**Ce que vous avez fait ? Comment ? Pourquoi ? Avec quels outils et méthodes ?**

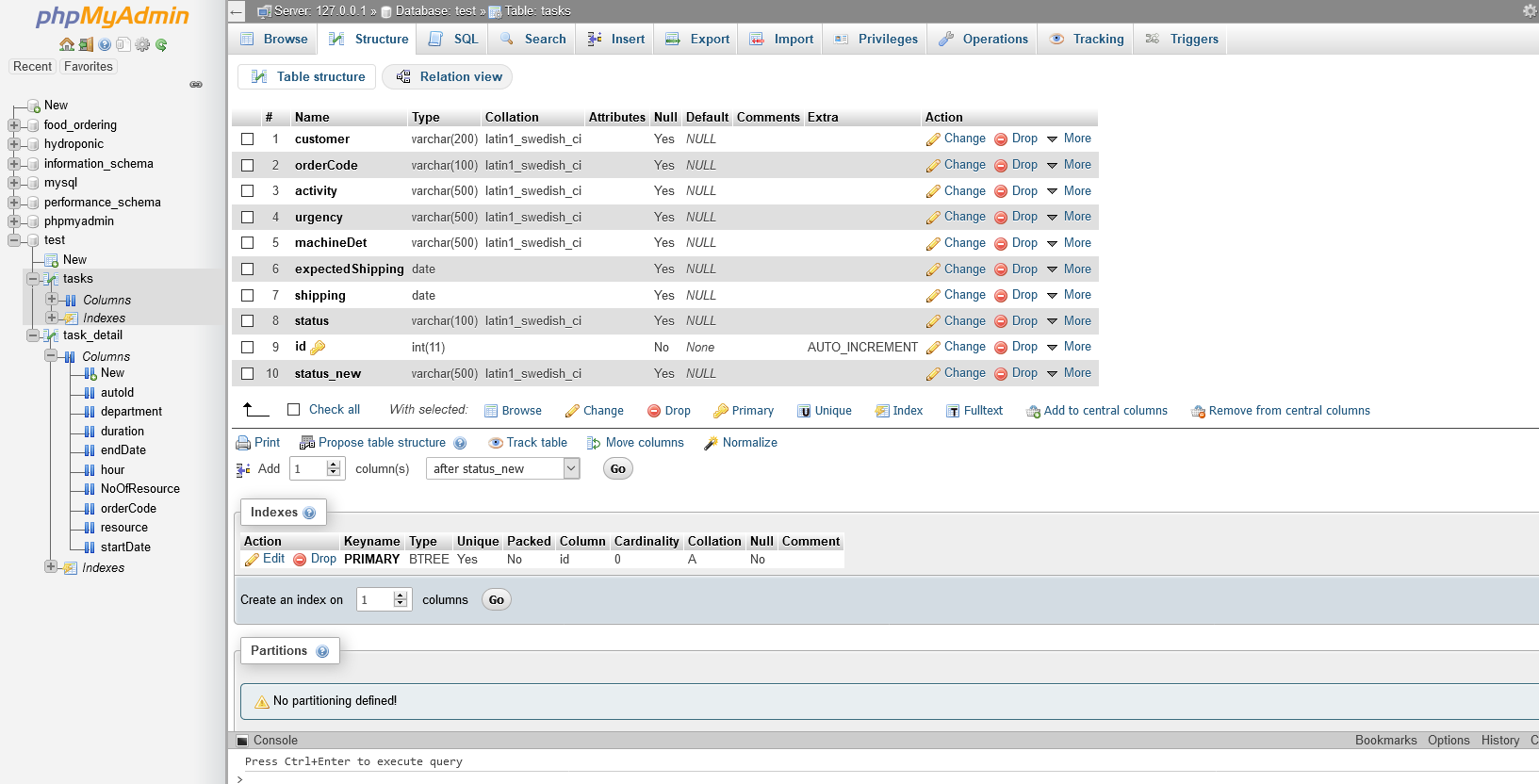
**What did you do? How? 'Or' What ? Why ? With what tools and methods?**

As we need to develop a Project Task Management App, according to the requirements we need to develop the front hand, backend and the database for saving.

So, for starters we create two tables, one master of Task and one transaction for task activities. One table consists of columns which are going to be saved once and in another table there is a reference of master and task activities details.

We use a MYSQL database which can be executed using XAMPP, phpMyAdmin. After creating the database and database table, first we create the table name **TASK** we use the following columns:

