Table of Contents

Table of Figures

1.0 Introduction

The multiple responsibilities faced by students in universities create overwhelming situations because I along with other classmates deal with assignments, exam preparation, club obligations, and academic discussions. A majority of students, including me face difficulties in staying organized because they do not have access to a designated system for centralizing personalized tasks.

The common issue among university students motivated me to develop a Student Planner Web Application dedicated to their needs. The main objective was designing an automated system to enhance student academic along with extracurricular activities management. Students benefit from a single platform implementing this planner which enables them to keep track and schedule and order their tasks and monitor their progress while reducing stress and boosting productivity.

Users can input their tasks into the application that contains features to record tasks using title and description information with due date and priority level assignment and specific category selection. The program enables task monitoring through status management features and task sorting and filtering functions together with an archive system for completed tasks. The system includes real-time notification functions for upcoming tasks which appear during user login as well as inside a designated notifications panel.

The implementation combines PHP along with MySQL for administrative parts accompanied by HTML together with CSS and JavaScript for interface development. The application uses AJAX technology to boost responsiveness together with lowered page reloading which enhances overall user experience. The modern appearance of the application emerges from its combination of dark themes and light card components which provide both visual appeal and user-friendly interface during nighttime work.

Security functions as well as user experience stood as important factors during development. Secure user authentication happens through hashed passwords in this system and real-time AJAX.pth-based validation during registration also exists along with session-based access control protecting user data. Students can modify their profile while resetting forgotten passwords without email dependency and handle their planner tasks effortlessly.

The developed web application serves my personal student needs yet targets users who wish to enhance their academic performance. The web application provides practical assistance for university students to maintain their academic progress while keeping themselves organized.

2.0 Brief Description and Evaluation of the Implementation

The Student Planner Web Application merges PHP backend development with MySQL data storage and frontend implementation through HTML CSS JavaScript and AJAX to achieve its functionalities. The system architecture contains modular core features which create clear boundaries between components and establishes one centralized database connection through student_planner_db.php. A light-card-on-dark-background design theme spans through the entire application across the 12 pages to provide excellent visibility while creating a contemporary user interface that suits academic settings.

2.1 Task Recording (add_task.php)

Explanation:
Users access a structured form to submit new tasks which incorporates critical fields for title, description, due date, category, priority and status. The application divides its content into academic work like assignments and tests together with personal tasks. Additionally, the system performs client-side and server-side validation to guarantee all entered data fulfills complete requirements. Users apply the due date to activate notifications appropriately. After submission the data inserts securely into the tasks table which exists within the student_planner database.

How it works:
Both backend PHP components and frontend HTML form with modal functionality exist within the Add Task page. The application checks for user authentication before moving onto task status updates which set overdue entries to "Pending". The form transmits sanitized data about the task title, description, and due date while priority, status, category to the database through prepared statements. The system demonstrates task information through a preview modal in case the operation is successful. Next the HTML shows a form with style before JavaScript directs users to the dashboard through a modal window post-submission. The code implements a system for secure database management along with session control features that provide user-friendly feedback.

2.2 Task Monitoring (dashboard.php, view_tasks.php)

Explanation:
Users can access the Task Manager content from dashboard.php since the page dynamically retrieves this information for display purposes. Black cards present the task menu in the task manager which provides a contemporary neat design. Users can find all active tasks presented in readable format through the view_tasks.php page table display. Students can focus on their important tasks through task sorting options that include categories and due dates and priorities which are available for filtering. End-users need to pass session checks which control authorized access to authenticated users on both pages.

How it works:
The program begins with starting a session followed by adding the database connection. The code ensures that an active session exists by examining user_id; if no active session exists it displays the login page. The request gets the sort of variable from the URL as $_GET['sort'] while a switch Correlates this value with SQL order instructions that can implement due date sorting alongside high-to-low and low-to-high priority ordering using FIELD. Through a prepared SQL statement, the application retrieves all tasks assigned to the authenticated student according to their specified sort criteria. Through HTML the system generates a table with the listed tasks which shows title, due date and sets status to Pending for incomplete deadlines and includes priority and category tags using colored classes. The program displays notifications when no tasks exist. Users make sorting choices through the form while dashboard navigation is provided by the link.

2.3 Task Status Management (task_status.php)

Explanation:
The system provides students with a view of their tasks which can be identified as Pending, On-going or Completed. Users have the capability to indicate task completion after finishing

their work. The system presents students with a confirmation pop-up after which they can finalize their task status changes. Task statuses are kept within the database system and appear the same on each webpage which lets users accurately monitor their ongoing development.

How it works:
The program initiates a check for user login status before it updates unfinished overdue tasks to "Pending" status. A POST request to complete a task will trigger the task to become archived and automatically vanish from the active list. Next the script obtains all existing tasks accessible to the user. Each task receives its own "Mark as Complete" button within a styled table which displays the tasks by category in HTML. When you click the button, the system displays a confirmation modal through basic JavaScript.

