Andromeda

College Registration System

System Manual

For

Web Based Registration System

SYSTEM DESIGN AND IMPLEMENTATION

CS 5510

SPRING 2018

Professor Naresh Gupta

James McCarthy	Tommy Pederson	Jose Martinez
Email: jmccar16@oldwestbury.edu	Email: tpederso@oldwestbury.edu	Email: nytgen@gmail.com

Table of Contents

Mission Statement

The document's purpose is to give our client a detailed overview of the project proposal for the Andromeda Web-based Registration System. This document will elaborate on the purpose of the system, as well as expand on the system's functional specifications which include, what our system will do, who will use it, how the user will interact with the system, and the task breakdowns and implementation strategies. This document will be used to get approval on the web-based registration system. It acts as a liaison between the client and developers of this project to maintain a well-informed record of the projects vision and direction.

Project Proposal Scope

The Andromeda Web-based Registration System will be used for registering students to the Andromeda University. The system will allow students to search and register for courses in the following school semester. This system will be easy to use and will alleviate any additional workload staff at the University would have encountered previously.

As noted, the web-based registration system will be used for all registered students at the University to create a school schedules. Students will be able to use multiple search filters to narrow down their search. These filters include refining by subject, instructor, and course number. The student will also be allowed to add or drop courses to modify courses their existing schedule.

This system will contain a hierarchy of privileges to perform functions which need higher level privileges (administrative, faculty functions) and lower level privileges (student, researcher). Administrator(s) hold all privileges, allowing them to manage every aspect of the system. Tasks include removing or adding access to faculty and maintaining the database integrity.

The University faculty will be able to add/delete subjects to the registration system, as well as course numbers and other instructors. Changing the dates and times of the courses will also be a function that faculty will be given to make adjustments as they deem necessary. Like students, they will also have the authorization to add or remove students from the courses they instruct

The last group to use this system will be researchers. They have limited access to the system and will not have the same privileges as students, faculty, or administrators. Their role will be to perform statistical analysis on data retrieved from the database of the registration system. This will help the University assess the number of students currently enrolled in each of the archived semesters as well give them a better understanding of what classes are the most popular. This significantly helps the school, giving them a better understanding of where to allocate their resources for future semesters.

Procedure

Our combined knowledge of Computer Science and Management Information Systems, we will fully implement a produce a full stack web-based application that satisfies the client requirement and standards. We will use the following tools to complete such a task:

- **HTML -** Hypertext Markup Language is the standard markup language for creating web pages and web applications.
- **CSS –** Cascading Style Sheets is a stylesheet language used for describing the presentation of a document written in a markup language like HTML.
- **JavaScript –** High level programming language used to control web page interactions and populate data sent from the back-end (Python Django Framework) into front-end HTML components.
- **Node.JS –** Open-source cross-platform environment that executed JavaScript code on the server-side, Node.js is used in our application to create a Render Server to render all React code produced on the server side.
- **React –** A declarative, efficient, and flexible JavaScript library for building user interfaces.
- **Python –** An interpreted high-level language for general purpose programming.
- **Django –** A free and open-source web-framework which abides by the MVC architecture, popular In modern back-end applications.
- **SQL** – Query language used to run queries on the database.
- **PostgreSQL** – Open-source Database management system used to manage our database.
- **AJAX** – Asynchronous JavaScript and XML is a set of web development techniques using web technologies on the client side to create asynchronous applications.
- **JSON –** An open-standard file format that uses human-readable text to transmit data objects consists of attribute-value pairs and array data types.

Feasibility

Based on our experience working with various programming languages, web frameworks, software design patterns, and database management systems. We believe that we can fully implement the system and complete the project in a timely manner while abiding by all business requirements. To avoid falling behind we use a weekly schedule to stay on track with our goals.

System Versions

Bare Bone

A fully functional system that fully implements all business rules and constraints, however very little aesthetic styling.

Standard System

Styling is implemented to give a modern look & feel to the system. All functions are fully operational.

Deluxe System

Fully implement a messaging system (using email notifications) to allow students and advisers to contact each other through the system.

Conclusion

Andromeda Registration system will help any university provide its user's a fully-functional system that provides academic information in a scalable, durable, and consistent manner. Our registration system will satisfy all business requirements and constraints, as well as provide an easily modifiable design pattern for any patches or updates in the future.

Business Rules

- **System Requirements**

  Explicitly Required

  a. At the end of the semester system will deregister students who don't fulfill prerequisite requirements for enrolled courses.

- **Non-Registered and Registered Users**

  Explicitly Required

  a. Users can search the master schedule.

  b. Searches can be filtered by department name, professor name, course name, day/period, course section, and term/year.

- **Student/Faculty/Admin/Researcher Users**

  Explicitly Required

  a. All users must be registered to access their accounts.

  b. All users must be logged into access their account privileges.

  c. All users must register with their email.

Explicitly Forbidden

a. Users from one group cannot access the functions of another (i.e. Students are forbidden from using admin functions).

- **Student**

Explicitly Required

a. There are two types of students: Part-time and fulltime.

b. Fulltime students are required to take a minimum of 3 courses and a maximum of 5 courses.

a. Part-time students can take no more than 2 courses.

b. System will provide features for students to register for their courses.

c. Students are required to take prerequisites.

d. A course may require many prerequisites.

e. A prerequisite may be required for many courses.

f. Students can view their grades and classes for many past semesters.

g. Students should be able to run graduation audits.

h. Students should be able to request their official transcripts.

i. Students may view any holds placed on their account.

j. Students should be able to add/drop/update their course schedule.

k. A Student has many advisors.

l. Students can contact their advisors through the system.

m. Students can request to declare a major or minor through the system.

n. Students may request to declare no more than 2 majors.

o. Students may request to declare no more than 2 minors.

p. Students can only claim 3 areas of study.

Explicitly Forbidden

a. The system forbids the student from registering for courses if there is a hold placed on their account.
b. Part-time/Full-time students must register for the correct number of courses.
c. Students are not allowed to register for two courses scheduled at the same.
d. Students are forbidden to add or remove courses outside the registration period.
e. Students are forbidden to register for courses if they lack the sufficient perquisites.

- **Faculty**

Explicitly Required

a. There are two types of faculty: Part-time and Fulltime.
b. Ability to view all student schedules.
c. Part-time faculty can teach a maximum of 2 courses.
d. Fulltime faculty can teach 0-5 courses.
e. Faculty may view all courses taught in the current semester.
f. Faculty can view all student's courses and transcripts.
g. Faculty can assign grades to students within an allotted period.
h. A student can receive only one grade per course.

Explicitly Forbidden

a. Faculty is forbidden form altering grades outside of the prescribed time period.
b. Faculty cannot teach two courses that meet at the same time.

- **Admin**

Administrators share requirements with both faculty and students. For this reason, these requirements will not be reiterated.

a. The requirements of faculty and student can be performed on the system as if administrator were any singular student or faculty member.

Explicitly Required

a. Admin can set permissions, remove users, and register new accounts.
b. Admin can add/update financial holds.
c. Admin may add courses to the master schedule to make new courses available.
d. Admin must provide necessary information to create or update a course.
e. Admin may remove existing courses from the master schedule.

Explicitly Forbidden

a. Admins cannot change grades already entered into the system.
b. Admins cannot create two courses that meet at the same room at the same time.

- **Researcher**

Explicitly Required

a. Researchers have access to all data produced by the system.
b. Researches can view student grading data, enrollment data, graduation data, etc.
c. Researchers perform data visualization and analysis.

d. The goals of data visualization and analysis is to define conclusions about patterns within the data of the system.
e. All retrieval from data sources are done with minimal to no processing or additional organization.
f. Researchers can choose which data source to pull information from and what parameters to filter with.
g. Researchers may view the analytics from the site or download a file to their machine.
h. Researchers must select boundary conditions such as timeframe or department.

Entity – Attributes

User

- User_ID
- Email
- Password
- First Name
- Last Name
- User Type

Student

- Student ID
- Type
- Date Of Birth

Full Time Students

- Student ID
- Credits

Part Time Students

- Student ID
- Credits

Student Schedule History

- Student ID
- Enrollment ID
- Status

Student Major

- Major ID
- Student ID

Student Minor

- Minor ID
- Student ID

Major

- Major_ID
- Major Name
- Department ID

Minor

- Minor_ID
- Minor Name
- Deparment ID

Major Requirements

- Major_ID
- Course ID

Minor Requirements

- Minor ID
- Course ID

Enrollment

- Section ID
- Enrollment ID
- Student ID
- Grade

Semester

- Semester_ID
- Semester Year
- Semester Quarter

Advising

- Faculty ID
- Student ID

Faculty

- Faculty_ID
- Department ID
- Faculty Type

Full Time Faculty

- Faculty_ID

Part Time Faculty

- Faculty_ID

Admin

- Admin_ID

Researcher

- Researcher_ID

Courses

- Course ID
- Department ID
- Name
- Description
- Credits

Section

- Section ID
- Course ID
- Faculty ID
- Time Slot ID
- Seats Taken
- Seat Capacity
- Building
- Room Number

Attendance

- Section Id
- Student Id
- Date
- Present/Absent

Prerequisite

- Course ID (course that requires prerequisite)
- Course Requirement ID

Department

- Department ID
- Name
- Chair of the Department
- Building
- Room Number
- Phone Number

Timeslot

- Timeslot ID
- Day ID
- Period ID
- Semester ID

Day

- Day ID
- Meeting Days

Period

- Period ID
- Start Time
- End Time

Semester/Year

- Semester ID
- Season
- Year

Room

- Room_Number
- Building ID
- Room Type
- Room Capacity

Building

- Building ID
- Building Name
- Building Location

Report

- Report_ID
- Researcher ID
- Report Date
- Description

Holds

- Hold ID
- Name

Relationships

Part Time Faculty Teaches 0-2 Sections

- Faculty ID
- Course ID

Full Time Faculty Teaches 0-5 Sections

- Faculty ID
- Course ID

Student Has Many Advisors

- Faculty ID
- Student ID

An Advisor Has Many Students

- Faculty ID
- Student ID

A Full Time Student is enrolled in 3-5 courses

- Student ID
- Course ID

A Part Time Student is enrolled in 1-2 courses

- Student ID
- Course ID

A Student Has 0-2 Majors

- Student ID
- Major ID

A Major Has Many Students

- Student ID
- Major ID

A Student Has 0-2 Minors

- Student ID
- Minor ID

A Minor Has Many Students

- Student ID
- Minor ID

A Course Has Many Students

- Student ID
- Course ID
- Letter Grade

A Course Has Many Sections

- Course ID
- Section ID

A Course Has 0 or More Prerequisites

- Course Required By ID
- Required Course ID

A Prerequisite Has 1 or More Courses

- Course Required By ID
- Required Course ID

Constraints

Student

- PK - Student ID
- FK – Major ID
- FK – Minor ID

Full Time Student

- PK - Student ID
- FK – Major ID
- FK – Minor ID

Part Time Student

- PK - Student ID
- FK – Major ID
- FK – Minor ID

Major

- PK – Major ID
- FK – Department ID

Minor

- PK – Minor ID
- FK – Department ID

Department

- PK – Department ID

Section

- PK – Section ID
- FK – Course ID

Course

- PK – Course ID
- FK – Department ID

Enrollment

- PK – Student ID
- FK – Course ID

Faculty

- PK – Faculty ID
- FK – Department ID

Full Time Faculty

- PK – Faculty ID
- FK – Department ID

Part Time Faculty

- PK – Faculty ID
- FK – Department ID

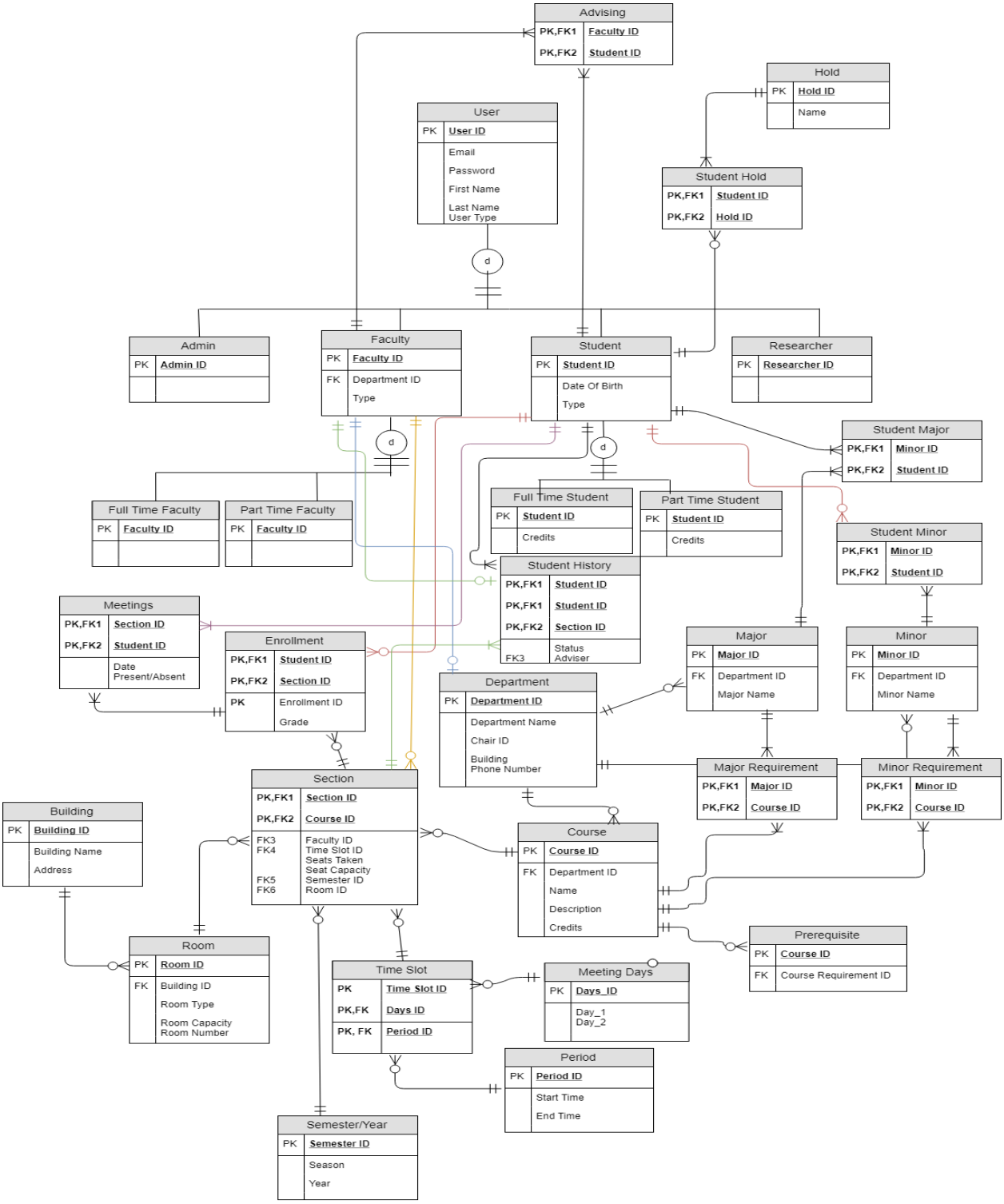
Admin

- PK – Admin ID

Researcher

- PK – Researcher ID

ERD


Advising
PK,FK1 Faculty ID
PK,FK2 Student ID
Hold
PK Hold ID
Name
User
PK User ID
Email
Password
First Name
Last Name
User Type
Student Hold
PK,FK1 Student ID
PK,FK2 Hold ID
d
Admin
PK Admin ID
Faculty
PK Faculty ID
FK Department ID
Type
Student
PK Student ID
Date Of Birth
Type
Researcher
PK Researcher ID
Student Major
PK,FK1 Minor ID
PK,FK2 Student ID
Full Time Faculty
PK Faculty ID
Part Time Faculty
PK Faculty ID
Full Time Student
PK Student ID
Credits
Part Time Student
PK Student ID
Credits
Student Minor
PK,FK1 Minor ID
PK,FK2 Student ID
Student History
PK,FK1 Student ID
PK,FK1 Student ID
PK,FK2 Section ID
FK3 Status
Adviser
Meetings
PK,FK1 Section ID
PK,FK2 Student ID
Date
Present/Absent
Enrollment
PK,FK1 Student ID
PK,FK2 Section ID
PK Enrollment ID
Grade
Major
PK Major ID
FK Department ID
Major Name
Minor
PK Minor ID
FK Department ID
Minor Name
Department
PK Department ID
Department Name
Chair ID
Building
Phone Number
Major Requirement
PK,FK1 Major ID
PK,FK2 Course ID
Minor Requirement
PK,FK1 Minor ID
PK,FK2 Course ID
Section
PK,FK1 Section ID
PK,FK2 Course ID
FK3 Faculty ID
FK4 Time Slot ID
Seats Taken
Seat Capacity
FK5 Semester ID
FK6 Room ID
Building
PK Building ID
Building Name
Address
Course
PK Course ID
FK Department ID
Name
Description
Credits
Prerequisite
PK Course ID
FK Course Requirement ID
Room
PK Room ID
FK Building ID
Room Type
Room Capacity
Room Number
Time Slot
PK Time Slot ID
PK,FK Days ID
PK, FK Period ID
Meeting Days
PK Days_ID
Day_1
Day_2
Period
PK Period ID
Start Time
End Time
Semester/Year
PK Semester ID
Season
Year

Performance Goals

There are performance goals which will be tested to ensure that not only the system is reliable, but that it also rectifies flaws in prior web-based registration systems. Because load times are a major concern in building full stack web applications, a major goal will be caching to significantly reduce server requests for rendering content such as images (.png, .svg, etc), media, user information, etc. A major goal is to avoid cookie storage for authentication/authorization and use session storage (stored on server instead of client computer) to avoid security concerns. Security is one of the major concern, users should be able to feel their credentials are kept safe in our system. One way that this will be achieved is using a single sign-in prompt as well as a multi-factor authentication system (able to login/register with gmail, facebook accounts) that will enable users to seamlessly go in and out of the web-based registration system. Authentication will use OAuth which is a standard for token-based authentication and authorization for the internet.

Schedule / Time Slots

Date	Task	Start Time	End Time	Members Present
1/29/2018	Project Proposal	6:00pm	9:00pm	James, Jose, Tommy
2/5/2018	Updating Project Proposal	6:00pm	9:00pm	James, Jose, Tommy
2/9/2018	Updating Business Rules and Use Cases	7:00pm	9:15pm	James, Jose, Tommy
2/13/2018	Updated Business Rules and added entities, constraints, and relationships.	7:00pm	9:15pm	James, Jose, Tommy
2/25/2018	Prepared project implementation.	7:00pm	9:15pm	James, Jose, Tommy
3/01/2018	Setup Environments.	7:00pm	9:15pm	James, Jose, Tommy
3/07/2018	Admin implementation	7:00pm	9:15pm	James, Jose, Tommy
3/15/2018	Admin implementation	7:00pm	9:15pm	James, Jose, Tommy
3/28/2018	Faculty Implementation	7:00pm	9:15pm	James, Jose, Tommy
4/08/2018	Student Implementation	7:00pm	9:15pm	James, Jose, Tommy
4/19/2018	Student Implementation	7:00pm	9:15pm	James, Jose, Tommy
5/1/2018	Researcher Implementation /Wrap-up	7:00pm	9:15pm	James, Jose, Tommy

Templating With Django Framework

To simply replicating boilerplate html code, The Django framework is built on top of the Python programming language. This provides server-side rendering of html code and the use of a templating engine. With this functionality we loop over data structures and use conditional statements on the front-end HTML. These snippets can be found through-out the html files in the implementation section of the system manual.

An example of such : {% if 'condition' %} {% end if %} {% for r in records %} {% endfor %}

Routing With Django

All request sent to the back-end are routed to the appropriate function by what path is sent in the request. This provides a way to organize all routing request to be served by the appropriate endpoints. An example of routing can be seen in a snippet taken from our urls.py file.

```
from django.conf.urls import url, include
from registration_system import views as core_views

urlpatterns = [
    url(r'^$', core_views.home, name='home'),
    url(r'^user_display/$', core_views.UserDisplay.as_view(), name='display'),
    url(r'^logout/$', core_views.logout_view, name='registration_system_logout'),
    url(r'^declare_major/$', core_views.DeclareMajor.as_view(), name='declare_major'),
    url(r'^declare_minor/$', core_views.DeclareMinor.as_view(), name='declare_minor'),
    url(r'^student/view_student_schedule/$', core_views.StudentViewSchedule.as_view(),
        name='student_view_student_schedule'),
    url(r'^view_student_schedule/$', core_views.ViewStudentSchedule.as_view(), name='view_student_schedule'),
    url(r'^view_faculty_schedule/$', core_views.ViewFacultySchedule.as_view(), name='view_faculty_schedule'),
    url(r'^student/view_student_transcript/$', core_views.StudentViewStudentTranscript.as_view(),
        name='student_view_student_transcript'),
    url(r'^view_student_transcript/$', core_views.ViewStudentTranscript.as_view(), name='view_student_transcript'),
    url(r'^view_student_transcript/(?P<student_id>\d+)/', core_views.ViewStudentTranscriptResult.as_view(),
        name='view_student_transcript_result'),
    url(r'^create_course/$', core_views.CreateCourse.as_view(), name='create_course'),
    url(r'^create_course/prerequisites/(?P<course_id>\d+)/', core_views.CreatePrerequisite.as_view()
        , name='create_prerequisites'),
```

Public Features

Search Master Schedule:

- Based on the options selected from the dropdown menu's, a standard query is built to retrieve sections that match. All inputs are dropdown menus except for one text input to input course names (remember a section is an instance of a course).

Front-end Implementation

```
import React from 'react';
import { Container, Divider, Header, Table, Form, Segment, Dimmer, Loader } from 'semantic-ui-react';
import MasterTable from './table';


class App extends React.Component {

    constructor(props){
        super(props);
        this.state = {
            isLoading: true,
        };
        this.onBlur = this.onBlur.bind(this);
        this.handleInputChange = this.handleInputChange.bind(this);
        this.handleSelectChange = this.handleSelectChange.bind(this);
    }

    componentDidMount(){
        fetch("/student_system/student_system_api/get_general_data/")
            .then((response) => {
                if(!response.ok){
                    throw Error(response.statusText);
                }

                return response;
            })
            .then((response) => response.json())
            .then((data) => {
                console.log(data);
                let department_options = [];
                let time_period_options = [];
                let meeting_day_options = [];
                let semester_options = [];
                let building_options = [];
                let room_options = [];
                let faculty_options = [];
                let default_option = { key: 0, text: '------', value: true};
                semester_options.push(default_option);
                department_options.push(default_option);
                time_period_options.push(default_option);
                meeting_day_options.push(default_option);
                building_options.push(default_option);
                room_options.push(default_option);
                faculty_options.push(default_option);
                let d, f, b, t, m, r, s;
                for( let s of data.semesters){
                    let semester_data = {
                        key: s.semester_id,
                        text: s.semester_season + '-' + s.semester_year + "---"+ s.semester_status,
                        value: s.semester_id
                    };
                    semester_options.push(semester_data);
```

```
            semester_options.push(semester_data);
        }
        for( let d of data.departments){
            let department_data = {
                key:  d.department_id,
                text: d.department_name,
                value: d.department_id,
            };
            department_options.push(department_data);
        }
        for( let t of data.time_periods){
            let time_period_data = {
                key:  t.time_period_id,
                text: t.time_range,
                value: t.time_period_id,
            };
            time_period_options.push(time_period_data);
        }
        for( let m of data.meeting_days){
            let meeting_day_data = {
                key:  m.days_id,
                text: m.day_range,
                value: m.days_id,
            };
            meeting_day_options.push(meeting_day_data);
        }
        for( let b of data.buildings){

            let buildings_data = {
                key:  b.building_id,
                text: b.building_name,
                value: b.building_id,
            };
            building_options.push(buildings_data);
        }
        for( let r of data.rooms){
            let room_data = {
                key:  r.rooms_id,
                text: r.room_number,
                value: r.rooms_id,
            };
            room_options.push(room_data);
        }
        for( let f of data.faculty){
            let faculty_data = {
                key:  f.faculty_id,
                text: f.full_name,
                value: f.faculty_id,
            };
            faculty_options.push(faculty_data);
        }
        this.baseState = {
```

```
            this.baseState = {
                departments: department_options,
                time_periods: time_period_options,
                meeting_days: meeting_day_options,
                semester: semester_options,
                building: building_options,
                submittedValue: false,
                rooms: room_options,
                faculty: faculty_options,
                isLoading: false
            };

            this.setState(this.baseState);
        })
        .catch((error) => {
            console.error(error);
        })
}

handleInputChange(event) {
    this.setState({
        [event.target.id]: event.target.value
    })
}

onBlur(){
    this.setState({
        submittedValue: this.state.course_name_input,
        attributeFlag: 'course_name'
    })
}

handleSelectChange(value, id){
    console.log(value);
    console.log(id);
    if( value === true){
        console.log(this.state);
        this.setState(this.baseState);

    } else {
        let attribute_flag;
        switch (id) {
            case 'semester_select':
                attribute_flag = 'semester';
                break;
            case 'department_select':
                attribute_flag = 'department';
                break;
            case 'faculty_select':
                attribute_flag = 'faculty';
                break;
            case 'days_select':
```

```
        }
        console.log(value);
        this.setState({
            submittedValue: value,
            attributeFlag: attribute_flag
        });
    }
}

render() {

    if( this.state.isLoading){
        return(
            <Segment>
              <Dimmer active>
                <Loader size='massive'>Loading</Loader>
              </Dimmer>
            </Segment>
        );
    }
    return (
        <Container >
            <Form>
                <Form.Group widths='equal'>
                    <Form.Select id='department_select' onChange={(e, {value, id}) => this.handleSelectChange(value, id)} fluid label='Department Name'  options={this.state.departments} placeholder='Department Name'/
                    <Form.Input id='course_name_input' onBlur={this.onBlur} onChange={this.handleInputChange} fluid label='Course Name' placeholder='Course Name'/>
                </Form.Group>
                <Form.Select id='faculty_select' onChange={(e, {value, id}) => this.handleSelectChange(value, id)} fluid label='Faculty' options={this.state.faculty} placeholder='Faculty'/>
                <Form.Group widths='equal'>
                    <Form.Select id='days_select' onChange={(e, {value, id}) => this.handleSelectChange(value, id)} fluid label='Meeting Days' options={this.state.meeting_days} placeholder='Meeting Days'/>
                    <Form.Select id='time_period_select' onChange={(e, {value, id}) => this.handleSelectChange(value, id)} fluid label='Time Periods' options={this.state.time_periods} placeholder='Time Periods'/>
                    <Form.Select id='semester_select' onChange={(e, {value, id}) => this.handleSelectChange(value, id)} fluid label='Semester' options={this.state.semester} placeholder='Semesters'/>
                </Form.Group>
                <Form.Group widths='equal'>
                    <Form.Select id='building_select' onChange={(e, {value, id}) => this.handleSelectChange(value, id)} fluid label='Buildings' options={this.state.building} placeholder='Buildings'/>
                    <Form.Select id='room_select' onChange={(e, {value, id}) => this.handleSelectChange(value, id)} fluid label='Rooms' options={this.state.rooms} placeholder='Rooms'/>
                </Form.Group>

            </Form>
            <Divider horizontal>Results</Divider>
            { (this.state.submittedValue) &&
                <MasterTable search_value={this.state.submittedValue} attribute_flag={this.state.attributeFlag} />
            }
        </Container>
    );
}

xport default App;
```

Help Make PyCharm Better

To improve your experience, we would like to collect data on the plugins and features you use. No personal data will be collected. An archive of a few kilobytes will be sent weekly.