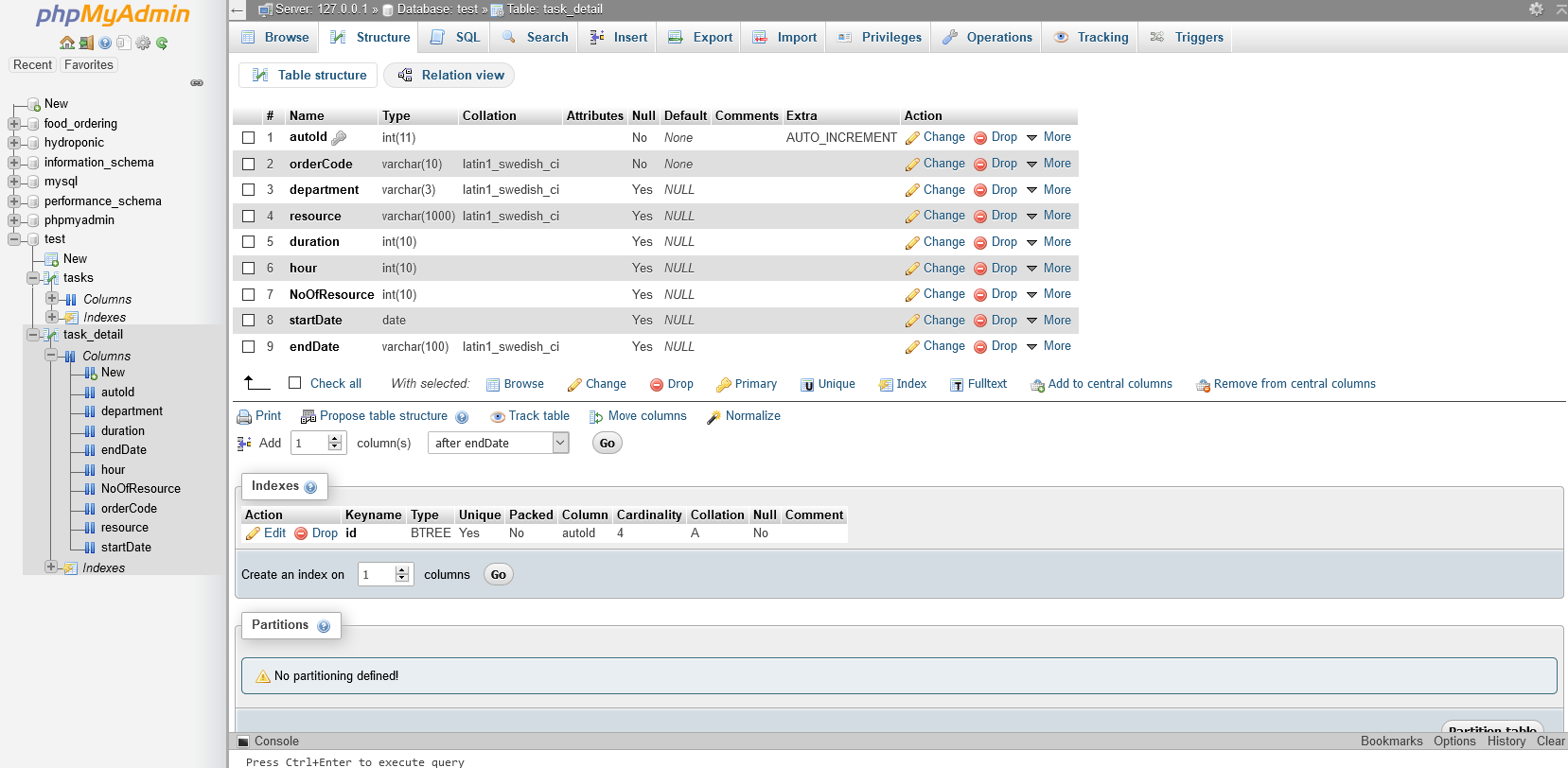
1. ID
2. Status
3. orderCode
4. Urgency
5. machineDet
6. expected Shipping
7. Shipping
8. Customer
9. Activity
10. Status\_new



Here ID is a primary key and is an auto increment which is handled by the database. The other fields are the basic information where we save the master data of Project.

Then we create a transaction table called **TASK DETAIL** with following table

1. autoID
2. orderCode
3. Department
4. Resource
5. Duration
6. Hour
7. No Of Resource
8. Start date
9. End date