2.4 Task Archiving (archived_task.php)

Explanation:
Tasks come to completion do not receive deletion whereas they transition into the archive view keeping everything non-destroyed. The archive page shows the entire collection of tasks that both meet the Completed status and the Deleted status. All tasks which become archived still remain available for later review or auditing requirements to help retrieve previous activity data.

How it works:
Session begins before the script verifies user log-in status through the session variable user_id. The system goes to the login page when login status is denied. The program connects to the database after which it executes a prepared statement to obtain archived tasks linked to the current user's ID. The results exist in the form of an associative array. The HTML area first verifies if the database contains tasks and if this condition is true the query results display in a styled table with category tags or else it presents the "No archived tasks found" notification. The page contains a footer in addition to the included dashboard back link.

2. 5. Task Prioritization

Explanation:
Any new task or existing task update allows users to assign a High, Medium or Low priority tag through the platform. The priority level shows itself through badges that appear on the archived, view and task status pages. Priority works as an attribute that enables filterable and sortable functionality to help students easily detect their urgent tasks.

How it works:
The program bases its assessment of importance on the priority field that resides inside each task object which establishes three distinct ranking levels: High, Medium or Low. The application retrieves the sorting option from the URL query string as either due_date, priority_high or priority_low before it displays the tasks. From this input the SQL query modifies its ORDER BY clause with the help of FIELD() to perform priority-level specific sort operations (High to Low or Low to High). The filter system helps students see their tasks by their urgency levels which simplifies task selection for first priority assignments.

2. 6. Reminder System (notifications.php)

Explanation:

The system checks tasks that possess reminder dates matching or exceeding the present date while staying marked as pending or ongoing following successful login. Two notification displays connect to the login process: an immediate popup modal and an accessible notification panel inside the dashboard through a designated button. Students receive deadline alerts through two methods to maintain consistent awareness of tasks that need completion.

How it works:

The program code divides its functionality between PHP backend processing along with HTML frontend representation. In the highest part of the code PHP controls session functionality and database connectivity as well as user verification and active task data retrieval through prepared SQL commands. After data retrieval from the server the program directs itself to HTML to display the frontend The HTML displays either the formatted list of tasks or an empty task notification through PHP conditional sections. The styling tags go inside the <head> element and the responsive layout appears as a centered card which features task details and a footer.

2. 7. Secure User Authentication and Management

Explanation:

The login system (login.php) provides password security through its implementation of password_hash() and password_verify() functions. The form located at register.php allows students to submit their name together with age along with student ID and email followed by program and year of study and including phone number and gender and address. Student IDs are checked for duplications by an AJAX-based method to prevent submission. The forgot password system checks student IDs to show the linked name before enabling automatic password resetting at forgot_password.php. The edit_user.php page enables logged-in students to update their data through built-in validation processes that protect the database integrity.

How it works:

The program contains separate PHP back-end operations at the beginning followed by HTML/CSS front-end elements at the bottom. The PHP expects to begin a session then open database connectivity to process login form information by verifying accounts with prepared statements and password_verify() before setting session data upon successful authentication. The program stores errors in the $error variable for displaying them through the form. A card containing the login form displays inside the center of the page using HTML design elements that include form inputs along with the submit button and access to "Forgot Password" and "Sign Up." The layout together with colors and responsive design characteristics are controlled through CSS code embedded in the <head> section.

2.8 Task Editing and Updating (edit_task.php, update_task.php)

Explanation:

All users can alter their tasks through an interface keeping the same look as the adding task form for easy navigation. Students benefit from pre-populated task data in the form which simplifies their review process and modification tasks. Through prepared statements the system protects its database from SQL injection vulnerabilities while handling all task updates securely.

How it works:
The Edit Task application passes information to Update Task through a two-part update procedure. The URL contains a task ID value in edit_task.php which allows the system to fetch database records to automatically fill the existing form fields with title, description, date, time, reminder, and category information. Users send form data using POST to update_task.php for database record update through SQL UPDATE queries after the script validates inputs. User redirection happens to the dashboard upon successful update but failure results in displaying an error message. The program uses this partition method to maintain separate areas for displaying forms and managing database interactions.

2.9 Session Control and Logout (logout.php)

Explanation:
During session handling the system uses secure session-start and proper session termination through session_start() function calls on all protected pages and session destruction upon logout. Each protected script in the system starts by validating the session status to prevent unauthorized access to restricted content or functionalities.

How it works:
The logout script starts by using session_start() before accessing any current session data. The script utilizes session_unset() to remove stored session variables which contain information about user_id. The session.Destroy() function terminates all session operations so the session becomes inactive to future requests. The script finishes execution after it sends dashboard.php to the user through header("Location: dashboard.php") then executes exit(). The program architecture provides the user with a safe/logout process that ends their session properly before redirecting.

