**Group 1**

**SRS Link:** [**https://github.com/tejakumarsai6309/44691-03-GDP-team1/wiki/Software-requirements-specification-(SRS)**](https://github.com/tejakumarsai6309/44691-03-GDP-team1/wiki/Software-requirements-specification-(SRS))

**Software requirements specification (SRS)**

# Project Information:

# Link: <https://github.com/tejakumarsai6309/44691-03-GDP-team1/wiki/Project-Information>

# Project Charter

# Project Charter: TaskMaster

## **1. Brief Summary of the Problem:**

Taskmaster app for managing tasks, whether you're working alone or with a team. It's really easy to use and has lots of cool features. You can make lists for your tasks, organize them how you like, and even share them with your team. It works on computers and phones, so you can use it anywhere. Plus, it reminds you of deadlines, helps you figure out what to do first, and tracks your progress. You can also customize it to fit your way of working, and it works with other apps too. Basically, Taskmaster makes it easy to get stuff done!

## **2. Motivation for Solving the Problem:**

It attempts to address the problems people have daily with task management. Taskmaster aims to improve productivity by providing an easy-to-use solution that streamlines job prioritizing, organizing, and collaboration. It guarantees that users may remain productive whether using a computer and solves the requirement for dependable access to task management tools, particularly as teams expand. Taskmaster aims to enable people and teams to effortlessly accomplish their objectives by providing a streamlined UI and crucial features such as real-time synchronization and reminders.

## **3. Required Functionality:**

Taskmaster will do a bunch of things to solve these problems:

* **Task Management:** You can create, assign, and keep track of tasks easily.
* **Deadline Management:** Get reminders so you don't miss important deadlines.
* **Reliability:** The app will always be available and work well.
* **Scalability:** It'll grow as more people use it without slowing down.
* **Easy to Use:** The app will be simple and intuitive to use, with filters and search to find tasks quickly.
* **Calendar View:** You can see tasks on a calendar to plan better.
* **Backend Stuff:** The behind-the-scenes stuff like user accounts and data storage will be secure and reliable.
* **Real-Time Updates:** Changes made on one device will show up instantly on others.
* **Security:** Your data will be safe with strong login and encryption.
* **Database:** It'll use either a standard or a flexible type of database, depending on what's best for the application.

#### Optional Requirements

* **Collaboration:** Work together with your team by assigning tasks and leaving comments.

# About the Developers Page:

# About the Developers

| **Username** | **Git Link** | **Strengths in Computing** | **Interests** |
| --- | --- | --- | --- |
| laxminarayana2911 | [GitHub](https://github.com/laxminarayana2911) | **Programming:** Java, C, Python, Ruby, Scala **Web Development:** HTML, CSS, JavaScript, ReactJS, AngularJS **Cloud:** AWS, Google **Databases:** MongoDB, MySQL | Interested in learning Ethical Hacking and Web Development |
| tejakumarsai6309 | [GitHub](https://github.com/tejakumarsai6309) | **Programming:** Java, C, Apex **Web Development:** HTML, CSS, JavaScript, ReactJS **Cloud:** AWS, Google, Salesforce **Databases:** PgAdmin, MongoDB, MySQL | Interested in Cloud Computing, Machine Learning, Artificial Intelligence |
| Harichaithanya Kotapati | [GitHub](https://github.com/HariChaithanya) | **Programming:** Java, Python **Web Development:** HTML, CSS, JavaScript **Cloud:** AWS, GCP **Databases:** MongoDB, MySQL | Interested in making Web Projects using Java |
| Vaishnavi130820 | [GitHub](https://github.com/Vaishnavi130820) | **Programming:** Java, C, C++ **Web Development:** HTML, CSS, JavaScript, AngularJS **Cloud:** Azure **Databases:** MongoDB, MySQL | Interested in learning Cyber Security, Artificial Intelligence |
| Manichandra Kuntala | [GitHub](https://github.com/MANICHANDRA900) | **Programming:** Java, C **Web Development:** HTML, CSS, JavaScript **Cloud:** Microsoft Azure **Databases:** MongoDB, MySQL | Interested in learning coding practices |

# Problem Statement

## **Problem Statement (Draft 2)**

### **Inventory Management Inefficiencies:**

Users are unable to create, assign, organize, or monitor tasks with the present task management system. This results in a disorganized work environment, unclear roles, and missed deadlines. The workflow becomes disorganized and imprecise when tasks are not assigned or tracked clearly.

