**Design & Implementation of a Dynamic Web-Based Registration System for U.A. University**

Systems Design And Implementation

Professor Naresh Gupta

| **Hunter Oriol Email: horiol@oldwestbury.edu Phone #: (516)941-8490 Team Leader**  Ganesh Ramcharan Email: gramchar@oldwestbury.edu Phone #: (347)369-1881  Jeriel Salvador Email: jsalvado@oldwestbury.edu Phone #: (631)807-1049 |
| --- |

**Table of Contents**

Content Pg No.

**1. Meeting Log 4-12**

**2. Problem Statement 13**

**3. System Requirements Specification 14-23**

3.1 Introduction 14

*3.1.1 Purpose*  14

*3.1.2 Scope*  14

3.2 Specific Requirements 14-16

*3.2.1 Student User Requirements* 14-16

3.2.1a Explicitly Required Specifications 14-16

3.2.1b Forbidden SRS 16

*3.2.2 Admin User Requirements 17-20*

3.2.2a Explicitly Required Specifications 17-19

3.2.2b Forbidden SRS 19-20

*3.2.3 Faculty User Requirements 20-22*

3.2.3a Explicitly Required Specifications 20-21

3.2.3b Forbidden SRS 21-22

*3.2.4 Statistics Office Requirements 22-23*

3.2.4a Explicitly Required Specifications 22-23

**4. Use Case Tables 24-92**

4.1 Universal 24-27

4.2 Student 28-38

4.3 Admin 39-72

4.4 Faculty 72-86

4.5 Statistics Office 86-92

**5. Enhanced Entity Relationship Diagram 93-94**

**6. Schema 95-141**

**7. Glossary 142**

**1. Meeting Log**

| **Location** | **Date/Time** | **Members** | **Present** | **Discussed Topics** | **Decisions Made** |
| --- | --- | --- | --- | --- | --- |
| Discord (Virtual) | 09/09/2023 Start: 8:40PM  Finish: 9:30pm | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | First Meeting. Discussion of assignment due Tuesday. | Email professor for more clarification on assignment. Create Problem Statement on Monday if no answer is received. |
| Discord (Virtual) | 9/11/2023 Start:5:50PM  Finish: 7:00pm | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Problem Statement Assignment | Work was performed on the cover page and problem statement. |
| Class | 09/12/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Staff Office, Use Cases, Safety Issues |  |
| Discord (Virtual) | 09/13/2023  Start: 6:00 PM  Finish: 9:00pm | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔    ✔ | SRS Assignment | Revision of problem statement. Addition of Table of Contents, Meeting Log, and SRS sections |
| Class | 09/14/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Overview of Parts needed for System Manual |  |
| Discord(Virtual) | 09/16/2023 Start: 7:00PM  Finish : 8:30PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Use Case Tables | Split Use Case Table work into four sections and have each member do one section and work together on the Statistic Office Use Cases on a later date |
| Discord(Virtual) | 09/18/2023 Start: 7:30PM  Finish: 9:00PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Use Case Tables | Addition/Revision of Use Case Tables |
| Class | 09/19/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Use Case Tables, ER Diagram | Further revision of Use Case Tables, Start ER Diagrams |
| Class | 09/21/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | ER Diagram Overview |  |
| Discord(Virtual) | 9/24/2023  Start: 1:00PM  Finish: 2:30PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  **X** | ER Diagrams | Jeriel is absent due to a power outage. Create ER Diagrams draft. |
| Class | 09/26/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | ER Diagrams Revision |  |
| Discord(Virtual) | 9/27/2023  Start: 7:00PM  Finish: 9:00PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | ER Diagrams revision, previous sections’ revision | Revise ER diagrams, reprint the whole document packet after corrections. |
| Class | 09/28/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | ER Diagrams |  |
| Class | 10/03/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  **X**  ✔ | ER Diagrams  Database Creation |  |
| Discord(Virtual) | 10/6/2023  Start: 8:00PM  Finish: 10:30PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database Creation | Start creation of data. |
| Class | 10/10/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database requirements for midterm | Start on additional usecases and tables for database. |
| Discord(Virtual) | 10/11/2023  Start: 7:30PM  Finish: 10:30PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Midterm |  |
| Class | 10/12/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database requirements for midterm | Start on additional usecases and tables for database. |
| Discord(Virtual) | 10/16/2023  Start: 7:30PM  Finish: 10:30PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Midterm | Finish required website pages for midterm |
| Class | 10/17/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database | Start on department, classes, faculty, etc. |
| Class | 10/19/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database |  |
| Class | 10/24/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database | Continue implementing department, classes, etc. |
| Discord(Virtual) | 10/25/2023  Start: 7:00PM  Finish: 9:30PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, ERD | Start on ERD Revamp, EERD. Divide up Database population between members |
| Class | 10/26/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, ERD | . |
| Discord(Virtual) | 10/27/2023  Start: 7:30PM  Finish: 9:00PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, ERD | Continue populating Database |
| Discord(Virtual) | 11/2/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, ERD |  |
| Discord(Virtual) | 11/6/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, ERD |  |
| Class | 11/7/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Coding | . |
| Discord(Virtual) | 11/11/2023  Start: 7:30PM  Finish: 9:30PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Coding | Fix foreign keys, import rest of data, finish student and admin side. |
| Discord(Virtual) | 11/12/2023  Start: 7:15PM  Finish: 9:40PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Coding | Goal: attempt to finish all code by end of month |
| Class | 11/14/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Coding |  |
| Class | 11/16/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Coding |  |
| Discord(Virtual) | 11/19/2023  Start: 7:00PM  Finish: 9:50PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Coding |  |
| Discord(Virtual) | 11/21/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Coding |  |
| Discord(Virtual) | 11/25/2023  Start: 3:00PM  Finish: 5:00PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Coding |  |
| Discord(Virtual) | 11/27/2023  Start: 3:00PM  Finish: 5:00PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Coding |  |
| Class | 11/28/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Coding |  |
| Class | 11/30/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Coding | Goal not met, continue working. |
| Class | 12/5/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Database, Coding |  |
| Class | 12/7/2023  Start: 3:50PM  Finish: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Project Overview and Presentation |  |
| Discord(Virtual) | 12/10/2023  Start: 7:00PM  Finish: 10:00PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador | ✔  ✔  ✔ | Finishing the project, Jeriel’s health issues | Complete code by Wed night/Thu morning if possible. User manual has priority completion over Systems Manual. |
| Discord(Virtual) | 12/13/2023  Start:  Finish: | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador |  | Project Presentation Preparations | Continue working on code, as much as possible. Finish Systems Manual/user Manual by Tomorrow afternoon to print |
| Class | 12/14/2023  Start: 3:50PM  End: 5:20PM | Hunter Oriol  Ganesh Ramcharan  Jeriel Salvador |  | Project Overview and Presentation |  |

**2. Problem Statement**

This project entails the careful integration of user privileges, ensuring that the administrators, faculty, students and statistics office can interact with the website in accordance with their roles and responsibilities. Our team has been tasked with the design and implementation of a dynamic web-based registration system for U.A. University, and we have been presented with an extensive set of system required specifications to guide us through the development process. These instructions serve as a comprehensive roadmap, delineating the specific actions and functionalities that each user type would be able to access and utilize on the website. However, within this framework, there are instances where certain actions and functionalities have been designated as ‘Required’ or 'Forbidden' for the designated user roles. To successfully navigate this project, we must interpret and implement these instructions accurately, taking into account the nuanced restrictions and permissions that apply in different scenarios.

**3. System Requirements Specification**

**3.1 Introduction**

**3.1.1 Purpose**

The purpose of the SRS is to outline the functional and nonfunctional requirements for the development of the dynamic web-based registration system.

**3.1.2 Scope**

This SRS document focuses on the System requirements for four user types: Students, Admins, Faculty members and Stats Office. It defines the necessary features and constraints for each user type.

**3.2 Specific Requirements**

**3.2.1 Student User Requirements**

**3.2.1a Explicitly Required Specifications**

1. Graduate students (part-time or full-time) are supported.
2. Undergrad students (part-time or full-time) are supported.
3. Full-time students can register for up to 12 credits.
4. Part-time students can register for up to 6 credits.
5. Users are required to sign in using their email and password.
6. The system will lock out users after 3 incorrect password attempts.
7. Students can view the master schedule for Fall 2023
8. Students can view the master schedule for Spring 2024.
9. The system allows students to search the master schedule by:
   * 1. CRN
     2. Course ID
     3. Course Name
     4. Section Number
     5. Department
     6. Days
     7. Building
     8. Room
     9. Time Slot
     10. Professor Name
     11. Professor ID
     12. Semester
     13. Available Seats

10. Students can view their Fall 2023 Personal Schedule.

11. Students can view their Spring 2024 Personal Schedule.

12. Students can view their holds (academic, disciplinary, finance, health)

13. Students can view their semester Grades.

14. Students can view their Unofficial transcript.

15. Students can view their Degree audit.

16. Students can view their Advisors.

17. Students can view their Personal information.

18. Students can view a list of courses in a department.

19. Students can view a list of Majors in a department.

20. Students can view a list of Minors in a department

21. Students can view Course prerequisites

22. Students can view assigned advisors.

23. Students can change Address and password.

24. Students can declare up to 2 majors

25. Students can declare up to 1 minor

26. Change a declared major

27. Change a declared minor

28. Add a course section within the time window.

29. Drop a course section within the time window.

**3.2.1b Forbidden Systems Requirements Specification**

1. Students cannot register for course sections if they have a hold.
2. Course section registration is prohibited if course prerequisites are not met.
3. Full-time students are not allowed to register for more than 12 credits.
4. Part-time students are not permitted to register for more than 6 credits.
5. Course section registration outside the assigned time window is not allowed.
6. Students cannot register for a course section with no available seating.
7. Students registering for two-course sections with the same time window is not allowed.
8. Students cannot register for a course section if they have already received a passing grade.
9. Students cannot drop a course section after a given time window.
10. Students cannot change User ID and email.
11. Students cannot change grades.
12. Students cannot change attendance records.
13. Undergraduate students cannot take graduate courses.
14. Graduate students cannot take undergraduate courses.
15. Students cannot have more than 2 majors or 1 major and 1 minor.

**3.2.2 Admin User Requirements**

**3.2.2a Explicitly Required Specifications**

1. Admins are full-time employees only.
2. Admins can log in to the system.
3. Admins are assigned a priority level, priority level zero or priority level one.
4. Admins can create a user including:
   * 1. User ID
     2. Name
     3. Email
     4. Password
     5. Address
     6. User type (Student, Faculty, Stats Office, Admin)
     7. Student information (graduate or undergrad)
     8. Faculty information (departments, full-time, part-time)
     9. Stats Office Information (Client Name)
     10. Admin information (Priority Level)
5. Admins can update all user info except for User ID.
6. Admins can create the Master Schedule for Fall 2023.
7. Admins can update the Master Schedule for Fall 2023.
8. Admins can create the Master Schedule for Spring 2024.
9. Admins can update the Master Schedule for Spring 2024.
10. Search the Master Schedule by:
    * 1. CRN
      2. Course ID
      3. Course Name
      4. Section Number
      5. Department
      6. Days
      7. Building
      8. Room
      9. Time Slot
      10. Professor Name
      11. Professor ID
      12. Semester
      13. Available Seats

11. Admins can create courses.

12. Admins can update courses

13. Admins can create course sections.

14. Admins can update course sections.

15. Admins can create departments.

16. Admins can update departments.

17. Admins can create majors.

18. Admins can create minors.

19. Admins can update majors.

20. Admins can update minors.

21. Admins can create course prerequisites.

22. Admins can create major requirements.

23. Admins can create minor requirements.

24. Admins can assign faculty to departments.

25. Admins can assign faculty to course sections.

26. Admins can assign time-slots to course sections.

27. Admins can assign rooms to course sections.

28. Admins can assign the number of available seats to course sections.

29. Admins can delete course sections.

30. Admins can update course section by faculty.

31. Admins can update course section by room.

32. Admins can update course section by available seats.

33. Admins can update course section by time slot.

34. Admins can add students to course sections.

35. Admins can remove students from course sections.

36. Admins can update students' grades in course sections.

37. Admins can assign a major to a student.

38. Admins can assign a minor to a student.

39. Admins can drop a major for a student.

40. Admins can drop a minor for a student.

41. Admins can view Student degree audits.

38. Admins can view Student Unofficial transcripts.

39. Admins can view Student Contact information.

40. Admins can view Faculty department

41. Admins can view Faculty schedules.

42. Admins can view Class rosters for course sections.

43. Admins can add holds for a student.

44. Admins can remove hold from a student.

**3.2.2b Forbidden Systems Requirements Specification**

1. For assigning faculty, Full-time faculty members cannot be assigned to more than 2 courses.
2. For assigning faculty, Part-time faculty members can only be assigned to 1 course.
3. For assigning faculty, a faculty member cannot be assigned to 2 courses in the same time-slot.
4. For assigning course sections, more than one course section cannot be assigned to a room in the same time-slot.
5. For assigning faculty, faculty members cannot be assigned to a course section outside their department.

