## **Task Management System with React: Project Plan**

### **Phase 1: Project Planning & Requirements Gathering**

#### **1.1 Project Proposal**

* Overview: Develop a Task Management System to help users create, update, and delete tasks. Users can categorize tasks, set priority levels, and filter tasks by status.
* Objective:
  + Build a user-friendly task management tool with React and Redux for state management.
  + Enable users to create, assign, and manage tasks with features like categorization, priority levels, and due dates.
  + Implement filtering and search functionality for tasks.
* Scope:
  + Task creation, updating, and deletion.
  + Task categorization and priority levels.
  + Filtering and searching tasks by status (completed, in-progress).
  + ~~Backend integration with Node.js and MongoDB.~~

#### **1.2 Stakeholder Analysis**

* Internal Stakeholders:
  + Team Members: Mohammed amir , Abdelrhman Khalid , Abdelrahman OKda, Mohamed zaghlol.
* External Stakeholders:
  + Users: Individuals or teams managing tasks.
  + ~~Suppliers: Third-party API providers (if needed) and hosting services.~~

#### **1.3 Requirements Gathering**

* Functional Requirements:
  + User authentication (optional for this project).
  + Task creation, updating, and deletion.
  + Task categorization and priority levels.
  + Filtering and searching tasks by status.
  + Data visualization (e.g., task progress charts).
* ~~Non-Functional Requirements:~~
  + ~~Performance: API response time < 200ms, dashboard load time < 3 seconds.~~
  + ~~Security: Secure authentication (if implemented), data encryption.~~
  + ~~Usability: Responsive design, intuitive UI.~~
  + ~~Scalability: Handle 1000+ concurrent users.~~

#### **1.~~4 Risk Assessment & Mitigation~~**

* ~~Data Integrity Risks: Implement database backups and input validation.~~
* ~~Scalability Risks: Use cloud-based solutions like AWS or Firebase.~~
* ~~UX Risks: Conduct user testing and iterate on design.~~
* ~~Third-Party Dependency Risks: Regularly update libraries and have fallback solutions.~~
* ~~Timeline Risks: Follow Agile methodology with clear milestones.~~

### **Phase 2: System Design & Architecture**

#### **2.1 System Architecture**

* Frontend: React with Redux for state management.
* ~~Backend: Node.js/Express for API handling.~~
* ~~Database: MongoDB for storing tasks and user data (if authentication is added).~~
* Data Flow:
  + Frontend sends requests to the backend API.
  + Backend processes requests and interacts with MongoDB.
  + Backend sends responses back to the frontend.

#### **2.2 ~~Database Design~~**

* ~~Entities:~~
  + ~~Tasks: Store task details (title, description, priority, due date, status).~~
  + ~~Users (optional): Store user account details (if authentication is added).~~
* ~~Relationships:~~
  + ~~One user can have many tasks (if authentication is added).~~
  + ~~Each task belongs to a specific user (if authentication is added).~~

#### **2.3 UI/UX Design**

* Wireframes: Design layouts for task lists, task creation, and filtering.
* Prototypes: Create interactive prototypes using Figma.
* Design Principles:
  + Consistency: Uniform design across all pages.
  + Simplicity: Avoid clutter and prioritize essential elements.
  + Feedback: Provide real-time updates and visual feedback.

### **Phase 3: Development**

#### **3.1 Frontend Development**

* Tasks:
  + Develop React components for task lists, task creation, and filtering.
  + Integrate Redux for global state management.
  + Implement data visualization (e.g., task progress charts).
* Deliverables:
  + Functional UI with responsive design.
  + ~~Integration with backend API for data fetching and updates.~~

#### **3.2 ~~Backend Development~~**

* ~~Tasks:~~
  + ~~Build RESTful API endpoints for task creation, updating, and deletion.~~
  + ~~Implement data validation and error handling.~~
  + ~~Integrate MongoDB for data storage.~~
* ~~Deliverables:~~
  + ~~Secure and scalable backend API.~~
  + ~~Database schema and CRUD operations.~~

#### **~~3.3 Integration~~**

* ~~Tasks:~~
  + ~~Connect frontend with backend API.~~
  + ~~Test API endpoints using Postman.~~
  + ~~Ensure data consistency between frontend and backend.~~
* ~~Deliverables:~~
  + ~~Fully integrated application with end-to-end functionality.~~

### **~~Phase 4: Testing & Quality Assurance~~**

#### **~~4.1 Unit Testing~~**

* ~~Frontend: Test React components using Jest and React Testing Library.~~
* ~~Backend: Test API endpoints using Mocha/Chai.~~
* ~~Key Test Cases:~~
  + ~~Task creation, updating, and deletion.~~
  + ~~Task filtering and searching.~~

#### **~~4.2 Integration Testing~~**

* ~~Tools: Postman for API testing, Cypress for end-to-end UI testing.~~
* ~~Scenarios:~~
  + ~~User creates a task and views it in the task list.~~
  + ~~User filters tasks by status and priority.~~

#### **~~4.3 User Acceptance Testing (UAT)~~**

* ~~Participants: Stakeholders and end-users.~~
* ~~Environment: Staging deployment on Netlify (frontend) and Render (backend).~~
* ~~Test Cases:~~
  + ~~Validate task creation, updating, and deletion.~~
  + ~~Test task filtering and searching.~~

### **Phase 5: Deployment**

#### **5.1 Hosting Environment**

* Frontend: Deploy React app to Netlify or Vercel.
* ~~Backend: Deploy Node.js API to Render or Heroku.~~
* ~~Database: Use MongoDB Atlas for cloud-based hosting.~~

#### **5.2 CI/CD Pipeline**

* Frontend: Automate deployment using Netlify/Vercel.
* ~~Backend: Use GitHub Actions or Render auto-deploy.~~
* ~~Database: Configure automatic backups and scaling.~~

#### **5.3 Monitoring & Logging**

* Tools:
  + Netlify Analytics for frontend traffic.
  + Render Logs for backend performance.
  + MongoDB Atlas Metrics for database monitoring.
* Alerts: Set up notifications for errors and performance issues.

### **Phase 6: Maintenance & Support**

#### **6.1 Bug Fixes**

* Address issues reported by users.
* Regularly update dependencies to patch vulnerabilities.

#### **6.2 Feature Enhancements**

* Add new features like user authentication, task sharing, and notifications.
* Optimize performance based on user feedback.

#### **6.3 Documentation**

* Update documentation with new features and changes.
* Provide user guides and API documentation.

## **Task Assignment**

| **Task** | **Assigned To** |
| --- | --- |
| Project Planning & Requirements | Mohammed Ahmed Zaghloul |
| UI/UX Design & Wireframes | Abdelrahman Oqda |
| Frontend Development (React) | Abdelrhman Khaled |
| Backend Development (Node.js) | Mohamed Amir |
| Integration & Testing | All Team Members |
| Deployment & Documentation | Mohammed Ahmed Zaghloul |

## **Timeline**

| **Week** | **Phase** | **Tasks** |
| --- | --- | --- |
| 1 | Planning & Requirements | Finalize project scope, gather requirements, conduct stakeholder analysis. |
| 2-3 | Design | Create system architecture, database schema, and UI/UX prototypes. |
| 4-6 | Development | Develop frontend and backend, integrate components, and test APIs. |
| 7 | Testing | Conduct unit, integration, and user acceptance testing. |
| 8 | Deployment | Deploy application, set up CI/CD pipeline, and configure monitoring. |
| 9+ | Maintenance | Address bugs, add features, and update documentation. |

Task\_name = انشاء واجهة لموقع ادارة

Type = UI design

Priority = very important (color )