Department of Computer Science
The University of Bradford

Report - Team 22

Enterprise Pro Task Management System

Team Members:

Hasan Akhtar (23011124) [Project Manager]

Humayun Razaq (23019270)

Hamza Khan (23013894)

Hammad Aziz (23009626)

Umair Siddiq (23010283)

Ehtesham Shah (23010000)

Rahat Nafees (23015702)

CONTENTS:

Requirements Specification Document Introduction

The Yorkshire and Humber regional Organised Crime Unit (YHROCU) is looking for a simple Project/Task Management System aiming to handle non-crime related activities. The projects and tasks will be stored in a database and supervisors will be able to view all of these and make edits if necessary. Normal users will just see the projects they are part of and tasks that have been assigned to them. The client of the system is the YHROCU. The system will be operated by some of the YHROCUs administrators and supervisors will be able to perform most functions. Deletion of tasks, projects, users, creation and modification of users, etc. will only be allowed by Admins and there will be task/project recovery. This specification document is based on an initial scenario and an interview with the client when the key functions, the data behind the database and the user interface have been discussed.

Team Expertise

Our team, Team 22, brings robust expertise in full-stack web development, combining proficiency in Python (Flask), Java (servlets), and JavaScript to deliver scalable, responsive solutions. Our experience building feature-rich systems—such as checkout workflows, CMS platforms, and admin interfaces during the Internet Technologies module—positions us strongly for complex projects requiring meticulous backend logic and intuitive frontend design. Our first-choice project, the Workflow Management System (P1), directly leverages our proven ability to optimize internal processes, as demonstrated in our Software Design module, while collaboration with YHROCU aligns with our enthusiasm for impactful stakeholder engagement. Similarly, the Job Portal (P5) aligns with our aptitude for crafting SEO-optimized, user-centric platforms, and the Assets Interface (P2) capitalizes on our experience in map integration and CMS development. By merging technical precision with a user-focused approach, we ensure our solutions are both functionally robust and strategically aligned with client objectives.

Rationale of Topic Choice

Our team, Team 22, excels in web development and modern web design, leveraging a broad range of technologies such as Python with Flask, Java with servlets, and JavaScript. We have a proven track record in creating responsive websites with comprehensive checkout processes, content management systems, and admin features, as demonstrated in our Internet Technologies module. Additionally, our expertise in SEO optimization ensures that our web applications are not only functional but also highly visible and user-friendly.

After carefully reviewing the project specifications, we have selected our top three project options. Our first choice is the Workflow management system for non-crime related activity (P1), as it aligns with our interest in creating an activity management system and our experience in improving internal workflows, similar to our work in the Software Design and Development module. We are particularly excited about the opportunity to collaborate with the police (YHROCU) to enhance their processes. Our second choice is the Job Advertisement Portal (P5), inspired by our desire to create a standout, high-quality job board. Finally, our third choice is the Web Interface to Interact with "Assets" (P2), where we look forward to working with Bradford Council to develop a robust map assets system. We are confident that our team's dedication and strategic approach will lead to the successful completion of any project we undertake.

Developed by Ehtesham Shah, Rahat Nafees

Reviewed (and slightly modified) **by** Hasan Akhtar

Initial Date: 04/02/2025; Latest Update: 17/02/2025

Software Requirements Specification

1. Introduction

1. 1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to define the functional and non-functional requirements for the **Enterprise Pro Task Management System**. The system aims to provide improved project and task management capabilities, addressing the limitations of the existing Microsoft services used by the client. The system will allow users to create, assign, track, and manage tasks efficiently with different user roles and permissions.

1.2 Product Scope

The Enterprise Pro Task Management System is designed to facilitate seamless project and task tracking within an organization. The system will provide functionalities to:

- Create, assign, and manage tasks under specific projects.
- Search and filter tasks based on assigned dates, status, and projects.
- Define roles and permissions for different levels of users.
- Improve oversight and security of tasks and projects.
- Ensure updates and tracking of tasks at regular intervals.

By implementing this system, the organization will benefit from:

- Enhanced project management and task organization.
- Role-based access control for better security.
- The ability to track and manage tasks effectively.