### **Data Management Challenges:**

Important information is hard to find because of the client's antiquated and ineffective data processing techniques. Inefficient use of data and discrepancies result from the lack of a centralized data repository. The company's capacity for wise decision-making and peak performance is hampered by this disarray.

### **Manual Processes and Outdated Technologies:**

Task management efficiency is reduced when antiquated technologies and laborious manual procedures are used. Task completion times are longer and error rates are higher in the absence of automation technologies. The system's efficiency and dependability are impacted by the absence of contemporary development tools.

### **Security Vulnerabilities:**

Because the existing system is not equipped with strong security measures, important task data is susceptible to hostile assaults and illegal access. Important data may be compromised by inadequate data encryption and shoddy user authentication, endangering user privacy and system integrity.

### **Lack of Prioritization and Tracking:**

* Current methods do not adequately support task prioritization or provide effective tracking mechanisms. Users struggle to prioritize their workload and often rely on personal judgment or rudimentary systems that do not scale well.
* Tracking task progress and completion is often manual and lacks real-time updates, making it difficult to assess the current status of projects and identify potential bottlenecks.

### **Deadlines and Reminders Management:**

Setting and managing deadlines and reminders with the current tools is difficult. Users often miss deadlines because there are no automatic reminders, which affects project timelines. Manually setting deadlines and reminders is prone to mistakes, causing missed or incorrect deadlines.

### **Summary**

Task organization inefficiencies, antiquated data management procedures, a reliance on manual processes, and serious security flaws afflict the client's present task management system. Together, these problems impair decision-making, productivity, and data security, calling for a thorough redesign to satisfy contemporary efficiency and security requirements.

**Project Design:**

**Link:** [**https://github.com/tejakumarsai6309/44691-03-GDP-team1/wiki/Design**](https://github.com/tejakumarsai6309/44691-03-GDP-team1/wiki/Design)

**Use Cases:**

# Use Cases (Iteration 2)

#### **Use Case 1: User Registration and Authentication**

**Accomplish:** Enable users to create accounts, authenticate securely, and manage their tasks effectively.

**Functionality:**

* Users input registration information: username or email, password.
* System validates and securely stores user credentials.
* Upon successful registration or login, users gain access to their personalized task management dashboard.

**Input:**

* User-provided information: username or email, password.

**Input Execution:**

* Validate user-provided information.
* Securely store user credentials.
* Authenticate users securely during login.

**Output:**

* Confirmation of successful registration or login.
* Access granted to the user's personalized task management dashboard.

#### **Use Case 2: Task Creation**

**Accomplish:** Allow users to create tasks with relevant details for effective task management.

**Functionality:**

* Users input task details: title, description, priority, deadline.
* System validates input and creates a new task with specified attributes.
* Users receive confirmation of successful task creation and can review the details.

**Input:**

* Task details: title, description, priority, deadline.

**Input Execution:**

* Validate task details.
* Create a new task with specified attributes.

**Output:**

* Confirmation of successful task creation.
* Display of the created task details.

#### **Use Case 3: Task Assignment and Prioritization**

**Accomplish:** Enable users to assign tasks to team members and prioritize them based on importance.

**Functionality:**

* Users select a task and assign it to individual team members or groups.
* System updates the task with assigned user(s) and displays its status as assigned.
* Users can prioritize tasks by setting priority levels.

**Input:**

* Selected task.
* Assigned user(s).
* Priority level.

**Input Execution:**

* Update task details with assigned user(s) and priority level.
* Display task status as assigned.

**Output:**

* Confirmation of successful task assignment.
* Clear visibility of assigned tasks and priorities.

#### **Use Case 4: Read Task Details**

**Accomplish:** Enable users to view and access detailed information about existing tasks.

**Functionality:**

* Users access the task list or search for specific tasks.
* Upon selection, users view task details including title, description, priority, deadline, assignee, status, comments, and attachments.

**Input:**

* Selected task.

**Input Execution:**

* Retrieve task details from the database.
* Display task details.

**Output:**

* Display comprehensive task details for user review.

#### **Use Case 5: Update Task Details**

**Accomplish:** Allow users to modify and update existing task attributes to reflect changes or updates.

**Functionality:**

* Users select the task to update and navigate to its details page.
* Users edit task attributes such as title, description, priority, deadline, assignee, and status.
* System validates and applies the changes to the task.
* Users receive confirmation of the successful update.