2.10 Centralized Database Configuration (student_planner_db.php)

Explanation:
A secure connection between the software and the student_planner database is achieved through mysqli_connect() function. Due to centralized file organization the system manages connections with the same ease that allows code to be reused by any script accessing the student_planner database. When a connection error occurs the script delivers clear instructions for developers to help them solve problems while developing their application.

How it works:
The PHP script uses the mysqli object-oriented method to create a connection with a MySQL database. This setup starts with the database connection details where it defines the server name as localhost and sets username to root while password uses 1234 for the database named student_planner. Through the credentials the application creates a new mysqli connection object $conn. To determine success the script examines $conn->connect_error following which execution stops with die() when encountering an error while displaying a connection failure notice. Before executing data operations, the application checks whether a connection with the database exists through this structure.

3.0 Brief description of working web pages

3.1 Dashboard (Main page)

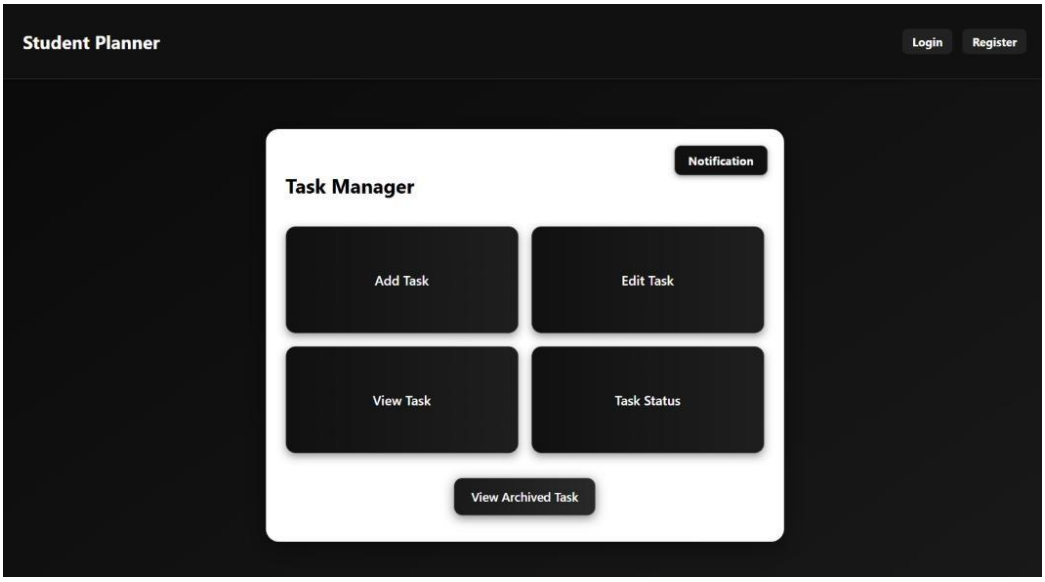


Figure 1: Dashboard Page (without Logging In)

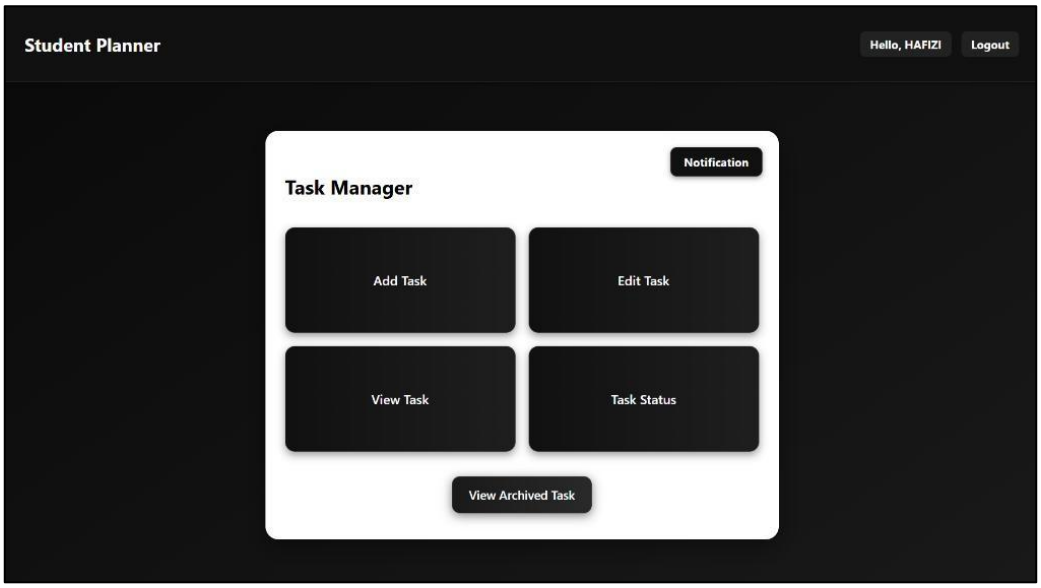


Figure 2: Dashboard Page (Logged In)

This is the dashboard page of the Student Planner. The user can view all the functions of the system from the Task Manager. Only logged in users can utilize the functions such as Add Task, Edit Task, View Task, Task Status, Archived Task, Notification, Edit User and Logout. If the user clicked the Logout button, the user be logged out and stay on the dashboard page.