2. Overall Description

2.1 Product Functions

The major functionalities of the system will include:

1. **Task Management**:
    - Create tasks with a title and details.
    - Assign tasks to users and set due dates.
    - Define task status: *New, In-Progress, Completed, Overdue*.
    - Search and filter tasks based on criteria such as assigned dates and project association.
2. **Project Management**:
    - Group tasks under projects.
    - Provide an overview of all projects.
    - Allow filtering of tasks by project and status.
3. **User Roles and Permissions**:
    - **User**: Can view assigned tasks, update task progress.
    - **Supervisor**: Read/edit access to amend tasks, add/remove users to/from projects.
    - **Admin**: Full access to create, modify, delete users, tasks, and projects.
4. **Security and Authentication**:
    - Internal login system with username and password.
    - Admin-controlled user registration.
    - Role-based access restrictions.
5. **Task Updates & Reporting**:
    - Users must provide updates on tasks instead of percentage-based progress tracking.
    - Task updates to be logged at least once a week.
    - Monthly task and project reports can be exported.

2.2 Constraints

- The system will be **internal only**; no external access is required.
- Minimal security setup, requiring only username and password authentication.
- No advanced security features like encryption or multi-factor authentication.

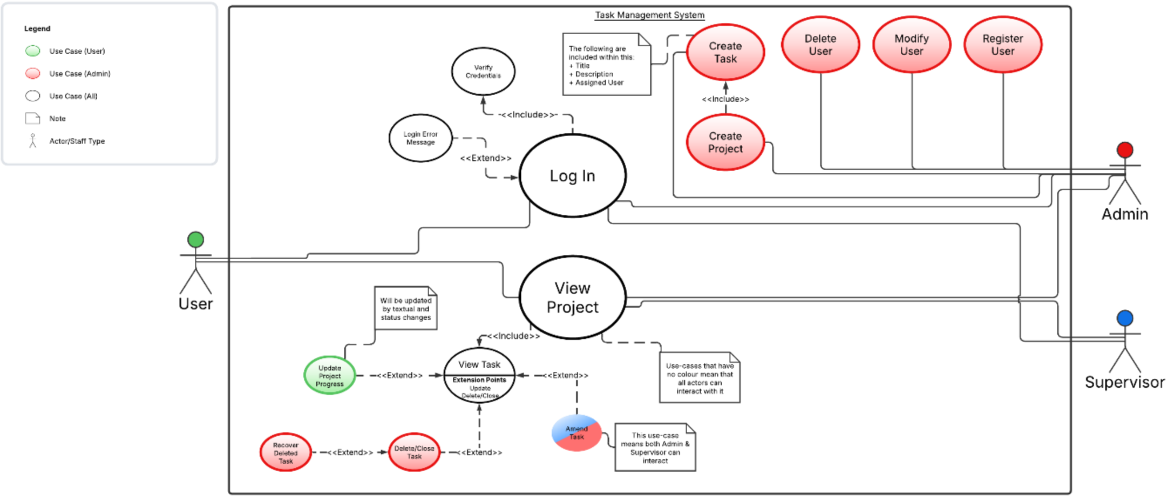
2.3 Assumptions and Dependencies

- The system assumes that all users have internal access credentials provided by an admin.
- Task updates will be made manually by users.
- Task deletions should have an undo or recovery option to prevent permanent loss.

3. Functional Requirements

ID	**Requirement Description**
FR1	Admins and Supervisors should be able to create tasks with a title, details, and assign users.
FR2	Tasks must have a status: *New, In-Progress, Completed, Overdue*.
FR3	Tasks should have assigned and due dates for sorting and filtering.
FR4	Users can update project progress through status changes.
FR5	Supervisors should have read/edit access to all tasks, and add/remove users to/from projects.
FR6	Admins should have full control to manage users, tasks, and projects.
FR7	The system must allow filtering of tasks based on project, date, and status.
FR8	Project updates should be required at least once a week.
FR9	Ability to have multiple users per task
FR10	Must have ability to see all task updates associated to a project in date order descending.
FR11	The system should allow exporting reports on tasks and projects monthly.
FR12	Only admins can register users, ensuring controlled access.

4. UML Use Case Diagram



5. Non-Functional Requirements

ID	Requirement Description
NFR1	The system should be available 99% of the time during business hours.
NFR2	The user interface should be intuitive and easy to navigate.
NFR3	The system should support at least 50 concurrent users.
NFR4	Response time for task queries should be less than 2 seconds.
NFR5	The system should provide a simple authentication mechanism (username/password).

