Project Title :**Task Management Application**

Team Members:

Name :**Aditya Jyoti Sahu**

CAN ID NUMBER :**33977669**

Name :**Ansh Kumar Jha**

CAN ID NUMBER :**33977303**

Name :**Gyan Deep**

CAN ID NUMBER :**33976087**

Name :**Aditya Kshatriya**

CAN ID NUMBER :**33977273**

Institution Name :**Acharya Institute of Technology**

Phase 3:Implementation of Project

Objective

The goal of Phase 3 is to implement the core components of the task management application based on the planned architecture. This includes developing the front-end and back-end systems, implementing features for task creation, assignment, and tracking, integrating user authentication, and conducting initial testing to ensure a smooth and efficient user experience.

1.Front-end Development

Overview

The front-end will serve as the user interface for creating, assigning, and managing tasks. It will use React.js to ensure responsiveness and interactivity, with a focus on task organization, progress tracking, and user collaboration features.

Implementation

1.1. Project Structure:

- **App Component:**
  - App.js serves as the root component, importing and rendering the TaskManager component.
  - Contains basic JSX structure and CSS import from App.css.
- **TaskManager Component:**
  - Core UI component for handling task input, search, list rendering, and task management actions.
  - Uses React hooks (useState, useEffect) for managing component state and lifecycle methods.

1.2. UI Components:

- **Input Fields:**
  - Input field for adding a new task.
  - Separate input field for searching tasks.
- **Action Buttons:**
  - Add Task: Adds a new task or updates an existing one.
  - Check/Uncheck: Marks a task as done or not done.
  - Edit: Sets the task to update mode.
  - Delete: Deletes a task.
- **List Rendering:**
  - Dynamically renders tasks using map.
  - Applies conditional CSS classes for completed tasks (line-through styling).

1.3. State Management:

- input: State to manage the task input field.
- tasks: State to store the list of tasks fetched from the backend.
- copyTasks: State to maintain a copy of tasks for search filtering.

- updateTask: State to manage the task currently being updated.

1.4. API Integration:

- API methods are imported from ./api.
    - CreateTask: Sends a POST request to create a new task.
    - GetAllTasks: Sends a GET request to fetch all tasks.
    - DeleteTaskById: Sends a DELETE request to remove a task by ID.
    - UpdateTaskById: Sends a PUT request to update a task.
- API requests are asynchronous, using async/await.
- Error handling: Uses try-catch blocks and displays error messages with notify.

1.5. Search Functionality:

- handleSearch function filters tasks based on the input value.
- Converts both search term and task names to lowercase for case-insensitive searching.
- Updates the tasks state with the filtered results.

1.6. Notifications:

- Uses react-toastify to show success or error messages.
- notify function dynamically displays toasts based on message type (success/error).

1.7. CSS and Styling:

- Uses Bootstrap for layout and responsive design.
- Toastify CSS for notification styling.
- Custom CSS is imported from App.css.

1.8. External Dependencies:

- react-icons: Provides icons for task actions (add, edit, check, delete).
- react-toastify: Provides toast notifications.

- bootstrap: Provides responsive UI components and grid system.

Outcome

By the end of this phase, the front-end successfully allows users to create, update, delete, search, and manage tasks with a smooth and interactive experience.

2.Back-End Development

Overview

The back-end of this task management application is built using Node.js and Express.js to provide secure and scalable APIs for task management.

2.1. Project Structure:

- **Task Model:**
  - Defines a Mongoose schema with taskName (String) and isDone (Boolean) fields.
  - Collection name: todos.
- **Database Connection:**
  - Connects to MongoDB using Mongoose.
  - Uses DB_URL from environment variables.
- **Controllers:**
  - createTask: Creates and saves a new task.
  - fetchAllTasks: Retrieves all tasks from the database.
  - updateTaskById: Updates a task by ID.
  - deleteTaskById: Deletes a task by ID.
- **Routes:**
  - Defines endpoints for creating, fetching, updating, and deleting tasks.
  - Base route: /tasks.

2.2. API Endpoints:

- GET /tasks: Fetch all tasks.
- POST /tasks: Create a new task.
- PUT /tasks/:id: Update a task by ID.
- DELETE /tasks/:id: Delete a task by ID.

2.3. Middleware:

- body-parser: Parses incoming JSON requests.
- cors: Enables Cross-Origin Resource Sharing.
- dotenv: Loads environment variables.

2.4. Server Setup:

- Express server listens on PORT (from environment variables or default 8080).
- Base API route: /tasks (handled by TaskRouter).

2.5. Environment and Configuration:

- DB_URL: MongoDB connection string from .env file.
- PORT: Server port configuration.
- Uses Nodemon for hot-reloading in development.

2.6. Deployment:

- Vercel Configuration:
    - vercel.json specifies build and route configurations.
    - Uses @vercel/node for deployment.

Outcome

By the end of this phase, the back-end successfully manages task creation, retrieval, updating, and deletion with secure, scalable, and efficient API endpoints.

3.Database Design:

Overview

The database for this task management application uses **MongoDB** to efficiently store and manage task data.

Schema Design:

- **Task Schema:**
  - taskName: String (required) - Represents the name of the task.
  - isDone: Boolean (required) - Indicates whether the task is completed.

Data Relationships:

- The database maintains a simple, flat structure with tasks stored in the todos collection.
- Each task document contains a unique _id (auto-generated by MongoDB).

Outcome

By the end of this phase, the database successfully handles task creation, retrieval, updating, and deletion with data integrity and fast access.

4.Testing and Feedback:

Overview

Thorough testing ensures the application functions correctly, performs well, and delivers a great user experience.

Testing Implementation:

- **Unit Testing:**
- Test individual components and back-end routes using Jest or Mocha.
- **Integration Testing:**
    - Verify seamless communication between front-end, back-end, and database.
- **End-to-End Testing:**
    - Simulate user flows such as task creation, editing, deletion, and search.

Feedback Loop:

- Gather feedback from test users.
- Identify UX issues and performance bottlenecks.
- Iterate with improvements based on collected feedback.

Outcome

By the end of testing, the task management application runs smoothly, handles edge cases, and provides a refined user experience.

Challenges and Solutions:

Challenge 1: API Error Handling

- **Issue:** Ensuring smooth error handling for failed API calls (e.g., network issues or server downtime).
- **Solution:** Implemented try-catch blocks around async API calls and used toast notifications to inform users of errors.

Challenge 2: Real-Time State Synchronization

- **Issue:** Keeping the UI state (tasks) synchronized with the database after creation, deletion, and updates.
- **Solution:** Triggered fetchAllTasks after every task operation to ensure the UI stays updated.

Challenge 3: Case-Insensitive Search

- **Issue:** Search functionality needed to be case-insensitive and performant.
- **Solution:** Converted task names and search input to lowercase before filtering the results.

Challenge 4: Database Connection Reliability

- **Issue:** Ensuring MongoDB stays connected, especially on deployment.
- **Solution:** Used Mongoose connection with retry strategies and logged connection states for better monitoring.

Outcomes of Phase 3:

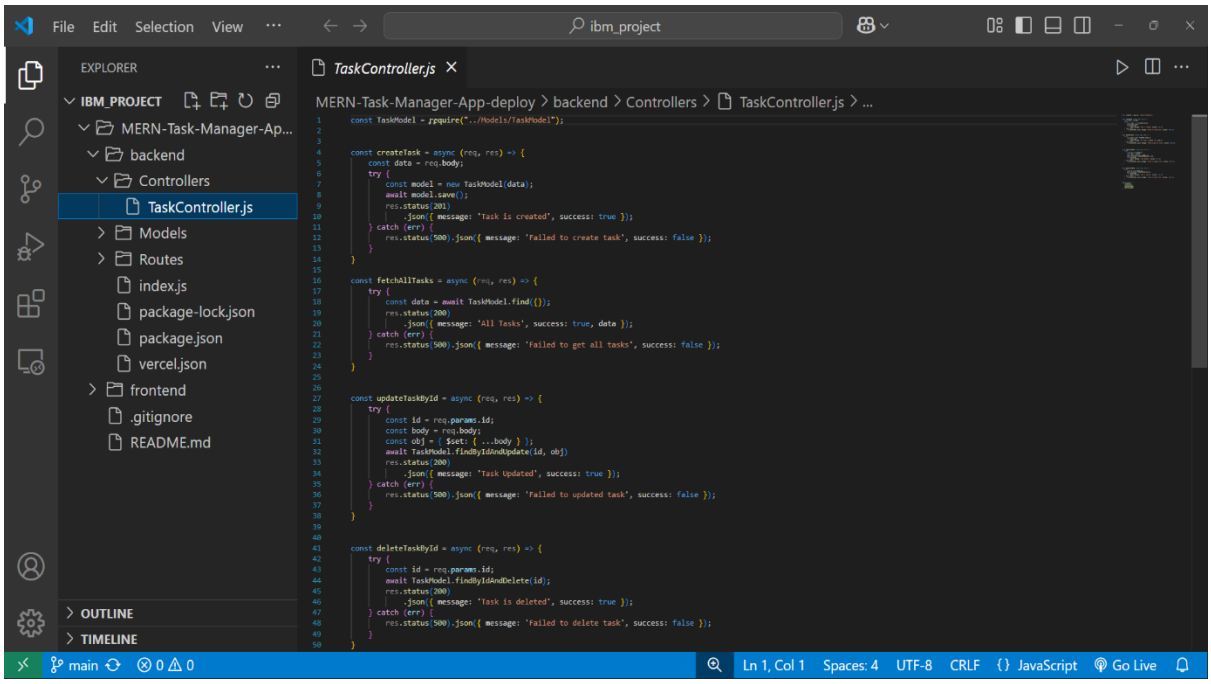
- **Front-End:** A functional UI for task creation, management, and search.
- **Back-End:** Secure and scalable APIs for task operations.
- **Database:** A structured MongoDB database for storing tasks efficiently.
- **Deployment:** Fully deployed application with a smooth user experience.
- **Testing:** Ensures stability, performance, and a seamless workflow.
- **Feedback:** Insights gathered for iterative improvements in the next phase.

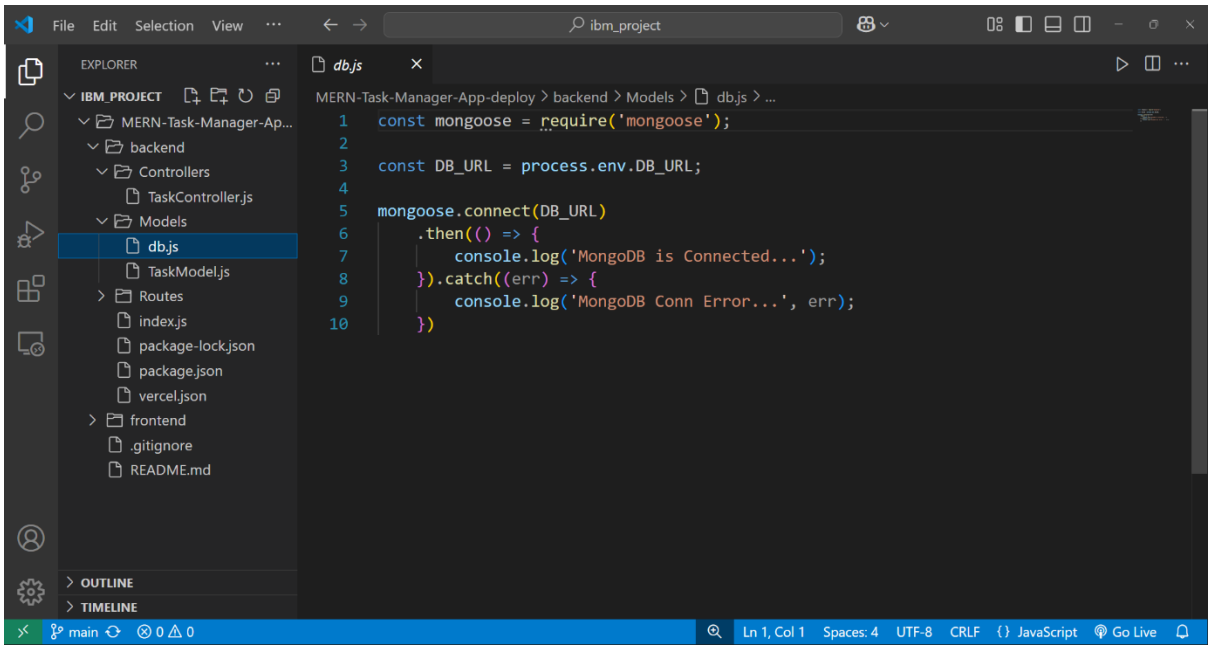
Next Steps for Phase 4:

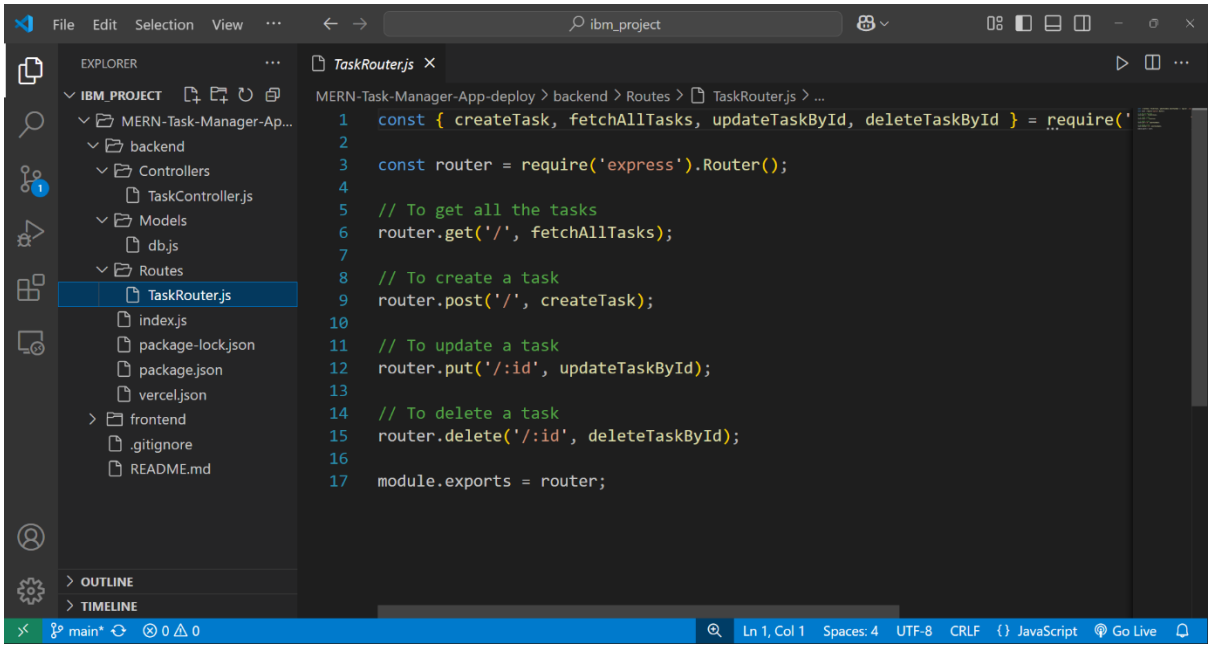
- Optimize performance by implementing lazy loading for large task lists.
- Introduce user authentication to manage individual task lists securely.
- Enhance the UI/UX with drag-and-drop functionality for task reordering.