3.2 Login, Register User and Forgot Password Page

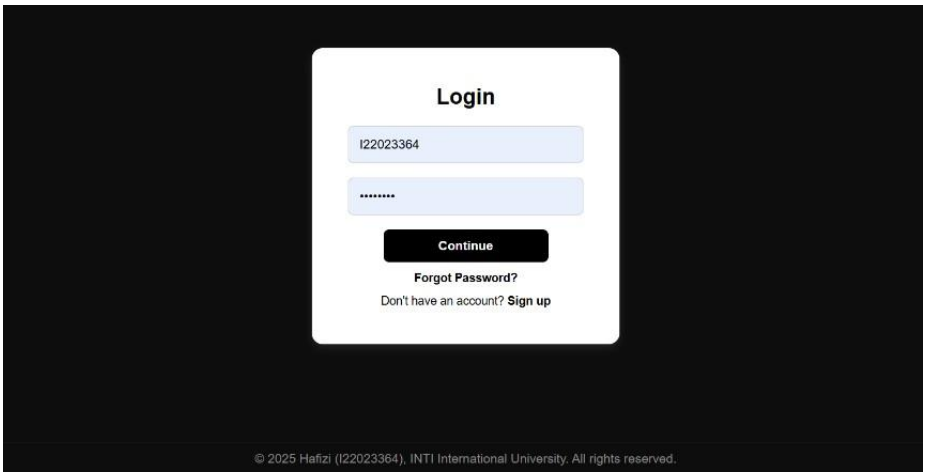


Figure 3: Login Page

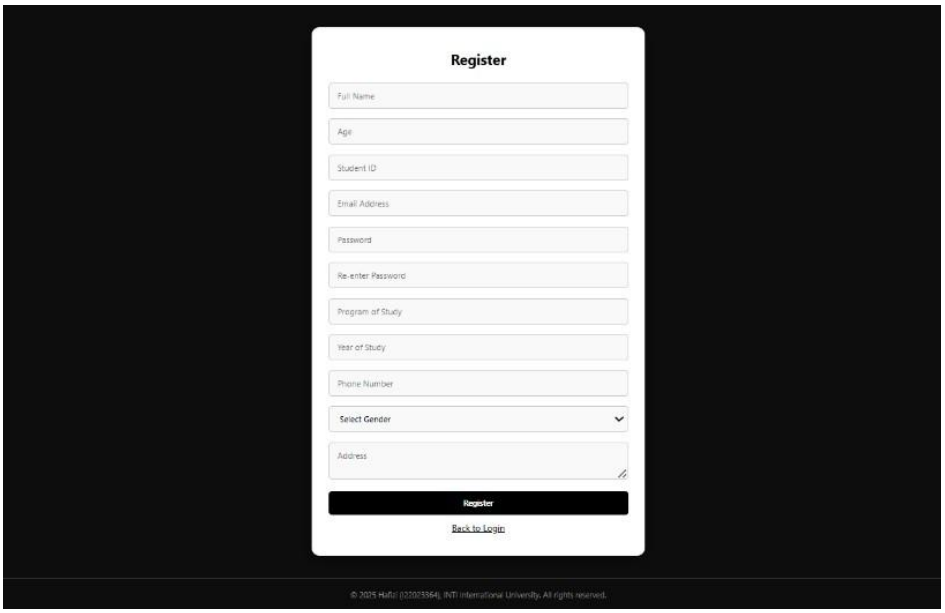
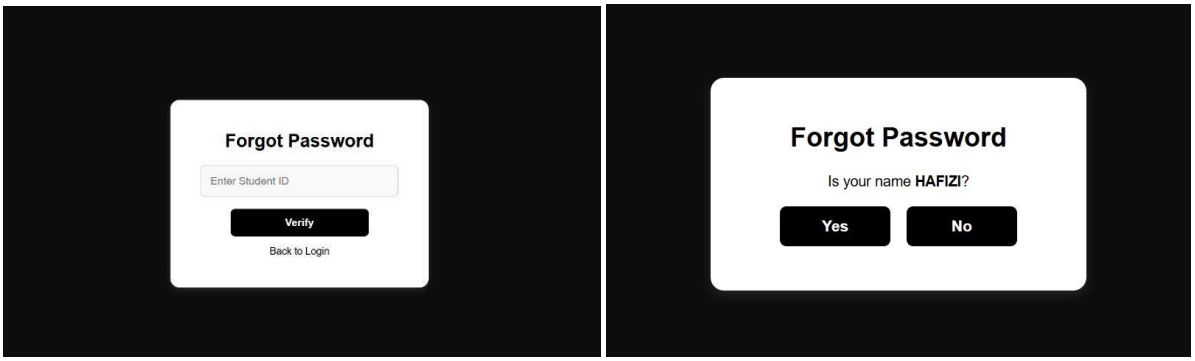