**3.2.3 Faculty User Requirements**

**3.2.3a Explicitly Required Specifications**

1. Full-time faculty can teach 2 courses
2. Part-time faculty can teach 1 course.
3. Faculty can belong up to 2 departments.
4. Faculty members can view the master schedule for Fall 2023.
5. Faculty members can view the master schedule for Spring 2024.
6. Faculty can search the master schedule by:
   * 1. CRN
     2. Course ID
     3. Course Name
     4. Section Number
     5. Department
     6. Days
     7. Building
     8. Room
     9. Time Slot
     10. Professor Name
     11. Professor ID
     12. Semester
     13. Available Seats
7. Faculty can view their personal schedule for Fall 2023
8. Faculty can view their personal schedule for Spring 2024
9. Faculty can, in their personal schedules, view the class roster
10. Faculty can, In their class rosters, view the IDs of students assigned to the class
11. Faculty can, In their class rosters, view the emails of students assigned to the class
12. Faculty can, In their class rosters, view the transcripts of students assigned to the class
13. Faculty can, In their class rosters, view the advisors of students assigned to the class
14. Faculty can, In their class rosters, view the degree audits of students assigned to the class
15. Faculty can, In their class rosters, view the personal information of students assigned to the class
16. Faculty can record attendance only on the day of class.
17. Faculty can assign semester grades within a specified time window.
18. Faculty can view their advised students.
19. Faculty can view the Unofficial transcripts of their advised students
20. Faculty can view the Degree audits of their advised students.
21. Faculty can view the personal information of their advised students.
22. Faculty can view their Personal information.
23. Faculty can view a list of courses in a department.
24. Faculty can view a list of Majors in a department.
25. Faculty can view a list of Minors in a department.
26. Faculty can view courses in a department.
27. Faculty can view course prerequisites.
28. The system will lock out users after 3 incorrect password attempts.

**3.2.3b Forbidden Systems Requirements Specification**

1. Faculty members cannot teach more than **2 classes** if they are **full-time** or **1 class** if they are **part-time**.
2. Faculty members cannot be in more than 2 departments.
3. Faculty members teaching outside their department is not allowed.
4. Faculty members cannot view class attendance outside of their assigned course sections.
5. Faculty members cannot assign grades outside the given time window.
6. Faculty members can only advise up to 10 students.
7. Changing email addresses and user IDs is not permitted.

**3.2.4 Stats Office Requirements**

**3.2.4a Explicitly Required Specifications**

1. Stats office can view overall school statistics including:
   * 1. Percentage of Graduate Students.
     2. Percentage of Full-Time Students.
     3. Percentage of Part-Time Students.
2. Stats office can view departmental statistics including:
   * 1. Number of Masters (M.S.) Students per Discipline (Department).
     2. Number of Ph.D. Students per Discipline (Department).
3. Stats office can view the current graduation rate of the school.
4. Stats office can view major and minor statistics including:
   * 1. Number of Majors in different academic majors.
     2. Number of Minors in different academic minors.
5. Stats Office can view the master schedule for Fall 2023.
6. Stats Office can view the master schedule for Spring 2024.
7. Stats office can search the master schedule by:
   * 1. CRN
     2. Course ID
     3. Course Name
     4. Section Number
     5. Department
     6. Days
     7. Building
     8. Room
     9. Time Slot
     10. Professor Name
     11. Professor ID
     12. Semester
8. Stats office can view their Personal information.
9. Stats office can view a list of courses in a department.
10. Stats office can view a list of Majors in a department.
11. Stats office can view a list of Minors in a department.
12. Stats office can view courses in a department.
13. Stats office can view course prerequisites.
14. The system will lock out users after 3 incorrect password attempts.

**4. Use Case Tables**

**4.1 UNIVERSAL**

| **USE CASE** | **Successful Login** |
| --- | --- |
| List of actors | Student, Faculty, Admin or Statistics Office |
| Initial Conditions | User attempts to log into their account using the Sign-In page. |
| Flow of events | 1. User clicks on “Login” button on the website.  2. User types in username (email) and password into their respective input boxes.  3. User clicks the “Login Now” button.  4. User’s Email and Password are valid. |
| Exit | User is granted access to their account and is signed in. |

| **USE CASE** | **Unsuccessful Login** |
| --- | --- |
| List of actors | Student, Faculty, Admin or Statistics Office |
| Initial Conditions | User attempts to log into their account using the Sign-In page but has failed three(3) times. |
| Flow of events | 1. User attempts to login with Invalid credentials.  2. Login count exceeds 3 allowed attempts.  3. User account is locked out  4. Admin receives an email notifying them that a user account has been locked out and given a temporary password.  5. Admin contacts user and gives them the temporary password.  6. User receives temporary password and uses it to log in to their account. |
| Exit | User can now attempt to login to their account using given temporary password. |

| **USE CASE** | **View List of Departments** |
| --- | --- |
| List of actors | Student, Faculty, Admin or Statistics Office |
| Initial Conditions | User logs in and attempts to view the list of departments on the University website. |
| Flow of events | 1. User is on the front page of the University website.  2. User clicks on the “Departments” button  3. User is given a list of all departments in the university. |
| Exit | User can now choose which department to view from the list. |

| **USE CASE** | **View Majors in a Department** |
| --- | --- |
| List of actors | Student, Faculty, Admin or Statistics Office |
| Initial Conditions | User logs in and attempts to view the majors in a department |
| Flow of events | 1. User is on the front page of the University website.  2. User clicks on the “Departments” button  3. User is given a list of all departments in the university.  4. User chooses a specific department from the list  5. User scrolls down to the “Major Offerings” section.  6. User is given a list of all available majors in the department. |
| Exit | User can now view the available majors in the department. |

| **USE CASE** | **View Minors in a Department** |
| --- | --- |
| List of actors | Student, Faculty, Admin or Statistics Office |
| Initial Conditions | User logs in and attempts to view the minors in a department |
| Flow of events | 1. User is on the front page of the University website.  2. User clicks on the “Departments” button  3. User is given a list of all departments in the university.  4. User chooses a specific department from the list  5. User scrolls down to the “Minor Offerings” section.  6. User is given a list of all available minors in the department. |
| Exit | User can now view the available minors in the department. |

| **USE CASE** | **View Courses in a Department** |
| --- | --- |
| List of actors | Student, Faculty, Admin or Statistics Office |
| Initial Conditions | User logs in and attempts to view the minors in a department. |
| Flow of events | 1. User is on the front page of the University website.  2. User clicks on the “Departments” button.  3. User is given a list of all departments in the university.  4. User chooses a specific department from the list.  5. User scrolls down to the “Course Offerings” section.  6. User is given a list of all available courses in the department. |
| Exit | User can now view the available courses in the department. |

| **USE CASE** | **View Course Prerequisites** |
| --- | --- |
| List of actors | Student, Faculty, Admin or Statistics Office |
| Initial Conditions | User logs in and attempts to view the course prerequisites of a specific course. |
| Flow of events | 1. User is on the front page of the University website.  2. User clicks on the “Departments” button.  3. User is given a list of all departments in the university.  4. User chooses a specific department from the list.  5. User scrolls down to the “Course” section.  6. User chooses a specific course to view.  7. User is given a list of the course prerequisites for the chosen course. |
| Exit | User can now view the available courses in the department. |

**4.2 STUDENT**

| **USE CASE** | **View Personal Information** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to view their personal information. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button |
| Exit | Student can view their personal information. |

| **USE CASE** | **Update Address** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempts to change their address. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button.  4. Student scrolls down to “User Options” section.  5. Student is given list of what is valid to be changed.  6. Student inputs new address into their respective fields.  7. Student clicks the “Update User” button. |
| Exit | User has now changed the Address listed on their account. |

| **USE CASE** | **Update Password** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempts to change their password. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button.  4. Student scrolls down to “User Options” section.  5. Student is given list of what is valid to be changed.  6. Student inputs new Password into the respective field.  7. Student clicks the “Update User” button. |
| Exit | Student has now changed the password on their account. |

| **USE CASE** | **View Master Schedule** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student logs in and attempts to view the Master Schedule on the University website. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Master Schedule” button. |
| Exit | Student can now view the Master Schedule. |

| **USE CASE** | **Search Master Schedule** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student logs in and attempts to search the Master schedule using a filter  (Department name, Course Name, Professor Name, Time Slot, Room/Building or Available Seats) |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Master Schedule” button.  3. Student selects which filter is to be used.  4. If applicable, Student inputs a specific value for the filter for a more specific search. |
| Exit | Student can search the Master Schedule using the filters they have chosen. |

| **USE CASE** | **View Hold** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to view a hold placed on their account. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button.  3. Student views the “hold” section on the “Academic Profile” page. |
| Exit | Student can view their academic hold on their account. |

| **USE CASE** | **View Personal Schedule - Fall 2023** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student logs in and attempts to view their personal schedule. |
| Flow of events | 1. Student logs in to their account.  2. Student Personal Schedule is located on the Student Portal home page.  4. User chooses “Fall 2023” schedule from the filter. |
| Exit | Student can view their Personal Schedule for the Fall 2023 semester. |

| **USE CASE** | **View Personal Schedule - Spring 2024** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student logs in and attempts to view their personal schedule. |
| Flow of events | 1. Student logs in to their account.  2. Student Personal Schedule is located on the Student Portal home page.  4. User chooses “Spring 2024” schedule from the filter. |
| Exit | Student can view their Personal Schedule for the Spring 2024 semester. |

| **USE CASE** | **View Grades** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to view their grades. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button.  3. Student views the “Current Grades” section on the “Academic Profile” page. |
| Exit | The Students can view their grades for the all semesters. |

| **USE CASE** | **View Unofficial Transcript** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to view their unofficial transcript. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button.  3. Student clicks on the “Unofficial Transcript” Button. |
| Exit | Student can view their Unofficial Transcript. |

| **USE CASE** | **View Degree Audit** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to view their degree audit. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button  3. Student clicks on the “Degree Audit” Button. |
| Exit | Student can view their Degree Audit. |

| **USE CASE** | **View Assigned Advisors** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to view their assigned advisors. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button  3. Student views the “Student Advisors” section on the “Academic Profile” page. |
| Exit | Student can view their assigned advisors. |

| **USE CASE** | **View Current Majors** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to view their majors. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button.  3. Student scrolls down to the “Current Majors” Section.  4. Student is given list of declared majors on their account, if applicable. |
| Exit | Student can view their current majors. |

| **USE CASE** | **View Current Minors** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to view their minors. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button.  3. Student scrolls down to the “Current Minors” Section.  4. Student is given list of declared minors on their account, if applicable. |
| Exit | Student can view their current minors. |

| **USE CASE** | **Declare Major - Success** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to declare a major. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “View Academic Profile” Button.  3. Student scrolls down to the “Assign Major” Section.  4. Student picks a major from the “Select a major” drop-down box.  5. Student is given a list of all majors available to them  6. Student chooses a major from the list and clicks the “Assign Major” button. |
| Exit | Student has added a major to their account. |

| **USE CASE** | **Declare Major - Failure** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has attempted to declare a major but has 2 majors or 1 major 1 minor. |
| Flow of events | 1. Student attempts to add a major.  2. Student has 2 majors or 1 major and 1 minor.  3. System rejects the transaction and returns an error. |
| Exit | Student has been prevented from adding a major to their account. |

| **USE CASE** | **Declare Minor - Success** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to declare a major. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “View Academic Profile” Button.  3. Student scrolls down to the “Assign Minor” Section.  4. Student picks a major from the “Select a minor” drop-down box.  5. Student is given a list of all minors available to them  6. Student chooses a minor from the list and clicks the “Assign Minor” button. |
| Exit | Student has added a minor to their account. |

| **USE CASE** | **Declare Minor - Failure** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has attempted to declare a minor but has 2 majors or 1 major 1 minor. |
| Flow of events | 1. Student attempts to add a minor.  2. Student has 2 majors or 1 major and 1 minor.  3. System rejects the transaction and returns an error. |
| Exit | Student has been prevented from adding a minor to their account. |

| **USE CASE** | **Drop Major** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to drop one of their declared majors. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button.  3. Student scrolls down to the “Drop Major” section  4. Student chooses a major from the “Select a major to drop” drop down box.  5. Student clicks on the “Drop Major” button |
| Exit | Student has dropped a major on their account. |

| **USE CASE** | **Drop Minor** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to drop one of their declared minors. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button.  3. Student scrolls down to the “Drop Minor” section  4. Student chooses a minor from the “Select a major to drop” drop down box.  5. Student clicks on the “Drop Minor” button |
| Exit | Student has dropped a minor on their account. |

| **USE CASE** | **Register Course - Successful** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to Register for a course. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button.  3. Student clicks on the “Create Schedule” Button.  4. Student is given list of current course sections on their account  5. Student chooses a course section from the master schedule given to them.  6. Student clicks on the “Assign Course” button.  6. Holds, Available Seats, Prerequisites, Total Credits,Schedule Conflict and Time Window requirements are met. |
| Exit | Student has added a course to their personal schedule. |

| **USE CASE** | **Register Course - Unsuccessful** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to Register for a course. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button.  3. Student clicks on the “Create Schedule” Button.  4. Student is given list of current course sections on their account  5. Student chooses a course section from the master schedule given to them.  6. Student clicks on the “Assign Course” button.  7. One of the following requirements is not met: Holds, Available Seats, Prerequisites, Total Credits, Schedule Conflict or Time Window.  8. System rejects the transaction and returns an error. |
| Exit | Student is forbidden from adding the course section to their schedule. |

| **USE CASE** | **Drop Course - Successful** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to drop a course from their schedule. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button.  3. Student clicks on the “Create Schedule” Button.  4. Student is given list of current course sections on their account  5. Student chooses a current course section on their schedule and clicks on the “Drop Course” button.  6. Time Window requirements are met and course is allowed to be dropped. |
| Exit | Student has dropped a course to their personal schedule. |

| **USE CASE** | **Drop Course - Unuccessful** |
| --- | --- |
| List of actors | Student |
| Initial Conditions | Student has logged in and attempted to drop a course from their schedule. |
| Flow of events | 1. Student logs in to their account.  2. Student clicks on the “Academic Profile” Button.  3. Student clicks on the “Create Schedule” Button.  4. Student is given list of current course sections on their account  5. Student chooses a current course section on their schedule and clicks on the “Drop Course” button.  6. Outside of “Drop Course” Time Window, action not allowed. |
| Exit | Student has been prevented from dropping the course from their schedule. |