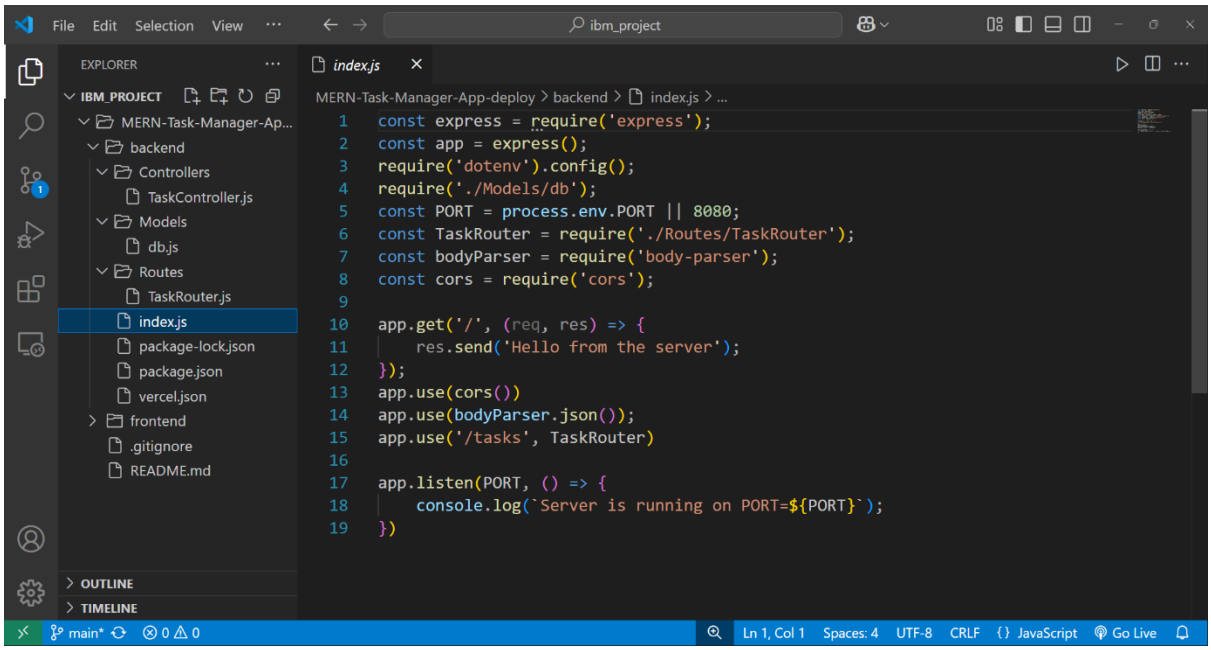
ScreenShots of Code and Progress

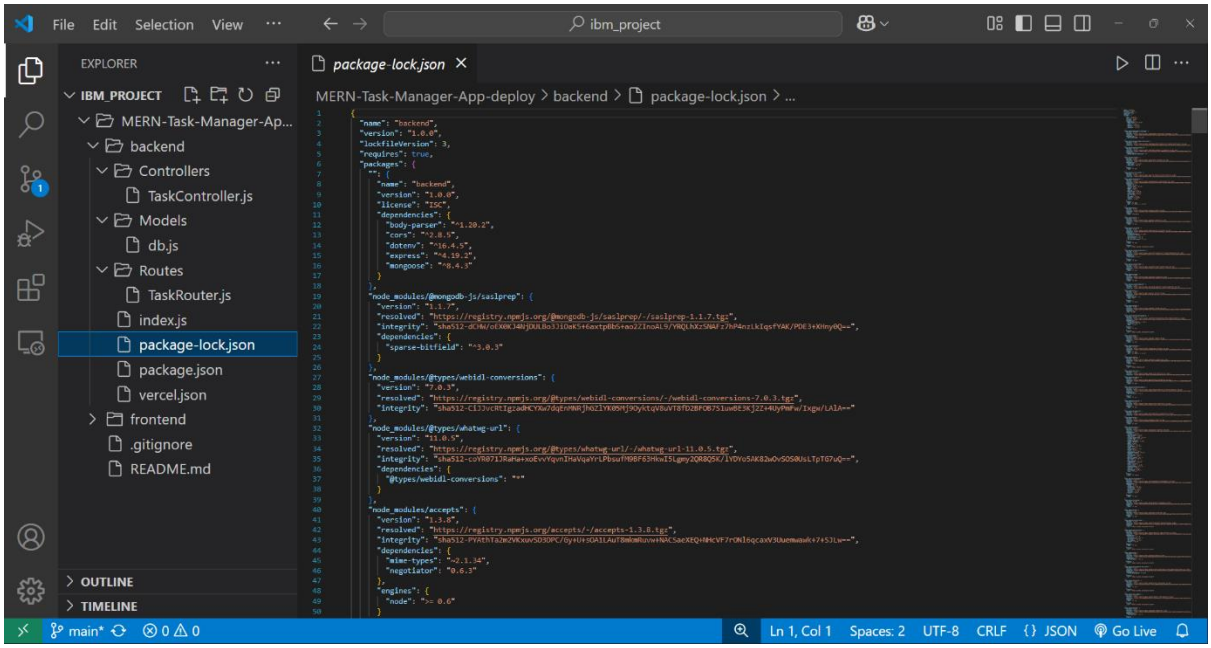
Back-End Side

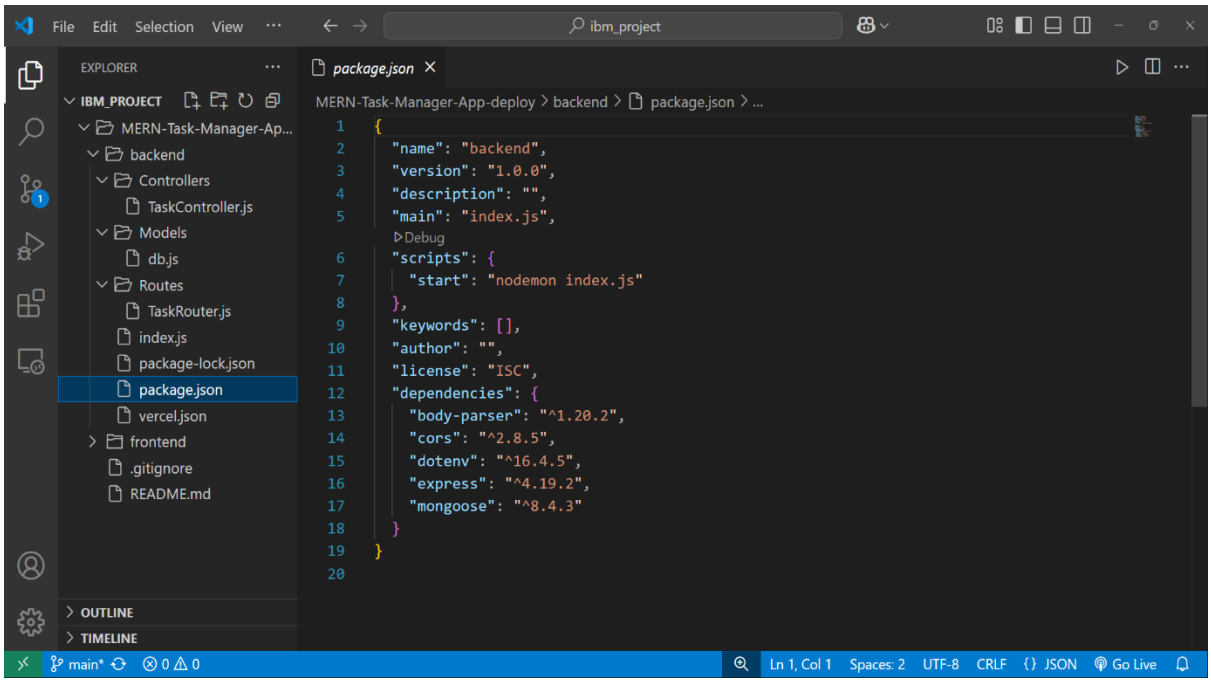


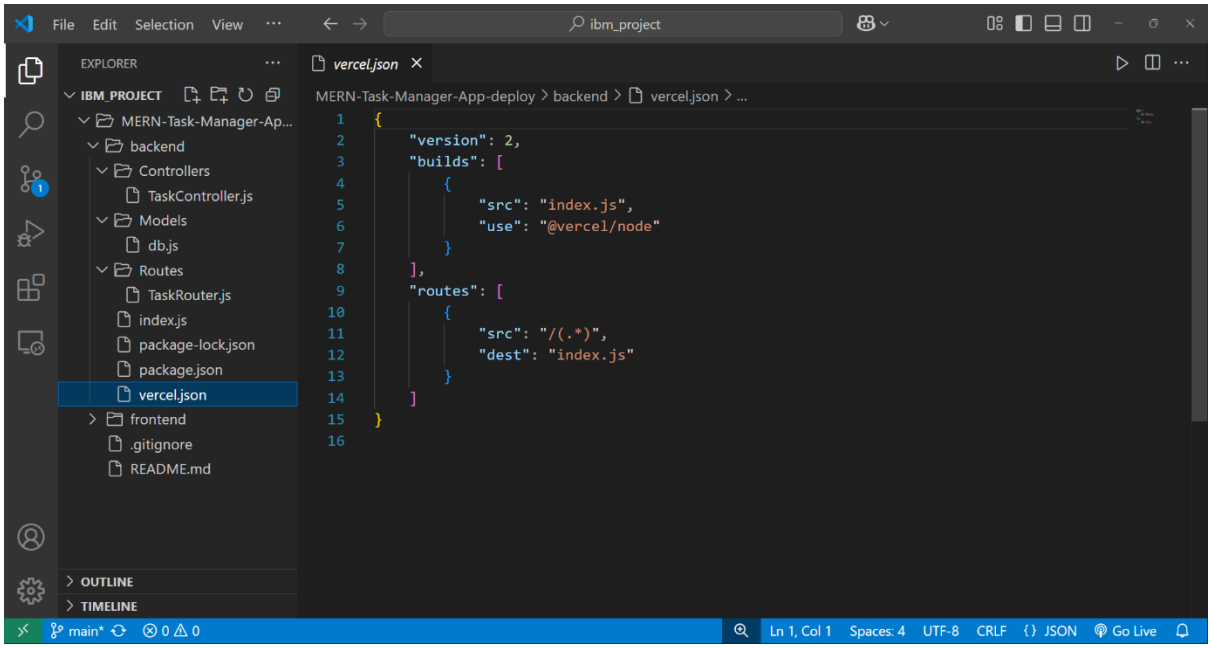



File Edit Selection View
ibm_project
EXPLORER
IBM_PROJECT
MERN-Task-Manager-Ap...
backend
Controllers
TaskController.js
Models
db.js
Routes
TaskRouter.js
index.js
package-lock.json
package.json
vercel.json
frontend
.gitignore
README.md
OUTLINE
TIMELINE
TaskRouter.js
MERN-Task-Manager-App-deploy > backend > Routes > TaskRouter.js > ...
const { createTask, fetchAllTasks, updateTaskById, deleteTaskById } = require('
const router = require('express').Router();
// To get all the tasks
router.get('/', fetchAllTasks);
// To create a task
router.post('/', createTask);
// To update a task
router.put('/:id', updateTaskById);
// To delete a task
router.delete('/:id', deleteTaskById);
module.exports = router;
main*
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript
Go Live

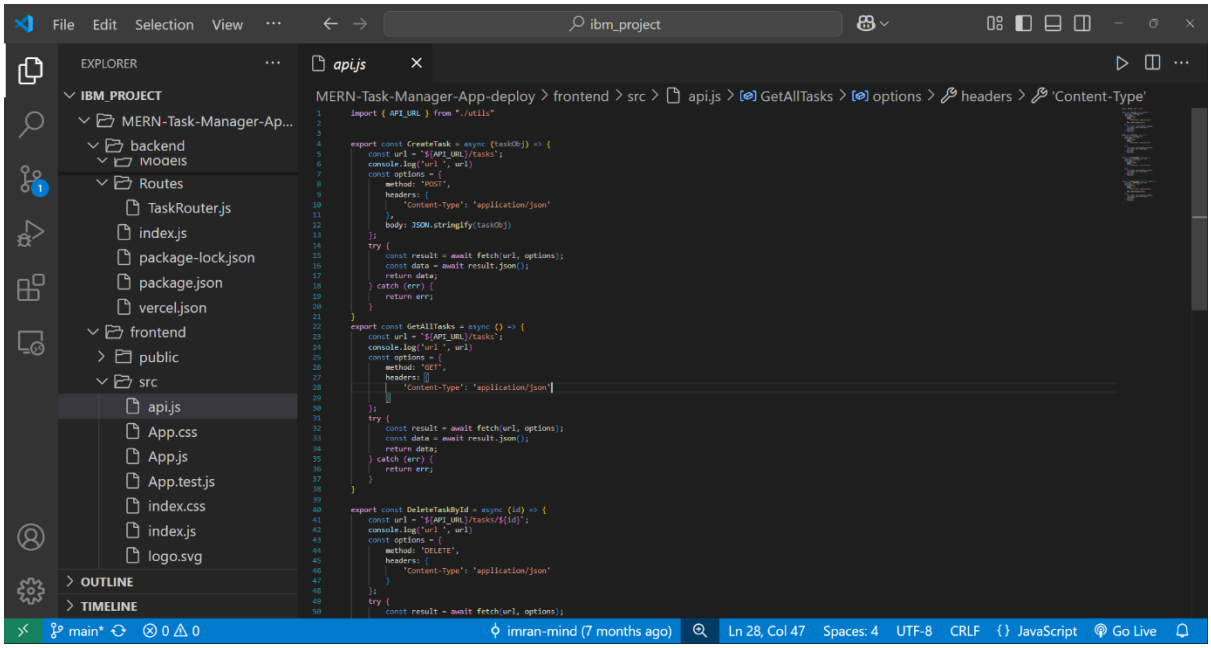

File Edit Selection View
ibm_project
EXPLORER
IBM_PROJECT
MERN-Task-Manager-Ap...
backend
Controllers
TaskController.js
Models
db.js
Routes
TaskRouter.js
