

# **vSarthi Hackathon**

## **Task-Master(To-Do App)**

Presented by-

- Shubham Garg
- Jitender Sharma
- Davik Chaudhary
- Vaibhav Kumar Mishra

# Contents

- Overview
- User flow mockup
- Database Schema

# Overview

Here's an overview of the Taskmaster app:

## 1. Purpose:

The Taskmaster app is designed to help users manage their tasks and organize them into different boards. Users can create, update, and delete tasks within boards, move tasks between different stages (such as "To Do", "In Progress", "Completed"), and view their tasks in a visually organized manner.

## 2. Key Features:

- **User Authentication:** Users can sign up, log in, and log out of the app.
- **Board Management:** Users can create multiple boards to categorize their tasks based on different projects, categories, or any other criteria.
- **Task Management:**
  - Create, update, and delete tasks within each board.
  - Tasks are organized into columns representing different stages of completion (e.g., "To Do", "In Progress", "Completed").
  - Users can move tasks between columns to track their progress.
- **Board Sharing:** Option to share boards with other users for collaboration.
- **Responsive Design:** The app is designed to work well on various devices, including desktops, tablets, and smartphones.

## 3. Architecture:

- **Backend:**
  - Developed using Node.js with Express.js for the server.
  - MongoDB database to store user information, boards, and tasks.
  - Mongoose ORM for modeling and interacting with MongoDB.
  - User authentication using JWT (JSON Web Tokens).
  - RESTful API endpoints for user management, board operations, and task management.
- **Frontend:**
  - Developed using React.js for the user interface.
  - Components for different views such as login/signup, board creation, task lists, task creation/editing, etc.
  - Axios for making HTTP requests to the backend API.
  - React Router for navigation between different pages.

#### 4. Workflow

##### - **User Registration and Authentication:**

- New users can sign up for an account with a unique username and password.
- Existing users can log in using their credentials.
- Authentication is handled using JWT tokens, providing a secure way to authenticate requests.

##### - **Board Management:**

- Users can create new boards with unique names.
- Each board can have multiple columns (e.g., "To Do", "In Progress", "Completed").
- Boards are stored in the database with references to the tasks they contain.

##### - **Task Management:**

- Users can add, edit, and delete tasks within each board.
- Tasks have properties such as title, description, priority, due date, etc.
- Tasks are moved between columns to represent their progress.

##### - **Sharing Boards** (Future Enhancement):

- A feature to share boards with other users for collaborative task management.

#### 5. Data Models:

- **User Model:** Contains user details such as username, email, password, and an array of board IDs.

- **Board Model:** Represents a board with a name and an array of task IDs.

- **Task Model:** Represents a task with properties such as title, description, priority, due date, status, etc.

#### 6. API Endpoints:

##### - **User Management:**

- `POST /register`: Register a new user.
- `POST /login`: Log in an existing user.
- `POST /logout`: Log out the current user.