**4.3 ADMIN**

| **USE CASE** | **Admin Log-In** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin Logged in |
| Flow of events | 1. Admin logs into their account  2. The system grants access to the system based on the Admin's priority level.  3. Admin uses the system according to their priority level |
| Exit | Admin can log into the system at their priority level. |

| **USE CASE** | **View Personal Information** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempted to view their personal information. |
| Flow of events | 1. Admin logs in to their account.  2. Admin clicks on the “Personal Information” Button |
| Exit | Admin can view their personal information. |

| **USE CASE** | **Update Address** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to change their address. |
| Flow of events | 1. Admin logs in to their account.  2. Admin clicks on the “Personal Information” Button.  4. Admin scrolls down to “User Options” section.  5. Admin is given list of what is valid to be changed.  6. Admin inputs new address into their respective fields.  7. Admin clicks the “Update User” button. |
| Exit | Admin has now changed the Address listed on their account. |

| **USE CASE** | **Update Password** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to change their password. |
| Flow of events | 1. Admin logs in to their account.  2. Admin clicks on the “Personal Information” Button.  4. Admin scrolls down to “User Options” section.  5. Admin is given list of what is valid to be changed.  6. Admin inputs new Password into the respective field.  7. Admin clicks the “Update User” button. |
| Exit | Admin has now changed the password on their account. |

| **USE CASE** | **View Master Schedule** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin logs in and attempts to view the Master Schedule on the University website. |
| Flow of events | 1. Admin logs in to their account.  2. Admin clicks on the “Master Schedule” button. |
| Exit | Admin can now view the Master Schedule. |

| **USE CASE** | **Search Master Schedule** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin logs in and attempts to search the Master schedule using a filter  (Department name, Course Name, Professor Name, Time Slot, Room/Building or Available Seats) |
| Flow of events | 1. Admin logs in to their account.  2. Admin clicks on the “Master Schedule” button.  3. Admin selects which filter is to be used.  4. If applicable, Admin inputs a specific value for the filter for a more specific search. |
| Exit | Admin can search the Master Schedule using the filters they have chosen. |

| **USE CASE** | **Create a user account** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to create a user account |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Update a user” button.  3. Admin clicks on the “create a user” button.  4. Admin inputs the user's name.  5. Admin inputs the user’s email.  6. Admin inputs the user’s password.  7. Admin inputs the users’ address.  8. The admin reviews the information and clicks the “Confirm” button. |
| Exit | The user is created. |

| **USE CASE** | **Update Student Information** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update student information |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Update a user” button.  3. Admin searches for student by Student ID.  4. Admin clicks on the “Update Student Information” button.  5. Admin changes the student’s information and clicks the “Confirm” button. |
| Exit | The admin has updated the student information. |

| **USE CASE** | **Add Faculty to Department - Successful** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update faculty department information. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Departments” button.  3. Admin clicks on the “Edit Faculty” button.  3. Admin searches for the faculty by Department and is given a list of all faculty members in their department.  4. Alternatively, Admin scrolls down to “Faculty Members in Two Departments” section.  5. Admin inputs Faculty ID of the faculty that they want to add in the “Add Department to Faculty” section.  6. Faculty ID chosen is currently not in two departments, requirements are met. |
| Exit | Admin has added a faculty member to a department. |

| **USE CASE** | **Add Faculty to Department - Failure** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update faculty department information. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Departments” button.  3. Admin clicks on the “Edit Faculty” button.  3. Admin searches for the faculty by Department and is given a list of all faculty members in their department.  4. Alternatively, Admin scrolls down to “Faculty Members Not Currently Assigned to a Department” section.  5. Admin inputs Faculty ID of the faculty that they want to add in the “Add Department to Faculty” section.  6. Faculty ID chosen is currently in two departments, requirements are not met. |
| Exit | Admin has been prevented from adding a faculty member to a department |

| **USE CASE** | **Drop Faculty from Department - Successful** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update faculty department information. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Departments” button.  3. Admin clicks on the “Edit Faculty” button.  3. Admin searches for the faculty by Department and is given a list of all faculty members in their department.  4. Alternatively, Admin scrolls down to “Faculty Members Not Currently Assigned to a Department” section.  5. Admin inputs Faculty ID of the faculty that they want to add in the “Drop Department from Faculty” section.  6. Faculty ID chosen is currently not teaching a class in that department, requirements are met. |
| Exit | Admin has removed a faculty member from a department. |

| **USE CASE** | **Drop Faculty from Department - Failure** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update faculty department information. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Departments” button.  3. Admin clicks on the “Edit Faculty” button.  3. Admin searches for the faculty by Department and is given a list of all faculty members in their department.  4. Alternatively, Admin scrolls down to “Faculty Members Not Currently Assigned to a Department” section.  5. Admin inputs Faculty ID of the faculty that they want to add in the “Drop Department from Faculty” section.  6. Faculty ID chosen is currently teaching a class in that department, requirements are not met. |
| Exit | Admin has been prevented from removing a faculty member from a department |

| **USE CASE** | **Update Faculty Employment Status (Full Time)** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update faculty to full-time. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Users” button.  3. Admin searches for the faculty by faculty ID.  4. Admin changes the faculty employment status to full-time and clicks the “Update” button. |
| Exit | Admin has updated the faculty members' employment status to full-time |

| **USE CASE** | **Update Faculty Employment Status (Part Time)** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update faculty to part-time. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Users” button.  3. Admin searches for the faculty by facutly ID.  4. Admin changes the faculty employment status to part-time and clicks the “Update” buttom. |
| Exit | Admin has updated the faculty members' employment status to part-time |

| **USE CASE** | **Create Sections for Fall 2023 - Failure** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and has to create a master schedule course section for Fall 2023. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Master Schedule” button.  3. Admin clicks on the “Create Course Section” button  4. Admin cannot input Fall 2023 as the course section’s SemesterID. |
| Exit | Admin has been prevented from creating a course section for Fall 2023, Semester in progress. |

| **USE CASE** | **Update Course Section for Spring 2024** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update a course section for Spring 2024. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Master Schedule” button.  3. Admin clicks on the “Edit” button next to a Fall 2023 course section.  4. Admin reviews the course section and changes any necessary information to the schedule.  5. Admin clicks on the “Confirm” button. |
| Exit | The course section for Fall 2023 is updated. |

| **USE CASE** | **Create Course Sections for Spring 2024 - Success** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and has to create a master schedule course section for Spring 2024. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Master Schedule” button.  3. Admin clicks on the “Create Course Section” button  4. Admin inputs the information of the course section.  5. Admin reviews the information for the course section.  6. Admin clicks on the “Create Course” button. |
| Exit | The course section for Spring 2024 is created. |

| **USE CASE** | **Create Course Sections for Spring 2024 - Failure** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and has to create a master schedule course section for Spring 2024. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Master Schedule” button.  3. Admin clicks on the “Create Course Section” button  4. Admin inputs the information of the course section.  5. Admin reviews the information for the course section.  6. Admin clicks on the “Create Course” button.  7. The CRN currently exists, transaction forbidden. |
| Exit | The course section for Spring 2024 is prevented from being created. |

| **USE CASE** | **Update Course Section for Spring 2024** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update a course section for Spring 2024. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Master Schedule” button.  3. Admin clicks on the “Edit” button next to a Spring 2024 course section.  4. Admin reviews the course section and changes any necessary information to the schedule.  5. Admin clicks on the “Confirm” button. |
| Exit | The course section for Spring 2024 is updated. |

| **USE CASE** | **Admin Creates Department** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to create a Department |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Departments” button.  3. Admin clicks on the “Create” button.  4. Admin names the department.  5. Admin gives the department a description.  6. Admin reviews the information and clicks on the “Confirm” button. |
| Exit | The department is created. |

| **USE CASE** | **Create a Course - Success** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to create a course. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Course Catalog” button.  3. Admin clicks on the “Create Course” button.  4. Admin names the course.  5. Admin gives the course a description.  6. Admin assigns the department of the Course.  7. Admin assigns the credits of the course.  8. Admin assigns the Description of the course.  9. Admin assigns the Course Type of the course.  10. Admin assigns the initial Prerequisite Course ID of the course (if applicable).  11. Admin assigns the minimum grade of the initial Prerequisite Course (if applicable).  12. Admin assigns the DOLU of the Prerequisite Course (if applicable).  13. Admin clicks “Create Course” |
| Exit | The new course is created and added to the system's course catalog. |

| **USE CASE** | **Create a Course - Failure** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to create a course. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Course Catalog” button.  3. Admin clicks on the “Create Course” button.  4. Admin names the course.  5. Admin gives the course a description.  6. Admin assigns the department of the Course.  7. Admin assigns the credits of the course.  8. Admin assigns the Description of the course.  9. Admin assigns the Course Type of the course.  10. Admin assigns the initial Prerequisite Course ID of the course (if applicable).  11. Admin assigns the minimum grade of the initial Prerequisite Course (if applicable).  12. Admin assigns the DOLU of the Prerequisite Course (if applicable).  13. Admin clicks “Create Course”  14. The CourseID currently exists, transaction forbidden. |
| Exit | The new course is prevented from being created and added to the system's course catalog. |

| **USE CASE** | **Update Course Section Number - Success** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update the courses’ section number. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Master Schedule” button.  3. Admin is presented a list of available course sections.  4. Admin chooses the target course section.  5. Admin clicks on the “Update” button.  6. Admin changes the Section Number to the target number.  7. Admin reviews the information and clicks on the “Confirm” button.  8. There currently is no duplicate course section number for the CourseID. |
| Exit | The course section’s number has been updated |

| **USE CASE** | **Update Course Section Number - Failure** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update the courses’ section number. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Master Schedule” button.  3. Admin is presented a list of available course sections.  4. Admin chooses the target course section.  5. Admin clicks on the “Update” button.  6. Admin changes the Section Number to the target number.  7. Admin reviews the information and clicks on the “Confirm” button.  8. There currently is a duplicate course section number for the CourseID. |
| Exit | Admin has been prevented from updating Course Section Number. |

| **USE CASE** | **Update Course Name** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update the course name. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Course Catalog” button.  3. Admin is presented a list of available courses.  4. Admin clicks on the target course.  5. Admin clicks on the “Update” button.  6. Admin changes the course name to the target name.  7. Admin reviews the information and clicks on the “Confirm” button. |
| Exit | The name for the selected course is now updated. |

| **USE CASE** | **Update Course Time Slot** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update the course time slot. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Master Schedule” button.  3. Admin is presented a list of available course sections.  4. Admin clicks on the target course sections.  5. Admin clicks on the “Edit” button.  6. Admin changes the course section’s time slot to the target time slot.  7. Admin reviews the information and clicks on the “Confirm” button. |
| Exit | The time slot for the selected course section is updated in the system. |

| **USE CASE** | **Update Course Building** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update the Course building. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Master Schedule” button.  3. Admin is presented a list of available course sections.  4. Admin clicks on the target course sections.  5. Admin clicks on the “Edit” button.  6. Admin changes the course section’s time slot to the target time slot with the building desired.  7. Admin reviews the information and clicks on the “Confirm” button. |
| Exit | The building location for the selected course is updated in the system. |

| **USE CASE** | **Update Course Room Number** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update the Course Room number. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Master Schedule” button.  3. Admin is presented a list of available course sections.  4. Admin clicks on the target course sections.  5. Admin clicks on the “Edit” button.  6. Admin changes the course section’s time slot to the target time slot with the room desired.  7. Admin reviews the information and clicks on the “Confirm” button. |
| Exit | The room location for the course is updated in the system. |