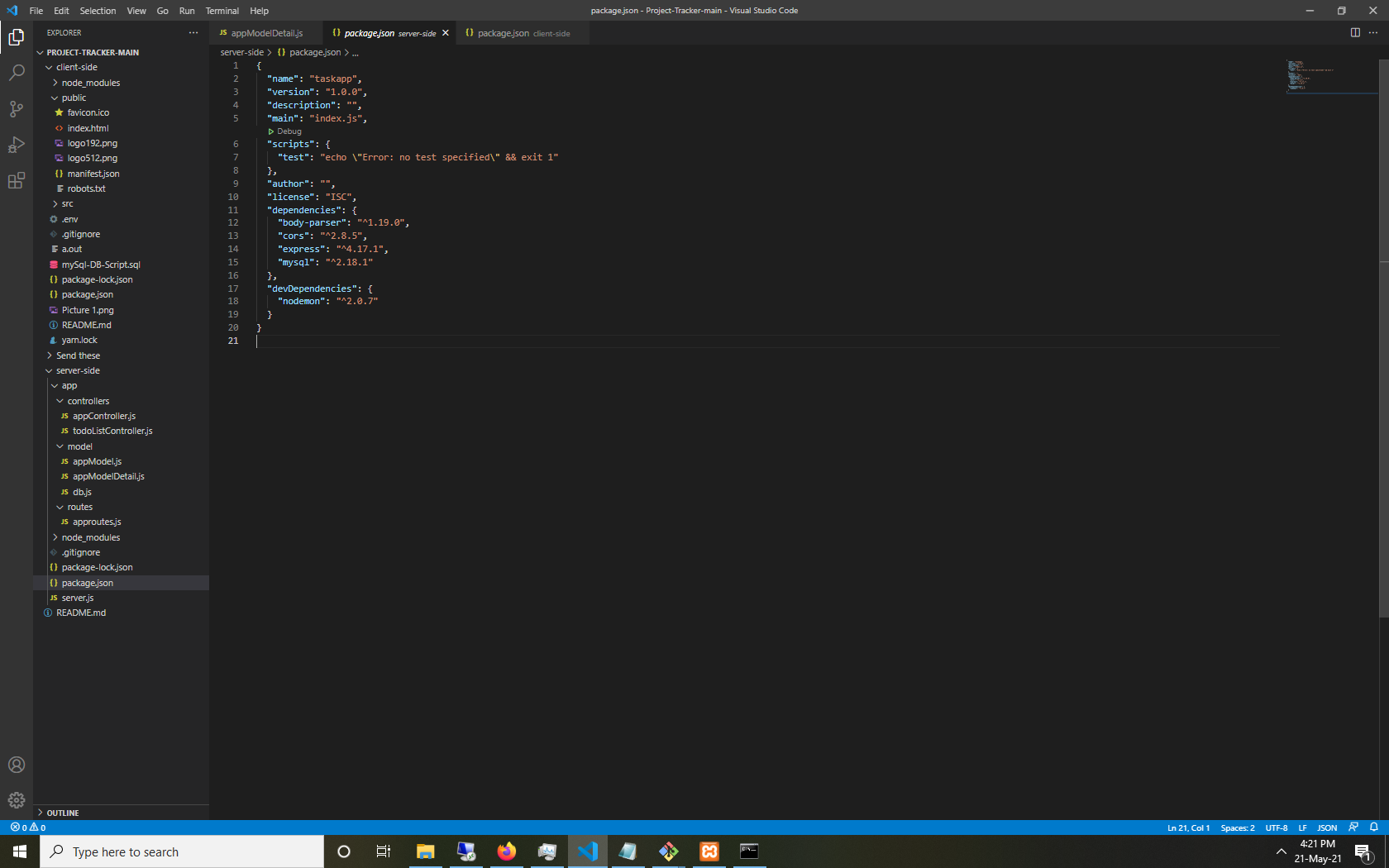
In this table autoID is the primary key and is auto incremented by database. Here the data saved in this table are stored and all the task activities are referenced in the master table. We store all the details regarding the task.

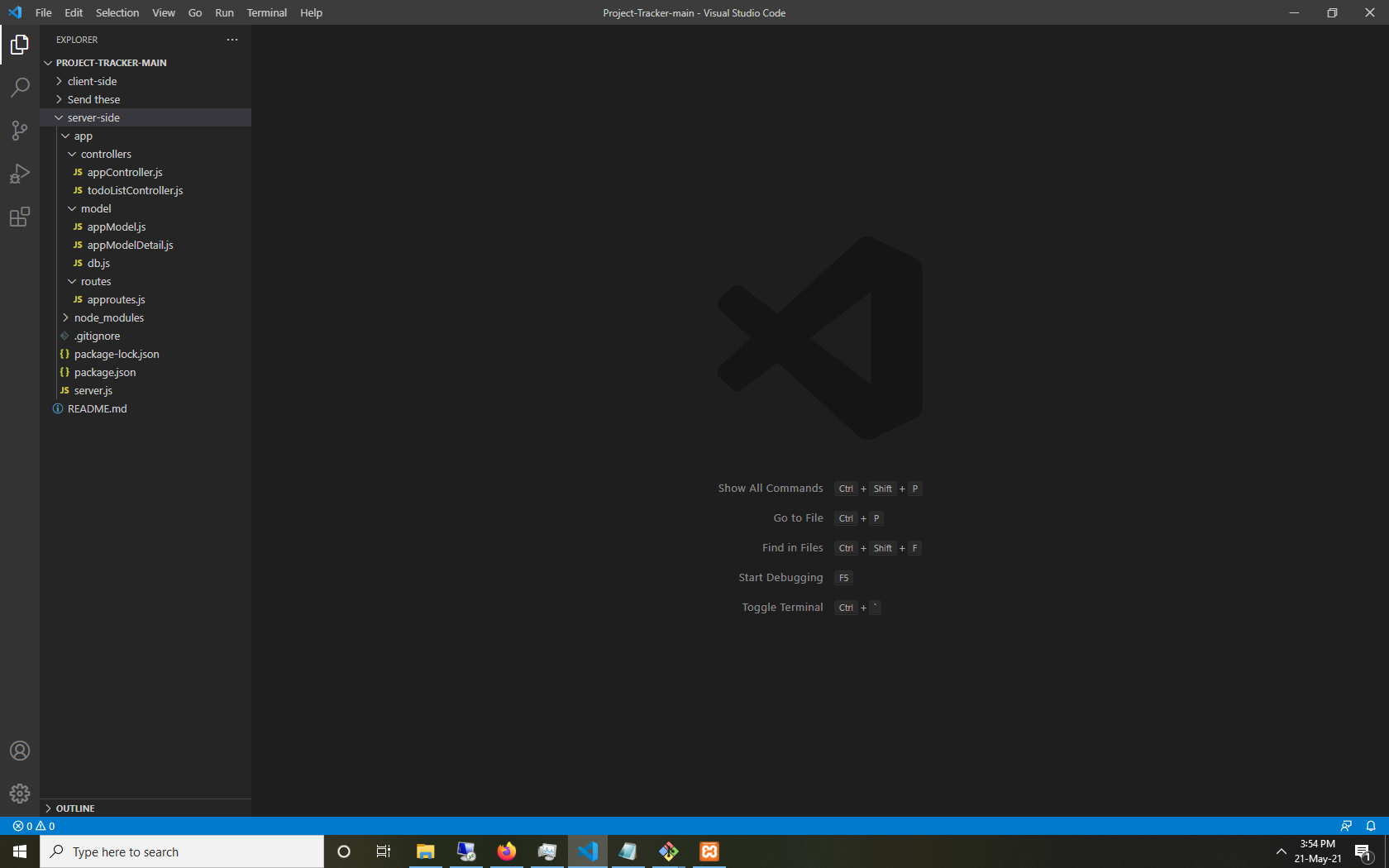
After creating the database. We will be designing our server side which are also called APIs. We are using the following technologies:

