
رواد مصر الرقمية


وزارة الاتصالات
وتكنولوجيا المعلومات


LEARNING
SOLUTIONS

Task Management System

React Frontend Track

DEPI Round 2

Eng. Ashraf Sadek

2025 April 15th

Mohamed Karim
Raymond Maher
Ahmed Makbool
Nour Eldin
Peter Atef

1. Project Planning & Management

Project Proposal

Description

The proposed software is a task management tool website designed for both firms and individuals to efficient task organization and collaboration. It enables users to create, assign, and monitor tasks efficiently while ensuring deadlines and priorities are met. The system allows task status updates to enhance productivity. Users can categorize tasks, set due dates, and assign roles to contributors, making project management more structured. The platform aims to minimize manual workload and paperwork and enhance teamwork by offering an online workspace for task coordination. This website is ideal for businesses and individuals looking to optimize their workflow.

Objective

The objective of this software is to provide an intuitive and efficient task management platform for both firms and individuals, streamlining task organization and collaboration. It aims to enhance productivity by enabling users to create, assign, and track tasks seamlessly while ensuring deadlines and priorities are effectively managed. The system is designed to facilitate real-time task updates, structured project management, and role-based task delegation, reducing manual effort and paperwork. By offering a centralized online workspace, the platform seeks to optimize workflow efficiency, improve teamwork, and support seamless task coordination for businesses and individuals alike.

Scope

The task management platform is designed to cater to both individuals and businesses, offering a structured approach to task organization, assignment, and tracking. It supports task categorization, priority setting, due date management, and role-based task delegation to ensure seamless collaboration. The system includes real-time status updates, notifications, and progress tracking, enhancing team coordination and productivity.

Users can create task lists, set reminders. The platform supports multiple views, such as Kanban boards, list views, and calendar-based task management, allowing users to work in their preferred style. Additionally, it includes access controls to manage user roles and permissions, ensuring secure and efficient task distribution.

Designed for scalability, the software can accommodate teams of varying sizes, from freelancers managing personal projects to enterprises handling complex workflows. By minimizing manual workload and paperwork, the platform optimizes efficiency and ensures that deadlines and project goals are met effectively.

Project Plan

Milestones

Week	Tasks
1	• Setup Development Environment (React, NodeJS, MongoDB, Express) • Create Wireframe (Desing website layout using Figma) • Develop Layout (Develop Basic layout with React and CSS) • Document The Work
2	• Implement Task Creation, Update, Deletion • Backend Integration: NodeJS, MongoDB, Express • Integrate Redux (Manage Global State of tasks across component) • Document The Work
3	• Implement Task Filtering (Task status, priority) • Add Search Functionality (Search for tasks) • Test Core Feature (Develop Test Cases, Run Manual Test) • Document The Work
4	• Enhance UI (Improve UI using CSS) • Deploy Website to cloud platform (Netlify) • Complete Documentation (i.e. How to use system, architecture)

Deliverables

Week	Deliverables
1	• Project Setup with React and backend API (NodeJS) • Wireframe for the Entire Application • Basic Layout (React and CSS)
2	• Functional task (Create, Update, Delete) • Integration with Backend and MongoDB • Integration with Redux
3	• Task Filtering and Priority • Search Functionality • Fully Functional Task Management System
4	• Final Responsive UI • Deployed Task Management System • Complete Project Documentation

Task Assignment & Roles

Name	Contribution
Mohamed Karim	Project Plan (Milestones, Deliverables) Use Case Diagram, Database Design & Modeling (ERD, RD)
Ahmed Makbool	Risk Assessment & Mitigation Plan Technology Stack Deployment Plan UI/UX Design
Raymond Maher	Intro (Description, Objective, Scope), System Analysis & Design Problem Statement & Objectives
Nour Eldin	Requirement Gathering (User Stories & Use Case, Functional Requirements, Non-Functional Requirement) System Analysis & Design (Functional Requirements & Non-Functional Requirement)
Peter Atef	Requirements Gathering (Stakeholder Analysis), System Analysis & Design (Software Architecture),UI/UX Design & Prototyping (Wireframes & Mockups)

Risk Assessment & Mitigation Plan

In the development of the Task Manager web application, we encountered a few challenges that could have impacted on the project's timeline or quality. Below is a summary of the risks faced during development, their potential impact, and the solutions implemented to mitigate them.

