# Task Manager

A simple Task Manager application built with Node.js, Express, MongoDB, and React, allowing users to create, update, delete, and mark tasks as complete. The app is structured with a REST API backend and a React frontend, designed for easy deployment with Docker and Kubernetes.

## Features

* - User Authentication: Each user can register, log in, and access their tasks only.
* - Task Management: Create, update, delete, and mark tasks as complete.
* - Task Details: Each task has a title, description, and completion status.
* - API Security: Protected routes with JWT-based authentication.

## Technologies Used

* - Backend: Node.js, Express, MongoDB, Mongoose
* - Frontend: React, Material-UI
* - Authentication: JWT (JSON Web Token)
* - Deployment: Docker, Kubernetes

## Setup and Installation

### Prerequisites

* - Node.js (v12 or later)
* - MongoDB (v4 or later)
* - Docker and Docker Compose
* - Kubernetes (for deployment)

### Backend Setup

1. Clone the repository:

git clone https://github.com/geco97/task-manager.git  
 cd task-manager

2. Install dependencies for each microservice:

- For `auth-service`:

cd backend/auth-service  
 npm install

- For `task-service`:

cd ../task-service  
 npm install

3. Create `.env` files in each microservice with required environment variables.

### Frontend Setup

1. Navigate to the frontend directory:

cd ../frontend

2. Install dependencies:

npm install

3. Create a `.env` file in the frontend directory with API URLs.

4. Run the frontend server:

npm start

## API Endpoints

### Authentication

* POST /auth/register: Register a new user.
* POST /auth/login: Log in to get an authentication token.

### Tasks

* GET /tasks: Get all tasks for the logged-in user.
* POST /tasks: Create a new task.
* PUT /tasks/:id: Update a task by ID (title and completion status).
* DELETE /tasks/:id: Delete a task by ID.

## Docker and Kubernetes Deployment

### Building and Pushing Docker Images

1. Navigate to each directory and build images:

- For `auth-service`: docker build -t your-dockerhub-username/auth-service:latest ./backend/auth-service

- For `task-service`: docker build -t your-dockerhub-username/task-service:latest ./backend/task-service

- For `frontend`: docker build -t your-dockerhub-username/frontend:latest ./frontend

2. Push images to Docker Hub:

- docker push your-dockerhub-username/auth-service:latest

- docker push your-dockerhub-username/task-service:latest

- docker push your-dockerhub-username/frontend:latest

### Docker Compose

To build and run services using Docker Compose:

docker-compose up --build

### Kubernetes Deployment

1. Create Kubernetes manifests for each service.

2. Apply configurations:

kubectl apply -f kubernetes/auth-service-deployment.yaml

kubectl apply -f kubernetes/task-service-deployment.yaml

kubectl apply -f kubernetes/mongo-deployment.yaml

kubectl apply -f kubernetes/frontend-deployment.yaml

## Project Structure

task-manager/  
├── backend/  
│ ├── auth-service/ # User Authentication Microservice  
│ │ ├── src/  
│ │ │ ├── controllers/  
│ │ │ │ └── authController.js # Handles registration, login, profile actions  
│ │ │ ├── models/  
│ │ │ │ └── User.js # User model/schema for MongoDB  
│ │ │ ├── routes/  
│ │ │ │ └── authRoutes.js # Routes for auth-related endpoints  
│ │ │ ├── utils/  
│ │ │ │ ├── jwtUtils.js # Helper functions for JWT handling  
│ │ │ │ └── hashUtils.js # Helper functions for password hashing  
│ │ │ ├── app.js # Main Express app configuration  
│ │ │ └── server.js # Server entry point  
│ ├── Dockerfile # Dockerfile for auth service  
│ ├── .env # Environment variables (JWT\_SECRET, DB\_URI)  
│ └── package.json # Dependencies for auth service  
├── task-service/ # Task Management Microservice  
│ ├── src/  
│ │ ├── controllers/  
│ │ │ └── taskController.js # Handles CRUD actions for tasks  
│ │ ├── models/  
│ │ │ └── Task.js # Task model/schema for MongoDB  
│ │ ├── routes/  
│ │ │ └── taskRoutes.js # Routes for task-related endpoints  
│ │ ├── middleware/  
│ │ │ └── authMiddleware.js # Middleware to verify JWT tokens  
│ │ ├── app.js # Main Express app configuration  
│ │ └── server.js # Server entry point  
│ ├── Dockerfile # Dockerfile for task service  
│ ├── .env # Environment variables (DB\_URI)  
│ └── package.json # Dependencies for task service  
├── frontend/  
│ ├── src/  
│ │ ├── components/ # React components (TaskManager, Login, Register)  
│ │ ├── api/ # API functions for task and auth  
│ │ ├── App.js # Main React component  
│ │ └── index.js # ReactDOM entry point  
├── kubernetes/ # Kubernetes deployment files  
├── docker-compose.yml # Docker Compose setup for local deployment  
└── README.md # Project documentation

## License

This project is licensed under the MIT License.