6. Risks, Ethical, and Security Considerations

- **Risk of data loss**: A task recovery mechanism should be included.
- **Ethical Considerations**: Users should only have access to tasks relevant to them.
- **Security**: The system should ensure internal data privacy, avoiding unauthorized changes.

7. Conclusion

The Enterprise Pro Task Management System will address current project management inefficiencies by implementing a structured task management process with appropriate user roles, permissions, and tracking mechanisms. The focus will be on ease of use and task management efficiency rather than advanced security measures.

Approvals Section

Client Signature:

Name: __Lee Baragwanath__ Title: _________DI_________

Signature: _ Date: _____17/02/25______

Team Leader Signature:

Name: _____Hasan Akhtar_____ Title: ____Student____

Signature: ______ H. Akhtar ______ Date: ___18/02/2025___

Developed by Hammad Aziz**,** Ehtesham Shah, Rahat Nafees

Reviewed (and slightly modified) **by** Hasan Akhtar

Initial Date: 11/02/2025; Latest Update: 06/03/2025

Software Design Document

1. Introduction

1.1 Purpose

The purpose of this software design document is to fully describe the architecture of the Enterprise Pro Task Management system. This web-based system is designed for the Yorkshire and Humber Regional Organised Crime Unit (YHROCU) support department to manage non-crime related tasks. The system will enable the assignment, tracking, and reporting of both individual and collaborative tasks, ensuring timely notifications and robust audit trails. This document serves as a blueprint for developers, stakeholders, and future maintainers.

1.2 Overview

This document outlines the following key areas:

Architectural Design: The overall structure of the system using a 3-tier architecture.

Software Design Requirements: A detailed list of functional and non-functional requirements, including task management, notifications, and user access controls.

Data Description: A description of the database tables and configurations used for managing users, tasks, projects, assigned tasks, task updates and users in projects.

System Architecture: An in-depth look at the system layers and component decomposition, supported by diagrams.

Conclusion: A summary of the design and its alignment with the departmental objectives.

2. Software Design Requirements

The Workflow Management System must meet the following requirements:

Task Assignment and Notification:

Enable tasks to be assigned to one or more staff members.

Automatically email the assigned staff member(s) when a task is allocated.

Task Update and Logging:

Allow users to update task status, and input progress updates to a rolling log.

Prevent staff from deleting a task or any previous update entries to preserve the historical record.

Access Control and Supervisory Functions:

Provide supervisory access to view all tasks, with options to restrict visibility to only the assigned persons or to all users. Allow a supervisor to close or delete a task as needed.

Dashboard and Reporting:

Offer a dashboard that categorizes and summarizes tasks, featuring filters such as status and due date. Include an export function to generate reports in CSV or PDF formats.

System Flexibility and Search:

Design with built-in flexibility to allow the addition of new data fields to tasks to support future changes. Implement a robust search function for efficient task retrieval.

User Authentication:

Implement user authentication using OpenAuth or a similar protocol to ensure compatibility with existing infrastructure and secure system access. This is an optional requirement.

3. Data Description

The system will interact with the following database tables to manage tasks and staff information:

Users table:

This table contains records of all staff members, including usernames, passwords, roles and team. It supports user authentication, retrieval and modification by admins.

Tasks table:

Stores task details such as task ID, title, details, status, due dates, and assigned dates. Each line represents a unique task.

Task Updates table:

Maintains a log of all updates made to tasks, including progress updates and status changes, ensuring a complete audit trail.

Project Users table:

This table contains records of all staff members and which projects they are part of.

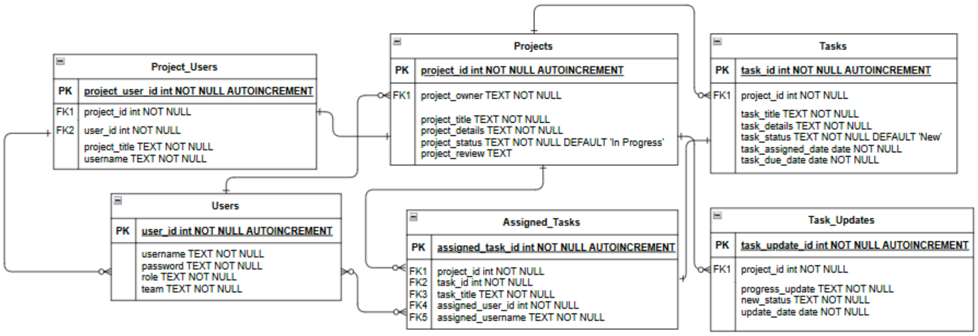
Projects table:

Stores Project details such as Project ID, title, details, status, owners and review dates. Each line represents a unique project.

