

College Code : 9220

College Name : Sri vidya College of Engineering and Technology

Department : Computer Science and Engineering

Student NM –ID **: FACA50A3DF97E21AOD6427BE36EO**

Roll No : CS2315

Date : 06.10.25

Completed the project named as Phase 5

TECHNOLOGY PROJECT NAME : TO DO LIST APPLICATION

Submitted by ,

Name : A. Josephin Saral

Mobile No : 6369972826

**PHASE 5:TO DO LIST APPLICATION**

**Final Demo Walkthrough**

**1. Launch & Login:**

* User opens the app and logs in securely (or signs up).

**2. Dashboard Overview:**

* Clean, user-friendly interface displaying current tasks.

**3. Add a Task:**

* User enters task title, optional description, due date.
* Clicks “Add Task” → Task appears in the list instantly.

**4. Manage Tasks:**

* Mark as complete: Task gets crossed out or moved to “Completed” section.
* Edit: User can update task details.
* Delete: Removes task permanently.

**5. Filter & Sort:**

* Tasks can be filtered by status (All, Active, Completed) or sorted by due date.

**6. Responsive Design:**

* Fully functional on desktop, tablet, and mobile.

**7. Logout:**

* User logs out securely.

### Project Report

### Project Title:To-Do List Web Application

### 1.Objective

The main objective of the To-Do List application is to provide users with a digital tool for managing their daily tasks efficiently. It aims to help individuals improve productivity, keep track of responsibilities, and organize work in a user-friendly and accessible format.

### 2.Introduction

Task management is a crucial aspect of personal and professional life. In the digital era, manual to-do lists are increasingly being replaced by online applications that are accessible anytime and anywhere. This To-Do List application serves as a simple yet powerful tool where users can create, read, update, and delete tasks (CRUD operations), set due dates, and manage their workflow.

### 3.Problem Statement

Many individuals and professionals struggle with organizing their tasks and maintaining productivity due to lack of structure and reminders. While several task management apps exist, they are often complex or feature-heavy. This project solves the problem by offering a **minimal, intuitive, and responsive** to-do list application tailored for ease of use and fast access.

### 4.Scope of the Project

* Users can create, view, edit, and delete tasks.
* Tasks can be marked as completed or pending.
* Users can filter and search tasks.
* The application is responsive across devices (desktop/tablet/mobile).
* Provides secure user authentication (login/register).
* Potential to expand with features like reminders, priorities, and recurring tasks.

### 5.System Requirements

* Frontend: HTML, CSS, JavaScript, React.js (or Vue/Angular)
* Backend: Node.js with Express.js / Django / Flask
* Database: MongoDB / MySQL / Firebase
* Optional: Git, GitHub, Postman

### 6.System Architecture:

* **Client (Frontend):** Handles the UI and user interactions.
* **Server (Backend):** Processes requests, handles business logic, and communicates with the database.
* **Database:** Stores user and task information persistently.

### 7.Modules of the Project

### User Authentication Module

* Sign Up
* Login
* Logout
* Session management

### Task Management Module

* Create task
* View task list
* Edit task
* Delete task
* Mark as complete

### Filter and Search Module

* Filter tasks by status (All, Completed, Pending)
* Sort by due date
* Search by keywords

### User Interface (UI) Module

* Responsive design
* Interactive elements (checkboxes, modals, etc.)
* Accessibility features

### 8. Database Design

### Users Table / Collection

* UserID (Primary Key)
* Name
* Email
* Password (hashed)

### Tasks Table / Collection

* TaskID (Primary Key)
* UserID (Foreign Key)
* Title
* Description
* Due Date
* Status (Pending / Completed)
* Created At
* Updated At

### 9. Advantages of the Application

* Enhances productivity and time management
* Easy to use, even for non-technical users
* Works across devices
* Can be extended with more features (reminders, collaboration, etc.)
* Encourages task tracking and habit formation

### 10. Future Enhancements

* **Reminders & Notifications** (Email or Push)
* **Task Prioritization**
* **Recurring Tasks**
* **Dark Mode**
* **Calendar View Integration**
* **Collaboration Features** (shared lists)
* **Offline Mode / PWA Support**

### Screenshots and API Documentation

### 1. Screenshots

Screenshots serve as **visual evidence** of the application’s user interface and demonstrate how different features appear and function. Including screenshots in a project report helps readers quickly understand the application’s usability, design, and workflow.