**Input:**

* Selected task.
* Updated task attributes: title, description, priority, deadline, assignee, status.

**Input Execution:**

* Validate updated task attributes.
* Update task details.

**Output:**

* Confirmation of successful task update.
* Display of the revised task details.

#### **Use Case 6: Delete Task**

**Accomplish:** Enable users to remove tasks that are no longer relevant or necessary from the task management system.

**Functionality:**

* Users select the task to delete.
* Users confirm the deletion action.
* System prompts for confirmation to prevent accidental deletion.
* Upon confirmation, the system removes the task from the database.

**Input:**

* Selected task.

**Input Execution:**

* Prompt user for confirmation.
* Remove task from the database.

**Output:**

* Confirmation of successful task deletion.
* Task removed from the task management system.

#### **Use Case 7: Task Tracking and Monitoring**

**Accomplish:** Enable users to track the progress of tasks and monitor their status in real-time.

**Functionality:**

* Users view task status including pending, in progress, completed, or overdue.
* Real-time updates and notifications inform users of any changes.
* Task progress can be monitored through visual indicators or progress bars.

**Input:**

* Selected task.

**Input Execution:**

* Retrieve task status and progress.
* Provide real-time updates and notifications.

**Output:**

* Display of task status and progress.
* Real-time updates on task changes.

#### **Use Case 8: Task Filtering and Search Functionality**

**Accomplish:** Enable users to filter and search for tasks based on various criteria for efficient task organization.

**Functionality:**

* Users filter tasks by attributes like priority, status, assignee, and deadline.
* Advanced search allows users to search for specific tasks using keywords.
* Results are displayed for easy navigation.

**Input:**

* Filtering criteria: priority, status, assignee, deadline.
* Search query: keywords or phrases.

**System Functionality:**

* Apply filtering criteria.
* Execute search query.

**Output:**

* Display of filtered or searched tasks.
* Facilitated navigation and access.

#### **Use Case 9: Calendar View for Task Management**

**Accomplish:** Provide users with a calendar view to visualize task deadlines and schedules for better planning and organization.

**Functionality:**

* Users switch to a calendar view to see tasks represented as events with deadlines.
* Task deadlines and due dates are displayed.
* Users can adjust deadlines by dragging tasks on the calendar.

**Input:**

* Selected calendar view option.

**Input Execution:**

* Display tasks as events on the calendar.
* Show task deadlines and due dates.

**Output:**

* Visual representation of tasks on the calendar.
* Enhanced planning and organization.

### **Use Case 10: Forgot Password**

**Accomplish:** Enable users to reset their passwords securely if they have forgotten them.

**Functionality:**

1. Users initiate the password reset process by clicking on the "Forgot Password" link on the login page.
2. The system prompts users to enter their registered email address.
3. Users submit their email address, triggering a password reset email to be sent to their inbox.
4. The email contains a unique password reset link that expires after a specified period for security purposes.
5. Users click on the reset link, redirecting them to a password reset page where they can enter a new password.
6. After submitting the new password, the system verifies and updates it in the database, allowing users to log in with their new credentials.

**Input:**

* User's registered email address.

**Input Execution:**

* Initiate password reset process by clicking "Forgot Password" link.
* Submit email address for password reset.
* Click on password reset link received in email.
* Enter and submit new password on password reset page.

**Output:**

* Password reset email sent to user's inbox.
* Confirmation message after successfully resetting the password.
* Ability for users to log in with their new password.

### **Use Case 11: Send Reminders**

**Accomplish:** Provide users with reminders to keep them informed about upcoming tasks and changes.

**Functionality:**

1. A reminder is sent to users for approaching deadlines or task updates.

**Input:**

* Selected task.
* Reminder settings.

