + str(no)
                    print(no)
                    new_entry = ClassDivisions(
                        classid=no,
                        classname=name + "-" + chr(i +
ord("A")),
                        subject=sub,
                        depid=user,
                        classalloc=0,
                        exp=0,
                        sem=sem,
                        subtype=subtype,
                    )
                    new_entry.save()
                    k = k + 1
            i = i + 1

        return HttpResponse("OK",
status=status.HTTP_200_OK)


class othersplitview(generics.CreateAPIView):
    serializer_class = otherSubjectserializer

    def post(self, request, *args, **kwargs):
        requestbody = dict(request.data)
        print(requestbody)
        sid = requestbody["subid"]
        subname = requestbody["subname"]
        sem = requestbody["sem"]
        did = requestbody["depid"]
        subtype = requestbody["subtype"]
        count=requestbody['count']
        user = Department.objects.get(depid=did)
        sub = Subject.objects.get(subid=sid)
        name = user.depname
```

```
        i=0

        while i<count:

            if subtype == "E" or "H" or "M":
                no = random.randint(10000,
99999)
                no = "C" + str(no)
                print(no)
                new_entry = ClassDivisions(
                    classid=no,
                    classname=name ,
                    subject=sub,
                    depid=user,
                    classalloc=0,
                    exp=0,
                    sem=sem,
                    subtype=subtype,
                )
                new_entry.save()



            i=i+1
        return HttpResponse("OK",
status=status.HTTP_200_OK)



class phase1view(generics.ListCreateAPIView):
    serializer_class = Phasenoserializer
    queryset = phaseno.objects.all()

    def post(self, request, *args, **kwargs):
        requestbody = dict(request.data)
        Phase.objects.all().delete()
        phaseno.objects.all().delete()
        title = requestbody["no"]
        a = requestbody["active"]

        new_start = phaseno(no=title, active=a)
        new_start.save()
        p = Teacher.objects.all()
        arrtid = []
        arrexp = []
        arrmail = []
        for i in p:
            arrtid.append(i.tid)
            current_year = date.today().year
            val = current_year -
int(i.year.strftime("%Y"))
            arrexp.append(val)
            arrmail.append(i.tmail)
        count = len(arrexp)
        for i in range(count - 1):
            for j in range(count - i - 1):
                if arrexp[j] < arrexp[j + 1]:
                    arrexp[j], arrexp[j + 1] = arrexp[j +
1], arrexp[j]
                    arrtid[j], arrtid[j + 1] = arrtid[j + 1],
arrtid[j]
                    arrmail[j], arrmail[j + 1] = arrmail[j
+ 1], arrmail[j]

        count = count // 3
        print(arrtid)
        for k in range(count):
            val = arrtid[k]
            no = random.randint(10000, 99999)
            no = "P" + str(no)
            obj = Teacher.objects.get(tid=val)
            new_entry = Phase(
                no=0,
                pid=no,
                tid=obj,
                alloc=0,
                status="ON",
                exp=arrexp[k],
                sub1="",
                sub2="",
                sub3="",
                sub4="",
                sub5="",
                sub6="",
                academicyear=title,
                mail=arrmail[k],
            )
            new_entry.save()
        return HttpResponse("OK",
status=status.HTTP_200_OK)


class phase2view(generics.ListCreateAPIView):
    serializer_class = Phasenoserializer
    queryset = phaseno.objects.all()
```

```
    def post(self, request, *args, **kwargs):
        requestbody = dict(request.data)
        phaseno.objects.all().delete()

        title = requestbody["no"]
        a = requestbody["active"]
        new_start = phaseno(no=title, active=a)
        new_start.save()
        p = Teacher.objects.all()
        arrtid = []
        arrexp = []
        arrmail = []
        for i in p:
            arrtid.append(i.tid)
            current_year = date.today().year
            val = current_year -
int(i.year.strftime("%Y"))
            arrexp.append(val)
            arrmail.append(i.tmail)
        count = len(arrexp)
        for i in range(count - 1):
            for j in range(count - i - 1):
                if arrexp[j] < arrexp[j + 1]:
                    arrexp[j], arrexp[j + 1] = arrexp[j +
1], arrexp[j]
                    arrtid[j], arrtid[j + 1] = arrtid[j + 1],
arrtid[j]
                    arrmail[j], arrmail[j + 1] = arrmail[j
+ 1], arrmail[j]