Figure 4: Register New User Page



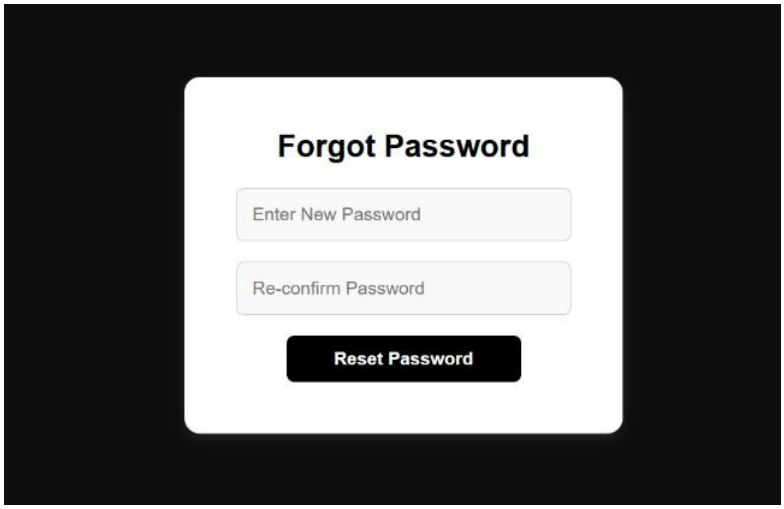


Figure 5: Forgot Password Page

The user can login to the system using is registered Student ID and Password. If it is a firsttime user, the user can register by clicking the sign-up link on the login page. Then, if the user forgets the password, the user can reset it by clicking the forgot password link on the login page and redirect it to Forgot Password Page. There, the user needs to put in the registered Student ID and the system will show a pop-up modal for the authentication. If the user clicked “Yes” he can key in his new password. If the user clicked “No” the user will stay on the same page and the user can go back to the Login page by clicking the link.

3.3 Add Task Page

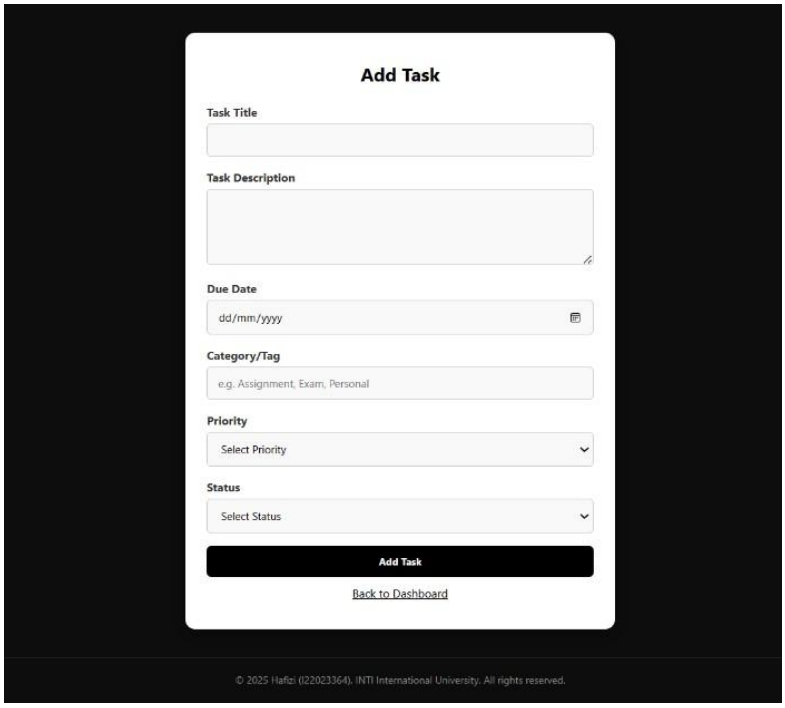


Figure 6: Add Task Page

Here the user can add tasks by filling in the form. The task will be saved in the student_planner database. The user will get a pop-up modal upon adding the task. The user can go back to the dashboard by clicking the Back to Dashboard link under the Add Task button.