### Key Screenshots to Include:

* **Login and Registration Screen**:  
  Displays how users can securely access the app by logging in or creating an account. Typical elements include email and password input fields, action buttons, and validation messages.
* **Dashboard / Task List View**:  
  The main screen where users view their tasks. It typically lists all active, completed, and overdue tasks with options for filtering or searching. It also displays buttons to add new tasks.
* **Add New Task Form**:  
  Shows the interface where users input task details such as title, description, due date, and priority level. This form highlights user interaction and input validation.
* **Edit Task Screen**:  
  Demonstrates how users can update existing tasks. It includes pre-filled task information with editable fields.
* **Task Completion State**:  
  Illustrates how a task appears once marked complete, such as text strikethrough, checkbox ticked, or moved to a completed section.
* **Responsive Design View**:  
  Captures how the application adapts to different devices like tablets and smartphones, showing mobile-friendly layouts.

### 2. API Documentation

The To-Do List application typically uses a **RESTful API** to allow the frontend interface to communicate with the backend server. The API endpoints facilitate task management and user authentication.

### Purpose of API Documentation

API documentation describes available endpoints, request methods, required parameters, request and response formats, and expected status codes. It is essential for developers to understand how to interact with the backend and integrate frontend features.

### Base URL

All API endpoints are prefixed with the base URL where the backend is hosted.  
**Example:**

https://api.example-todolist.com

### User Authentication APIs

* **Register User** (POST /auth/register):  
  Allows new users to create an account by providing name, email, and password.
* **Login User** (POST /auth/login):  
  Authenticates existing users using email and password, returning an authentication token.
* **Logout User** (POST /auth/logout):  
  Ends the user session by invalidating the token.

### Task Management APIs

* **Get All Tasks** (GET /tasks):  
  Retrieves a list of all tasks belonging to the authenticated user.
* **Create Task** (POST /tasks):  
  Adds a new task with attributes like title, description, due date, and priority.
* **Get Single Task** (GET /tasks/{id}):  
  Retrieves details of a specific task identified by its ID.
* **Update Task** (PUT /tasks/{id}):  
  Modifies an existing task’s details.
* **Delete Task** (DELETE /tasks/{id}):  
  Removes a task permanently.

### Request and Response Formats

* **Request Body**:  
  Typically JSON format containing necessary fields, such as:
* **Response Body**:  
  JSON format providing confirmation, task data, or error messages.
* **Headers**:  
  Includes authentication tokens in the format:
* Authorization: Bearer <token>

### Error Handling and Status Codes

* **200 OK**: Successful GET or PUT requests.
* **201 Created**: Successful creation of a resource.
* **204 No Content**: Successful deletion without response body.
* **400 Bad Request**: Invalid input or missing fields.
* **401 Unauthorized**: Authentication failure or missing token.
* **404 Not Found**: Resource (e.g., task) does not exist.
* **500 Internal Server Error**: Server-side error.

**Conclusion**

The To-Do List application is a practical and scalable solution to everyday task management. It incorporates core web development concepts like CRUD operations, authentication, REST APIs, and responsive design. The app provides a foundation that can be built upon for more advanced productivity solutions, making it ideal for personal use or as a portfolio project demonstrating full-stack development skills.

### Challenges & Solutions

### User Authentication & Security

* **Problem:**  
  Implementing secure user authentication, protecting sensitive data like passwords, and managing sessions can be complex. Risks include data breaches, unauthorized access, and token theft.
* **Solution:**  
  Use well-established authentication methods such as **JWT (JSON Web Tokens)** or **OAuth**.  
  Passwords should be **hashed and salted** (e.g., using bcrypt).  
  Implement **HTTPS** to encrypt data in transit.  
  Add **token expiration** and **refresh tokens** to maintain session security.  
  Regularly validate inputs to avoid injection attacks.

