**Task Management System Documentation**

**Overview**

The Task Management System is a comprehensive tool designed to help users efficiently manage their daily tasks. The system allows users to create, update, delete, and view tasks, set priorities, and track due dates. It uses a modern tech stack, including MongoDB, Express.js, React.js, and Node.js (MERN).

**Features**

1. **Task Management**
   * Create, update, and delete tasks.
   * View all tasks in a list.
   * Filter tasks based on priority or due date.
2. **Prioritization**
   * Assign priorities to tasks (Low, Medium, High).
3. **Due Date Tracking**
   * Set and view due dates for tasks.
4. **Responsive UI**
   * A user-friendly and mobile-responsive interface.
5. **Persistent Storage**
   * Stores data securely in MongoDB.

**Technologies Used**

**Frontend**

* React.js: For creating a dynamic and interactive user interface.
* CSS: For styling the application with a clean and modern design.

**Backend**

* Node.js: As the runtime environment.
* Express.js: For handling API requests and routing.

**Database**

* MongoDB: For storing tasks persistently.

**Others**

* Mongoose: For Object Data Modeling (ODM) in MongoDB.
* Axios: For API requests.

**System Architecture**

The Task Management System follows a modular design:

1. **Frontend**
   * React components handle user interactions and display data.
2. **Backend**
   * RESTful API endpoints to handle CRUD operations.
   * Middleware for validation and error handling.
3. **Database**
   * MongoDB for data storage, managed using Mongoose schemas.

**Setup and Installation**

**Prerequisites**

* Node.js and npm installed.
* MongoDB installed locally or a MongoDB Atlas cluster.
* Git installed.

**Steps**

1. **Clone the repository:**
2. git clone <repository-url>
3. cd task-management-system
4. **Install dependencies:**
5. npm install
6. cd client
7. npm install
8. **Set up environment variables:** Create a .env file in the root directory and add:
9. MONGO\_URI=mongodb+srv://<username>:<password>@cluster0.mongodb.net/<dbname>?retryWrites=true&w=majority
10. PORT=5000
11. **Start the application:**
    * Run the backend:
    * npm start
    * Run the frontend:
    * cd client
    * npm start
12. **Access the application:** Open your browser and navigate to http://localhost:3000.

**API Endpoints**

**Base URL: http://localhost:5000/api**

1. **GET /tasks**
   * Fetch all tasks.
   * Response: Array of tasks.
2. **POST /tasks**
   * Create a new task.
   * Body: { "title": "Task title", "description": "Task description", "priority": "High", "dueDate": "2024-12-31" }
3. **DELETE /tasks/:id**
   * Delete a task by ID.
4. **PUT /tasks/:id**
   * Update a task by ID.
   * Body: { "title": "Updated title", "completed": true }

**Database Schema**

**Task Schema**

const taskSchema = new mongoose.Schema({

title: { type: String, required: true },

description: { type: String },

priority: { type: String, enum: ['Low', 'Medium', 'High'], default: 'Medium' },

dueDate: { type: Date, default: Date.now },

completed: { type: Boolean, default: false },

});

**Frontend Components**

1. **TaskList**
   * Displays the list of tasks.
2. **TaskForm**
   * A form to create or update tasks.
3. **TaskItem**
   * Represents an individual task with options to edit or delete.

**Future Enhancements**

1. Add user authentication.
2. Introduce notifications for upcoming due dates.
3. Export tasks to CSV or Excel. 4