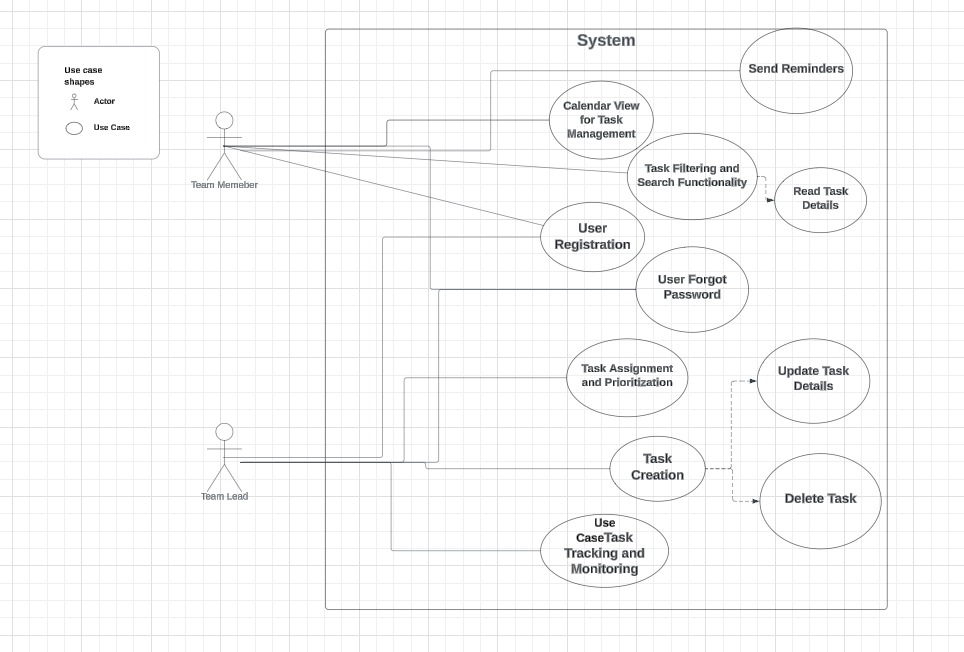
**Input Execution:**

* Set reminders for selected tasks.
* Send reminders for approaching deadlines.

**Output:**

* Notification of upcoming tasks.
* Improved task management and time management.

## **Use Case Diagram**



**Functional Requirements:**

**Functional Requirements Iteration 2**

**Absolute Requirements**

1. The system SHALL Create tasks based on user input.
2. The system SHALL Assign tasks to specific users or teams.
3. The system SHALL Prioritize tasks according to user-defined criteria.
4. The system SHALL Secure user authentication and authorization.
5. The system SHALL Store and retrieve task data securely.
6. The system SHALL Provide a clean and intuitive user interface for easy navigation and task management.
7. The system SHALL Offer tasks lists, filter and search functionality.
8. The system SHALL Display tasks and their deadlines in a calendar view to aid in scheduling and planning.
9. The system SHALL Allow team members to collaborate on tasks by sharing updates and comments.
10. The system SHALL encrypt and store the Passwords securely.
11. The system SHALL enable the users to reset their passwords.
12. The system SHALL store the users actions after logging in.

### **Recommended Requirements**

1. The system SHOULD Synchronize task data across platforms in real-time to ensure consistency.
2. The system SHOULD Track the progress of tasks throughout their lifecycle.
3. The system SHOULD send email notifications to users for password resetting.
4. The system SHOULD store the sessions information and its timestamp for user authentication.

### **Optional Requirements:**

1. The system MAY Set deadlines for tasks based on user input.
2. The system MAY Send reminders to users about upcoming task deadlines.

**Non-Functional Requirements:**

* Role-based access control to manage user permissions.
* MySQL Database scalability to accommodate the growing volume of task data and users.
* Encryption of sensitive task data to protect against unauthorized access.
* Implementation of secure protocols for user authentication and session management.
* The application should be compatible with a wide range of web browsers (Chrome, Firefox, Safari, Edge).
* High availability and reliability.
* Scalability to accommodate an increasing user base.

**Data Management Plan**

**Data Management Plan (Iteration 2)**

**Users Table:**

| **Field** | **Data Type** | **Description** |
| --- | --- | --- |
| User ID | INTEGER (Primary Key) | Unique identifier for each user. |
| Username | VARCHAR(50) | User's chosen username. |
| Email | VARCHAR(100) | User's email address. |
| Password | VARCHAR(255) (encrypted) | Encrypted password for user authentication. |
| Role | VARCHAR(100) | Role of the user (team leader or team member). |

