## Introduction

Our project is a university portal system for university students and faculties. This system will keep important information and records of students and faculties saved and they can be checked anytime. They have to be registered first and after that, they can be logged into the account. Users can search for any task in the search bar. The system will be user-friendly and users will find it very simple to use. Besides, it will ensure the strong privacy of users personal information and records.

## Background And Product Context

The University portal is an inevitable tool for universities throughout the world. Hence the demand for university portal to develop is very high. Our portal has a sign-up and log-in system for both faculties and students. There will be grades for each student that can be accessed only by the faculties. Students and faculty’s personal information will be stored and displayed here. Students can also see the taken courses when they want. This can be accessed only by the students. Besides, there will be many services to make life easier such as semester drop, etc. Besides, there will be attendance of students and curriculum of departments.

## User Story & Use Case

1. **Title:** Home

**Actor**: Admin, Faculty, Instructor, Student

**Description**: It will have the following attributes:

* Name (Faculty, Instructor, Student)
* ID (Student)
* Email (Faculty, Instructor, Student)
* Degree (Student)
* Curriculum Name (Student)
* Faculty advisor (Student)
* Faculty Name (Student)
* Faculty Email (Student)

**Precondition:** None.

**Exception:** None.

1. **Title:** Profile Information

**Actor:** Faculty, Instructor, Student

**Description:** Students will able to see their information

**User Story:** As a student, I can check student information, as per my need.

**Precondition:** None.

**Exception:** None.

1. **Title:** Change Password

**Actor:** Admin, Faculty, Instructor, Student

**Description:** Admin, Student, Faculty, and Instructor will able to change their own password. But the admin will able to change everyone’s password.

**User Story:** As Admin/Student/Faculty/Instructor, I can change my password so that I can work on my account if I forget the password.

**Precondition:** Enter current password.

**Exception:** The following are the exceptions:

* Entered the wrong current password.
* New password must match criteria.
* Confirm password doesn’t match.

1. **Title:** Grade History

**Actor:** Student

**Description:** Check the grade history of all the completed courses.

**User Story:** As a student, I can check my previous semester’s grades so that I will be able to know my condition.

**Precondition:** None.

**Exception:** None.

1. **Title:** Payment Status

**Actor:** Student

**Description:** Check current balance.

**User Story:** As a student, I want to check my current balance if it is needed.

**Precondition:** None.

**Exception:** None.

1. **Title:** Attendance

**Actor:** Student

**Description:** View attendance.

**User Story:** As a student, I want to view attendance so that I can know about attendances of each courses of any semesters.

**Precondition:** None.

**Exception:** None.

1. **Title:** Faculty Evaluation

**Actor:** Student

**Description:** Evaluate faculty.

**User Story:** As a student, I can do faculty evaluations so that I will be eligible for advising.

**Precondition:** None.

**Exception:** All points must be checked.

1. **Title:** Curriculum

**Actor:** Student

**Description:** View enrolled curriculum.

**User Story:** As a student, I can see the curriculum so that I can take courses mentioned in the curriculum.

**Precondition:** None.

**Exception:** None.

1. **Title:** Message History

**Actor:** Student

**Description:** Check message inbox.

**User Story:** As a student, I can check messages so that I can learn about a new notice from the chairman, VC, and IT.

**Precondition:** None.

**Exception:** None.

1. **Title:** User Guides

**Actor:** Admin, Faculty, Instructor, Student

**Description:** View the following guidelines:

* Recovery Password
* Access Portal
* Join Online Class
* Email Password Reset
* Message History
* Login to Classroom Using the Mobile
* Email Through Mobile
* Add NID & Birth Certificate
* Advising
* Degree guideline

**User Story:** As a student/faculty/admin/instructor, I need a special option so that I can do the mentioned tasks as per my need.

**Precondition:** None.

**Exception:** None.

1. **Title:** Take Attendance

**Actor:** Faculty, Instructor

**Description:** Take students’ attendance.

**User Story:** As a faculty/instructor, I can take attendance so that I can do regular activity of academic without doing manually.