### Data Persistence and Synchronization

* **Problem:**  
  Ensuring that tasks are reliably stored and synchronized across multiple devices and sessions can be tricky, especially with offline access.
* **Solution:**  
  Use a robust database like **MongoDB**, **Firebase**, or **SQL databases** to persist data.  
  Implement **RESTful APIs** for communication between frontend and backend.  
  For offline capability, use **local storage** or **IndexedDB** on the client-side and sync data when online (Progressive Web App techniques).

### Responsive and Intuitive UI Design

* **Problem:**  
  Creating a user interface that works seamlessly across devices (desktop, tablet, mobile) while remaining intuitive can be difficult.
* **Solution:**  
  Apply **responsive design principles** using CSS frameworks like **Bootstrap** or **Tailwind CSS**.  
  Test extensively on different devices and screen sizes.  
  Use consistent UI components and accessible design practices (keyboard navigation, screen reader support).

### Managing State and Performance

* **Problem:**  
  Handling dynamic data changes (adding, editing, deleting tasks) in real-time without lag or UI glitches, especially when the task list grows large.
* **Solution:**  
  Use efficient frontend state management libraries like **Redux**, **Vuex**, or React’s **Context API**.  
  Implement **pagination** or **lazy loading** if the task list is large.  
  Optimize API calls to reduce server load and improve responsiveness.

### Handling Errors and Edge Cases

* **Problem:**  
  Users might input invalid data, experience connectivity issues, or encounter unexpected app behavior.
* **Solution:**  
  Implement **input validation** both client-side and server-side.  
  Use **try-catch** blocks and proper error messages to guide users.  
  Provide **offline mode** or graceful fallback when network is unavailable.  
  Log errors for monitoring and debugging.

### Scalability and Future Feature Integration

* **Problem:**  
  Designing the app to accommodate future features like reminders, notifications, collaboration without major rework.
* **Solution:**  
  Follow **modular architecture** principles, separating concerns (authentication, task management, UI).  
  Use **RESTful APIs** with versioning.  
  Choose flexible databases and scalable hosting solutions (cloud services).  
  Write clean, maintainable code with proper documentation.

### Time Zone and Date Management

* **Problem:**  
  Handling due dates across different time zones to ensure tasks are due accurately for all users.
* **Solution:**  
  Store all date/time values in **UTC** format on the backend.  
  Convert to user’s local time zone on the frontend for display.  
  Use libraries like **Moment.js**, **date-fns**, or **Luxon** for date-time manipulation.

### README & Setup Guide

The To-Do List Application is a simple yet powerful web app designed to help users manage their daily tasks. It allows users to register and log in securely, add new tasks, edit existing ones, delete tasks, and mark tasks as completed. The application features a responsive design that works well on desktops, tablets, and mobile devices.

### Features

* Secure user registration and login with authentication
* Create, read, update, and delete tasks (CRUD operations)
* Mark tasks as completed or pending
* Filter and search tasks by status or keywords
* Responsive and user-friendly interface
* Backend API to manage data securely

### Technology Stack

* **Frontend:** Built with React.js (or your preferred framework)
* **Backend:** Node.js with Express.js for RESTful API development
* **Database:** MongoDB for storing user data and tasks
* **Authentication:** JSON Web Tokens (JWT) for secure user sessions
* **Styling:** CSS framework like Bootstrap or Tailwind CSS for responsive design

### Application Set Up

### Prerequisites

* Node.js (version 14 or above recommended)
* npm (Node Package Manager) or yarn
* MongoDB (local installation or access to a cloud service like MongoDB Atlas)