1. Node Js
2. Express Js
3. Body-parser
4. Cors
5. MySql

We use NPM Package Manager for installing dependencies.

This Below package.json file belongs to the server side.





Node.js is primarily used for non-blocking, event driver servers due to its single threaded nature. Here non blocking means that the operations couldnt be blocked if one operation is being processed, other operations could be performed. Non-blocking methods are executed asynchronously. In node, non-blocking usually refers to I/O Operations.

Express js is a pre-built Node JS framework that helps in creating server-side web applications. Simplicity, Minimalism, Flexibility, Scalability are some characteristics.

Body-Parser to handle HTTP Post requests in Express JS you need a middleware. Body-parser extracts the entire body portion of incoming request stream and exposes it on req.body. It also parses the JSON, buffer, string and URL Encoded Data submitted using HTTP POST request.

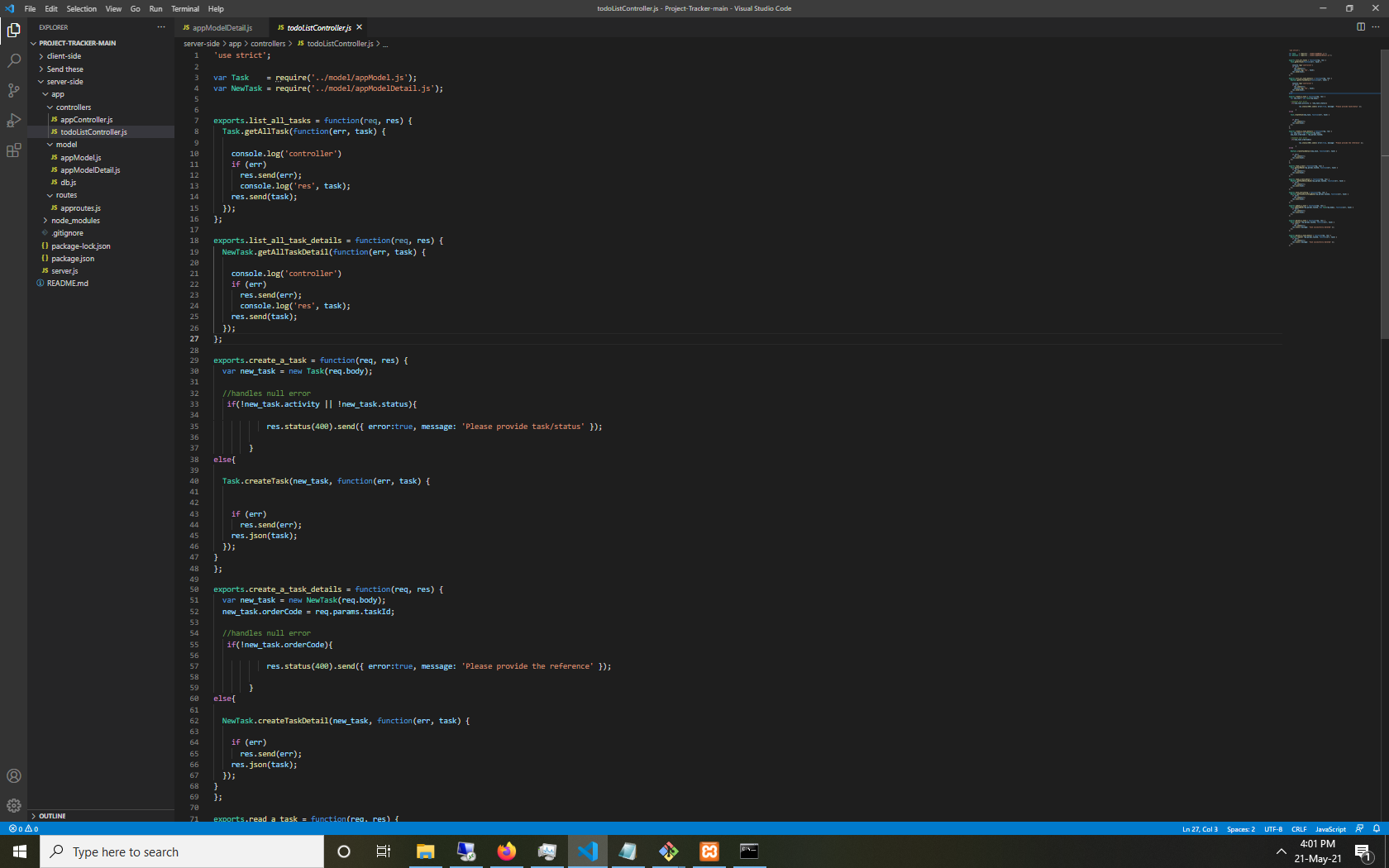
CORS, Cross Origin Resource Sharing, allows you to make requests from one website to another or in our example from Front End to Backend which is normally prohibited by another browser policy called the Same-Origin Policy (SOP).