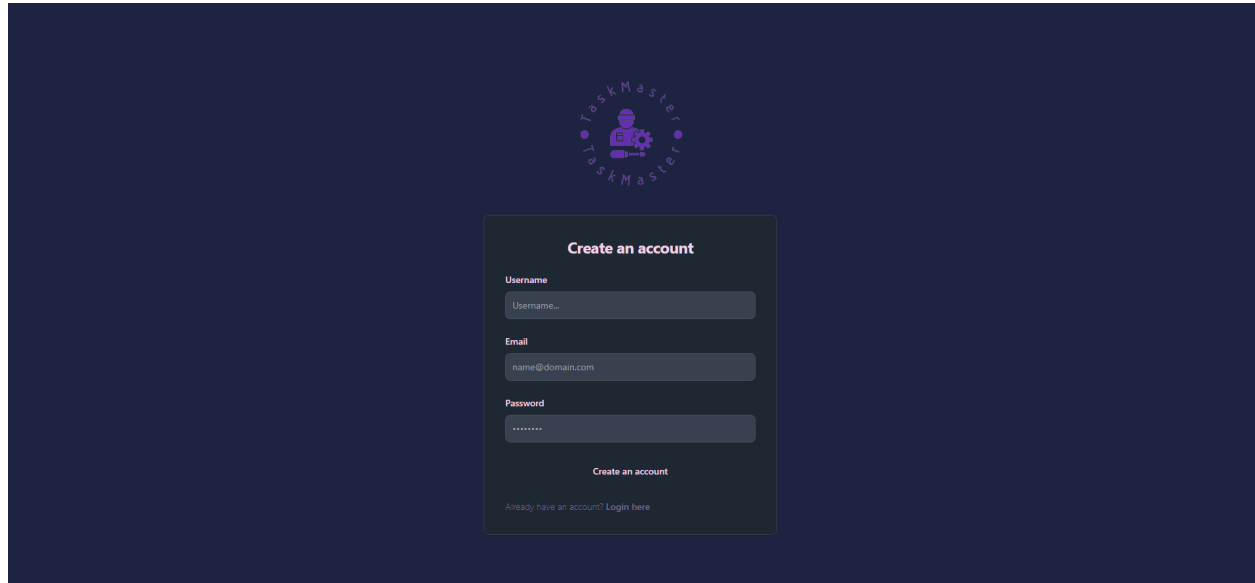
##### - **Board Operations:**

This API defines routes for managing user boards. GET /user/:userId/boards retrieves all boards belonging to a user. POST /user/:userId/boards creates a new board for the specified user. GET /user/:userId/board/:boardId retrieves a specific board by its ID, while GET /user/:userId/board/:boardName retrieves a board by its name. PUT /user/:userId/board/:boardId updates the details of a board identified by its ID. Finally, DELETE /user/:userId/board/:boardId deletes a board and its associated tasks.

## UserFlow Mockup

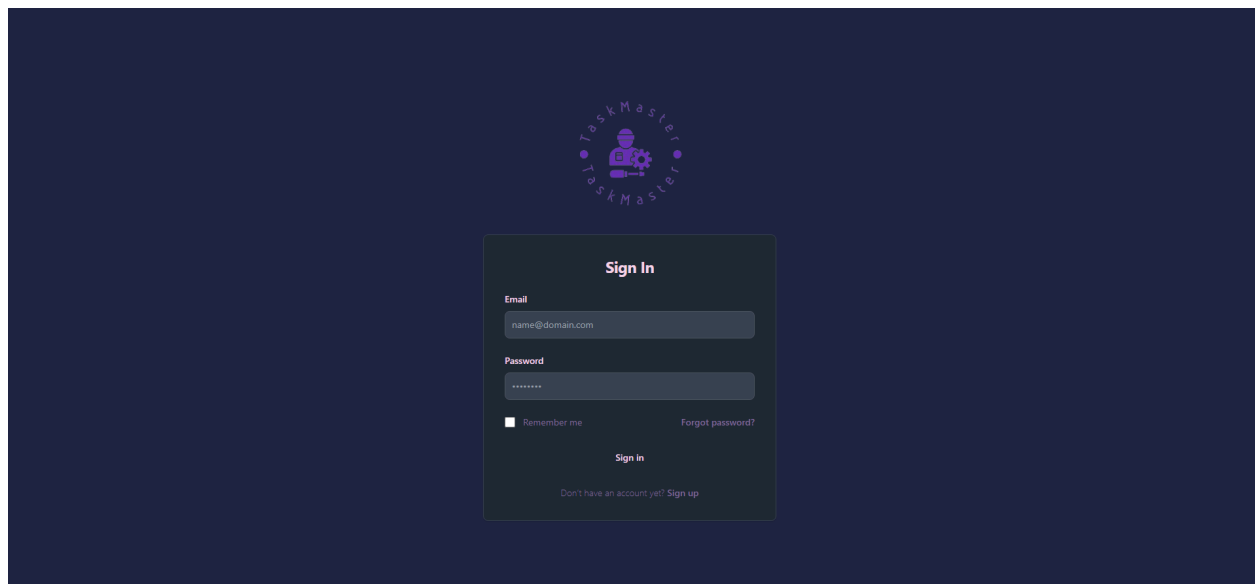
The entire userFlow can be understood by following:-