| **USE CASE** | **Update Course Available Seats** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to update the Courses’ available seats. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Master Schedule” button.  3. Admin is presented a list of available course sections.  4. Admin clicks on the target course sections.  5. Admin clicks on the “Edit” button.  6. Admin changes the courses available seats to the target amount of available seats.  7. Admin reviews the information and clicks on the “Confirm” button. |
| Exit | The available seats for the selected course are updated in the system. |

| **USE CASE** | **Admin Creates a Major** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to create a major. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Majors” button.  3. Admin clicks on the “Create” button.  4. Admin adds the major name, major description, department.  5. Admin reviews the information and clicks on the “Confirm” button. |
| Exit | The major is created and added to the system's list of available majors. |

| **USE CASE** | **Admin Creates a Minor** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to create a minor. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Minors” button.  3. Admin clicks on the “Create” button.  4. Admin adds the minor name, minor description, department.  5. Admin reviews the information and clicks on the “Confirm” button. |
| Exit | The minor is created and added to the system's list of available minors. |

| **USE CASE** | **Admin Deletes a Major - Success** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to delete a major. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Majors” button.  3. Admin selects a major to delete.  4. Admin clicks the “delete” button next to the major  5. Major is not assigned to students, can be dropped. |
| Exit | The major is deleted and removed from the system's list of available majors. |

| **USE CASE** | **Admin Deletes a Minor - Success** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to delete a minor. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Minors” button.  3. Admin selects a minor to delete.  4. Admin clicks the “delete” button next to the minor  5. Minor is not assigned to students, can be dropped. |
| Exit | The minor is deleted and removed from the system's list of available minors. |

| **USE CASE** | **Admin Deletes a Major - Failure** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to delete a major. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Majors” button.  3. Admin selects a major to delete.  4. Admin clicks the “delete” button next to the major  5. Major is currently assigned to students, can not be dropped. |
| Exit | The major is prevented from being deleted by the admin. |

| **USE CASE** | **Admin Deletes a Minor - Failure** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to delete a minor. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Minors” button.  3. Admin selects a minor to delete.  4. Admin clicks the “delete” button next to the minor  5. Minor is currently assigned to students, can not be dropped. |
| Exit | The minor is prevented from being deleted by the admin. |

| **USE CASE** | **Admin Sets Major Prerequisites** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to set major prerequisites |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Majors” button.  3. Admin selects target major to add prerequisite to.  4. Admin clicks on “Edit Prerequisites” button for the target major  5. Admin selects which course to add to the major.  6. Admin inputs the Minimum Grade and DOLU for the Major prerequisite  6. Admin reviews the information and clicks on the “Add Prerequisite” button. |
| Exit | The prerequisites for the target major are in the system. |

| **USE CASE** | **Admin Sets Minor Prerequisites** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to set minor prerequisites |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Minors” button.  3. Admin selects target minor to add prerequisite to.  4. Admin clicks on “Edit Prerequisites” button for the target minor  5. Admin selects which course to add to the minor .  6. Admin inputs the Minimum Grade and DOLU for the minor prerequisite  6. Admin reviews the information and clicks on the “Add Prerequisite” button. |
| Exit | The prerequisites for the target minor are in the system. |

| **USE CASE** | **Admin Edits Major Prerequisites** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to edit major prerequisites |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Majors” button.  3. Admin selects target major to edit prerequisites.  4. Admin clicks on “Edit Prerequisites” button for the target major  5. Admin selects which prerequisite to edit.  6. Admin chooses “Update” as the action for the prerequisite  7. Admin changes desired information  8. Admin reviews the information and clicks on the “Save Changes” button. |
| Exit | The prerequisites for the target major have the new information. |

| **USE CASE** | **Admin Edits Minor Prerequisites** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to edit minor prerequisites |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Minors” button.  3. Admin selects target minor to edit prerequisites.  4. Admin clicks on “Edit Prerequisites” button for the target minor  5. Admin selects which prerequisite to edit.  6. Admin chooses “Update” as the action for the prerequisite  7. Admin changes desired information  8. Admin reviews the information and clicks on the “Save Changes” button. |
| Exit | The prerequisites for the target minor have the new information. |

| **USE CASE** | **Admin Drops Major Prerequisites** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to drop major prerequisites |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Majors” button.  3. Admin selects target major to edit prerequisites.  4. Admin clicks on “Edit Prerequisites” button for the target major  5. Admin selects which prerequisite to edit.  6. Admin chooses “Delete” as the action for the prerequisite  7. Admin changes desired information  8. Admin reviews the information and clicks on the “Save Changes” button. |
| Exit | The prerequisites for the target major have been dropped. |

| **USE CASE** | **Admin Drops Minor Prerequisites** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to drop minor prerequisites |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Minors” button.  3. Admin selects target minor to edit prerequisites.  4. Admin clicks on “Edit Prerequisites” button for the target minor  5. Admin selects which prerequisite to edit.  6. Admin chooses “Delete” as the action for the prerequisite  7. Admin changes desired information  8. Admin reviews the information and clicks on the “Save Changes” button. |
| Exit | The prerequisites for the target minor have been dropped. |

| **USE CASE** | **Admin Sets Course Prerequisites** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to set course prerequisites |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Course Catalog” button.  3. Admin clicks on the “Create Course Prerequisite” button  4. Admin selects the target Course to give a prerequisite.  5. Admin sets the course prerequisites for the target course.  6. Admin reviews the information and clicks on the “Confirm” button. |
| Exit | The prerequisites for the target course are in the system. |

| **USE CASE** | **Admin Drops Course Prerequisites** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to drop course prerequisites |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Course Catalog” button.  3. Admin selects the target Course to delete the prerequisite from.  4. Admin clicks on the “Update” button next to the selected course under the “Manage Course Prerequisites” column  5. Admin selects which Course Prerequisite to delete.  6. Admin reviews the information and clicks the “Delete” button for that course prerequisite. |
| Exit | The prerequisites for the target course have been dropped. |

| **USE CASE** | **Add Student to Course** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to add student to course |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Update a User” button.  3. Admin is presented a list of Users.  4. Admin clicks on the target’s “View Academic Profile” button.  5. Admin clicks on the “Create Schedule” button.  6. Admin is presented a list of available course sections  7. Admin selects the target course section to be added to the student.  8. Admin reviews the information and clicks on the “Assign Course” button. |
| Exit | The student is now added to the Course |

| **USE CASE** | **Drop student from Course Section** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to drop student from course |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Update a User” button.  3. Admin is presented a list of Users.  4. Admin clicks on the target’s “View Academic Profile” button.  5. Admin clicks on the “Create Schedule” button.  6. Admin is presented a list of the student’s current course sections.  7. Admin selects the target course section to be dropped from the student.  8. Admin reviews the information and clicks on the “Drop” button. |
| Exit | The student is now dropped from the course section. |

| **USE CASE** | **Add hold to student account** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to add hold to student account |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Update a user” button.  3. Admin clicks on the “Create hold” button.  4. Admin is presented a list of available students.  5. Admin chooses the StudentID of the target student.  6. Admin creates the hold and adds the hold to the student account.  7. Admin reviews the information and clicks on the “Confirm” button. |
| Exit | The student now has a hold on their account. |

| **USE CASE** | **Remove hold from student account** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to remove hold from student account |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Update a User” button.  3. Admin clicks on the “Create Hold” button.  4. Admin is presented a list of available students.  5. Admin finds the target student with a hold.  6. Admin removes the hold to the student account by clicking the “Remove” button next to the student’s entry on the list. |
| Exit | The student no longer has a hold on their account. |

| **USE CASE** | **Update Student Grade in Course** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts update a students grade |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Master Schedule” button.  3. Admin is presented a list of available course sections.  4. Admin clicks the “Class Roster” button on the target course sections.  5. Admin is presented a list of students taking the target course.  6. Admin chooses the target student.  7. Admin updates the target student’s grade.  8. Admin reviews the information and clicks on the “Assign Grade” button. |
| Exit | Admin has updated the students grade |

| **USE CASE** | **View Student information** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to view a student’s information |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Update a User” button.  3. Admin finds the target student in the list of Users.  4. Admin clicks on the target student’s “View Academic Profile” button  5. The student's information is now available to the admin. |
| Exit | The admin can view the students information. |

| **USE CASE** | **View Faculty Schedules** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to view Faculty schedules |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Update a User” button.  3. Admin finds the target faculty in the list of Users.  3. Admin clicks on the target faculty’s “View Academic Profile”.  4. The faculty information is now available to the admin. |
| Exit | The admin can view the faculty schedule. |

| **USE CASE** | **View class roster for course** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to view the class roster for a course section. |
| Flow of events | 1. Admin logs in.  2. Admin clicks on the “Master Schedule” button.  3. Admin is presented a list of available course sections.  4. Admin clicks the target course section’s “Class Roster” button.  5. The class roster is available to for the admin to see. |
| Exit | The admin can review the class roster in the target course. |

| **USE CASE** | **Add major to student** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to add a major to student’s academic profile. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Update a user” button.  3. Admin is provided a list of users.  4. Admin clicks on the target student’s “View Academic Profile” button.  5. Admin scrolls down to the “Assign Major” section  6. Admin chooses a major from the list to assign to the student  7. Admin reviews information and clicks on the “Assign Major” button. |
| Exit | Admin has added the major to the student |

| **USE CASE** | **Drop major from student** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to drop a major from student information. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Update a user” button.  3. Admin is provided a list of users.  4. Admin clicks on the target student’s “View Academic Profile” button.  5. Admin scrolls down to the “Drop Major” section  6. Admin chooses a major from the list to drop from the student  7. Admin reviews information and clicks on the “Drop Major” button. |
| Exit | Admin can drop the major from the student. |

| **USE CASE** | **Add minor to student** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to add a minor to student academic profile. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Update a user” button.  3. Admin is provided a list of users.  4. Admin clicks on the target student’s “View Academic Profile” button.  5. Admin scrolls down to the “Assign Minor” section  6. Admin chooses a minor from the list to assign to the student  7. Admin reviews information and clicks on the “Assign Minor” button. |
| Exit | Admin has added the minor to the students information |

| **USE CASE** | **Drop minor from student** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to drop a minor from student information. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Update a user” button.  3. Admin is provided a list of users.  4. Admin clicks on the target student’s “View Academic Profile” button.  5. Admin scrolls down to the “Drop Minor” section  6. Admin chooses a minor from the list to drop from the student  7. Admin reviews information and clicks on the “Drop Minor” button. |
| Exit | Admin can drop the minor from the students information. |

| **USE CASE** | **Assign Full Time faculty to course section - Failure** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin logged in and attempts to assign more than 2 course sections to full-time faculty member |
| Flow of events | 1. Admin logs in.  2. Admin clicks on “Master Schedule”.  3. Admin is presented a list of course sections.  4. Admin selects the target course.  5. Admin clicks on the “Edit” button.  6. Admin inputs the target faculty member.  7. Admin attempts to update the course section.  8. System identifies faculty member as full-time  9. System counts faculty course assignments to be more than 2.  10. System enforces the course assignment limit. |
| Exit | The admin failed to assign target faculty member to course because of the course assignment limit. |

| **USE CASE** | **Assign Part Time faculty to Course Section - Failure** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to assign more than 1 course section to part-time faculty member |
| Flow of events | 1. Admin logs in.  2. Admin clicks on “Master Schedule”.  3. Admin is presented a list of course sections.  4. Admin selects the target course.  5. Admin clicks on the “Edit” button.  6. Admin inputs the target faculty member.  7. Admin attempts to update the course section.  8. System identifies faculty member as part-time.  9. System counts faculty course assignments to be more than 1.  10. System enforces the course assignment limit. |
| Exit | The admin failed to assign target faculty member to course because of the course assignment limit. |

| **USE CASE** | **Assign Full Time faculty to course section - Success** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin logged in and attempts to assign a course section to a full-time faculty member |
| Flow of events | 1. Admin logs in.  2. Admin clicks on “Master Schedule”.  3. Admin is presented a list of course sections.  4. Admin selects the target course.  5. Admin clicks on the “Edit” button.  6. Admin inputs the target faculty member.  7. Admin attempts to update the course section.  8. System identifies faculty member as full-time  9. System counts faculty course assignments to be less than 2.  10. System allows the Faculty assignment |
| Exit | The admin has assigned a Full Time faculty member to a course section |

| **USE CASE** | **Assign Part Time faculty to Course Section - Success** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to assign a course section to a part-time faculty member |
| Flow of events | 1. Admin logs in.  2. Admin clicks on “Master Schedule”.  3. Admin is presented a list of course sections.  4. Admin selects the target course.  5. Admin clicks on the “Edit” button.  6. Admin inputs the target faculty member.  7. Admin attempts to update the course section.  8. System identifies faculty member as part-time.  9. System counts faculty course assignments to be less than 1.  10. System allows the Faculty assignment |
| Exit | The admin has assigned a Part Time faculty member to a course section |

| **USE CASE** | **Add Advisor to student - Success** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to add an advisor to a student. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Advisement” button.  3. Admin selects the desired faculty member to add an advisee to.  4. Admin clicks on the “Show Advised Students” to show their current advisees.  5. Admin selects the same faculty member selected in step 3 in the “Assign Faculty to Advise Student” section.  6. Admin chooses a student from the “Select Student” list to assign  7. Admin reviews information and clicks on the “Assign” button.  8. Student does not have more than 2 advisors or Advisor does not advise more than 10 students. |
| Exit | Admin has added an advisor to the student. |

| **USE CASE** | **Add Advisor to student - Failure** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to add an advisor to a student. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Advisement” button.  3. Admin selects the desired faculty member to add an advisee to.  4. Admin clicks on the “Show Advised Students” to show their current advisees.  5. Admin selects the same faculty member selected in step 3 in the “Assign Faculty to Advise Student” section.  6. Admin chooses a student from the “Select Student” list to assign  7. Admin reviews information and clicks on the “Assign” button.  8. Student has more than 2 advisors or Advisor advises more than 10 students. |
| Exit | Admin has been prevented from adding an advisor to the student. |

| **USE CASE** | **Remove Advisor from student** |
| --- | --- |
| List of actors | Admin |
| Initial Conditions | Admin has logged in and attempts to remove an advisor from a student. |
| Flow of events | 1. Admin logs in  2. Admin clicks on the “Advisement” button.  3. Admin selects the desired faculty member to remove an advisee from.  4. Admin clicks on the “Show Advised Students” to show their current advisees.  5. Admin selects the same faculty member selected in step 3 in the “Unassign Faculty from student” section.  6. Admin chooses a student from the “Select Student” list to unassign  7. Admin reviews information and clicks on the “Unassign” button. |
| Exit | Admin has removed an advisor from the student. |