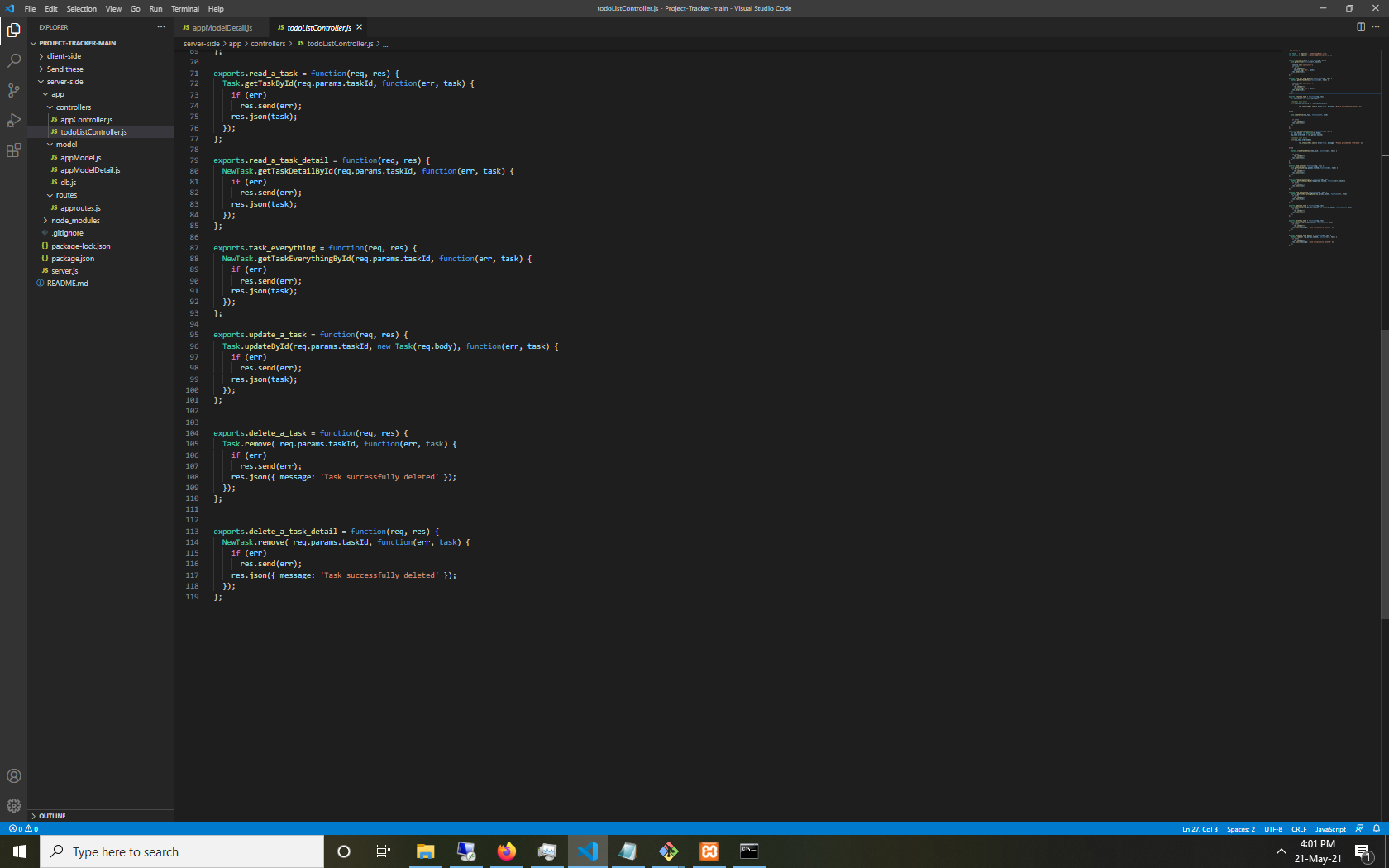
MySQL is a popular relational database and open source, easy to use and has good documentation. It is the best purpose for Web Database.

For building a server side we used the basic structure which consists of a Server js file which consists of Database connections and starting our backend server. It also has all our route details imported. Then we have an app folder which consists of the Controller, Model and Routes folder.