1. Difficulty with UI Design and Layout
    - **Problem:** During the development, we initially struggled with designing the user interface, especially for complex components like the task list and the task detail's view. The layout was not responsive across different screen sizes, which led to inconsistent user experiences on mobile and desktop devices.
    - **Impact:** This issue could have resulted in poor user experience, particularly for users accessing the app from mobile devices, and could have delayed the project delivery.
    - **Solution:** We addressed the problem by researching and experimenting with CSS Grid and Flexbox for better responsiveness. Additionally, we consulted the project instructor to get feedback and guidance on best practices for building responsive layouts using CSS. Their suggestions helped us optimize the layout and ensure it functioned seamlessly across different screen sizes.
    - **Mitigation Plan:** Going forward, we will integrate a design system to maintain consistency and improve the overall design workflow. We will also use media queries for better control over responsive elements.

2. State Management Complexity
    - **Problem:** Reacts state management became challenging as the application grew in complexity, especially when managing the tasks' status and priority. Updating the state in multiple places led to synchronization issues where the UI did not always reflect the changes made to the tasks in the backend.
    - **Impact:** This led to bugs where users couldn't see their changes (e.g., task statuses or priorities) in real time, which would decrease the usability and reliability of the application.
    - **Solution:** We implemented React Context API for centralized state management, which allowed us to maintain a single source of truth for tasks across the app. This approach also simplified prop drilling and made it easier to manage complex state updates.
    - **Mitigation Plan:** In future development, we plan to incorporate more advanced state management libraries like Redux or Recoil if the application's complexity increases further. We will also use tools like React Dev Tools to debug and visualize state changes in real-time.

3. Data Persistence and API Integration
    - **Problem:** Initially, the task data was not being persisted after the page was refreshed. This was especially problematic for users who wanted to save their tasks for later. We also faced challenges integrating with the API to fetch and update task data.
    - **Impact:** Without persistence, users would lose their tasks upon reloading the page, which could cause frustration and result in poor user experience. Inconsistent data syncing with the backend would lead to outdated or missing tasks.
    - **Solution:** To address this, we integrated local Storage to persist data locally in the user's browser, which allowed tasks to remain visible even after a page reload. We also sought assistance from the instructor to ensure our API requests were correctly formatted and to troubleshoot errors when fetching and posting data.
    - **Mitigation Plan:** Moving forward, we plan to implement a more robust persistence solution using a backend database (e.g., MongoDB) with proper API endpoints. This will ensure data is reliably stored and synced, even in case of page refreshes or errors.

4. Testing and Debugging
    - **Problem:** During testing, we encountered issues with test reliability. Some of our unit tests failed intermittently, making it difficult to identify specific bugs. Additionally, our test coverage was not as comprehensive as we would have liked.
    - **Impact:** Inadequate testing could have resulted in undetected bugs making their way into production, leading to an unstable and unreliable app for users.
    - **Solution:** We dedicated time to properly configuring the Jest and React Testing Library for unit testing and integration testing. After consulting the instructor, we learned about techniques like mocking API requests and using snapshot testing for React components. This helped stabilize our tests and ensured higher test coverage.
    - **Mitigation Plan:** We will continuously run tests throughout the development cycle to catch errors early. In addition, we will set up CI/CD pipelines that automatically run tests before deploying new changes. This will ensure the stability of the app with every release.

5. Version Control and Collaboration Issues
    - **Problem:** During team collaboration, we faced conflicts in version control when multiple people worked on the same files simultaneously. This led to merged conflicts and slowed down development.
    - **Impact:** Merge conflicts can cause delays and inconsistencies, particularly when integrating features developed by different team members.
    - **Solution**: To mitigate this risk, we set up GitHub as our branching model, where each team member worked on separate feature branches. We also scheduled regular stand-up meetings to synchronize progress and avoid conflicts. Additionally, the instructor advised us on best practices for managing pull requests and resolving conflicts efficiently.
    - **Mitigation Plan:** We will continue using GitHub for future projects and make sure to review pull requests more frequently to catch potential conflicts early. We will also adopt automated code merging tools to reduce the risk of conflicts.