Share Anonymous Statistics Don't Share

```
import React from 'react';
import { Container, Divider, Header, Table, List, Segment, Dimmer, Loader } from 'semantic-ui-react';
import ScheduleRow from './row';


// TODO: Create table and implement data fetching
class MasterTable extends React.Component {

    constructor(props){
        super(props);
        this.state = {
            attribute_flag: this.props.attribute_flag,
            isLoading: true,
            search_value: this.props.search_value
        };
        // this.onBlur = this.onBlur.bind(this);
        // this.handleInputChange = this.handleInputChange.bind(this);
        // this.handleSelectChange = this.handleSelectChange.bind(this);
    }

    componentDidMount(){
         fetch("/student_system/student_system_api/get_schedule_data/"+ this.state.attribute_flag +"/" + this.state.search_value +"/")
            .then((response) => {
                if(!response.ok){
                    throw Error(response.statusText);
                }

                return response;
            })
            .then((response) => response.json())
            .then((data) => {
                console.log(data);
                this.setState({
                    isLoading: false,
                    table_data: data
                })
            })
            .catch((error) => {
                console.error(error);
            })
    }

    componentWillReceiveProps(nextProps){
        if(this.props.search_value !== nextProps.search_value){
             fetch("/student_system/student_system_api/get_schedule_data/"+ nextProps.attribute_flag +"/" + nextProps.search_value +"/")
            .then((response) => {
                if(!response.ok){
                    throw Error(response.statusText);
                }
                return response;
            })
            .then((response) => response.json())
             then((data) => {
```

MasterTable › componentDidMount() › callback for then()

```
                throw Error(response.statusText);
            }
            return response;
        })
        .then((response) => response.json())
        .then((data) => {
            console.log(data);
            this.setState({
                isLoading: false,
                table_data: data
            })
        })
        .catch((error) => {
            console.error(error);
        })
    }

}

render() {
     if( this.state.isLoading){
        return(
            <div style={{height: '100%'}}><Segment >
              <Dimmer active>
                <Loader size='massive'>Loading</Loader>
              </Dimmer>
            </Segment></div>
        );
    }
    let rows;
    if( this.state.table_data){
        rows = this.state.table_data.map((item, i) => {
            return <ScheduleRow
                    key={ i }
                    data={ item }
                    // onClick={() => this.handleClick(item.section_id)}
                    />
        })
    }

    return(
        <div>
            <Table color={'black'} celled selectable padded>
                <Table.Header >
                    <Table.Row>
                        <Table.HeaderCell singleLine>Section</Table.HeaderCell>
                        <Table.HeaderCell>Course Name</Table.HeaderCell>
                        <Table.HeaderCell>Professor</Table.HeaderCell>
                        <Table.HeaderCell>Credits</Table.HeaderCell>
                        <Table.HeaderCell width={3}>Seating</Table.HeaderCell>
                        <Table.HeaderCell>Time Slot</Table.HeaderCell>
                        <Table.HeaderCell>Semester</Table.HeaderCell>
```

MasterTable › componentDidMount() › callback for then()

Back-end Implementation

```
def get_master_schedule_input_data(request):
    departments = Department.objects.all()
    general_data = {}
    department_array = []
    for d in departments:
        data = {
            'department_id': d.department_id,
            'department_name': d.name
        }
        department_array.append(data)
    faculty = []
    for f in Faculty.objects.raw("SELECT u.first_name, u.last_name, f.faculty_id_id "
                                 "FROM registration_system_faculty AS f, auth_user as u, registration_system_userprofile as up "
                                 "WHERE up.user_id = u.id "
                                 "AND up.id = f.faculty_id_id"):
        faculty.append({
            'first_name': f.first_name,
            'last_name': f.last_name,
            'full_name': f.first_name + " " + f.last_name,
            'faculty_id': f.faculty_id_id
        })
    time_periods = []
    for t in Period.objects.all():
        time_periods.append({
            'time_period_id': t.period_id,
            'time_range': t.start_time.strftime('%H:%M %p') + ' ' + t.end_time.strftime('%H:%M %p')
        })
    meeting_days = []
    for m in MeetingDays.objects.all():
        if m.day_3:
            meeting_days.append({
                'days_id': m.days_id,
                'day_1': m.day_1,
                'day_2': m.day_2,
                'day_3': m.day_3,
                'day_range': m.day_1 + " " + m.day_2 + " " + m.day_3
            })
        elif m.day_2 and m.day_3 is None:
            meeting_days.append({
                'days_id': m.days_id,
                'day_1': m.day_1,
                'day_2': m.day_2,
                'day_range': m.day_1 + " " + m.day_2
            })
        elif m.day_1 and m.day_2 is None:
            meeting_days.append({
                'days_id': m.days_id,
```

```
            meeting_days.append({
                'days_id': m.days_id,
                'day_1': m.day_1,
                'day_range': m.day_1
            })
    buildings = []
    for b in Building.objects.all():
        buildings.append({
            'building_id': b.building_id,
            'building_name': b.name
        })
    rooms = []
    for arrgh in Room.objects.all():
        rooms.append({
            'rooms_id': arrgh.room_id,
            'room_number': arrgh.room_number
        })
    semester = []
    for s in Semester.objects.all():
        semester.append({
            'semester_id': s.semester_id,
            'semester_year': s.year,
            'semester_season': s.season,
            'semester_status': s.status
        })
    # print(semester)
    general_data['semesters'] = semester
    general_data['time_periods'] = time_periods
    general_data['meeting_days'] = meeting_days
    general_data['buildings'] = buildings
    general_data['rooms'] = rooms
    general_data['faculty'] = faculty
    general_data['departments'] = department_array

    return HttpResponse(json.dumps(general_data), content_type="application/json")
```

```
def get_master_schedule_search_data(request, attribute_flag, search_value):
    section = None
    schedule_data = []
    if attribute_flag == 'department':
        section = Section.objects.filter(course_id__department_id_id=int(search_value))
    elif attribute_flag == 'faculty':
        section = Section.objects.filter(faculty_id__faculty_id_id=int(search_value))
    elif attribute_flag == 'days':
        section = Section.objects.filter(time_slot_id__days_id_id=int(search_value))
    elif attribute_flag == 'time_period':
        section = Section.objects.filter(time_slot_id__period_id_id=int(search_value))
    elif attribute_flag == 'building':
        section = Section.objects.filter(course_id__department_id__building_id_id=int(search_value))
    elif attribute_flag == 'rooms':
        section = Section.objects.filter(room_id_id=int(search_value))
    elif attribute_flag == 'course_name':
        section = Section.objects.filter(course_id__name__contains=search_value)
    elif attribute_flag == 'semester':
        section = Section.objects.filter(semester_id=int(search_value))

    for s in section:
        faculty = Faculty.objects.get(pk=int(s.faculty_id_id))
        faculty_name = faculty.faculty_id.user.first_name + ' ' + faculty.faculty_id.user.last_name
        prerequisites = Prerequisite.objects.filter(course_id=s.course_id)
        prereq_array = []
        for p in prerequisites:
            prereq_array.append({
                'name': p.course_required_id.name
            })
        data = {
            'faculty': faculty_name,
            'course_name': s.course_id.name,
            'course_description': s.course_id.description,
            'department': s.course_id.department_id.name,
            'semester_year': s.semester_id.year,
            'semester_season': s.semester_id.season,
            'semester_status': s.semester_id.status,
            'section_id': s.section_id,
            'credits': s.course_id.credits,
            'seats_taken': s.seats_taken,
            'capacity': s.room_id.capacity,
            'meeting_days': s.time_slot_id.days_id.day_1 + ' ' + s.time_slot_id.days_id.day_2,
            'time_period': s.time_slot_id.period_id.start_time.strftime('%H:%M %p') + '-'
                           + s.time_slot_id.period_id.end_time.strftime('%H:%M %p'),
            'room': s.room_id.room_number,
            'building': s.room_id.building_id.name,
            'prerequisites': prereq_array
        }
        schedule_data.append(data)

    return HttpResponse(json.dumps(schedule_data), content_type="application/json")
```

Login and User Authentication

Back-end Implementation

- When the request to login is fired from the front-end page, the system utilizes cookie-based user sessions to handle users, accounts, and permissions. This system takes care of both authorization and authentication by verifying the current UserID against the one currently set for the session.
- If the user is found in the system and the encrypted password matches the user's record saved in the database the user will be redirected to their appropriate path, be it an Admin, Faculty, Student, or Researcher.
- If not, the login form is displayed with an error message indicating what went wrong.

```python
class LoginView(SuccessURLAllowedHostsMixin, FormView):
    """
    Display the login form and handle the login action.
    """
    form_class = AuthenticationForm
    authentication_form = None
    redirect_field_name = REDIRECT_FIELD_NAME
    template_name = 'registration/login.html'
    redirect_authenticated_user = False
    extra_context = None

    @method_decorator(sensitive_post_parameters())
    @method_decorator(csrf_protect)
    @method_decorator(never_cache)
    def dispatch(self, request, *args, **kwargs):
        if self.redirect_authenticated_user and self.request.user.is_authenticated:
            redirect_to = self.get_success_url()
            if redirect_to == self.request.path:
                raise ValueError(
                    "Redirection loop for authenticated user detected. Check that "
                    "your LOGIN_REDIRECT_URL doesn't point to a login page."
                )
            return HttpResponseRedirect(redirect_to)
        return super().dispatch(request, *args, **kwargs)

    def get_success_url(self):
        url = self.get_redirect_url()
        return url or resolve_url(settings.LOGIN_REDIRECT_URL)

    def get_redirect_url(self):
        """Return the user-originating redirect URL if it's safe."""
        redirect_to = self.request.POST.get(
            self.redirect_field_name,
            self.request.GET.get(self.redirect_field_name, '')
        )
        url_is_safe = is_safe_url(
            url=redirect_to,
            allowed_hosts=self.get_success_url_allowed_hosts(),
            require_https=self.request.is_secure(),
        )
        return redirect_to if url_is_safe else ''

    def get_form_class(self):
        return self.authentication_form or self.form_class

    def get_form_kwargs(self):
        kwargs = super().get_form_kwargs()
        kwargs['request'] = self.request
        return kwargs

    def form_valid(self, form):
```

ed for coding assistance. (today 3:31 PM)

```
    def get_form_kwargs(self):
        kwargs = super().get_form_kwargs()
        kwargs['request'] = self.request
        return kwargs

    def form_valid(self, form):
        """Security check complete. Log the user in."""
        auth_login(self.request, form.get_user())
        return HttpResponseRedirect(self.get_success_url())

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        current_site = get_current_site(self.request)
        context.update({
            self.redirect_field_name: self.get_redirect_url(),
            'site': current_site,
            'site_name': current_site.name,
        })
        if self.extra_context is not None:
            context.update(self.extra_context)
        return context


def login(request, template_name='registration/login.html',
          redirect_field_name=REDIRECT_FIELD_NAME,
          authentication_form=AuthenticationForm,
          extra_context=None, redirect_authenticated_user=False):
    warnings.warn(
        'The login() view is superseded by the class-based LoginView().',
        RemovedInDjango21Warning, stacklevel=2
    )
    return LoginView.as_view(
        template_name=template_name,
        redirect_field_name=redirect_field_name,
        form_class=authentication_form,
        extra_context=extra_context,
        redirect_authenticated_user=redirect_authenticated_user,
    )(request)
```

- I use a middleware named LoginRequiredMixin for all routes that are exclusive to authenticated users.

```
class UserDisplay(LoginRequiredMixin, generic.View):
```

- Before any function exclusive to Admin, Student, Faculty, and Researcher is invoked each method contains an authorization check like the one below.

```
is_part_time_student = False
is_full_time_student = False
is_part_time_faculty = False
is_full_time_faculty = False
is_faculty = False
is_admin = False
is_researcher = False

def get(self, request):
    user = request.user
    userprofile = UserProfile.objects.get(user=user)
    if userprofile:
        if userprofile.has_student():
            student = Student.objects.get(student_id=userprofile)
            if student.has_full_time_student():
                self.is_full_time_student = True
            elif student.has_part_time_student():
                self.is_part_time_student = True
        elif userprofile.has_admin():
            self.is_admin = True
        elif userprofile.has_researcher():
            self.is_researcher = True
        elif userprofile.has_faculty():
            faculty = Faculty.objects.get(faculty_id=userprofile)
            # print(faculty)
            self.is_faculty = True
    is_student = self.is_full_time_student or self.is_part_time_student
```

Front-end Implementation

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}
    <div id="login-content" class="m-top12">
      <h2 class="ui black image header"><div class="content">Log-in to your account</div></h2>
      <form class="ui large form " method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button class="ui fluid medium black submit button" type="submit">Login</button>
      </form>
      <br>
{#      <p><strong>-- OR --</strong></p>#}
{#      <a href="{% url 'social:begin' 'github' %}">Login with GitHub</a><br>#}
    </div>
{% endblock %}
```

Admin Implementation:

Add/Update Hold

Header	
Use Case ID	UC9
Use Case Version	1.0
Body	
Title	Add/UpdateHold
Actors	Admin
Input	Student ID
Output	Hold updated
Normal Flow	1. Admin navigates to the Add/UpdateHold option from the administrative interface. 2. Admin is displayed menu to query student by name or id. 3. Admin queries for a student to add/update an academic/financial hold. 4. Student information is displayed to the admin. 5. Admin updates or adds hold. 6. Confirmation message displayed. (*Optional: E-mail sent to student*).
Alternate Flows	N/A
Entry Condition	Admin is logged in.
Exit Condition	Hold is updated and user is notified.

- Admin is required to input Student First Name and Last Name.
- Students are appended to the table structure. If students have a hold placed on their account already the remove button is displayed allowing admins to remove a hold placed on the student account. If a hold is not currently placed Admins may choose from the different hold types (Disciplinary, Medical, Financial, Disciplinary) to place on the selected student account.

Front-end Implementation

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

     <div class="ui container">
        {{ rendered|safe }}
    </div>

    <div class="ui container m-top2">
        <div class="ui form" id="ui-form">
        {{ csrf }}
            <div class="two fields">
                <div class="field">
                    <label for="first_name_id">First Name</label>
                    <input id="first_name_id"   placeholder="Search by first name..." type="text" />
                </div>
                <div class="field">
                    <label for="last_name_id">Last Name</label>
                    <input id="last_name_id" placeholder="Search by last name..." type="text" />
                </div>
            </div>
            <div class="display-inline ">
                <button class="ui primary button" id="search" type="button">Search</button>
            </div>
            <table class="ui celled table">
                <thead>
                    <tr>
                        <th>User ID</th>
                        <th>Username</th>
                        <th>Email</th>
                        <th>First Name</th>
                        <th>Last Name</th>
                        <th>Hold Type</th>
                        <th>Add/Remove Hold</th>
                    </tr>
                </thead>
                <tbody id="students_added">

                </tbody>
            </table>
        </div>
        <div id="success">

        </div>
    </div>

    <script>
        function addStudent(){
```

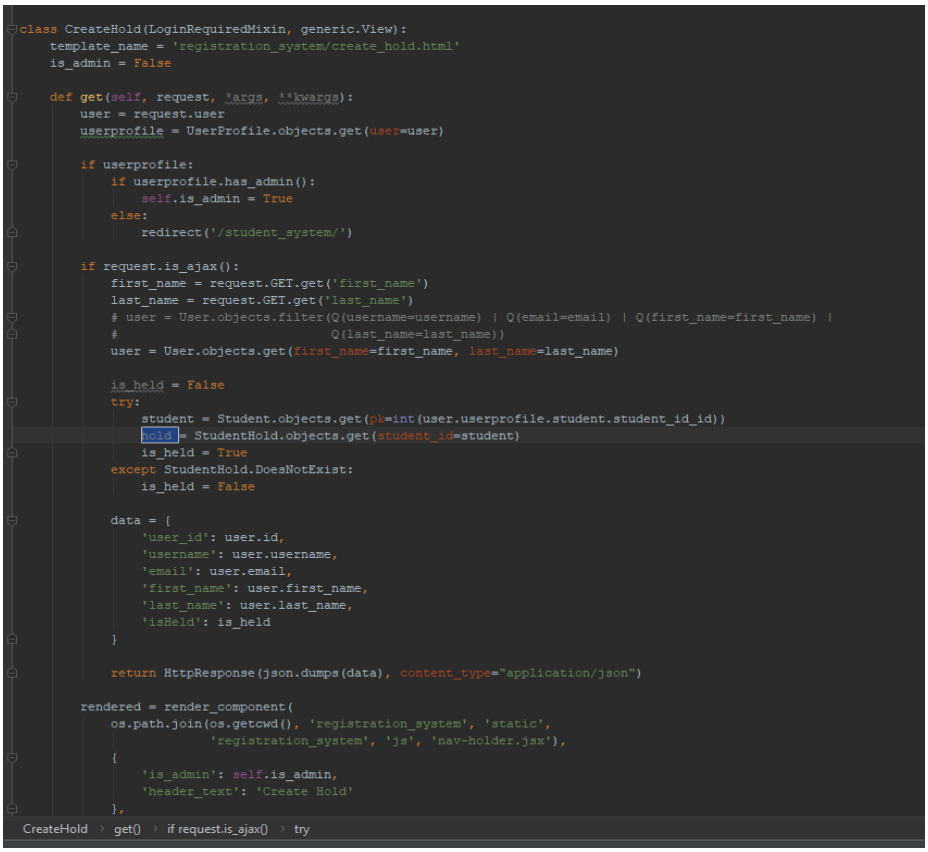
```
<script>
    function addStudent(){
        console.log("here");
        var first_name = $('#first_name_id').val();
        var last_name = $('#last_name_id').val();
        if (!first_name || !last_name) {
            $('#success').append("<h2>Please Specify first and last name!</h2>");
            return;
        } else {
            $('#success').empty();
        }
        $.ajax({
            type: "GET",
            url: "/student_system/create_hold/",
            data : {
                first_name: first_name,
                last_name: last_name
            },
            success: function(element) {
                var tBody = document.getElementById('students_added');
                var newRow = tBody.insertRow(tBody.rows.length);
                newRow.id = element.user_id + '_row';
                var userIdCell = newRow.insertCell(0);
                var usernameCell = newRow.insertCell(1);
                var emailCell = newRow.insertCell(2);
                var firstNameCell = newRow.insertCell(3);
                var lastNameCell = newRow.insertCell(4);
                var holdTypeCell = newRow.insertCell(5);
                var updateCell = newRow.insertCell(6);
                var userIdText = document.createTextNode(element.user_id);
                var usernameText = document.createTextNode(element.username);
                var emailText = document.createTextNode(element.email);
                var firstNameText = document.createTextNode(element.first_name);
                var lastNameText = document.createTextNode(element.last_name);
                userIdCell.appendChild(userIdText);
                usernameCell.appendChild(usernameText);
                emailCell.appendChild(emailText);
                firstNameCell.appendChild(firstNameText);
                lastNameCell.appendChild(lastNameText);
                holdTypeCell.insertAdjacentHTML('afterbegin','<select class="ui fluid dropdown" name="hold_name" id="hold_name_id"><option val
                if(element.isHeld){
                    updateCell.insertAdjacentHTML('afterbegin', '<button type="button" onclick="deleteHold(this.id)" class="ui negative button
                } else {
                    updateCell.insertAdjacentHTML('afterbegin', '<button type="button" onclick="createHold(this.id)" class="ui positive button
                }

            }
        })

    }
```

```
function createHold(user_id) {
    $('#success').empty();
    $.ajax({
        type: "POST",
        url: "/student_system/create_hold/",

        data: {
            userID: user_id,
            hold: $('#hold_name_id').val(),
            csrfmiddlewaretoken: '{{ csrf_token }}'
        },

        success: function(){
            $('#'+user_id+'_row').remove();
            $('#success').append("<div class=\"ui positive message\">\n" +
                "  <div class=\"header\">\n" +
                "    Hold Appended successfully!\n" +
                "  </div>\n" +
                "</div>")
        }
    })
}

function deleteHold(user_id) {
    $('#success').empty();
    $.ajax({
        type: "POST",
        url: "/student_system/create_hold/",

        data: {
            userID: user_id,
            isRemove: true,
            csrfmiddlewaretoken: '{{ csrf_token }}'
        },

        success: function(){
            $('#'+user_id+'_row').remove();
            $('#success').append("<div class=\"ui positive message\">\n" +
                "  <div class=\"header\">\n" +
                "    Hold Removed successfully!\n" +
                "  </div>\n" +
                "</div>")
        }
    })

}

$('#search').click(addStudent);
$('#clear').click(removeStudents);
```

ript › createHold() › success()

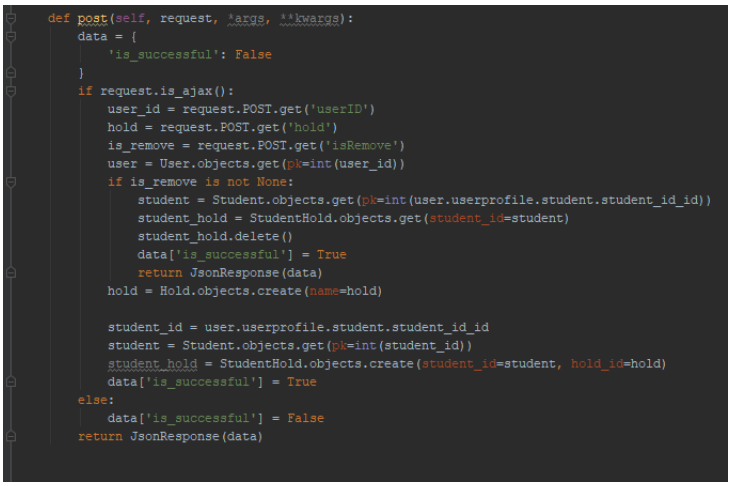
Back-end Implementation

- Back-end's POST route receive Ajax request from front-end template and creates a hold on the student id receive by the Ajax request.
- Back-end's GET route receives Ajax request for student record and sends the JSON response data containing student record information.



```
class CreateHold(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/create_hold.html'
    is_admin = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)

        if userprofile:
            if userprofile.has_admin():
                self.is_admin = True
            else:
                redirect('/student_system/')

        if request.is_ajax():
            first_name = request.GET.get('first_name')
            last_name = request.GET.get('last_name')
            # user = User.objects.filter(Q(username=username) | Q(email=email) | Q(first_name=first_name) |
            #                            Q(last_name=last_name))
            user = User.objects.get(first_name=first_name, last_name=last_name)

            is_held = False
            try:
                student = Student.objects.get(pk=int(user.userprofile.student.student_id_id))
                hold = StudentHold.objects.get(student_id=student)
                is_held = True
            except StudentHold.DoesNotExist:
                is_held = False

            data = {
                'user_id': user.id,
                'username': user.username,
                'email': user.email,
                'first_name': user.first_name,
                'last_name': user.last_name,
                'isHeld': is_held
            }

            return HttpResponse(json.dumps(data), content_type="application/json")

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_admin': self.is_admin,
                'header_text': 'Create Hold'
            },
```

CreateHold › get() › if request.is_ajax() › try



```
    def post(self, request, *args, **kwargs):
        data = {
            'is_successful': False
        }
        if request.is_ajax():
            user_id = request.POST.get('userID')
            hold = request.POST.get('hold')
            is_remove = request.POST.get('isRemove')
            user = User.objects.get(pk=int(user_id))
            if is_remove is not None:
                student = Student.objects.get(pk=int(user.userprofile.student.student_id_id))
                student_hold = StudentHold.objects.get(student_id=student)
                student_hold.delete()
                data['is_successful'] = True
                return JsonResponse(data)
            hold = Hold.objects.create(name=hold)

            student_id = user.userprofile.student.student_id_id
            student = Student.objects.get(pk=int(student_id))
            student_hold = StudentHold.objects.create(student_id=student, hold_id=hold)
            data['is_successful'] = True
        else:
            data['is_successful'] = False
        return JsonResponse(data)
```

Create Course

Header	
Use Case ID	UC10
Use Case Version	1.0
Body	
Title	CreateCourse
Actors	Admin
Input	Course Information: Course Name, Prereq's, Teachers,etc.
Output	Course created
Normal Flow	1. Admin navigates to the CreateCourse options from their interface. 2. Form is displayed for admin to enter all required information to create a course. 3. Form is submitted with all required information filled out. 4. Course Is created and confirmation message is displayed.
Alternate Flows	3a. Form inputs were not filled out correctly, error message displayed, move back to step 2.
Entry Condition	Admin is logged in.
Exit Condition	Course is created and confirmation to user is displayed.

- Admin is required to input Course Name, Credits, Description, and Select a department to create a course.
- Upon Submit Admin is redirected the CreatePrerequisite page to append prerequisite courses the course created.

Front-end Implementation

```
{% block navbar %}
   {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}
{#    {% csrf_token %}#}
   <div class="ui container">
       {{ rendered|safe }}
   </div>

   <div class="ui container">
       <form action="{{ url }}" method="post" class="ui form m-top2 ">
           {% csrf_token %}

               <div class="two fields">
                   <div class="field">
                       <label for="id_name">Course Name</label>
                       <input type="text" name="name" maxlength="100" placeholder="Enter Course Name" id="id_name" required/>
                   </div>
                   <div class="field">
                       <label for="id_credits">Credits</label>
                       <input type="number" name="credits" step="1" min="2" max="4" id="id_credits" required>
                   </div>
               </div>

           <div class="field">
               <label for="id_description">Description</label>
               <textarea class="text-area-styled" placeholder="Enter Course Description..." name="description" maxlength="1000" id="id_description" required
               </textarea>
           </div>
           <div class="inline fields">
               <div class="field">
                   <label for="id_department_id"></label>
                   <select class="ui fluid dropdown" name="department_id" id="id_department_id" required>
                       <option value="true">-------</option>
                       {% if departments %}
                           {% for department in departments %}
                               <option value="{{ department.department_id }}" label="{{ department.name }}">{{ department.name }}</option>
                           {% endfor %}
                       {% endif %}
                   </select>
               </div>
           </div>
           <div class="field">
               <button class="ui button" role="button">Submit</button>
           </div>
       </form>
   </div>


{#    {{ form }}#}
```

- Once the course is submitted the user is redirected to the prerequisites page to append prerequisite courses to the one the Admin created.

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

     <div class="ui container">
        {{ rendered|safe }}
    </div>

    <div class="ui container m-top2">
        <div class="ui form" id="ui-form">
            <div class="two fields">
                <div class="field">
                    <label for="course_id">Course ID</label>
                    <input id="course_id" type="text" value="{{ value }}" readonly/>
                </div>
                <div class="field">
                    <label for="courses_required_id">Course Required</label>
                    <select id="courses_required_id" class="ui fluid dropdown">
                        <option value="0">--------</option>
                        {% if courses %}
                            {% for course in courses %}
                                <option value="{{ course.course_id }}" label="{{ course.name }}">{{ course.name }}</option>
                            {% endfor %}
                        {% endif %}
                    </select>
                </div>
            </div>
        {{ csrf }}
            <div class="display-inline ">
                <button class="ui primary button" id="add_course" type="button">Add Course</button>
                <button class="ui negative button" id="clear_courses" type="button">Clear</button>
            </div>
            <table class="ui celled table">
                <thead>
                    <tr>
                        <th>Course ID</th>
                        <th>Course Name</th>
                    </tr>
                </thead>
                <tbody id="courses_added">

                </tbody>
            </table>
            <button class="ui positive button" id="submit" type="button">Submit</button>
        </div>
        <div id="success">

        </div>
script  ›  submitCourses()  ›  success()
```

```
function addCourse(){
    console.log("here");
    var course = document.getElementById('courses_required_id');
    var course_name = course.options[course.selectedIndex].label;
    var course_id = course.options[course.selectedIndex].value;
    var tBody = document.getElementById('courses_added');
    var newRow = tBody.insertRow(tBody.rows.length);
    var courseIdCell = newRow.insertCell(0);
    var courseNameCell = newRow.insertCell(1);
    var courseNameText = document.createTextNode(course_name);
    var courseIdText = document.createTextNode(course_id);
    courseIdCell.appendChild(courseIdText);
    courseNameCell.appendChild(courseNameText);

}

function removeCourses() {
    $('#courses_added').empty();
}

function submitCourses() {
    var table = document.getElementById('courses_added');
    var course_id = document.getElementById('course_id').value;
    var courseIDs = [];

    for( var j=0; j < table.rows.length; j++){
        courseIDs.push(table.rows[j].cells[0].textContent);
    }

    $.ajax({
        type: "POST",
        url: "/student_system/create_course/prerequisites/"+course_id+"/",

        data: {
            courseID: course_id,
            prerequisites: JSON.stringify(courseIDs),
            csrfmiddlewaretoken: '{{ csrf_token }}'
        },

        success: function(){
            $('#ui-form').hide();
            $('#success').append("<div class=\"ui positive message\">\n" +
                "  <div class=\"header\">\n" +
                "    Course Prerequisite created successfully!\n" +
                "  </div>\n" +
                "</div>")
        }
    })

}
```

script › submitCourses() › success()

Control

Back-end Implementation

- If the front-end form inputs are valid the webpage will be redirected to the CreatePrerequisites page.

```
class CreateCourse(LoginRequiredMixin, generic.View):
    form_class = CreateCourseForm
    template_name = 'registration_system/create_course.html'
    is_admin = False

    def get(self, request, *args, **kwargs):
        form = self.form_class()
        user = request.user
        userprofile = UserProfile.objects.get(user=user)
        if userprofile:
            if userprofile.has_admin():
                self.is_admin = True
            else:
                redirect('/student_system/')

        departments = Department.objects.all()

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_admin': self.is_admin,
                'header_text': 'Create Course'
            },
            to_static_markup=False,
        )
        return render(request, self.template_name, {'rendered': rendered, 'form': form, 'departments': departments})

    def post(self, request, *args, **kwargs):
        form = self.form_class(request.POST)
        # print(form.is_valid())
        # print(form.errors)
        if form.is_valid():
            name = form.cleaned_data['name']
            description = form.cleaned_data['description']
            credits_value = form.cleaned_data['credits']
            department_id = form.cleaned_data['department_id']
            course = Course.objects.create(name=name, description=description, credits=credits_value,
                                           department_id=department_id)
            return redirect('create_prerequisites', course_id=course.course_id)
        return redirect('/student_system/create_course/')


class UserDisplay(LoginRequiredMixin, generic.View):
```

- The CreatePrerequisite back-end function will utilize the data sent in the get requests URL path to allow the user to create prerequisites for the course created.
- Post route utilizes data sent in the Ajax request from the create_prerequisite template to make the necessary insert/updates into the database and return a success flag to the sender.