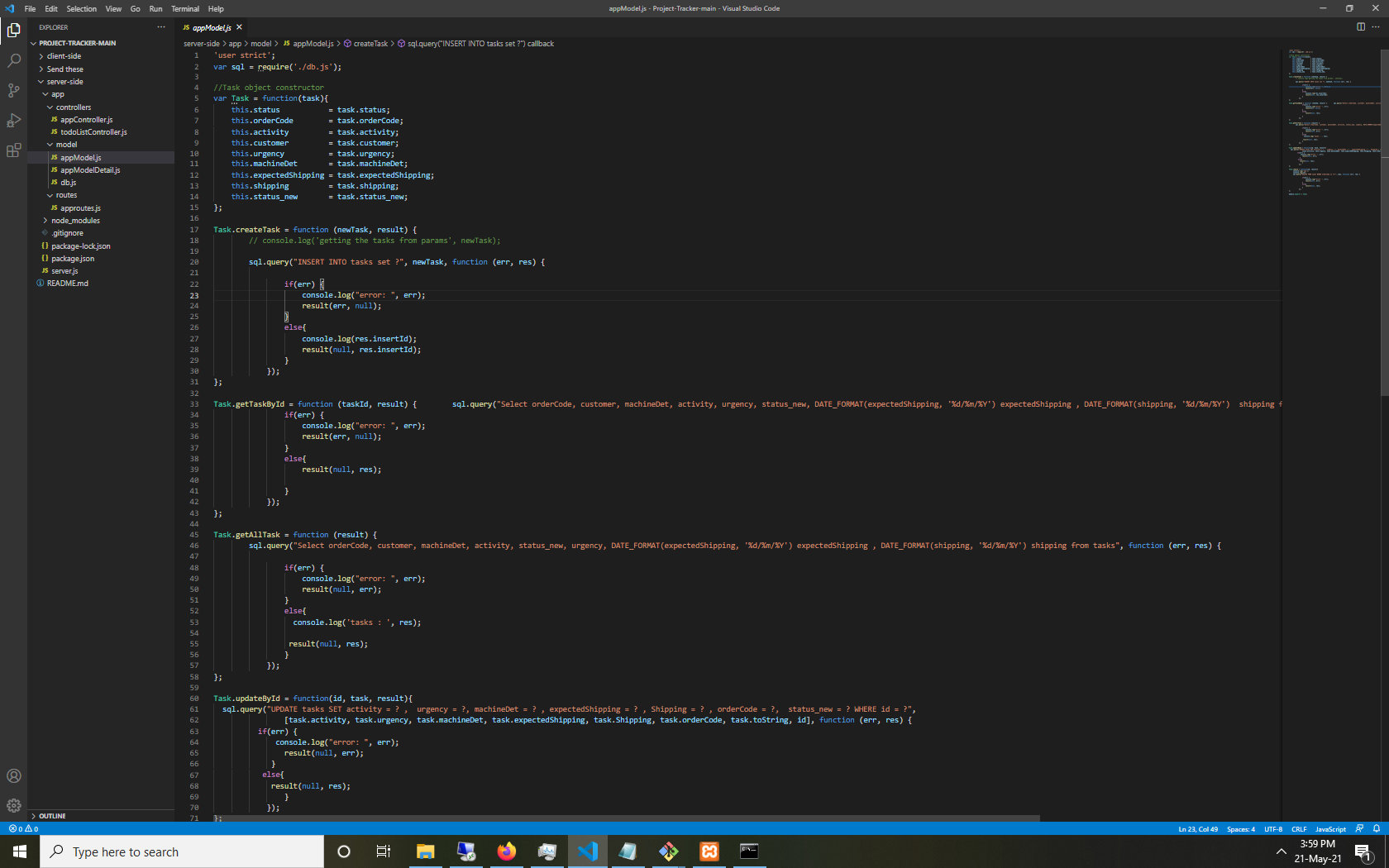
Quick overview about the above mentioned folders:

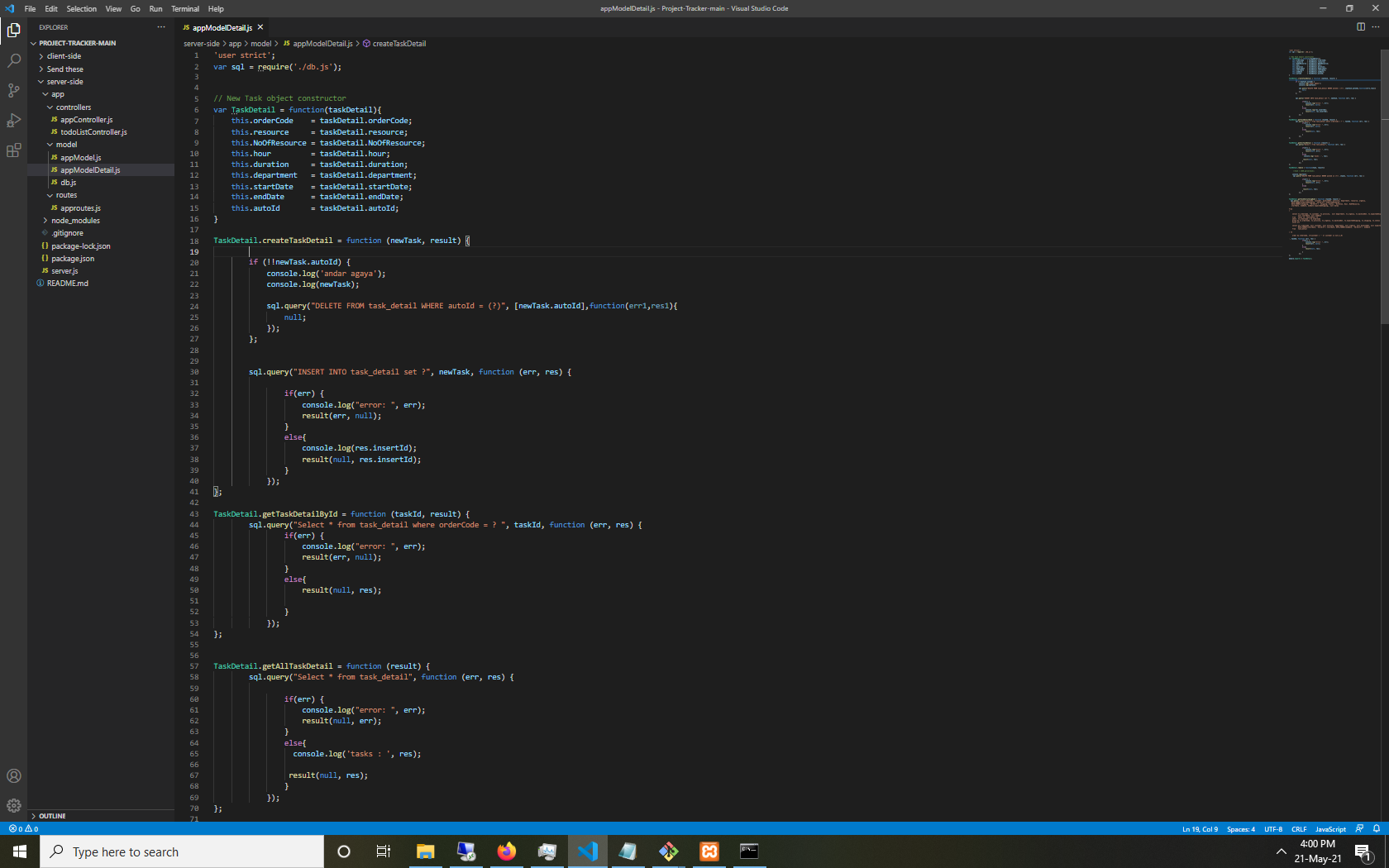
1. **Controller:** It gets all the requests from our model folder, then it creates an HTML page that displays all the data and returns it to the user to view in the browser. It is a part of MVC, the C here is the Controller Support. It controls the request and response of Model View. You add all the functionality and logic to develop our dynamic web applications. It takes data requests from views and sends it to the model and sends the response back.