2. Requirements Gathering

Stakeholder Analysis

Our clients for now are Dr. Ashraf and DEPI organization. Clients using our task management system need an easy-to-use platform that helps them organize, track, and prioritize tasks. They require real-time updates and notifications to stay on top of deadlines and changes, with a focus on collaboration among team members. The system should be simple to navigate, mobile-friendly, and customizable to fit different workflows. Security is important, with control over who can access tasks and data. Clients also benefit from integrations with other tools they use and from features like task prioritization, progress tracking, and reporting to improve productivity and project management.

User Stories & Use Case

1. User Registration and Authentication
    - As a new user, I want to sign up by email and password so that I can access the system.
    - As a registered user, I want to log in securely so that I can access my tasks.

2. Workspace and Boards Management
    - As a user, I want to create a board within a workspace so that I can manage tasks for different projects.
    - As a user, I want to rename or delete a board so that I can keep my workspace organized.

3. Task Management (Cards)
    - As a user, I want to create a task (card) so that I can track work items.
    - As a user, I want to set due dates on tasks so that I can track deadlines.
    - As a user, I want to move tasks between lists (e.g., To Do, In Progress, Done) so that I can update their status.
    - As a user, I want to comment on tasks so that I can collaborate with my team.
    - As a user, I want to delete completed tasks so that I can keep my board clean.

4. Reporting and Insights
    - As a user, I want to filter and search for tasks so that I can quickly find relevant items.

5. System Settings & Customization
    - As a user, I want to customize board backgrounds so that I can personalize my workspace.

Use Cases

1. User Registration & Login
   Actors: New User, System
   Precondition: User is not registered.
   Main Flow:
    1. The user navigates to the signup page.
    2. The user enters their email, password, and name.
    3. The system validates the input and creates an account.
    4. The user logs in using email and password.

   Alternative Flow:
    - If the email is already in use, the system prompts the user to log in instead.

2. Creating a Task
   Actors: Registered User
   Precondition: The user is logged in and has a board.
   Main Flow:
    1. The user selects a board.
    2. The user clicks the "Add Task" button.
    3. The user enters a task title and optional description.
    4. The user sets the due date.
    5. The user saves the task, and it appears on the board.

   Alternative Flow:
    - If the user forgets to enter a title, the system does not allow submission.

3. Moving a Task Between Lists
   Actors: Registered User
   Precondition: A task exists on the board.
   Main Flow:
    1. The user drags a task from one list to another.
    2. The system updates the task status accordingly.

   Alternative Flow:
    - If the user drops the task at an invalid location, the system prevents the action.

4. Adding a Comment to a Task
   Actors: Registered User
   Precondition: A task exists.
   Main Flow:
   1. The user opens a task.
   2. The user enters a comment and submits it.
   3. The system displays the comment.

   Alternative Flow:
   - If the user enters an empty comment, the system does not allow submission.

5. Deleting a Task
   Actors: Registered User
   Precondition: The user is logged in and has permission to delete tasks.
   Main Flow:
   1. The user selects a task.
   2. The user clicks the delete button.
   3. The system asks for confirmation.
   4. The user confirms deletion.
   5. The system removes the task from the board.

   Alternative Flow:
   - If the user cancels, the task remains unchanged.

Functional Requirements

User Management

- The system allows users to sign up using email and passwords.
- The system allows users to log in securely.

Workspace & Board Management

- The system shall allow users to create, rename, and delete boards.

Task (Card) Management

- The system shall allow users to create, edit, move, and delete tasks.
- The system shall allow users to assign tasks to specific team members.
- The system shall allow users to set due dates for tasks.
- The system shall allow users to add task descriptions.
- The system shall allow users to drag and drop tasks between lists.

Collaboration & Communication

- The system shall allow users to comment on tasks.

Search & Filtering

- The system shall allow users to search for tasks by title, description, or assigned user.
- The system shall allow users to filter tasks based on labels, due dates, or assignment.

Non-Functional Requirements

Performance Requirements

- The system shall be able to handle at least 1,000 concurrent users without performance degradation.
- The system shall respond to user actions within 1 second for 95% of requests.

Scalability Requirements

- The system shall be designed to scale horizontally to support an increasing number of users.
- The system shall support cloud-based storage for attachments and images.

Security Requirements

- The system shall encrypt user passwords using industry standards (Argon2).
- The system shall use HTTPS for all communications.

Availability & Reliability

- The system shall have 99.9% uptime to ensure continuous access.
- The system shall provide automatic data backups every 24 hours.

