

ITE 1942 – ICT PROJECT

PROJECT REPORT

Level 01

TaskMaster: Comprehensive Task Management System

Submitted by:

PRABATH P T

E2248301

Bachelor of Information Technology (External Degree)
Faculty of Information Technology
University of Moratuwa

Table of Contents

1.	INTRODUCTION.....	5
2.	RELATED WORK.....	7
3.	SYSTEMS ANALYSIS.....	8
4.	SYSTEM DESIGN	9
5.	SYSTEM IMPLEMENTATION.....	10
6.	APPENDIX.....	11
7.	REFERENCES.....	12

1. Introduction

The TaskMaster project introduces a comprehensive task management system designed to streamline organizational workflows by efficiently handling task creation, assignment, monitoring, tracking, and notification functionalities. At its core, TaskMaster aims to centralize and optimize task management processes, enabling stakeholders to enhance productivity, collaboration, and accountability within their teams. The main idea behind TaskMaster is to provide organizations with a user-friendly platform that simplifies the management of tasks across projects and teams. By offering intuitive interfaces for task creation, assignment, and monitoring, TaskMaster empowers users of all technical proficiencies to effectively manage their workload. Real-time updates and notifications ensure that stakeholders stay informed about task progress, impending deadlines, and any changes or developments, facilitating timely decision-making and action. Stakeholders across various organizational levels benefit from TaskMaster's features and functionalities. Team members experience increased efficiency in managing their tasks, with clear visibility into priorities and deadlines. Managers gain insights into task progress and resource allocation, enabling them to make informed decisions and adjustments to ensure project success. Additionally, stakeholders benefit from improved collaboration and communication, as TaskMaster provides a centralized platform for sharing information, discussing tasks, and coordinating efforts. Ultimately, TaskMaster empowers organizations to optimize their task management processes, leading to enhanced productivity, streamlined workflows, and improved outcomes. By facilitating efficient task management, TaskMaster enables stakeholders to focus their time and efforts on driving innovation, achieving goals, and delivering value to their customers and stakeholders.

TaskMaster is essential as it streamlines task management processes, ensuring efficiency and productivity within organizations. In traditional paper-based or manual systems, tasks can easily get lost, deadlines missed, and communication breakdowns occur. A computerized system like TaskMaster centralizes task creation, assignment, monitoring, and tracking, providing real-time updates and notifications. This ensures that all team members are aligned on project goals, deadlines are met, and progress is transparent. Additionally, TaskMaster enables better resource allocation and time management, as managers can easily assess workload distribution and prioritize tasks accordingly. Furthermore, the system facilitates collaboration by allowing seamless communication between team members, leading to enhanced coordination and teamwork. Ultimately, TaskMaster enhances organizational effectiveness by optimizing task management processes and fostering a culture of accountability and productivity.

1.1 Background & Motivation

In the contemporary workplace, managing tasks effectively is crucial for maintaining productivity and ensuring that projects are completed on time. However, many organizations struggle with task management due to the lack of an efficient system that can handle task creation, assignment, monitoring, and completion seamlessly. Traditional methods, such as manual tracking with spreadsheets or simple to-do lists, often lead to confusion, missed deadlines, and an overall decrease in productivity. The need for a robust and comprehensive

task management system like TaskMaster becomes evident when considering the complexities and dynamics of modern work environments, where collaboration and communication are key to success.

The lack of an efficient task management system has a significant impact on various stakeholders within an organization. For project managers, it becomes increasingly challenging to monitor the progress of multiple tasks, leading to difficulties in resource allocation and timeline management. Team members, on the other hand, often face issues with task visibility and prioritization, resulting in duplicated efforts or neglected tasks. Ultimately, this inefficiency can lead to missed deadlines, reduced quality of work, and decreased morale among employees. Clients and end-users also suffer from the fallout, as delays and miscommunications can impact the delivery and quality of services or products. By addressing these problems with an effective solution like TaskMaster, organizations can enhance productivity, improve collaboration, and ensure timely project completion.

There are several task management tools available in the market today, such as Microsoft Todo, Asana, and Trello, each offering a range of features designed to streamline task management. Microsoft Todo, for instance, provides basic task tracking and reminders but lacks advanced collaboration features and integration capabilities necessary for complex project management. Asana and Trello offer more comprehensive solutions with features like project timelines and team communication channels. However, they often require extensive customization and can become overwhelming for users due to their complexity. TaskMaster aims to bridge the gap by offering an intuitive, user-friendly interface combined with powerful features tailored to meet the specific needs of diverse teams and projects. By learning from the limitations of existing solutions, TaskMaster seeks to provide a balanced and efficient approach to task management, ensuring that all stakeholders benefit from its implementation.

1.2 Problem in brief

In today's fast-paced business environment, effective task management is essential to maintain productivity and ensure successful project completion. However, many organizations struggle with the inefficiencies and limitations of traditional task management methods. These methods often involve manual tracking with spreadsheets, emails, or simple to-do lists, which are prone to errors, miscommunication, and lack of transparency. Project managers face significant challenges in overseeing multiple tasks, monitoring progress, and ensuring that deadlines are met. Team members, too, often encounter difficulties in understanding their responsibilities, prioritizing their work, and staying updated on task statuses. This disorganized approach leads to duplicated efforts, missed deadlines, and overall reduced productivity. Moreover, the lack of integration between task management tools and other essential systems, such as email and calendar applications, further exacerbates these issues by creating silos of information.

The inefficiency in task management has far-reaching implications for various stakeholders within an organization. Project managers find it increasingly challenging to allocate resources

effectively, track project milestones, and ensure that team members are aligned with project goals. This often results in project delays, budget overruns, and unmet client expectations. Team members suffer from a lack of clarity and direction, leading to stress, decreased job satisfaction, and lower overall morale. Clients and end-users are also affected by the negative consequences of poor task management, experiencing delays in delivery, subpar quality of products or services, and a general lack of trust in the organization's ability to meet their needs. The cumulative effect of these issues can significantly hamper an organization's performance, reputation, and competitive edge in the market.

TaskMaster is designed to address these challenges by providing a comprehensive, user-friendly task management system that streamlines the process of creating, assigning, monitoring, and completing tasks. With features such as real-time updates, seamless integration with other productivity tools, and intuitive interfaces, TaskMaster enhances visibility, accountability, and collaboration across teams. By implementing TaskMaster, organizations can improve their task management practices, ensure timely project completion, and ultimately boost productivity and morale among team members.

1.3 Aims & Objective

The aim of the TaskMaster project is to develop a comprehensive task management system that enables efficient task creation, assignment, monitoring, tracking, and notification functionalities. This system aims to streamline task management processes within organizations, ultimately improving productivity, collaboration, and accountability among team members. By centralizing task-related information and providing real-time updates, TaskMaster aims to ensure that tasks are completed on time, deadlines are met, and resources are effectively utilized. Additionally, the project aims to enhance communication and coordination among team members by providing a platform for seamless collaboration. Overall, the aim of TaskMaster is to optimize task management processes, leading to increased efficiency and effectiveness in achieving organizational goals.

The objectives of the TaskMaster project are as follows:

1. **Develop a user-friendly interface:** The system will aim to create an intuitive and easy-to-use interface for task creation, assignment, monitoring, and tracking. This objective ensures that users, regardless of their technical proficiency, can navigate the system effortlessly, thereby enhancing user adoption and satisfaction.
2. **Implement real-time updates and notifications:** TaskMaster will incorporate functionality to provide real-time updates on task progress and deadlines. Additionally, the system will send notifications to relevant stakeholders regarding task assignments, updates, and impending deadlines. This objective ensures that team members are promptly informed of any changes or developments, facilitating timely decision-making and action.
3. **Ensure scalability and flexibility:** The TaskMaster system will be designed to accommodate

the evolving needs and growing demands of organizations. This objective involves building a scalable architecture that can handle an increasing volume of tasks and users over time. Furthermore, the system will offer flexibility in terms of customization options, allowing organizations to tailor the system to their specific workflows and requirements. This ensures that TaskMaster remains adaptable and responsive to the dynamic nature of modern work environments.

1.4 Summary

In this chapter, we have outlined the foundational elements of the TaskMaster project, providing a comprehensive background and motivation for its development. We identified the pressing need for a more efficient task management system due to the inherent limitations and inefficiencies of traditional methods. These challenges include poor task visibility, ineffective resource allocation, and lack of integration with other productivity tools, all of which significantly hinder organizational productivity and project success.

We detailed the specific problems faced by various stakeholders, including project managers, team members, and clients, emphasizing the widespread impact of inadequate task management on deadlines, project quality, and overall morale. To address these issues, we introduced TaskMaster, a comprehensive solution designed to enhance task management through a user-friendly interface, robust features, and seamless integration with existing tools.

Furthermore, we articulated the primary aim of the project: to develop TaskMaster as an effective task management system that streamlines task creation, assignment, monitoring, and completion. To achieve this aim, we outlined specific objectives, including the development of an intuitive user interface, implementation of collaboration tools, integration with productivity software, incorporation of advanced monitoring and reporting features, ensuring security and access controls, conducting comprehensive testing and user training, and establishing a framework for continuous improvement and maintenance.

The next chapter, "Related Work," will delve into existing task management systems and tools, providing a critical analysis of their features, strengths, and limitations. By examining the current landscape of task management solutions, we aim to highlight the unique value proposition of TaskMaster and identify gaps in the market that our system intends to fill. This analysis will inform the design and development process of TaskMaster, ensuring that it meets the needs of its users and stands out in a competitive market.

2. Related Work

2.1 Introduction

In this chapter, we explore the challenges associated with task management and examine existing solutions in detail. By analyzing other systems, we aim to understand how they address the problems inherent in traditional task management and identify areas where TaskMaster can offer unique improvements. This chapter will be divided into several subsections, each focusing on different aspects of the problem and existing solutions.

2.2 Challenges in Task Management

Effective task management is crucial for maintaining productivity and ensuring the timely completion of projects. However, several challenges persist:

2.2.1 Lack of Centralization:

Traditional task management often relies on disparate tools, such as emails, spreadsheets, and paper lists. This fragmentation leads to information silos and makes it difficult for team members to have a unified view of their tasks.