index.js
package-lock.json
package.json
vercel.json
frontend
.gitignore
README.md
OUTLINE
TIMELINE
index.js
MERN-Task-Manager-App-deploy > backend > index.js > ...
const express = require('express');
const app = express();
require('dotenv').config();
require('./Models/db');
const PORT = process.env.PORT || 8080;
const TaskRouter = require('./Routes/TaskRouter');
const bodyParser = require('body-parser');
const cors = require('cors');
app.get('/', (req, res) => {
res.send('Hello from the server');
});
app.use(cors())
app.use(bodyParser.json());
app.use('/tasks', TaskRouter)
app.listen(PORT, () => {
console.log(`Server is running on PORT=${PORT}`);
})
main*
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript
Go Live

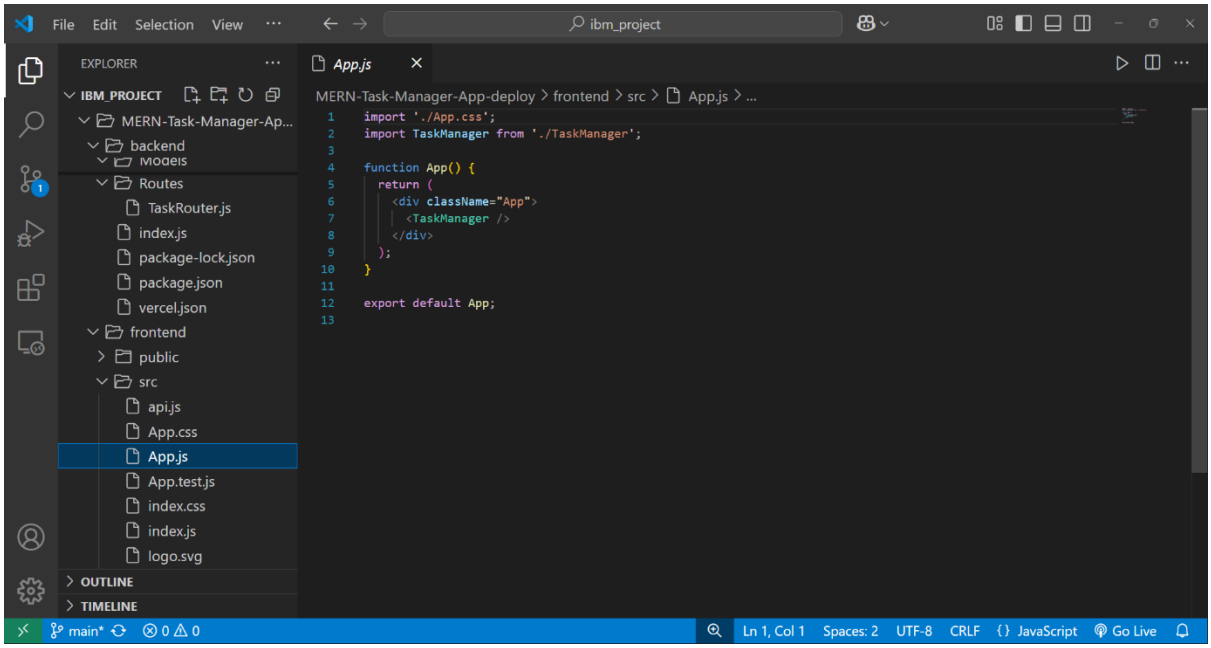

File Edit Selection View
ibm_project
EXPLORER
IBM_PROJECT
MERN-Task-Manager-Ap...
backend
Controllers
TaskController.js
Models
db.js
Routes
TaskRouter.js
index.js
package-lock.json
package.json
vercel.json
frontend
.gitignore
README.md
OUTLINE
TIMELINE
package-lock.json
MERN-Task-Manager-App-deploy > backend > package-lock.json > ...
main*
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
JSON
Go Live