### Step-by-Step Setup

1. **Clone the project repository** from GitHub using the command:  
   git clone https://github.com/yourusername/todo-list-app.git
2. **Navigate to the project directory:**  
   cd todo-list-app
3. **Backend setup:**
   * Go to the backend folder: cd backend
   * Install dependencies by running: npm install
   * Create a .env file in the backend folder with the following variables:
   * PORT=5000
   * MONGO\_URI=your\_mongodb\_connection\_string
   * JWT\_SECRET=your\_jwt\_secret\_key
4. **Frontend setup:**
   * Open a new terminal window and navigate to the frontend folder: cd frontend
   * Install dependencies: npm install
   * Start the frontend server: npm start  
     The frontend will be available at http://localhost:3000

### Application

* Open the frontend URL in your browser.
* Register a new user account or log in with existing credentials.
* Add tasks by filling out the task form with title, description, due date, and priority.
* View your tasks in the dashboard, edit or delete tasks as needed.
* Mark tasks as completed by checking the checkbox next to each task.
* Use filters and search to find specific tasks quickly.

### API Overview

* **User Authentication:**
  + POST /auth/register for registering new users
  + POST /auth/login for user login and token generation
  + POST /auth/logout to invalidate user sessions
* **Task Management:**
  + GET /tasks to fetch all user tasks
  + POST /tasks to create a new task
  + GET /tasks/:id to retrieve a specific task
  + PUT /tasks/:id to update a task
  + DELETE /tasks/:id to delete a task

**Final submission (repo +deployed link)**

**Purpose:**

* Helps users organize and prioritize daily tasks effectively
* Boosts productivity by keeping task management simple and accessible
* Provides a digital alternative to traditional pen-and-paper lists

**Core Functionalities:**

* User authentication with secure signup/login to protect personal data
* Task management: create, edit, delete, and mark tasks as completed
* Categorize tasks by priority levels (e.g., low, medium, high)
* Due date setting and reminders (if implemented) to track deadlines
* Filter tasks by completion status or priority for better visibility
* Search functionality for quick access to specific tasks
* Responsive UI design supporting all device types for ease of use

**Technology Importance:**

* Enables real-time updates and smooth user interaction
* Supports multi-user environments and data security

**Project Details**

**GitHub Repository:**

* Contains well-structured frontend and backend directories
* Includes environment configuration files and .gitignore for security
* Detailed README with setup instructions, API docs, and contribution guidelines
* Version control enables collaboration and continuous improvement

**Deployment Details:**

* Hosted on platforms like Netlify, Vercel, or Heroku for reliability
* Supports HTTPS for secure data transfer
* Continuous deployment set up to reflect the latest changes automatically
* Provides easy access and scalability for growing user base

**Technology Stack Breakdown:**

* **Frontend:** React.js with functional components, hooks, and state management for efficient UI rendering
* **Backend:** Node.js with Express.js, structured REST API endpoints, and middleware for authentication and validation
* **Database:** MongoDB with Mongoose ODM for schema management and data integrity
* **Authentication:** JWT with token-based security, token expiration, and refresh mechanisms
* **Styling:** Bootstrap/Tailwind CSS ensures mobile-first design and consistent theming
* **Tools:** Axios for API calls, ESLint and Prettier for code quality

**Final Notes & Access**

**Key Features Recap:**

* Comprehensive task CRUD operations with user-friendly forms and validation
* Secure and scalable user authentication to safeguard data privacy
* Dynamic UI updates ensuring seamless user experience without page reloads
* Search and filter functionalities to improve task management efficiency
* Responsive layout ensuring accessibility on desktops, tablets, and smartphones
* Error handling with user notifications to guide actions and prevent mistakes
* Modular code architecture supporting future enhancements like notifications, collaboration, or offline mode

**Access & Usage Instructions:**

* GitHub repo link provides source code and all necessary documentation
* Live deployed version accessible via browser for immediate use
* Detailed setup guide includes steps for local environment configuration, dependency installation, and running servers
* Encourages community contributions with clear guidelines for pull requests and issues

**Future Improvements (Optional):**

* Adding push notifications or email reminders
* Implementing task sharing and collaboration features
* Offline support with data synchronization
* Integrating calendar view for better task visualization