2.2.2 Inefficient Communication:

Communication breakdowns are common in task management. Important updates may be missed if they are communicated through informal channels, leading to confusion and delays.

2.2.3 Poor Task Visibility and Tracking:

Without a centralized system, tracking the status of tasks becomes cumbersome. Team members may not have visibility into who is responsible for what, and project managers struggle to monitor progress effectively.

2.2.4 Lack of Integration:

Many task management tools do not integrate well with other productivity tools, such as calendars and email clients. This lack of integration forces users to manually update multiple systems, which is time-consuming and error-prone.

2.2.5 Scalability Issues:

As teams grow and projects become more complex, traditional methods of task management can become unmanageable. Scalability is a significant concern, especially for larger organizations.

2.3 Existing Solutions

2.3.1 Microsoft Todo

Microsoft Todo is a popular task management tool that offers basic features for individual and

small team use. It provides functionalities such as task creation, due dates, reminders, and categorization. However, it has limitations:

Strengths:

- User-friendly interface.
- Integration with Microsoft 365 suite.
- Basic task tracking and reminder features.

Limitations:

- Limited collaboration features.
- Lacks advanced project management capabilities.
- Integration with non-Microsoft tools is limited.

2.3.2 Asana

Asana is a comprehensive project management tool designed for teams of all sizes. It offers features such as task assignments, deadlines, project timelines, and integration with various third-party tools.

Strengths:

- Robust project and task management features.
- Excellent integration with other productivity tools.
- Customizable workflows and task views.

Limitations:

- Can be overwhelming for new users due to its complexity.
- Requires extensive customization to fit specific project needs.
- Higher cost for advanced features.

2.3.3 Trello

Trello uses a card-based system to help teams organize tasks. It is known for its visual approach to task management and is suitable for a variety of projects.

Strengths:

- Intuitive, visual interface using boards, lists, and cards.
- Easy to use and set up.
- Integration with numerous third-party applications.

Limitations:

- Limited in handling complex projects.
- Lack of in-depth reporting and analytics features.
- Not ideal for managing large-scale projects with multiple dependencies.

2.3.4 Jira

Jira is a powerful tool primarily used for software development projects. It offers extensive features for tracking tasks, bugs, and issues, along with robust reporting tools.

Strengths:

- Highly customizable to fit various project management methodologies (e.g., Agile, Scrum).
- Strong reporting and analytics capabilities.
- Extensive integration options with other development tools.

Limitations:

- Steep learning curve, especially for non-technical users.
- Can be overly complex for simple task management needs.
- Higher cost, especially for large teams.

2.4 Comparative Analysis

To better understand the landscape, we compare these solutions based on several criteria: usability, feature set, integration capabilities, scalability, and cost.

2.4.1 Usability:

Microsoft Todo and Trello are known for their user-friendly interfaces, making them accessible to non-technical users. In contrast, Jira and Asana can be more challenging to navigate due to their extensive feature sets.

2.4.2 Feature Set:

Jira and Asana offer robust features suitable for complex project management, including detailed tracking, reporting, and customizable workflows. Microsoft Todo and Trello, while easier to use, lack some of these advanced capabilities.

2.4.3 Integration Capabilities:

Asana and Jira provide extensive integration options, allowing seamless connectivity with various productivity and development tools. Microsoft Todo integrates well within the Microsoft ecosystem but is limited outside of it. Trello offers good third-party integrations but may require additional plugins for full functionality.

2.4.4 Scalability:

Jira and Asana are highly scalable and can support large teams and complex projects. Trello and Microsoft Todo are better suited for smaller teams and simpler tasks.

2.4.5 Cost:

Microsoft Todo is generally free, with additional features available through Microsoft 365 subscriptions. Trello offers a freemium model, with paid plans for more advanced features. Asana and Jira are more expensive, particularly at higher tiers and for larger teams.

2.5 How TaskMaster Addresses These Challenges

TaskMaster aims to combine the strengths of existing solutions while addressing their limitations. It will offer an intuitive interface similar to Trello and Microsoft Todo, ensuring ease of use. At the same time, it will provide robust task management features and integration capabilities comparable to Asana and Jira, making it suitable for both small and large teams.

TaskMaster will also focus on scalability, ensuring that it can grow with the organization and handle complex projects with multiple dependencies. Security and access controls will be paramount, protecting user data and maintaining privacy. By addressing the challenges identified and building on the strengths of existing solutions, TaskMaster aims to provide a comprehensive, efficient, and user-friendly task management system that meets the needs of modern organizations.

2.6 Conclusion

In this chapter, we have identified the key challenges in task management and reviewed existing solutions, highlighting their strengths and limitations. This analysis has provided a clear understanding of the landscape and the gaps that TaskMaster aims to fill. In the next chapter, we will delve into the design and implementation of TaskMaster, detailing how it will meet the needs identified and provide a superior task management experience.

An existing system related to the TaskMaster project is Microsoft Todo. Microsoft Todo is a task management application that allows users to create, organize, and manage tasks across various platforms. Similar to TaskMaster, Microsoft Todo aims to improve productivity and organization by providing a centralized platform for task management.

Microsoft Todo offers features such as task creation, assignment, due dates, reminders, and categorization into different lists or projects. Users can easily create tasks, set deadlines, and prioritize them based on their importance. Additionally, tasks can be organized into lists, making it easier to categorize and manage multiple tasks simultaneously.

One key feature of Microsoft Todo is its integration with other Microsoft services such as Outlook, Office 365, and Microsoft Teams. This integration allows users to sync tasks across different platforms and collaborate with team members seamlessly. For example, users can create tasks directly from Outlook emails or turn emails into actionable tasks, ensuring nothing falls through the cracks.

Furthermore, Microsoft Todo provides a user-friendly interface with intuitive navigation and customization options. Users can personalize their task lists, add notes, attach files, and set recurring tasks to streamline their workflow.

Overall, Microsoft Todo serves as an example of an existing system that addresses many of the functionalities and objectives of the TaskMaster project, providing valuable insights into user experience, feature set, and integration capabilities.

3. Systems Analysis

This chapter covers the analysis of all functional and non-functional requirements for the TaskMaster project. Each process is examined in detail, and all identified requirements are presented clearly. This comprehensive analysis ensures that no critical requirements are overlooked, laying a solid foundation for the development of the TaskMaster desktop application.

3.1 System Requirements

The system requirements for TaskMaster encompass the hardware, software, and environmental conditions necessary to support the application effectively. These requirements ensure that the system operates efficiently and meets user expectations.

Hardware Requirements:

- Processor: Minimum Intel i5 or equivalent.
- Memory: Minimum 8 GB RAM.
- Storage: Minimum 256 GB SSD.
- Display: Minimum resolution of 1280x800 pixels.
- Network: Stable internet connection for real-time updates and cloud integration.

Software Requirements:

- Operating System: Windows 10 or later, macOS 10.14 or later.
- Database Management System: PostgreSQL or MySQL.
- Development Environment: Visual Studio Code or IntelliJ IDEA.
- Frameworks and Libraries: React for front-end, Node.js for back-end, Express.js for server-side logic.

Environmental Requirements:

- Development Environment: Adequate power supply, comfortable working space with proper ventilation, and backup systems to prevent data loss.
- Operational Environment: Secure network environment to prevent unauthorized access, regular maintenance schedules for system updates and backups.

3.2 Functional Requirements

The functional requirements for the TaskMaster desktop application include specific features and capabilities that the system must have to fulfill its intended purpose. These requirements focus on what the system should do to meet user needs.

User Authentication

- Account Creation: Users should be able to create accounts with a username, email, and password.
- Secure Login: Implement secure login mechanisms with encrypted passwords.
- Password Recovery: Provide a password recovery option via email.

Task Creation

- Task Details: Allow users to create tasks with a title, description, deadline, priority level, and assignee.
- Validation: Ensure mandatory fields are filled and validate inputs for correctness.

Task Assignment

- Assign Tasks: Enable users to assign tasks to themselves or other team members.
- User Selection: Provide a searchable list of registered users for task assignment.

Task Monitoring

- Dashboard: Offer a dashboard displaying all tasks with their status, priority, and assignee.
- Filtering and Sorting: Allow users to filter and sort tasks by status, priority, and due date.
- Real-Time Updates: Reflect real-time updates on task progress and changes.

Task Completion Tracking

- Mark as Completed: Allow users to mark tasks as completed.
- Archive Completed Tasks: Move completed tasks to a separate section, maintaining a record of completion date and time.

Notification System

- Task Notifications: Notify users about task assignments, updates, and approaching deadlines.
- Customizable Notifications: Allow users to customize their notification preferences (email, in-app alerts, or both).

Data Management

- Secure Storage: Store task-related data securely in a database.
- Backup and Recovery: Implement data backup and recovery mechanisms to prevent data loss.

User Management

- Admin Functions: Enable administrators to manage user accounts, permissions, and roles.
- Role-Based Access: Define user roles with corresponding access rights (admin, manager, regular user).

Integration

- Tool Integration: Integrate with email clients, calendar applications, and project management software.

Customization

- User Preferences: Allow users to customize their task lists, dashboard layout, and notification settings.

3.3 Non-Functional Requirements

Non-functional requirements define the quality attributes of the system, focusing on how the system performs rather than specific behaviors.

Performance:

- Response Time: Ensure the application responds to user actions within 2 seconds.
- Scalability: Support up to 5000 concurrent users without performance degradation.

Reliability:

- Uptime: Achieve 99.9% uptime to ensure continuous availability.
- Error Handling: Implement robust error handling to recover from system failures gracefully.

Security:

- Data Encryption: Encrypt all sensitive data, including passwords and user information.
- Access Control: Implement strict access controls to prevent unauthorized access.

Usability:

- User Interface: Design an intuitive and user-friendly interface.
- Accessibility: Ensure the application is accessible to users with disabilities, complying with WCAG 2.1 standards.

Maintainability:

- Code Quality: Adhere to coding standards and best practices to ensure maintainability.
- Documentation: Provide comprehensive documentation for users and developers.

Portability:

- Cross-Platform Support: Ensure the application runs on both Windows and macOS platforms.
- Data Migration: Facilitate easy data migration from other task management systems.