File Edit Selection View
ibm_project
EXPLORER
IBM_PROJECT
MERN-Task-Manager-Ap...
backend
Controllers
TaskController.js
Models
db.js
Routes
TaskRouter.js
index.js
package-lock.json
package.json
vercel.json
frontend
.gitignore
README.md
OUTLINE
TIMELINE
package.json
MERN-Task-Manager-App-deploy > backend > package.json > ...
{
"name": "backend",
"version": "1.0.0",
"description": "",
"main": "index.js",
Debug
"scripts": {
"start": "nodemon index.js"
},
"keywords": [],
"author": "",
"license": "ISC",
"dependencies": {
"body-parser": "^1.20.2",
"cors": "^2.8.5",
"dotenv": "^16.4.5",
"express": "^4.19.2",
"mongoose": "^8.4.3"
}
}
main*
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
JSON
Go Live



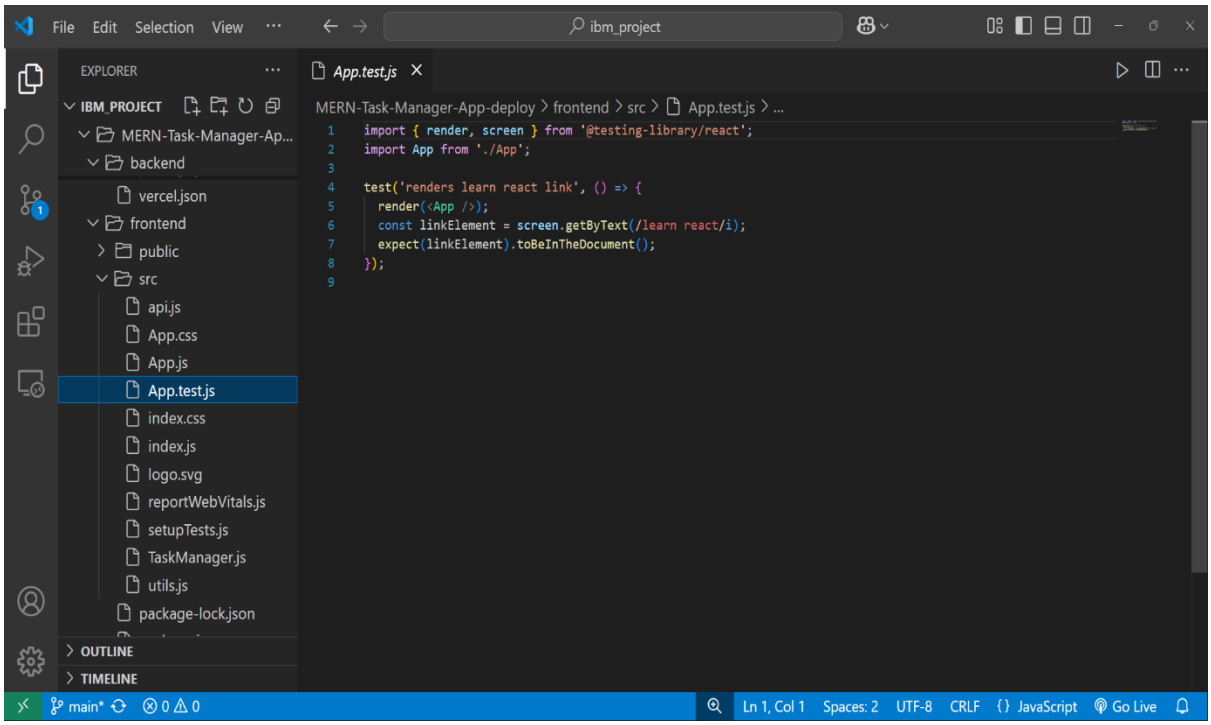
Front-end code:

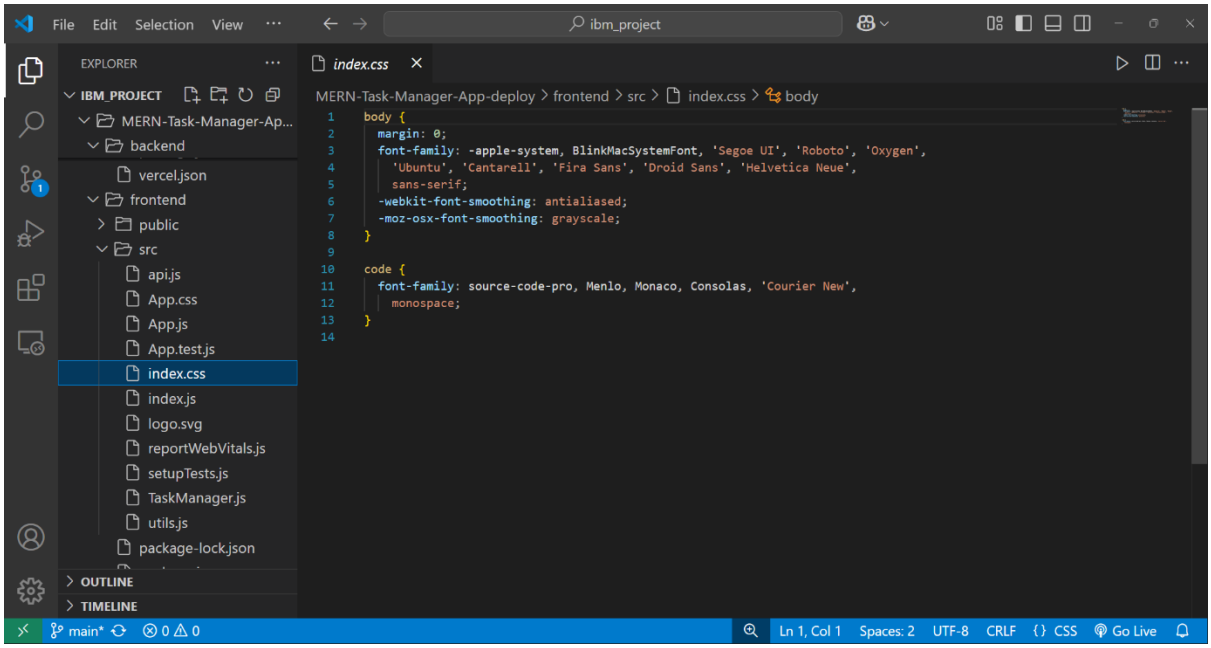


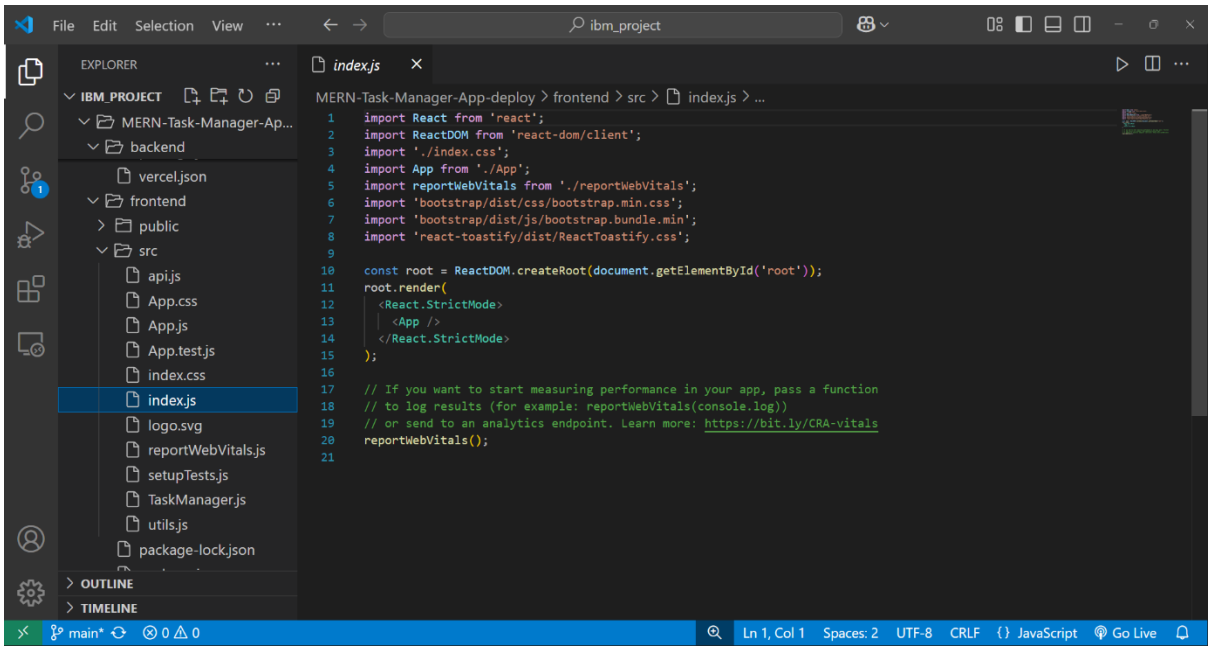

File Edit Selection View
ibm_project
EXPLORER
IBM_PROJECT
MERN-Task-Manager-Ap...
backend
Models
Routes
TaskRouter.js
index.js
package-lock.json
package.json
vercel.json
frontend
public
src
api.js
App.css
App.js
App.test.js
index.css
index.js
logo.svg
OUTLINE
TIMELINE
App.js
MERN-Task-Manager-App-deploy > frontend > src > App.js > ...
import ./App.css';
import TaskManager from ./TaskManager';
function App() {
return (

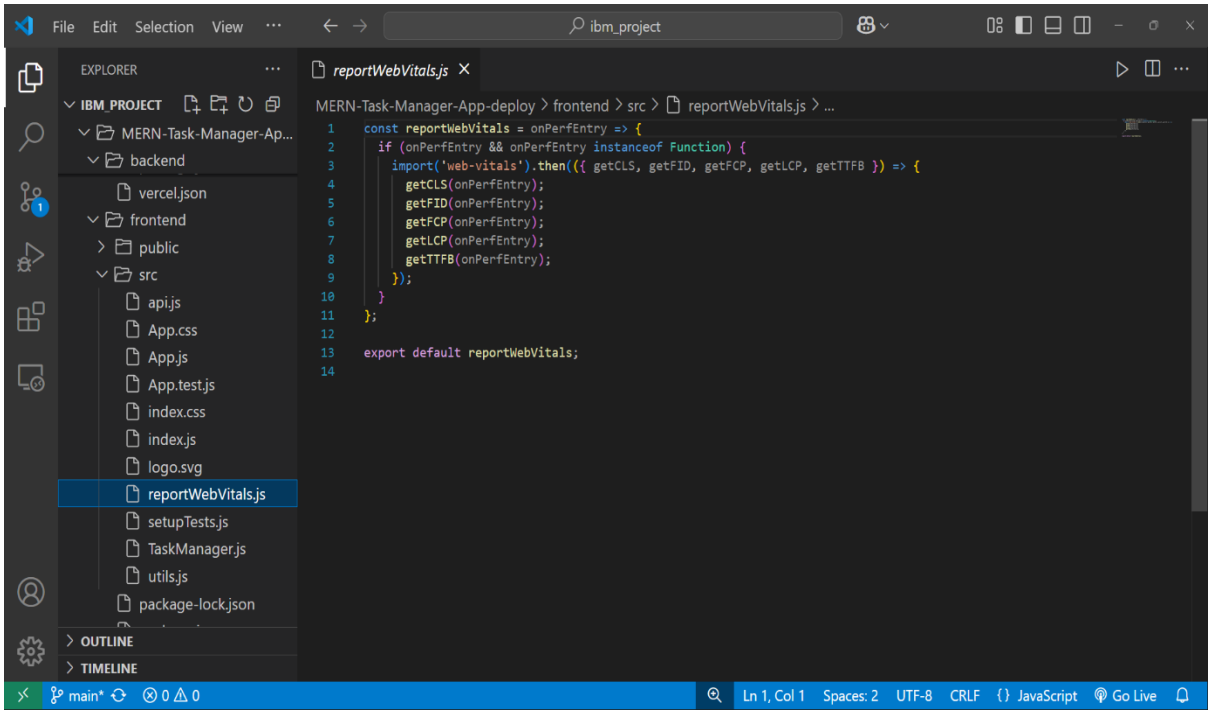
<TaskManager />

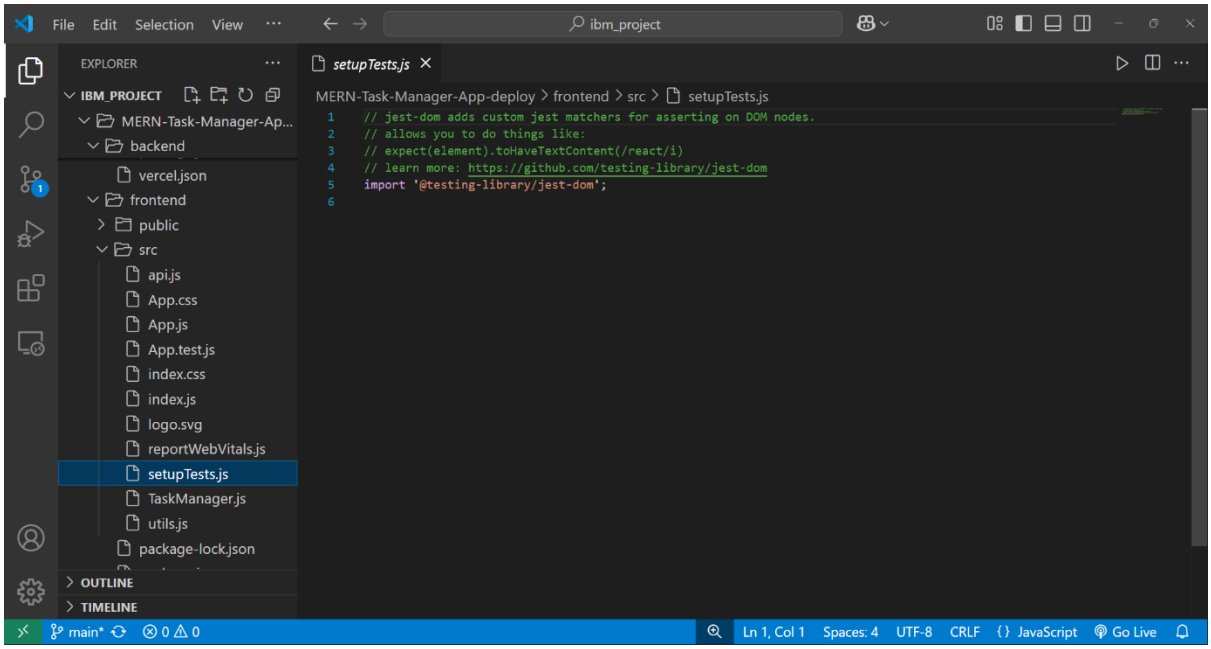
);
}
export default App;
main*
0 0
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
JavaScript
Go Live