**Precondition:** None.

**Exception:** Fill in all the student’s attendance.

1. **Title:** Send Message

**Actor:** Faculty

**Description:** Send a message to students.

**User Story:** As a faculty, I can send messages to students so that I can inform students about class cancellations.

**Precondition:** None.

**Exception:** None.

1. **Title:** Grade Submission

**Actor:** Faculty

**Description:** Submit student’s grades.

**User Story:** As a faculty, I can submit students’ grades because of university authorities’ rules.

**Precondition:** None.

**Exception:** Must select from the predefined grade letter list.

1. **Title:** View Faculty Evaluation

**Actor:** Admin

**Description:** View faculty evaluation done by students.

**User Story:** As an admin, I can view faculty evaluations to judge faculties.

**Precondition:** None.

**Exception:** None.

1. **Title:** Add Student

**Actor:** Admin

**Description:** Add student to the database.

**User Story:** As an admin, I can add a new student admitted or newly advised.

**Precondition:** None.

**Exception:** The followings are exceptions:

* All the information must be given.
* Cannot have duplicates.

1. **Title:** Edit Student Information

**Actor:** Admin

**Description:** Edit student information.

**User Story:** As an admin, I can edit student information as per the student’s needs.

**Precondition:** The student must be in the database.

**Exception:** Cannot delete any information.

1. **Title:** Add Faculty

**Actor:** Admin

**Description:** Add faculty to the database.

**User Story:** As an admin, I can add faculty who are newly joined the university.

**Precondition:** None.

**Exception:** The followings are an exception:

* All the information must be given.
* Cannot have duplicates.

1. **Title:** Edit Faculty

**Actor:** Admin

**Description:** Edit faculty information.

**User Story:** As an admin, I can edit faculty information if it is needed by faculties.

**Precondition:** Faculty must be in the database.

**Exception:** Cannot delete any information.

1. **Title:** Add Instructor

**Actor:** Admin

**Description:** Add the instructor to the database.

**User Story:** As an admin, I can add instructors who are newly joined the institution.

**Precondition:** None.

**Exception:** The followings are exceptions:

* All the information must be given.
* Cannot have duplicates.

1. **Title:** Edit Instructor

**Actor:** Admin

**Description:** Edit instructor information.

**User Story:** As an admin, I can edit student information if it is needed.

**Precondition:** The instructor must be in the database.

**Exception:** Cannot delete any information.

1. **Title:** Advising

**Actor:** Admin, Faculty

**Description:** Enroll students in the course section.

**User Story:** As an admin, I can advise enrolling a student in each semester who has done a faculty evaluation.

**Precondition:** None.

**Exception:** The followings are exceptions:

* All the information must be given.
* Cannot have duplicates.
* Section cannot be full.

1. **Title:** Faculty Assign

**Actor:** Admin

**Description:** Assign faculty to the course section.

**User Story:** As an admin, I can assign faculty members to each section so that all sections can get their faculty.

**Precondition:** None.

**Exception:** The followings are an exception:

* All the information must be given.
* Cannot have duplicates.
* Only one faculty per section.

1. **Title:** Instructor Assign

**Actor:** Admin

**Description:** Assign instructor to course section.

**User Story:** As an admin, I can assign an instructor to each section so that all section can get their instructors.

**Precondition:** None.

**Exception:** The followings are exceptions:

* All the information must be given.
* Cannot have duplicates.
* Only one instructor per section.