        count = len(p)-1
        k = len(p)//3
        Phase.objects.all().delete()
        print(k, count)
        print(arrtid)

        while k <= count:
            val = arrtid[k]
            no = random.randint(10000, 99999)
            no = "P" + str(no)
            obj = Teacher.objects.get(tid=val)
            new_entry = Phase(
                no=0,
                pid=no,
                tid=obj,
                alloc=0,
                status="ON",
                exp=arrexp[k],
                sub1="",
                sub2="",
                sub3="",
                sub4="",
                sub5="",
                sub6="",
                academicyear=title,
                mail=arrmail[k],
            )
            new_entry.save()
            k = k + 1
        return HttpResponse("OK",
status=status.HTTP_200_OK)


class
phasestatusview(generics.ListCreateAPIView):
    queryset = phaseno.objects.all()
    serializer_class = Phasenoserializer

#finalselection function

def final_selectfun():
        p=Phase.objects.all()
        clash.objects.all().delete()
        arrtid=[]
        for i in p:
            nosub=i.no

            t=i.tid

teacherobj=Teacher.objects.get(tid=t.tid)
            pos=teacherobj.pos
            #print(pos,t,nosub)

            flag=0
            if pos==2 or pos==1:
                if nosub<1:
                    flag=1
            elif pos==0:
                if nosub<2:
                    flag=1


            if(flag==1):
                arrtid.append(t.tid)
                new_entry =clash(clashid=t.tid)
```

```
            new_entry.save()


        print(arrtid)

        if len(arrtid)!=0:
            return HttpResponse(arrtid,
status=status.HTTP_200_OK)

        elif len(arrtid)==0:
            p1=Phase.objects.all()
            clash.objects.all().delete()
            pno=phaseno.objects.all()
            pval=pno[0].active
            print()


            for i in p1:
                nosub=i.no
                t=i.tid
                #print(t)

teacherobj=Teacher.objects.get(tid=t.tid)
                pos=teacherobj.pos
                if pos==2 or pos==1:
                    max=1
                elif pos==0:
                    max=2

                count=0
                check=0
                sub1=i.sub1
                sub2=i.sub2
                sub3=i.sub3
                sub4=i.sub4
                sub5=i.sub5
                sub6=i.sub6
                pno=phaseno.objects.all()
                year=pno[0].no
                arr=[]

                #sub1
                cd=ClassDivisions.objects.all()
                for j in cd:
                    cid=j.classid
                    if(cid==sub1 and sub1!=""):
                        calloc=j.classalloc
                        #print("in",calloc)
                        if int(calloc)==0:
                            #print("in calloc")
                            if(count<max):
                                #print("in count")
                                arr.append(sub1)
                                j.classalloc=1
                                j.save()
                                count=count+1
                                break
                if count>=max:
                    check=1


                #sub2
                cd=ClassDivisions.objects.all()
                for j in cd:
                    cid=j.classid
                    if(cid==sub2 and sub2!=""):
                        calloc=j.classalloc
                        if int(calloc) ==0:

                            if(count<max):
                                arr.append(sub2)
                                j.classalloc=1
                                j.save()
                                count=count+1
                                break
                if count>=max:
                    check=1

                #sub3
                cd=ClassDivisions.objects.all()
                for j in cd:
                    cid=j.classid
                    if(cid==sub3 and sub3!=""):
                        #print("inside")
                        calloc=j.classalloc
                        if int(calloc) ==0:
                            #print("in calloc3")
                            if(count<max):
                                #print("in count3")
                                arr.append(sub3)
                                j.classalloc=1
                                j.save()
                                count=count+1
                                break

                if count>=max:
```

```
            check=1

#sub4
cd=ClassDivisions.objects.all()
for j in cd:
    cid=j.classid
    if(cid==sub4 and sub4!=""):
        calloc=j.classalloc
        if int(calloc) ==0:

                if(count<max):
                    arr.append(sub4)
                    j.classalloc=1
                    j.save()
                    count=count+1
                    break

if count>=max:
    check=1

#sub5
cd=ClassDivisions.objects.all()
for j in cd:
    cid=j.classid
    if(cid==sub5 and sub5!=""):
        calloc=j.classalloc
        if int(calloc) ==0:

                if(count<max):
                    arr.append(sub5)
                    j.classalloc=1
                    j.save()
                    count=count+1
                    break
                if count>max:
                    check=1
                    break

if count>=max:
    check=1

#sub6
cd=ClassDivisions.objects.all()
for j in cd:
    cid=j.classid
    if(cid==sub6 and sub6!=""):
        calloc=j.classalloc
        if int(calloc) ==0:

                    if(count<max):
                        arr.append(sub6)
                        j.classalloc=1
                        j.save()
                        count=count+1
                        break
                    if count>max:
                        check=1
                        break
            if count>=max:
                check=1


            if len(arr)==1:
                arr.insert(1," ")
                nosub=1
            elif len(arr)==2:
                nosub=2
            elif len(arr)==0:
                print(t.tid)
            #teacherselection

            if len(arr)!=0 and check==1:
                no = random.randint(10000,
99999)
                print(t.tid)
                no = "S" + str(no)
                #print(arr)

                new_val=TeacherSelection(tid=t,
                sub1=arr[0],
                sub2=arr[1],
                count=nosub,
                selectionid=no,
                year=year)

Phase.objects.filter(pid=i.pid).delete()
                new_val.save()
                new_entry =clash(clashid='OK')
                new_entry.save()

            elif check==0:
                i.no=0
                i.status='UPDATE'
                i.sub1=""
                i.sub2=""
                i.sub3=""
```

```
                i.sub4=""
                i.sub5=""
                i.sub6=""
                i.save()
                new_entry =clash(clashid=t.tid)
                new_entry.save()
                new_entry
=resolve.objects.get(rid=0)
                new_entry.rid=1
                new_entry.save()
                #print('stop')
                print(i.mail)
                break
                return HttpResponse("ok",
status=status.HTTP_200_OK)

        if pval=='phase2' and
Phase.objects.count()==0:
            cd=ClassDivisions.objects.all()
            #print("val")
            for g in cd:
                if int(g.classalloc)==1:
                    #print(g.tid)
                    g.classalloc=0
                    g.save()
            return HttpResponse("ok",
status=status.HTTP_200_OK)

        return HttpResponse("ok",
status=status.HTTP_200_OK)




class subselect(generics.ListCreateAPIView):
    queryset = Phase.objects.all()
    serializer_class = Addclassserializer

    def post(self, request, *args, **kwargs):
        requestbody = dict(request.data)
        tid = requestbody["tid"]
        count = requestbody["no"]
        sub1 = requestbody["sub1"]
        sub2 = requestbody["sub2"]
        sub3 = requestbody["sub3"]
        sub4 = requestbody["sub4"]
        sub5 = requestbody["sub5"]
        sub6 = requestbody["sub6"]
        teacherid = Teacher.objects.get(tid=tid)
        current_year = date.today().year
        exp = current_year -
int(teacherid.year.strftime("%Y"))
        subarray=[]
        p = Phase.objects.all()

        for i in p:

            if   i.sub1!='':
                d={}
                d['sub1']=i.sub1
                subarray.append(d)

            if i.sub2!='':
                d={}
                d['sub2']=i.sub2
                subarray.append(d)
                print(i.pid)

            if i.sub3!='':
                d={}
                d['sub3']=i.sub3
                subarray.append(d)

            if i.sub4!='':
                d={}
                d['sub4']=i.sub4
                subarray.append(d)

            if i.sub5!='':
                d={}
                d['sub5']=i.sub5
                subarray.append(d)

            if i.sub6!='':
                d={}
                d['sub6']=i.sub6
                subarray.append(d)

        #print(subarray)

        def fun_order(sub,i):

                    if(i=='sub1'):
```

```
                p =
Phase.objects.get(sub1=sub)
            elif(i=='sub2'):
                p =
Phase.objects.get(sub2=sub)
            elif(i=='sub3'):
                p =
Phase.objects.get(sub3=sub)
            elif(i=='sub4'):
                p =
Phase.objects.get(sub4=sub)
            elif(i=='sub5'):
                p =
Phase.objects.get(sub5=sub)
            elif(i=='sub6'):
                p =
Phase.objects.get(sub6=sub)