3. System Analysis & Design

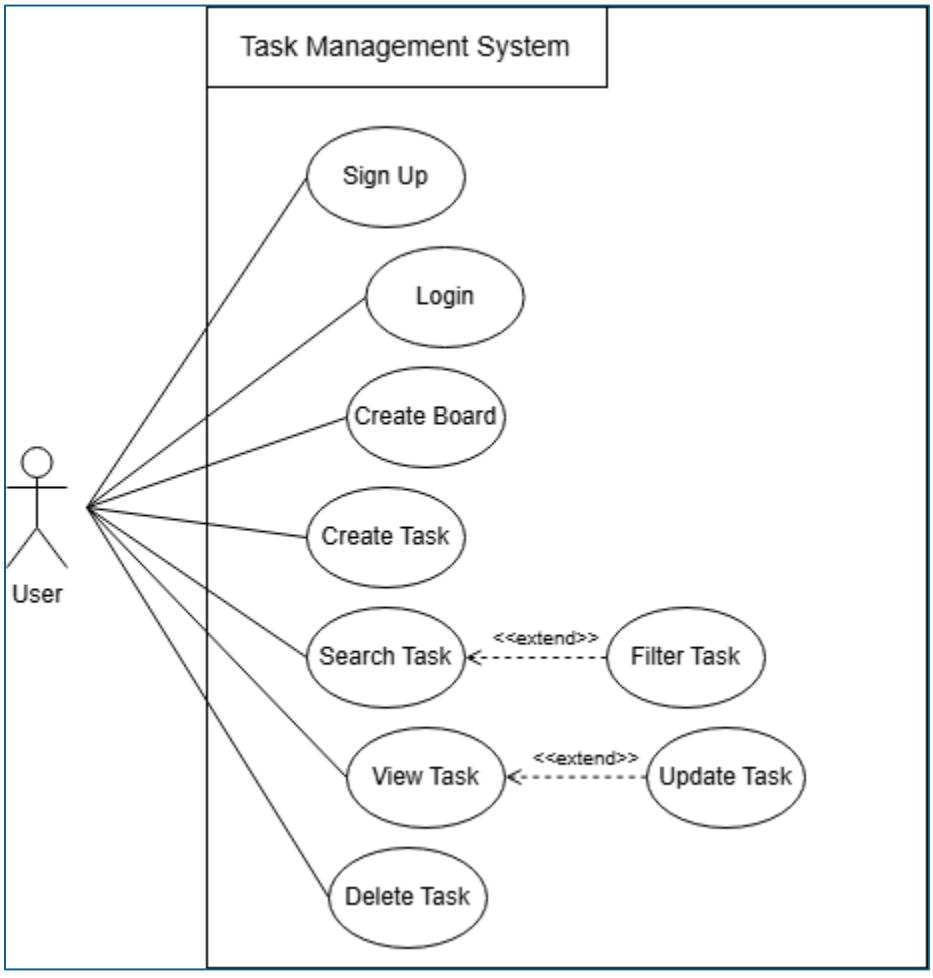
Problem Statement & Objectives

Managing tasks efficiently is a challenge for both individuals and businesses, often leading to missed deadlines, poor collaboration, and disorganized workflows. Traditional methods, such as manual tracking and scattered tools, lack real-time updates and structured task delegation.

Objectives

- Develop a centralized task management platform for seamless task organization and collaboration.
- Enable task creation, assignment, categorization, and prioritization.
- Implement real-time status updates, reminders.
- Provide multiple task views (Kanban, list, calendar) for user flexibility.
- Minimize manual workload, enhance productivity, and improve workflow efficiency.

Use Case Diagram



Function & Non-Functional Requirements

The task management system will allow users to sign up, log in. Users can create, rename, and delete boards, and manage tasks by adding descriptions, due dates, and assignments. Collaboration features include comments. Users can search and filter tasks. The system will be highly performant, supporting 1,000+ concurrent users with fast response times. It will be scalable. Security measures include encrypted passwords, HTTPS. The system will ensure 99.9% uptime and provide a responsive UI for desktop and mobile users. Additionally, it will comply with GDPR and ISO 27001 security standards while offering customization options.