```python
class CreatePrerequisite(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/create_prerequisite.html'
    is_admin = False

    def get(self, request, course_id, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)

        if userprofile:
            if userprofile.has_admin():
                self.is_admin = True
            else:
                redirect('/student_system/')

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_admin': self.is_admin,
                'header_text': 'Create Prerequisites'
            },
            to_static_markup=False,
        )

        courses = Course.objects.all()
        context = {
            'courses': courses,
            'rendered': rendered,
            'value': course_id
        }
        return render(request, self.template_name, context)

    @csrf_exempt
    def post(self, request, *args, **kwargs):
        if request.is_ajax():
            message = "Ajax"
            course_id = request.POST.get('courseID')
            prerequisites = json.loads(request.POST.get('prerequisites'))
            course = Course.objects.get(pk=int(course_id))
            for p in prerequisites:
                prerequisite_course = Course.objects.get(pk=int(p))
                Prerequisite.objects.create(course_id=course, course_required_id=prerequisite_course)
        else:
            message = "No Ajax"
        return HttpResponse(message)
```

CreatePrerequisite › get()

Create Advising

Header	
Use Case ID	UC11
Use Case Version	1.0
Body	
Title	Create Advising
Actors	Admin
Input	Create Advising
Output	Advising Assigned.
Normal Flow	1. Admin navigates to the CreateAdvising option from their administrative interface. 2. Inputs for first name and last name of student are displayed. 3. Administrator assigns faculty to student from the faculty dropdown column. 4. Confirmation displayed.
Alternate Flows	N/A
Entry Condition	Admin is logged in.
Exit Condition	Faculty is assigned to advise the student chosen.

Front-end Implementation

- Admin is shown two text inputs to fill in Student First name or Last name.
- Students are appended to a table structure, each row has a dropdown and button attached to it to assign advisers, or update advisers.

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
   {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

    <div class="ui container">
       {{ rendered|safe }}
   </div>

   <div class="ui container m-top2">
      <div class="ui form" id="ui-form">
      {{ csrf }}
          <div class="two fields">
             <div class="field">
                <label for="first_name_id">First Name</label>
                <input id="first_name_id"   placeholder="Search by first name..." type="text" />
             </div>
             <div class="field">
                <label for="last_name_id">Last Name</label>
                <input id="last_name_id" placeholder="Search by last name..." type="text"/>
             </div>
          </div>
          <div class="display-inline ">
             <button class="ui primary button" id="search" type="button">Search</button>
          </div>
          <table class="ui celled table">
             <thead>
                <tr>
                   <th>User ID</th>
                   <th>Username</th>
                   <th>Email</th>
                   <th>First Name</th>
                   <th>Last Name</th>
                   <th>Faculty</th>
                   <th>Add/Update Advising</th>
                </tr>
             </thead>
             <tbody id="students_added">

             </tbody>
          </table>
      </div>
      <div id="success">

      </div>
   </div>

   <script>
      function addStudent(){
script 〉 createAdvising() 〉 success()
```

```
function addStudent(){
    console.log("here");
    var first_name = $('#first_name_id').val();
    var last_name = $('#last_name_id').val();
    $.ajax({
        type: "GET",
        url: "/student_system/create_advising/",
        data : {
            first_name: first_name,
            last_name: last_name
        },
        success: function(data) {
            console.log(data);
            var tBody = document.getElementById('students_added');
            var newRow = tBody.insertRow(tBody.rows.length);
            newRow.id = data.user_id + '_row';
            var userIdCell = newRow.insertCell(0);
            var usernameCell = newRow.insertCell(1);
            var emailCell = newRow.insertCell(2);
            var firstNameCell = newRow.insertCell(3);
            var lastNameCell = newRow.insertCell(4);
            var facultyCell = newRow.insertCell(5);
            var updateCell = newRow.insertCell(6);
            var userIdText = document.createTextNode(data.user_id);
            var usernameText = document.createTextNode(data.username);
            var emailText = document.createTextNode(data.email);
            var firstNameText = document.createTextNode(data.first_name);
            var lastNameText = document.createTextNode(data.last_name);
                userIdCell.appendChild(userIdText);
                usernameCell.appendChild(usernameText);
                emailCell.appendChild(emailText);
                firstNameCell.appendChild(firstNameText);
                lastNameCell.appendChild(lastNameText);
                facultyCell.insertAdjacentHTML('afterbegin',
                    '<select id="id_faculty_id_'+data.user_id+'" class="ui input" style="min-width: 9rem;">' +
                    '</select>' );
                var facultySelect = facultyCell.firstChild;
                data.faculty_array.forEach(function(element){
                    console.log("here");
                    facultySelect.insertAdjacentHTML('afterbegin',
                        '<option value="'+element.faculty_id+'" >'+element.full_name+'</option>')
                });
                if(data.isAdvised === true){
                    updateCell.insertAdjacentHTML('afterbegin', '<button type="button" onclick="updateAdvising(this.id)" class="ui info but
                } else {
                    updateCell.insertAdjacentHTML('afterbegin', '<button type="button" onclick="createAdvising(this.id)" class="ui positive
                }
        }
    })

}
```

```
function removeStudents() {
    $('#students_added').empty();
}

function createAdvising(user_id) {
    $('#success').empty();
    $.ajax({
        type: "POST",
        url: "/student_system/create_advising/",

        data: {
            userID: user_id,
            faculty: $('#id_faculty_id_'+user_id).val(),
            csrfmiddlewaretoken: '{{ csrf_token }}'
        },

        success: function(){
            $('#'+user_id+'_row').remove();
            $('#success').append("<div class=\"ui positive message\">\n" +
                "  <div class=\"header\">\n" +
                "    Advising created successfully!\n" +
                "  </div>\n" +
                "</div>")
        }
    })
}
{# TODO: FINISH UPDATE ADVISING        #}
function updateAdvising(user_id) {
    $('#success').empty();
    $.ajax({
        type: "POST",
        url: "/student_system/create_advising/",

        data: {
            userID: user_id,
            faculty: $('#id_faculty_id_'+user_id).val(),
            isUpdate: true,
            csrfmiddlewaretoken: '{{ csrf_token }}'
        },

        success: function(){
            $('#'+user_id+'_row').remove();
            $('#success').append("<div class=\"ui positive message\">\n" +
                "  <div class=\"header\">\n" +
                "    Advising updated successfully!\n" +
                "  </div>\n" +
                "</div>")
        }
    })
```

Back-end Implementation

- If a HTTP GET Request is sent from Ajax, back-end responds with the student data related to the input.
- If a HTTP POST Request is sent, back-end applies the Advising the student account selected.

```
class CreateAdvising(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/create_advising.html'
    is_admin = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)

        if userprofile:
            if userprofile.has_admin():
                self.is_admin = True
            else:
                redirect('/student_system/')

        if request.is_ajax():
            first_name = request.GET.get('first_name')
            last_name = request.GET.get('last_name')
            # user = User.objects.filter(Q(username=username) | Q(email=email) | Q(first_name=first_name) |
            #                            Q(last_name=last_name))
            user = User.objects.get(first_name=first_name, last_name=last_name)

            is_advised = True
            try:
                student = Student.objects.get(pk=int(user.userprofile.student.student_id_id))
                advising = Advising.objects.get(student_id=student)
            except Advising.DoesNotExist:
                is_advised = False

            faculty = []
            for f in Faculty.objects.raw("SELECT u.first_name, u.last_name, f.faculty_id_id "
                                         "FROM registration_system_faculty AS f, auth_user as u, registration_system_userprofile as up "
                                         "WHERE up.user_id = u.id "
                                         "AND up.id = f.faculty_id_id"):
                faculty.append({
                    'first_name': f.first_name,
                    'last_name': f.last_name,
                    'full_name': f.first_name + " " + f.last_name,
                    'faculty_id': f.faculty_id_id
                })

            data = {
                'user_id': user.id,
                'username': user.username,
                'email': user.email,
                'first_name': user.first_name,
                'last_name': user.last_name,
                'isAdvised': is_advised,
```

CreateAdvising

ntrol

```
                'first_name': user.first_name,
                'last_name': user.last_name,
                'isAdvised': is_advised,
                'faculty_array': faculty
            }

            return HttpResponse(json.dumps(data), content_type="application/json")

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_admin': self.is_admin,
                'header_text': 'Create Advising'
            },
            to_static_markup=False,
        )

        context = {
            'rendered': rendered
        }

        return render(request, self.template_name, context)

    def post(self, request, *args, **kwargs):
        data = {
            'is_successful': False
        }
        if request.is_ajax():
            user_id = request.POST.get('userID')
            faculty_id = request.POST.get('faculty')
            faculty = Faculty.objects.get(pk=int(faculty_id))
            # = request.POST.get('hold')
            is_update = request.POST.get('isUpdate')
            student = Student.objects.get(student_id__user_id=user_id)
            if is_update is not None:
                # TODO: Update advising
                advising = Advising.objects.get(student_id=student)
                advising.faculty_id = faculty
                advising.save()
                data['is_successful'] = True
                return JsonResponse(data)
            advising = Advising.objects.create(faculty_id=faculty, student_id=student)
            data['is_successful'] = True
        else:
            data['is_successful'] = False
        return JsonResponse(data)
```

CreateAdvising

Update Course

Header	
Use Case ID	UC12
Use Case Version	1.0
Body	
Title	UpdateCourse
Actors	Admin
Input	Course ID
Output	Course updated
Normal Flow	1. Admin navigates to the UpdateCourse options from their interface, Dropdown is displayed to select from Department. 2. Form is submitted with Courses are appended to table. 3. Admin Fills out updated course information and presses submit. 4. Course I updated.
Alternate Flows	5a. Form inputs were not filled out correctly, error message displayed, move back to step 5.
Entry Condition	Admin is logged in.
Exit Condition	Course information updated.

Front-end Implementation

- Admin is show dropdown to choose which department to update course from.
- Courses are shown in table rows with inputs for updating Description, Credits, and Name.

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

     <div class="ui container">
        {{ rendered|safe }}
    </div>

    <div class="ui container m-bottom4 m-top2">
        {% if sections %}
            {% for s in sections %}
                <div class="ui segments" id="{{ s.section_id }}_segment">
                    <div class="ui segment">
                      <h1 style="...">{{ s.course_name }}</h1>  <span class="ui right aligned float-right course-close" style="..."><
                    </div>
                    <div class="ui segments">
                        <h5 class="ui attached header">
                          Professor
                        </h5>
                        <div class="ui attached segment">
                          <p>{{ s.professor }}</p>
                        </div>
                        <h5 class="ui attached header">
                          Credits
                        </h5>
                        <div class="ui attached segment">
                          <p>{{ s.credits }}</p>
                        </div>
                        <h5 class="ui attached header">
                            Location
                        </h5>
                        <div class="ui attached segment">
                          <p>{{ s.building }} -- {{ s.room_number }}</p>
                        </div>
                        <h5 class="ui attached header">
                            Time Slot
                        </h5>
                        <div class="ui attached segment">
                          <p>{{ s.meeting_days }}</p><br/>
                            <p>{{ s.time_period }}</p>
                        </div>
                        <h5 class="ui attached header">
                            Seating
                        </h5>
                        <div class="ui attached segment">
                          <p>{{ s.seats_taken }}/{{ s.seating_capacity }}</p>
                        </div>
```

```
                    <h5 class="ui attached header">
                        Prerequisites
                    </h5>
                    <div class="ui attached segment">
                        <div class="ui celled list">
                        {% for p in s.prerequisites %}
                            <div class="item">
                                <div class="content">
                                    <div class="header">{{ p.name }}</div>
                                </div>
                            </div>
                        {% endfor %}
                        </div>
                    </div>
                </div>
            </div>
        {% endif %}

        <div id="success" class="ui basic modal">
          <div id="id_modal_section" class="ui green header">

          </div>
        </div>
    </div>
{# TODO: Ajax call to building table and ajax post to submit update #}
    <script>

        function dropCourse(section_id) {
            $.ajax({
                type: "POST",
                url: "/student_system/drop_course/",

                data: {
                    section_id: section_id,
                    csrfmiddlewaretoken: '{{ csrf_token }}'
                },

                success: function(){
                    $('.ui.modal').modal('show');
                    $('#id_modal_section').empty();
                    $('#id_modal_section').append("Course Dropped successfully!\n" );

                    $('#'+section_id+"_segment").empty();
                }
            })

        }
```

Back-end Implementation

- If an HTTP GET request is sent from Ajax, all courses related to the department_id are appended to a JSON data response.
- If an HTTP POST request is sent, the data sent in the request message is used to update the Course record in the database.

```
class UpdateCourse(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/search_course.html'
    is_admin = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)

        if userprofile:
            if userprofile.has_admin():
                self.is_admin = True
            else:
                redirect('/student_system/')

        if request.is_ajax():
            department_id = request.GET.get('department_id')
            department_name = request.GET.get('department_name')
            # user = User.objects.filter(Q(username=username) | Q(email=email) | Q(first_name=first_name) |
            #                            Q(last_name=last_name))
            courses = Course.objects.filter(department_id=int(department_id))
            course_array = []
            for c in courses:
                # print(c)
                data = {
                    'course_id': c.course_id,
                    'department_name': department_name,
                    'department_id': department_id,
                    'course_name': c.name,
                    'course_description': c.description,
                    'course_credits': c.credits
                }
                course_array.append(data)
            return HttpResponse(json.dumps(course_array), content_type="application/json")

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_admin': self.is_admin,
                'header_text': 'Search Course'
            },
            to_static_markup=False,
        )
        departments = Department.objects.all()
        context = {
            'rendered': rendered,
            'departments': departments
```

```
        return render(request, self.template_name, context)

    def post(self, request, *args, **kwargs):
        data = {
            'is_successful': False
        }
        if request.is_ajax():
            course_id = request.POST.get('courseID')
            course_name = request.POST.get('courseName')
            course_description = request.POST.get('course_descripti
            course_credits = request.POST.get('course_credits')
            course = Course.objects.get(pk=int(course_id))
            course.name = course_name
            course.description = course_description
            course.credits = course_credits
            course.save()
            data['is_successful'] = True
        else:
            data['is_successful'] = False
        return JsonResponse(data)
```

Create Section

Header	
Use Case ID	UC14
Use Case Version	1.0
Body	
Title	CreateSection
Actors	Admin
Input	Information Necessary to create a section
Output	Section Created.
Normal Flow	1. Admin navigates to the CreateSection options from their interface. 2. Form is displayed for admin to enter all required information to create a section. 3. Form is submitted with all required information filled out. 4. Section is created, and confirmation message is displayed.
Alternate Flows	3a. Form inputs were not filled out correctly, error message displayed, move back to step 2.
Entry Condition	Admin is logged in.
Exit Condition	Section created.

Front-end Implementation

- Admin is show various dropdowns to enter in Course relevant information to create a section. Department and building dropdowns send Ajax request to append the appropriate data to the Course and Room dropdowns.
- Admin has a button to reveal and conceal the Create Time Slot window to create Time slots that may not already appear in the dropdown.

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}
{#    {% csrf_token %}#}
    <div class="ui container">
        {{ rendered|safe }}
    </div>

    <div class="ui container">
        <form id="section-form" action="{{ url }}" method="post" class="ui form m-top2 ">
            {% csrf_token %}

            <div class="fields">
                <div class="ten wide required field">
                    <label for="id_department_id">Department</label>
                    <select class="ui fluid dropdown" name="department_id" id="id_department_id" required>
                        <option value="true">-------</option>
                        {% if departments %}
                            {% for department in departments %}
                                <option value="{{ department.department_id }}" label="{{ department.name }}">{{ department.name }}</option
                            {% endfor %}
                        {% endif %}
                    </select>
                </div>
                <div class="six wide required field">
                    <label for="id_course_id">Course</label>
                    <select class="ui fluid dropdown" name="course_id" id="id_course_id" required>
                    </select>
                </div>
            </div>

            <div class=" required field">
                <label for="id_faculty_id">Faculty</label>
                <select class="ui fluid dropdown" name="faculty_id" id="id_faculty_id" required>
{#                    <option value="true">-------</option>#}
{#                    {% if faculty %}#}
{#                        {% for f in faculty %}#}
{#                            <option value="{{ f.faculty_id }}" label="{{ f.first_name }} {{ f.last_name }}">{{ f.first_name }} {{ f.last
{#                        {% endfor %}#}
{#                    {% endif %}#}
                </select>
            </div>

            <div class=" fields">
                <div class="seven wide required field">
                    <label for="id_building_id">Building</label>
                    <select class="ui fluid dropdown" name="building_id" id="id_building_id" required>
                        <option value="true">-------</option>
```

div.ui.container

n Control

```
            </div>
            <div class="nine wide required field">
                <label for="id_room_id">Room</label>
                <select class="ui fluid dropdown" name="room_id" id="id_room_id" required>
                </select>
            </div>
        </div>
        <div class="fields">
            <div class="four wide required field">
                <label for="id_semester_id">Semester</label>
                <select class="ui fluid dropdown" name="semester_id" id="id_semester_id" required>
                    {% if semesters %}
                        {% for s in semesters %}
                            <option value="{{ s.semester_id }}" >{{ s.season }} {{ s.year }}</option>
                        {% endfor %}
                    {% endif %}
                </select>
            </div>
            <div class="four wide required field">
                <label for="id_days_id">Days</label>
                <select class="ui fluid dropdown" name="days_id" id="id_days_id" required>
                    {% if days %}
                        {% for d in days %}
                            <option value="{{ d.days_id }}" >{{ d.day_1 }} {{ d.day_2 }} {% if d.day_3 is not None %}{{ d.
                        {% endfor %}
                    {% endif %}
                </select>
            </div>
            <div class="four wide required field">
                 <label for="id_time_slot_id">Time Slot</label>
                <select class="ui fluid dropdown" name="time_slot_id" id="id_time_slot_id" required>
                    {% if time_periods %}
                        {% for p in time_periods %}
                            <option value="{{ p.period_id }}" >{{ p.start_time }}---{{ p.end_time }} </option>
                        {% endfor %}
                    {% endif %}
                </select>
             </div>
        </div>
        <div class="inline fields">
            <div id="submit_div" class="field">
                <button type="button" id="submit" class="ui button" role="button">Submit</button>
            </div>
            <div class="field">
                <button type="button" id="add_time_segment" class="ui button" role="button">Create Time Slot</button>
            </div>
        </div>
    </form>
    <div id="success"></div>
    <div id="time_segment">
```

div.ui.container

Control

```
</div>

<script>
    function getCourses(){
        removeCourses();
        {#$('#id_faculty_id').empty();#}
        var department_id = $('#id_department_id').val();
        var department_name = $('select option:selected').text();
        $.ajax({
            type: "GET",
            url: "/student_system/create_section/",
            data : {
                department_id: department_id,
                department_name: department_name
            },
            success: function(data) {
                console.log(data);
                $('#id_faculty_id').empty();
                data.course_array.forEach(function(element){
                    $('#id_course_id').append("<option value='"+element.course_id+"'>"+element.course_name+"</option>" );
                });
                data.faculty_array.forEach(function(e){
                    {#$('#id_faculty_id').empty();#}
                    $('#id_faculty_id').append("<option value='"+e.faculty_id+"'>"+e.faculty_name+"</option>" );
                })
            }
        });

    }

    function getRooms(){
        removeRooms();
        var building_id = $('#id_building_id').val();
        $.ajax({
            type: "GET",
            url: "/student_system/create_section/",
            data : {
                building_id: building_id.trim()
            },
            success: function(data) {
                console.log(data);
                data.forEach(function(element){
                    $('#id_room_id').append("<option value='"+element.room_id+"'>"+element.room_number+" : "+element.room_type+" : "+element.room_capacity+"
                });
            }
        });

    }
iv.ui.container
ontrol
```

```
function addTimeSegment(){
    $('#time_segment').append("<div class=\"ui segment\">" +
                    "           <h2 class=\"ui left floated header\">Add Time Slot</h2>" +
                    "           <div class='ui clearing divider'></div>"+
                    "           <form id='time-slot-form' action='' method='post' class='ui form m-top2'>" +
                    "              <div class='three fields'> "+
                    "                  <div class='required field'>"+
                    "                      <label for='start_time_id'>Start Time</label>"+
                    "                      <input type='time' name='start_time' id='start_time_id' required/>"+
                    "                  </div>"+
                    "                  <div class='required field'>"+
                    "                      <label for='end_time_id'>End Time</label>"+
                    "                      <input type='time' name='end_time' id='end_time_id' required/>"+
                    "                  </div>"+
                    "                  <div style='padding-top: 1.7rem;' class='field'>"+
                    "                      <button type='button' onclick='submitTimePeriod()' id='submit_time' class='ui button' role='button'>Submit</button>"+
                    "                  </div>"+
                    "              </div>"+
                    "           </form>"+
                    "       </div>");
    var time_segment = $('#add_time_segment');
    time_segment.text("Hide Time Segment");
    time_segment.click(hideTimeSegment)

}

function hideTimeSegment(){
    var time_segment = $('#add_time_segment');
    $('#time_segment').empty();
    time_segment.text("Add Time Segment");
    time_segment.click(addTimeSegment);
}

function removeRooms(){
    $('#id_room_id').empty();
}

function removeCourses() {
    $('#id_course_id').empty();
    {#$('#id_faculty_id').empty();#}
 }

function submitTimePeriod(){
     $('#time_success').empty();
    var start_time = $('#start_time_id').val();
    var end_time = $('#end_time_id').val();
    $.ajax({
```

div.ui.container

```
        var start_time = $('#start_time_id').val();
        var end_time = $('#end_time_id').val();
        $.ajax({
            type: "POST",
            url: "/student_system/create_section/time_slot/",

            data: {
                start_time: start_time,
                end_time: end_time,
                csrfmiddlewaretoken: '{{ csrf_token }}'
            },
            success: function(){
                $('#time_success').append("<div class=\"ui positive message\">\n" +
                    "  <i class=\"close icon\"></i>\n" +
                    "  <div class=\"header\">\n" +
                    "    Time Slot successfully!\n" +
                    "  </div>\n" +
                    "</div>")
            }
        })

    }

    function submitSection(event){
        event.preventDefault();
        var department = $('#id_department_id').val();
        var course = $('#id_course_id').val();
        var faculty = $('#id_faculty_id').val();
        var building = $('#id_building_id').val();
        var room = $('#id_room_id').val();
        var semester = $('#id_semester_id').val();
        var days = $('#id_days_id').val();
        var time_period = $('#id_time_slot_id').val();
        $.ajax({
            type: "POST",
            url: "/student_system/create_section/",

            data: {
                department: department,
                course: course,
                faculty: faculty,
                building: building,
                room: room,
                semester: semester,
                days: days,
                time_period: time_period,
                csrfmiddlewaretoken: '{{ csrf_token }}'
            },

            success: function(){
                $('#section-form').hide();
```

div.ui.container

Back-end Implementation

- If a HTTP GET request is sent through Ajax with a building_id inside the header, all rooms located in the building are fetched and displayed inside the front-end dropdown
- If a HTTP GET request is sent through Ajax with a department_id inside the header all courses and faculty related to that department are fetched and displayed inside the front-end dropdown.
- If a HTTP POST request is sent through Ajax a section is created with all data sent inside request. Back-end propagates back a success flag to display and error or success message on the front-end.

```
class CreateSection(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/create_section.html'
    is_admin = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)
        if userprofile:
            if userprofile.has_admin():
                self.is_admin = True
            else:
                redirect('/student_system/')

        departments = Department.objects.all()
        buildings = Building.objects.all()
        semester = Semester.objects.all()
        days = MeetingDays.objects.all()
        time_period = Period.objects.all()
        faculty = []
        for f in Faculty.objects.raw("SELECT u.first_name, u.last_name, f.faculty_id_id "
                                     "FROM registration_system_faculty AS f, auth_user as u, registration_system_userprofi
                                     "WHERE up.user_id = u.id "
                                     "AND up.id = f.faculty_id_id"):
            faculty.append({
                'first_name': f.first_name,
                'last_name': f.last_name,
                'faculty_id': f.faculty_id_id
            })

        if request.is_ajax():
            building_number = request.GET.get('building_id')
            if building_number is not None:
                rooms = Room.objects.filter(building_id__building_id=int(building_number))
                rooms_array = []
                for r in rooms:
                    data = {
                        'room_id': r.room_id,
                        'room_number': r.room_number,
                        'room_type': r.type,
                        'room_capacity': r.capacity
                    }
                    rooms_array.append(data)
                return HttpResponse(json.dumps(rooms_array), content_type="application/json")
            else:
                department_id = request.GET.get('department_id')
                department_name = request.GET.get('department_name')
```

CreateSection

```
            #                           Q(last_name=last_name))
            courses = Course.objects.filter(department_id=int(department_id))
            course_array = []
            for c in courses:
                # print(c)
                data = {
                    'course_id': c.course_id,
                    'department_name': department_name,
                    'department_id': department_id,
                    'course_name': c.name,
                    'course_description': c.description,
                    'course_credits': c.credits
                }
                course_array.append(data)
            faculty_array = []
            for f in Faculty.objects.filter(department_id=int(department_id)):
                data = {
                    'faculty_name': f.faculty_id.user.first_name + ' ' + f.faculty_id.user.last_name,
                    'faculty_id': f.faculty_id_id

                }
                faculty_array.append(data)
            data_obj = {
                'faculty_array': faculty_array,
                'course_array': course_array
            }
            return HttpResponse(json.dumps(data_obj), content_type="application/json")

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_admin': self.is_admin,
                'header_text': 'Create Section'
            },
            to_static_markup=False,
        )
        return render(request, self.template_name, {'rendered': rendered, 'departments': departments,
                                                    'buildings': buildings, 'semesters': semester, 'days': days,
                                                    'time_periods': time_period, 'faculty': faculty})

    def post(self, request, *args, **kwargs):
        data = {
            'is_successful': False
        }
        if request.is_ajax():
            department = request.POST.get('department')
```

reateSection

```
def post(self, request, *args, **kwargs):
    data = {
        'is_successful': False
    }
    if request.is_ajax():
        department = request.POST.get('department')
        course_id = request.POST.get('course')
        course = Course.objects.get(pk=int(course_id))
        faculty_id = request.POST.get('faculty')
        faculty = Faculty.objects.get(pk=int(faculty_id))
        building_id = request.POST.get('building')
        building = Building.objects.get(pk=int(building_id))
        room_id = request.POST.get('room')
        room = Room.objects.get(pk=int(room_id))
        semester_id = request.POST.get('semester')
        semester = Semester.objects.get(pk=int(semester_id))
        days_id = request.POST.get('days')
        days = MeetingDays.objects.get(pk=int(days_id))
        time_period_id = request.POST.get('time_period')
        time_period = Period.objects.get(pk=int(time_period_id))
        time_slot = None
        try:
            time_slot = TimeSlot.objects.get(days_id=days, period_id=time_period)
        except TimeSlot.DoesNotExist:
            time_slot = TimeSlot.objects.create(days_id=days, period_id=time_period)
        section = Section.objects.create(course_id=course, time_slot_id=time_slot,
                                         faculty_id=faculty, semester_id=semester, room_id=room)
        data['is_successful'] = True
    else:
        data['is_successful'] = False
    return JsonResponse(data)
```

- Back-end post route receives all form information submitted by the front-end to create a new Section Record in the database.

Update Section

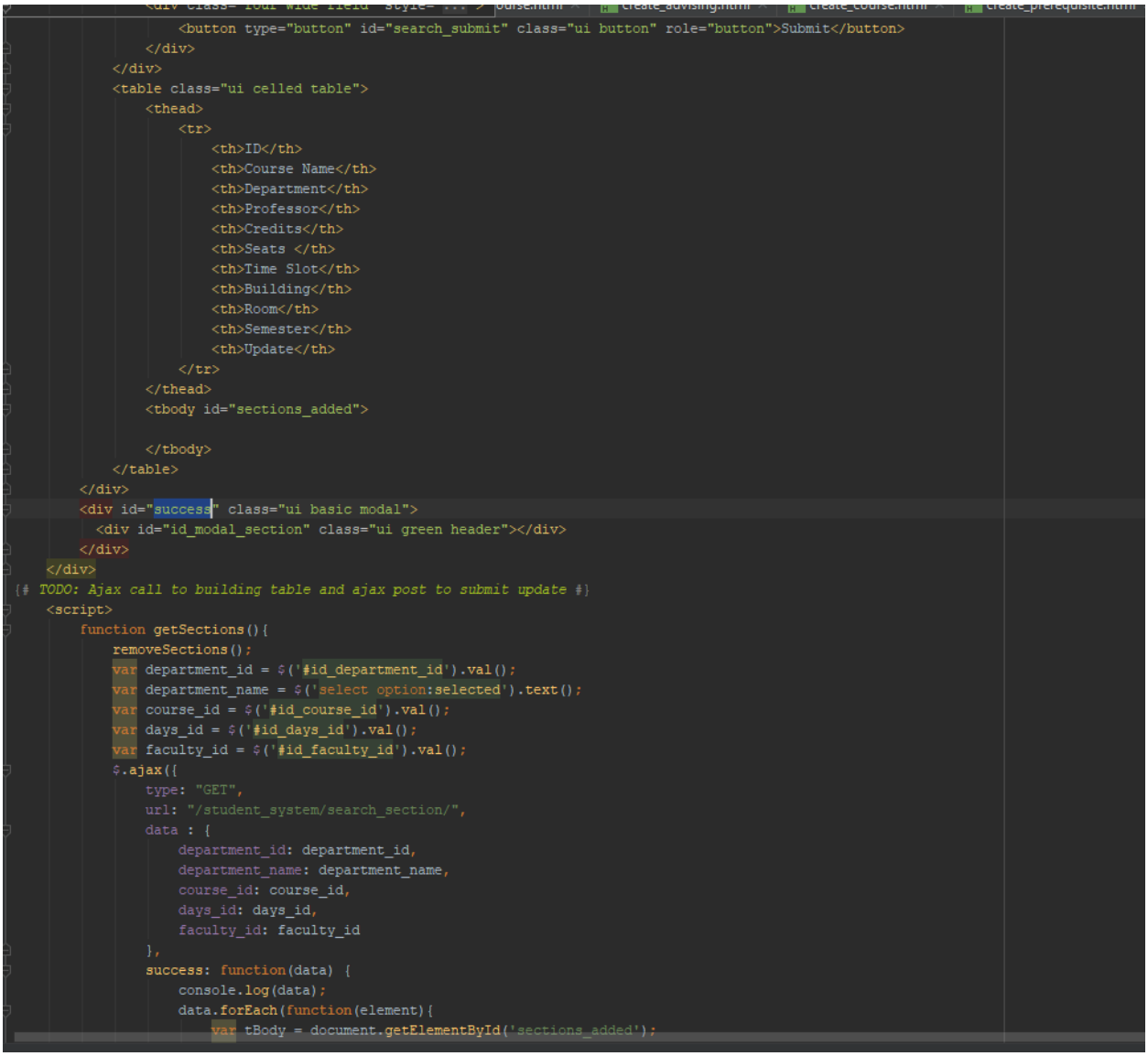
Header	
Use Case ID	UC13
Use Case Version	1.0
Body	
Title	UpdateSection
Actors	Admin
Input	Section ID
Output	Section deleted
Normal Flow	1. Admin navigates to the UpdateSection option from their administrative interface and is presented form to enter search filters for the appropriate course. 2. Admin searches for sections. 3. Sections are appended to the table rows containing inputs to update professor, time-slot, room and building. 4. Course Updated.
Alternate Flows	3a. Ongoing courses cannot be removed, error message displayed.
Entry Condition	Admin is logged in.
Exit Condition	Course/Section deleted.