Assigned tasks table:

Maintains records of all tasks and the staff member(s) that they have been assigned to.

Figure 1: Database ER Diagram


Project_Users
PK project_user_id int NOT NULL AUTOINCREMENT
FK1 project_id int NOT NULL
FK2 user_id int NOT NULL
project_title TEXT NOT NULL
username TEXT NOT NULL
Users
PK user_id int NOT NULL AUTOINCREMENT
username TEXT NOT NULL
password TEXT NOT NULL
role TEXT NOT NULL
team TEXT NOT NULL
Projects
PK project_id int NOT NULL AUTOINCREMENT
FK1 project_owner TEXT NOT NULL
project_title TEXT NOT NULL
project_details TEXT NOT NULL
project_status TEXT NOT NULL DEFAULT 'In Progress'
project_review TEXT
Assigned_Tasks
PK assigned_task_id int NOT NULL AUTOINCREMENT
FK1 project_id int NOT NULL
FK2 task_id int NOT NULL
FK3 task_title TEXT NOT NULL
FK4 assigned_user_id int NOT NULL
FK5 assigned_username TEXT NOT NULL
Tasks
PK task_id int NOT NULL AUTOINCREMENT
FK1 project_id int NOT NULL
task_title TEXT NOT NULL
task_details TEXT NOT NULL
task_status TEXT NOT NULL DEFAULT 'New'
task_assigned_date date NOT NULL
task_due_date date NOT NULL
Task_Updates
PK task_update_id int NOT NULL AUTOINCREMENT
FK1 project_id int NOT NULL
progress_update TEXT NOT NULL
new_status TEXT NOT NULL
update_date date NOT NULL

4. System Architecture

4.1 Architectural Design

The Workflow Management System is built using a 3-tier architecture:

Client Layer (Blue):

Provides a web-based interface where users, supervisors and administrators interact with the system. This layer is responsible for displaying dashboards, task forms, project overviews, progress logs, search functions and more.

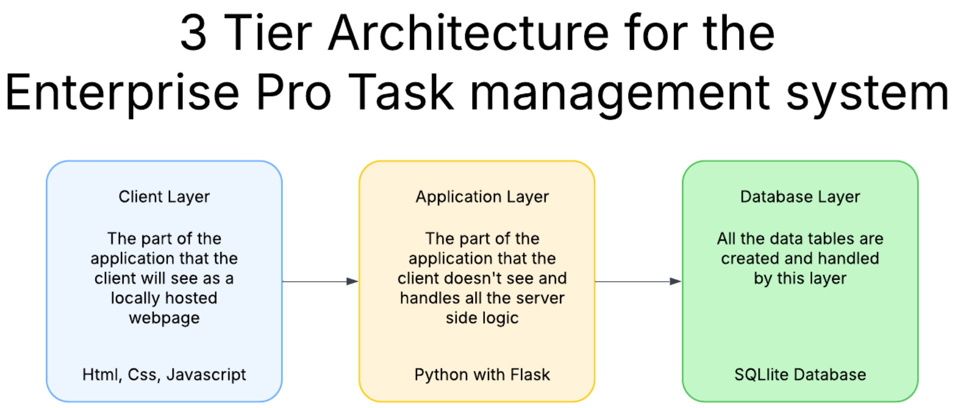
Application Layer (Green):

Contains the business logic for task management, including assignment, update processing, notifications, and access control. It acts as the intermediary between the client interface and the database.

Database Layer (Red):

Manages persistent storage of all data, including user details, task records, and update logs. It ensures data integrity, security, and efficient retrieval.

Figure 2: 3-Tier Architecture System



3 Tier Architecture for the
Enterprise Pro Task management system

Client Layer

The part of the application that the client will see as a locally hosted webpage

Html, Css, Javascript

Application Layer

The part of the application that the client doesn't see and handles all the server side logic

Python with Flask

Database Layer

All the data tables are created and handled by this layer

SQLlite Database

(Illustration: Client Interface → Application Logic → Database Storage)

4.2 Decomposition Description

The system is decomposed into several key modules:

User Interface Manager:

Manages the web interface, including dashboards, task forms, and search functionality. It handles real-time user interactions and displays task data.