Software Architecture

For our task management system, we implemented a Single Page Application (SPA) to enhance the user experience by ensuring smoother and faster interactions. The core idea behind using SPA is that once the initial page is loaded, the application doesn't reload the entire page every time the user interacts with it. Instead, only the required data is fetched and dynamically rendered, making the system feel more responsive and fluid.

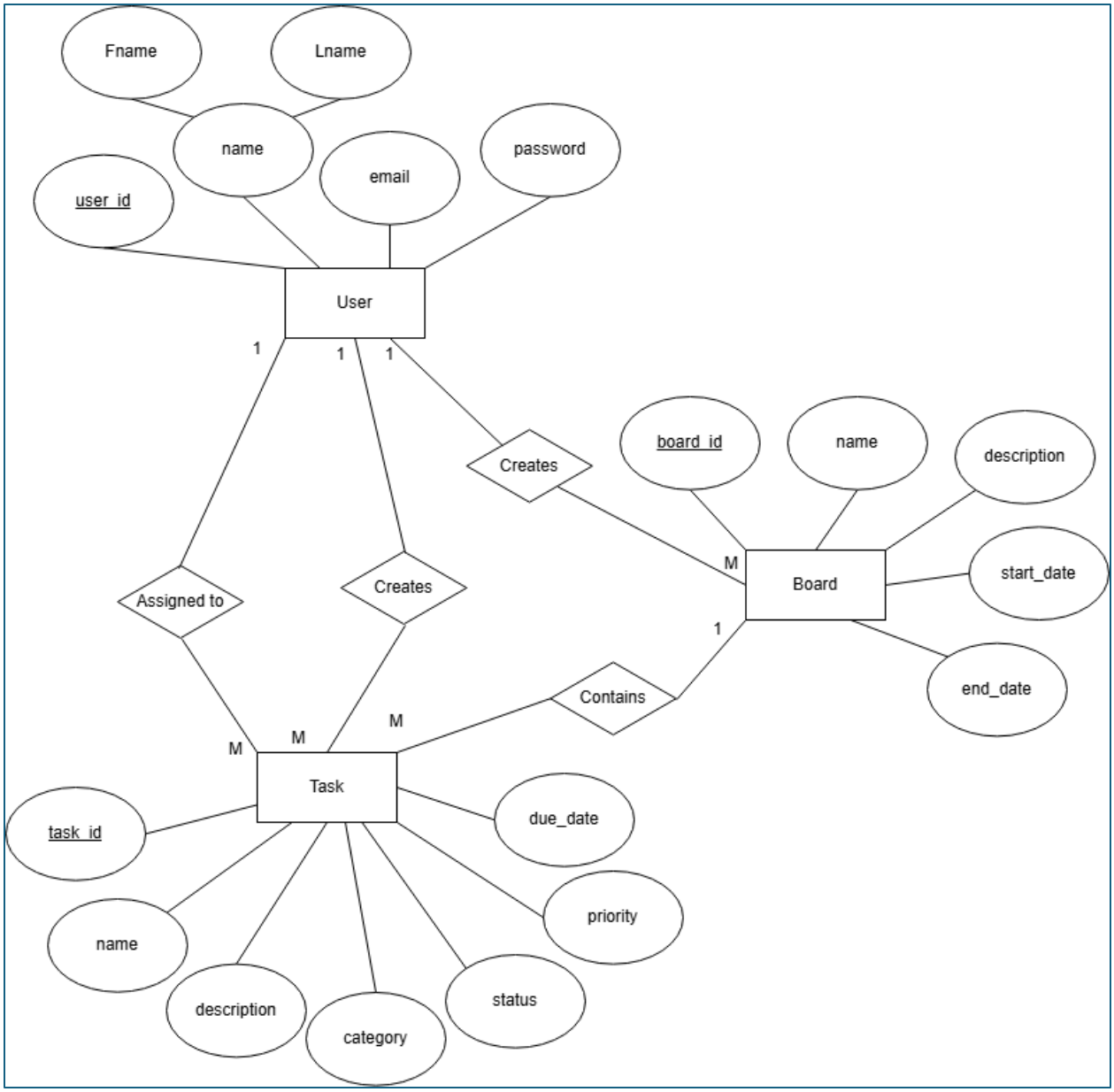
In our system, users can add, edit, or delete tasks in real-time. For example, when a user updates a task, the changes are reflected instantly without the need for a full page refresh. This creates a seamless experience as users don't need to wait for the page to reload.