Front-end Implementation

- Admin is supplied a dropdown to query sections by Department, Course, Faculty, or Meeting Days.
- Once searched all sections related to search inputs are appended to the table structure.
- Each section record in the table structure contains dropdowns to update the sections professor, time slot, building, or room. A button is also appended to the record to commit these changes.

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

     <div class="ui container">
        {{ rendered|safe }}
    </div>

    <div class="ui container m-top2 " style="...">
        <div class="ui form" id="ui-form">
            <div class="inline fields">
                <div class="field">
                    <label for="id_department_id">Department</label>
                    <select class="ui fluid dropdown" name="department_id" id="id_department_id" >
                        <option value="0">-------</option>
                        {% if departments %}
                            {% for department in departments %}
                                <option value="{{ department.department_id }}" label="{{ department.name }}">{{ department.name
                            {% endfor %}
                        {% endif %}
                    </select>
                </div>
            </div>
            <div class="inline fields">
                <div class="field">
                    <label for="id_course_id">Course</label>
                    <select class="ui fluid dropdown" name="course_id" id="id_course_id" >
                    <option value="0">-------</option>
                        {% if courses %}
                            {% for course in courses %}
                                <option value="{{ course.course_id }}" label="{{ course.name }}">{{ course.name }}</option>
                            {% endfor %}
                        {% endif %}
                    </select>
                </div>
            </div>
            <div class="fields">
                <div class="eight wide field">
                    <label for="id_faculty_id">Faculty</label>
                    <select class="ui fluid dropdown" name="faculty_id" id="id_faculty_id">
                        <option value="0">-------</option>
                        {% if faculty %}
                            {% for f in faculty %}
                                <option value="{{ f.faculty_id }}" >{{ f.first_name }} {{ f.last_name }}</option>
                            {% endfor %}
                        {% endif %}
                    </select>
                </div>
```

div.ui.container.m-top2 › div#success.ui.basic.modal



```
                    <button type="button" id="search_submit" class="ui button" role="button">Submit</button>
                </div>
            </div>
            <table class="ui celled table">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Course Name</th>
                        <th>Department</th>
                        <th>Professor</th>
                        <th>Credits</th>
                        <th>Seats </th>
                        <th>Time Slot</th>
                        <th>Building</th>
                        <th>Room</th>
                        <th>Semester</th>
                        <th>Update</th>
                    </tr>
                </thead>
                <tbody id="sections_added">

                </tbody>
            </table>
        </div>
        <div id="success" class="ui basic modal">
          <div id="id_modal_section" class="ui green header"></div>
        </div>
    </div>
{# TODO: Ajax call to building table and ajax post to submit update #}
    <script>
        function getSections(){
            removeSections();
            var department_id = $('#id_department_id').val();
            var department_name = $('select option:selected').text();
            var course_id = $('#id_course_id').val();
            var days_id = $('#id_days_id').val();
            var faculty_id = $('#id_faculty_id').val();
            $.ajax({
                type: "GET",
                url: "/student_system/search_section/",
                data : {
                    department_id: department_id,
                    department_name: department_name,
                    course_id: course_id,
                    days_id: days_id,
                    faculty_id: faculty_id
                },
                success: function(data) {
                    console.log(data);
                    data.forEach(function(element){
                        var tBody = document.getElementById('sections_added');
```

```
                    var courseNameCell = newRow.insertCell(1);
                    var departmentCell = newRow.insertCell(2);
                    var facultyCell = newRow.insertCell(3);
                    var creditsCell = newRow.insertCell(4);
                    var seatsCell = newRow.insertCell(5);
                    var timeSlotCell = newRow.insertCell(6);
                    var buildingCell = newRow.insertCell(7);
                    var roomNumberCell = newRow.insertCell(8);
                    var semesterCell = newRow.insertCell(9);
                    var updateCell = newRow.insertCell(10);
# editing faculty, meeeting days, time period, building, and room number  #}
                    var sectionIdText = document.createTextNode(element.section_id);
                    var courseNameText = document.createTextNode(element.course_name);
                    var departmentText = document.createTextNode(element.course_department);
                    var creditsText = document.createTextNode(element.credits);
                    var seatsText = $("<ul style='list-style:none; padding-left:0px'><li><span style='font-weight:bold'>Taken:</span>"+element.seat
                    sectionIdCell.appendChild(sectionIdText);
                    courseNameCell.appendChild(courseNameText);
                    departmentCell.appendChild(departmentText);
                    creditsCell.appendChild(creditsText);
                    semesterCell.insertAdjacentHTML('afterbegin',
                        "<ul style='list-style:none; padding-left:0px'><li>"+element.semester_season+"</li><li>"+element.semester_year+"</li><li>"+
                    );
                    $('#'+element.section_id + '_row').find('td:eq(5)').append(seatsText);
                    facultyCell.insertAdjacentHTML('afterbegin',
                        '<select id="id_faculty_id_'+element.section_id+'" class="ui input" style="min-width: 9rem;">' +
                        '</select>' );
                    var facultySelect = facultyCell.firstChild;
                    element.faculty_array.forEach(function(el){
                        {#console.log("here");#}
                        if( el.faculty_id === element.faculty_id ){
                            facultySelect.insertAdjacentHTML('afterbegin',
                                '<option value="' + el.faculty_id + '" selected >' + el.full_name + '</option>')
                        } else {
                            facultySelect.insertAdjacentHTML('afterbegin',
                                '<option value="' + el.faculty_id + '" >' + el.full_name + '</option>');
                        }
                    });
                    timeSlotCell.insertAdjacentHTML('afterbegin',
                        '<select id="id_time_period_id_'+element.section_id+'" class="ui input" style="min-width: 12rem;">' +
                        '</select>' );
                    var timeSelect = timeSlotCell.firstChild;
                    element.time_periods_array.forEach(function(el){
                        {#console.log("here");#}
                        console.log(el.time_period_id);
                        console.log(element.time_slot_id);
                        if( el.time_period_id === element.time_period_id){
                            timeSelect.insertAdjacentHTML('afterbegin',
                                '<option value="'+el.time_period_id+'" selected>'+el.time_range+'</option>');
                        } else {
                            timeSelect.insertAdjacentHTML('afterbegin',
```

cript › getSections() › data

ontrol

```
    }

    function removeSections() {
        $('#sections_added').empty();
    }

    function updateSection(section_id) {
        var faculty = $('#id_faculty_id_'+section_id).val();
        var time_period = $('#id_time_period_id_'+section_id).val();
        var days = $('#id_days_id_'+section_id).val();
        var building = $('#id_building_id_'+section_id).val();
        var room = $('#id_room_id_'+section_id).val();
        console.log(days);
        console.log(time_period);
        $.ajax({
            type: "POST",
            url: "/student_system/search_section/",

            data: {
                faculty: faculty,
                time_period: time_period,
                days: days,
                building: building,
                section_id: section_id,
                room: room,
                csrfmiddlewaretoken: '{{ csrf_token }}'
            },

            success: function(){
                $('.ui.modal').modal('show');
                $('#id_modal_section').empty();
                $('#id_modal_section').append("Section updated successfully!\n" );

            }
        })

    }

    $('#search_submit').click(getSections)


</script>

{% endblock %}
```

Back-end Implementation

- If an HTTP GET Request is sent from an Ajax request, back-end uses conditional statements to deduce what sections to fetch from the database and send back to the front-end as a JSON data response.
- If an HTTP POST Request is sent, form inputs are extracted and used to update the section record in the database.

```
class UpdateSection(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/search_section.html'
    is_admin = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)

        if userprofile:
            if userprofile.has_admin():
                self.is_admin = True
            else:
                redirect('/student_system/')

        if request.is_ajax():
            department_id = request.GET.get('department_id')
            department_name = request.GET.get('department_name')
            course_id = request.GET.get('course_id')
            days_id = request.GET.get('days_id')
            faculty_id = request.GET.get('faculty_id')
            section = None
            # print(department_id)

            if (course_id is not None and int(course_id) != 0) and (faculty_id is not None and int(faculty_id) != 0) and (days_id is not None and int(days_id) !=0):
                section = Section.objects.filter(faculty_id=int(faculty_id), time_slot_id__days_id=int(days_id),
                                                 course_id=int(course_id))
            if (faculty_id is not None and int(faculty_id) != 0) and (days_id is not None and int(days_id) != 0):
                section = Section.objects.filter(faculty_id=int(faculty_id), time_slot_id__days_id=int(days_id))
            if (department_id is not None and int(department_id) != 0) and (course_id is not None and int(course_id) != 0):
                section = Section.objects.filter(course_id__department_id_id=int(department_id),
                                                 course_id=int(course_id))
            elif (department_id is not None and int(department_id) != 0) and (faculty_id is not None and int(faculty_id) != 0):
                section = Section.objects.filter(course_id__department_id_id=int(department_id),
                                                 faculty_id=int(faculty_id))
            elif (department_id is not None and int(department_id) != 0) and (days_id is not None and int(days_id) != 0):
                section = Section.objects.filter(course_id__department_id_id=int(department_id),
                                                 time_slot_id__days_id=int(days_id))
            elif (course_id is not None and int(course_id) != 0) and (faculty_id is not None and int(faculty_id) != 0):
                section = Section.objects.filter(course_id=int(course_id),
                                                 faculty_id=int(faculty_id))
            elif (course_id is not None and int(course_id) != 0) and (days_id is not None and int(days_id) != 0):
                section = Section.objects.filter(course_id=int(course_id),
                                                 time_slot_id__days_id=int(days_id))
            elif department_id is not None and int(department_id) != 0:
                section = Section.objects.filter(course_id__department_id_id=int(department_id))
            elif course_id is not None and int(course_id) != 0:
                section = Section.objects.filter(course_id=int(course_id))
```

```
        elif days_id is not None and int(days_id) != 0:
            section = Section.objects.filter(time_slot_id__days_id=int(days_id))
        elif faculty_id is not None and int(faculty_id) != 0:
            section = Section.objects.filter(faculty_id=int(faculty_id))

        faculty = []
        for f in Faculty.objects.raw("SELECT u.first_name, u.last_name, f.faculty_id_id "
                                     "FROM registration_system_faculty AS f, auth_user as u, registration_system_userprofile as up "
                                     "WHERE up.user_id = u.id "
                                     "AND up.id = f.faculty_id_id"):
            faculty.append({
                'first_name': f.first_name,
                'last_name': f.last_name,
                'full_name': f.first_name + " " + f.last_name,
                'faculty_id': f.faculty_id_id
            })
        departments = []
        for d in Department.objects.all():
            departments.append({
                'department_id': d.department_id,
                'department_name': d.name
            })
        time_periods = []
        for t in Period.objects.all():
            time_periods.append({
                'time_period_id': t.period_id,
                'time_range': t.start_time.strftime('%H:%M %p') + ' ' + t.end_time.strftime('%H:%M %p')
            })
        meeting_days = []
        for m in MeetingDays.objects.all():
            if m.day_3:
                meeting_days.append({
                    'days_id': m.days_id,
                    'day_1': m.day_1,
                    'day_2': m.day_2,
                    'day_3': m.day_3,
                    'day_range': m.day_1 + " " + m.day_2 + " " + m.day_3
                })
            elif m.day_2 and m.day_3 is None:
                meeting_days.append({
                    'days_id': m.days_id,
                    'day_1': m.day_1,
                    'day_2': m.day_2,
                    'day_range': m.day_1 + " " + m.day_2
                })
            elif m.day_1 and m.day_2 is None:
                meeting_days.append({
                    'days_id': m.days_id,
```

UpdateSection

rol

assistance. (42 minutes ago)

```
        for arrgh in Room.objects.all():
            rooms.append({
                'rooms_id': arrgh.room_id,
                'room_number': arrgh.room_number
            })

        section_array = []
        for s in section:
            # print(s)
            # print("here\n")
            data = {
                'faculty_array': faculty,
                'departments_array': departments,
                'time_periods_array': time_periods,
                'meeting_days_array': meeting_days,
                'buildings_array': buildings,
                'semester_id': s.semester_id_id,
                'semester_year': s.semester_id.year,
                'semester_season': s.semester_id.season,
                'semester_status': s.semester_id.status,
                'rooms_array': rooms,
                'section_id': s.section_id,
                'course_id': s.course_id.course_id,
                'course_department': s.course_id.department_id.name,
                'course_name': s.course_id.name,
                'faculty_id': s.faculty_id.faculty_id_id,
                'faculty_name': s.faculty_id.faculty_id.user.first_name + " " + s.faculty_id.faculty_id.user.last_name,
                'credits': s.course_id.credits,
                'seats_taken': s.seats_taken,
                'seating_capacity': s.room_id.capacity,
                'time_slot_id': s.time_slot_id.time_slot_id,
                'time_period_id': s.time_slot_id.period_id.period_id,
                'time_period_range': s.time_slot_id.period_id.start_time.strftime('%H:%M %p')
                                    + " " + s.time_slot_id.period_id.end_time.strftime('%H:%M %p'),
                'meeting_days_id': s.time_slot_id.days_id.days_id,
                'meeting_days': s.time_slot_id.days_id.day_1 + s.time_slot_id.days_id.day_2,
                'building_id': s.room_id.building_id.building_id,
                'building_name': s.room_id.building_id.name,
                'room_id': s.room_id.room_id,
                'room_number': s.room_id.room_number
            }
            section_array.append(data)

        return HttpResponse(json.dumps(section_array), content_type="application/json")

    rendered = render_component(
        os.path.join(os.getcwd(), 'registration_system', 'static',
```

UpdateSection

```
        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_admin': self.is_admin,
                'header_text': 'Update Section'
            },
            to_static_markup=False,
        )
        departments = Department.objects.all()
        days = MeetingDays.objects.all()
        course = Course.objects.all()
        faculty = []
        for f in Faculty.objects.raw("SELECT u.first_name, u.last_name, f.faculty_id_id "
                                     "FROM registration_system_faculty AS f, auth_user as u, registration_system_userprofile
                                     "WHERE up.user_id = u.id "
                                     "AND up.id = f.faculty_id_id"):
            faculty.append({
                'first_name': f.first_name,
                'last_name': f.last_name,
                'faculty_id': f.faculty_id_id
            })

        context = {
            'rendered': rendered,
            'departments': departments,
            'days': days,
            'faculty': faculty,
            'courses': course
        }
        return render(request, self.template_name, context)

    def post(self, request, *args, **kwargs):
        data = {
            'is_successful': False
        }
        if request.is_ajax():
            faculty_id = request.POST.get('faculty')
            faculty = Faculty.objects.get(pk=int(faculty_id))
            time_period = request.POST.get('time_period')
            days = request.POST.get('days')
            # print(time_period)
            # print(days)
            try:
                time_slot = TimeSlot.objects.get(days_id=int(days), period_id=int(time_period))
            except TimeSlot.DoesNotExist:
                time_slot = TimeSlot.objects.create(days_id_id=int(days), period_id=Period.objects.get(pk=int(time_period)))
```

UpdateSection › post() › if request.is_ajax()

```
        building_id = request.POST.get('building')
        building = Building.objects.get(pk=int(building_id))
        room_id = request.POST.get('room')
        room = Room.objects.get(pk=int(room_id))
        section_id = request.POST.get('section_id')
        section = Section.objects.get(pk=int(section_id))
        section.faculty_id = faculty
        section.time_slot_id = time_slot
        section.building_id = building
        section.room_id = room
        section.save()
        data['is_successful'] = True
    else:
        data['is_successful'] = False
    return JsonResponse(data)
```

Create User

Header	
Use Case ID	UC15
Use Case Version	1.0
Body	
Title	CreateUser
Actors	Admin
Normal Flow	1. Admin navigates to the CreateUser menu item, Form is displayed ot input Username, password, email, first name, last name, and user type. 2. User type is chosen 3. Necessary inputs for user type are displayed, Admin fills in and submits the form. 4. User is successfully created.
Alternate Flows	6 a. Student is chosen a2. Birthdate, Student Type, and Credits input displayed 6 b. Faculty is chosen B2. Department and Faculty type input are displayed.
Entry Condition	Prerequisite: Admin must be logged in and be in the ViewMasterSchedule.

Front-end Implementation

- Form is displayed to admin, required input fields include: Username, password, first name, last name, and user type.
- Depending on which user type is chosen from the dropdown extra inputs may appear. If Admin or Researcher is chosen no extra inputs need to be filled out.
- When a student is chosen admins must pick which type, enter their birth date, and enter credits if transfer credits are applicable.
- When a faculty is chosen, admin must choose the faculty type and department their assigned to.

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}
{#    {% csrf_token %}#}
    <div class="ui container">
        {{ rendered|safe }}
    </div>

    <div class="ui container">
        <form id="user-form" action="{{ url }}" method="post" class="ui form m-top2 ">
            {% csrf_token %}

            <div class="three fields">
                <div class="required field">
                    <label for="id_username">Username</label>
                    <input type="text" name="username" maxlength="150" placeholder="Enter Username..." id="id_username" r
                </div>
                <div class="required field">
                    <label for="id_password">Password</label>
                    <input type="password" name="password" id="id_password" placeholder="Enter Password..." required/>
                </div>
                <div class="required field">
                    <label for="id_email">Email</label>
                    <input autocomplete="email" type="text" name="email" maxlength="254" placeholder="Enter Email..." id=
                </div>
            </div>
             <div class="two fields">
                <div class=" required field">
                    <label for="id_first_name">First Name</label>
                    <input autocomplete="given-name" type="text" name="first_name" maxlength="30" placeholder="Enter Firs
                </div>
                 <div class=" required field">
                     <label for="id_last_name">Last Name</label>
                     <input autocomplete="family-name" type="text" name="last_name" maxlength="150" placeholder="Enter La
                 </div>
             </div>

            <div class="inline fields">
                <div class="required field">
                    <label for="id_user_type"></label>
                    <select class="ui fluid dropdown" name="user_type" id="id_user_type" required>
                        <option value="true">--------</option>
                        <option value="A">Admin</option>
                        <option value="F">Faculty</option>
                        <option value="S">Student</option>
                        <option value="R">Researcher</option>
                    </select>
                </div>
```

script › submitUser() › success()

```
        <div id="success"></div>
    </div>

    <script>
        function getUserTypeForm(){
            var user_type = document.getElementById('id_user_type');
            var user_type_value = user_type.options[user_type.selectedIndex].value;
            switch( user_type_value){
                case 'A':
                case 'R':
                    $('#user_type_form').empty();
                    break;
                case 'F':
                    displayFacultyForm();
                    break;
                case 'S':
                    displayStudentForm();
                    break;

            }
        }

        function displayStudentForm(){
            $('#user_type_form').empty();
            $('#user_type_form').append("<div class='three fields'>" +
                                            "<div class='required field'>" +
                                                "<label for='id_date_of_birth'>Date of Birth</label>" +
                                                "<input id='id_date_of_birth' type='date' name='date_of_bir
                                            "</div>" +
                                            "<div class='required field'>" +
                                                "<label for='id_student_type'>Student Type</label>" +
                                                "<select class='ui fluid dropdown' name='student_type' id='
                                                    "<option value='F'>Full Time</option>"+
                                                    "<option value='P'>Part Time</option>"+
                                                "</select>"+
                                            "</div>" +
                                            "<div class='field'>"+
                                                "<label for='id_credits'>Credits</label>"+
                                                "<input id='id_credits' name='credits' type='number' step='
                                            "</div>"+
                                        "</div> ");
        }

        function displayFacultyForm() {
            $('#user_type_form').empty();
            $('#user_type_form').append("<div class='two fields'>" +
                                            "<div id='add_departments' class='required field'>" +
                                                "<label for='id_department_id'>Department</label>" +
                                                "<select class='ui fluid dropdown' name='department_id' id=
```

```
                                "</div> ");
        $.ajax({
            type: "GET",
            url: "/student_system/create_user/departments/",

            success: function(data) {
                console.log(data);
                data.forEach(function(element){
                    $('#id_department_id').append("<option value="+element.department_id+">"+element.department_name+"</opt
                });
            }
        });

    }

    function submitUser(event){
        event.preventDefault();
        var username = $('#id_username').val();
        var password = $('#id_password').val();
        var first_name = $('#id_first_name').val();
        var last_name = $('#id_last_name').val();
        var user_type = $('#id_user_type').val();
        var date_of_birth, student_type, credits, department_id, faculty_type = ''
        if( user_type.trim() === 'S'){
            date_of_birth = $('#id_date_of_birth').val();
            student_type = $('#id_student_type').val();
            credits = $('#id_credits').val() ? $('#id_credits').val() : false;
        } else if( user_type.trim() === 'F'){
            department_id = $('#id_department_id').val();
            faculty_type = $('#id_faculty_type').val();
        }
        $.ajax({
            type: "POST",
            url: "/student_system/create_user/",

            data: $('#user-form').serialize(),

            success: function(){
                $('#user-form').hide();
                $('#success').append("<div class=\"ui positive message\">\n" +
                    "  <div class=\"header\">\n" +
                    "    User created successfully!\n" +
                    "  </div>\n" +
                    "</div>")
            }
        })
    }


    $('#submit').click(submitUser);
```

script › submitUser() › success()

Back-end Implementation

- If an HTTP POST request is sent, back-end utilizes conditional statements to deduce which type of account (Admin, Full-time/Part-time Student, Full-time/Part-time Faculty, and Researcher) to create determined the input sent from the front-end.

```
class CreateUser(LoginRequiredMixin, generic.View):
    user_form_class = CreateUserParentForm
    user_profile_form_class = CreateUserProfileForm
    student_form_class = CreateStudentForm
    faculty_form_class = CreateFacultyForm
    template_name = 'registration_system/create_user.html'
    is_admin = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)

        if userprofile:
            if userprofile.has_admin():
                self.is_admin = True
            else:
                redirect('/student_system/')

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_admin': self.is_admin,
                'header_text': 'Create User'
            },
            to_static_markup=False,
        )

        context = {
            'rendered': rendered
        }

        return render(request, self.template_name, context)

    def post(self, request, *args, **kwargs):
        data = {
            'is_successful': False
        }
        if request.POST:

            user_form = self.user_form_class(request.POST, instance=User())
            user_profile_form = self.user_profile_form_class(request.POST, instance=UserProfile())
            # print(user_form.errors)
            # print(user_profile_form.errors)
            if user_form.is_valid() and user_profile_form.is_valid():
                username = user_form.cleaned_data['username']
                password = user_form.cleaned_data['password']
```

```
            first_name = user_form.cleaned_data['first_name']
            last_name = user_form.cleaned_data['last_name']
            email = user_form.cleaned_data['email']
            user_type = user_profile_form.cleaned_data['user_type']
            user = User.objects.create_user(username=username, password=password, first_name=first_name,
                                            last_name=last_name, email=email)
            user_profile = user.userprofile
            if user_type == 'A':
                user_profile.user_type = 'A'
                user_profile.save()
                admin = Admin.objects.create(admin_id=user_profile)
                data['is_successful'] = True
                # print(admin)
            elif user_type == 'R':
                user_profile.user_type = 'R'
                user_profile.save()
                researcher = Researcher.objects.create(researcher_id=user_profile)
                data['is_successful'] = True
            elif user_type == 'S':
                student_form = self.student_form_class(request.POST, instance=Student())
                if student_form.is_valid():
                    date_of_birth = student_form.cleaned_data['date_of_birth']
                    student_type = student_form.cleaned_data['student_type']
                    credits_variable = None
                    if request.POST.get("credits") != 0:
                        credits_variable = request.POST.get("credits")
                        user_profile.user_type = 'S'
                        user_profile.save()
                        student = Student.objects.create(student_id=user_profile, date_of_birth=date_of_birth,
                                                         student_type=student_type.strip())
                        if student.student_type == 'F':
                            student.fulltimestudent.credits = int(credits_variable)
                            student.fulltimestudent.save()
                        else:
                            student.parttimestudent.credits = int(credits_variable)
                            student.parttimestudent.save()
                    else:
                        user_profile.user_type = 'S'
                        user_profile.save()
                        student = Student.objects.create(student_id=user_profile, date_of_birth=date_of_birth,
                                                         student_type=student_type.strip())
                    data['is_successful'] = True
            elif user_type == 'F':
                faculty_form = self.faculty_form_class(request.POST, instance=Faculty())
                if faculty_form.is_valid():
                    department_id = faculty_form.cleaned_data['department_id']
                    faculty_type = faculty_form.cleaned_data['faculty_type']
                    user_profile.user_type = 'F'
```

```
            user_profile.save()
            faculty = Faculty.objects.create(faculty_id=user_profile, department_id=department_id,
                                             faculty_type=faculty_type.strip())
            data['is_successful'] = True
            # print(faculty)
    # data['errors'] = self.user_form_class.errors if self.user_form_class.errors else None
    # data['errors'] = self.user_profile_form_class.errors if self.user_profile_form_class.errors else None
    # data['errors'] = self.faculty_form_class.errors if self.faculty_form_class.errors else None
    # data['errors'] = self.student_form_class.errors if self.student_form_class.errors else None
    return JsonResponse(data)
```

Update User

Header	
Use Case ID	UC16
Use Case Version	1.0
Body	
Title	Update User
Actors	Admin
Normal Flow	1. Admin navigates to the Update User menu item, form is displayed to input Username, Email, User Type, First Name, or Last Name. Only one has to be filled in. 2. The system will fetch all users related to search query and append them to table structure, each row record has a button appended to remove the user from the system. 3. Admin Chooses to remove the searched user from the system 4. System removes user and all user associations from the system. 5.
Alternate Flows	N/A
Entry Condition	Admin must be logged in and be in the ViewMasterSchedule

Front-end Implementation

- Form is displayed to admin, required input search fields. Admins may search by username, email, user type, first name or last name.
- All results are appended to the table displayed on the front end.
- Upon querying admin may remove the user and all associations from the system by pressing the remove button on the row of the user.

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

     <div class="ui container">
        {{ rendered|safe }}
    </div>

    <div class="ui container m-top2">
        <div class="ui form" id="ui-form">
            <div class="three fields">
                <div class="field">
                    <label for="username_id">Username</label>
                    <input id="username_id"   placeholder="Search by username..." type="text" />
                </div>
                <div class="field">
                    <label for="email_id">Email</label>
                    <input id="email_id" autocomplete="email" placeholder="Search by email..." type="text"/>
                </div>
                <div class="field">
                    <label for="user_type_id">User Type</label>
                    <select class="ui fluid dropdown" name="user_type_id" id="user_type_id" required>
                        <option value="0">-------</option>
                        <option value="faculty" label="Faculty">Faculty</option>
                        <option value="student" label="Student">Student</option>
                        <option value="admin" label="Admin">Admin</option>
                        <option value="researcher" label="Researcher">Researcher</option>
                    </select>
                </div>
            </div>
        {{ csrf }}
             <div class="two fields">
                <div class="field">
                    <label for="first_name_id">First Name</label>
                    <input id="first_name_id"   placeholder="Search by first name..." type="text" />
                </div>
                <div class="field">
                    <label for="last_name_id">Last Name</label>
                    <input id="last_name_id" placeholder="Search by last name..." type="text"/>
                </div>
            </div>
            <div class="display-inline ">
                <button class="ui primary button" id="search" type="button">Search</button>
                <button class="ui negative button" id="clear" type="button">Clear</button>
            </div>
            <table class="ui celled table">
                <thead>
                    <tr>
```

script › deleteStudent() › success()

```
<script>
    function addStudent(){
        console.log("here");
        removeStudents();
        var first_name = $('#first_name_id').val();
        var last_name = $('#last_name_id').val();
        var username = $('#username_id').val();
        var email = $('#email_id').val();
        var user_type = $("#user_type_id").val();
        $.ajax({
            type: "GET",
            url: "/student_system/search_user/",
            data : {
                first_name: first_name,
                last_name: last_name,
                username: username,
                email: email,
                user_type: user_type
            },
            success: function(data) {
                console.log(data);
                data.forEach(function(element){
                    var tBody = document.getElementById('students_added');
                    var newRow = tBody.insertRow(tBody.rows.length);
                    newRow.id = element.user_id + '_row';
                    var userIdCell = newRow.insertCell(0);
                    var usernameCell = newRow.insertCell(1);
                    var emailCell = newRow.insertCell(2);
                    var firstNameCell = newRow.insertCell(3);
                    var lastNameCell = newRow.insertCell(4);
                    var userTypeCell = newRow.insertCell(5);
                    var removeCell = newRow.insertCell(6);
                    var userIdText = document.createTextNode(element.user_id);
                    var usernameText = document.createTextNode(element.username);
                    var emailText = document.createTextNode(element.email);
                    var firstNameText = document.createTextNode(element.first_name);
                    var lastNameText = document.createTextNode(element.last_name);
                    var userTypeText = document.createTextNode(element.user_type);
                    userIdCell.appendChild(userIdText);
                    usernameCell.appendChild(usernameText);
                    emailCell.appendChild(emailText);
                    firstNameCell.appendChild(firstNameText);
                    lastNameCell.appendChild(lastNameText);

                    if(element.department_name)
                        userTypeCell.insertAdjacentHTML('afterbegin', '<div><strong>Department:</strong> '+element.department_name+'<br/>'+element.user_type+"<
                    else
                        userTypeCell.appendChild(userTypeText);
                    removeCell.insertAdjacentHTML('afterbegin', '<button type="button" onclick="deleteStudent(this.id)" class="ui negative button" id="'+elemen
                });
```

script › deleteStudent() › success()

```
    function removeStudents() {
        $('#students_added').empty();
    }

    function deleteStudent(user_id) {
        $('#success').empty();
        $.ajax({
            type: "POST",
            url: "/student_system/search_user/",

            data: {
                userID: user_id,
                csrfmiddlewaretoken: '{{ csrf_token }}'
            },

            success: function(){
                $('#'+user_id+'_row').remove();
                $('#success').append("<div class=\"ui positive message\">\n" +
                                     "  <div class=\"header\">\n" +
                                     "    User Removed successfully!\n" +
                                     "  </div>\n" +
                                     "</div>")
            }
        })

    }

    $('#search').click(addStudent);
    $('#clear').click(removeStudents);


 </script>

endblock %}
```

Back-end Implementation

- If an HTTP Get request is sent from Ajax, Users related to the search inputs are fetched form the database and sent back in a JSON data response to the front-end to be appended to the table structure created.
- If an HTTP Post request is sent from Ajax, User_id is used to remove the user from the database.