**4.4 FACULTY**

| **USE CASE** | **View Personal Information** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty has logged in and attempted to view their personal information. |
| Flow of events | 1. Faculty logs in to their account.  2. Faculty clicks on the “Personal Information” Button |
| Exit | Faculty can view their personal information. |

| **USE CASE** | **Update Address** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty has logged in and attempts to change their address. |
| Flow of events | 1. Faculty logs in to their account.  2. Faculty clicks on the “Personal Information” Button.  4. Faculty scrolls down to “User Options” section.  5. Faculty is given list of what is valid to be changed.  6. Faculty inputs new address into their respective fields.  7. Faculty clicks the “Update User” button. |
| Exit | Faculty has now changed the Address listed on their account. |

| **USE CASE** | **Update Password** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty has logged in and attempts to change their password. |
| Flow of events | 1. Faculty logs in to their account.  2. Faculty clicks on the “Personal Information” Button.  4. Faculty scrolls down to “User Options” section.  5. Faculty is given list of what is valid to be changed.  6. Faculty inputs new Password into the respective field.  7. Faculty clicks the “Update User” button. |
| Exit | Faculty has now changed the password on their account. |

| **USE CASE** | **View Master Schedule** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty logs in and attempts to view the Master Schedule on the University website. |
| Flow of events | 1. Faculty logs in to their account.  2. Faculty clicks on the “Master Schedule” button. |
| Exit | Faculty can now view the Master Schedule. |

| **USE CASE** | **Search Master Schedule** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty logs in and attempts to search the Master schedule using a filter  (Department name, Course Name, Professor Name, Time Slot, Room/Building or Available Seats) |
| Flow of events | 1. Faculty logs in to their account.  2. Faculty clicks on the “Master Schedule” button.  3. Faculty selects which filter is to be used.  4. If applicable, Faculty inputs a specific value for the filter for a more specific search. |
| Exit | Faculty can search the Master Schedule using the filters they have chosen. |

| **USE CASE** | **Faculty can view personal Fall ‘23 schedule** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty has logged in and is attempting to view personal schedule of Fall ‘23 |
| Flow of events | 1. Faculty logs in. 2. Faculty is given all their course sections that they are teaching on their schedule. 3. Faculty filters the schedule to Fall ‘23. |
| Exit | The Faculty can view their Fall ‘23 Schedule |

| **USE CASE** | **Faculty can view personal Spring ‘24 schedule** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty viewing personal schedule of Spring ‘24 |
| Flow of events | 1. Faculty logs in. 2. Faculty is given all their course sections that they are teaching on their schedule. 3. Faculty filters the schedule to Spring ‘24. |
| Exit | The Faculty can view their Spring ‘24 Schedule |

| **USE CASE** | **Faculty views Roster for class of Fall ‘23 or Spring ‘24** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty attempts to view Roster |
| Flow of events | 1. Faculty logs in. 2. Faculty is given all their course sections that they are teaching on their schedule. 3. Faculty chooses which course section to view their class roster. 4. Faculty clicks on the “Class Roster” button for that course section. |
| Exit | The Faculty can view their Class Roster for a specific course section. |

| **USE CASE** | **Faculty views Student ID for class of Fall ‘23 or Spring ‘24** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty attempts to view a Student ID |
| Flow of events | 1. Faculty logs in. 2. Faculty is given all their course sections that they are teaching on their schedule. 3. Faculty chooses which course section to view their class roster. 4. Faculty clicks on the “Class Roster” button for that course section. 5. Faculty views the Class roster for the course section. |
| Exit | The Faculty can view a Student ID Number for a specific class roster. |

| **USE CASE** | **Faculty views Student Email for class of Fall ‘23 or Spring ‘24** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty attempts to view a Student Email |
| Flow of events | 1. Faculty logs in. 2. Faculty is given all their course sections that they are teaching on their schedule. 3. Faculty chooses which course section to view their class roster. 4. Faculty clicks on the “Class Roster” button for that course section. 5. Faculty views the Class roster for the course section. |
| Exit | The Faculty can view a Student’s Email for a specific class roster. |

| **USE CASE** | **Faculty views Transcript of Student for class of Fall ‘23 or Spring ‘24** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty attempts to view a Student Transcript |
| Flow of events | 1. Faculty logs in. 2. Faculty is given all their course sections that they are teaching on their schedule. 3. Faculty chooses which course section to view their class roster. 4. Faculty clicks on the “Class Roster” button for that course section. 5. Faculty views the Class roster for the course section. 6. Faculty chooses which student to view from the list. 7. Faculty clicks on the Student’s name 8. Faculty is taken to the Student’s Academic Profile 9. Faculty clicks on “Unofficial Transcript” 10. Faculty is taken to the Student’s Unofficial Transcript |
| Exit | The Faculty can view a Student’s Unofficial Transcript for a specific class roster. |

| **USE CASE** | **Faculty views Advisors of Student for class of Fall ‘23 or Spring ‘24** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty attempts to view a Student Advisors |
| Flow of events | 1. Faculty logs in. 2. Faculty is given all their course sections that they are teaching on their schedule. 3. Faculty chooses which course section to view their class roster. 4. Faculty clicks on the “Class Roster” button for that course section. 5. Faculty views the Class roster for the course section. 6. Faculty chooses which student to view from the list. 7. Faculty clicks on the Student’s name 8. Faculty is taken to the Student’s Academic Profile |
| Exit | The Faculty can view a Student’s Advisors for a specific class roster. |

| **USE CASE** | **Faculty keeping track of attendance - Success** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty marking which Students are in class and which Students are not in class |
| Flow of events | 1. Faculty logs in. 2. Faculty is given all their course sections that they are teaching on their schedule. 3. Faculty chooses which course section to view their class roster. 4. Faculty clicks on the “Class Roster” button for that course section. 5. Faculty views the Class roster for the course section. 6. It is currently a course day and the class has started but is before 11:59PM 7. Faculty assigns attendance for each student. |
| Exit | The Faculty updated today’s class attendance before 11:59pm. |

| **USE CASE** | **Faculty keeping track of attendance - Failure** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty marking which Students are in class and which Students are not in class |
| Flow of events | 1. Faculty logs in. 2. Faculty is given all their course sections that they are teaching on their schedule. 3. Faculty chooses which course section to view their class roster. 4. Faculty clicks on the “Class Roster” button for that course section. 5. Faculty views the Class roster for the course section. 6. It is currently not a course day or the class has not started or is after 11:59PM on the day of the class. 7. Faculty cannot assign attendance for each student. |
| Exit | The Faculty was prevented from taking attendance for the class roster. |

| **USE CASE** | **Faculty assigning semester grade - Success** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty has to assign semester grade before time-slot |
| Flow of events | 1. Faculty logs in.  2. Faculty is given all their course  sections that they are teaching on their schedule.  3. Faculty chooses which course section to view their class roster.  4. Faculty clicks on the “Class Roster” button for that course section.  5. Faculty views the Class roster for the course section.  6. It is currently between the end of the semester and 4 days after.  7. Faculty assigns grades to each student. |
| Exit | The Faculty assigned semester grades during the allotted time slot |

| **USE CASE** | **Faculty assigning semester grade - Failure** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty has to assign semester grade before time-slot |
| Flow of events | 1. Faculty logs in.  2. Faculty is given all their course  sections that they are teaching on their schedule.  3. Faculty chooses which course section to view their class roster.  4. Faculty clicks on the “Class Roster” button for that course section.  5. Faculty views the Class roster for the course section.  6. It is currently not between the end of the semester and 4 days after.  7. Faculty cannot assign grades. |
| Exit | The Faculty has been prevented from assigning grades. |

| **USE CASE** | **Faculty Teaching Restrictions** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty teaching load based on employment status. |
| Flow of events | 1. Faculty member's employment status is identified as full-time or part-time. 2. If the faculty member is full-time, the system checks if they are already teaching 2 classes. If not, they can be assigned to another class. If they are teaching 2 classes, no further assignments are allowed. 3. If the faculty member is part-time, the system checks if they are already teaching 1 class. If not, they can be assigned to another class. If they are teaching 1 class, no further assignments are allowed. |
| Exit | Faculty members are assigned to classes per the defined limits: full-time can teach up to 2 classes, part-time can teach up to 1 class. |

| **USE CASE** | **Limit on Faculty Affiliations** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | The constraint on the number of departments a faculty member can be affiliated with. |
| Flow of events | 1. Faculty member's affiliation with departments is identified. 2. The system checks the number of departments the faculty member is currently affiliated with. 3. If the faculty member is affiliated with 3 or fewer departments, they can continue their affiliation with additional departments if desired. 4. If the faculty member is already affiliated with 3 departments, the system prevents further affiliations and displays an error message. |
| Exit | Faculty members affiliated with more than 3 Dept. is not allowed. |

| **USE CASE** | **Restriction on Teaching Outside Department** |
| --- | --- |
| List of actors | Faculty Member, Department Coordinator |
| Initial Conditions | Faculty members cannot teach courses outside of their affiliated department. |
| Flow of events | 1.Faculty member expresses interest in teaching a course.  2. The system checks the faculty member's department affiliation.  3. The system checks the department assignment of the course.  4. If the faculty member's department affiliation matches the department assignment of the course, they are allowed to teach the course.  5. If the faculty member's department affiliation does not match the department assignment of the course, the system prevents the assignment and notifies the faculty member and Department Coordinator. |
| Exit | Faculty Members are not allowed to teach outside their dept. |

| **USE CASE** | **Restriction on Taking Attendance Outside Department** |
| --- | --- |
| List of actors | Faculty Members , Department Coordinator |
| Initial Conditions | Faculty members cannot take attendance for courses outside of their affiliated department. |
| Flow of events | 1.Faculty member is responsible for taking attendance in a course.  2. The system checks the faculty member's department affiliation.  3. The system checks the department assignment of the course for which attendance is to be taken.  4. If the faculty member's department affiliation matches the department assignment of the course, they are allowed to take attendance.  5. If the faculty member's department affiliation does not match the department assignment of the course, the system prevents them from taking attendance and notifies the faculty member and Department Coordinator |
| Exit | Faculty members cannot take and update attendance outside their dept. |

| **USE CASE** | **Restriction on Faculty Grade Assignment** |
| --- | --- |
| List of actors | Faculty Members, Department Coordinator, Registra |
| Initial Conditions | Faculty members cannot assign grades to students for courses |
| Flow of events | 1.Faculty member completes the teaching of a course.  2. The system identifies that the course has been completed.  3. The system checks for faculty member privileges.  4. Since faculty members do not have grading privileges, they are unable to assign grades directly.  5. The registrar or Department Coordinator who have the authority to assign grades, are responsible for grading the students in the course. |
| Exit | Faculty members cannot assign grades directly. |

| **USE CASE** | **Faculty Advisor Requirement** |
| --- | --- |
| List of actors | Faculty |
| Initial Conditions | Faculty members cannot have more than 10 students assigned to them. |
| Flow of events | 1.Faculty member's role as an advisor is identified.  2. The system checks the number of advisees the faculty member currently has.  3. If the faculty member has 10 or more advisees, they meet the requirement and can continue advising.  4. If the faculty member has fewer than 10 advisees, the system notifies them and recommends assigning more advisees to meet the requirement. |
| Exit | Faculty members can have up to 10 students assigned to them. |

| **USE CASE** | **Restriction on Changing Email Addresses and User IDs** |
| --- | --- |
| List of actors | Faculty, Admin |
| Initial Conditions | Faculty is not allowed to change their email addresses and user ID. |
| Flow of events | 1. User requests to change their email address or user ID. 2. The system checks the user's request. 3. The system verifies that the requested change is related to their email address or user ID. 4. If the change request is related to their email address or user ID, the system prevents the change and notifies the user that such changes are not permitted. 5. If the change request is related to other account details (e.g., password, contact information), the system allows the user to make the necessary updates. |
| Exit | Faculty are unable to change their email addresses or user IDs within the system. |

| **USE CASE** | **Restriction on Faculty Access to User Information** |
| --- | --- |
| List of actors | Faculty, Admin |
| Initial Conditions | Faculty members are not allowed to access other users' information, except for their advisees and students in their assigned course sections. |
| Flow of events | 1. Faculty member attempts to access user information. 2. The system checks the faculty member's access permissions. 3. If the faculty member's access request is related to their own advisees, the system allows access to the information. 4. If the access request is related to other users who are not their advisees, the system denies access and notifies the faculty member that they do not have permission to view this information. |
| Exit | Faculty can only access the information of their advisees and students in their assigned course sections. Access to other users' information is restricted. |