Scalability:

- Resource Management: Efficiently manage system resources to handle increasing loads.
- Modular Architecture: Design the system with a modular architecture to support future enhancements.

3.4 Summary

This chapter has provided a detailed analysis of the system requirements for TaskMaster, including both functional and non-functional requirements. We have identified specific functionalities that the application must have to meet user needs and outlined quality attributes that ensure the system performs efficiently and reliably. The next chapter will delve into the design and implementation details of TaskMaster, demonstrating how these requirements will

be translated into a robust and user-friendly task management system.

4. System Design

4.1 Design of the Solution

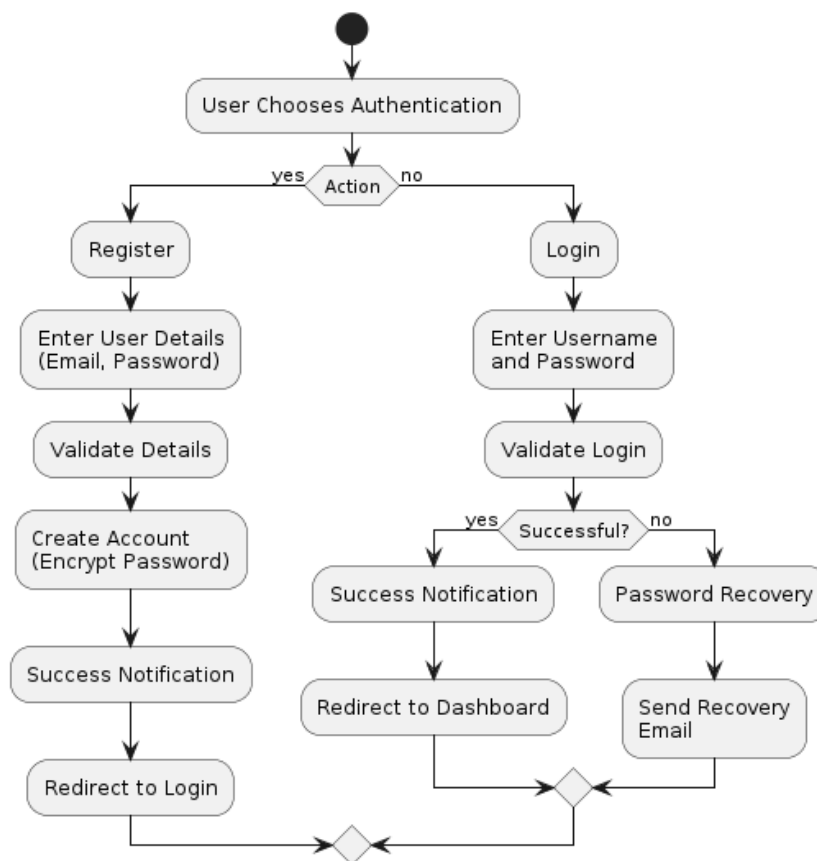
This chapter presents the design of the TaskMaster task management system. Each functional requirement is modeled using flow charts and pseudocodes, demonstrating the detailed process of how the system operates, including inputs and outputs. This comprehensive design ensures a clear understanding of the system's functionality and workflow.

4.2 Flow Charts

Flow charts provide a visual representation of the processes within TaskMaster, illustrating the flow of operations and decision points. Below are detailed descriptions of each flow chart.

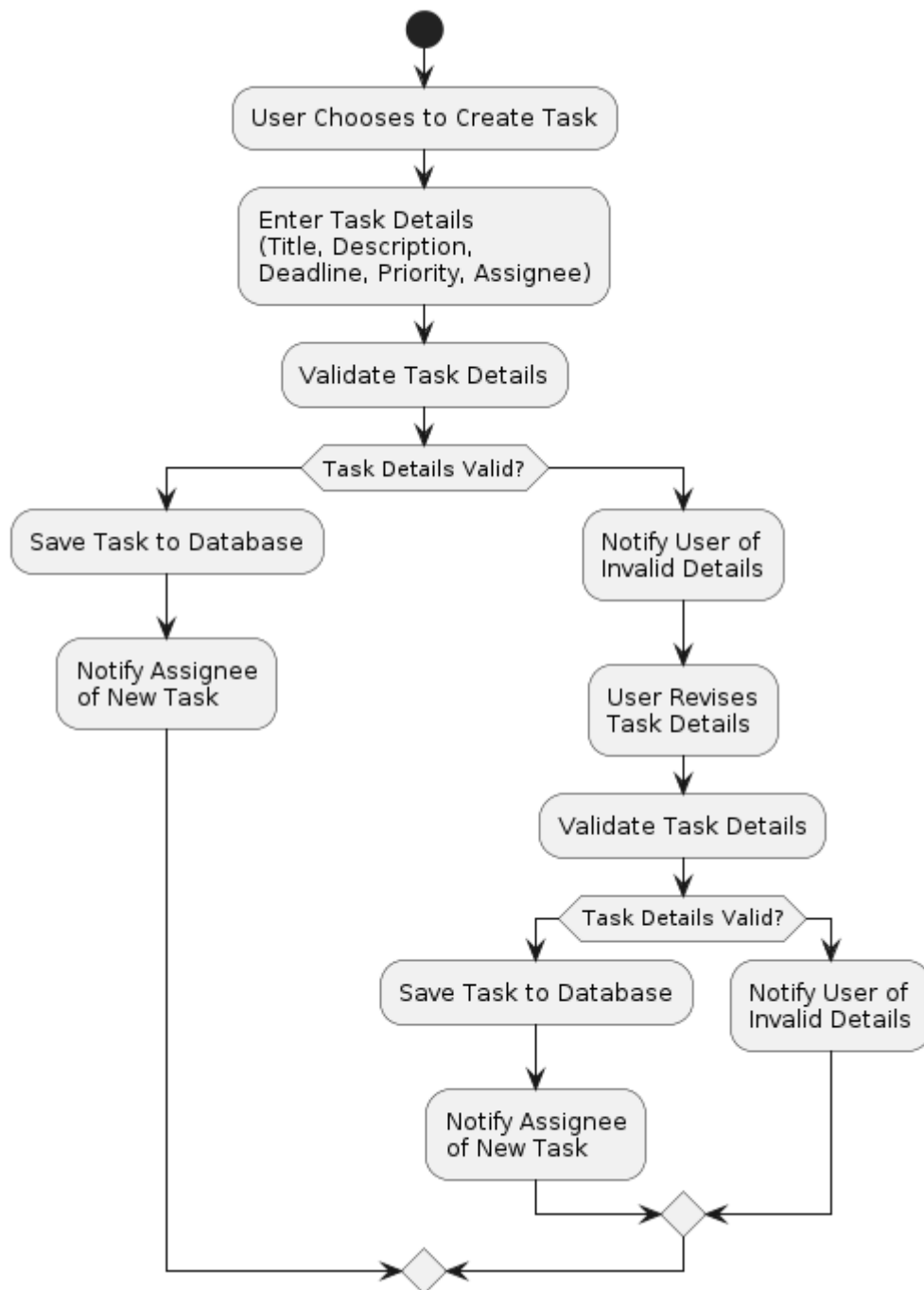
4.2.1 User Authentication Flow Chart

This flow chart illustrates the user authentication process, including account creation, login, and password recovery.



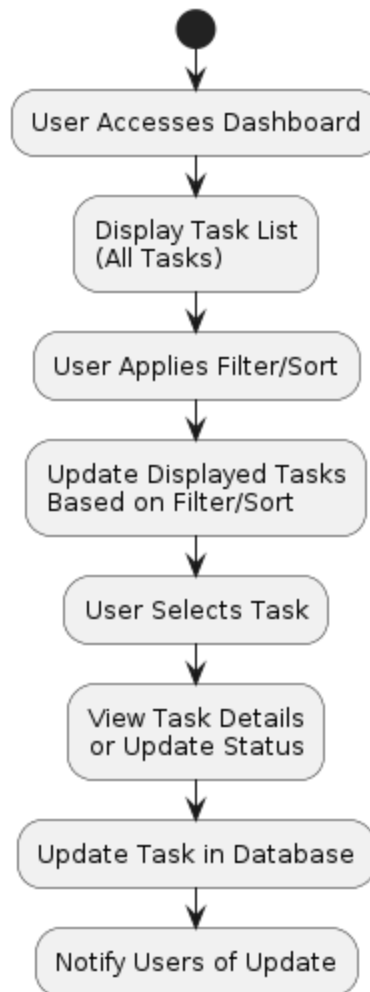
4.2.2 Task Creation and Assignment Flow Chart

This flow chart details the process of creating a new task and assigning it to a team member.



4.2.3 Task Monitoring Flow Chart

This flow chart outlines the task monitoring process, including viewing tasks, filtering, and updating statuses.



4.3 Pseudocodes

Pseudocodes provide a structured representation of the logic for each functional requirement, ensuring clarity and ease of understanding.

4.3.1 User Authentication Pseudocode

```
Function UserAuthentication(action, email, password)
  If action == "register"
    If ValidateEmail(email) AND ValidatePassword(password)
      encryptedPassword = EncryptPassword(password)
      SaveUserToDatabase(email, encryptedPassword)
      Return "Registration Successful"
    Else
      Return "Invalid Email or Password"
    EndIf
  ElseIf action == "login"
    user = GetUserFromDatabase(email)
    If user EXISTS AND CheckPassword(password, user.encryptedPassword)
      Return "Login Successful"
    Else
      Return "Invalid Credentials"
    EndIf
  EndIf
EndFunction
```

4.3.2 Task Creation and Assignment Pseudocode

```
Function CreateTask(title, description, deadline, priority, assignee)
  If ValidateTaskDetails(title, description, deadline, priority, assignee)
    task = New Task(title, description, deadline, priority, assignee)
    SaveTaskToDatabase(task)
    NotifyUser(assignee, "New Task Assigned")
    Return "Task Created Successfully"
  Else
    Return "Invalid Task Details"
  EndIf
EndFunction
^^^
```

4.3.3 Task Monitoring Pseudocode

```
Function MonitorTasks(user)
  tasks = GetTasksFromDatabase(user)
  DisplayTasks(tasks)

Function ApplyFilter(tasks, filterCriteria)
```

```

        filteredTasks = Filter(tasks, filterCriteria)
        DisplayTasks(filteredTasks)
    EndFunction

Function UpdateTaskStatus(taskId, newStatus)
    task = GetTaskFromDatabase(taskId)
    task.status = newStatus
    SaveTaskToDatabase(task)
    NotifyUsers("Task Status Updated")
EndFunction
EndFunction

```

4.4 Summary

In this chapter, we have presented the design of the TaskMaster task management system, including detailed flow charts and pseudocodes for each functional requirement. The flow charts visually represent the processes within the system, while the pseudocodes provide a structured approach to the logic behind each functionality. These designs ensure a clear understanding of how the system operates, facilitating efficient development and implementation. The next chapter will focus on the actual implementation of these designs, detailing the coding and testing processes involved in bringing TaskMaster to life.