```python
class UpdateUser(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/search_user.html'
    is_admin = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)

        if userprofile:
            if userprofile.has_admin():
                self.is_admin = True
            else:
                redirect('/student_system/')

        if request.is_ajax():
            first_name = request.GET.get('first_name')
            last_name = request.GET.get('last_name')
            email = request.GET.get('email')
            username = request.GET.get('username')
            user_type = request.GET.get('user_type')
            # user = User.objects.filter(Q(username=username) | Q(email=email) | Q(first_name=first_name) |
            #                            Q(last_name=last_name))
            if first_name and last_name:
                user = User.objects.filter(first_name=first_name, last_name=last_name)
            if username:
                user = User.objects.filter(username=username)
            elif email:
                user = User.objects.filter(email=email)
            elif user_type and user_type != '0':
                if user_type == 'faculty':
                    user = User.objects.filter(userprofile__user_type='F')
                elif user_type == 'admin':
                    user = User.objects.filter(userprofile__user_type='A')
                elif user_type == 'student':
                    user = User.objects.filter(userprofile__user_type='S')
                elif user_type == 'researcher':
                    user = User.objects.filter(userprofile__user_type='R')
            elif first_name:
                user = User.objects.filter(first_name=first_name)
            elif last_name:
                user = User.objects.filter(last_name=last_name)
            user_array = []
            for u in user:
                # print(u)
                user_profile = UserProfile.objects.get(user_id=u.id)
                data = {
                    'user_id': u.id,
```

UpdateUser

```
                user_profile = UserProfile.objects.get(user_id=u.id)
                data = {
                    'user_id': u.id,
                    'username': u.username,
                    'email': u.email,
                    'first_name': u.first_name,
                    'last_name': u.last_name,
                    'user_type': user_profile.user_type
                }
                if user_profile.has_faculty():
                    data['department_name'] = user_profile.faculty.department_id.name
                user_array.append(data)
            return HttpResponse(json.dumps(user_array), content_type="application/json")

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_admin': self.is_admin,
                'header_text': 'Search User'
            },
            to_static_markup=False,
        )

        context = {
            'rendered': rendered
        }

        return render(request, self.template_name, context)

    def post(self, request, *args, **kwargs):
        data = {
            'is_successful': False
        }
        if request.is_ajax():
            user_id = request.POST.get('userID')
            user = User.objects.get(pk=int(user_id))
            user.delete()
            data['is_successful'] = True
        else:
            data['is_successful'] = False
        return JsonResponse(data)
```

Change Semester

- Used to change the semester status to one of the following:
  - Open Grading
  - Closed Grading
  - Open Registration
  - Closed Registration
- When registration is closed all students, who do not fulfill prerequisites are dropped from the courses that require them.
- If registration is open, students can register for courses in the Semester year that has a status of OPEN REGISTRATION
- When grading is closed all grades are submitted and cannot be changed by faculty thereafter.
- If grading is open grades can be submitted and updated by faculty for sections taught.

Front-end Implementation

- Two dropped downs are displayed to the user to choose the Semester/Year and status to apply.
- Upon submit a success message is displayed to the user.

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

    <div class="ui container">
        {{ rendered|safe }}
    </div>

    <div class="ui container m-top2">
        <div class="ui form" id="ui-form">
        {{ csrf }}
            <div class="two fields">
                <div class="field">
                    <label for="semester_id">Semesters</label>
                    <select id="semester_id" class="ui fluid dropdown">
                        <option value="0">--------</option>
                        {% if semesters %}
                            {% for s in semesters %}
                                <option value="{{ s.semester_id }}" >{{ s.season }}--{{ s.year }}--{{ s.status }}</option>
                            {% endfor %}
                        {% endif %}
                    </select>
                </div>
                <div class="field">
                    <label for="staus_name_id">Status</label>
                    <select id="status_name_id" class="ui fluid dropdown">
                        <option value="OPEN_GRADING">OPEN GRADING</option>
                        <option value="CLOSE">CLOSE GRADING</option>
                        <option value="OPEN_REGISTRATION">OPEN REGISTRATION</option>
                        <option value="CLOSE">CLOSE REGISTRATION</option>
                    </select>
                </div>
            </div>
            <div class="display-inline ">
                <button class="ui primary button" id="submit" type="button">Submit</button>
            </div>
        </div>
        <div id="success">

        </div>
    </div>

    <script>

        function changeStatus() {
            $('#success').empty();
            var semester_id = $('#semester_id').val();
            var status = $('#status_name_id').val();
```

```
            $.ajax({
                type: "POST",
                url: "/student_system/change_semester/",

                data: {
                    semester_id: semester_id,
                    status: status,
                    csrfmiddlewaretoken: '{{ csrf_token }}'
                },

                success: function(data){
                    if( data.is_successful === true && data.students_disenrolled){
                        $('#success').append("<div class=\"ui positive message\">\n" +
                            "  <div id='success_content' class=\"header\">\n" +
                            "    Semester Status updated successfully!\n" +
                            "  </div>\n" +
                            "</div>");
                        data.students_disenrolled.forEach(function(e){
                            $('#success_content').append("Student "+e.student_name+" was removed from the enrollment in section " + e.section_id_c
                        });
                    }

                }
            })

        }

        $('#submit').click(changeStatus);


    </script>

{% endblock %}
```

Back-end Implementation

```
class ChangeSemesterStatus(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/change_semester.html'
    is_admin = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)

        if userprofile:
            if userprofile.has_admin():
                self.is_admin = True
            else:
                return redirect('/student_system/')

        semesters = Semester.objects.all()

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_admin': self.is_admin,
                'header_text': 'Create Advising'
            },
            to_static_markup=False,
        )
        context = {
            'rendered': rendered,
            'semesters': semesters
        }

        return render(request, self.template_name, context)

    # TODO: Add more logic to remove students who do not fulfill prerequisites if closing from open_registration
    def post(self, request, *args, **kwargs):
        data = {
            'is_successful': False
        }
        if request.is_ajax():
            semester_id = request.POST.get('semester_id')
            status = request.POST.get('status')
            semester = Semester.objects.get(pk=int(semester_id))
            student_disenrolled = []
            if semester.status == 'OPEN_REGISTRATION':
                if status == 'CLOSE':
                    for e in Enrollment.objects.filter(section_id__semester_id_id=int(semester_id)):
                        try:
                            prerequisites = Prerequisite.objects.filter(course_id=e.section_id.course_id)
                            enrolled_student = Student.objects.get(student_id=e.student_id)
```

```
                        try:
                            prerequisites = Prerequisite.objects.filter(course_id=e.section_id.course_id)
                            enrolled_student = Student.objects.get(student_id=e.student_id)
                            for se in Enrollment.objects.filter(student_id=enrolled_student):
                                for p in prerequisites:
                                    if se.section_id.course_id_id == p.course_required_id_id:
                                        if se.grade in ['A', 'A-', 'B', 'B-', 'C']:
                                            continue
                                        else:
                                            student_disenrolled.append({
                                                'student_name': se.student_id.student_id.user.first_name + ' '
                                                + se.student_id.student_id.user.last_name,
                                                'class_dropped': e.section_id.course_id.name,
                                                'section_id_dropped': e.section_id_id
                                            })
                                            se.delete()
                                    else:
                                        student_disenrolled.append({
                                            'student_name': se.student_id.student_id.user.first_name + ' '
                                            + se.student_id.student_id.user.last_name,
                                            'class_dropped': e.section_id.course_id.name,
                                            'section_id_dropped': e.section_id_id
                                        })
                                        se.delete()
                        except Prerequisite.DoesNotExist:
                            continue
                semester.status = status
                semester.save()
                data['is_successful'] = True
                data['students_disenrolled'] = student_disenrolled
        else:
            data['is_successful'] = False
        return JsonResponse(data)
```

Faculty Implementation:

ViewFacultySchedule

Header	
Use Case ID	UC17
Use Case Version	1.0
Body	
Title	ViewFacultySchedule
Actors	Faculty
Normal Flow	1. Faculty will log in the web-based registration system. 2. The system validates the log-in information and successfully gets access. Faculty will then be allowed to access the ViewFacultySchedule function, which will contain the following information: a. a list of all the current courses that faculty are teaching. b. scheduled day(s) and time(s) for each course that the faculty is assigned. c. the number of students that are currently enrolled in each course.
Alternate Flows	2a. Faculty will have an empty faculty schedule if they are not currently assigned to any courses for that semester.
Entry Condition	Faculty must have login access.

Front-end Implementation

- Dropdown is displayed allowing Faculty to choose which Semester they would to pull their faculty schedule information from.

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block extra_head %}
    {{ block.super }}
    <link rel="stylesheet" type="text/css" href="{% static 'registration_system/css/table.css' %}" />
{% endblock %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

     <div class="ui container">
        {{ rendered|safe }}
    </div>

    <div class="ui container m-top2" style="...">
        <div class="ui form" id="ui-form">
            <div class="inline fields">
                <div class="field">
                    <label for="id_semester_id">Semester/Year</label>
                    <select class="ui fluid dropdown" name="semester_id" id="id_semester_id" required>
                        <option value="true">-------</option>
                        {% if semesters %}
                            {% for s in semesters %}
                                <option value="{{ s.semester_id }}">{{ s.year }}-{{ s.season }}-{{ s.status }}</option>
                            {% endfor %}
                        {% endif %}
                    </select>
                </div>
            </div>
        </div>


        <div id="content">
            <table id="schedule_table">
              <thead>
                <tr>
                  <th></th>
                  <th>
                    <span class="day">1</span>
                    <span class="long">Monday</span>
                    <span class="short">Mon</span>
                  </th>
                  <th>
                    <span class="day">2</span>
                    <span class="long">Tuesday</span>
                    <span class="short">Tue</span>
                  </th>
                  <th>
                    <span class="day">3</span>
                    <span class="long">Wendsday</span>
                    <span class="short">We</span>
                  </th>
                  <th>
```

```
            <th>
              <span class="day">3</span>
              <span class="long">Wendsday</span>
              <span class="short">We</span>
            </th>
            <th>
              <span class="day">4</span>
              <span class="long">Thursday</span>
              <span class="short">Thur</span>
            </th>
            <th>
              <span class="day active">5</span>
              <span class="long">Friday</span>
              <span class="short">Fri</span>
            </th>
            <th>
              <span class="day">6</span>
              <span class="long">Saturday</span>
              <span class="short">Sat</span>
            </th>
            <th>
              <span class="day">7</span>
              <span class="long">Sunday</span>
              <span class="short">Sun</span>
            </th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td class="hour" rowspan="1"><span>8:00am</span></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
          </tr>
          <tr>
            <td class="hour" rowspan="1"><span>9:40am</span></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
          </tr>
          <tr>
            <td class="hour" rowspan="1"><span>11:20am</span></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
            <td class="cell_data"></td>
```

```
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
  </tr>
  <tr>
    <td class="hour" rowspan="1"><span>14:40pm</span></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
  </tr>
  <tr>
    <td class="hour" rowspan="1"><span>16:20pm</span></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
  </tr>
  <tr>
    <td class="hour" rowspan="1"><span>18:00pm</span></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
  </tr>
  <tr>
    <td class="hour" rowspan="1"><span>19:40pm</span></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
  </tr>
  <tr>
    <td class="hour" rowspan="1"><span>21:20pm</span></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
    <td class="cell_data"></td>
```

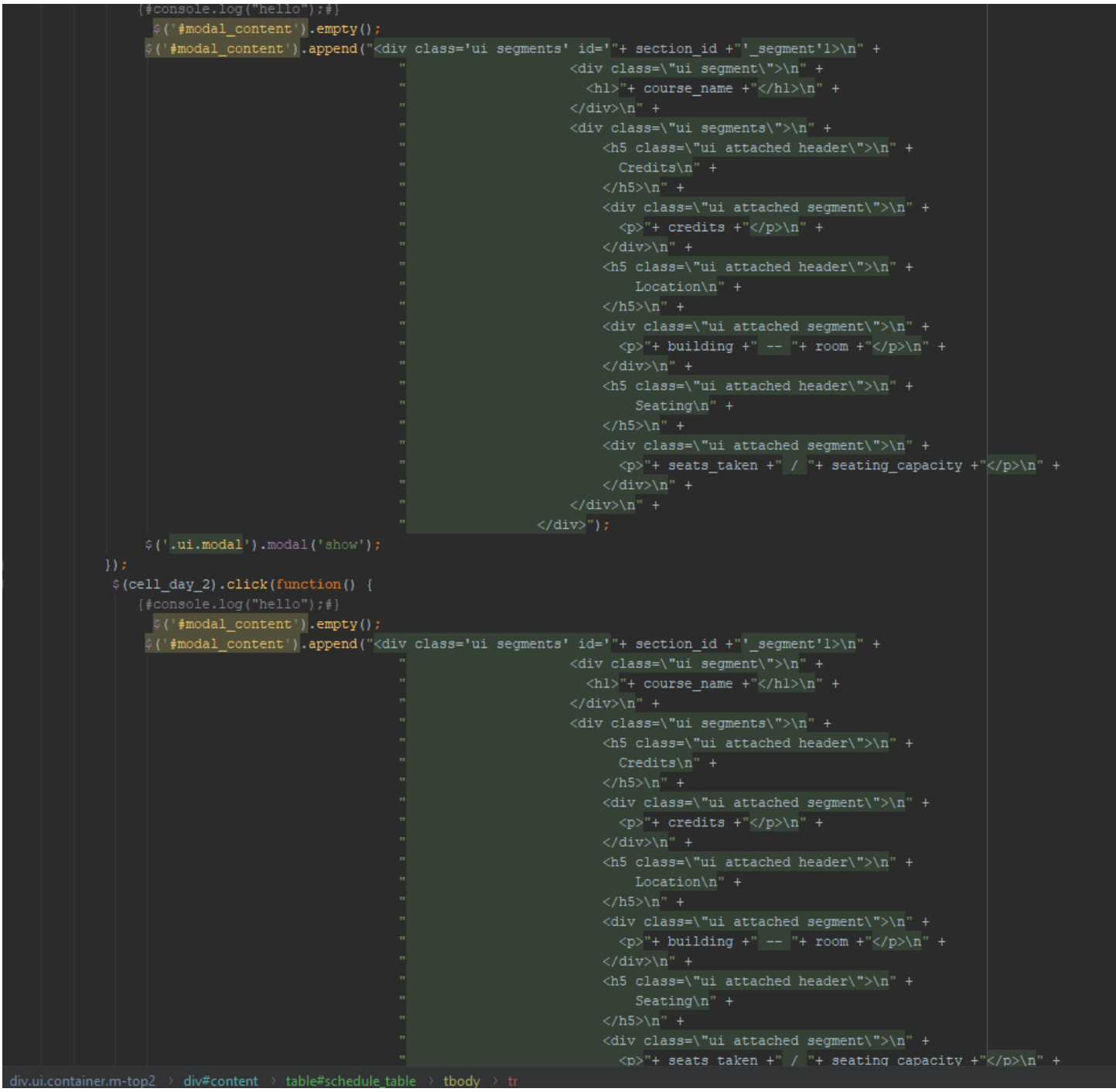
div.ui.container.m-top2 › div#content › table#schedule_table › tbody › tr

```
                <td class="cell_data"></td>
              </tr>
            </tbody>
          </table>
        </div>
        <div class="ui modal">
{#        <i class="close icon"></i>#}
          <div class="header">
          </div>
          <div id="modal_content" class="content">
          </div>
          <div class="actions">
            <div class="ui black deny button">
              Close
            </div>
          </div>
        </div>
    </div>

    <script>
        function getSections(){
            {#removeSections();#}
            removeCellData();
            var semester_id = $('#id_semester_id').val();
            $.ajax({
                type: "GET",
                url: "/student_system/view_faculty_schedule/",
                data : {
                    semester_id: semester_id
                },
                success: function(data) {
                     data.sections_array.forEach(function(s){
                          mapTimeSlotToSchedule(s.start_time, s.day_1, s.day_2, s.course_name, s.section_id, s.credits, s.building, s.room_number, s.seats_taken
                     });
                }
            });
        }

        function mapTimeSlotToSchedule(start_time, day1, day2, course_name, section_id, credits, building, room , seats_taken, seating_capacity){
            let table = $('#schedule_table')[0];
            let cell_day_1, cell_day_2;
            switch(start_time) {
                case '08:00':
                    switch(day1) {
                        case 'Monday':
                            cell_day_1 = table.rows[1].cells[1];
                            break;
                        case 'Tuesday':
                            cell_day_1 = table.rows[1].cells[2];
                            break;
                    }
                    switch(day2) {
                        case 'Wednesday':
```

div.ui.container.m-top2 › div#content › table#schedule_table › tbody › tr

view_faculty_schedule.html

```
            case '08:00':
                switch(day1) {
                    case 'Monday':
                        cell_day_1 = table.rows[1].cells[1];
            case '09:40':
                switch(day1) {
                    case 'Monday':
                        cell_day_1 = table.rows[2].cells[1];
                        break;
                    case 'Tuesday':
                        cell_day_1 = table.rows[2].cells[2];
                        break;
                }
                switch(day2) {
                    case 'Wednesday':
                        cell_day_2 = table.rows[2].cells[3];
                        break;
                    case 'Thursday':
                        cell_day_2 = table.rows[2].cells[4];
                        break;
                }
                break;
            case '11:20':
                switch(day1) {
                    case 'Monday':
                        cell_day_1 = table.rows[3].cells[1];
                        break;
                    case 'Tuesday':
                        cell_day_1 = table.rows[3].cells[2];
                        break;
                }
                switch(day2) {
                    case 'Wednesday':
                        cell_day_2 = table.rows[3].cells[3];
                        break;
                    case 'Thursday':
                        cell_day_2 = table.rows[3].cells[4];
                        break;
                }
                break;
            case '13:00':
                switch(day1) {
                    case 'Monday':
                        cell_day_1 = table.rows[4].cells[1];
                        break;
                    case 'Tuesday':
                        cell_day_1 = table.rows[4].cells[2];
                        break;
                }
                switch(day2) {
                    case 'Wednesday':
                        cell_day_2 = table.rows[4].cells[3];
                        break;
                    case 'Thursday':
                        cell_day_2 = table.rows[4].cells[4];
                        break;
```



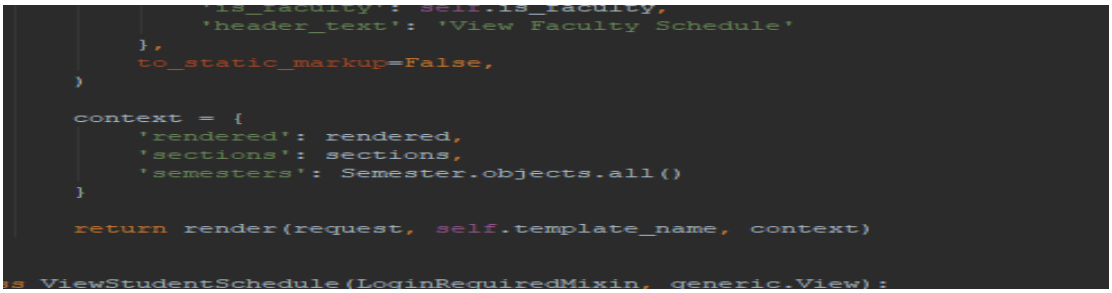
```
            {#console.log("hello");#}
              $('#modal_content').empty();
              $('#modal_content').append("<div class='ui segments' id='"+ section_id +"'_segment'1>\n" +
                                         "                    <div class=\"ui segment\">\n" +
                                         "                      <h1>"+ course_name +"</h1>\n" +
                                         "                    </div>\n" +
                                         "                    <div class=\"ui segments\">\n" +
                                         "                        <h5 class=\"ui attached header\">\n" +
                                         "                          Credits\n" +
                                         "                        </h5>\n" +
                                         "                        <div class=\"ui attached segment\">\n" +
                                         "                          <p>"+ credits +"</p>\n" +
                                         "                        </div>\n" +
                                         "                        <h5 class=\"ui attached header\">\n" +
                                         "                            Location\n" +
                                         "                        </h5>\n" +
                                         "                        <div class=\"ui attached segment\">\n" +
                                         "                          <p>"+ building +" -- "+ room +"</p>\n" +
                                         "                        </div>\n" +
                                         "                        <h5 class=\"ui attached header\">\n" +
                                         "                            Seating\n" +
                                         "                        </h5>\n" +
                                         "                        <div class=\"ui attached segment\">\n" +
                                         "                          <p>"+ seats_taken +" / "+ seating_capacity +"</p>\n" +
                                         "                        </div>\n" +
                                         "                    </div>\n" +
                                         "                </div>");
              $('.ui.modal').modal('show');
          });
           $(cell_day_2).click(function() {
              {#console.log("hello");#}
                $('#modal_content').empty();
                $('#modal_content').append("<div class='ui segments' id='"+ section_id +"'_segment'1>\n" +
                                           "                    <div class=\"ui segment\">\n" +
                                           "                      <h1>"+ course_name +"</h1>\n" +
                                           "                    </div>\n" +
                                           "                    <div class=\"ui segments\">\n" +
                                           "                        <h5 class=\"ui attached header\">\n" +
                                           "                          Credits\n" +
                                           "                        </h5>\n" +
                                           "                        <div class=\"ui attached segment\">\n" +
                                           "                          <p>"+ credits +"</p>\n" +
                                           "                        </div>\n" +
                                           "                        <h5 class=\"ui attached header\">\n" +
                                           "                            Location\n" +
                                           "                        </h5>\n" +
                                           "                        <div class=\"ui attached segment\">\n" +
                                           "                          <p>"+ building +" -- "+ room +"</p>\n" +
                                           "                        </div>\n" +
                                           "                        <h5 class=\"ui attached header\">\n" +
                                           "                            Seating\n" +
                                           "                        </h5>\n" +
                                           "                        <div class=\"ui attached segment\">\n" +
                                           "                          <p>"+ seats_taken +" / "+ seating_capacity +"</p>\n" +
div.ui.container.m-top2 › div#content › table#schedule_table › tbody › tr
```

```
                                                "                          Credits\n" +
                                                "                        </h5>\n" +
                                                "                        <div class=\"ui attached segment\">\n" +
                                                "                          <p>"+ credits +"</p>\n" +
                                                "                        </div>\n" +
                                                "                        <h5 class=\"ui attached header\">\n" +
                                                "                            Location\n" +
                                                "                        </h5>\n" +
                                                "                        <div class=\"ui attached segment\">\n" +
                                                "                          <p>"+ building +" -- "+ room +"</p>\n" +
                                                "                        </div>\n" +
                                                "                        <h5 class=\"ui attached header\">\n" +
                                                "                            Seating\n" +
                                                "                        </h5>\n" +
                                                "                        <div class=\"ui attached segment\">\n" +
                                                "                          <p>"+ seats_taken +" / "+ seating_capacity +"</p>\n" +
                                                "                        </div>\n" +
                                                "                    </div>\n" +
                                                "                </div>");
                $('.ui.modal').modal('show');
            });
            $(cell_day_1).addClass('highlighted_row');
            $(cell_day_1).append('<strong>'+course_name+' <br/>Sec. ID: '+section_id+'</strong>');
            $(cell_day_2).addClass('highlighted_row');
            $(cell_day_2).append('<strong>'+course_name+' <br/>Sec. ID: '+section_id+'</strong>');

        }

        function removeCellData(){
            $('#schedule_table tbody tr .cell_data').empty();
            $('#schedule_table tbody tr .cell_data').removeClass('highlighted_row');
        }

        function removeSections(){
            $('#content').empty();
        }

        $('#id_semester_id').change(getSections);
    </script>

{% endblock %}
```

Back-end Implementation

- If an HTTP GET Ajax requests are used to respond with the appropriate section JSON data according to which semester is chosen from the dropdown on the front-end.

```
class ViewFacultySchedule(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/view_faculty_schedule.html'
    is_faculty = True

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)
        if userprofile and userprofile.has_faculty():
            self.is_faculty = True
        else:
            redirect('/student_system/')

        if request.is_ajax():
            semester_id = request.GET.get('semester_id')
            sections_array = []
            for e in Enrollment.objects.filter(section_id__faculty_id=userprofile.faculty,
                                               section_id__semester_id__semester_id=int(semester_id)):
                prerequisites = Prerequisite.objects.filter(course_id=e.section_id.course_id)
                prereq_array = []
                for p in prerequisites:
                    prereq_array.append({
                        'name': p.course_required_id.name
                    })
                faculty_name = e.section_id.faculty_id.faculty_id.user.first_name + " " + e.section_id.faculty_id.
                sections_array.append({
                    'section_id': e.section_id_id,
                    'course_name': e.section_id.course_id.name,
                    'professor': faculty_name,
                    'credits': e.section_id.course_id.credits,
                    'room_number': e.section_id.room_id.room_number,
                    'building': e.section_id.room_id.building_id.name,
                    'meeting_days': e.section_id.time_slot_id.days_id.day_1 + " " + e.section_id.time_slot_id.days
                    'time_period': e.section_id.time_slot_id.period_id.start_time.strftime('%H:%M %p') + "-"
                                   + e.section_id.time_slot_id.period_id.end_time.strftime('%H:%M %p'),
                    'seats_taken': e.section_id.seats_taken,
                    'seating_capacity': e.section_id.room_id.capacity,
                    'prerequisites': prereq_array
                })

            data = {
                'sections_array': sections_array,
                'faculty_name': user.first_name + " " + user.last_name,
                'is_successful': True
            }
            return HttpResponse(json.dumps(data), content_type="application/json")

        sections = Section.objects.filter(faculty_id=userprofile.faculty)
        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
ViewFacultySchedule
```



SubmitGrading

Header	
Use Case ID	UC18
Use Case Version	1.0
Body	
Title	SubmitGrading
Actors	Faculty
Normal Flow	1. Faculty navigates to the SubmitGrading menu item from the faculty menu a dropdown is displayed to filter sections by semester . 2. Admin chooses a semester. 3. System displays section taught. 4. If the grading period for that semester is open The system will display to the Faculty member a list of courses taught by the professor, with a SubmitGrading button appended to the bottom of the component. 5. Faculty chooses to submit grades for one of their classes. 6. System displays all students enrolled in the section taught by the faculty member. 7. Admin Submits grades for a student 8. System updates grading information.
Alternate Flows	4a. Grading period is closed, no button shown.
Entry Condition	Faculty will view their current semester schedule and select a course
Exit Condition	The grade submission is sent.

Front-end Implementation

- Faculty may filter the courses taught by semester using the dropdown displayed.
- If the grading period is open a button is displayed to submit grading.
- Upon click, modal is displayed with a list of all students enrolled in the course as well as an input to submit their grades.

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

     <div class="ui container">
        {{ rendered|safe }}
    </div>

    <div class="ui container m-top2">
        <div class="ui form" id="ui-form">
            <div class="inline fields">
                <div class="field">
                    <label for="id_semester_id">Semester/Year</label>
                    <select class="ui fluid dropdown" name="semester_id" id="id_semester_id" required>
                        <option value="true">--------</option>
                        {% if semesters %}
                            {% for s in semesters %}
                                <option value="{{ s.semester_id }}">{{ s.year }}-{{ s.season }}-{{ s.status }}</option>
                            {% endfor %}
                        {% endif %}
                    </select>
                </div>
            </div>
         </div>
         <div id="content"></div>
        <div class="ui modal">
          <div id="id_modal_section" class="header"></div>
          <div  class="scrolling content">
              <div id="id_modal_content" class="ui celled list"></div>
          </div>
            <div id="id_modal_success"></div>
        </div>
    </div>

    <script>

        function getSectionsTaught() {
            $('#content').empty();
            let semester_id = $('#id_semester_id').val();
            if(semester_id === "true"){ return; }
            $.ajax({
                type: "GET",
                url: "/student_system/submit_grades/get_sections/",
                data : {
                    semester_id: semester_id
                },
                success: function(data) {
                    data.sections_array.forEach(function(s){
                        $('#content').append("<div class=\"ui segments\">\n" +
                            "                    <div class=\"ui segment\">\n" +
```

```
                data.sections_array.forEach(function(s){
                    $('#content').append("<div class=\"ui segments\">\n" +
                        "                    <div class=\"ui segment\">\n" +
                        "                      <h1>"+s.course_name+"</h1>\n" +
                        "                    </div>\n" +
                        "                    <div class=\"ui segments\">\n" +
                        "                      <div class=\"ui segment\">\n" +
                        "                        <strong>Building Name: </strong><p>"+ s.building +"</p>\n" +
                        "                      </div>\n" +
                        "                      <div class=\"ui segment\">\n" +
                        "                        <strong>Room Number: </strong><p>"+ s.room_number +"</p>\n" +
                        "                      </div>\n" +
                        "                      <div class=\"ui segment\">\n" +
                        "                        <strong>Time Slot: </strong><p>"+s.meeting_days+" ---- "+s.time_period+"</p>\n" +
                        "                      </div>\n" +
                        "                      <div class=\"ui segment\">\n" +
                        "                          <strong>Season/Semester</strong><p>"+s.semester_season +" - " + s.semester_year +"</p>\n" +
                        "                      </div>\n" +
                        "                    </div>\n" +
                        "                      <div id="+s.section_id+"__grading' class=\"ui segment\">\n" +
                        "                      </div>\n" +
                        "\n" +
                        "                </div>");
                    {#console.log(s.semester_status);#}
                    if( s.semester_status === 'OPEN_GRADING') {
                        $('#' + s.section_id + "__grading").append("<button onclick='getSectionStudents("+ s.section_id +")' class='ui prim
                    }
                })
            }
        })
    }

    function getSectionStudents(section_id) {
        $('#id_modal_content').empty();
        $('#id_modal_section').empty();
        $.ajax({
            type: "GET",
            url: "/student_system/submit_grades/",
            data: {
                section_id: section_id
            },
            success: function (data) {
                {#console.log(data);#}
                $('#id_modal_section').append("<h1>"+data.section_name+"</h1>");
                {# hidden input for section_id #}
                 data.students_array.forEach(function(e) {
                    $('#id_modal_content').append("<div class='item'><div class=\"ui form\">\n" +
                        "  <div class=\"fields\">\n" +
                        "    <div class=\"field\">\n" +
                        "      <label>First name</label>\n" +
                        "      <input type=\"text\" name='first_name' value='" + e.first_name + "'>\n" +
                        "    </div>\n" +
                        "    <div class=\"field\">\n" +
                        "      <label>Last name</label>\n" +
```

```
                    "  <div class=\"fields\">\n" +
                    "    <div class=\"field\">\n" +
                    "      <label>First name</label>\n" +
                    "      <input type=\"text\" name='first_name' value='" + e.first_name + "'>\n" +
                    "    </div>\n" +
                    "    <div class=\"field\">\n" +
                    "      <label>Last name</label>\n" +
                    "      <input type=\"text\" name='last_name' value='" + e.last_name + "'>\n" +
                    "    </div>\n" +
                    "    <div class=\"field\">\n" +
                    "      <label>Semester Status</label>\n" +
                    "      <input type=\"text\" name='semester_status' value='" + data.semester_status + "'>\n" +
                    "    </div>\n" +
                    "    <div class=\"field\">\n" +
                    "      <label>Grade</label>\n" +
                    "      <input type=\"text\" name='grade' id='" + e.student_id + "__grade' maxlength='2' >\n" +
                    "    </div>\n" +
                    "    <div class=\"field\">\n" +
                    "      <label>Submit</label>\n" +
                    "      <button type='button' onclick='submitGrade(this.id," + e.student_id + ")' class='ui primary button' id='"
                    "    </div>\n" +
                    "  </div>\n" +
                    "</div> </div>")
            });
            $('.ui.modal').modal('show');
        }
    });
}

function submitGrade(section_id, student_id) {
    $('#id_modal_success').empty();
    $.ajax({
        type: "POST",
        url: "/student_system/submit_grades/",
        data: {
            section_id: section_id,
            student_id: student_id,
            letter_grade: $('#'+student_id+'__grade').val(),
            csrfmiddlewaretoken: '{{ csrf_token }}'
        },
        success: function (data) {
            console.log(data);
            $('#id_modal_success').append("<div class=\"ui positive message\">\n" +
                "  <div class=\"header\">\n" +
                "    Grade Submitted successfully!\n" +
                "  </div>\n" +
                "</div>")
        }
    });
}

$('#id_semester_id').change(getSectionsTaught);

</script>
```