Notification Module:

Automates email notifications to staff upon task assignment, ensuring timely alerts.

Task Manager:

Oversees task creation, assignment, status updates, and log management. It enforces rules to prevent unauthorized deletion of tasks or updates.

Access Control and Authentication Module:

Implements user authentication (using OpenAuth or similar) and enforces role-based access controls, differentiating between staff and supervisory functions.

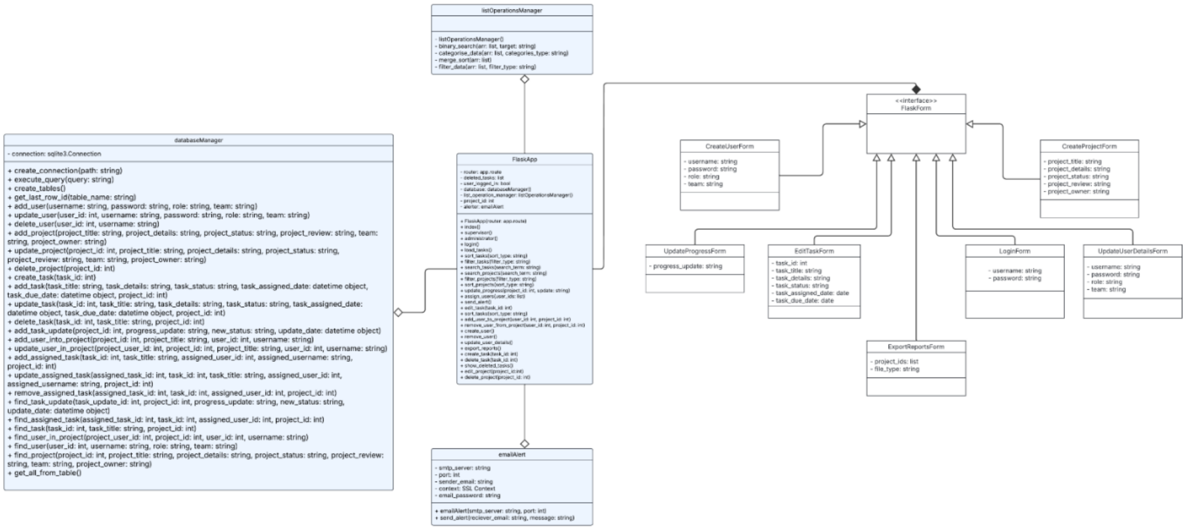
Reporting and Export Module:

Provides analytics through a dashboard and supports data export in CSV or PDF formats.

Database Manager:

Interfaces for operations with the underlying database tables (Users, Projects, Tasks, TaskUpdates, AssignedTasks, ProjectUsers) to perform secure read/write operations while ensuring consistency.

Figure 3: Class Diagram:


listOperationsManager
- listOperationsManager()
- binary_search(arr: list, target: string)
- categorise_data(arr: list, categories_type: string)
- merge_sort(arr: list)
- filter_data(arr: list, filter_type: string)
databaseManager
- connection: sqlite3.Connection
+ create_connection(path: string)
+ execute_query(query: string)
+ create_tables()
+ get_last_row_id(table_name: string)
+ add_user(username: string, password: string, role: string, team: string)
+ update_user(user_id: int, username: string, password: string, role: string, team: string)
+ delete_user(user_id: int, username: string)
+ add_project(project_title: string, project_details: string, project_status: string, project_review: string, team: string, project_owner: string)
+ update_project(project_id: int, project_title: string, project_details: string, project_status: string, project_review: string, team: string, project_owner: string)
+ delete_project(project_id: int)
+ create_task(task_id: int)
+ add_task(task_title: string, task_details: string, task_status: string, task_assigned_date: datetime object, task_due_date: datetime object, project_id: int)
+ update_task(task_id: int, task_title: string, task_details: string, task_status: string, task_assigned_date: datetime object, task_due_date: datetime object, project_id: int)
+ delete_task(task_id: int, task_title: string, project_id: int)
+ add_task_update(project_id: int, progress_update: string, new_status: string, update_date: datetime object)
+ add_user_into_project(project_id: int, project_title: string, user_id: int, username: string)
+ update_user_in_project(project_user_id: int, project_id: int, project_title: string, user_id: int, username: string)
+ add_assigned_task(task_id: int, task_title: string, assigned_user_id: int, assigned_username: string, project_id: int)
+ update_assigned_task(assigned_task_id: int, task_id: int, task_title: string, assigned_user_id: int, assigned_username: string, project_id: int)
+ remove_assigned_task(assigned_task_id: int, task_id: int, assigned_user_id: int, project_id: int)
+ find_task_update(task_update_id: int, project_id: int, progress_update: string, new_status: string, update_date: datetime object)
+ find_assigned_task(assigned_task_id: int, task_id: int, assigned_user_id: int, project_id: int)
+ find_task(task_id: int, task_title: string, project_id: int)
+ find_user_in_project(project_user_id: int, project_id: int, user_id: int, username: string)
+ find_user(user_id: int, username: string, role: string, team: string)
+ find_project(project_id: int, project_title: string, project_details: string, project_status: string, project_review: string, team: string, project_owner: string)
+ get_all_from_table()
FlaskApp
- router: app.route
- deleted_tasks: list
- user_logged_in: bool
- database: databaseManager()
- list_operation_manager: listOperationsManager()
- project_id: int
- alerter: emailAlert
+ FlaskApp(router: app.route)
+ index()
+ supervisor()
+ administrator()
+ login()
+ load_tasks()
+ sort_tasks(sort_type: string)
+ filter_tasks(filter_type: string)
+ search_tasks(search_term: string)
+ search_projects(search_term: string)
+ filter_projects(filter_type: string)
+ sort_projects(sort_type: string)
+ update_progress(project_id: int, update: string)
+ assign_users(user_ids: list)
+ send_alert()
+ edit_task(task_id: int)
+ sort_tasks(sort_type: string)
+ add_user_to_project(user_id: int, project_id: int)
+ remove_user_from_project(user_id: int, project_id: int)
+ create_user()
+ remove_user()
+ update_user_details()
+ export_reports()
+ create_task(task_id: int)
+ delete_task(task_id: int)
+ show_deleted_tasks()
+ edit_project(project_id:int)
+ delete_project(project_id: int)
emailAlert
- smtp_server: string
- port: int
- sender_email: string
- context: SSL Context
- email_password: string
+ emailAlert(smtp_server: string, port: int)
+ send_alert(reciever_email: string, message: string)
<<interface>>
FlaskForm
CreateUserForm
- username: string
- password: string
- role: string
- team: string
CreateProjectForm
- project_title: string
- project_details: string
- project_status: string
- project_review: string
- project_owner: string
UpdateProgressForm
- progress_update: string
EditTaskForm
- task_id: int
- task_title: string
- task_details: string
- task_status: string
- task_assigned_date: date
- task_due_date: date
LoginForm
- username: string
- password: string
UpdateUserDetailsForm
- username: string
- password: string
- role: string
- team: string
ExportReportsForm
- project_ids: list
- file_type: string

5. Conclusion

This Software Design Document outlines the comprehensive design of the Enterprise Pro Task Management system tailored for the YHROCU. Leveraging a robust 3-tier architecture and modular design, the system addresses key requirements such as automated notifications, secure access, detailed logging, and flexible task management. This design provides a strong foundation for improving operational efficiency and is well-prepared to accommodate future enhancements and changes. Feedback and further modifications are welcome to ensure that the final implementation meets all departmental objectives.