1. **Title:** Payment Status

**Actor:** Student

**Description:** Check current balance.

**User Story:** As a student, I want to check my current balance if it is needed.

**Precondition:** None.

**Exception:** None.

1. **Title:** Payment History

**Actor:** Student

**Description:** Check past payments.

**User Story:** As a student, I want to check past payments as per my need.

**Precondition:** None.

**Exception:** None.

## Solution Description

* Front-end plan
* Main page
* Login or register a student or a teacher
* Student information/teacher information
* Student grade (only for student login)
* Taken courses (only for student login)
* Course curriculum (only for student login)
* Attendance
* Services
* Front-end mechanisms
* Bootstrap(js)
* Ajax(js)
* Back-end development

1. Account Creating, Password Recover

* Register form, and verification email.
* Login
* Forgot Password
* MySQL Database

1. Profile Management:

* Plan design and dev: MongoDB
* Student Profile
* Teacher Profile
* Others

1. Searching facility:

* Category based
* Back-end mechanisms
* PHP
* HTML
* MYSQL
* Editors
* VSCode
* Atom
* WebStorm
* Eclipse
* PhpStorm
* MySQL Workbench

# General Strategies

1. **Security**
   * Always use HTTPS to protect your page integrity and your users' privacy and data.
   * Enable Content Security Policy (CSP) to prevent Cross-Site Scripting (XSS).
   * Protect against SQL Injection attacks by using prepared statements and parameterized queries.
   * Encrypt sensitive data like passwords using secure hashing algorithms like Bcrypt or Argon2.
2. **Performance**
   * Minify CSS and JavaScript files.
   * Use proper HTML semantics for better SEO and accessibility.
   * Optimize images and other media files.
   * Implement pagination, lazy loading and infinite scroll to handle large data sets.

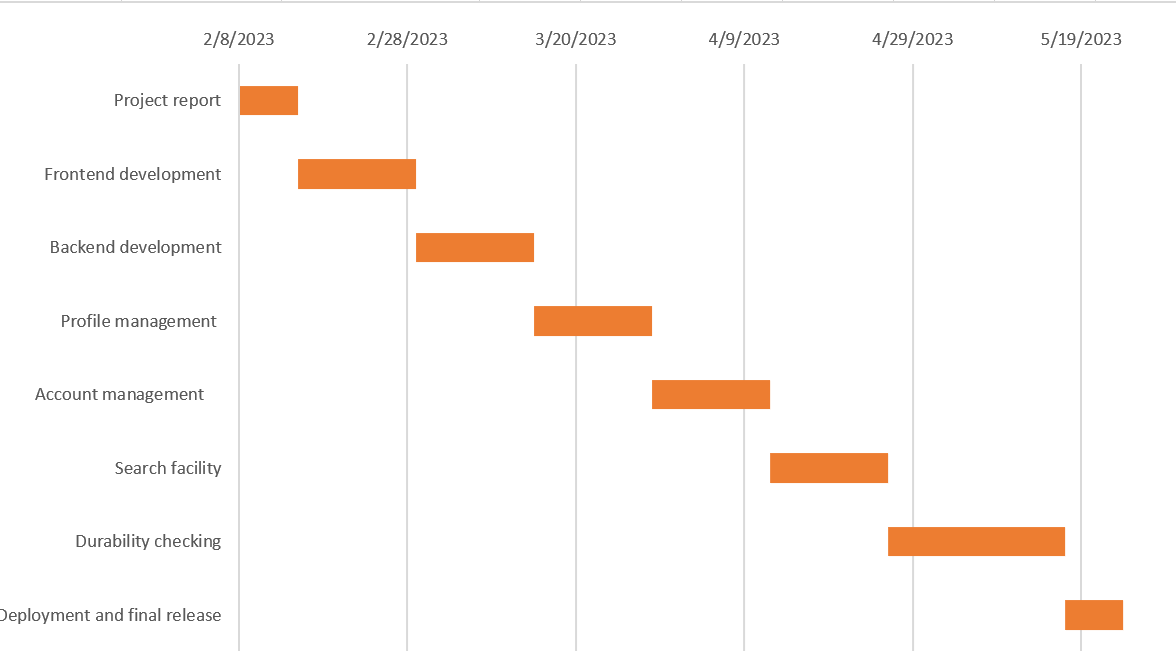
Now, for the individual pages:

1. **Login Page**
   * **Optimization:** Use AJAX to validate inputs and show validation errors without reloading the page.
   * **Security:** Use a server-side script (PHP) to validate the inputs again and authenticate the user. Avoid revealing specific reasons for failed logins (e.g., "incorrect password") to deter brute-force attacks.
2. **Password Reset Page**
   * **Optimization:** Implement AJAX for user interactions without reloading the page.
   * **Security:** Send a password reset link to the user's email instead of letting them input a new password directly. Always hash the new password before storing it in the database.
3. **Register Page**
   * **Optimization:** Use client-side validation (JavaScript) for a smoother user experience.
   * **Security:** Use server-side validation (PHP) to verify the inputs again. Encrypt passwords before storing them in the database.
4. **Dashboard Page**
   * **Optimization:** Load only necessary information at first, then use AJAX to load more as needed.
   * **Security:** Verify user roles and permissions before showing sensitive information or options.
5. **User Profile Page**
   * **Optimization:** Use AJAX for live editing of user details.
   * **Security:** Make sure only the authorized user and administrators can view and edit the profile.
6. **Department, Course, Section Pages**
   * **Optimization:** Implement pagination or infinite scrolling to handle large data sets.
   * **Security:** Use server-side checks to ensure only authorized users can make changes.
7. **Attendance, Grades, Payment, Message, Curriculum, Assignment, Evaluation Pages**
   * **Optimization:** Use AJAX for better interactivity and faster updates.
   * **Security:** Implement role-based access control. Only allow users with appropriate permissions to view and modify data.
8. **Admin Dashboard**
   * **Optimization:** Use AJAX to load data asynchronously and update the dashboard in real time.
   * **Security:** This page should only be accessible to users with admin privileges.
9. **User Management, Academic Management, Role Assignment Pages**
   * **Optimization:** Implement pagination to handle large number of users or data sets.
   * **Security:** Ensure only admins can view and modify data.

# Requirements

1. Hide visible errors such as db error dumps on the page. Show a generic error message to users and log the details server-side.
2. Cannot have any malformed page.
3. No errors detected on browser developer tools/console.
4. Good organization of codes.
5. Good functional flow, business flow, and cardinal flow.
6. Good styles used.
7. Variable naming and commenting. Must be readable.
8. Session handling.
9. Fast search option.
10. Good db connections, transactions, etc.
11. Use the latest technologies.
12. Good APIs used and proper calling.
13. Payment gateway integration (sandbox acceptable), PayPal, or Authorize.net.
14. Map integration.
15. Script loadings, and usage of CDN.
16. Cookie handling.
17. SQL injection prevention.
18. "Responsive" mobile-first development.
19. Oauth (from Google/Facebook/Twitter.
20. SMS integration (infobip etc..).
21. Good features: Lazy loading, Service worker.
22. Use password hashing provided by PHP.