Back-end Implementation

- If an HTTP GET request is sent through Ajax, all students enrolled in the section are fetched and sent back as a JSON response to populate the modal view
- If an HTTP POST request is sent Student grades are updated in the enrollment table, to reflect the grade submitted.

```
class SubmitGrades(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/submit_grades.html'
    is_faculty = False

    def get(self, request, *args, **kwargs):
        user = request.user
        user_profile = UserProfile.objects.get(user=user)

        if user_profile:
            if user_profile.has_faculty():
                self.is_faculty = True
            else:
                redirect('/student_system/')

        if request.is_ajax():
            section_id = request.GET.get('section_id')
            enrollments = Enrollment.objects.filter(section_id=section_id)
            section_name = Section.objects.get(pk=int(section_id)).course_id.name
            students_array = []
            for e in enrollments:

                students_array.append({
                    # 'section_id': e.section_id_id,
                    'student_id': e.student_id_id,
                    'first_name': e.student_id.student_id.user.first_name,
                    'last_name': e.student_id.student_id.user.last_name
                })
            data = {
                'students_array': students_array,
                'section_name': section_name,
                'section_id': section_id,
                'semester_status': Section.objects.get(pk=int(section_id)).semester_id.status
            }

            return HttpResponse(json.dumps(data), content_type="application/json")

        sections = Section.objects.filter(faculty_id=user_profile.faculty)

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_faculty': self.is_faculty,
                'header_text': 'Grading'
            },
            to_static_markup=False,
            )

        context = {
SubmitGrades › get()
ntrol
```

```
        context = {
            'rendered': rendered,
            'sections': sections,
            'semesters': Semester.objects.all()
        }

        return render(request, self.template_name, context)

    def post(self, request, *args, **kwargs):
        data = {
            'is_successful': False
        }
        if request.is_ajax():
            student_id = request.POST.get('student_id')
            section_id = request.POST.get('section_id')
            letter_grade = request.POST.get('letter_grade')
            enrollment = Enrollment.objects.get(section_id=section_id, student_id=student_id)
            enrollment.grade = letter_grade
            enrollment.save()
            data['is_successful'] = True
        else:
            data['is_successful'] = False
        return JsonResponse(data)
```

Student Implementation:

Header	
Use Case ID	UC1
Use Case Version	1.0
Body	
Title	ViewStudentSchedule
Actors	Student/Admin/Faculty
Input	Student ID
Output	Students Schedule
Normal Flow	1. User Is logged and chooses the option (menu option) to view their/a student schedule. 2. Student course schedule (includes course they have registered for) is presented to the user.
Alternate Flows	1. B. Admin requests student schedule.
Entry Condition	User is logged into the system.
Exit Condition	The user can see the student schedule.

Admin/Faculty Front-end Implementation

- Faculty must search the student my first and last Name, upon search all courses the student is currently enrolled in will be displayed.
- Students must choose from the dropdown which semesters schedule should be displayed.

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

    <div class="ui container">
        {{ rendered|safe }}
    </div>

    <div class="ui container m-top2">
        <div class="ui form" id="ui-form">
            <div class="two fields">
                <div class="field">
                    <label for="username_id">Username</label>
                    <input id="username_id"   placeholder="Search by username..." type="text" />
                </div>
                <div class="field">
                    <label for="email_id">Email</label>
                    <input id="email_id" autocomplete="email" placeholder="Search by email..." type="text"/>
                </div>
            </div>
        {{ csrf }}
             <div class="two fields">
                <div class="field">
                    <label for="first_name_id">First Name</label>
                    <input id="first_name_id"   placeholder="Search by first name..." type="text" />
                </div>
                <div class="field">
                    <label for="last_name_id">Last Name</label>
                    <input id="last_name_id" placeholder="Search by last name..." type="text"/>
                </div>
            </div>
            <div class="display-inline ">
                <button class="ui primary button" id="search" type="button">Search</button>
                <button class="ui negative button" id="clear" type="button">Clear</button>
            </div>
            <div id="full_name_id">

            </div>
            <table class="ui celled table">
                <thead>
                    <tr>
                        <th>Section ID</th>
                        <th>Course Name</th>
                        <th>Department</th>
                        <th>Professor</th>
                        <th>Credits</th>
                        <th>Seating</th>
                        <th>Time Slot</th>
div.ui.container.m-top2 › div#ui-form.ui.form › table.ui.celled.table › thead › tr › th
```

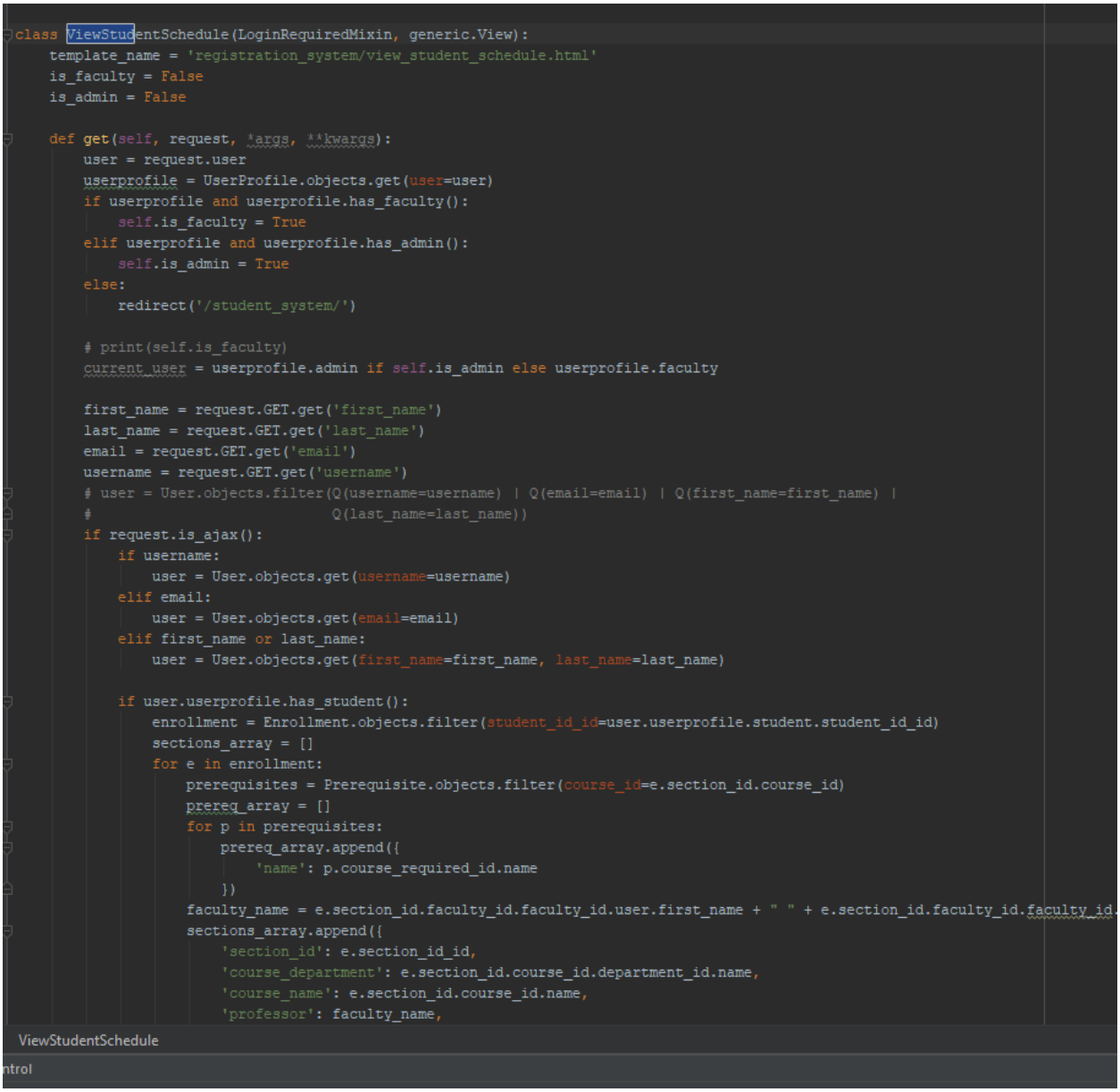
```
        </div>
        <div id="success">

        </div>
    </div>

    <script>
        function addStudentSections(){
            {#console.log("here");#}
            removeSections();
            var first_name = $('#first_name_id').val();
            var last_name = $('#last_name_id').val();
            var username = $('#username_id').val();
            var email = $('#email_id').val();
            $.ajax({
                type: "GET",
                url: "/student_system/view_student_schedule/",
                data : {
                    first_name: first_name,
                    last_name: last_name,
                    username: username,
                    email: email
                },
                success: function(data) {
                    console.log(data);
                    if(data.is_successful) {
                        data.sections_array.forEach(function (element) {
                            var tBody = document.getElementById('sections_added');
                            var newRow = tBody.insertRow(tBody.rows.length);
                            newRow.id = element.section_id + '_row';
                            var sectionIdCell = newRow.insertCell(0);
                            var courseNameCell = newRow.insertCell(1);
                            var departmentCell = newRow.insertCell(2);
                            var facultyCell = newRow.insertCell(3);
                            var creditsCell = newRow.insertCell(4);
                            var seatsCell = newRow.insertCell(5);
                            var timeSlotCell = newRow.insertCell(6);
                            var locationCell = newRow.insertCell(7);
                            var semesterCell = newRow.insertCell(8);
                            var prerequisitesCell = newRow.insertCell(9);
                            {# editing faculty, meeeting days, time period, building, and room number  #}
                            var sectionIdText = document.createTextNode(element.section_id);
                            var courseNameText = document.createTextNode(element.course_name);
                            var departmentText = document.createTextNode(element.course_department);
                            var creditsText = document.createTextNode(element.credits);
```

div.ui.container.m-top2 › div#ui-form.ui.form › table.ui.celled.table › thead › tr › th

```
                            var seatsText = $("<ul style='list-style:none; padding-left:0px'><li><span style='font-weight:bold'>Taken:</span> " + element.seats_t
                            sectionIdCell.appendChild(sectionIdText);
                            courseNameCell.appendChild(courseNameText);
                            departmentCell.appendChild(departmentText);
                            semesterCell.appendChild(semesterText);
                            creditsCell.appendChild(creditsText);
                            facultyCell.appendChild(facultyText);
                            {#prerequisitesCell.appendChild(prerequisitesText);#}
                            $('#' + element.section_id + '_row').find('td:eq(9)').append(prerequisitesText);
                            locationCell.insertAdjacentHTML('afterbegin', "<ul style=\'list-style:none; padding-left:0px\'><li><span style=\'font-weight:bold\'>
                            timeSlotCell.insertAdjacentHTML('afterbegin', "<ul style=\'list-style:none; padding-left:0px\'><li><span style=\'font-weight:bold\'>
                            $('#' + element.section_id + '_row').find('td:eq(5)').append(seatsText);
                            element.prerequisites.forEach(function (el) {
                                $('#' + element.section_id + "_prerequisites").append("<div class='item'>" + el.name + "</div>");
                            });


                        });
                        $('#full_name_id').append("<h1 class='ui header' >"+data.student_name+"'s Schedule</h1>")
                    } else {
                      $('#full_name_id').append("<h1 class='ui header red' >No Students connected to your search query.</h1>")
                    }

                }
            });

        }

        function removeSections() {
            $('#sections_added').empty();
            $('#full_name_id').empty();
        }


        $('#search').click(addStudentSections);
        $('#clear').click(removeSections);

    </script>
```

Admin/Faculty Back-end Implementation

- If an HTTP GET request is sent through Ajax all relevant sections, the student is enrolled in is propagated back to the front-end as JSON data to populate the table structure on the front-end.



```
class ViewStudentSchedule(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/view_student_schedule.html'
    is_faculty = False
    is_admin = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)
        if userprofile and userprofile.has_faculty():
            self.is_faculty = True
        elif userprofile and userprofile.has_admin():
            self.is_admin = True
        else:
            redirect('/student_system/')

        # print(self.is_faculty)
        current_user = userprofile.admin if self.is_admin else userprofile.faculty

        first_name = request.GET.get('first_name')
        last_name = request.GET.get('last_name')
        email = request.GET.get('email')
        username = request.GET.get('username')
        # user = User.objects.filter(Q(username=username) | Q(email=email) | Q(first_name=first_name) |
        #                            Q(last_name=last_name))
        if request.is_ajax():
            if username:
                user = User.objects.get(username=username)
            elif email:
                user = User.objects.get(email=email)
            elif first_name or last_name:
                user = User.objects.get(first_name=first_name, last_name=last_name)

            if user.userprofile.has_student():
                enrollment = Enrollment.objects.filter(student_id_id=user.userprofile.student.student_id_id)
                sections_array = []
                for e in enrollment:
                    prerequisites = Prerequisite.objects.filter(course_id=e.section_id.course_id)
                    prereq_array = []
                    for p in prerequisites:
                        prereq_array.append({
                            'name': p.course_required_id.name
                        })
                    faculty_name = e.section_id.faculty_id.faculty_id.user.first_name + " " + e.section_id.faculty_id.faculty_id.
                    sections_array.append({
                        'section_id': e.section_id_id,
                        'course_department': e.section_id.course_id.department_id.name,
                        'course_name': e.section_id.course_id.name,
                        'professor': faculty_name,
ViewStudentSchedule
ntrol
```

```
                    'semester': e.section_id.semester_id.season + '-' + str(e.section_id.semester_id.year) +
                    'credits': e.section_id.course_id.credits,
                    'room_number': e.section_id.room_id.room_number,
                    'building': e.section_id.room_id.building_id.name,
                    'meeting_days': e.section_id.time_slot_id.days_id.day_1 + " " + e.section_id.time_slot_
                    'time_period': e.section_id.time_slot_id.period_id.start_time.strftime('%H:%M %p') + "-
                                + e.section_id.time_slot_id.period_id.end_time.strftime('%H:%M %p'),
                    'seats_taken': e.section_id.seats_taken,
                    'seating_capacity': e.section_id.room_id.capacity,
                    'prerequisites': prereq_array
                })
            data = {
                'sections_array': sections_array,
                'student_name': user.first_name + " " + user.last_name,
                'is_successful': True
            }
            return HttpResponse(json.dumps(data), content_type="application/json")
        else:
            return HttpResponse(json.dumps({'is_successful': False}))

    rendered = render_component(
        os.path.join(os.getcwd(), 'registration_system', 'static',
                     'registration_system', 'js', 'nav-holder.jsx'),
        {
            'is_admin': self.is_admin,
            'is_faculty': self.is_faculty,
            'header_text': 'View Student Schedule'
        },
        to_static_markup=False,
    )

    context = {
        'rendered': rendered
    }

    return render(request, self.template_name, context)
```

Student Front-end Implementation

- A dropdown is displayed allowing students to choose which schedule to be displayed by semester.
- When a semester is chosen all sections enrolled in will be populated into the schedule.

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

    <div class="ui container">
       {{ rendered|safe }}
    </div>

    <div class="ui container m-top2 m-bottom4">
        <div class="ui form" id="ui-form">
            <div class="inline fields">
                <div class="field">
                    <label for="id_semester_id">Semester/Year</label>
                    <select class="ui fluid dropdown" name="semester_id" id="id_semester_id" required>
                        <option value="true">-------</option>
                        {% if semesters %}
                            {% for s in semesters %}
                                <option value="{{ s.semester_id }}">{{ s.year }}-{{ s.season }}-{{ s.status }}</option>
                            {% endfor %}
                        {% endif %}
                    </select>
                </div>
            </div>
        </div>

        <div id="content">

        </div>
        <div id="success">

        </div>
    </div>

    <script>
        function getSections(){
            removeSections();
            var semester_id = $('#id_semester_id').val();
            $.ajax({
                type: "GET",
                url: "/student_system/student/view_student_schedule/",
                data : {
                    semester_id: semester_id
                },
                success: function(data) {
                     data.sections_array.forEach(function(s){
                           $('#content').append("<div class='ui segments' id='"+ s.section_id +"'_segment'>\n" +
                                                "                    <div class=\"ui segment\">\n" +
                                                "                      <h1>"+ s.course_name +"</h1>\n" +
```

script

```
function getSections(){
    removeSections();
    var semester_id = $('#id_semester_id').val();
    $.ajax({
        type: "GET",
        url: "/student_system/student/view_student_schedule/",
        data : {
            semester_id: semester_id
        },
        success: function(data) {
            data.sections_array.forEach(function(s){
                $('#content').append("<div class='ui segments' id='"+ s.section_id +"'_segment'>\n" +
                                     "                    <div class=\"ui segment\">\n" +
                                     "                      <h1>"+ s.course_name +"</h1>\n" +
                                     "                    </div>\n" +
                                     "                    <div class=\"ui segments\">\n" +
                                     "                        <h5 class=\"ui attached header\">\n" +
                                     "                          Professor\n" +
                                     "                        </h5>\n" +
                                     "                        <div class=\"ui attached segment\">\n" +
                                     "                          <p>"+ s.professor +"</p>\n" +
                                     "                        </div>\n" +
                                     "                        <h5 class=\"ui attached header\">\n" +
                                     "                          Credits\n" +
                                     "                        </h5>\n" +
                                     "                        <div class=\"ui attached segment\">\n" +
                                     "                          <p>"+ s.credits +"</p>\n" +
                                     "                        </div>\n" +
                                     "                        <h5 class=\"ui attached header\">\n" +
                                     "                            Location\n" +
                                     "                        </h5>\n" +
                                     "                        <div class=\"ui attached segment\">\n" +
                                     "                          <p>"+ s.building +" -- "+ s.room_number +"</p>\n" +
                                     "                        </div>\n" +
                                     "                        <h5 class=\"ui attached header\">\n" +
                                     "                            Time Slot\n" +
                                     "                        </h5>\n" +
                                     "                        <div class=\"ui attached segment\">\n" +
                                     "                          <p>" + s.meeting_days +"</p><br/>\n" +
                                     "                            <p>" + s.time_period +"</p>\n" +
                                     "                        </div>\n" +
                                     "                        <h5 class=\"ui attached header\">\n" +
                                     "                            Seating\n" +
                                     "                        </h5>\n" +
                                     "                        <div class=\"ui attached segment\">\n" +
                                     "                          <p>"+ s.seats_taken +" / "+ s.seating_capacity +"</p>\n" +
                                     "                        </div>\n" +
                                     "                    </div>\n" +
                                     "                </div>")
            });
        }
ipt
ntrol
```

Student Back-end Implementation

- If an HTTP GET request is sent through AJAX all sections related the semester_id requested are propagated back to the front-end through JSON and populated into the schedule.

```
class StudentViewSchedule(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/student_view_student_schedule.html'
    is_student = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)

        if userprofile:
            if userprofile.has_student():
                self.is_student = True
            else:
                redirect('/student_system/')

        if request.is_ajax():
                semester_id = request.GET.get('semester_id')
                sections_array = []
                for e in Enrollment.objects.filter(student_id=userprofile.student, section_id__semester_id__semester_id=int(s
                    prerequisites = Prerequisite.objects.filter(course_id=e.section_id.course_id)
                    prereq_array = []
                    for p in prerequisites:
                        prereq_array.append({
                            'name': p.course_required_id.name
                        })
                    faculty_name = e.section_id.faculty_id.faculty_id.user.first_name + " " + e.section_id.faculty_id.faculty
                    sections_array.append({
                        'section_id': e.section_id_id,
                        'course_name': e.section_id.course_id.name,
                        'professor': faculty_name,
                        'credits': e.section_id.course_id.credits,
                        'room_number': e.section_id.room_id.room_number,
                        'building': e.section_id.room_id.building_id.name,
                        'meeting_days': e.section_id.time_slot_id.days_id.day_1 + " " + e.section_id.time_slot_id.days_id.day
                        'time_period': e.section_id.time_slot_id.period_id.start_time.strftime('%H:%M %p') + "-"
                                       + e.section_id.time_slot_id.period_id.end_time.strftime('%H:%M %p'),
                        'seats_taken': e.section_id.seats_taken,
                        'seating_capacity': e.section_id.room_id.capacity,
                        'prerequisites': prereq_array
                    })

                data = {
                    'sections_array': sections_array,
                    'student_name': user.first_name + " " + user.last_name,
                    'is_successful': True
                }
                return HttpResponse(json.dumps(data), content_type="application/json")

        rendered = render_component(
StudentViewSchedule
```

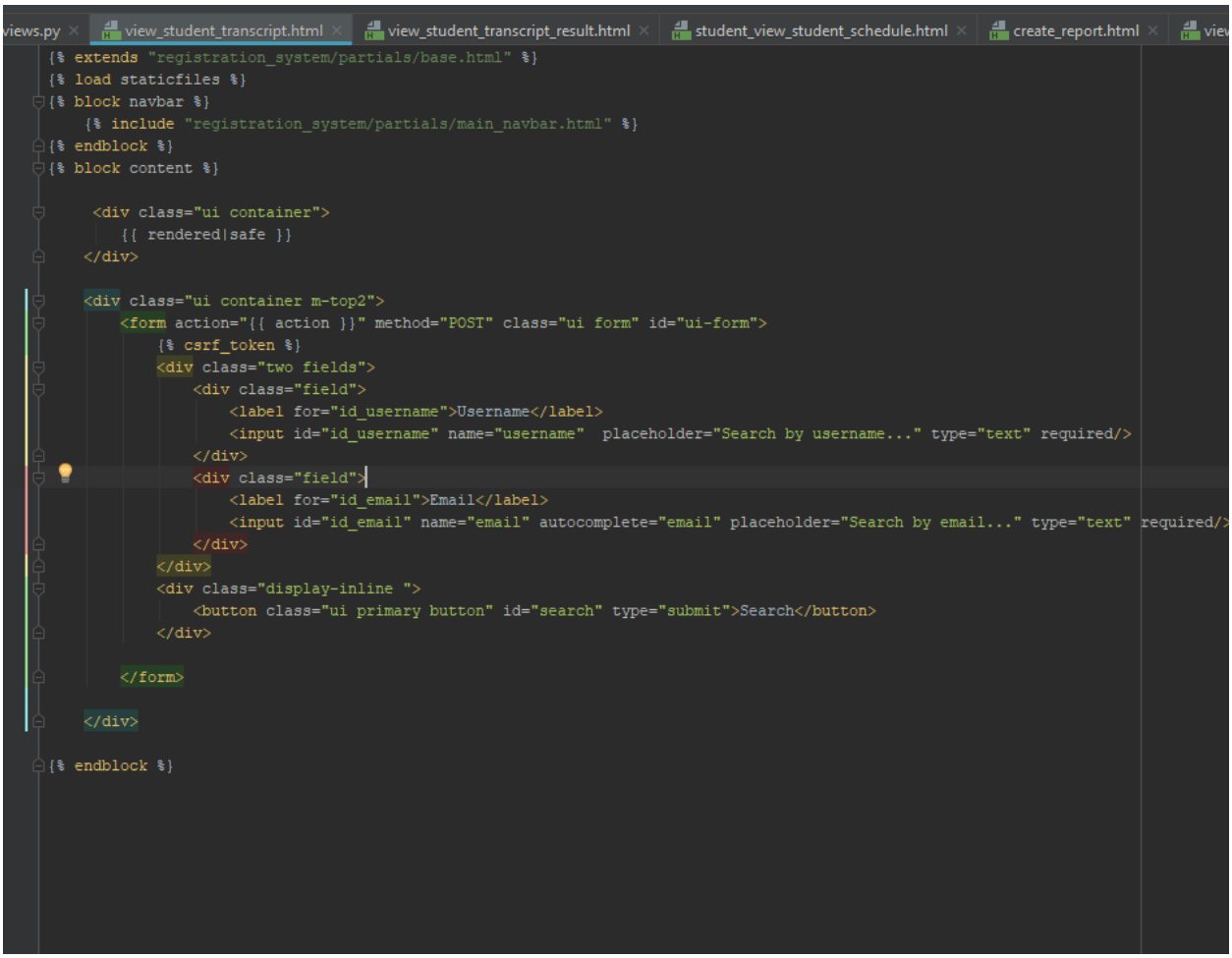
```
        return HttpResponse(json.dumps(data), content_type="application/json")

    rendered = render_component(
        os.path.join(os.getcwd(), 'registration_system', 'static',
                     'registration_system', 'js', 'nav-holder.jsx'),
        {
            'is_student': self.is_student,
            'header_text': 'Student Schedule'
        },
        to_static_markup=False,
    )

    context = {
        'semesters': Semester.objects.all(),
        'rendered': rendered,
    }
    return render(request, self.template_name, context)
```

ViewTranscript

Header	
Use Case ID	UC3
Use Case Version	1.0
Body	
Title	ViewTranscript
Actors	Student/Faculty/Admin
Input	Student ID
Output	Student transcript
Normal Flow	1. User is logged in and accesses their student account privileges. 2. User is presented menu option to request transcript. 3. Request is sent to the server. 4. All Transcript information relevant to the logged in student is extracted from their data source and processed into a transcript template for the user to view, print, etc.
Alternate Flows	2a. Faculty or Admin user is displayed the option to query student by name or id.
Entry Condition	User is logged in.
Exit Condition	User is presented the student transcript requested.