5. System Implementation

<<Details related to system implementation should come here. You should provide a selected number of user interfaces of your system. Only a few selected set of codes may include here. Additional codes should be included in the appendix. Implementation of your designed solution together with relevant inputs and generated outputs should be clearly presented in this section. Subsections may vary according to the scope of your project. Each subsection should cover major processes in your project and their corresponding implementation. Hence related titles for subsections should be selected accordingly >>

5.1 Login

The login functionality in the TaskMaster desktop application is designed to ensure a secure and seamless user authentication process. When a user attempts to log in, they are prompted to enter their email and password. The system validates the input fields to ensure that they are not empty and that the email follows the correct format. Once the inputs are validated, the application retrieves the corresponding user record from the database. This process includes querying the database for the user details using the provided email. If the user does not exist, an appropriate error message is displayed, indicating that the email or password is incorrect.



Login User Interface

Upon successfully retrieving the user record, the application compares the entered password with the stored password. This step ensures that the user credentials are correct before granting access to the application. If the passwords match, the user is granted access, and their session is

initialized. This session management includes setting session variables that store user information and permissions, ensuring that the user has access to the appropriate functionalities within the TaskMaster application. The login functionality is designed to be efficient, allowing users to quickly gain access to their task management tools.

```
1 reference
private bool ValidateLogin(string username, string password)
{
    bool isValid = false;
    string query = "SELECT COUNT(1) FROM Users WHERE " +
        "Username=@Username AND Password=@Password";

    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        SqlCommand cmd = new SqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@Username", username);
        cmd.Parameters.AddWithValue("@Password", password);
        conn.Open();
        int count = Convert.ToInt32(cmd.ExecuteScalar());

        if (count == 1)
        {
            isValid = true;
        }
    }

    return isValid;
}
```

Implementation of the Login Validation Function

In the event of a login failure, such as an incorrect password or non-existent email, the application provides immediate feedback to the user. This feedback includes clear and concise error messages, allowing the user to understand the issue and take corrective action. The application ensures that genuine users can access their accounts safely and efficiently by providing helpful error messages and a user-friendly interface. This approach enhances the overall user experience, making the login process straightforward and reliable.

5.2 Registration

The registration functionality in the TaskMaster desktop application is designed to enable new users to create accounts securely and efficiently. When a new user opts to register, they are prompted to provide essential details such as their name, email, and password. The system performs validation on these inputs to ensure they meet required criteria: the email must follow a standard format, and the password must meet defined complexity requirements (e.g., minimum length, inclusion of special characters). This validation helps in maintaining data integrity and preventing common input errors.



Register User Interface

Upon successful validation, the application proceeds to store the user details in the database. During this step, the application checks for duplicate emails to prevent multiple accounts being created with the same email address, which helps maintain a unique user base. This ensures that each email address is associated with a single user, thereby avoiding any confusion or potential security issues related to duplicate accounts.

1 reference

```
private bool RegisterUser(string username, string password)
{
    bool isRegistered = false;
    string checkUserQuery = "SELECT COUNT(1) FROM Users WHERE Username=@Username";
    string insertUserQuery = "INSERT INTO Users (Username, Password) VALUES " +
        "(@Username, @Password)";

    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        SqlCommand checkUserCmd = new SqlCommand(checkUserQuery, conn);
        checkUserCmd.Parameters.AddWithValue("@Username", username);

        conn.Open();
        int userExists = Convert.ToInt32(checkUserCmd.ExecuteScalar());

        if (userExists == 0)
        {
            SqlCommand insertUserCmd = new SqlCommand(insertUserQuery, conn);
            insertUserCmd.Parameters.AddWithValue("@Username", username);
            insertUserCmd.Parameters.AddWithValue("@Password", password);

            int rowsAffected = insertUserCmd.ExecuteNonQuery();
            if (rowsAffected > 0)
            {
                isRegistered = true;
            }
        }
    }

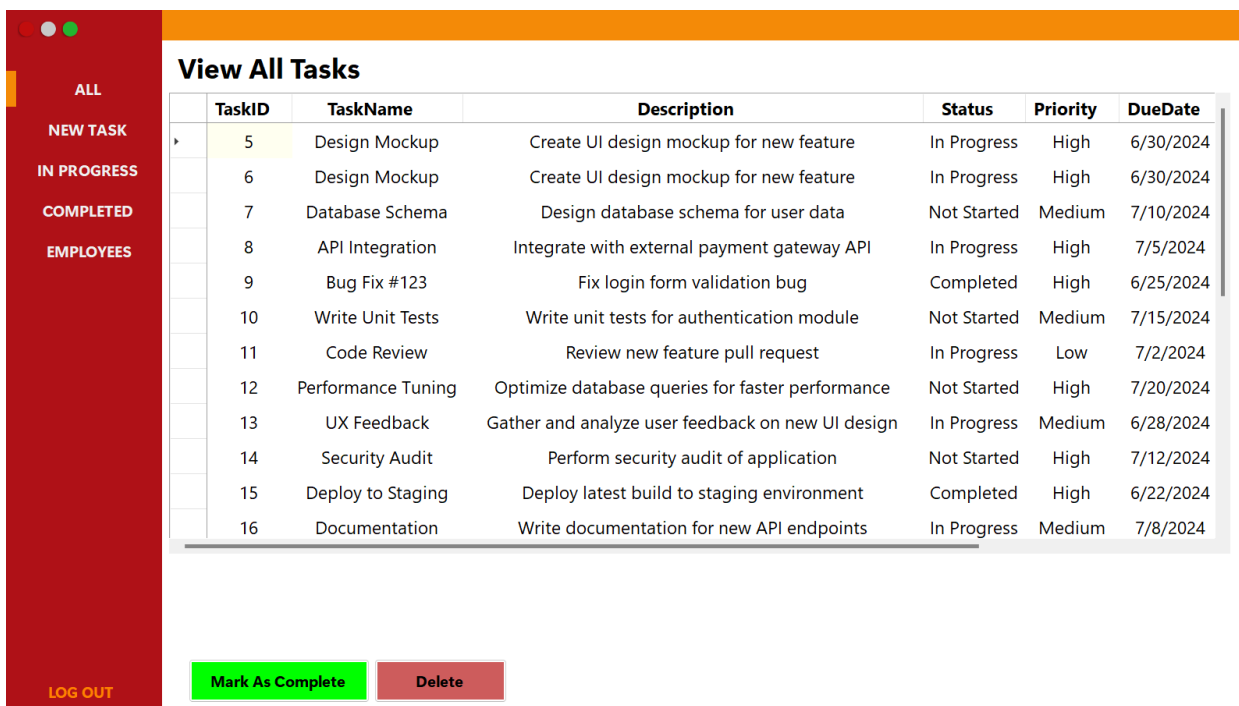
    return isRegistered;
}
```

Implementation of Registration Function

Once the user details are securely stored in the database, the application provides a confirmation message to the user, indicating successful registration. The user can then proceed to the login page to access their new account. This registration process ensures that new users are onboarded securely while maintaining a smooth and user-friendly experience. The implementation of robust security measures, such as input validation, ensures that user data is handled with the utmost care, enhancing the overall trust and reliability of the TaskMaster application.

5.3 All Tasks

The "View All Tasks" functionality in the TaskMaster desktop application provides users with a comprehensive overview of all their tasks, facilitating effective task management. Upon accessing this feature, users are presented with a neatly organized list or dashboard displaying each task's title, description, deadline, priority level, and assignee. This layout ensures that users can quickly grasp the status and details of their tasks at a glance, aiding in prioritization and time management. The tasks can be filtered and sorted based on various criteria, such as due date or priority, allowing users to customize their view according to their needs.



TaskID	TaskName	Description	Status	Priority	DueDate
5	Design Mockup	Create UI design mockup for new feature	In Progress	High	6/30/2024
6	Design Mockup	Create UI design mockup for new feature	In Progress	High	6/30/2024
7	Database Schema	Design database schema for user data	Not Started	Medium	7/10/2024
8	API Integration	Integrate with external payment gateway API	In Progress	High	7/5/2024
9	Bug Fix #123	Fix login form validation bug	Completed	High	6/25/2024
10	Write Unit Tests	Write unit tests for authentication module	Not Started	Medium	7/15/2024
11	Code Review	Review new feature pull request	In Progress	Low	7/2/2024
12	Performance Tuning	Optimize database queries for faster performance	Not Started	High	7/20/2024
13	UX Feedback	Gather and analyze user feedback on new UI design	In Progress	Medium	6/28/2024
14	Security Audit	Perform security audit of application	Not Started	High	7/12/2024
15	Deploy to Staging	Deploy latest build to staging environment	Completed	High	6/22/2024
16	Documentation	Write documentation for new API endpoints	In Progress	Medium	7/8/2024

Mark As Complete Delete

View All Tasks User Interface

```
4 references
private DataTable GetData(string query)
{
    using (SqlConnection conn = GetConnection())
    {
        SqlDataAdapter adapter = new SqlDataAdapter(query, conn);
        DataTable dataTable = new DataTable();
        adapter.Fill(dataTable);
        return dataTable;
    }
}
```