Another feature we integrated is real-time status updates. When users move tasks between different stages (such as "To Do," "In Progress," and "Completed"), the interface updates dynamically to show the new status, providing immediate feedback without delays.

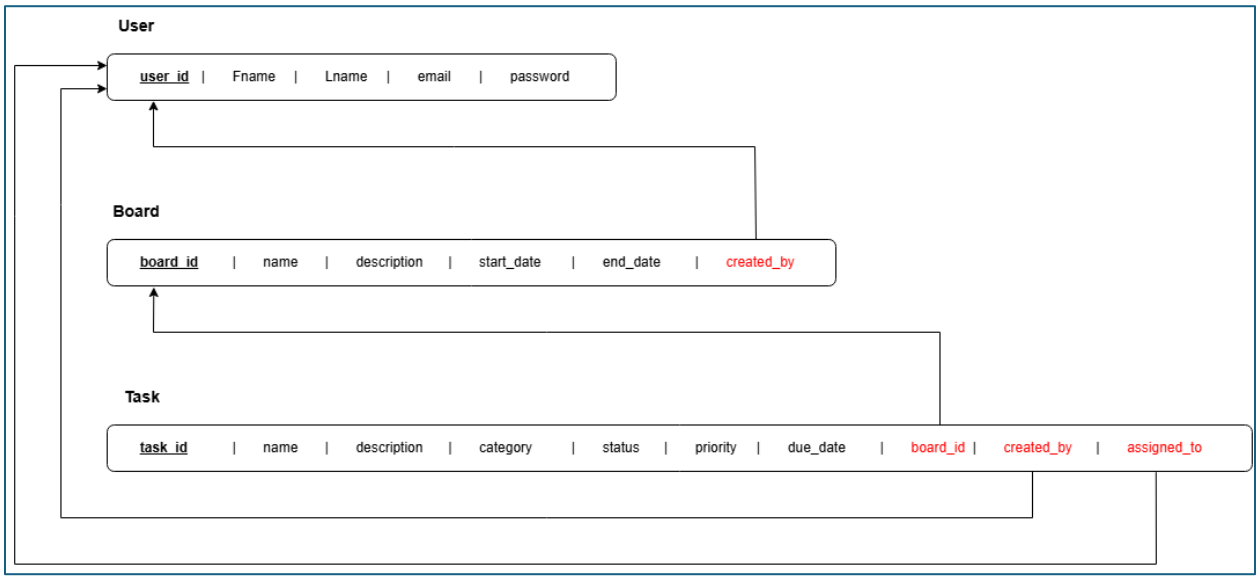
The system also supports real-time filtering and sorting of tasks. Users can filter by deadlines, priorities, or other criteria, and the task list updates instantly. For collaborative environments, we enabled real-time updates using technologies like WebSocket, ensuring that when multiple users are interacting with the task list, all changes are reflected in real time for everyone involved. This creates a more interactive and collaborative workspace. Additionally, notifications and alerts about upcoming deadlines or overdue tasks are pushed directly to the user without requiring a page reload, keeping everyone on track and informed.
Overall, by leveraging SPA technology, our task management system provides a more efficient, scalable, and engaging user experience with faster performance and real-time collaboration.

Database Design & Data Modeling

Entity Relationship Diagram



Relational Database



UI/UX Design & Prototyping

Wireframes & Mockups

UI/UX Guidelines

The UI/UX (User Interface and User Experience) design of the Task Manager web application plays a crucial role in ensuring that users can easily manage their tasks while interacting with a visually appealing and functional interface. Below is an overview of the UI/UX design decisions and strategies used in the development of the app.

UI Design Principles

The Task Manager app was designed with the following principles in mind to ensure a clean, user-friendly interface:

Simplicity

- The interface is designed to be minimalistic, with clear and easy-to-understand components.
- Key actions such as creating, editing, and deleting tasks are prominently displayed and accessible, with minimal steps required from the user.

Consistency

- The app follows consistent design patterns and components across all pages (e.g., task lists, task details, and user settings).
- Consistency in typography, button styles, and colors ensures a seamless user experience, reducing cognitive load on the user.

Accessibility

- Color contrast ratios were checked to ensure readability for users with visual impairments.
- Elements like buttons, icons, and forms are sized appropriately to be clickable on both desktop and mobile devices.
- The app follows accessibility best practices, such as focus management and keyboard navigation.