Admin/Faculty Front-end Implementation

- Admin/Faculty users must input Students First and Last name.
- Once Submitted the user is redirected to the full Student Transcript page with all transcript data shown.


views.py
view_student_transcript.html
view_student_transcript_result.html
student_view_student_schedule.html
create_report.html
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
{% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

{{ rendered|safe }}

<form action="{{ action }}" method="POST" class="ui form" id="ui-form">
{% csrf_token %}

<label for="id_username">Username</label>
<input id="id_username" name="username" placeholder="Search by username..." type="text" required/>

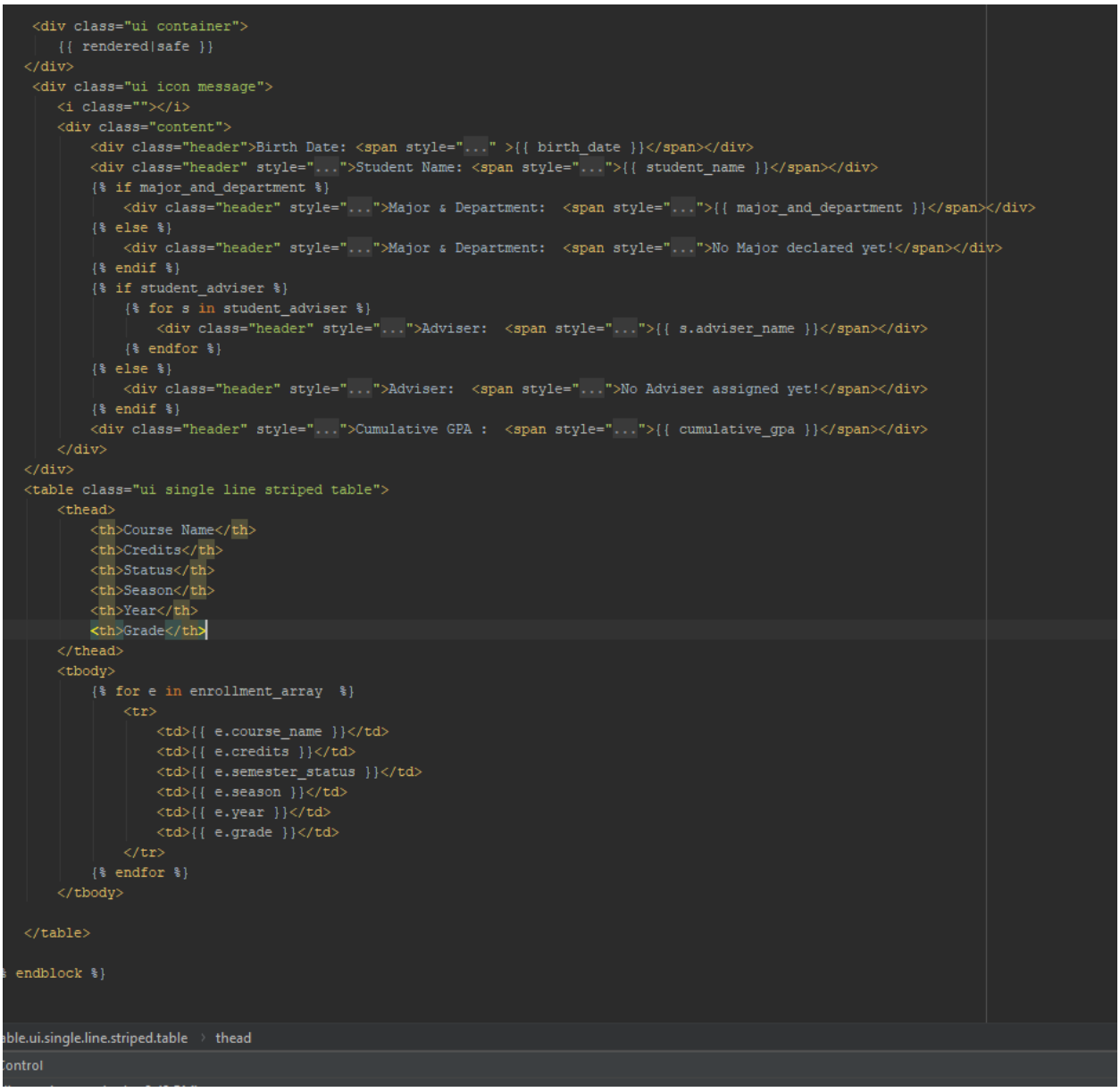
<label for="id_email">Email</label>
<input id="id_email" name="email" autocomplete="email" placeholder="Search by email..." type="text" required/>

<button class="ui primary button" id="search" type="submit">Search</button>

</form>

{% endblock %}

- The resulting template will be shown upon querying student my first and last name.



```
 <div class="ui container">
    {{ rendered|safe }}
</div>
 <div class="ui icon message">
    <i class=""></i>
    <div class="content">
        <div class="header">Birth Date: <span style="..." >{{ birth_date }}</span></div>
        <div class="header" style="...">Student Name: <span style="...">{{ student_name }}</span></div>
        {% if major_and_department %}
            <div class="header" style="...">Major & Department:  <span style="...">{{ major_and_department }}</span></div>
        {% else %}
            <div class="header" style="...">Major & Department:  <span style="...">No Major declared yet!</span></div>
        {% endif %}
        {% if student_adviser %}
            {% for s in student_adviser %}
                <div class="header" style="...">Adviser:  <span style="...">{{ s.adviser_name }}</span></div>
            {% endfor %}
        {% else %}
            <div class="header" style="...">Adviser:  <span style="...">No Adviser assigned yet!</span></div>
        {% endif %}
        <div class="header" style="...">Cumulative GPA :  <span style="...">{{ cumulative_gpa }}</span></div>
    </div>
</div>
<table class="ui single line striped table">
    <thead>
        <th>Course Name</th>
        <th>Credits</th>
        <th>Status</th>
        <th>Season</th>
        <th>Year</th>
        <th>Grade</th>
    </thead>
    <tbody>
        {% for e in enrollment_array  %}
            <tr>
                <td>{{ e.course_name }}</td>
                <td>{{ e.credits }}</td>
                <td>{{ e.semester_status }}</td>
                <td>{{ e.season }}</td>
                <td>{{ e.year }}</td>
                <td>{{ e.grade }}</td>
            </tr>
        {% endfor %}
    </tbody>

</table>

{% endblock %}
```

Admin/Faculty Back-end Implementation

- POST route is used to redirect the user the appropriate transcript result page depending on the student searched.

```
class ViewStudentTranscript(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/view_student_transcript.html'
    is_faculty = False
    is_admin = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)
        if userprofile and userprofile.has_faculty():
            self.is_faculty = True
        elif userprofile and userprofile.has_admin():
            self.is_admin = True
        else:
            redirect('/student_system/')

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_admin': self.is_admin,
                'is_faculty': self.is_faculty,
                'header_text': 'View Transcript'
            },
            to_static_markup=False,
        )

        context = {
            'rendered': rendered
        }

        return render(request, self.template_name, context)

    def post(self, request, *args, **kwargs):
        username = request.POST.get('username')
        email = request.POST.get('email')
        # print(request.POST)

        if username:
            user = User.objects.get(username=username)
        elif email:
            user = User.objects.get(email=email)
        else:
            user = False

        userprofile = UserProfile.objects.get(user=user)
        if userprofile.has_student():
            student = userprofile.student
            return redirect('view_student_transcript_result', student_id=student.student_id_id)
        return redirect('/student_system/view_student_transcript/')
```

```python
class ViewStudentTranscriptResult(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/view_student_transcript_result.html'
    is_faculty = False
    is_admin = False
    grading_key = {
        'A': 4.0,
        'A-': 3.5,
        'B': 3.0,
        'B-': 2.5,
        'C': 2.0,
        'C-': 1.5,
        'D': 1.0,
        'D-': 0.5,
        'F': 0
    }

    def get(self, request, student_id, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)
        if userprofile and userprofile.has_faculty():
            self.is_faculty = True
        elif userprofile and userprofile.has_admin():
            self.is_admin = True
        else:
            redirect('/student_system/')
        student = Student.objects.get(pk=int(student_id))
        enrollments = Enrollment.objects.filter(student_id=student)
        enrollments_array = []
        grades = 0.0
        counter = 0
        # print(enrollments)
        try:
            student_major_rec = StudentMajor.objects.get(student_id=student)
            student_major = student_major_rec.major_id.name
            student_major_dep = student_major_rec.major_id.department_id.name
        except StudentMajor.DoesNotExist:
            student_major = 'None Declared'
            student_major_dep = 'N/A'

        adviser_array = []
        for a in Advising.objects.filter(student_id=student):
            adviser_array.append({
                'adviser_name': a.faculty_id.faculty_id.user.first_name + ' ' + a.faculty_id.faculty_id
            })
        for e in enrollments:
            if e.grade != 'I' and e.grade != 'W' and e.grade != 'NA':
                grades += self.grading_key[e.grade]
```

ViewStudentTranscript › post()

```
        counter = counter + 1
        enrollments_array.append({
            'course_name': e.section_id.course_id.name,
            'credits': e.section_id.course_id.credits,
            'semester_status': e.section_id.semester_id.status,
            'season': e.section_id.semester_id.season,
            'year': e.section_id.semester_id.year,
            'grade': e.grade
        })
    if counter == 0 and grades == 0.0:
        cumulative_gpa = 0.0
    else:
        cumulative_gpa = float(grades / counter)
    rendered = render_component(
        os.path.join(os.getcwd(), 'registration_system', 'static',
                     'registration_system', 'js', 'nav-holder.jsx'),
        {
            'is_admin': self.is_admin,
            'is_faculty': self.is_faculty,
            'header_text': student.student_id.user.first_name+' '+student.student_id.user.last_name+'Transcript
        },
        to_static_markup=False,
    )

    context = {
        'rendered': rendered,
        'enrollment_array': enrollments_array,
        'cumulative_gpa': cumulative_gpa,
        'student_name': student.student_id.user.first_name + ' ' + student.student_id.user.last_name,
        'birth_date': student.date_of_birth.strftime("%Y-%m-%d"),
        'student_advisers': adviser_array,
        'major_and_department': student_major + " / " + student_major_dep
    }

    return render(request, self.template_name, context)
```

Student Front-end Implementation

- Student Transcript data is fetched and displayed immediately to the student signed in.

```
{% block content %}

    <div class="ui container">
        {{ rendered|safe }}
    </div>
    <div class="ui icon message">
        <i class=""></i>
        <div class="content">
            <div class="header">Birth Date: <span style="..." >{{ birth_date }}</span></div>
            <div class="header" style="...">Student Name: <span style="...">{{ student_name }}</span></div>
            {% if major_and_department %}
                <div class="header" style="...">Major & Department:  <span style="...">{{ major_and_department }}</span></di
            {% else %}
                <div class="header" style="...">Major & Department:  <span style="...">No Major declared yet!</span></div>
            {% endif %}
            {% if student_adviser %}
                {% for s in student_adviser %}
                    <div class="header" style="...">Adviser:  <span style="...">{{ s.adviser_name }}</span></div>
                {% endfor %}
            {% else %}
                <div class="header" style="...">Adviser:  <span style="...">No Adviser assigned yet!</span></div>
            {% endif %}
            <div class="header" style="...">Cumulative GPA :  <span style="...">{{ cumulative_gpa }}</span></div>
        </div>
    </div>
    <table class="ui single line striped table">
        <thead>
            <th>Course Name</th>
            <th>Credits</th>
            <th>Status</th>
            <th>Season</th>
            <th>Year</th>
            <th>Grade</th>
        </thead>
        <tbody>
            {% for e in enrollment_array  %}
                <tr>
                    <td>{{ e.course_name }}</td>
                    <td>{{ e.credits }}</td>
                    <td>{{ e.semester_status }}</td>
                    <td>{{ e.season }}</td>
                    <td>{{ e.year }}</td>
                    <td>{{ e.grade }}</td>
                </tr>
            {% endfor %}
        </tbody>

    </table>

div.ui.icon.message  ›  div.content
ion Control
```

Student Back-end Implementation

```
class StudentViewStudentTranscript(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/student_view_student_transcript.html'
    is_student = False
    grading_key = {
        'A': 4.0,
        'A-': 3.5,
        'B': 3.0,
        'B-': 2.5,
        'C': 2.0,
        'C-': 1.5,
        'D': 1.0,
        'D-': 0.5,
        'F': 0
    }

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)

        if userprofile:
            if userprofile.has_student():
                self.is_student = True
            else:
                redirect('/student_system/')

        student_major_rec = StudentMajor.objects.get(student_id=userprofile.student)
        student_major = student_major_rec.major_id.name
        student_major_dep = student_major_rec.major_id.department_id.name
        adviser_array = []
        for a in Advising.objects.filter(student_id=userprofile.student):
            adviser_array.append({
                'adviser_name': a.faculty_id.faculty_id.user.first_name + ' ' + a.faculty_id.faculty_id.user.last_name
            })

        enrollments = Enrollment.objects.filter(student_id=userprofile.student)
        enrollments_array = []
        grades = 0.0
        counter = 0
        for e in enrollments:
            if e.grade != 'I' and e.grade != 'W' and e.grade != 'NA':
                grades += self.grading_key[e.grade]

            counter = counter + 1
            enrollments_array.append({
                'course_name': e.section_id.course_id.name,
                'credits': e.section_id.course_id.credits,
                'semester_status': e.section_id.semester_id.status,
                'season': e.section_id.semester_id.season,
```

StudentViewStudentTranscript

ontrol

```
            'season': e.section_id.semester_id.season,
            'year': e.section_id.semester_id.year,
            'grade': e.grade
        })
    if grades == 0.0 and counter == 0:
        cumulative_gpa = 0.0
    else:
        cumulative_gpa = float(grades/counter)

    rendered = render_component(
        os.path.join(os.getcwd(), 'registration_system', 'static',
                     'registration_system', 'js', 'nav-holder.jsx'),
        {
            'is_student': self.is_student,
            'header_text': 'View Transcript'
        },
        to_static_markup=False,
    )

    context = {
        'rendered': rendered,
        'student_name': userprofile.user.first_name + '' + userprofile.user.last_name,
        'enrollment_array': enrollments_array,
        'birth_date': userprofile.student.date_of_birth.strftime("%Y-%m-%d"),
        'cumulative_gpa': cumulative_gpa,
        'student_advisers': adviser_array,
        'major_and_department': student_major + ' / ' + student_major_dep
    }

    return render(request, self.template_name, context)
```

RegisterCourse

Header	
Use Case ID	UC5
Use Case Version	1.0
Body	
Title	RegisterCourse
Actors	Student
Input	Course Name/ID
Output	Course registered
Normal Flow	1. Student is logged in and has navigated to the master schedule. 2. Student is presented all courses offered, along with the option to add the class to the schedule. 3. Student registers for the course from the master schedule. 4. The course will now be displayed in their student schedule as long there is no conflicts.
Alternate Flows	4a) The course the student tried to add to their schedule had some sort of constraint that prevented the student from registering (time constraint, seat availability, hold etc.)
Entry Condition	User is logged in.
Exit Condition	Course is registered for and added to student schedule.

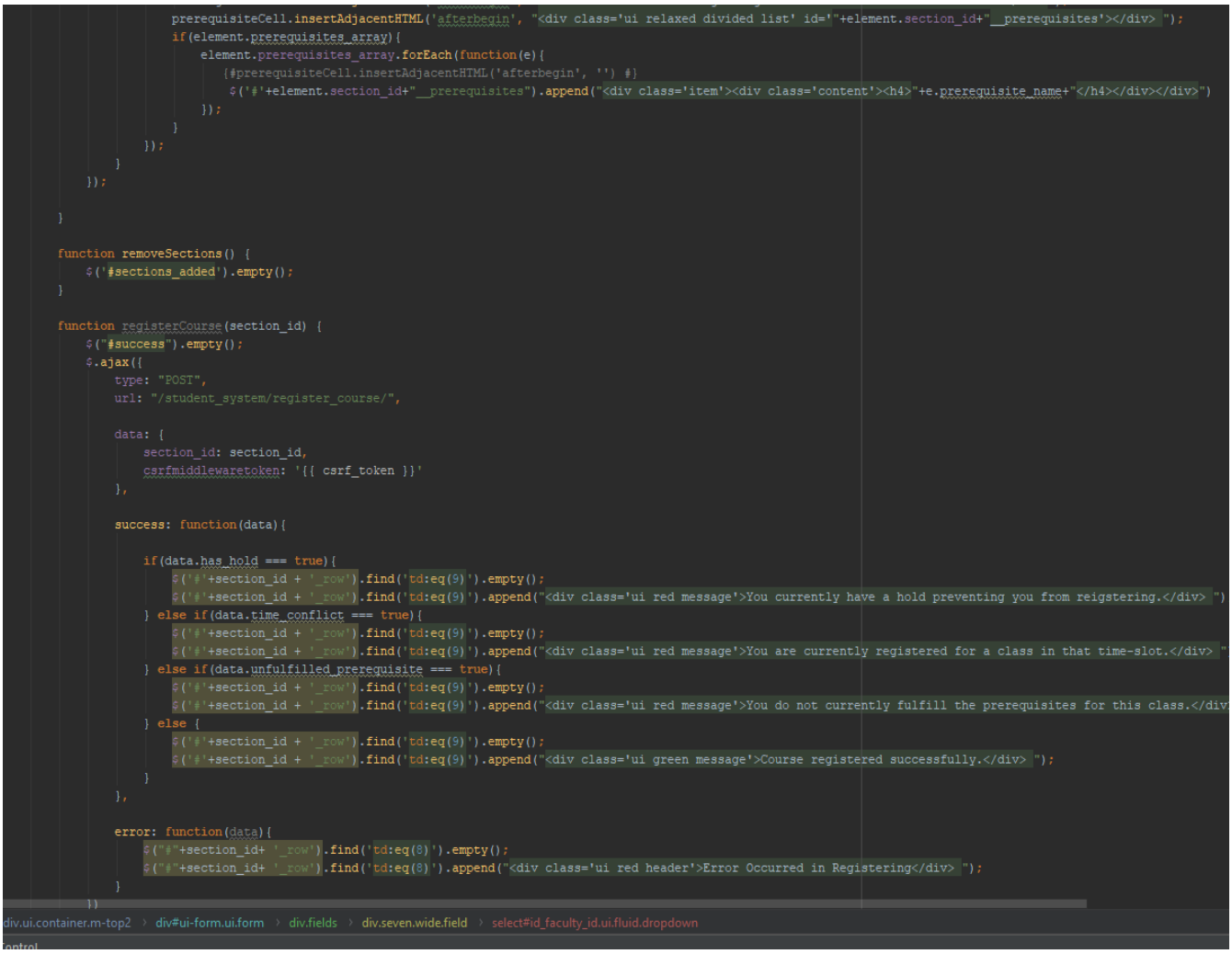
Front-end Implementation

- Students may search for sections using the Department, Course, Faculty, Meeting Days, or Time Period filters. Filters will further constrain courses for students to pinpoint specific sections.
- If the student has a hold, does not fulfill the prerequisite, or has a course registered during the same time slot, an error message is displayed to the user.

```
% extends "registration_system/partials/base.html" %}
% load staticfiles %}
% block navbar %}
   {% include "registration_system/partials/main_navbar.html" %}
% endblock %}
% block content %}

    <div class="ui container">
       {{ rendered|safe }}
   </div>

   <div class="ui container m-top2" style="...">
       <div class="ui form" id="ui-form">
           <div class="inline fields">
               <div class="field">
                   <label for="id_department_id">Department</label>
                   <select class="ui fluid dropdown" name="department_id" id="id_department_id" required>
                       <option value="" label="---------" selected></option>
                       {% if departments %}
                           {% for department in departments %}

                               <option value="{{ department.department_id }}" label="{{ department.name }}">{{ department.name }}</option>
                           {% endfor %}
                       {% endif %}
                   </select>
               </div>
           </div>
           <div class="inline fields">
               <div class="field">
                   <label for="id_course_id">Course</label>
                   <select class="ui fluid dropdown" name="course_id" id="id_course_id" required>
                        <option value="" label="---------" selected></option>
                       {% if courses %}
                           {% for course in courses %}
                               <option value="{{ course.course_id }}" label="{{ course.name }}">{{ course.name }}</option>
                           {% endfor %}
                       {% endif %}
                   </select>
               </div>
           </div>
           <div class="fields">
               <div class="seven wide field">
                   <label for="id_faculty_id">Faculty</label>
                   <select class="ui fluid dropdown" name="faculty_id" id="id_faculty_id" required>
                       <option value="" label="---------" selected></option>
                       {% if faculty %}
                           {% for f in faculty %}
                               <option value="{{ f.faculty_id }}" >{{ f.first_name }} {{ f.last_name }}</option>
                           {% endfor %}
                       {% endif %}
                   </select>
div.ui.container.m-top2 › div#ui-form.ui.form › table.ui.celled.table › thead › tr › th
```

```
            <div class="three wide field">
                <label for="id_days_id">Meeting Days</label>
                <select class="ui fluid dropdown" name="days_id" id="id_days_id" required>
                    <option value="" label="---------" selected></option>
                    {% if days %}
                        {% for d in days %}
                            <option value="{{ d.days_id }}" >{{ d.day_1 }}--{{ d.day_2 }}{% if d.day_3 is not None %}--{{ d.day_3 }}
                        {% endfor %}
                    {% endif %}
                </select>
            </div>
            <div class="three wide field">
                <label for="id_period_id">Time Period</label>
                <select class="ui fluid dropdown" name="period_id" id="id_period_id" required>
                    <option value="" label="---------" selected></option>
                    {% if time_periods %}
                        {% for t in time_periods %}
                            <option value="{{ t.period_id }}" >{{ t.start_time }}--{{ t.end_time }}</option>
                        {% endfor %}
                    {% endif %}
                </select>
            </div>
            <div class="three wide field" style="...">
                <button type="button" id="search_submit" class="ui button" role="button">Submit</button>
            </div>
        </div>
        <table class="ui celled table">
            <thead>
                <tr>
                    <th>Section</th>
                    <th>Course Name</th>
                    <th>Department</th>
                    <th>Professor</th>
                    <th>Credits</th>
                    <th>Seating</th>
                    <th>Time Slot</th>
                    <th>Location</th>
                    <th>Prerequisites</th>
                    <th>Semester</th>
                    <th>Register</th>
                </tr>
            </thead>
            <tbody id="sections_added">

            </tbody>
        </table>
    </div>
    <div id="success">

    </div>
div.ui.container.m-top2 › div#ui-form.ui.form › div.fields › div.seven.wide.field › select#id_faculty_id.ui.fluid.dropdown
```

```
</div>
{# TODO: Ajax call to building table and ajax post to submit update #}
    <script>
        function getSections(){
            removeSections();
            var department_id = $('#id_department_id').val();
            var department_name = $('select option:selected').text();
            var course_id = $('#id_course_id').val();
            var days_id = $('#id_days_id').val();
            var faculty_id = $('#id_faculty_id').val();
            var period_id = $('#id_period_id').val();
            $.ajax({
                type: "GET",
                url: "/student_system/register_course/",
                data : {
                    department_id: department_id,
                    department_name: department_name,
                    course_id: course_id,
                    days_id: days_id,
                    faculty_id: faculty_id,
                    period_id: period_id
                },
                success: function(data) {
                    console.log(data);
                    data.forEach(function(element){
                        var tBody = document.getElementById('sections_added');
                        var newRow = tBody.insertRow(tBody.rows.length);
                        newRow.id = element.section_id + '_row';
                        var sectionIdCell = newRow.insertCell(0);
                        var courseNameCell = newRow.insertCell(1);
                        var departmentCell = newRow.insertCell(2);
                        var facultyCell = newRow.insertCell(3);
                        var creditsCell = newRow.insertCell(4);
                        var seatsCell = newRow.insertCell(5);
                        var timeSlotCell = newRow.insertCell(6);
                        var locationCell = newRow.insertCell(7);
                        var prerequisiteCell = newRow.insertCell(8);
                        var semesterCell = newRow.insertCell(9);
                        var registerCell = newRow.insertCell(10);
{# editing faculty, meeeting days, time period, building, and room number  #}
                        var sectionIdText = document.createTextNode(element.section_id);
                        var courseNameText = document.createTextNode(element.course_name);
                        var departmentText = document.createTextNode(element.course_department);
                        var creditsText = document.createTextNode(element.credits);
                        var facultyText = document.createTextNode(element.faculty_name);
                        var semesterText = document.createTextNode(element.semester_info);
                        var seatsText = $("<ul style='list-style:none; padding-left:0px'><li><span style='font-weight:bold'>Taken:</span>"
                        sectionIdCell.appendChild(sectionIdText);
                        courseNameCell.appendChild(courseNameText);
                        departmentCell.appendChild(departmentText);
                        semesterCell.appendChild(semesterText);
```



```
                prerequisiteCell.insertAdjacentHTML('afterbegin', "<div class='ui relaxed divided list' id='"+element.section_id+"__prerequisites'></div> ");
                if(element.prerequisites_array){
                    element.prerequisites_array.forEach(function(e){
                        {#prerequisiteCell.insertAdjacentHTML('afterbegin', '') #}
                        $('#'+element.section_id+"__prerequisites").append("<div class='item'><div class='content'><h4>"+e.prerequisite_name+"</h4></div></div>")
                    });
                }
            });
        }
    });

}

function removeSections() {
    $('#sections_added').empty();
}

function registerCourse(section_id) {
    $("#success").empty();
    $.ajax({
        type: "POST",
        url: "/student_system/register_course/",

        data: {
            section_id: section_id,
            csrfmiddlewaretoken: '{{ csrf_token }}'
        },

        success: function(data){

            if(data.has_hold === true){
                $('#'+section_id + '_row').find('td:eq(9)').empty();
                $('#'+section_id + '_row').find('td:eq(9)').append("<div class='ui red message'>You currently have a hold preventing you from reigstering.</div> ")
            } else if(data.time_conflict === true){
                $('#'+section_id + '_row').find('td:eq(9)').empty();
                $('#'+section_id + '_row').find('td:eq(9)').append("<div class='ui red message'>You are currently registered for a class in that time-slot.</div> "
            } else if(data.unfulfilled_prerequisite === true){
                $('#'+section_id + '_row').find('td:eq(9)').empty();
                $('#'+section_id + '_row').find('td:eq(9)').append("<div class='ui red message'>You do not currently fulfill the prerequisites for this class.</div
            } else {
                $('#'+section_id + '_row').find('td:eq(9)').empty();
                $('#'+section_id + '_row').find('td:eq(9)').append("<div class='ui green message'>Course registered successfully.</div> ");
            }
        },

        error: function(data){
            $("#"+section_id+ '_row').find('td:eq(8)').empty();
            $("#"+section_id+ '_row').find('td:eq(8)').append("<div class='ui red header'>Error Occurred in Registering</div> ");
        }
    })
div.ui.container.m-top2 › div#ui-form.ui.form › div.fields › div.seven.wide.field › select#id_faculty_id.ui.fluid.dropdown
```

Back-end Implementation

- If an HTTP GET request is sent through Ajax all sections related to the students search query are fetched and propagated back to the front-end through JSON data to be populated into the front-end table structure.
- If an HTTP POST request is sent, the system first verifies that the student does not have a hold placed, time conflict, or unfulfilled prerequisite. If one of the above constraints is present the system sends back an error message to be displayed to the user, if not the student is registered into the section successfully and displayed a success message on the front-end.

```python
class RegisterCourse(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/register_course.html'
    is_student = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)

        if userprofile:
            if userprofile.has_student():
                self.is_student = True
            else:
                redirect('/student_system/')

        if request.is_ajax():
            department_id = request.GET.get('department_id')
            department_name = request.GET.get('department_name')
            course_id = request.GET.get('course_id')
            days_id = request.GET.get('days_id')
            faculty_id = request.GET.get('faculty_id')
            period_id = request.GET.get('period_id')
            section = None
            # print(department_id)
            # print(department_name)
            # if department_id == '---------' and course_id == '---------' and
            if department_id is not None and department_id != '':
                section = Section.objects.filter(course_id__department_id_id=int(department_id))
            elif course_id is not None and course_id != '':
                section = Section.objects.filter(course_id=int(course_id))
            elif days_id is not None and days_id != '':
                section = Section.objects.filter(time_slot_id__days_id=int(days_id))
            elif faculty_id is not None and faculty_id != '':
                section = Section.objects.filter(faculty_id=int(faculty_id))
            elif period_id is not None and period_id != '':
                section = Section.objects.filter(time_slot_id__period_id=int(period_id))
            else:
                section = Section.objects.all()

            student_id = user.userprofile.student.student_id_id
            section_array = []
            for s in section:
                # print(s)
                # print("here\n")
                can_register = True
                prerequisites_array = []
                for p in Prerequisite.objects.filter(course_id=s.course_id):
                    prerequisites_array.append({
                        'prerequisite_name': p.course_required_id.name,
```

RegisterCourse

ntrol

```
                prerequisites_array.append({
                    'prerequisite_name': p.course_required_id.name,
                    'prerequisite_id': p.course_required_id.course_id
                })
            for e in Enrollment.objects.all():
                if e.student_id_id == student_id:
                    if e.section_id.time_slot_id.period_id.start_time == s.time_slot_id.period_id.start_time or e.section_id.time_slot_id.p
                        can_register = False
            data = {
                'section_id': s.section_id,
                'semester_info': s.semester_id.season + '-' + str(s.semester_id.year) + '--' + s.semester_id.status,
                'course_id': s.course_id.course_id,
                'course_department': s.course_id.department_id.name,
                'course_name': s.course_id.name,
                'faculty_id': s.faculty_id.faculty_id_id,
                'faculty_name': s.faculty_id.faculty_id.user.first_name + " " + s.faculty_id.faculty_id.user.last_name,
                'credits': s.course_id.credits,
                'seats_taken': s.seats_taken,
                'seating_capacity': s.room_id.capacity,
                'time_slot_id': s.time_slot_id.time_slot_id,
                'prerequisites_array': prerequisites_array,
                'time_period_id': s.time_slot_id.period_id.period_id,
                'time_period_range': s.time_slot_id.period_id.start_time.strftime('%H:%M %p')
                                     + " " + s.time_slot_id.period_id.end_time.strftime('%H:%M %p'),
                'meeting_days_id': s.time_slot_id.days_id.days_id,
                'meeting_days': s.time_slot_id.days_id.day_1 + " " + s.time_slot_id.days_id.day_2,
                'building_id': s.room_id.building_id.building_id,
                'building_name': s.room_id.building_id.name,
                'room_id': s.room_id.room_id,
                'room_number': s.room_id.room_number,
                'can_register': can_register
            }
            section_array.append(data)

        return HttpResponse(json.dumps(section_array), content_type="application/json")

    rendered = render_component(
        os.path.join(os.getcwd(), 'registration_system', 'static',
                     'registration_system', 'js', 'nav-holder.jsx'),
        {
            'is_student': self.is_student,
            'header_text': 'Register Courses'
        },
        to_static_markup=False,
    )
    departments = Department.objects.all()
    days = MeetingDays.objects.all()
    course = Course.objects.all()
    time_periods = Period.objects.all()
    faculty = []
```

RegisterCourse

```
        for f in Faculty.objects.raw("SELECT u.first_name, u.last_name, f.faculty_id_id "
                                     "FROM registration_system_faculty AS f, auth_user as u, registration_system_userprofile as up "
                                     "WHERE up.user_id = u.id "
                                     "AND up.id = f.faculty_id_id"):
            faculty.append({
                'first_name': f.first_name,
                'last_name': f.last_name,
                'faculty_id': f.faculty_id_id
            })

        context = {
            'rendered': rendered,
            'departments': departments,
            'days': days,
            'faculty': faculty,
            'courses': course,
            'time_periods': time_periods
        }
        return render(request, self.template_name, context)

    def post(self, request, *args, **kwargs):
        data = {
            'is_successful': False
        }
        if request.is_ajax():
            user = request.user
            userprofile = UserProfile.objects.get(user=user)
            student = Student.objects.get(pk=userprofile.student.student_id_id)
            section_id = request.POST.get('section_id')
            section = Section.objects.get(pk=int(section_id))
            try:
                student_hold = StudentHold.objects.get(student_id=student)
                data['is_successful'] = False
                data['has_hold'] = True
            except StudentHold.DoesNotExist:
                current_enrollments = Enrollment.objects.filter(student_id=student)
                for e in current_enrollments:
                    if e.section_id.time_slot_id == section.time_slot_id:
                        data['is_successful'] = False
                        data['time_conflict'] = True
                        return JsonResponse(data)
                if section.course_id.has_prerequisites():
                    prerequisites = Prerequisite.objects.filter(course_id=section.course_id)
                    prerequisite_fulfilled = False
                    for e in current_enrollments:
                        for p in prerequisites:
                            if p.course_required_id == e.section_id.course_id:
                                prerequisite_fulfilled = True
```

RegisterCourse › get()

```
                                prerequisite_fulfilled = True
                                break
                        else:
                            continue
                        break
                    if not prerequisite_fulfilled:
                        data['is_successful'] = False
                        data['unfulfilled_prerequisite'] = True
                        # if e.section_id.course_id == section.course_id.p
                enrollment = Enrollment.objects.create(student_id=student, section_id=section)
                # meeting = Meetings.objects.create(enrollment_id=enrollment, student_id=student)
                student_history = StudentHistory.objects.create(student_id=student, enrollment_id=enrollment)
                data['is_successful'] = True
            else:
                data['is_successful'] = False
            return JsonResponse(data)
```

Update/DropCourse

Header	
Use Case ID	UC6
Use Case Version	1.0
Body	
Title	Update/DropCourse
Actors	Student
Input	Course Name/ID
Output	Course removed
Normal Flow	1. Student is logged in and has navigated to their own student schedule. 2. Student is displayed each course they have registered for as well as the option to remove the course from their schedule. 3. Student chooses to remove course. 4. Course is removed schedule is updated and refreshed with the up-to-date schedule.
Alternate Flows	N/A
Entry Condition	User is logged in and course is not currently being taught.
Exit Condition	Courser removed from student schedule.