            if p.exp < exp :
                if(i=='sub1'):
                    upsub=p.sub1
                elif(i=='sub2'):
                    upsub=p.sub2
                elif(i=='sub3'):
                    upsub=p.sub3
                elif(i=='sub4'):
                    upsub=p.sub4
                elif(i=='sub5'):
                    upsub=p.sub5
                elif(i=='sub6'):
                    upsub=p.sub6
                print(p.mail)
                p.status='UPDATE'
                cd =
ClassDivisions.objects.get(classid=upsub)
                #
subs=Subject.objects.get(subid=cd.subject)
                # print(subs.subname)
                subs=cd.subject
                print(subs.subname)
                #p.no=p.no-1
                classid =
ClassDivisions.objects.get(classid=sub)
                classid.exp = exp
                classid.save()
                p.save()

    for k in subarray:
        for i,j in k.items():
            if sub1==j :
                fun_order(sub1,i)

            if sub2==j :
                fun_order(sub2,i)

            if sub3==j :
                fun_order(sub3,i)

            if sub4==j :
                fun_order(sub4,i)

            if sub5==j :
                fun_order(sub5,i)

            if sub6==j :
                fun_order(sub6,i)


    i = 1
    while i <= count:
        sub = requestbody["sub" + str(i)]
        if sub!='':
            classid =
ClassDivisions.objects.get(classid=sub)
            classid.exp = exp
            classid.classalloc = 0
            classid.save()
        i = i + 1

    phaseid = Phase.objects.get(tid=tid)
    phaseid.no = count
    phaseid.sub1 = sub1
    phaseid.sub2 = sub2
    phaseid.sub3 = sub3
    phaseid.sub4 = sub4
    phaseid.sub5 = sub5
    phaseid.sub6 = sub6
    phasenoid = phaseno.objects.all()
    val = phasenoid[0].no
    phaseid.academicyear = val

    phaseid.mail = teacherid.tmail
    phaseid.status = "OFF"
    phaseid.save()
    try:
        new_entry =resolve.objects.get(rid=1)
```

```
            if new_entry.rid==1:
                new_entry.rid=0
                new_entry.save()
                final_selectfun()
        except:
            print("ok")


        return HttpResponse("OK",
status=status.HTTP_200_OK)


class Finalview(generics.CreateAPIView):
    serializer_class =Finalserializer

    def post(self, request, *args, **kwargs):
        requestbody = dict(request.data)
        return final_selectfun()



class
TeacherSelectionview(generics.ListCreateAPIVi
ew):
    queryset = TeacherSelection.objects.all()
    serializer_class = TeacherSelectionserializer



class clashview(generics.ListCreateAPIView):
    queryset = clash.objects.all()
    serializer_class = clashserializer


class
semtypeview(generics.ListCreateAPIView):
    queryset = semtype.objects.all()
    serializer_class = Semtypeserializer
    def post(self, request, *args, **kwargs):
        requestbody = dict(request.data)
        semtype.objects.all().delete()
        sem = requestbody["sem"]
        new_val=semtype(sem=sem)
        new_val.save()
        return HttpResponse("ok",
status=status.HTTP_200_OK)


class phasegetview(generics.CreateAPIView):
    serializer_class = phasegetserializer

    def post(self, request, *args, **kwargs):
        requestbody = dict(request.data)
        Phaseteacher.objects.all().delete()
        p = Phase.objects.all()

        for i in p:
            t=i.tid

teacherobj=Teacher.objects.get(tid=t.tid)
            s1=i.sub1
            s2=i.sub2
            s3=i.sub3
            s4=i.sub4
            s5=i.sub5
            s6=i.sub6
            teachername=t.tname
            arr=[]
            k=0
            while k<12:
                arr.insert(k," ")
                k=k+1
            #print(teachername)
            k=0
            cd=ClassDivisions.objects.all()

            for j in cd:
                cid=j.classid
                check=0

                if(cid==s1):
                    subid=j.subject
                    classname=j.classname
                    check=1
                elif(cid==s2):
                    subid=j.subject
                    classname=j.classname
                    check=1
                elif(cid==s3):
                    subid=j.subject
                    classname=j.classname
                    check=1
                elif(cid==s4):
                    subid=j.subject
                    classname=j.classname
```