### 1) Sign-up page



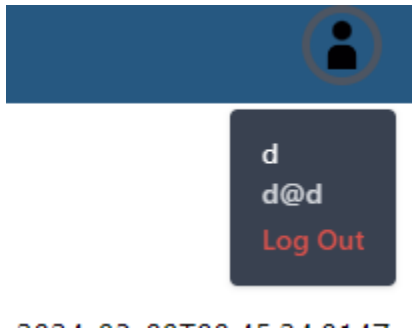
The sign-up page features a dark blue background. At the top center is the TaskMaster logo, which consists of a circular arrangement of dots with a central icon of a person holding a gear. Below the logo is a white card with the title "Create an account". The card contains three input fields: "Username" with a placeholder "Username...", "Email" with a placeholder "name@domain.com", and "Password" with a placeholder ".....". Below these fields is a "Create an account" button. At the bottom of the card is a link that says "Already have an account? Login here".

### 2) Login Page

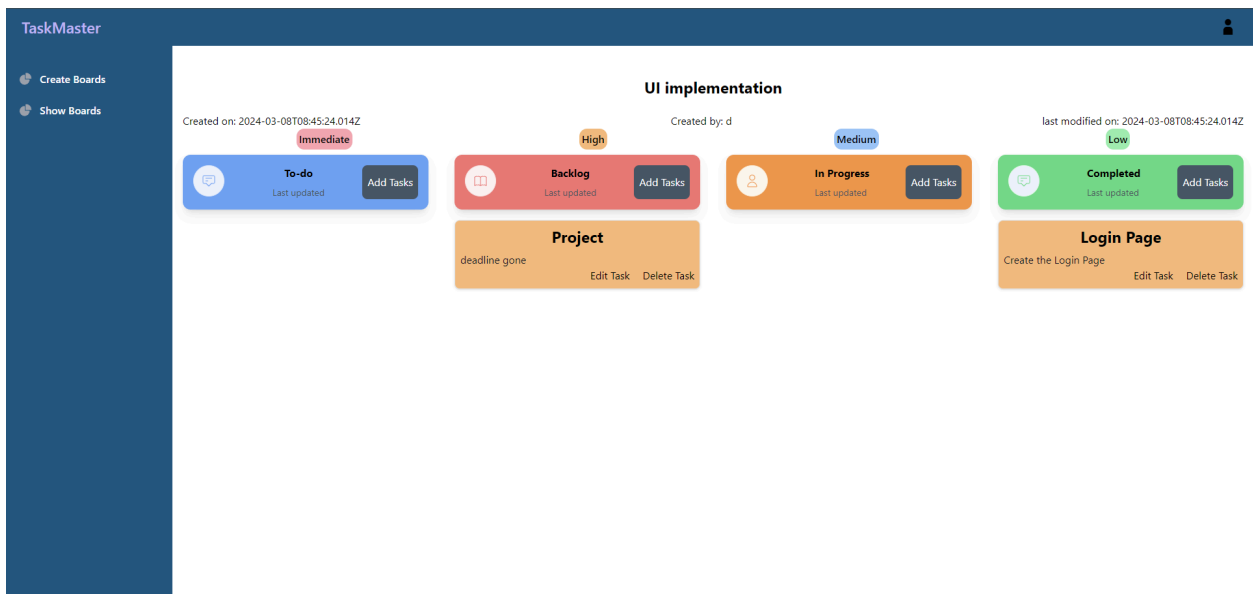


The login page features a dark blue background. At the top center is the TaskMaster logo, which consists of a circular arrangement of dots with a central icon of a person holding a gear. Below the logo is a white card with the title "Sign In". The card contains two input fields: "Email" with a placeholder "name@domain.com" and "Password" with a placeholder ".....". Below these fields are two links: "Remember me" with a checkbox and "Forgot password?". Below these links is a "Sign in" button. At the bottom of the card is a link that says "Don't have an account yet? Sign up".