Front-end Implementation

- Sections enrolled in the current OPEN REGISTRATION semester can only be dropped or updated.
- A Close icon is located on the Section segment to allow the user to drop the course.
- If successful a success message is displayed verifying the course was dropped successfully

```
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

    <div class="ui container">
      {{ rendered|safe }}
    </div>

    <div class="ui container m-bottom4 m-top2">
        {% if sections %}
            {% for s in sections %}
                <div class="ui segments" id="{{ s.section_id }}_segment">
                    <div class="ui segment">
                      <h1 style="...">{{ s.course_name }}</h1>
                        <span class="ui right aligned float-right course-close" style="...">
                            <i id="{{ s.section_id }}" onclick="dropCourse(this.id)" class="far fa-times-circle " style="..."></i>
                        </span>
                    </div>
                    <div class="ui segments">
                        <h5 class="ui attached header">
                          Professor
                        </h5>
                        <div class="ui attached segment">
                          <p>{{ s.professor }}</p>
                        </div>
                        <h5 class="ui attached header">
                          Credits
                        </h5>
                        <div class="ui attached segment">
                          <p>{{ s.credits }}</p>
                        </div>
                        <h5 class="ui attached header">
                            Location
                        </h5>
                        <div class="ui attached segment">
                          <p>{{ s.building }} -- {{ s.room_number }}</p>
                        </div>
                        <h5 class="ui attached header">
                            Time Slot
                        </h5>
                        <div class="ui attached segment">
                          <p>{{ s.meeting_days }}</p><br/>
                            <p>{{ s.time_period }}</p>
                        </div>
                        <h5 class="ui attached header">
                            Seating
                        </h5>
                        <div class="ui attached segment">
```

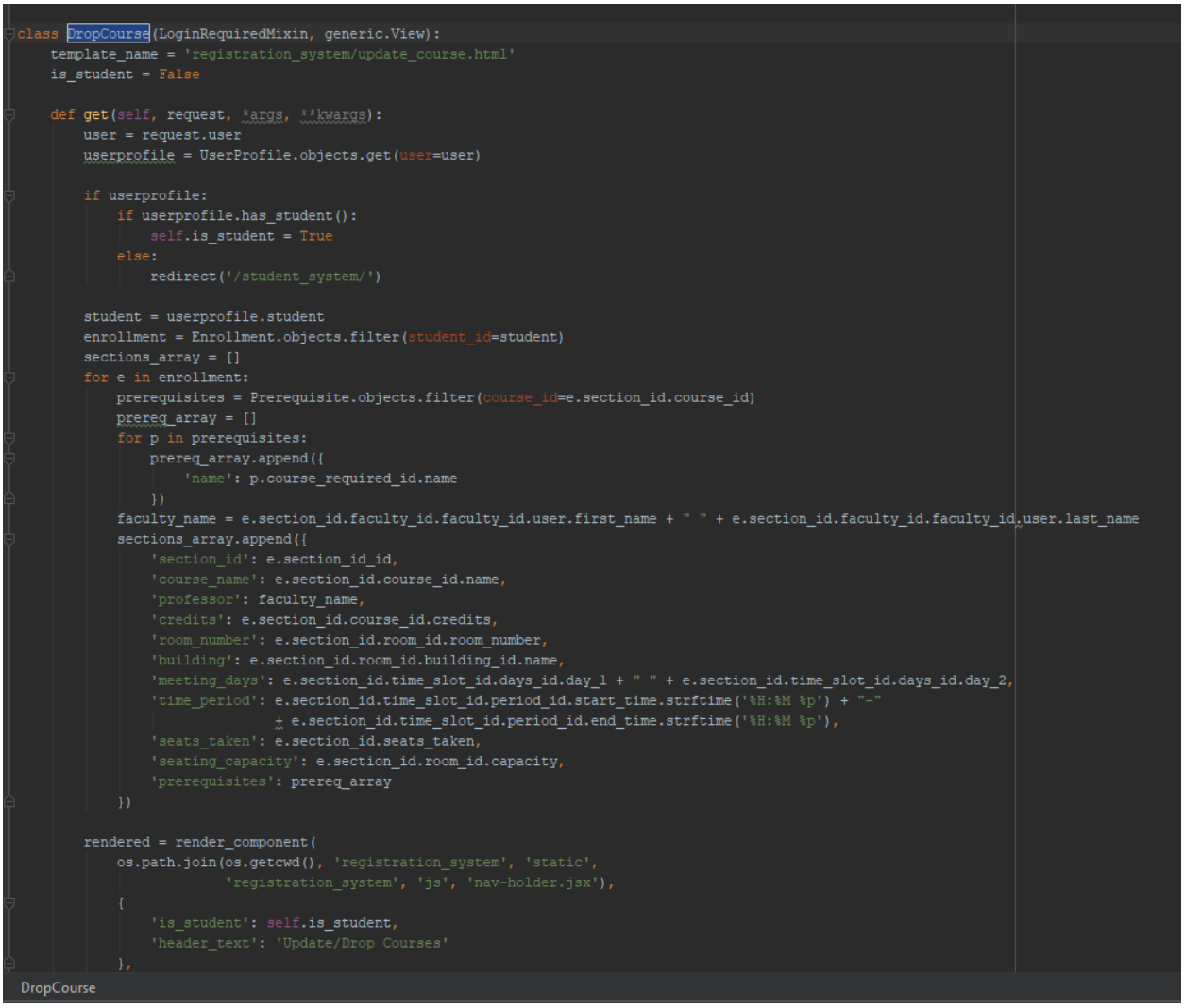
div.ui.container.m-bottom4.m-top2 › div.ui.segments › div.ui.segment › span.ui.right.aligned.float-right.course-close

Control

```
                        </h5>
                        <div class="ui attached segment">
                            <div class="ui celled list">
                            {% for p in s.prerequisites %}
                                <div class="item">
                                    <div class="content">
                                        <div class="header">{{ p.name }}</div>
                                    </div>
                                </div>
                            {% endfor %}
                            </div>
                        </div>
                    </div>
                {% endfor %}
            {% endif %}

            <div id="success" class="ui basic modal">
              <div id="id_modal_section" class="ui green header">

              </div>
            </div>
        </div>
    {# TODO: Ajax call to building table and ajax post to submit update #}
        <script>

            function dropCourse(section_id) {
                $.ajax({
                    type: "POST",
                    url: "/student_system/drop_course/",

                    data: {
                        section_id: section_id,
                        csrfmiddlewaretoken: '{{ csrf_token }}'
                    },

                    success: function(){
                        $('.ui.modal').modal('show');
                        $('#id_modal_section').empty();
                        $('#id_modal_section').append("Course Dropped successfully!\n" );

                        $('#'+section_id+"_segment").empty();
                    }
                })

            }



        </script>
```

Back-end Implementation

- POST route uses the data sent through the AJAX request to remove the student's enrollment in the section.



```
class DropCourse(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/update_course.html'
    is_student = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)

        if userprofile:
            if userprofile.has_student():
                self.is_student = True
            else:
                redirect('/student_system/')

        student = userprofile.student
        enrollment = Enrollment.objects.filter(student_id=student)
        sections_array = []
        for e in enrollment:
            prerequisites = Prerequisite.objects.filter(course_id=e.section_id.course_id)
            prereq_array = []
            for p in prerequisites:
                prereq_array.append({
                    'name': p.course_required_id.name
                })
            faculty_name = e.section_id.faculty_id.faculty_id.user.first_name + " " + e.section_id.faculty_id.faculty_id.user.last_name
            sections_array.append({
                'section_id': e.section_id_id,
                'course_name': e.section_id.course_id.name,
                'professor': faculty_name,
                'credits': e.section_id.course_id.credits,
                'room_number': e.section_id.room_id.room_number,
                'building': e.section_id.room_id.building_id.name,
                'meeting_days': e.section_id.time_slot_id.days_id.day_1 + " " + e.section_id.time_slot_id.days_id.day_2,
                'time_period': e.section_id.time_slot_id.period_id.start_time.strftime('%H:%M %p') + "-"
                               + e.section_id.time_slot_id.period_id.end_time.strftime('%H:%M %p'),
                'seats_taken': e.section_id.seats_taken,
                'seating_capacity': e.section_id.room_id.capacity,
                'prerequisites': prereq_array
            })

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_student': self.is_student,
                'header_text': 'Update/Drop Courses'
            },
```

DropCourse

```
                'header_text': 'Update/Drop Courses'
            },
            to_static_markup=False,
        )

        context = {
            'rendered': rendered,
            'sections': sections_array
        }
        return render(request, self.template_name, context)

    def post(self, request, *args, **kwargs):
        data = {
            'is_successful': False
        }
        if request.is_ajax():
            user = request.user
            userprofile = UserProfile.objects.get(user=user)
            student = Student.objects.get(pk=userprofile.student.student_id_id)
            section_id = request.POST.get('section_id')
            section = Section.objects.get(pk=int(section_id))
            enrollment = Enrollment.objects.get(student_id=student, section_id=section)
            enrollment.delete()
            data['is_successful'] = True
        else:
            data['is_successful'] = False
        return JsonResponse(data)
```

DeclareMinor/Major

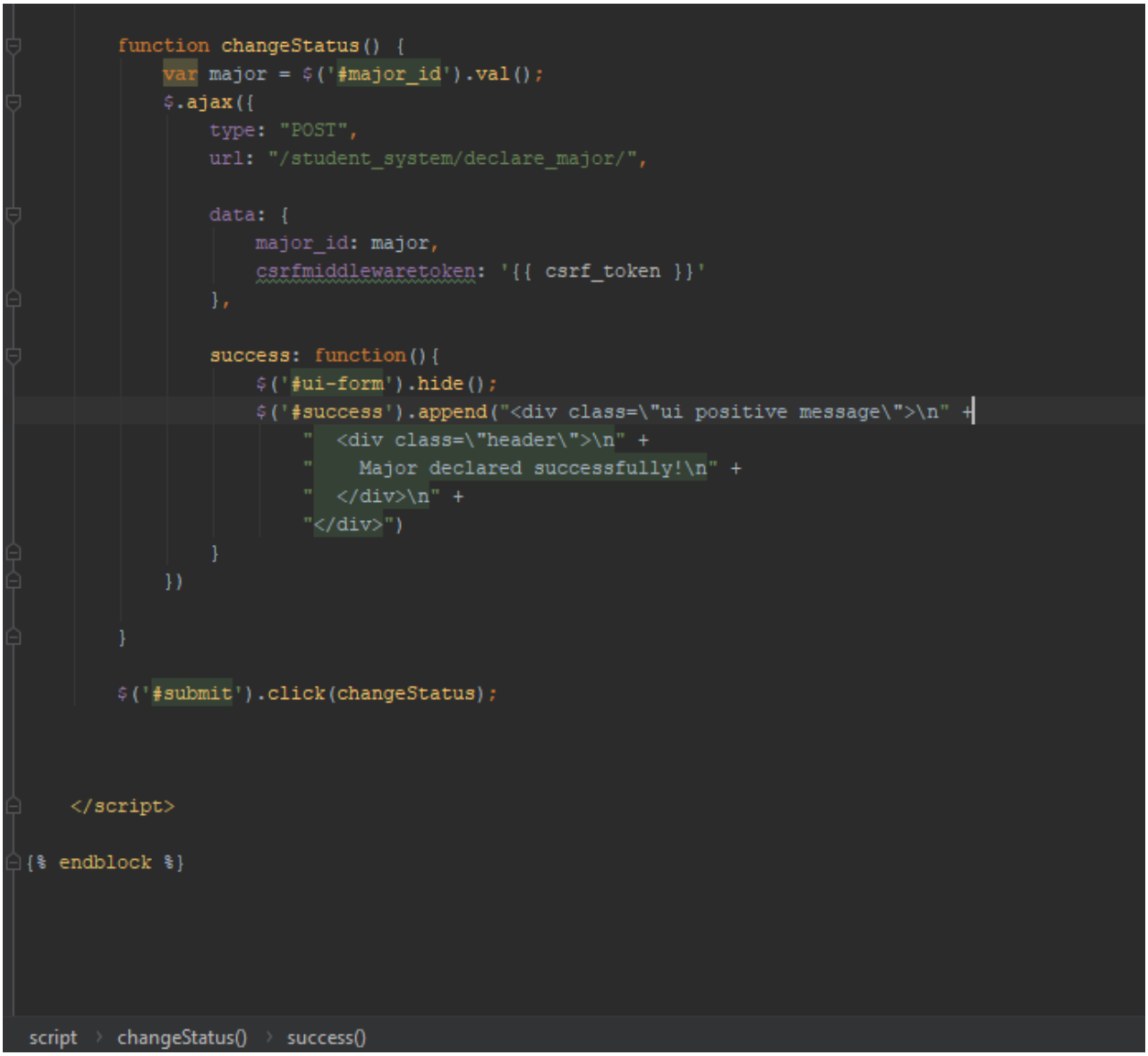
Header	
Use Case ID	UC7
Use Case Version	1.0
Body	
Title	DeclareMajor/Minor
Actors	Student
Input	Student ID
Output	Major/Minor Declared
Normal Flow	1. Student is logged in and has navigated to the Declare Major/Minor Option from the menu. 2. System displays Dropdown of majors/minors are displayed to the student. 3. Student chooses a Major/Minor and presses submit. 4. System inserts Major/Minor declaration into database.
Alternate Flows	
Entry Condition	User is logged in.
Exit Condition	Graduation Audit displayed to user.

Front-end Implementation

- Students may choose from the dropdown of Majors/Minors offered at Andromeda College.
- Upon submit the major will be declared, student is unable to change major again without administrative approval.
- Minors may be added as a student can take up to 2 minors in total.

```
{% endblock %}
{% block content %}

    <div class="ui container">
        {{ rendered|safe }}
    </div>

    <div class="ui container m-top2">

        {% if student_major %}
            <div class="ui feed">
              <div class="event">
                <div class="label">
                  <i class="fas fa-graduation-cap"></i>
                </div>
                <div class="content">
                    You are currently a {{ student_major }} major.
                </div>
              </div>
            </div>
        {% else %}
            <div class="ui form" id="ui-form">
            {{ csrf }}
                    <div class="field">
                        <label for="major_id">Majors</label>
                        <select id="major_id" class="ui fluid dropdown">
                            <option value="0">--------</option>
                            {% if majors %}
                                {% for m in majors %}
                                    <option value="{{ m.major_id }}" >{{ m.name }}</option>
                                {% endfor %}
                            {% endif %}
                        </select>
                    </div>

                <div class="display-inline ">
                    <button class="ui primary button" id="submit" type="button">Submit</button>
                </div>
            </div>
            <div id="success">

            </div>
        {% endif %}
    </div>

    <script>

        function changeStatus() {
            var major = $('#major_id').val();
            $.ajax({
```

cript › changeStatus() › success()



```
        function changeStatus() {
            var major = $('#major_id').val();
            $.ajax({
                type: "POST",
                url: "/student_system/declare_major/",

                data: {
                    major_id: major,
                    csrfmiddlewaretoken: '{{ csrf_token }}'
                },

                success: function(){
                    $('#ui-form').hide();
                    $('#success').append("<div class=\"ui positive message\">\n" +
                        "  <div class=\"header\">\n" +
                        "    Major declared successfully!\n" +
                        "  </div>\n" +
                        "</div>")
                }
            })

        }

        $('#submit').click(changeStatus);



    </script>

{% endblock %}
```

script › changeStatus() › success()

Back-end Implementation

- POST route receives the Student ID and the ID of the major desired then utilizes both to create the StudentMajor record in the database.

```python
class DeclareMajor(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/declare_major.html'
    is_student = False

    def get(self, request, *args, **kwargs):
        user = request.user
        userprofile = UserProfile.objects.get(user=user)

        if userprofile:
            if userprofile.has_student():
                self.is_student = True
            else:
                redirect('/student_system/')

        majors = Major.objects.all()
        has_major = True
        try:
            student = Student.objects.get(pk=int(user.userprofile.student.student_id_id))
            student_major = StudentMajor.objects.get(student_id=student).major_id.name
        except StudentMajor.DoesNotExist:
            has_major = False
            student_major = False

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_student': self.is_student,
                'header_text': 'Declare Major'
            },
            to_static_markup=False,
        )

        context = {
            'rendered': rendered,
            'majors': majors,
            'student_major': student_major
        }

        return render(request, self.template_name, context)

    def post(self, request, *args, **kwargs):
        data = {
            'is_successful': False
        }
        if request.is_ajax():
            user = request.user
```

DeclareMajor

```
def post(self, request, *args, **kwargs):
    data = {
        'is_successful': False
    }
    if request.is_ajax():
        user = request.user
        userprofile = UserProfile.objects.get(user=user)
        student = Student.objects.get(pk=userprofile.student.student_id_id)
        major_id = request.POST.get('major_id')
        major = Major.objects.get(pk=int(major_id))
        StudentMajor.objects.create(student_id=student, major_id=major)
        data['is_successful'] = True
    else:
        data['is_successful'] = False
    return JsonResponse(data)
```

Researcher Implementation:

ViewGraphs

Header	
Use Case ID	UC19
Use Case Version	1.0
Body	
Title	VIewGraphs
Actors	Researcher
Normal Flow	1. Researcher logs into the system. 2. The system provides the researcher with their own navigation menu 3. Researcher chooses the ViewGraphs Menu Item. 4. System fetches overall grading and average attendance graphs.
Alternate Flows	N/A
Entry Condition	Must have been given permission by the administrator to view reports.

Front-end Implementation

- Through server-side rendering data is delivered to the front-end templates and graphs are formulated using Plotly.js (Graphing Library) to display a visual model of the data.
- One graph displays the overall grading percentages, second graph displays the most registered for courses.

```
<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
{% extends "registration_system/partials/base.html" %}
{% load staticfiles %}
{% block navbar %}
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}
{#    {% csrf_token %}#}
    <div class="ui container">
        {{ rendered|safe }}
    </div>
    <h1>Student Overrall Grading Graph:</h1>
    <div id="student_plot" style="..." class="ui container">

    </div>

    <h1>Student Attendance Graph</h1>
    <div id="attendance_plot" style="..." class="ui container">

    </div>

    <script>
        let ATTENDANCE_PLOT = document.getElementById('attendance_plot');
        Plotly.plot(ATTENDANCE_PLOT, [{
            type: 'bar',
            x: {{ meeting_dates_x_axis|safe }},
            y: {{ meeting_dates_y_axis|safe }}
        }], {
            margin: { t: 0}
        });
        let PLOT = document.getElementById('student_plot');
        Plotly.plot(PLOT, [{
            type: 'bar',
            x: {{ x_axis|safe }},
            y: {{ y_axis|safe }}
        }], {
            margin: { t: 0}
        });
    </script>

    <script>

    </script>
{#    {{ form }}#}

{#    <script>document.getElementsByClassName('ui dropdown').dropdown()</script>#}
{% endblock %}
```

Back-end Implementation

- Both graphs X and Y axis are formulated and packaged into arrays to be propagated as JSON data to the front-end to populate Plotly graphs.

```
class ViewGraphs(LoginRequiredMixin, generic.View):
    template_name = 'registration_system/view_graphs.html'
    is_researcher = False
    grading_key = {
        'A': 4.0,
        'A-': 3.5,
        'B': 3.0,
        'B-': 2.5,
        'C': 2.0,
        'C-': 1.5,
        'D': 1.0,
        'D-': 0.5,
        'F': 0
    }

    def get(self, request, *args, **kwargs):
        user = request.user
        user_profile = UserProfile.objects.get(user=user)

        if user_profile:
            if user_profile.has_researcher():
                self.is_researcher = True
            else:
                redirect('/student_system/')

        rendered = render_component(
            os.path.join(os.getcwd(), 'registration_system', 'static',
                         'registration_system', 'js', 'nav-holder.jsx'),
            {
                'is_researcher': self.is_researcher,
                'header_text': 'View Graphs '
            },
            to_static_markup=False,
        )
        grades_tally = [0] * 9
        grades = [4.0, 3.5, 3.0, 2.5, 2.0, 1.5, 1.0, 0.5, 0]
        grades = ['A', 'A-', 'B', 'B-', 'C', 'C-', 'D', 'D-', 'F']
        for e in Enrollment.objects.all():
            if e.grade == 'A':
                grades_tally[0] = grades_tally[0] + 1
            elif e.grade == 'A-':
                grades_tally[1] = grades_tally[1] + 1
            elif e.grade == 'B':
                grades_tally[2] = grades_tally[2] + 1
            elif e.grade == 'B-':
                grades_tally[3] = grades_tally[3] + 1
            elif e.grade == 'C':
                grades_tally[4] = grades_tally[4] + 1
```

```
            grades_tally[4] = grades_tally[4] + 1
        elif e.grade == 'C-':
            grades_tally[5] = grades_tally[5] + 1
        elif e.grade == 'D':
            grades_tally[6] = grades_tally[6] + 1
        elif e.grade == 'D-':
            grades_tally[7] = grades_tally[7] + 1
        elif e.grade == 'F':
            grades_tally[8] = grades_tally[8] + 1

    meeting_dates = {}
    for m in Meetings.objects.values('meeting_date').distinct():
        # print(m)
        meeting_dates[m['meeting_date'].strftime("%Y-%m-%d")] = {
            'date': m['meeting_date'].strftime("%Y-%m-%d"),
            'tally': 0
        }
    for m in Meetings.objects.all():
        if m.meeting_date.strftime("%Y-%m-%d") == meeting_dates[m.meeting_date.strftime("%Y-%m-%d")]['date']:
            meeting_dates[m.meeting_date.strftime("%Y-%m-%d")]['tally'] = meeting_dates[m.meeting_date.strftime("%Y-%m-%d")]['tally
    print(meeting_dates)
    meeting_date_x_axis = []
    meeting_date_y_axis = []
    for key, value in meeting_dates.items():
        meeting_date_x_axis.append(value['date'])
        meeting_date_y_axis.append(value['tally'])

    context = {
        'rendered': rendered,
        'x_axis': grades,
        'y_axis': grades_tally,
        'meeting_dates_x_axis': meeting_date_x_axis,
        'meeting_dates_y_axis': meeting_date_y_axis
    }

    return render(request, self.template_name, context)
```

CreateReports

Header	
Use Case ID	UC19
Use Case Version	1.0
Body	
Title	CreateReports
Actors	Researcher
Normal Flow	1. Researcher logs into the system. 2. The system provides the researcher with their own navigation menu 3. Researcher chooses the CreateReport Menu Item. 4. System displays dropdown menu to choose which report to generate: grading report or attendance report. 5. Researcher chooses a report to create. 6. System downloads csv report onto user's system
Alternate Flows	N/A
Entry Condition	Must have been given permission by the administrator to view reports.

Front-end Implementation

- Researcher is displayed a dropdown to choose which report to generate.
- Choices are: Grading Report or Registered Courses report.
- Upon submission a CSV file is downloaded onto the user's system.

```
    {% include "registration_system/partials/main_navbar.html" %}
{% endblock %}
{% block content %}

     <div class="ui container">
        {{ rendered|safe }}
    </div>

    <div class="ui container m-top2">

            <div class="ui form" id="ui-form">
            {{ csrf }}
                    <div class="field">
                        <label for="report_id">Create Report</label>
                        <select id="report_id" class="ui fluid dropdown">
                            <option value="0">--------</option>
                            <option value="grade_report">Grade Report</option>
                            <option value="generate_stats">Generate Statistical Analysis</option>
                            <option value="attendance_report">Attendance Report</option>
                        </select>
                    </div>
                <form method="get" action="{{ url }}/get_csv_report/">
                    {% csrf_token %}
                    <div class="display-inline ">
                        <button class="ui primary button" id="submit" type="button">Download</button>
                    </div>
                </form>
                <a id="grade_report_link" style="..." href="/student_system/create_report/get_csv_report/"></a>

            </div>
            <div id="success">

            </div>
    </div>

    <script>

        function getReport() {
            var report_id = $('#report_id').val();
            switch(report_id){
                case 'grade_report':
                    window.location = '/student_system/create_report/get_csv_report/';
                    break;
                case 'attendance_report':
                    window.location = '/student_system/create_report/get_attendance_csv_report/';
                    break;
            }

        }

        $('#submit').click(getReport);
script  >  getReport()
```

Back-end Implementation

- All data to create csv reports are formulated and organized in the backend. Once formulated data is sent back in csv format to the user to be downloaded by their browser client.

```python
# TODO: Finish CSV Report
def get_csv_report(request):
    grading_key = {
        'A': 4.0,
        'A-': 3.5,
        'B': 3.0,
        'B-': 2.5,
        'C': 2.0,
        'C-': 1.5,
        'D': 1.0,
        'D-': 0.5,
        'F': 0
    }
    response = HttpResponse(content_type='text/csv')
    response['Content-Disposition'] = 'attachment; filename="somefilename.csv"'
    writer = csv.writer(response)
    writer.writerow(['Student Name', 'Grade'])

    for s in Student.objects.all():
        enrollments = Enrollment.objects.filter(student_id=s)
        # enrollments_array = []
        grades = 0.0
        counter = 0
        for e in enrollments:
            if e.grade != 'I' and e.grade != 'W' and e.grade != 'NA':
                grades += grading_key[e.grade]

            counter = counter + 1

        if grades == 0.0 and counter == 0:
            cumulative_gpa = 0.0
        else:
            cumulative_gpa = float(grades / counter)
        writer.writerow([s.student_id.user.first_name + ' ' + s.student_id.user.last_name, cumulative_gpa])

    return response
```

```python
def get_attendance_csv_report(request):
    response = HttpResponse(content_type='text/csv')
    response['Content-Disposition'] = 'attachment; filename="somefilename.csv"'
    writer = csv.writer(response)
    writer.writerow(['Date ', 'Class', 'Tally'])

    meeting_dates = {}
    for m in Meetings.objects.values('meeting_date').distinct():
        # print(m)
        meeting_dates[m['meeting_date'].strftime("%Y-%m-%d")] = {
            'date': m['meeting_date'].strftime("%Y-%m-%d"),
            'tally': 0,
            'class': None
        }
    for m in Meetings.objects.all():
        if m.meeting_date.strftime("%Y-%m-%d") == meeting_dates[m.meeting_date.strftime("%Y-%m-%d")]['date']:
            meeting_dates[m.meeting_date.strftime("%Y-%m-%d")]['tally'] = meeting_dates[m.meeting_date.strftime("%Y-%m-%d")]['tally'
            meeting_dates[m.meeting_date.strftime("%Y-%m-%d")]['class'] = m.enrollment_id.section_id.course_id.name

    # print(meeting_dates)

    for key, value in meeting_dates.items():
        writer.writerow([value['date'], value['class'], value['tally']])
    return response
```