```
                check=1
            elif(cid==s5):
                subid=j.subject
                classname=j.classname
                check=1
            elif(cid==s6):
                subid=j.subject
                classname=j.classname
                check=1
            if check==1:
                arr.insert(k,subid.subname)
                arr.insert(k+1,classname)
                print(subid.subname)
                print(classname)
                k=k+2

new_val=Phaseteacher(tname=teachername,
        sub1=arr[0],
        classname1=arr[1],
        sub2=arr[2],
        classname2=arr[3],
        sub3=arr[4],
        classname3=arr[5],
        sub4=arr[6],
        classname4=arr[7],
        sub5=arr[8],
        classname5=arr[9],
        sub6=arr[10],
        classname6=arr[11],
        )
        new_val.save()

    return HttpResponse("ok",
status=status.HTTP_200_OK)



class
Phaseteacherview(generics.ListCreateAPIView
):
  queryset = Phaseteacher.objects.all()
  serializer_class = phaseteacherserializer
```

Appendix C: CO-PSO Mapping

COURSE OUTCOMES:

After completion of the course the student will be able to

SL. NO	DESCRIPTION	Blooms' Taxonomy Level
CO1	Identify technically and economically feasible problems (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO2	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO3	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions of minimal complexity by using modern tools & advanced programming techniques (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO4	Prepare technical report and deliver presentation (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO5	Apply engineering and management principles to achieve the goal of the project (Cognitive Knowledge Level: Apply)	Level 3: Apply

CO-PO AND CO-PSO MAPPING

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO3
CO1	3	3	3	3		2	2	3	2	2	2	3	2	2	2
CO2	3	3	3	3	3	2		3	2	3	2	3	2	2	2
CO3	3	3	3	3	3	2	2	3	2	2	2	3			2
CO4	2	3	2	2	2			3	3	3	2	3	2	2	2
CO5	3	3	3	2	2	2	2	3	2		2	3	2	2	2

3/2/1: high/medium/low

JUSTIFICATIONS FOR CO-PO MAPPING

MAPPING	LOW/ MEDIUM/ HIGH	JUSTIFICATION
100003/CS622T.1-PO1	**HIGH**	Identify technically and economically feasible problems by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.1-PO2	**HIGH**	Identify technically and economically feasible problems by analysing complex engineering problems reaching substantiated conclusions using first principles of mathematics.
100003/CS622T.1-PO3	**HIGH**	Design solutions for complex engineering problems by identifying technically and economically feasible problems.
100003/CS622T.1-PO4	**HIGH**	Identify technically and economically feasible problems by analysis and interpretation of data.
100003/CS622T.1-PO6	**MEDIUM**	Responsibilities relevant to the professional engineering practice by identifying the problem.
100003/CS622T.1-PO7	**MEDIUM**	Identify technically and economically feasible problems by understanding the impact of the professional engineering solutions.
100003/CS622T.1-PO8	**HIGH**	Apply ethical principles and commit to professional ethics to identify technically and economically feasible problems.
100003/CS622T.1-PO9	**MEDIUM**	Identify technically and economically feasible problems by working as a team.
100003/CS622T.1-PO10	**MEDIUM**	Communicate effectively with the engineering community by identifying technically and economically feasible problems.
100003/CS622T.1-P011	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles by selecting the technically and economically feasible problems.
100003/CS622T.1-PO12	**HIGH**	Identify technically and economically feasible problems for long term learning.
100003/CS622T.1-PSO1	**MEDIUM**	Ability to identify, analyze and design solutions to identify technically and economically feasible problems.
100003/CS622T.1-PSO2	**MEDIUM**	By designing algorithms and applying standard practices in software project development and Identifying technically and economically feasible problems.
100003/CS622T.1-PSO3	**MEDIUM**	Fundamentals of computer science in competitive research can be applied to Identify technically and economically feasible problems.
100003/CS622T.2-PO1	**HIGH**	Identify and survey the relevant by applying the knowledge of mathematics, science, engineering fundamentals.

100003/CS622T.2-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems get familiarized with software development processes.
100003/CS622T.2-PO3	**HIGH**	Design solutions for complex engineering problems and design based on the relevant literature.
100003/CS622T.2-PO4	**HIGH**	Use research-based knowledge including design of experiments based on relevant literature.
100003/CS622T.2-PO5	**HIGH**	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes by using modern tools.
100003/CS622T.2-PO6	**MEDIUM**	Create, select, and apply appropriate techniques, resources, by identifying and surveying the relevant literature.
100003/CS622T.2-PO8	**HIGH**	Apply ethical principles and commit to professional ethics based on the relevant literature.
100003/CS622T.2-PO9	**MEDIUM**	Identify and survey the relevant literature as a team.
100003/CS622T.2-PO10	**HIGH**	Identify and survey the relevant literature for a good communication to the engineering fraternity.
100003/CS622T.2-PO11	**MEDIUM**	Identify and survey the relevant literature to demonstrate knowledge and understanding of engineering and management principles.