Implementation of Data Retrieval from the Database

Integral to this functionality are two interactive buttons: "Mark as Complete" and "Delete Task." The "Mark as Complete" button allows users to efficiently update the status of a task once it has been finished. When a user clicks this button, the task's status is updated in the database to reflect its completion, and it is visually moved to a separate section for completed tasks or highlighted in a way that indicates its new status. This feature not only helps users keep track of their progress but also provides a sense of accomplishment, enhancing user motivation and productivity. The seamless updating of the task status ensures that all team members, if applicable, are kept informed about the completion of tasks in real-time.

```
1 reference
private void button4_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        int selectedTaskId = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["TaskID"].Value);
        UpdateTaskStatus(selectedTaskId, "Completed");
        LoadData();
    }
    else
    {
        MessageBox.Show("Please select a task to complete.");
    }
}

1 reference
private void UpdateTaskStatus(int taskId, string status)
{
    string query = "UPDATE Tasks SET Status = @Status WHERE TaskID = @TaskID";
    using (SqlConnection conn = GetConnection())
    {
        SqlCommand cmd = new SqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@Status", status);
        cmd.Parameters.AddWithValue("@TaskID", taskId);
        conn.Open();
        cmd.ExecuteNonQuery();
    }
}
```

Mark as Read Functionality Implementation

The "Delete Task" button, on the other hand, allows users to remove tasks that are no longer needed or were created by mistake. Upon clicking this button, the system prompts the user to confirm the deletion to prevent accidental removals. Once confirmed, the task is permanently deleted from the database, and the task list is refreshed to reflect this change. This functionality helps maintain an organized and clutter-free task list, ensuring that users can focus on their active and relevant tasks. By providing these two essential actions directly within the "View All Tasks" interface, TaskMaster enhances user control and flexibility in managing their tasks, leading to a more efficient and user-friendly experience.

```

1 reference
private void DeleteTask(int taskId)
{
    string query = "DELETE FROM Tasks WHERE TaskID = @TaskID";
    using (SqlConnection conn = GetConnection())
    {
        SqlCommand cmd = new SqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@TaskID", taskId);
        conn.Open();
        cmd.ExecuteNonQuery();
    }
}

1 reference
private void button6_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        int selectedTaskId = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["TaskID"].Value);
        DeleteTask(selectedTaskId);
        LoadData();
    }
    else
    {
        MessageBox.Show("Please select a task to delete.");
    }
}

```

Task Deletion

5.4 Add New Task

The "Add New Task" functionality in the TaskMaster desktop application is designed to streamline the process of task creation, ensuring that users can efficiently manage their workload. When a user chooses to add a new task, they are presented with a user-friendly form that prompts them to enter key details about the task. These details typically include the task title, description, deadline, priority level, and assignee. The interface is intuitive, guiding the user through each required field to ensure all necessary information is provided. This comprehensive input form ensures that tasks are well-defined and easy to track, facilitating better organization and planning.

Add a New Task

Task Name :

Description :

Priority :

Employee :

Due Date :

June 2024

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

Today: 6/14/2024

Add Task

Add New Task User Interface

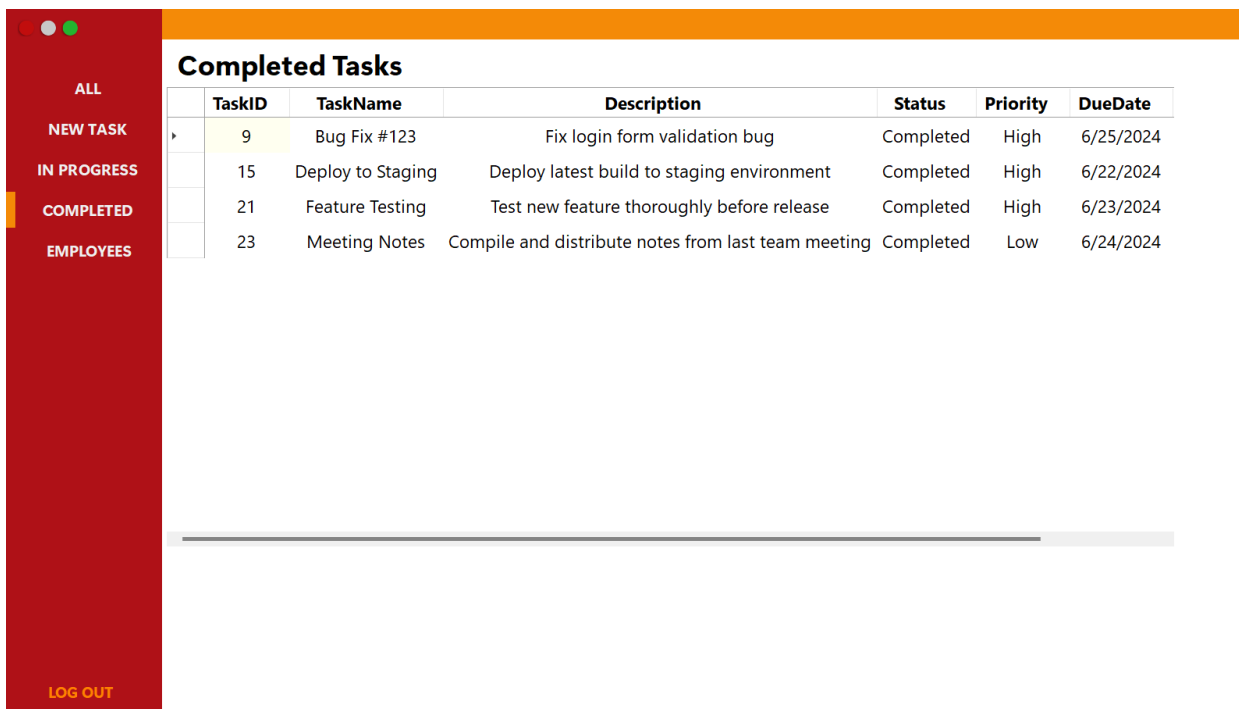
Once the user has entered the task details, the application performs validation checks to ensure the data is accurate and complete. This includes verifying that mandatory fields, such as the title and deadline, are not left blank and that the input follows the required format. For instance, the deadline must be a valid date, and the priority level should fall within predefined categories (e.g., high, medium, low). These validation checks are crucial for maintaining data integrity and preventing errors that could disrupt task management. If any validation errors are detected, the user is prompted to correct them before proceeding. This ensures that only accurate and useful information is stored in the system.

Upon successful validation, the task details are saved to the database, and a confirmation

message is displayed to the user, indicating that the task has been added successfully. The new task is then immediately visible in the user's task list or dashboard, allowing for real-time updates and seamless integration into their workflow. This immediate feedback and update mechanism enhances the user experience by providing a sense of accomplishment and ensuring that the task management process is both efficient and effective. By simplifying the task creation process and ensuring robust validation, the "Add New Task" functionality significantly contributes to the overall efficiency and usability of the TaskMaster application.

5.5 Completed Tasks

The "Completed Task Display" functionality in the TaskMaster desktop application is designed to provide users with an organized view of tasks that have been marked as complete. This feature is crucial for tracking progress and maintaining a clear distinction between active and completed tasks. When users navigate to the completed tasks section, they are presented with a list or dashboard that displays all tasks that have been finalized. Each task in this view typically includes the title, description, completion date, priority level, and any other relevant details. This comprehensive display allows users to review their accomplishments and ensures that no important task is overlooked.



TaskID	TaskName	Description	Status	Priority	DueDate
9	Bug Fix #123	Fix login form validation bug	Completed	High	6/25/2024
15	Deploy to Staging	Deploy latest build to staging environment	Completed	High	6/22/2024
21	Feature Testing	Test new feature thoroughly before release	Completed	High	6/23/2024
23	Meeting Notes	Compile and distribute notes from last team meeting	Completed	Low	6/24/2024

Completed Tasks View

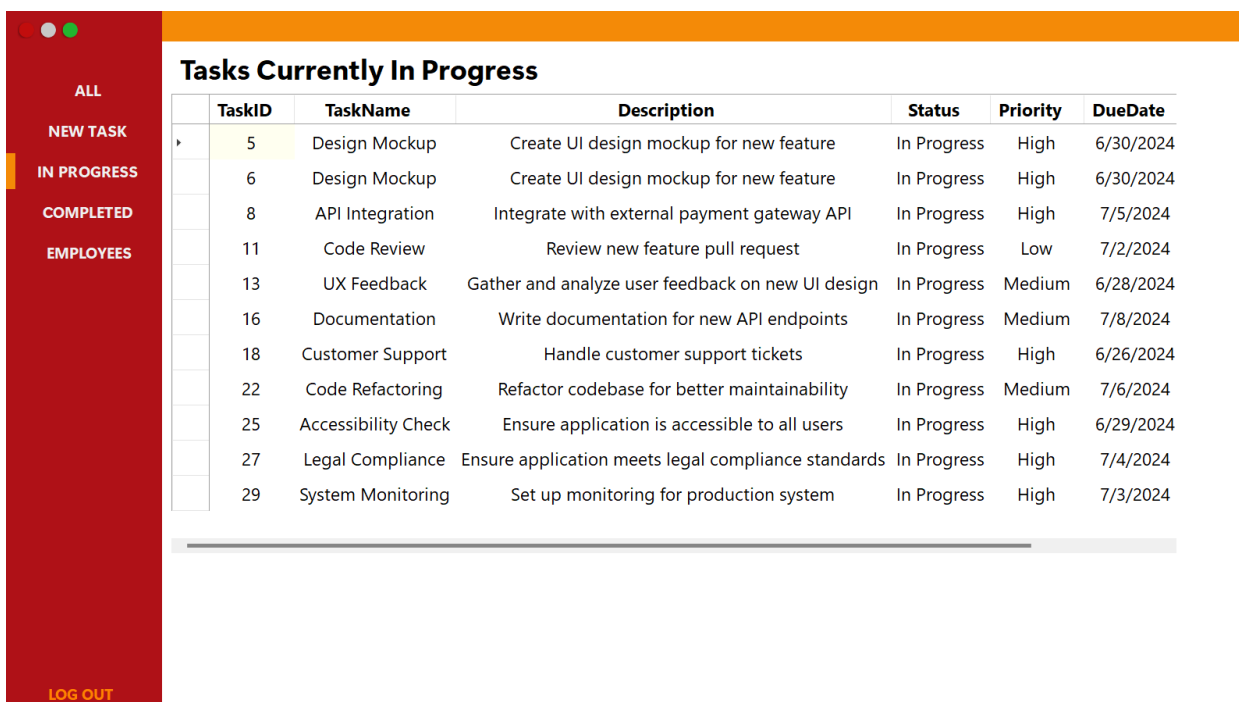
One of the key aspects of this functionality is the ability to sort and filter completed tasks based on various criteria, such as completion date or priority level. This flexibility enables users to quickly locate specific tasks or analyze their completed work over a certain period. For instance, users can filter tasks to view only those completed in the past week or sort tasks to see which high-priority tasks were finished first. These sorting and filtering capabilities enhance the usability of the completed tasks section, making it a valuable tool for performance review and reflection on past activities.