**4.5 STATS OFFICE**

| **USE CASE** | **View Personal Information** |
| --- | --- |
| List of actors | Stats Office |
| Initial Conditions | Stats Office has logged in and attempted to view their personal information. |
| Flow of events | 1. Stats Office logs in to their account.  2. Stats Office clicks on the “Personal Information” Button |
| Exit | Stats Office can view their personal information. |

| **USE CASE** | **Update Address** |
| --- | --- |
| List of actors | Stats Office |
| Initial Conditions | Stats Office has logged in and attempts to change their address. |
| Flow of events | 1. Stats Office logs in to their account.  2. Stats Office clicks on the “Personal Information” Button.  4. Stats Office scrolls down to “User Options” section.  5. Stats Office is given list of what is valid to be changed.  6. Stats Office inputs new address into their respective fields.  7. Stats Office clicks the “Update User” button. |
| Exit | Stats Office has now changed the Address listed on their account. |

| **USE CASE** | **Update Password** |
| --- | --- |
| List of actors | Stats Office |
| Initial Conditions | Stats Office has logged in and attempts to change their password. |
| Flow of events | 1. Stats Office logs in to their account.  2. Stats Office clicks on the “Personal Information” Button.  4. Stats Office scrolls down to “User Options” section.  5. Stats Office is given list of what is valid to be changed.  6. Stats Office inputs new Password into the respective field.  7. Stats Office clicks the “Update User” button. |
| Exit | Stats Office has now changed the password on their account. |

| **USE CASE** | **View Master Schedule** |
| --- | --- |
| List of actors | Stats Office |
| Initial Conditions | Stats Office logs in and attempts to view the Master Schedule on the University website. |
| Flow of events | 1. Stats Office logs in to their account.  2. Master Schedule is located on the Stats Office home page. |
| Exit | Stats Office can now view the Master Schedule. |

| **USE CASE** | **Search Master Schedule** |
| --- | --- |
| List of actors | Stats Office |
| Initial Conditions | Stats Office logs in and attempts to search the Master schedule using a filter  (Department name, Course Name, Professor Name, Time Slot, Room/Building or Available Seats) |
| Flow of events | 1. Stats Office logs in to their account.  2. Stats Office views the Master Schedule.  3. Stats Office selects which filter is to be used.  4. If applicable, Stats Office inputs a specific value for the filter for a more specific search. |
| Exit | Stats Office can search the Master Schedule using the filters they have chosen. |

| **USE CASE** | **View Percentage of Graduate Students** |
| --- | --- |
| List of actors | Statistics Office |
| Initial Conditions | Statistics Office attempts to view the percentage of graduate students |
| Flow of events | 1. The stats office logs in.  2. The stats office clicks the “Statistics” button.  3. The system retrieves the previously generated school statistics from the database. |
| Exit | System displays the retrieved statistics to the stats office. |

| **USE CASE** | **View Percentage of Full-Time Students** |
| --- | --- |
| List of actors | Statistics Office |
| Initial Conditions | Statistics Office attempts to view the percentage of Full-Time Students |
| Flow of events | 1. The stats office logs in.  2. The stats office clicks the “Statistics” button.  3. The system retrieves the previously generated school statistics from the database. |
| Exit | System displays the Percentage of Full-Time Students. |

| **USE CASE** | **View Percentage of Part-Time Students** |
| --- | --- |
| List of actors | Statistics Office |
| Initial Conditions | Statistics Office attempts to view the percentage of part-time students. |
| Flow of events | 1. The stats office logs in.  2. The stats office clicks the “Statistics” button.  3. The system retrieves the previously generated school statistics from the database. |
| Exit | System displays the retrieved statistics to the stats office. |

| **USE CASE** | **View Number of Master Students** |
| --- | --- |
| List of actors | Statistics Office |
| Initial Conditions | Statistics Office attempts to view the percentage of Master students. |
| Flow of events | 1. The stats office logs in.  2. The stats office clicks the “Statistics” button.  3. The system retrieves the previously generated school statistics from the database. |
| Exit | System displays the retrieved statistics to the stats office. |

| **USE CASE** | **View Number of Ph.D. Students** |
| --- | --- |
| List of actors | Statistics Office |
| Initial Conditions | Statistics Office attempts to view the Number of Ph.D. students. |
| Flow of events | 1. The stats office logs in.  2. The stats office clicks the “Statistics” button.  3. The system retrieves the previously generated school statistics from the database. |
| Exit | System displays the retrieved statistics to the stats office. |

| **USE CASE** | **View Graduation Rate** |
| --- | --- |
| List of actors | Statistics Office |
| Initial Conditions | Statistics Office attempts to view the Graduation rate based on the number of years specified by Stats Office |
| Flow of events | 1. The stats office logs in.  2. The stats office clicks the “Statistics” button.  3. The system retrieves the previously generated school statistics from the database. |
| Exit | System displays the retrieved statistics to the stats office. |

| **USE CASE** | **View Number of Majors** |
| --- | --- |
| List of actors | Statistics Office |
| Initial Conditions | Statistics Office attempts to view the Number of different academic Majors |
| Flow of events | 1. The stats office logs in.  2. The stats office clicks the “Statistics” button.  3. The system retrieves the previously generated school statistics from the database. |
| Exit | System displays the retrieved statistics to the stats office. |

| **USE CASE** | **View Number of Minors** |
| --- | --- |
| List of actors | Statistics Office |
| Initial Conditions | Statistics Office attempts to view the Number of different academic Minors. |
| Flow of events | 1. The stats office logs in.  2. The stats office clicks the “Statistics” button.  3. The system retrieves the previously generated school statistics from the database. |
| Exit | System displays the retrieved statistics to the stats office. |

| **USE CASE** | **Access Restrictions for the Statistics Office** |
| --- | --- |
| List of actors | Statistics Office |
| Initial Conditions | The system has user roles and permissions defined in the SRS. The Statistics Office user accounts are created and configured according to their specified roles.. |
| Flow of events | 1. The System Administrator configures user roles and permissions based on the Statistics Office's SRS requirements.  2. The Statistics Office users log into the system with their credentials.  3. The system checks the user's role and permissions for each action they attempt to perform.  4. If the user's role and permissions permit the action, the system allows it. Otherwise, it denies access and displays an error message. |
| Exit | The Statistics Office users have access to system resources and functions as defined in their Required SRS. |

**6. Schema**

-- phpMyAdmin SQL Dump

-- version 5.2.1

-- https://www.phpmyadmin.net/

--

-- Host: 127.0.0.1:3306

-- Generation Time: Dec 11, 2023 at 01:13 AM

-- Server version: 10.6.16-MariaDB-cll-lve

-- PHP Version: 7.2.34

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Database: `u353026399\_systemdesign`

--

-- --------------------------------------------------------

--

-- Table structure for table `admin`

--

CREATE TABLE `admin` (

`AdminID` int(11) NOT NULL,

`AdminType` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `adminpl0`

--

CREATE TABLE `adminpl0` (

`AdminID` int(11) NOT NULL,

`priorityType` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `adminpl1`

--

CREATE TABLE `adminpl1` (

`AdminID` int(11) NOT NULL,

`priorityType` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `advisor`

--

CREATE TABLE `advisor` (

`StudentID` int(11) NOT NULL,

`FacultyID` int(11) NOT NULL,

`DOA` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `attendance`

--

CREATE TABLE `attendance` (

`StudentID` int(11) NOT NULL,

`CRN` int(11) NOT NULL,

`CourseID` varchar(100) DEFAULT NULL,

`ClassDate` date NOT NULL,

`Present` tinyint(1) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `building`

--

CREATE TABLE `building` (

`BuildingID` varchar(25) NOT NULL,

`BuildingName` varchar(100) DEFAULT NULL,

`BuildingType` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `classroom`

--

CREATE TABLE `classroom` (

`RoomID` varchar(25) NOT NULL,

`NumOfSeats` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `course`

--

CREATE TABLE `course` (

`CourseID` varchar(100) NOT NULL,

`CourseName` varchar(100) DEFAULT NULL,

`DeptID` varchar(25) DEFAULT NULL,

`Credits` int(11) DEFAULT NULL,

`Description` varchar(5000) DEFAULT NULL,

`CourseType` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `courseprerequisite`

--

CREATE TABLE `courseprerequisite` (

`CourseID` varchar(100) NOT NULL,

`PRcourseID` varchar(100) NOT NULL,

`MinGrade` varchar(3) DEFAULT NULL,

`DOLU` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `coursesection`

--

CREATE TABLE `coursesection` (

`CRN` int(11) NOT NULL,

`CourseID` varchar(100) DEFAULT NULL,

`SectionNum` int(11) DEFAULT NULL,

`FacultyID` int(11) DEFAULT NULL,

`TimeSlotID` varchar(100) DEFAULT NULL,

`RoomID` varchar(25) DEFAULT NULL,

`SemesterID` int(11) DEFAULT NULL,

`AvailableSeats` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `day`

--

CREATE TABLE `day` (

`DayID` int(11) NOT NULL,

`WeekDay` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `dept`

--

CREATE TABLE `dept` (

`DeptID` varchar(25) NOT NULL,

`DeptName` varchar(100) DEFAULT NULL,

`ChairID` int(11) DEFAULT NULL,

`DeptManager` int(11) DEFAULT NULL,

`Email` varchar(100) DEFAULT NULL,

`PhoneNum` varchar(100) DEFAULT NULL,

`RoomID` varchar(25) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `enrollment`

--

CREATE TABLE `enrollment` (

`StudentID` int(11) NOT NULL,

`CRN` int(11) NOT NULL,

`Grade` varchar(3) DEFAULT NULL,

`DOE` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `faculty`

--

CREATE TABLE `faculty` (

`FacultyID` int(11) NOT NULL,

`Position` varchar(100) DEFAULT NULL,

`Specialty` varchar(100) DEFAULT NULL,

`FacultyType` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `facultydept`

--

CREATE TABLE `facultydept` (

`FacultyID` int(11) NOT NULL,

`DeptID` varchar(25) NOT NULL,

`PercentTime` varchar(100) DEFAULT NULL,

`DOA` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `facultyft`

--

CREATE TABLE `facultyft` (

`FacultyID` int(11) NOT NULL,

`NumOfClass` int(11) DEFAULT NULL,

`OfficeID` varchar(25) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `facultyhistory`

--

CREATE TABLE `facultyhistory` (

`FacultyID` int(11) NOT NULL,

`CRN` int(11) NOT NULL,

`CourseID` varchar(100) DEFAULT NULL,

`SemesterID` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `facultypt`

--

CREATE TABLE `facultypt` (

`FacultyID` int(11) NOT NULL,

`NumOfClass` int(11) DEFAULT NULL,

`OfficeID` varchar(25) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `gradstudent`

--

CREATE TABLE `gradstudent` (

`StudentID` int(11) NOT NULL,

`DeptID` varchar(25) DEFAULT NULL,

`GradStudentType` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `gradstudentft`

--

CREATE TABLE `gradstudentft` (

`StudentID` int(11) NOT NULL,

`Standing` varchar(100) DEFAULT NULL,

`CreditEarned` int(11) DEFAULT NULL,

`QualifyExam` tinyint(1) DEFAULT NULL,

`Thesis` tinyint(1) DEFAULT NULL,

`LowCredits` int(11) DEFAULT NULL,

`HighCredits` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `gradstudentpt`

--

CREATE TABLE `gradstudentpt` (

`StudentID` int(11) NOT NULL,

`Standing` varchar(100) DEFAULT NULL,

`CreditEarned` int(11) DEFAULT NULL,

`QualifyExam` tinyint(1) DEFAULT NULL,

`Thesis` tinyint(1) DEFAULT NULL,

`LowCredits` int(11) DEFAULT NULL,

`HighCredits` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `hold`

--

CREATE TABLE `hold` (

`HoldID` int(11) NOT NULL,

`StudentID` int(11) DEFAULT NULL,

`DateOfHold` date DEFAULT NULL,

`HoldType` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `lab`

--

CREATE TABLE `lab` (

`RoomID` varchar(25) NOT NULL,

`NumOfWorkStations` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `logintable`