### 3) User Credentials (as seen on UI side)



### 4) Dashboard



## 5) Responsivity

TaskMaster

UI implementation

Created on: 2024-03-08T08:45:24.014Z by: d last modified on: 2024-03-08T08:45:24.014Z

Immediate

High

Medium

Low

Create New Task

×

Task Name

Login Page

Priority

Immediate

▼

Task Description

Create the Login Page

Add new Task



## UI implementation

Created on: 2024-03-08T08:45:24.014Z by: d last modified on: 2024-03-08T08:45:24.014Z

Immediate

High

Medium

Low



### To-do

Last updated

Add Tasks

## Login Page

Create the Login Page

Edit Task

Delete Task



### Backlog

Last updated

Add Tasks



### In Progress

Last updated

Add Tasks



### Completed

Last updated

Add Tasks



# Database Schema

Database: MongoDB

Collections:

Board:

- *Fields:*
  - **name (String)**: Name of the board.
  - **createdBy (String)**: Username of the user who created the board.
  - **collaborators (Array of Strings)**: List of usernames of members associated with the board.
  - **createdAt (Date)**: Timestamp of when the board was created.
  - **lastModifiedAt (Date)**: Timestamp of when the board was last modified.
  - **columns (ObjectId)**: Consists of todo, backlog, inProgress and completed which are of types ObjectId.

```
const boardSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    unique: true
  },
  createdBy: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User', // Reference to user document
    required: true,
  },
  collaborators: [{
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User', // Reference to user document
  }],
  createdAt: {
    type: Date,
    default: Date.now,
  },
  lastModifiedAt: {
    type: Date,
    default: Date.now,
  },
  columns: {
    todo: [{
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Task',
    }],
  },
});
```

```

    backlog: [{
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Task',
    }],
    inProgress: [{
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Task',
    }],
    completed: [{
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Task',
    }],
  },
],
},
{
  collection: "Board",
}
);

const Board = mongoose.model('Board', boardSchema);

module.exports = Board;

```

Task:

- *Fields:*
  - **title (String):** Title of the task.
  - **description (String):** Description of the task.
  - **priority (String):** Priority of the task (e.g., "Immediate", "High", "Medium", "Low").
  - **status (String):** Status of the task (e.g., "To Do", "In Progress", "Completed").
  - **assignee (String):** defines a reference to a user document in MongoDB using Mongoose.
  - **createdAt (Date):** Timestamp of when the task was created.
  - **updatedAt (Date):** Timestamp of when the task was last updated.

```

const mongoose = require('mongoose');
const taskSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  description: {
    type: String,
    required: true,
  },
  priority: {
    type: String,
    enum: ['Immediate', 'High', 'Medium', 'Low'],
    required: true,
  },
  status: {
    type: String,
    enum: ['todo', 'backlog', 'inProgress', 'completed'],
    default: 'todo',
  },
  assignee: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User', // Reference to user document
  },
  createdAt: {
    type: Date,
    default: Date.now,
  },
});

```

```

    updatedAt: {
      type: Date,
      default: Date.now,
    }
  ],
  {
    collection: "Task",
  }
);

const Task = mongoose.model('Task', taskSchema);

module.exports = Task;

```

## UserInfo:

- *Fields:*
  - **username (String)**: Username of the user.
  - **email (String)**: Email address of the user.
  - **password(String)**: password of the user.
  - **boards**: It represents an array of references to board documents in MongoDB

```
const mongoose = require("mongoose");

const UserDetailsSchema = new mongoose.Schema(
  {
    username: {type: String, unique: true},
    email: {type: String, unique: true},
    password: String,
    boards: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Board' }],
  },
  {
    collection: "UserInfo",
  }
);

mongoose.model("UserInfo", UserDetailsSchema);
```