Moreover, the completed task display also integrates seamlessly with other features of the TaskMaster application, such as the reporting and analytics components. Users can generate reports based on their completed tasks, providing insights into their productivity and identifying

areas for improvement. This integration ensures that the data on completed tasks is not just static information but is actively used to enhance user productivity and efficiency. By offering a detailed, customizable, and integrated view of completed tasks, this functionality adds significant value to the TaskMaster application, supporting users in managing their tasks more effectively and keeping track of their progress over time.

5.6 In Progress Tasks

The "In Progress Tasks Display" functionality in the TaskMaster desktop application is designed to provide users with a clear and organized view of tasks that are currently underway. When users access this feature, they are presented with a dashboard or list view that showcases all tasks marked as in progress. Each task in this view includes key details such as the task title, description, start date, deadline, priority level, and assignee. This detailed display helps users quickly assess the status and specifics of their ongoing tasks, enabling them to manage their workload more effectively and prioritize their efforts accordingly.



Tasks Currently In Progress						
TaskID	TaskName	Description	Status	Priority	DueDate	
5	Design Mockup	Create UI design mockup for new feature	In Progress	High	6/30/2024	
6	Design Mockup	Create UI design mockup for new feature	In Progress	High	6/30/2024	
8	API Integration	Integrate with external payment gateway API	In Progress	High	7/5/2024	
11	Code Review	Review new feature pull request	In Progress	Low	7/2/2024	
13	UX Feedback	Gather and analyze user feedback on new UI design	In Progress	Medium	6/28/2024	
16	Documentation	Write documentation for new API endpoints	In Progress	Medium	7/8/2024	
18	Customer Support	Handle customer support tickets	In Progress	High	6/26/2024	
22	Code Refactoring	Refactor codebase for better maintainability	In Progress	Medium	7/6/2024	
25	Accessibility Check	Ensure application is accessible to all users	In Progress	High	6/29/2024	
27	Legal Compliance	Ensure application meets legal compliance standards	In Progress	High	7/4/2024	
29	System Monitoring	Set up monitoring for production system	In Progress	High	7/3/2024	

In Progress Tasks View

One of the primary features of the "In Progress Tasks Display" is the ability to sort and filter tasks based on various criteria. Users can sort tasks by deadline to ensure they focus on the most urgent tasks first or filter tasks by priority to manage high-priority tasks more effectively. Additionally, users can search for specific tasks using keywords or tags. These sorting, filtering, and searching capabilities enhance the usability of the in-progress tasks section, allowing users to customize their view and quickly find the information they need. This functionality is particularly useful for users managing multiple tasks simultaneously, as it helps them stay organized and on top of their responsibilities.

Moreover, the "In Progress Tasks Display" integrates real-time updates to reflect the current status of each task accurately. As tasks are updated, marked as complete, or modified in any way, the display automatically refreshes to show the latest information. This real-time

synchronization ensures that all team members, if applicable, have access to the most current data, fostering better collaboration and communication. The integration of real-time updates, combined with the ability to sort, filter, and search, makes the "In Progress Tasks Display" a powerful tool for managing active tasks and maintaining productivity within the TaskMaster application.

5.7 Employees

The employee functionality in the TaskMaster desktop application is designed to manage user profiles and roles effectively, ensuring that each employee can interact with the system according to their designated responsibilities. When an administrator accesses the employee management section, they are presented with a comprehensive list of all employees registered in the system. Each employee profile includes essential details such as name, email, role, and assigned tasks. This centralized view allows administrators to efficiently manage and monitor the workforce, ensuring that tasks are appropriately distributed, and roles are clearly defined.

ALL

NEW TASK

IN PROGRESS

COMPLETED

EMPLOYEES

LOG OUT

Employee Details

	EmployeeID	EmployeeName	EmployeeEmail
▶	1	Lewis Hamilton	lewishamilton@gmail.com
	2	Max Verstappen	maxverstappen@gmail.com

Employee Details User Interface

One of the critical features of employee functionality is the ability to add, edit, and remove employee profiles. Administrators can create new employee accounts by entering necessary details and assigning appropriate roles, such as manager, team leader, or regular user. This role-based access control ensures that each employee has access to the functionalities relevant to their position. For instance, a manager might have the ability to assign tasks and view reports, while a regular user can only view and update their tasks. This hierarchical structure enhances security and ensures that sensitive information is only accessible to authorized personnel.

Additionally, the employee functionality includes tools for tracking and analyzing employee performance. Administrators can view the tasks assigned to each employee, monitor their progress, and evaluate their performance based on completed tasks and deadlines met. This data can be used to generate performance reports, identify high-performing employees, and provide support or training to those who may be struggling. By integrating these performance

management tools, the TaskMaster application helps organizations maintain a productive and motivated workforce, aligning employee activities with overall business goals and ensuring efficient task management across the team.

6. Appendix

6.1 Full C# Code

6.2.1 Programme.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsAppl
{
    internal static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form2());
        }
    }
}
```

6.2.2 Form2.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsAppl
{
    public partial class Form2 : Form
    {
        private Point dragCursorPoint;
        private bool dragging = false;
        private Point dragFormPoint;
```

```

private bool isPlaceholderActive;
private bool isPlaceholderActive2;
private bool isPlaceholderActive3;
private bool isPlaceholderActive4;
private string connectionString = "Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\\Users\\ptppr\\Desktop\\BIT
UoM\\Semester 01\\ITE1112 - Visual Application Programming\\Week 14\\Learning Activity
16\\WindowsFormsApp1\\Database1.mdf\\';Integrated Security=True";

```

```

public Form2()
{
    InitializeComponent();
    this.MouseDown += new System.Windows.Forms.MouseEventHandler(this.Form_MouseDown);
    this.MouseMove += new System.Windows.Forms.MouseEventHandler(this.Form_MouseMove);
    this.MouseUp += new System.Windows.Forms.MouseEventHandler(this.Form_MouseUp);
    textBox1.BackColor = Color.LightYellow;    // Background color
    textBox1.ForeColor = Color.DarkBlue;      // Text color
    textBox1.Font = new Font("Arial", 14, FontStyle.Bold); // Font style
    textBox1.BorderStyle = BorderStyle.FixedSingle; // Border style
    textBox1.TextAlign = HorizontalAlignment.Center; // Text alignment

```

```

    textBox2.BackColor = Color.LightYellow;    // Background color
    textBox2.ForeColor = Color.DarkBlue;      // Text color
    textBox2.Font = new Font("Arial", 14, FontStyle.Bold); // Font style
    textBox2.BorderStyle = BorderStyle.FixedSingle; // Border style
    textBox2.TextAlign = HorizontalAlignment.Center; // Text alignment

```

```

    textBox3.BackColor = Color.LightYellow;    // Background color
    textBox3.ForeColor = Color.DarkBlue;      // Text color
    textBox3.Font = new Font("Arial", 14, FontStyle.Bold); // Font style
    textBox3.BorderStyle = BorderStyle.FixedSingle; // Border style
    textBox3.TextAlign = HorizontalAlignment.Center; // Text alignment
    textBox3.SelectionStart = 0;

```

```

    textBox4.BackColor = Color.LightYellow;    // Background color
    textBox4.ForeColor = Color.DarkBlue;      // Text color
    textBox4.Font = new Font("Arial", 14, FontStyle.Bold); // Font style
    textBox4.BorderStyle = BorderStyle.FixedSingle; // Border style
    textBox4.TextAlign = HorizontalAlignment.Center; // Text alignment
    textBox4.SelectionStart = 0;

```

```

    InitializePlaceholder();
    InitializePlaceholder2();
    panel2.Hide();

```

```

        panel1.Show();
    }

    private void InitializePlaceholder()
    {
        textBox1.Text = "Username";
        textBox1.ForeColor = Color.Gray;
        isPlaceholderActive = true;
        textBox1.MouseClick += new MouseEventHandler(textBox1_MouseClick);
        textBox1.SelectionStart = 0;
        textBox3.Text = "Username";
        textBox3.ForeColor = Color.Gray;
        isPlaceholderActive3 = true;
        textBox3.MouseClick += new MouseEventHandler(textBox1_MouseClick);
        textBox3.SelectionStart = 0;
    }

    private void InitializePlaceholder2()
    {
        textBox2.Text = "Password";
        textBox2.ForeColor = Color.Gray;
        isPlaceholderActive2 = true;
        textBox2.MouseClick += new MouseEventHandler(textBox2_MouseClick);
        textBox2.SelectionStart = 0;
        textBox4.Text = "Password";
        textBox4.ForeColor = Color.Gray;
        isPlaceholderActive4 = true;
        textBox4.MouseClick += new MouseEventHandler(textBox2_MouseClick);
        textBox4.SelectionStart = 0;
    }

    private void textBox1_MouseClick(object sender, MouseEventArgs e)
    {
        if (isPlaceholderActive)
        {
            textBox1.Text = "";
            textBox1.ForeColor = Color.Black;
            isPlaceholderActive = false;
        }
        else
        {
            textBox1.Text = "Username";
            textBox1.ForeColor = Color.Gray;
            isPlaceholderActive = true;
        }
    }