--

CREATE TABLE `logintable` (

`UID` int(11) NOT NULL,

`Email` varchar(100) DEFAULT NULL,

`Password` varchar(100) DEFAULT NULL,

`NumOfLogin` varchar(100) DEFAULT NULL,

`LockedOut` tinyint(1) DEFAULT NULL,

`UserType` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `major`

--

CREATE TABLE `major` (

`MajorID` int(11) NOT NULL,

`DeptID` varchar(25) DEFAULT NULL,

`MajorName` varchar(300) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `majorprerequisite`

--

CREATE TABLE `majorprerequisite` (

`MajorID` int(11) NOT NULL,

`PRmajorID` varchar(11) NOT NULL,

`MinGrade` varchar(3) DEFAULT NULL,

`DOLU` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `minor`

--

CREATE TABLE `minor` (

`MinorID` int(11) NOT NULL,

`DeptID` varchar(25) DEFAULT NULL,

`MinorName` varchar(300) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `minorprerequisite`

--

CREATE TABLE `minorprerequisite` (

`MinorID` int(11) NOT NULL,

`PRminorID` varchar(11) NOT NULL,

`MinGrade` varchar(3) DEFAULT NULL,

`DOLU` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `office`

--

CREATE TABLE `office` (

`RoomID` varchar(25) NOT NULL,

`OfficeID` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `periodd`

--

CREATE TABLE `periodd` (

`PeriodID` int(11) NOT NULL,

`StartTime` time NOT NULL,

`EndTime` time NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `room`

--

CREATE TABLE `room` (

`RoomID` varchar(25) NOT NULL,

`BuildingID` varchar(25) NOT NULL,

`RoomNum` int(11) DEFAULT NULL,

`RoomType` varchar(50) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `semester`

--

CREATE TABLE `semester` (

`SemesterID` int(11) NOT NULL,

`SemesterName` varchar(100) DEFAULT NULL,

`SemesterYear` year(4) NOT NULL,

`StartTime` date NOT NULL,

`EndTime` date NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `statsoffice`

--

CREATE TABLE `statsoffice` (

`StatsID` int(11) NOT NULL,

`GAD` date DEFAULT NULL,

`ClientName` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `student`

--

CREATE TABLE `student` (

`StudentID` int(11) NOT NULL,

`StudentYear` varchar(100) DEFAULT NULL,

`StudentType` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `studenthistory`

--

CREATE TABLE `studenthistory` (

`StudentID` int(11) NOT NULL,

`CRN` int(11) NOT NULL,

`CourseID` varchar(100) DEFAULT NULL,

`SemesterID` int(11) DEFAULT NULL,

`Grade` varchar(3) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `studentmajor`

--

CREATE TABLE `studentmajor` (

`StudentID` int(11) NOT NULL,

`MajorID` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `studentminor`

--

CREATE TABLE `studentminor` (

`StudentID` int(11) NOT NULL,

`MinorID` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `timeslot`

--

CREATE TABLE `timeslot` (

`TimeSlotID` varchar(100) NOT NULL,

`DayID` int(11) NOT NULL,

`PeriodID` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `undergradstudent`

--

CREATE TABLE `undergradstudent` (

`StudentID` int(11) NOT NULL,

`DeptID` varchar(25) DEFAULT NULL,

`UnderGradStudentType` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `undergradstudentft`

--

CREATE TABLE `undergradstudentft` (

`StudentID` int(11) NOT NULL,

`Standing` varchar(100) DEFAULT NULL,

`LowCredits` int(11) DEFAULT NULL,

`HighCredits` int(11) DEFAULT NULL,

`CreditEarned` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `undergradstudentpt`

--

CREATE TABLE `undergradstudentpt` (

`StudentID` int(11) NOT NULL,

`Standing` varchar(100) DEFAULT NULL,

`LowCredits` int(11) DEFAULT NULL,

`HighCredits` int(11) DEFAULT NULL,

`CreditEarned` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `user`

--

CREATE TABLE `user` (

`UID` int(11) NOT NULL,

`FirstName` varchar(100) DEFAULT NULL,

`LastName` varchar(100) DEFAULT NULL,

`Gender` varchar(2) DEFAULT NULL,

`DOB` date DEFAULT NULL,

`Street` varchar(100) DEFAULT NULL,

`City` varchar(100) DEFAULT NULL,

`State` varchar(100) DEFAULT NULL,

`ZipCode` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

--

-- Indexes for dumped tables

--

--

-- Indexes for table `admin`

--

ALTER TABLE `admin`

ADD PRIMARY KEY (`AdminID`),

ADD KEY `AdminID` (`AdminID`);

--

-- Indexes for table `adminpl0`

--

ALTER TABLE `adminpl0`

ADD PRIMARY KEY (`AdminID`),

ADD KEY `AdminID` (`AdminID`);

--

-- Indexes for table `adminpl1`

--

ALTER TABLE `adminpl1`

ADD PRIMARY KEY (`AdminID`),

ADD KEY `AdminID` (`AdminID`);

--

-- Indexes for table `advisor`

--

ALTER TABLE `advisor`

ADD PRIMARY KEY (`StudentID`,`FacultyID`),

ADD KEY `FacultyID` (`FacultyID`),

ADD KEY `StudentID` (`StudentID`);

--

-- Indexes for table `attendance`

--

ALTER TABLE `attendance`

ADD PRIMARY KEY (`StudentID`,`CRN`,`ClassDate`),

ADD KEY `CRN` (`CRN`),

ADD KEY `StudentID` (`StudentID`),

ADD KEY `attendance\_ibfk\_3` (`CourseID`);

--

-- Indexes for table `building`

--

ALTER TABLE `building`

ADD PRIMARY KEY (`BuildingID`),

ADD KEY `BuildingID` (`BuildingID`);

--

-- Indexes for table `classroom`

--

ALTER TABLE `classroom`

ADD PRIMARY KEY (`RoomID`),

ADD KEY `RoomID` (`RoomID`);

--

-- Indexes for table `course`

--

ALTER TABLE `course`

ADD PRIMARY KEY (`CourseID`),

ADD KEY `CourseID` (`CourseID`),

ADD KEY `course\_ibfk\_1` (`DeptID`);

--

-- Indexes for table `courseprerequisite`

--

ALTER TABLE `courseprerequisite`

ADD PRIMARY KEY (`CourseID`,`PRcourseID`) USING BTREE,

ADD KEY `CourseID` (`CourseID`,`PRcourseID`),

ADD KEY `courseprerequisite\_ibfk\_2` (`PRcourseID`);

--

-- Indexes for table `coursesection`

--

ALTER TABLE `coursesection`

ADD PRIMARY KEY (`CRN`),

ADD KEY `CRN` (`CRN`),

ADD KEY `coursesection\_ibfk\_1` (`CourseID`),

ADD KEY `coursesection\_ibfk\_2` (`FacultyID`),

ADD KEY `coursesection\_ibfk\_4` (`RoomID`),

ADD KEY `coursesection\_ibfk\_5` (`SemesterID`),

ADD KEY `coursesection\_ibfk\_6` (`TimeSlotID`);

--

-- Indexes for table `day`

--

ALTER TABLE `day`

ADD PRIMARY KEY (`DayID`),

ADD KEY `DayID` (`DayID`);

--

-- Indexes for table `dept`

--

ALTER TABLE `dept`

ADD PRIMARY KEY (`DeptID`),

ADD UNIQUE KEY `DeptID\_2` (`DeptID`),

ADD KEY `DeptID` (`DeptID`),

ADD KEY `dept\_ibfk\_1` (`RoomID`),

ADD KEY `dept\_ibfk\_2` (`ChairID`),

ADD KEY `dept\_ibfk\_3` (`DeptManager`);

--

-- Indexes for table `enrollment`

--

ALTER TABLE `enrollment`

ADD PRIMARY KEY (`StudentID`,`CRN`),

ADD KEY `CRN` (`CRN`),

ADD KEY `StudentID` (`StudentID`);

--

-- Indexes for table `faculty`

--

ALTER TABLE `faculty`

ADD PRIMARY KEY (`FacultyID`),

ADD KEY `FacultyID` (`FacultyID`);

--

-- Indexes for table `facultydept`

--

ALTER TABLE `facultydept`

ADD PRIMARY KEY (`FacultyID`,`DeptID`) USING BTREE,

ADD KEY `FacultyID` (`FacultyID`),

ADD KEY `facultydept\_ibfk\_1` (`DeptID`);

--

-- Indexes for table `facultyft`

--

ALTER TABLE `facultyft`

ADD PRIMARY KEY (`FacultyID`),

ADD KEY `FacultyID` (`FacultyID`),

ADD KEY `OfficeID` (`OfficeID`);

--

-- Indexes for table `facultyhistory`

--

ALTER TABLE `facultyhistory`

ADD PRIMARY KEY (`FacultyID`,`CRN`),

ADD KEY `CRN` (`CRN`),

ADD KEY `FacultyID` (`FacultyID`),

ADD KEY `facultyhistory\_ibfk\_3` (`CourseID`),

ADD KEY `facultyhistory\_ibfk\_4` (`SemesterID`);

--

-- Indexes for table `facultypt`

--

ALTER TABLE `facultypt`

ADD PRIMARY KEY (`FacultyID`),

ADD KEY `FacultyID` (`FacultyID`),

ADD KEY `OfficeID` (`OfficeID`);

--

-- Indexes for table `gradstudent`

--

ALTER TABLE `gradstudent`

ADD PRIMARY KEY (`StudentID`),

ADD KEY `StudentID` (`StudentID`),

ADD KEY `DeptID` (`DeptID`);

--

-- Indexes for table `gradstudentft`

--

ALTER TABLE `gradstudentft`

ADD PRIMARY KEY (`StudentID`),

ADD KEY `StudentID` (`StudentID`);

--

-- Indexes for table `gradstudentpt`

--

ALTER TABLE `gradstudentpt`

ADD PRIMARY KEY (`StudentID`),

ADD KEY `StudentID` (`StudentID`);

--

-- Indexes for table `hold`

--

ALTER TABLE `hold`

ADD PRIMARY KEY (`HoldID`),

ADD KEY `StudentID` (`StudentID`);

--

-- Indexes for table `lab`

--

ALTER TABLE `lab`

ADD PRIMARY KEY (`RoomID`),

ADD KEY `RoomID` (`RoomID`);

--

-- Indexes for table `logintable`

--

ALTER TABLE `logintable`

ADD PRIMARY KEY (`UID`),

ADD KEY `UID` (`UID`);

--

-- Indexes for table `major`

--

ALTER TABLE `major`

ADD PRIMARY KEY (`MajorID`),

ADD KEY `MajorID` (`MajorID`),

ADD KEY `DeptID` (`DeptID`);

--

-- Indexes for table `majorprerequisite`

--

ALTER TABLE `majorprerequisite`

ADD PRIMARY KEY (`MajorID`,`PRmajorID`),

ADD KEY `MajorID` (`MajorID`,`PRmajorID`),

ADD KEY `majorprerequisite\_ibfk\_2` (`PRmajorID`);

--

-- Indexes for table `minor`

--

ALTER TABLE `minor`

ADD PRIMARY KEY (`MinorID`),

ADD KEY `MinorID` (`MinorID`),

ADD KEY `DeptID` (`DeptID`);

--

-- Indexes for table `minorprerequisite`

--

ALTER TABLE `minorprerequisite`

ADD PRIMARY KEY (`MinorID`,`PRminorID`),

ADD KEY `MinorID` (`MinorID`,`PRminorID`),

ADD KEY `PRminorID` (`PRminorID`);

--

-- Indexes for table `office`

--

ALTER TABLE `office`

ADD PRIMARY KEY (`RoomID`) USING BTREE,

ADD KEY `RoomID` (`RoomID`),

ADD KEY `OfficeID` (`OfficeID`);

--

-- Indexes for table `periodd`

--

ALTER TABLE `periodd`

ADD PRIMARY KEY (`PeriodID`),

ADD KEY `PeriodID` (`PeriodID`);

--

-- Indexes for table `room`

--

ALTER TABLE `room`

ADD PRIMARY KEY (`RoomID`) USING BTREE,

ADD KEY `BuildingID` (`BuildingID`),

ADD KEY `RoomID` (`RoomID`);

--

-- Indexes for table `semester`

--

ALTER TABLE `semester`

ADD PRIMARY KEY (`SemesterID`),

ADD KEY `SemesterID` (`SemesterID`);

--

-- Indexes for table `statsoffice`

--

ALTER TABLE `statsoffice`

ADD PRIMARY KEY (`StatsID`),

ADD UNIQUE KEY `StatsID` (`StatsID`);

--

-- Indexes for table `student`

--

ALTER TABLE `student`

ADD PRIMARY KEY (`StudentID`),

ADD KEY `StudentID` (`StudentID`);

--

-- Indexes for table `studenthistory`

--

ALTER TABLE `studenthistory`

ADD PRIMARY KEY (`StudentID`,`CRN`),

ADD KEY `StudentID` (`StudentID`),

ADD KEY `CourseID` (`CourseID`),

ADD KEY `CRN` (`CRN`),

ADD KEY `SemesterID` (`SemesterID`);