**Tasks Table:**

| **Field** | **Data Type** | **Description** |
| --- | --- | --- |
| Task ID | INTEGER (Primary Key) | Unique identifier for each task. |
| Description | TEXT | Description of the task. |
| Priority | VARCHAR(100) | Priority level of the task.('Low', 'Medium', 'High') |
| Deadline | DATETIME | Deadline for completing the task. |
| Status | VARCHAR(100) | Current status of the task. ('Pending', 'In Progress', 'Completed') |
| Assigned To | INTEGER (Foreign Key to Users) | User ID of the user to whom the task is assigned. |

**Task Assignments Table:**

| **Field** | **Data Type** | **Description** |
| --- | --- | --- |
| Assignment ID | INTEGER (Primary Key) | Unique identifier for each task assignment. |
| Task ID | INTEGER (Foreign Key to Tasks) | Task ID of the assigned task. |
| User ID | INTEGER (Foreign Key to Users) | User ID of the user assigned to the task. |

**Sessions Table:**

| **Field** | **Data Type** | **Description** |
| --- | --- | --- |
| Session ID | INTEGER (Primary Key) | Unique identifier for each session. |
| User ID | INTEGER (Foreign Key to Users) | User ID of the user associated with the session. |
| Timestamp | DATETIME | Timestamp indicating when the session occurred. |

**Logs Table:**

| **Field** | **Data Type** | **Description** |
| --- | --- | --- |
| ID | INTEGER (Primary Key) | Unique identifier for each log entry. |
| User ID | INTEGER (Foreign Key to Users) | User ID of the user associated with the log entry. |
| Action | TEXT | Description of the action performed (e.g., task creation, assignment). |
| Timestamp | DATETIME | Timestamp indicating when the action occurred. |

**Reminder Table:**

| **Field** | **Data Type** | **Description** |
| --- | --- | --- |
| rem\_id | INTEGER (Primary Key) | Unique identifier for each reminder. |
| Task\_ID | INTEGER (Foreign Key) | Reference to the associated task. |
| reminderTime | TIMESTAMP | Date and time when the reminder is scheduled. |

**Relationships Between Tables:**

1. **Users Table**:
   * One-to-Many Relationship with **Tasks Table**: One user can have many tasks assigned to them.
   * One-to-Many Relationship with **Task Assignments Table**: One user can be assigned to many tasks.
   * One-to-Many Relationship with **Sessions Table**: One user can have multiple sessions.
2. **Tasks Table**:
   * One-to-Many Relationship with **Task Assignments Table**: One task can have multiple assignments.
3. **Task Assignments Table**:
   * Many-to-One Relationship with **Users Table**: Many task assignments can be associated with one user.
   * Many-to-One Relationship with **Tasks Table**: Many task assignments can be associated with one task.
4. **Sessions Table**:
   * Many-to-One Relationship with **Users Table**: Many sessions can be associated with one user.
5. **Logs Table**:
   * Many-to-One Relationship with **Users Table**: Many logs can be associated with one user.
6. **Reminder Table**:
   * One-to-one Relationship with **Tasks Table**: one reminder can be associated with one task.