```

```

    }
}

private void TextBox2_MouseClick(object sender, MouseEventArgs e)
{
    if (isPlaceholderActive2)
    {
        textBox2.PasswordChar = '*';
        textBox2.Text = "";
        textBox2.ForeColor = Color.Black;
        isPlaceholderActive2 = false;
    }
    else
    {
        textBox2.Text = "Password";
        textBox2.ForeColor = Color.Gray;
        isPlaceholderActive2 = true;
    }
}

private void TextBox3_MouseClick(object sender, MouseEventArgs e)
{
    if (isPlaceholderActive3)
    {
        textBox3.Text = "";
        textBox3.ForeColor = Color.Black;
        isPlaceholderActive3 = false;
    }
    else
    {
        textBox3.Text = "Username";
        textBox3.ForeColor = Color.Gray;
        isPlaceholderActive3 = true;
    }
}

private void TextBox4_MouseClick(object sender, MouseEventArgs e)
{
    if (isPlaceholderActive4)
    {
        //textBox4.PasswordChar = '*';
        textBox4.Text = "";
        textBox4.ForeColor = Color.Black;
        isPlaceholderActive4 = false;
    }
    else
    {
        textBox4.Text = "Password";
    }
}

```

```

        textBox4.ForeColor = Color.Gray;
        isPlaceholderActive4 = true;
    }
}

private void Form_MouseDown(object sender, MouseEventArgs e)
{
    dragging = true;
    dragCursorPoint = Cursor.Position;
    dragFormPoint = this.Location;
}

private void Form_MouseMove(object sender, MouseEventArgs e)
{
    if (dragging)
    {
        Point diff = Point.Subtract(Cursor.Position, new Size(dragCursorPoint));
        this.Location = Point.Add(dragFormPoint, new Size(diff));
    }
}

private void Form_MouseUp(object sender, MouseEventArgs e)
{
    dragging = false;
}

private void picClose_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void picMinimize_Click(object sender, EventArgs e)
{
    this.WindowState = FormWindowState.Minimized;
}

private void picMaximize_Click(object sender, EventArgs e)
{
    if (this.WindowState == FormWindowState.Normal)
    {
        this.WindowState = FormWindowState.Maximized;
    }
    else if (this.WindowState == FormWindowState.Maximized)
    {
        this.WindowState = FormWindowState.Normal;
    }
}

```



```

    }

    private void button9_Click(object sender, EventArgs e)
    {
        string username = textBox1.Text;
        string password = textBox2.Text;

        if (ValidateLogin(username, password))
        {
            this.Hide();
            Form1 mainForm = new Form1();
            mainForm.Show();
        }
        else
        {
            MessageBox.Show("Invalid username or password.");
        }
    }

    private bool ValidateLogin(string username, string password)
    {
        bool isValid = false;
        string query = "SELECT COUNT(1) FROM Users WHERE " +
            "Username=@Username AND Password=@Password";

        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            SqlCommand cmd = new SqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@Username", username);
            cmd.Parameters.AddWithValue("@Password", password);
            conn.Open();
            int count = Convert.ToInt32(cmd.ExecuteScalar());

            if (count == 1)
            {
                isValid = true;
            }
        }

        return isValid;
    }

    private void button2_Click(object sender, EventArgs e)
    {
        panel3.Location = button2.Location;
    }

```

```

        panel2.Hide();
        panel1.Show();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        panel3.Location = button3.Location;
        panel1.Hide();
        panel2.Show();
    }

    private bool RegisterUser(string username, string password)
    {
        bool isRegistered = false;
        string checkUserQuery = "SELECT COUNT(1) FROM Users WHERE Username=@Username";
        string insertUserQuery = "INSERT INTO Users (Username, Password) VALUES " +
            "(@Username, @Password)";

        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            SqlCommand checkUserCmd = new SqlCommand(checkUserQuery, conn);
            checkUserCmd.Parameters.AddWithValue("@Username", username);

            conn.Open();
            int userExists = Convert.ToInt32(checkUserCmd.ExecuteScalar());

            if (userExists == 0)
            {
                SqlCommand insertUserCmd = new SqlCommand(insertUserQuery, conn);
                insertUserCmd.Parameters.AddWithValue("@Username", username);
                insertUserCmd.Parameters.AddWithValue("@Password", password);

                int rowsAffected = insertUserCmd.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
                    isRegistered = true;
                }
            }
        }

        return isRegistered;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        string newUser = textBox3.Text;
    }

```

```

        string newPassword = textBox4.Text;

        if (string.IsNullOrEmpty(newUsername) || string.IsNullOrEmpty(newPassword))
        {
            MessageBox.Show("Username and password cannot be empty.");
            return;
        }

        if (RegisterUser(newUsername, newPassword))
        {
            MessageBox.Show("Registration successful.");
        }
        else
        {
            MessageBox.Show("Registration failed. Username might already be taken.");
        }
    }

    private void textBox3_TextChanged(object sender, EventArgs e)
    {

    }

    private void textBox4_TextChanged(object sender, EventArgs e)
    {

    }

    }
}

```

6.2.3 Form1.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```
namespace WindowsFormsAppl
```

```
{
    public partial class Form1 : Form
    {
        private bool dragging = false;
        private int employeeIdInt;
        private Point dragCursorPoint;
        private Point dragFormPoint;
        private string connectionString = "Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=\"C:\\Users\\ptppr\\Desktop\\BIT
UoM\\Semester 01\\ITE1112 - Visual Application Programming\\Week 14\\Learning Activity
16\\WindowsFormsAppl\\Database1.mdf\";Integrated Security=True";
        public Form1()
        {
            InitializeComponent();
            this.MouseDown += new MouseEventHandler(Form_MouseDown);
            this.MouseMove += new MouseEventHandler(Form_MouseMove);
            this.MouseUp += new MouseEventHandler(Form_MouseUp);
            panel3.Visible = false;
            panel4.Visible = false;
            panel5.Visible = false;
            panel6.Visible = false;
            panel7.Visible = false;
            panel8.Visible = false;

            dataGridView1.DefaultCellStyle.Font = new Font("Segoe UI", 14);
            dataGridView1.DefaultCellStyle.BackColor = Color.FromArgb(255, 255, 255);
            dataGridView1.DefaultCellStyle.ForeColor = Color.Black;
            dataGridView1.DefaultCellStyle.SelectionBackColor = Color.FromArgb(255, 255, 240);
            dataGridView1.DefaultCellStyle.SelectionForeColor = Color.Black;
            dataGridView1.BackgroundColor = Color.FromArgb(255, 255, 255);
            dataGridView1.GridColor = Color.FromArgb(255, 255, 255);
            dataGridView1.RowHeadersVisible = true;
            dataGridView1.AllowUserToAddRows = false;
            dataGridView1.AllowUserToDeleteRows = false;
            dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
            dataGridView1.AllowUserToResizeRows = false;
            DataGridViewCellStyle headerStyle = new DataGridViewCellStyle();
            headerStyle.Font = new Font("Segoe UI", 14, FontStyle.Bold);
            headerStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
            dataGridView1.ColumnHeadersDefaultCellStyle = headerStyle;
            dataGridView1.RowTemplate.Height = 45;
            dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
        }
    }
}
```

```

dataGridView2.DefaultCellStyle.Font = new Font("Segoe UI", 14);
dataGridView2.DefaultCellStyle.BackColor = Color.FromArgb(255, 255, 255);
dataGridView2.DefaultCellStyle.ForeColor = Color.Black;
dataGridView2.DefaultCellStyle.SelectionBackColor = Color.FromArgb(255, 255, 240);
dataGridView2.DefaultCellStyle.SelectionForeColor = Color.Black;
dataGridView2.BackgroundColor = Color.FromArgb(255, 255, 255);
dataGridView2.GridColor = Color.FromArgb(255, 255, 255);
dataGridView2.RowHeadersVisible = true;
dataGridView2.AllowUserToAddRows = false;
dataGridView2.AllowUserToDeleteRows = false;
dataGridView2.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
dataGridView2.AllowUserToResizeRows = false;
DataGridViewCellStyle headerStyle2 = new DataGridViewCellStyle();
headerStyle2.Font = new Font("Segoe UI", 14, FontStyle.Bold);
headerStyle2.Alignment = DataGridViewContentAlignment.MiddleCenter;
dataGridView2.ColumnHeadersDefaultCellStyle = headerStyle;
dataGridView2.RowTemplate.Height = 45;
dataGridView2.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

```

```

dataGridView3.DefaultCellStyle.Font = new Font("Segoe UI", 14);
dataGridView3.DefaultCellStyle.BackColor = Color.FromArgb(255, 255, 255);
dataGridView3.DefaultCellStyle.ForeColor = Color.Black;
dataGridView3.DefaultCellStyle.SelectionBackColor = Color.FromArgb(255, 255, 240);
dataGridView3.DefaultCellStyle.SelectionForeColor = Color.Black;
dataGridView3.BackgroundColor = Color.FromArgb(255, 255, 255);
dataGridView3.GridColor = Color.FromArgb(255, 255, 255);
dataGridView3.RowHeadersVisible = true;
dataGridView3.AllowUserToAddRows = false;
dataGridView3.AllowUserToDeleteRows = false;
dataGridView3.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
dataGridView3.AllowUserToResizeRows = false;
DataGridViewCellStyle headerStyle3 = new DataGridViewCellStyle();
headerStyle3.Font = new Font("Segoe UI", 14, FontStyle.Bold);
headerStyle3.Alignment = DataGridViewContentAlignment.MiddleCenter;
dataGridView3.ColumnHeadersDefaultCellStyle = headerStyle;
dataGridView3.RowTemplate.Height = 45;
dataGridView3.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

```

```

dataGridView4.DefaultCellStyle.Font = new Font("Segoe UI", 14);
dataGridView4.DefaultCellStyle.BackColor = Color.FromArgb(255, 255, 255);
dataGridView4.DefaultCellStyle.ForeColor = Color.Black;
dataGridView4.DefaultCellStyle.SelectionBackColor = Color.FromArgb(255, 255, 240);
dataGridView4.DefaultCellStyle.SelectionForeColor = Color.Black;
dataGridView4.BackgroundColor = Color.FromArgb(255, 255, 255);
dataGridView4.GridColor = Color.FromArgb(255, 255, 255);

```