--

-- Indexes for table `studentmajor`

--

ALTER TABLE `studentmajor`

ADD PRIMARY KEY (`MajorID`,`StudentID`),

ADD KEY `studentmajor\_ibfk\_2` (`StudentID`);

--

-- Indexes for table `studentminor`

--

ALTER TABLE `studentminor`

ADD PRIMARY KEY (`MinorID`,`StudentID`),

ADD KEY `studentminor\_ibfk\_2` (`StudentID`);

--

-- Indexes for table `timeslot`

--

ALTER TABLE `timeslot`

ADD PRIMARY KEY (`TimeSlotID`,`DayID`) USING BTREE,

ADD KEY `DayID` (`DayID`),

ADD KEY `PeriodID` (`PeriodID`);

--

-- Indexes for table `undergradstudent`

--

ALTER TABLE `undergradstudent`

ADD PRIMARY KEY (`StudentID`),

ADD KEY `StudentID` (`StudentID`),

ADD KEY `DeptID` (`DeptID`);

--

-- Indexes for table `undergradstudentft`

--

ALTER TABLE `undergradstudentft`

ADD PRIMARY KEY (`StudentID`),

ADD KEY `StudentID` (`StudentID`);

--

-- Indexes for table `undergradstudentpt`

--

ALTER TABLE `undergradstudentpt`

ADD PRIMARY KEY (`StudentID`),

ADD KEY `StudentID` (`StudentID`);

--

-- Indexes for table `user`

--

ALTER TABLE `user`

ADD PRIMARY KEY (`UID`),

ADD KEY `UID` (`UID`);

--

-- AUTO\_INCREMENT for dumped tables

--

--

-- AUTO\_INCREMENT for table `hold`

--

ALTER TABLE `hold`

MODIFY `HoldID` int(11) NOT NULL AUTO\_INCREMENT;

--

-- Constraints for dumped tables

--

--

-- Constraints for table `admin`

--

ALTER TABLE `admin`

ADD CONSTRAINT `admin\_ibfk\_1` FOREIGN KEY (`AdminID`) REFERENCES `user` (`UID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `adminpl0`

--

ALTER TABLE `adminpl0`

ADD CONSTRAINT `adminpl0\_ibfk\_1` FOREIGN KEY (`AdminID`) REFERENCES `admin` (`AdminID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `adminpl1`

--

ALTER TABLE `adminpl1`

ADD CONSTRAINT `adminpl1\_ibfk\_1` FOREIGN KEY (`AdminID`) REFERENCES `admin` (`AdminID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `advisor`

--

ALTER TABLE `advisor`

ADD CONSTRAINT `advisor\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `student` (`StudentID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `advisor\_ibfk\_2` FOREIGN KEY (`FacultyID`) REFERENCES `faculty` (`FacultyID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `attendance`

--

ALTER TABLE `attendance`

ADD CONSTRAINT `attendance\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `student` (`StudentID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `attendance\_ibfk\_2` FOREIGN KEY (`CRN`) REFERENCES `coursesection` (`CRN`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `attendance\_ibfk\_3` FOREIGN KEY (`CourseID`) REFERENCES `course` (`CourseID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `classroom`

--

ALTER TABLE `classroom`

ADD CONSTRAINT `classroom\_ibfk\_1` FOREIGN KEY (`RoomID`) REFERENCES `room` (`RoomID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `course`

--

ALTER TABLE `course`

ADD CONSTRAINT `course\_ibfk\_1` FOREIGN KEY (`DeptID`) REFERENCES `dept` (`DeptID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `courseprerequisite`

--

ALTER TABLE `courseprerequisite`

ADD CONSTRAINT `courseprerequisite\_ibfk\_1` FOREIGN KEY (`CourseID`) REFERENCES `course` (`CourseID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `courseprerequisite\_ibfk\_2` FOREIGN KEY (`PRcourseID`) REFERENCES `course` (`CourseID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `coursesection`

--

ALTER TABLE `coursesection`

ADD CONSTRAINT `coursesection\_ibfk\_2` FOREIGN KEY (`FacultyID`) REFERENCES `faculty` (`FacultyID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `coursesection\_ibfk\_4` FOREIGN KEY (`RoomID`) REFERENCES `room` (`RoomID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `coursesection\_ibfk\_5` FOREIGN KEY (`SemesterID`) REFERENCES `semester` (`SemesterID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `coursesection\_ibfk\_6` FOREIGN KEY (`TimeSlotID`) REFERENCES `timeslot` (`TimeSlotID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `coursesection\_ibfk\_7` FOREIGN KEY (`CourseID`) REFERENCES `course` (`CourseID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `dept`

--

ALTER TABLE `dept`

ADD CONSTRAINT `dept\_ibfk\_1` FOREIGN KEY (`RoomID`) REFERENCES `room` (`RoomID`) ON DELETE SET NULL ON UPDATE CASCADE,

ADD CONSTRAINT `dept\_ibfk\_2` FOREIGN KEY (`ChairID`) REFERENCES `faculty` (`FacultyID`) ON DELETE SET NULL ON UPDATE CASCADE,

ADD CONSTRAINT `dept\_ibfk\_3` FOREIGN KEY (`DeptManager`) REFERENCES `faculty` (`FacultyID`) ON DELETE SET NULL ON UPDATE CASCADE;

--

-- Constraints for table `enrollment`

--

ALTER TABLE `enrollment`

ADD CONSTRAINT `enrollment\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `student` (`StudentID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `enrollment\_ibfk\_2` FOREIGN KEY (`CRN`) REFERENCES `coursesection` (`CRN`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `faculty`

--

ALTER TABLE `faculty`

ADD CONSTRAINT `faculty\_ibfk\_1` FOREIGN KEY (`FacultyID`) REFERENCES `user` (`UID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `facultydept`

--

ALTER TABLE `facultydept`

ADD CONSTRAINT `facultydept\_ibfk\_1` FOREIGN KEY (`DeptID`) REFERENCES `dept` (`DeptID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `facultydept\_ibfk\_2` FOREIGN KEY (`FacultyID`) REFERENCES `faculty` (`FacultyID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `facultyft`

--

ALTER TABLE `facultyft`

ADD CONSTRAINT `facultyft\_ibfk\_1` FOREIGN KEY (`FacultyID`) REFERENCES `faculty` (`FacultyID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `facultyft\_ibfk\_2` FOREIGN KEY (`OfficeID`) REFERENCES `office` (`RoomID`) ON DELETE SET NULL ON UPDATE CASCADE;

--

-- Constraints for table `facultyhistory`

--

ALTER TABLE `facultyhistory`

ADD CONSTRAINT `facultyhistory\_ibfk\_1` FOREIGN KEY (`FacultyID`) REFERENCES `faculty` (`FacultyID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `facultyhistory\_ibfk\_4` FOREIGN KEY (`SemesterID`) REFERENCES `semester` (`SemesterID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `facultyhistory\_ibfk\_6` FOREIGN KEY (`CourseID`) REFERENCES `course` (`CourseID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `facultyhistory\_ibfk\_7` FOREIGN KEY (`CRN`) REFERENCES `coursesection` (`CRN`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `facultypt`

--

ALTER TABLE `facultypt`

ADD CONSTRAINT `facultypt\_ibfk\_1` FOREIGN KEY (`FacultyID`) REFERENCES `faculty` (`FacultyID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `facultypt\_ibfk\_2` FOREIGN KEY (`OfficeID`) REFERENCES `office` (`RoomID`) ON DELETE SET NULL ON UPDATE CASCADE;

--

-- Constraints for table `gradstudent`

--

ALTER TABLE `gradstudent`

ADD CONSTRAINT `gradstudent\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `student` (`StudentID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `gradstudent\_ibfk\_2` FOREIGN KEY (`DeptID`) REFERENCES `dept` (`DeptID`) ON DELETE SET NULL ON UPDATE CASCADE;

--

-- Constraints for table `gradstudentft`

--

ALTER TABLE `gradstudentft`

ADD CONSTRAINT `gradstudentft\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `gradstudent` (`StudentID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `gradstudentpt`

--

ALTER TABLE `gradstudentpt`

ADD CONSTRAINT `gradstudentpt\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `gradstudent` (`StudentID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `hold`

--

ALTER TABLE `hold`

ADD CONSTRAINT `hold\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `student` (`StudentID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `lab`

--

ALTER TABLE `lab`

ADD CONSTRAINT `lab\_ibfk\_1` FOREIGN KEY (`RoomID`) REFERENCES `room` (`RoomID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `logintable`

--

ALTER TABLE `logintable`

ADD CONSTRAINT `logintable\_ibfk\_1` FOREIGN KEY (`UID`) REFERENCES `user` (`UID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `major`

--

ALTER TABLE `major`

ADD CONSTRAINT `major\_ibfk\_1` FOREIGN KEY (`DeptID`) REFERENCES `dept` (`DeptID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `majorprerequisite`

--

ALTER TABLE `majorprerequisite`

ADD CONSTRAINT `majorprerequisite\_ibfk\_1` FOREIGN KEY (`MajorID`) REFERENCES `major` (`MajorID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `majorprerequisite\_ibfk\_2` FOREIGN KEY (`PRmajorID`) REFERENCES `course` (`CourseID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `minor`

--

ALTER TABLE `minor`

ADD CONSTRAINT `minor\_ibfk\_1` FOREIGN KEY (`DeptID`) REFERENCES `dept` (`DeptID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `minorprerequisite`

--

ALTER TABLE `minorprerequisite`

ADD CONSTRAINT `minorprerequisite\_ibfk\_1` FOREIGN KEY (`MinorID`) REFERENCES `minor` (`MinorID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `minorprerequisite\_ibfk\_2` FOREIGN KEY (`PRminorID`) REFERENCES `course` (`CourseID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `office`

--

ALTER TABLE `office`

ADD CONSTRAINT `office\_ibfk\_1` FOREIGN KEY (`RoomID`) REFERENCES `room` (`RoomID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `room`

--

ALTER TABLE `room`

ADD CONSTRAINT `room\_ibfk\_1` FOREIGN KEY (`BuildingID`) REFERENCES `building` (`BuildingID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `statsoffice`

--

ALTER TABLE `statsoffice`

ADD CONSTRAINT `statsoffice\_ibfk\_1` FOREIGN KEY (`StatsID`) REFERENCES `user` (`UID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `student`

--

ALTER TABLE `student`

ADD CONSTRAINT `student\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `user` (`UID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `studenthistory`

--

ALTER TABLE `studenthistory`

ADD CONSTRAINT `studenthistory\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `student` (`StudentID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `studenthistory\_ibfk\_2` FOREIGN KEY (`CourseID`) REFERENCES `course` (`CourseID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `studenthistory\_ibfk\_3` FOREIGN KEY (`CRN`) REFERENCES `coursesection` (`CRN`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `studenthistory\_ibfk\_4` FOREIGN KEY (`SemesterID`) REFERENCES `semester` (`SemesterID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `studentmajor`

--

ALTER TABLE `studentmajor`

ADD CONSTRAINT `studentmajor\_ibfk\_1` FOREIGN KEY (`MajorID`) REFERENCES `major` (`MajorID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `studentmajor\_ibfk\_2` FOREIGN KEY (`StudentID`) REFERENCES `student` (`StudentID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `studentminor`

--

ALTER TABLE `studentminor`

ADD CONSTRAINT `studentminor\_ibfk\_1` FOREIGN KEY (`MinorID`) REFERENCES `minor` (`MinorID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `studentminor\_ibfk\_2` FOREIGN KEY (`StudentID`) REFERENCES `student` (`StudentID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `timeslot`

--

ALTER TABLE `timeslot`

ADD CONSTRAINT `timeslot\_ibfk\_1` FOREIGN KEY (`DayID`) REFERENCES `day` (`DayID`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `timeslot\_ibfk\_2` FOREIGN KEY (`PeriodID`) REFERENCES `periodd` (`PeriodID`) ON DELETE SET NULL ON UPDATE CASCADE;

--

-- Constraints for table `undergradstudent`

--

ALTER TABLE `undergradstudent`

ADD CONSTRAINT `undergradstudent\_ibfk\_1` FOREIGN KEY (`DeptID`) REFERENCES `dept` (`DeptID`) ON DELETE SET NULL ON UPDATE CASCADE,

ADD CONSTRAINT `undergradstudent\_ibfk\_2` FOREIGN KEY (`StudentID`) REFERENCES `student` (`StudentID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `undergradstudentft`

--

ALTER TABLE `undergradstudentft`

ADD CONSTRAINT `undergradstudentft\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `undergradstudent` (`StudentID`) ON DELETE CASCADE ON UPDATE CASCADE;

--

-- Constraints for table `undergradstudentpt`

--

ALTER TABLE `undergradstudentpt`

ADD CONSTRAINT `undergradstudentpt\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `undergradstudent` (`StudentID`) ON DELETE CASCADE ON UPDATE CASCADE;

COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

**7. Glossary**

Admin - Administrator

CRN - Course Reference Number

ERD - Entity relationship diagram

EERD - Enhanced entity relationship diagram

SRS - System requirements specification

Stats Office - Statistics Office

Undergrad - Undergraduate