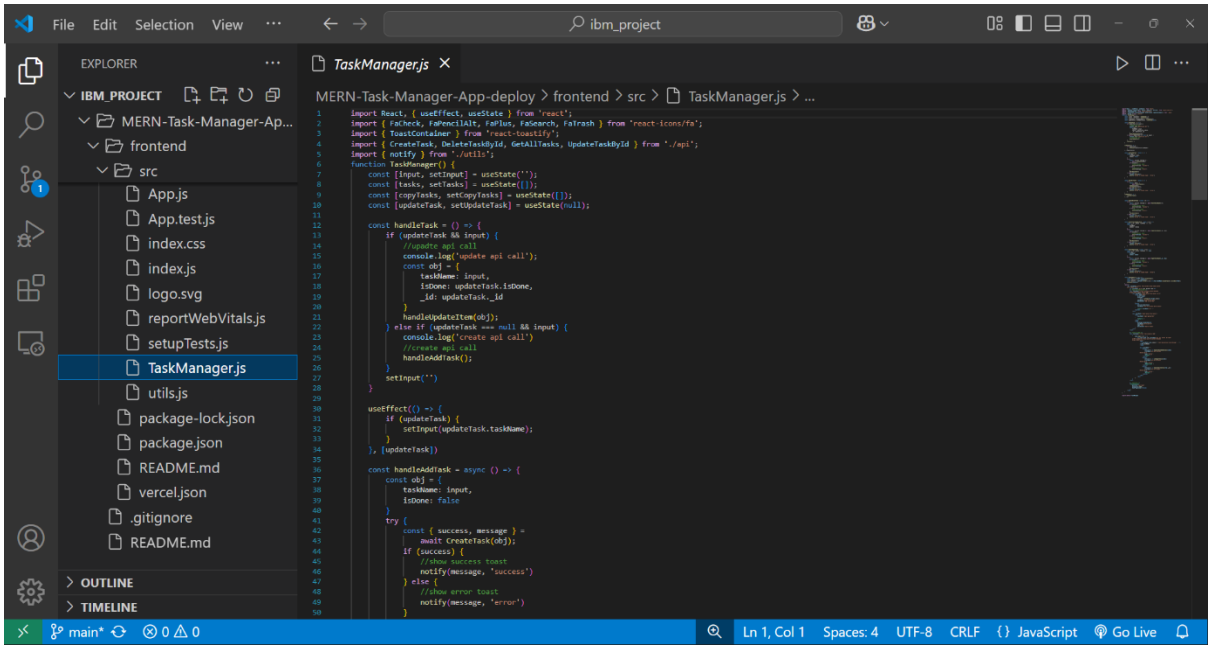

File Edit Selection View
ibm_project
EXPLORER
IBM_PROJECT
MERN-Task-Manager-Ap...
backend
vercel.json
frontend
public
src
api.js
App.css
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
TaskManager.js
utils.js
package-lock.json
OUTLINE
TIMELINE
App.test.js
MERN-Task-Manager-App-deploy > frontend > src > App.test.js > ...
import { render, screen } from '@testing-library/react';
import App from ./App';
test('renders learn react link', () => {
render(<App />);
const linkElement = screen.getByText(/learn react/i);
expect(linkElement).toBeInTheDocument();
});
main*
0 0
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
JavaScript
Go Live

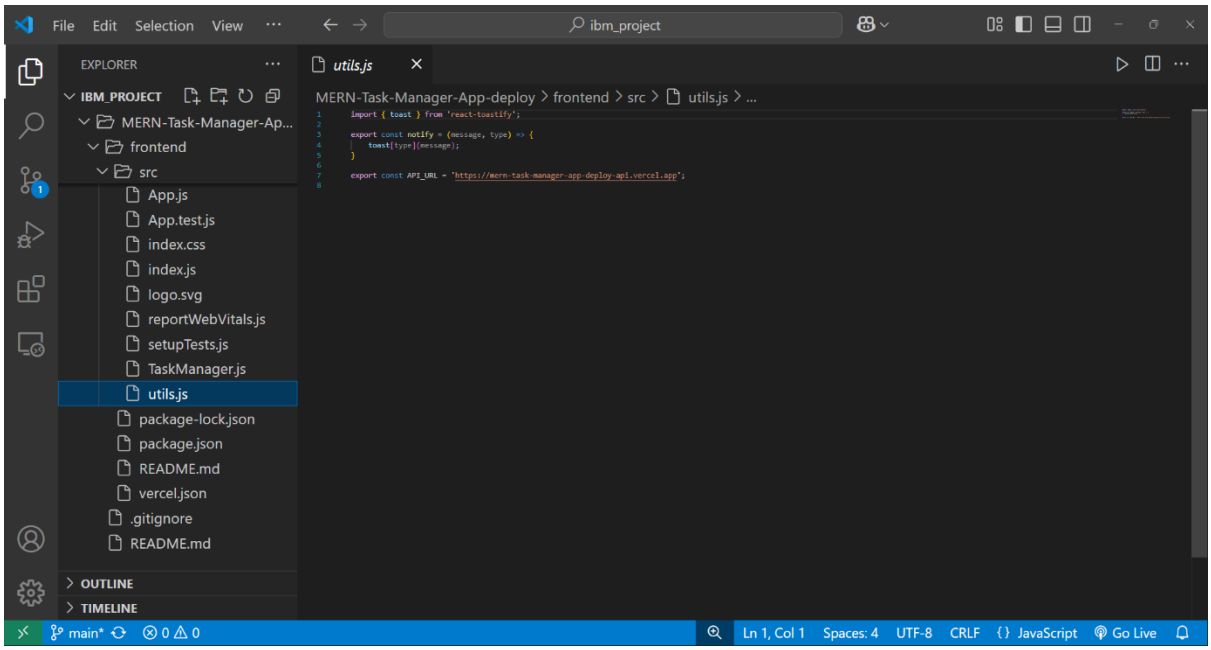

File Edit Selection View
ibm_project
EXPLORER
IBM_PROJECT
MERN-Task-Manager-Ap...
backend
vercel.json
frontend
public
src
api.js
App.css
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
TaskManager.js
utils.js
package-lock.json
OUTLINE
TIMELINE
index.css
MERN-Task-Manager-App-deploy > frontend > src > index.css > body
body {
margin: 0;
font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
sans-serif;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
code {
font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
monospace;
}
main*
0 0
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
CSS
Go Live


File Edit Selection View
ibm_project
EXPLORER
IBM_PROJECT
MERN-Task-Manager-Ap...
backend
vercel.json
frontend
public
src
api.js
App.css
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
TaskManager.js
utils.js
package-lock.json
OUTLINE
TIMELINE
index.js
MERN-Task-Manager-App-deploy > frontend > src > index.js > ...
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap/dist/js/bootstrap.bundle.min';
import 'react-toastify/dist/ReactToastify.css';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
<React.StrictMode>
<App />
</React.StrictMode>
);
// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
main*
0 0
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
JavaScript
Go Live

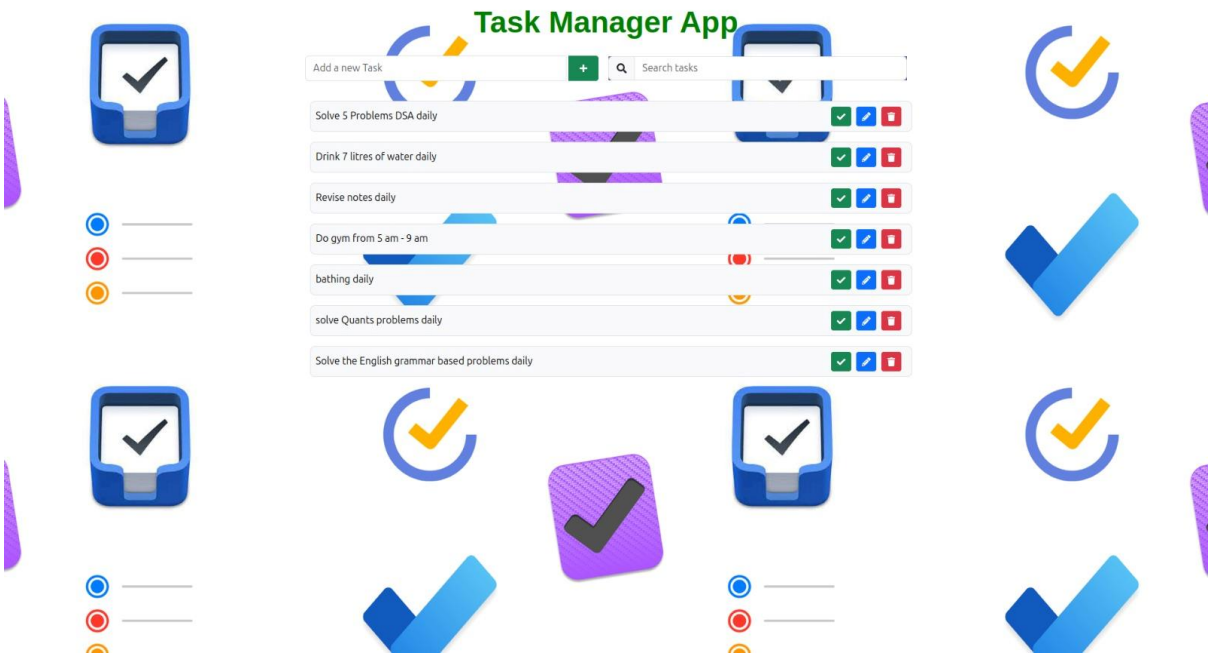

File Edit Selection View
ibm_project
EXPLORER
IBM_PROJECT
MERN-Task-Manager-Ap...
backend
vercel.json
frontend
public
src
api.js
App.css
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
TaskManager.js
utils.js
package-lock.json
OUTLINE
TIMELINE
reportWebVitals.js
MERN-Task-Manager-App-deploy > frontend > src > reportWebVitals.js > ...
const reportWebVitals = onPerfEntry => {
if (onPerfEntry && onPerfEntry instanceof Function) {
import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
getCLS(onPerfEntry);
getFID(onPerfEntry);
getFCP(onPerfEntry);
getLCP(onPerfEntry);
getTTFB(onPerfEntry);
});
}
};
export default reportWebVitals;
main*
0 0
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
JavaScript
Go Live

