#### 1. Backend Development

##### 1.1 Set Up Backend Server

mkdir task-manager

cd task-manager

npm init -y

npm install express

const express = require('express');

const app = express();

const port = 3000;

app.get('/', (req, res) => {

res.send('Task Manager API');

});

app.listen(port, () => {

console.log(`Server running on http://localhost:${port}`);

});

##### Implement User Authentication and Authorization

npm install jsonwebtoken bcryptjs dotenv

const jwt = require('jsonwebtoken');

const bcrypt = require('bcryptjs');

const dotenv = require('dotenv');

dotenv.config();

// User registration route

app.post('/register', async (req, res) => {

// Registration logic (hash password, save user)

});

##### 1.3 Create API Endpoints for Task Management

const tasks = []; // In-memory task storage (replace with DB integration)

app.post('/tasks', auth, (req, res) => {

// Create task logic

});

app.get('/tasks', auth, (req, res) => {

// Get tasks logic

});

app.put('/tasks/:id', auth, (req, res) => {

// Update task logic

});

app.delete('/tasks/:id', auth, (req, res) => {

// Delete task logic

});

##### 1.4 Validation and Error Handling

const { body, validationResult } = require('express-validator');

app.post('/register', [

body('email').isEmail(),

body('password').isLength({ min: 6 })

], (req, res) => {

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({ errors: errors.array() });

}

// Registration logic

});

// Error handling middleware

app.use((err, req, res, next) => {

res.status(500).json({ error: err.message });

});

#### 2. Database Setup

##### 2.1 Choose and Set Up Database

npm install mongoose

const mongoose = require('mongoose');

mongoose.connect(process.env.MONGODB\_URI, { useNewUrlParser: true, useUnifiedTopology: true });

const userSchema = new mongoose.Schema({

email: { type: String, required: true, unique: true },

password: { type: String, required: true }

});

const User = mongoose.model('User', userSchema);

const taskSchema = new mongoose.Schema({

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

description: String,

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

owner: { type: mongoose.Schema.Types.ObjectId, required: true, ref: 'User' }

});

const Task = mongoose.model('Task', taskSchema);

#### 3. Frontend Development

##### 3.1 Develop Frontend Interface

npx create-react-app task-manager-client

cd task-manager-client

##### 3.2 Implement User Registration and Login Forms

import React, { useState } from 'react';

const UserForm = ({ onSubmit }) => {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const handleSubmit = (e) => {

e.preventDefault();

onSubmit({ email, password });

};

return (

<form onSubmit={handleSubmit}>

<input type="email" value={email} onChange={(e) => setEmail(e.target.value)} />

<input type="password" value={password} onChange={(e) => setPassword(e.target.value)} />

<button type="submit">Submit</button>

</form>

);

};

export default UserForm;

##### 3.3 Create Task Management Pages

import React, { useState, useEffect } from 'react';

import axios from 'axios';

const TaskList = () => {

const [tasks, setTasks] = useState([]);

useEffect(() => {

axios.get('/api/tasks').then(response => {

setTasks(response.data);

});

}, []);

return (

<div>

<h1>Tasks</h1>

<ul>

{tasks.map(task => (

<li key={task.\_id}>{task.title}</li>

))}

</ul>

</div>

);

};

export default TaskList;

#### 4. Integration

##### 4.1 Integrate Frontend with Backend APIs

npm install axios

import axios from 'axios';

const registerUser = async (userData) => {

const response = await axios.post('/api/register', userData);

return response.data;

};

const loginUser = async (userData) => {

const response = await axios.post('/api/login', userData);

return response.data;

};

##### 4.2 Implement User Authentication

const loginUser = async (userData) => {

const response = await axios.post('/api/login', userData);

localStorage.setItem('token', response.data.token);

};

axios.interceptors.request.use(config => {

const token = localStorage.getItem('token');

if (token) {

config.headers.Authorization = `Bearer ${token}`;

}

return config;

});