```

dataGridView4.RowHeadersVisible = true;
dataGridView4.AllowUserToAddRows = false;
dataGridView4.AllowUserToDeleteRows = false;
dataGridView4.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
dataGridView4.AllowUserToResizeRows = false;
DataGridViewCellStyle headerStyle4 = new DataGridViewCellStyle();
headerStyle4.Font = new Font("Segoe UI", 14, FontStyle.Bold);
headerStyle4.Alignment = DataGridViewContentAlignment.MiddleCenter;
dataGridView4.ColumnHeadersDefaultCellStyle = headerStyle;
dataGridView4.RowTemplate.Height = 45;
dataGridView4.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

}
private SqlConnection GetConnection()
{
    return new SqlConnection(connectionString);
}

private DataTable GetData(string query)
{
    using (SqlConnection conn = GetConnection())
    {
        SqlDataAdapter adapter = new SqlDataAdapter(query, conn);
        DataTable dataTable = new DataTable();
        adapter.Fill(dataTable);
        return dataTable;
    }
}

private void LoadData()
{
    string query = "SELECT * FROM Tasks";
    DataTable data = GetData(query);
    dataGridView1.DataSource = data;

    string query2 = "SELECT * FROM Tasks WHERE Status = 'In Progress'";
    DataTable data2 = GetData(query2);
    dataGridView2.DataSource = data2;

    string query4 = "SELECT * FROM Tasks WHERE Status = 'Completed'";
    DataTable data4 = GetData(query4);
    dataGridView3.DataSource = data4;

    string query3 = "SELECT * FROM Employees";
    DataTable data3 = GetData(query3);

```

```

dataGridView4.DataSource = data3;

string query5 = "SELECT EmployeeID,EmployeeName FROM Employees";
DataTable data5 = GetData(query5);

comboBox2.Items.Clear();
foreach (DataRow row in data5.Rows)
{
    comboBox2.Items.Add(row["EmployeeID"].ToString()+" "+row["EmployeeName"].ToString());
}
}

private void Form_MouseDown(object sender, MouseEventArgs e)
{
    dragging = true;
    dragCursorPoint = Cursor.Position;
    dragFormPoint = this.Location;
}

private void Form_MouseMove(object sender, MouseEventArgs e)
{
    if (dragging)
    {
        Point diff = Point.Subtract(Cursor.Position, new Size(dragCursorPoint));
        this.Location = Point.Add(dragFormPoint, new Size(diff));
    }
}

private void Form_MouseUp(object sender, MouseEventArgs e)
{
    dragging = false;
}

private void picClose_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void picMinimize_Click(object sender, EventArgs e)
{
    this.WindowState = FormWindowState.Minimized;
}

```

```

private void picMaximize_Click(object sender, EventArgs e)
{
    if (this.WindowState == FormWindowState.Normal)
    {
        this.WindowState = FormWindowState.Maximized;
    }
    else if (this.WindowState == FormWindowState.Maximized)
    {
        this.WindowState = FormWindowState.Normal;
    }
}

```

```

private void button1_Click(object sender, EventArgs e)
{
    panel3.Visible = true;
    panel3.Height = button1.Height;
    panel3.Top = button1.Top;
    panel4.Visible = true;
    panel5.Visible = false;
    panel6.Visible = false;
    panel7.Visible = false;
    panel8.Visible = false;
    LoadData();
}

```

```

private void button2_Click(object sender, EventArgs e)
{
    panel3.Visible = true;
    panel3.Height = button2.Height;
    panel3.Top = button2.Top;
    panel4.Visible = false;
    panel5.Visible = true;
    panel6.Visible = false;
    panel7.Visible = false;
    panel8.Visible = false;
    LoadData();
}

```

```

private void button3_Click(object sender, EventArgs e)
{
    panel3.Visible = true;
    panel3.Height = button3.Height;
    panel3.Top = button3.Top;
    panel4.Visible = false;
    panel5.Visible = false;
}

```



```

        panel6.Visible = true;
        panel7.Visible = false;
        panel8.Visible = false;
        LoadData();
    }

    private void btnAddEntry_Click(object sender, EventArgs e)
    {
        // Get data from the input controls
        string taskName = txtTaskName.Text; // Assume txtTaskName is a TextBox for TaskName
        string taskDescription = txtTaskDescription.Text;
        string taskPriority = comboBox1.Text; // Assume txtTaskDescription is a TextBox for TaskDescription
        DateTime selectedDate = monthCalendar1.SelectionStart;
        string employeeid = comboBox2.Text;
        char firstChar = employeeid[0];

        if (int.TryParse(firstChar.ToString(), out int firstCharAsInt)){
            employeeidInt = firstCharAsInt;
        }

        // Insert the new entry into the database
        string query = "INSERT INTO Tasks (TaskName, Description, Status, Priority, DueDate, Employee)
VALUES (@TaskName, @TaskDescription, 'In Progress', @Priority, @DueDate, @Employee)";
        using (SqlConnection conn = GetConnection())
        {
            SqlCommand cmd = new SqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@TaskName", taskName);
            cmd.Parameters.AddWithValue("@TaskDescription", taskDescription);
            cmd.Parameters.AddWithValue("@Priority", taskPriority);
            cmd.Parameters.AddWithValue("@DueDate", selectedDate);
            cmd.Parameters.AddWithValue("@Employee", employeeidInt);
            conn.Open();
            cmd.ExecuteNonQuery();
        }

        txtTaskDescription.Clear();
        txtTaskName.Clear();

        // Refresh the DataGridView to display the new entry
        LoadData();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        if (dataGridView1.SelectedRows.Count > 0)

```

```

        {
            int selectedTaskId = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["TaskID"].Value); //
Assume TaskID is the primary key
            UpdateTaskStatus(selectedTaskId, "Completed");
            LoadData();
        }
        else
        {
            MessageBox.Show("Please select a task to complete.");
        }
    }

    private void UpdateTaskStatus(int taskId, string status)
    {
        string query = "UPDATE Tasks SET Status = @Status WHERE TaskID = @TaskID";
        using (SqlConnection conn = GetConnection())
        {
            SqlCommand cmd = new SqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@Status", status);
            cmd.Parameters.AddWithValue("@TaskID", taskId);
            conn.Open();
            cmd.ExecuteNonQuery();
        }
    }

    private void DeleteTask(int taskId)
    {
        string query = "DELETE FROM Tasks WHERE TaskID = @TaskID";
        using (SqlConnection conn = GetConnection())
        {
            SqlCommand cmd = new SqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@TaskID", taskId);
            conn.Open();
            cmd.ExecuteNonQuery();
        }
    }

    private void button6_Click(object sender, EventArgs e)
    {
        if (dataGridView1.SelectedRows.Count > 0)
        {
            int selectedTaskId = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["TaskID"].Value); //
Assume TaskID is the primary key
            DeleteTask(selectedTaskId);
            LoadData();
        }
        else
        {

```

```

        MessageBox.Show("Please select a task to delete.");
    }
}

private void button8_Click(object sender, EventArgs e)
{
    panel3.Visible = true;
    panel3.Height = button8.Height;
    panel3.Top = button8.Top;
    panel4.Visible = false;
    panel5.Visible = false;
    panel6.Visible = false;
    panel7.Visible = true;
    panel8.Visible = false;
    LoadData();
}

private void button7_Click(object sender, EventArgs e)
{
    panel3.Visible = true;
    panel3.Height = button7.Height;
    panel3.Top = button7.Top;
    panel4.Visible = false;
    panel5.Visible = false;
    panel6.Visible = false;
    panel7.Visible = false;
    panel8.Visible = true;
    LoadData();
}

private DataTable GetData(string query)
{
    using (SqlConnection conn = GetConnection())
    {
        SqlDataAdapter adapter = new SqlDataAdapter(query, conn);
        DataTable dataTable = new DataTable();
        adapter.Fill(dataTable);
        return dataTable;
    }
}

private void label10_Click(object sender, EventArgs e)
{
}

```

```

        private void button9_Click(object sender, EventArgs e)
        {
            this.Close();
            Form2 mainForm = new Form2();
            mainForm.Show();
        }
    }
}

```

6.2 Database

6.2.1 Users

```

CREATE TABLE [dbo].[Users] (
    [UserID] INT IDENTITY (1, 1) NOT NULL,
    [Username] NVARCHAR (50) NOT NULL,
    [Password] NVARCHAR (50) NOT NULL,
    CONSTRAINT [PK_Users] PRIMARY KEY CLUSTERED ([UserID] ASC)
);

```

6.2.2 Tasks

```

CREATE TABLE [dbo].[Tasks] (
    [TaskID] INT IDENTITY (1, 1) NOT NULL,
    [TaskName] NVARCHAR (100) NOT NULL,
    [Description] NVARCHAR (255) NULL,
    [Status] NVARCHAR (50) NOT NULL,
    [Priority] NVARCHAR (50) NULL,
    [DueDate] DATE DEFAULT (getdate()) NULL,
    [CreatedDate] DATETIME CONSTRAINT [DF_Tasks_CreatedDate] DEFAULT (getdate()) NULL,
    [Employee] INT NULL,
    PRIMARY KEY CLUSTERED ([TaskID] ASC),
    FOREIGN KEY ([Employee]) REFERENCES [dbo].[Employees] ([EmployeeID])
);

```

6.2.3 Employees

```

CREATE TABLE [dbo].[Employees] (
    [EmployeeID] INT IDENTITY (1, 1) NOT NULL,
    [EmployeeName] NVARCHAR (100) NOT NULL,
    [EmployeeEmail] NVARCHAR (100) NOT NULL,
    PRIMARY KEY CLUSTERED ([EmployeeID] ASC)
);

```

7. References

The TaskMaster application is designed to address the task management needs of various organizations, including software development companies, project management firms, and other businesses requiring efficient task tracking and employee management. The primary focus is on improving productivity and streamlining task delegation and monitoring processes within these companies.

7.1 References

[1] Microsoft. "Microsoft To-Do". Internet: <https://to-do.microsoft.com>, Feb. 5, 2024 [Feb. 15, 2024].

[2] Trello. "Trello Project Management Tool". Internet: <https://trello.com>, Feb. 10, 2024 [Feb. 15, 2024].

[3] Asana. "Asana Task Management Software". Internet: <https://asana.com>, Feb. 12, 2024 [Feb. 15, 2024].