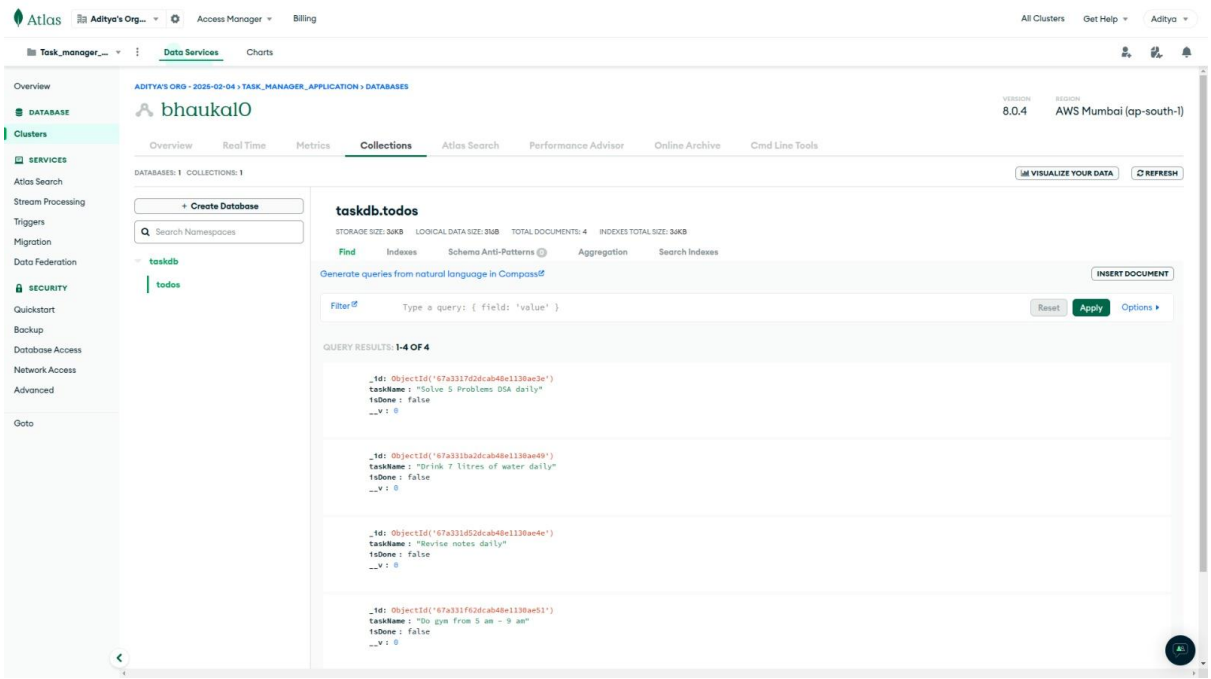

File Edit Selection View
ibm_project
EXPLORER
IBM_PROJECT
MERN-Task-Manager-Ap...
backend
vercel.json
frontend
public
src
api.js
App.css
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
TaskManager.js
utils.js
package-lock.json
OUTLINE
TIMELINE
setupTests.js
MERN-Task-Manager-App-deploy > frontend > src > setupTests.js
// jest-dom adds custom jest matchers for asserting on DOM nodes.
// allows you to do things like:
// expect(element).toHaveTextContent(/react/i)
// learn more: https://github.com/testing-library/jest-dom
import '@testing-library/jest-dom';
main*
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript
Go Live


File Edit Selection View
ibm_project
EXPLORER
IBM_PROJECT
MERN-Task-Manager-Ap...
frontend
src
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
TaskManager.js
utils.js
package-lock.json
package.json
README.md
vercel.json
.gitignore
README.md
OUTLINE
TIMELINE
TaskManager.js
MERN-Task-Manager-App-deploy > frontend > src > TaskManager.js > ...
import React, { useEffect, useState } from 'react';
import { FaCheck, FaPencilAlt, FaPlus, FaSearch, FaTrash } from 'react-icons/fa';
import { ToastContainer } from 'react-toastify';
import { CreateTask, DeleteTaskById, GetAllTasks, UpdateTaskById } from './api';
import { notify } from './utils';
function TaskManager() {
const [input, setInput] = useState('');
const [tasks, setTasks] = useState([]);
const [copyTasks, setCopyTasks] = useState([]);
const [updateTask, setUpdateTask] = useState(null);
const handleTask = () => {
if (updateTask && input) {
//update api call
console.log('update api call');
const obj = {
taskName: input,
isDone: updateTask.isDone,
_id: updateTask._id
}
handleUpdateItem(obj);
} else if (updateTask === null && input) {
console.log('create api call')
//create api call
handleAddTask();
}
setInput('')
}
useEffect(() => {
if (updateTask) {
setInput(updateTask.taskName);
}
}, [updateTask])
const handleAddTask = async () => {
const obj = {
taskName: input,
isDone: false
}
try {
const { success, message } =
await CreateTask(obj);
if (success) {
//show success toast
notify(message, 'success')
} else {
//show error toast
notify(message, 'error')
}
main*
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript
Go Live


File Edit Selection View
ibm_project
EXPLORER
IBM_PROJECT
MERN-Task-Manager-Ap...
frontend
src
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
TaskManager.js
utils.js
package-lock.json
package.json
README.md
vercel.json
.gitignore
README.md
OUTLINE
TIMELINE
utils.js
MERN-Task-Manager-App-deploy > frontend > src > utils.js > ...
import { toast } from 'react-toastify';
export const notify = (message, type) => {
toast[type](message);
}
export const API_URL = 'https://mern-task-manager-app-deploy-api.vercel.app';
main*
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript
Go Live

Progress


Task Manager App
Add a new Task
Search tasks
Solve 5 Problems DSA daily
Drink 7 litres of water daily
Revise notes daily
Do gym from 5 am - 9 am
bathing daily
solve Quants problems daily
Solve the English grammar based problems daily


Atlas
Aditya's Org...
Access Manager
Billing
All Clusters
Get Help
Aditya
Task_manager_...
Data Services
Charts
Overview
DATABASE
Clusters
SERVICES
Atlas Search
Stream Processing
Triggers
Migration
Data Federation
SECURITY
Quickstart
Backup
Database Access
Network Access
Advanced
Goto
ADITYA'S ORG - 2025-02-04 › TASK_MANAGER_APPLICATION › DATABASES
bhaukal0
VERSION 8.0.4
REGION AWS Mumbai (ap-south-1)
Overview
Real Time
Metrics
Collections
Atlas Search
Performance Advisor
Online Archive
Cmd Line Tools
DATABASES: 1 COLLECTIONS: 1
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
taskdb
todos
taskdb.todos
STORAGE SIZE: 36KB
LOGICAL DATA SIZE: 31kB
TOTAL DOCUMENTS: 4
INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
Generate queries from natural language in Compass
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
QUERY RESULTS: 1-4 OF 4
_id: ObjectId('67a3317d2dcab48e1130ae3e')
taskName : "Solve 5 Problems DSA daily"
isDone : false
__v : 0
_id: ObjectId('67a331ba2dcab48e1130ae49')
taskName : "Drink 7 litres of water daily"
isDone : false
__v : 0
_id: ObjectId('67a331d52dcab48e1130ae4e')
taskName : "Revise notes daily"
isDone : false
__v : 0
_id: ObjectId('67a331f62dcab48e1130ae51')
taskName : "Do gym from 5 am - 9 am"
isDone : false
__v : 0