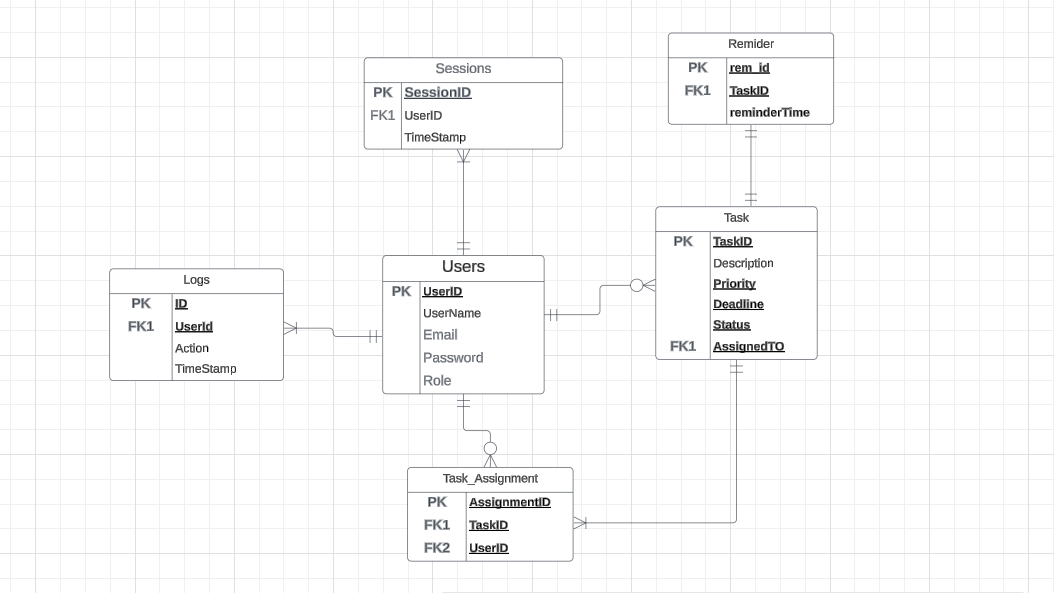
**Initial Plans to Secure Data:**

1. **Access Restriction:**
   * Implement role-based access control (RBAC) to restrict access to certain features or data based on user roles. For example, team leaders might have access to additional administrative functions compared to team members.
   * Ensure that sensitive operations, such as task assignment or user management, are only accessible to authorized team Leaders.
2. **Encryption:**
   * Utilize strong encryption algorithms (e.g., 128-bit AES) to encrypt sensitive data such as passwords before storing them in the database.
   * Apply encryption techniques to protect communication between the client and server, especially when transmitting sensitive information like login credentials or task details.
3. **Session Management:**
   * Implement secure session management techniques to prevent session hijacking and unauthorized access.
   * Utilize techniques such as session tokens, HTTPS protocol, and secure cookies to ensure the integrity and confidentiality of user sessions.
   * Implement session timeout mechanisms to automatically log out users after a period of inactivity, reducing the risk of unauthorized access.
4. **Data Integrity:**
   * Implement measures to maintain data integrity, such as data validation and sanitization, to prevent injection attacks (e.g., SQL injection, XSS).
   * Regularly monitor and audit access logs to detect and respond to any unauthorized access attempts or suspicious activities.

**Mapping of Functional Requirements to Data Storage:**

1. **Task Management:**
   * **Tasks, Task Assignments:** Store task data including descriptions, priorities, deadlines, and status in the Tasks table. Task assignments will link tasks to users, facilitating assignment and tracking.
   * **Users:** Store user data including usernames and roles to manage task assignment permissions.
2. **Collaboration:**
   * **User Data:** User data will include information about team members, enabling collaboration features such as task assignment and team communication.
   * **Task Assignments:** Task assignments will link tasks to users, facilitating collaboration by assigning tasks to team members.
3. **Security:**
   * **User Authentication Data:** Store user authentication data (e.g., usernames, encrypted passwords) in the Users table to authenticate users securely.
   * **Session Logs:** Store session logs containing user activity information (e.g., login/logout timestamps, actions performed) for security auditing purposes.

**ER Diagram**



**List of Proposed Prototypes**

**List of Proposed Prototypes Iteration 2**

This document describes the taskmaster application prototypes that will be developed before the completion of the semester.

**1. TaskMaster Authentication Prototype**

**Description:** TaskMaster Authentication focuses on providing secure user registration and authentication functionalities for accessing the TaskMaster web application.

**Features:**

* User-Friendly Registration: Easy sign-up form with validation checks.
* Secure Login: Robust authentication for protected access.

**2. TaskMaster UI Prototype**

**Description:** TaskMaster UI focuses on delivering a user-friendly interface for effective task management.

**Features:**

* User friendly interface for easy navigation.
* Clear task lists, filters, and search functionality.
* Calendar view for task deadlines.
* Seamless experience on web browsers.

**3. Task Prioritization Prototype**

**Description:** This prototype allows users to prioritize tasks based on user-defined criteria to ensure efficient task management.

**Features:**

* Users can assign priority levels (e.g., high, medium, low) to tasks based on their importance and urgency.
* Users can set priorities based on the task deadlines and priority.

**4. Secure Data Storage Prototype**

**Description:** This prototype ensures that task data is stored securely, will be maintained with confidentiality.

**Features:**

* Task data is encrypted to protect sensitive information from unauthorized access.
* Steps are taken to ensure that task data stays accurate and protected from attacks.

**5. Task Creation and Assignment Prototype**

**Description:** This prototype enables users to create tasks and assign them to specific individuals as per the requirement.

**Features:**

* Users can easily create new tasks with relevant details such as title, description, and due date.
* Tasks can be assigned to specific users or teams.