## Development Plan

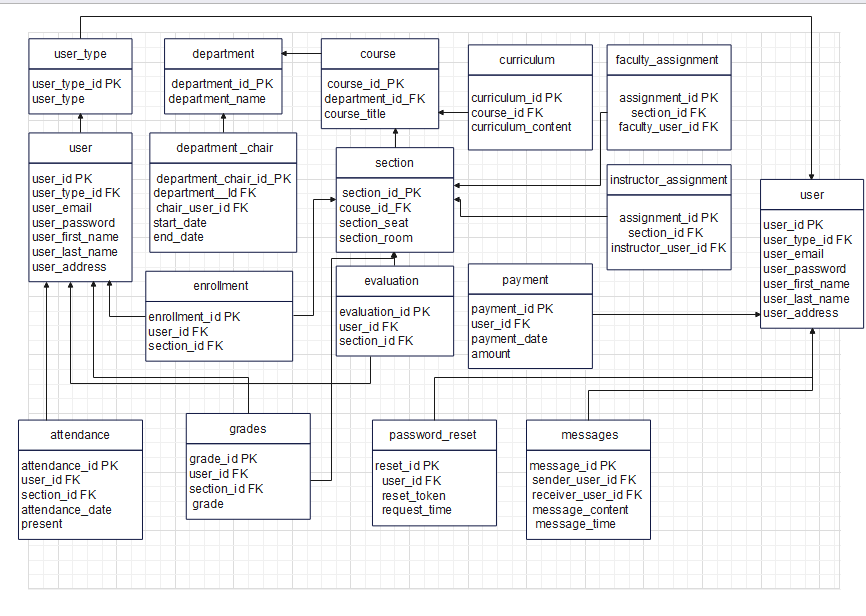


*Figure 1: Gantt chart of development plan*

## Collaboration Plan

| **Development** | **Member** |
| --- | --- |
| Front-end and back-end development | Sadman Sakib Khan Promit |
| Profile management (Database) | Sadman Sakib Khan Promit |
| Account management | Sadman Sakib Khan Promit |
| Durability checking | Sadman Sakib Khan Promit |
| Security checking | Sadman Sakib Khan Promit |

Database diagram:



Here's how the use cases map to the various pages and functionalities:

1. **Home:** Handled by the **Home Page** for all users.
2. **Profile Information:** Covered by the **Profile Page**.
3. **Change Password:** Managed in the **Profile Page** and for admins in the **User Management Page**.
4. **Grade History:** Part of the **Student Dashboard**.
5. **Payment Status:** Part of the **Student Dashboard** and **Finance Management Page** for Admin.
6. **Payment History:** Part of the **Student Dashboard** and **Finance Management Page** for Admin.
7. **Attendance:** Handled by the **Attendance Page** for students and the **Class Management Page** for faculty and instructors.
8. **Faculty Evaluation:** Handled by the **Evaluation Page** for students and viewed by admins on the **Academic Management Page**.
9. **Curriculum:** Part of the **Course Page** for students.
10. **Message History:** Handled by the **Messaging Page** for all users.
11. **User Guides:** This would be part of the **User Guides/Help Page**.
12. **Take Attendance:** Part of the **Class Management Page** for faculty and instructors.
13. **Send Message:** Handled by the **Messaging Page** for all users.
14. **Grade Submission:** Part of the **Grades Submission Page** for faculty.
15. **View Faculty Evaluation:** Part of the **Academic Management Page** for admins.
16. **View Student Payment History:** Handled in the **Finance Management Page** for admins.
17. **Add Student:** Part of the **User Management Page** for admins.
18. **Edit Student Information:** Also part of the **User Management Page** for admins.
19. **Add Faculty:** Part of the **User Management Page** for admins.
20. **Edit Faculty:** Also part of the **User Management Page** for admins.
21. **Add Instructor:** Part of the **User Management Page** for admins.
22. **Edit Instructor:** Also part of the **User Management Page** for admins.
23. **Advising:** This would be handled by the **Advising Page** for admins and faculty.
24. **Faculty Assign:** This would be handled by the **Role Assignment Page** for admins.
25. **Instructor Assign:** This would also be handled by the **Role Assignment Page** for admins.