3.4 Edit Task Page

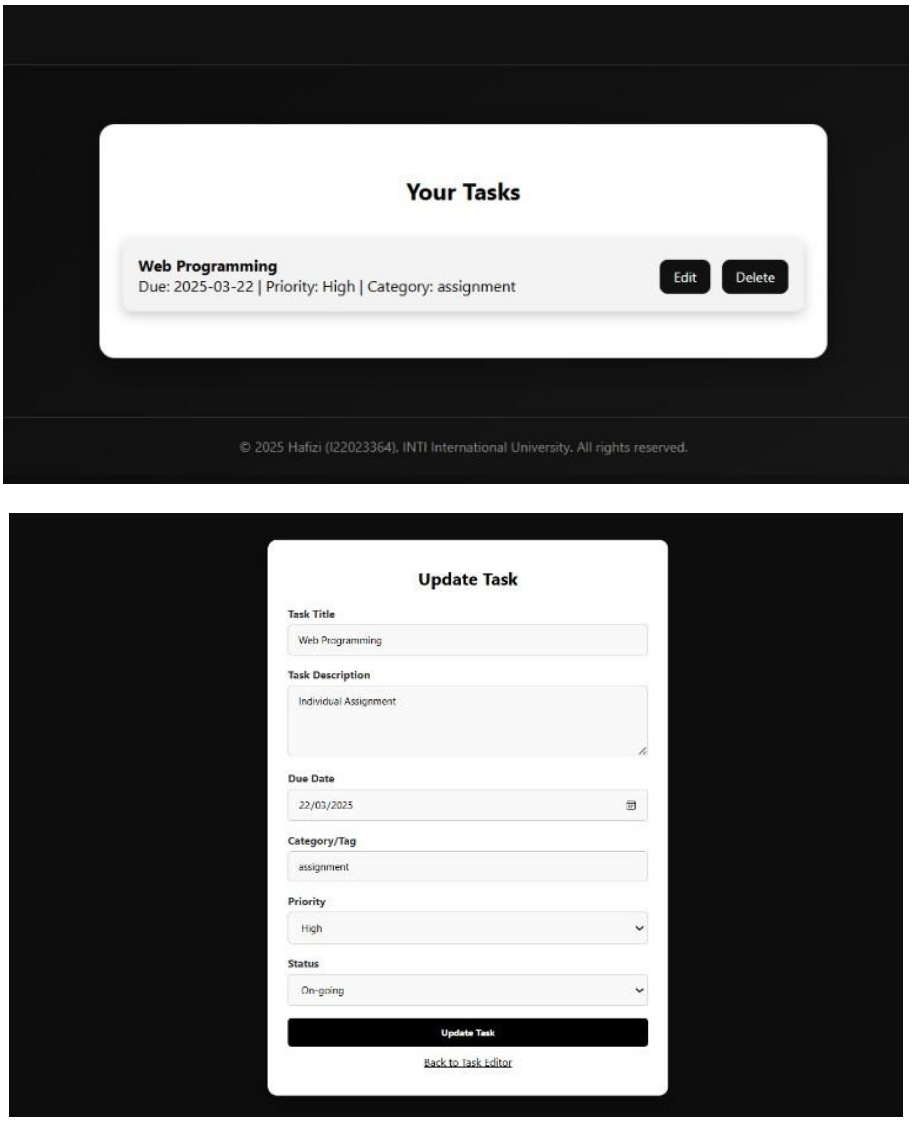


Figure 7: Edit Task Page

Here the user can choose whether the user wants to Edit or Delete the task. If the user chooses to edit, a pop-up modal will be shown and after confirming the user will be directed to the update task page. There the user can modify the task details and by clicking the Update Task button it will save it in the database. If the user chooses to delete, the user will see a confirmation pop-up modal. If the user clicked “Yes” the task will be moved to archived page. If the user clicked “No” the user will stay on the edit task page.

3.5 View Task Page

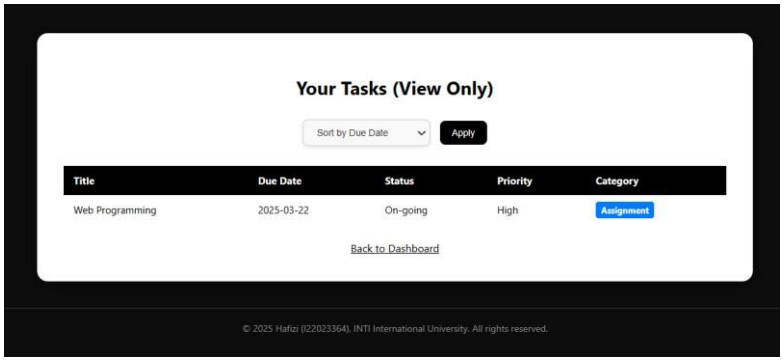


Figure 8: View Task Page

In this page, the user can only view all the On-going and Pending tasks. The user can also sort the task according to the due date and priority. The user can go back to the dashboard by clicking the Back to Dashboard link.