**6. Calendar View Prototype**

**Description:** This prototype displays tasks and their deadlines in a calendar view.

**Features:**

* Tasks are visually represented on a calendar, providing a clear overview of upcoming tasks and deadlines.
* Users can see tasks alongside their deadlines on a calendar, making it easy to plan and schedule them.

**7. TaskMaster Progress Tracking Prototype**

**Description:** This prototype allows users to track the progress of tasks and monitor their completion status.

**Features:**

* Users can mark tasks as 'in progress,' 'completed,' or 'pending' to track their status.
* Progress indicators visually represent the completion status of tasks, helping users stay organized.
* Team members can view each other's progress on shared tasks, facilitating collaboration and accountability.

**8.Task Management Dashboard prototype**

**Description:** The Task Management Dashboard serves as the central hub for users to oversee and manage all tasks within the Task Master web app. It provides a comprehensive overview of task-related information, enabling users to efficiently track progress, prioritize tasks, and allocate resources as needed.

**Features:**

* A visually organized display of all tasks, presenting key details such as status, priority, due dates, and assigned users in a clear and concise manner.
* Options to customize the dashboard view based on user preferences, allowing for personalized layouts and arrangements of task cards.

Functionality to filter tasks based on criteria such as status, priority, due dates, and assigned users, facilitating targeted navigation and focused task management.

**Meeting-Minutes**

**Link:** [**https://github.com/tejakumarsai6309/44691-03-GDP-team1/wiki/Meeting-Minutes**](https://github.com/tejakumarsai6309/44691-03-GDP-team1/wiki/Meeting-Minutes)

**Client Meeting Minutes**

**1. Who was present?**

* Harichaithanya Kotapati (S567067)
* Teja Kumar Muppala (S565960)
* Manichandra Kuntala (S565945)
* LaxmiNarayana Yadav Pakanati (S564223)
* Vaishnavi Inturi (S567121)
* Dr. Mark Chai (Client)

**2. Meeting agenda**

* Team introduction and their roles and responsibilities in the development of the project.
* Assignment of their roles towards the implementation and documentation of the project.
* Discussing about the technology stack
* Analyzing the project requirements (Both functional and non functional).
* Scheduling the presentations
* Developing the project charter
* Identify the Platform in which the application needs to be developed.

**3. Specific questions asked (and who asked them)**

* **Harichaithanya Kotapati**
  + What are some key deliverables we should focus on initially?
  + How can we ensure effective communication among team members?
  + What documentation strategies should we implement to track progress and changes?
  + What database system would best suit the project's needs at this early stage?
* **Teja Kumar Muppala**
  + What technologies do you recommend for the backend development considering the scalability and flexibility of the system?
  + What are the primary functionalities of the backend system that we need to prioritize for development?
  + How should we design the architecture?
* **Manichandra Kuntala**
  + What database system would best suitable for the project?
  + How can we ensure data integrity and security within the database system?
  + How do we model the database schema?
* **LaxmiNarayana Yadav Pakanati**
  + What are some key features or modules we should prioritize for development in the early stages?
  + How can we maintain code quality and ensure easy collaboration among developers?
  + What testing strategies should we employ to validate the functionality of the initial features?
* **Vaishnavi Inturi**
  + What types of documentation are essential for this early stage of the project?
  + What tools or platforms do we use for managing project documentation?
  + How can we make the documentation easily accessible and understandable for all team members?

**4. Specific answers given**

Specific answers given Prof Dr.Mark Chai (Client) asked several questions regarding our project they are as follows :

* What are the key functionalities that need to be implemented in the Task Master application?
  + The key functionalities include user registration and authentication, task creation, assignment, and tracking, task prioritization, deadline management.
* Is this a web based or mobile based application?
  + This is a web based application
* What are the technologies that you have chosen to implement this project?
  + The technologies we chosen are Java's Spring framework for backend development and for frontend we use HTML,CSS, Bootstrap and for database we use MySQL database.
* How do you implement User authentication and security?
  + We implement user authentication and security using the Spring security framework.

**5. Action items: what have you decided you will show the client in your next meeting**

* Developing the Project Charter
* Development of the project plan
* Defining the scope of the project
* Conduct a detailed analysis regarding the project documentation.

**6. When is the next meeting**

* The next client meeting is scheduled on 05/15/2024.