100003/CS622T.2-PO12	**HIGH**	Identify and survey the relevant literature for independent and lifelong learning.
100003/CS622T.2-PSO1	**MEDIUM**	Design solutions for complex engineering problems by Identifying and survey the relevant literature.
100003/CS622T.2-PSO2	**MEDIUM**	Identify and survey the relevant literature for acquiring programming efficiency by designing algorithms and applying standard practices.
100003/CS622T.2-PSO3	**MEDIUM**	Identify and survey the relevant literature to apply the fundamentals of computer science in competitive research.
100003/CS622T.3-PO1	**HIGH**	Perform requirement analysis, identify design methodologies by using modern tools & advanced programming techniques and by applying the knowledge of mathematics, science, engineering fundamentals.
100003/CS622T.3-PO2	**HIGH**	Identify, formulate, review research literature for requirement analysis, identify design methodologies and develop adaptable & reusable solutions.

100003/CS622T.3-PO3	**HIGH**	Design solutions for complex engineering problems and perform requirement analysis, identify design methodologies.
100003/CS622T.3-PO4	**HIGH**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS622T.3-PO5	**HIGH**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools.
100003/CS622T.3-PO6	**MEDIUM**	Perform requirement analysis, identify design methodologies and assess societal, health, safety, legal, and cultural issues.
100003/CS622T.3-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts and Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS622T.3-PO8	**HIGH**	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions by applying ethical principles and commit to professional ethics.
100003/CS622T.3-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS622T.3-PO10	**MEDIUM**	Communicate effectively with the engineering community and with society at large to perform requirement analysis, identify design methodologies.
100003/CS622T.3-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering requirement analysis by identifying design methodologies.
100003/CS622T.3-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS622T.3-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and prior to that perform requirement analysis, identify design methodologies.
100003/CS622T.4-PO1	**MEDIUM**	Prepare technical report and deliver presentation by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.4-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by preparing technical report and deliver presentation.

100003/CS622T.4-PO3	**MEDIUM**	Prepare Design solutions for complex engineering problems and create technical report and deliver presentation.
100003/CS622T.4-PO4	**MEDIUM**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions and prepare technical report and deliver presentation.
100003/CS622T.4-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and Prepare technical report and deliver presentation.
100003/CS622T.4-PO8	**HIGH**	Prepare technical report and deliver presentation by applying ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
100003/CS622T.4-PO9	**HIGH**	Prepare technical report and deliver presentation effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS622T.4-PO10	**HIGH**	Communicate effectively with the engineering community and with society at large by prepare technical report and deliver presentation.
100003/CS622T.4-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work by prepare technical report and deliver presentation.
100003/CS622T.4-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by prepare technical report and deliver presentation.
100003/CS622T.4-PSO1	**MEDIUM**	Prepare a technical report and deliver presentation to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas.
100003/CS622T.4-PSO2	**MEDIUM**	To acquire programming efficiency by designing algorithms and applying standard practices in software project development and to prepare technical report and deliver presentation.
100003/CS622T.4-PSO3	**MEDIUM**	To apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs by preparing technical report and deliver presentation.
100003/CS622T.5-PO1	**HIGH**	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.5-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by applying engineering and management principles to achieve the goal of the project.

100003/CS622T.5-PO3	**HIGH**	Apply engineering and management principles to achieve the goal of the project and to design solutions for complex engineering problems and design system components or processes that meet the specified needs.
100003/CS622T.5-PO4	**MEDIUM**	Apply engineering and management principles to achieve the goal of the project and use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS622T.5-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO6	**MEDIUM**	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities by applying engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts, and apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO8	**HIGH**	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice and to use the engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PSO1	**MEDIUM**	The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas. Apply engineering and management principles to achieve the goal of the project.

100003/CS622T.5-PSO2	**MEDIUM**	The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur and apply engineering and management principles to achieve the goal of the project.