3.6 Task Status Page

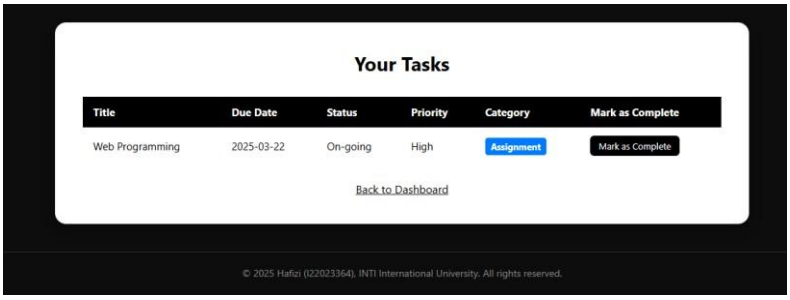


Figure 9: Task Status Page

In this page, the user can view the active task whether it is still On-going or Pending. The user can mark the task as complete by clicking the Mark as Complete button and the task will be moved to the Archived Page. The user can go back to the dashboard by clicking the Back to Dashboard link.

3.7 View Archived Task Page

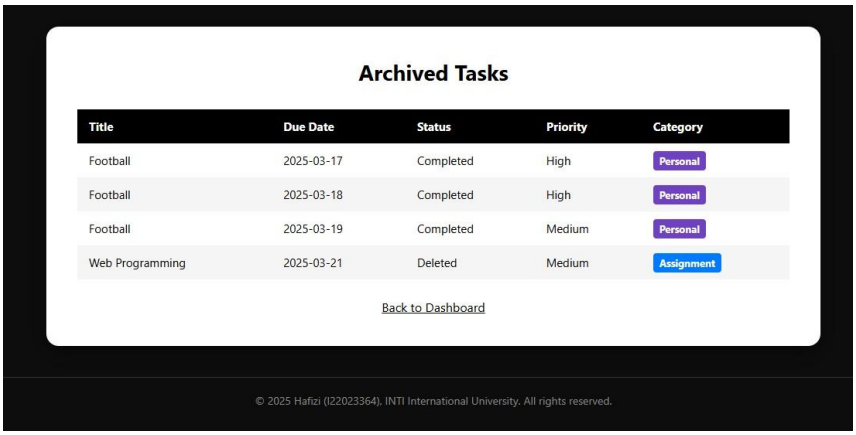


Figure 10: View Archived Task Page

In this page, the user can view all the Completed and Deleted task of the Student Planner system. The user can only view it and cannot make any changes. The user can go back to the dashboard by clicking the Back to Dashboard link.