Work Plan

Team structure and Roles

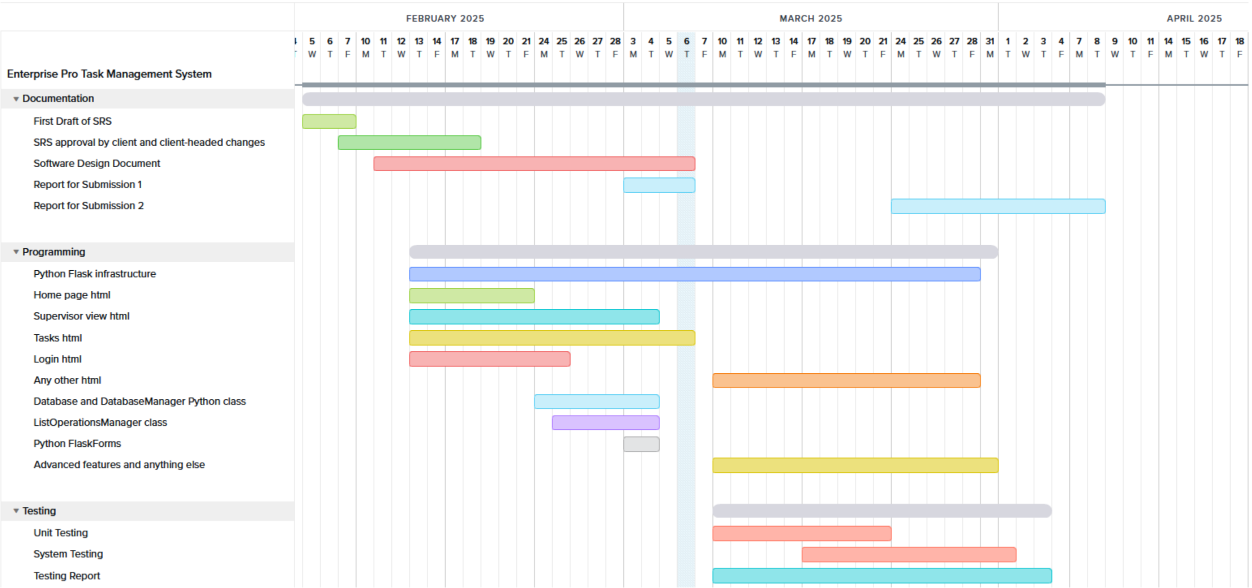
Roles:

Ehtesham: Documentation + Testing
Hammad: Documentation + Testing
Humayun (Secretary): Documentation + Programming
Hamza: Programming + Testing
Umair: Client Meeting + Programming
Rahat: Documentation + Client Meeting
Me (Lead): Programming + Testing

Structure:

Documentation: Ehtesham, Hammad, Humayun (Secretary), Rahat
Programming: Hasan, Umair, Hamza, Humayun
Client Meetings: Rahat, Umair
Testing: Ehtesham, Hammad, Hasan, Hamza

Gantt Chart



GitHub

The repository link is: https://github.com/HasanBradfordUni/Enterprise-Pro-Coursework

Prototype

The prototype has been produced and stored in the following repository folder:

https://github.com/HasanBradfordUni/Enterprise-Pro-Coursework/tree/main/Code

Meeting Minutes

The Meeting minutes have been produced and stored in the following repository folder:

https://github.com/HasanBradfordUni/Enterprise-Pro-Coursework/tree/main/Meeting%20Minutes

NDA

The non-disclosure agreement has been signed and stored at the following link:

https://github.com/HasanBradfordUni/Enterprise-Pro-Coursework/blob/main/Other%20Documents/NDA%20student-CEC.docx

Peer review

Hasan Akhtar (23011124)

Managed the full project, helped everyone with documentation and programming, made sure everyone concentrated on their tasks and that deadlines were met. Set up team meetings and administrated the entire project. Did a lot of the programming and facilitated the client meeting.

Humayun Razaq (23019270)

Was the secretary, administered all of the meeting minutes and had a programming task. Was tasked with creating a html login page and completing the application layer (Python with Flask) code that checked the login and connected to the database.

Hamza Khan (23013894)

Punctually attended all team meetings and was tasked with developing the supervisor html page and the Python methods for handling projects. Also helped with the meeting notes from the initial meeting with the client at the beginning of the project.

Hammad Aziz (23009626)

Was mainly tasked with producing the Software Design Document and Objectives/Scope documents, also contributed to programming with most of the Flask forms in forms.py.

Umair Siddiq (23010283)

Tried to lead the client meeting with Rahat, was mainly concentrating on his programming task of the application layer methods to do with the Tasks in projects in the system, also created the tasks html page.

Ehtesham Shah (23010000)

Produced most of the Software Requirements Specification document and contributed to the Software Design Document. Also was tasked with producing searching and sorting logic in Python which became the basis of the listOperationsManager class.

Rahat Nafees (23015702)

Produced diagrams for the Software Requirements Specification document and Software Design Document. Also tried to lead the client meeting with Umair. Helped with some database interaction Python methods in the databaseManager class.