UX Design Strategy

The UX of the Task Manager app is designed to provide a smooth, intuitive experience by following common user flow patterns that allow users to interact with tasks easily and efficiently.

User Flow

- **Task Creation:** Users can easily create tasks by clicking a prominent "Add Task" button. They can enter the task name, description, and set priorities or deadlines.
- **Task Management:** Tasks can be marked as completed, edited, or deleted from a simple list view. Each task displays key information (title, status, priority), and the user can interact with these items with minimal clicks.
- **Task Filters:** Users can filter tasks by priority, deadline, or status (e.g., completed, pending).
- **Responsive Layouts:** The app's layout adapts seamlessly to various screen sizes. On mobile, the app switches to a column-based layout for easy reading and navigation, while maintaining all core functionalities.

Navigation

- **Sidebar Navigation**: A clean sidebar allows users to navigate between sections like the Dashboard, Completed Tasks, and Task Settings.
- **Top Bar**: The top navigation bar gives users access to important features like their profile, app settings, and logout.

Visual Design

Color Scheme

The color palette for the Task Manager app was chosen to be both functional and aesthetic:

- **Primary Color:** A soothing blue is used for primary buttons and action items, providing a sense of trust and reliability.
- **Secondary Color:** A lighter color palette (like light greys and whites) ensures readability and creates a clean environment for the user to focus on tasks.
- **Accent Colors:** Green and red are used to represent success (completed tasks) and error/alert (overdue or high-priority tasks) states, respectively.

Typography

- A simple, easy-to-read sans-serif font is used for the majority of the app, ensuring a modern and clean look.
- **Font Size Hierarchy:** The app uses a clear typographic hierarchy, with headings being bold and large to indicate important sections, while body text is smaller for better readability.

Icons and Imagery

- Custom icons are used throughout the app for actions like add, edit, delete, and filtering tasks. The icons are intuitive and designed to be simple and clear.
- The app uses minimal imagery, focusing on the task management elements. This helps maintain a clean interface and ensures that users' attention remains on managing tasks.

Responsive Design

Mobile Responsiveness

- The app is fully responsive, ensuring optimal experience for users on mobile devices.
    - On smaller screens, the sidebar navigation collapses into a hamburger menu, allowing users to toggle between sections.
    - Task lists and other key components are stacked vertically, making them easier to read and interact with on mobile devices.
    -

Tablet and Desktop View

- The desktop version uses a wider layout with side-by-side panels, allowing users to view more information at once. This is especially useful when managing a large number of tasks.
- The task list is presented in a grid-like format, with each task occupying its own card for better organization.

User Interactions and Feedback

Task Creation and Editing

- When a user creates or edits a task, the app provides immediate **visual feedback** to confirm the action (e.g., a “Task created” notification).
- The input fields for task creation are simple and intuitive, with tooltips and placeholders to guide the user.

Task Status Updates

- A status dropdown allows users to easily update the task status (e.g., to "Completed" or "In Progress").
- Changes to the task’s status are reflected immediately in the task list with smooth transitions and color changes (e.g., completed tasks turn green).

Error and Success Messages

- The app provides feedback when actions are successful (e.g., “Task successfully created”) and error messages (e.g., “Failed to save task”). These messages are clear and concise to guide users.
- **Loading Indicators**: When the app is waiting for backend data, like when fetching tasks from the server, a loading spinner or animation is shown to indicate progress.

Usability Testing and Iteration

The design and functionality of the Task Manager app underwent **user testing** to ensure its usability. Feedback from users helped refine the following aspects:

- **Task Navigation:** Users felt that adding and organizing tasks was intuitive. However, they requested an easier way to prioritize tasks, so the ability to sort tasks by priority was added.
- **Mobile Experience:** Users on mobile devices requested a better way to edit tasks. The interface was adjusted to provide larger input areas and more accessible task management controls.
- **Filter Options:** Users requested more filter options for tasks (e.g., filtering by deadlines, due dates), and this was incorporated into the design.

Design Tools and Technologies

The following tools and technologies were used to design and implement the UI/UX for the Task Manager app:

- **Figma**: Used for wireframing and UI design mockups. Interactive prototypes were created to visualize the user experience before development.
- **ReactJS**: The framework used for building the interactive UI, with components responsible for managing the app’s state and rendering the task lists.
- **CSS (Flexbox, Grid)**: Used for creating responsive layouts and handling the positioning of elements on both mobile and desktop screens.
- **Material-UI**: A component library that provided ready-to-use design elements like buttons, sliders, and inputs that align with best practices for UI/UX.