3.8 Notifications Page

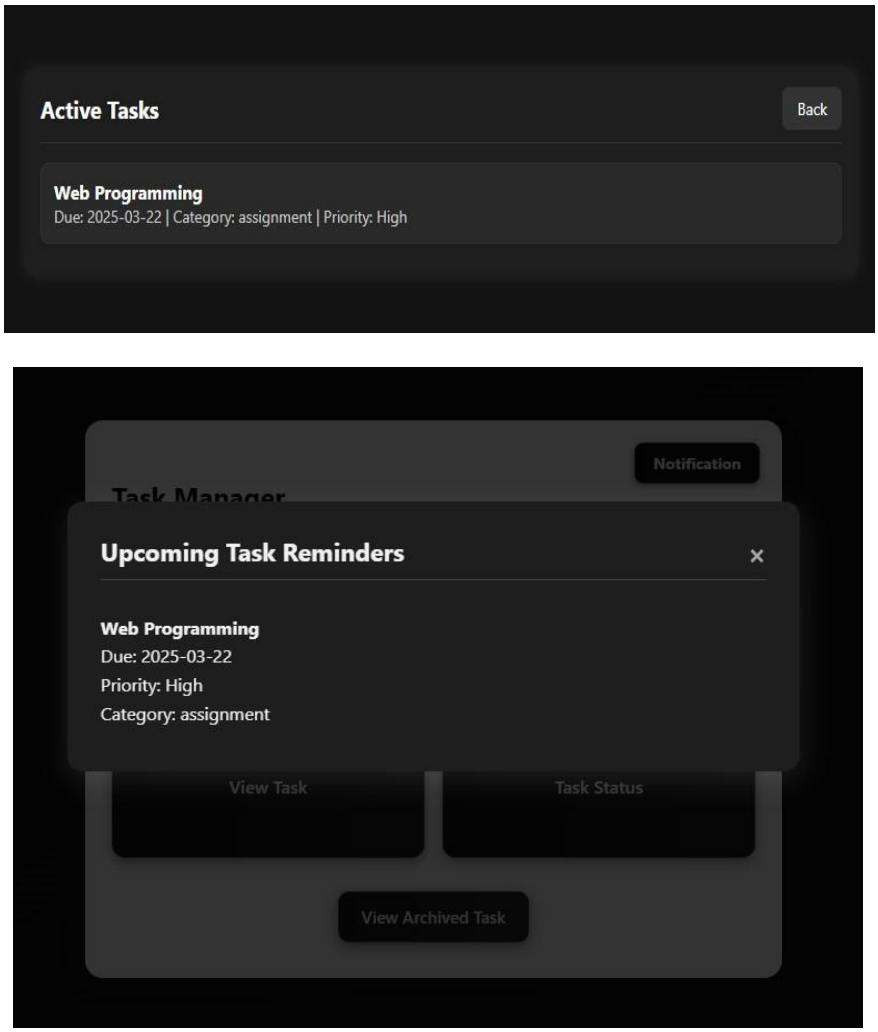


Figure 11: Notifications Page

The first image shows the ouput when the user clicked the Notifications button on the Task Manager. The user will be directed to the Notifications page. The second image shows the popup notification which shows up when the user is logged in the system. The user then can cancel the notification, but it will still be viewed on the Notifications page. Only active tasks will be seen on the Notifications page.

3.9 Edit Profile Page

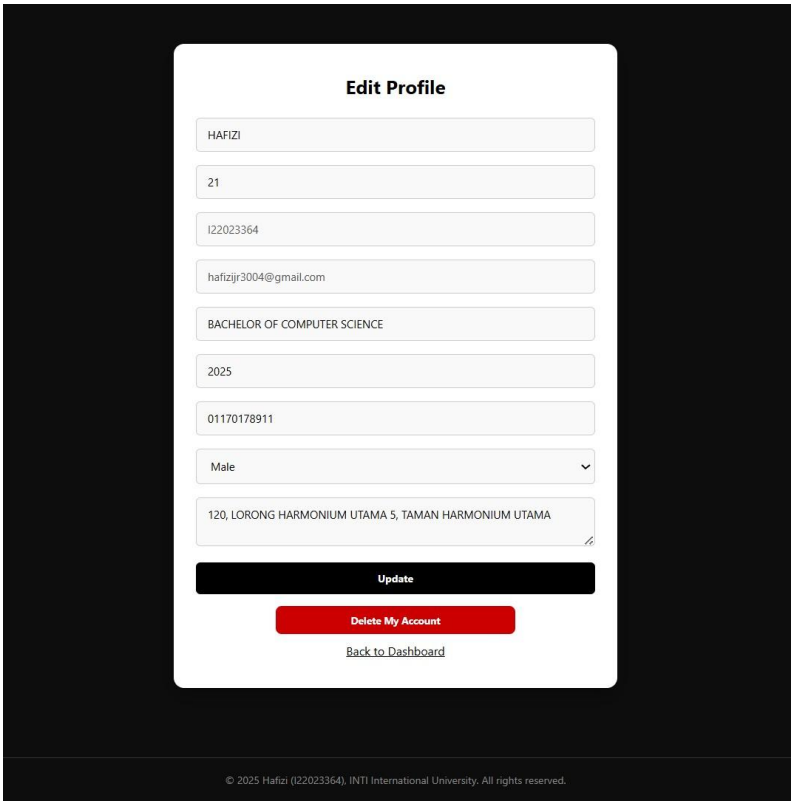


Figure 12: Edit Profile Page

The user can direct to this page by clicking the Name button located beside the logout button in the Dashboard. In this page, the user can update the user details, and it will be saved in the database. The user can also delete the account, and it will automatically delete the user from the database. The user can go back to the dashboard by clicking the Back to Dashboard link.

4.0 User Manual

4.1 System Requirements

- Before running the application, ensure the following requirements are met:
- Operating System: Windows / macOS / Linux
- Web Server: XAMPP or WAMP
- Browser: Google Chrome (recommended)
- Database: MySQL
- Code Editor: Notepad++ (optional)

4.2 Application Setup

Step 1: Start Local Server

- Open XAMPP Control Panel
- Click Start for:
  - Apache
  - MySQL
- Ensure both services are running successfully.

Step 2: Creating MySQL Database

- Download the provided MySQL database file from the GitHub repository.
- Execute the SQL file to create the required database and tables.
- ⚠ The database must be created first to prevent system errors.

Step 3: Place Project Files

- Download the provided Source Code files from the GitHub repository.
- Copy the project folder
- Paste it inside: C:\xampp\htdocs\
- Example: C:\xampp\htdocs\student-planner

Step 4: Run the Application

- Open a browser and enter: http://localhost/student-planner/dashboard.php