# Pages

1. **Login Page:** For user authentication. All users (admin, faculty, instructor, student) can log in.
2. **Password Reset Page:** For users who forgot their passwords. It uses the password\_reset\_request table.
3. **Register Page:** For new users to register. All user types can register on this page.
4. **Dashboard Page:** This would be the landing page after a successful login. It will display different options depending on the user type (admin, faculty, instructor, student).
5. **User Profile Page:** Display and update details of a user. All user types can view and update their personal details.
6. **Department Pages:** For each department to display the department's information, list of courses, and the department chair.
7. **Course Pages:** For each course to display course information, sections, and enrollments. Students can enroll in courses from this page.
8. **Section Pages:** For each section to display section details and the list of enrolled students. Faculty and instructors can see their assigned sections.
9. **Attendance Page:** For faculty and instructors to take attendance. They can update the attendance records of students in their sections.
10. **Grades Page:** For faculty and instructors to assign grades to students. Students can view their grades here.
11. **Payment Page:** For students to pay tuition and fees.
12. **Message Page:** For users to communicate with each other. All user types can send and receive messages.
13. **Curriculum Page:** For faculty and instructors to view and update the curriculum of their respective courses.
14. **Assignment Pages:** For faculty and instructors to manage their assigned sections.
15. **Evaluation Page:** For faculty and instructors to evaluate their students, and for students to view their evaluations.
16. **Admin Dashboard:** An overview of the system, possibly showing some statistics, recent activities, or notifications. This page serves as a central hub to navigate to other administrative functionalities.
17. **User Management Page:** Admins can manage all user accounts here, regardless of their type (admin, faculty, instructor, student). Functions include creating, viewing, editing, and deleting user profiles.
18. **Academic Management Page:** This page will house the management functions for departments, courses, and sections. Admins can create, view, edit, and delete information related to these academic entities.
19. **Role Assignment Page:** On this page, admins can manage department chairs, faculty assignments, and instructor assignments. They can assign roles to users and change these assignments when necessary.
20. **Map Page:** It would be beneficial to have a dedicated page that provides the campus map and directions.

**Development phase:**

We have tried building 25 use cases by using html css. But we couldn't build the frontend of the Edit Student Information Page, EDIT FACULTY INFORMATION PAGE, EDIT INSTRUCTOR PAGE.   
  
 Our PHP script is sometimes unable to connect to the database that contains the data on students and professors, so sometimes exceptions occurred.

University portal websites often deal with sensitive data such as grades and personal information. If the website has security flaws such as SQL injection or cross-site scripting (XSS) vulnerabilities. We have faced these issues.  
When users submit forms on the website, their input needs to be validated to ensure it meets the required format and does not contain any malicious code. So we had to put validation correctly,   
As the university portal website grows in size and complexity, performance issues can arise if the PHP code is not optimized. We tried to optimize the php code.

When creating a university portal website, a "TypeError: Cannot read property 'x' of undefined" error was encountered by us. This error happened because we tried to access a property or method that is undefined or null on an object.

Before attempting to access an object's properties or methods, we had to make sure that all essential variables and objects are correctly specified. Before attempting to change any items using JavaScript, you had to be sure that they have been correctly loaded into the DOM.

If a website loads slowly or is sluggish, users may lose patience and leave. There are a variety of potential causes for this, including huge picture files, inefficient JavaScript code, or a server that is overworked. So we had to optimize the JavaScript code.

Some people can find it challenging to access the website if it does not function correctly across all browsers and gadgets. This could be brought on by the usage of exclusive browser capabilities or the reliance on antiquated software that is no longer supported by modern browsers.

A cluttered and complex website can be confusing and overwhelming for users. So we had to keep the design simple and intuitive. We used clear navigation and page layout to make sure users can easily find what they are looking for. We included essential functionalities such as course registration, grade tracking, financial aid information, and student services. We had to Ensure that the portal is secure and user data is protected. So we used encryption to prevent unauthorized access.

We tried to give the website an intuitive navigation structure, making it easy for users to find what they're looking for. We gave Clear navigation labels and a logical hierarchy for the users to use the website comfortably.

We had to combine numerous AJAX calls into one in order to reduce the number of HTTP requests required to load the website. We also had to use caching techniques. we have used Asynchronous requests to allow the page to continue loading while waiting for data from the server. This can significantly improve page load times. We have used CDN . It can help reduce latency by serving static content from servers closer to the user.

We tired to implement Lazy loading. It allows you to only load the necessary data as the user interacts with the page, reducing the initial load time. We will Use tools like Google Analytics or New Relic to monitor performance and identify areas where optimization is needed.

We had to make sure your database queries are optimized for performance. Use indexing, caching, and other techniques to improve query times.