The UI/UX design of the Task Manager web app focuses on providing a simple, intuitive, and visually appealing experience for users. Through careful planning, testing, and iteration, the app was designed to help users effectively manage tasks with minimal effort, while maintaining a clean and responsive design. The overall goal was to create an easy-to-use interface with a seamless user experience that encourages productivity.

System Deployment & Integration

Technology Stack

HTML (Hypertext Markup Language) is the fundamental building block of web development. It provides the structure of a web page by using a system of tags and attributes to organize content like headings, paragraphs, images, links, and other elements. HTML ensures that browsers can correctly display content on websites, making it a core technology for web development. It forms the skeleton of any web page and is complemented by CSS for styling and JavaScript for interactivity.

CSS (Cascading Style Sheets) is used to control the visual presentation of our project web page. We use CSS to style the HTML structure, adjusting elements like layout, colors, fonts, and spacing and creating an engaging, user-friendly design. With CSS, we make our web pages responsive, ensuring they look great on any device, from desktops to mobile phones. It works hand-in-hand with HTML: while we use HTML to provide the structure of the page, we use CSS to enhance its appearance. We also use CSS for advanced styling techniques like animations, transitions, and grid layouts.

JavaScript is a dynamic programming language that brings interactivity and functionality to our website project. We use it to create interactive features like form validation, animations and dynamic content updates. JavaScript works on the client side (in the user's browser), making it crucial for creating fast, responsive web experiences. With JavaScript, we manipulate HTML and CSS elements, create interactive user interfaces, and communicate with servers asynchronously (using techniques like AJAX and Fetch). It integrates seamlessly with other web technologies and frameworks that we use in our project like React.

React is a popular JavaScript library used to build user interfaces, particularly for single-page applications (SPAs). Developed by Facebook. we use React to create reusable UI components that can manage state, allowing for efficient updates and rendering when the data changes. React uses a virtual DOM (Document Object Model) to optimize performance, making our website project faster than traditional methods of updating the real DOM. With React, we built interactive web application with a clean, declarative approach. React allowed our project to be dynamic, real-time updates and rich user interactions.

Redux is a state management library commonly used with React. It helps manage and centralize the state of an application, making it predictable and easier to debug. With Redux, the entire application state is stored in a single "store," and any changes to the state are made through actions and reducers. This allows for better control over how data flows through the application, ensuring that all parts of the app remain in sync. Redux is especially useful in large-scale applications with complex state interactions, as it prevents the issue of "prop drilling" (passing data through many layers of components). It also enables powerful debugging tools, like time-travel debugging, where you can see how the state changes over time.

Node.js is a powerful and efficient JavaScript runtime built on Chrome's V8 JavaScript engine, designed for building scalable network applications. node.js is highly suitable for I/O-heavy operations such as file handling, network requests, and real-time applications. Node.js is used in our project to handle requests from the client-side (such as user authentication), connect to the database, and serve dynamic content quickly.

MongoDB is a NoSQL, document-oriented database that is designed to store, manage, and process large volumes of data. MongoDB stores data in flexible, JSON-like documents, which makes it easy to handle. This structure is ideal for applications that require high availability, scalability, and the ability to handle large amounts of unstructured data. It provides a flexible way to manage this data without requiring rigid table structures. MongoDB is used in our project to store user data, such as login authentication, user profiles, and other dynamic content.

Express.js is a fast, minimalist web application framework built on top of Node.js. It simplifies the process of building robust and scalable web applications and APIs by providing a set of tools and features for handling routing, middleware, and request/response management. Express is widely used in the Node.js ecosystem due to its simplicity and flexibility, making it an excellent choice for both small applications and large-scale systems. Express is used in our project to handle incoming HTTP requests, define routes for user authentication, and interact with the database to validate login authentication or return user data.

Deployment

The Task Manager System is deployed using Netlify, providing an easy-to-use platform for continuous deployment. With automatic updates, custom domain support, and detailed performance monitoring, the app is always up-to-date and accessible to users. Future updates will be automatically deployed as the development team pushes changes to the GitHub repository.