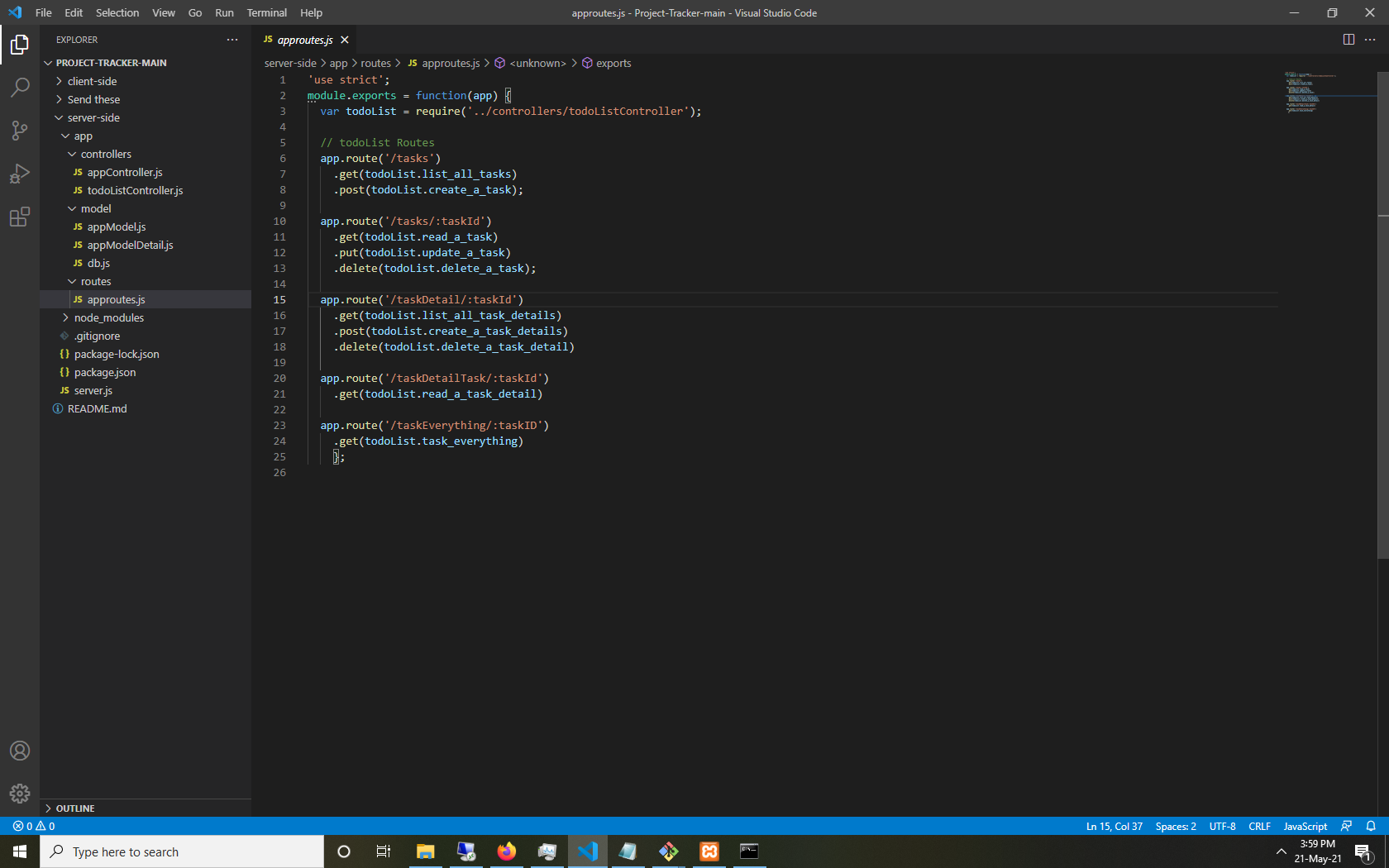


1. **Model:** We created two files in this folder appModel and appModelDetail, in appModel folder we add all the queries like create, insert, update and delete for the table Tasks and in appModelDetail we add queries for the table Task\_Detail of our database. These will be where all our schema or data models are required for the project.

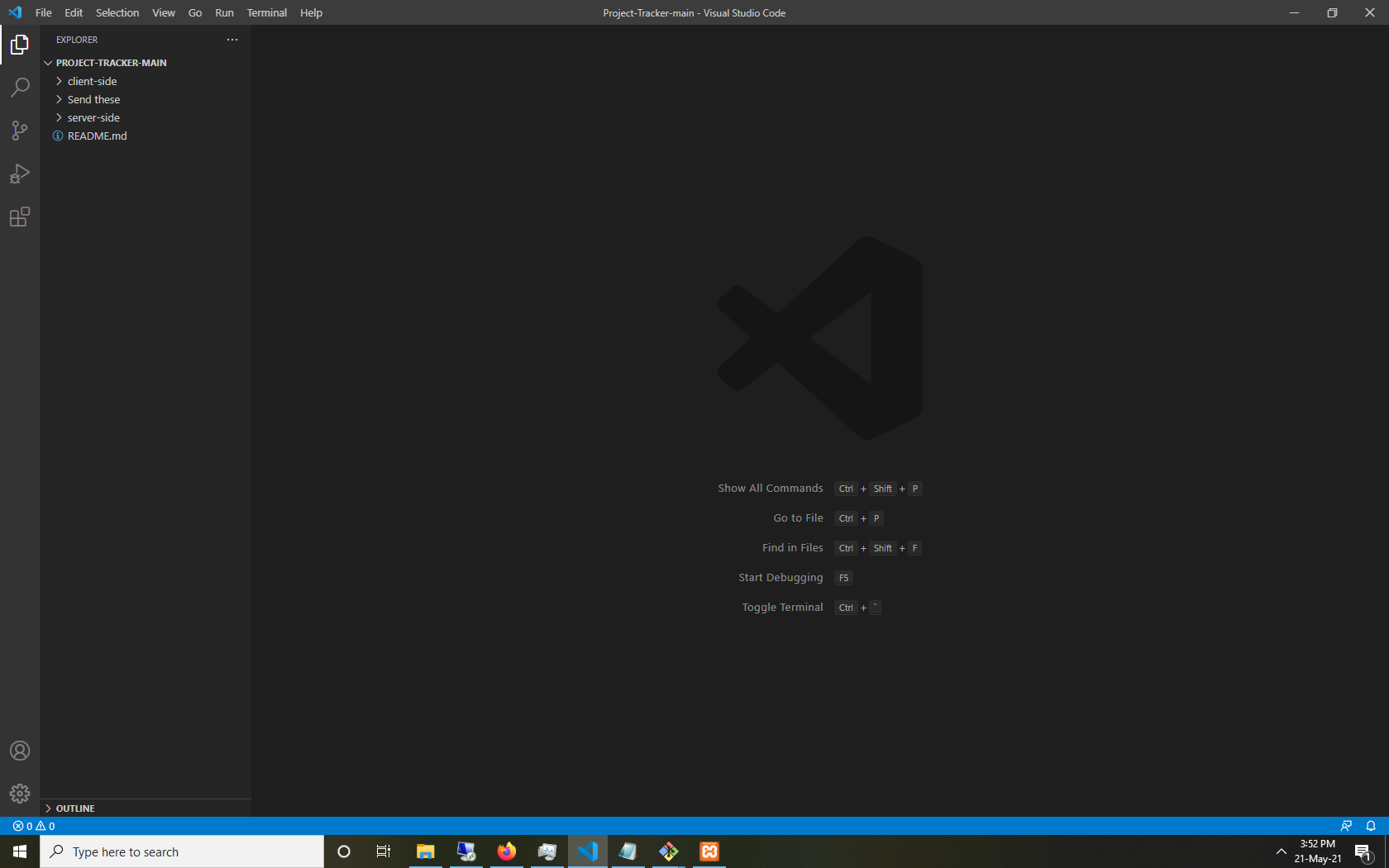




1. **Routes:** The word route itself is self explanatory, it consists of all the end points which our APIs are built and is connected to which controller will be executed when the API is requested. Any information encoded in request URLs to the appropriate controller functions. Every routing method has to specify callback functions or also called handler functions.



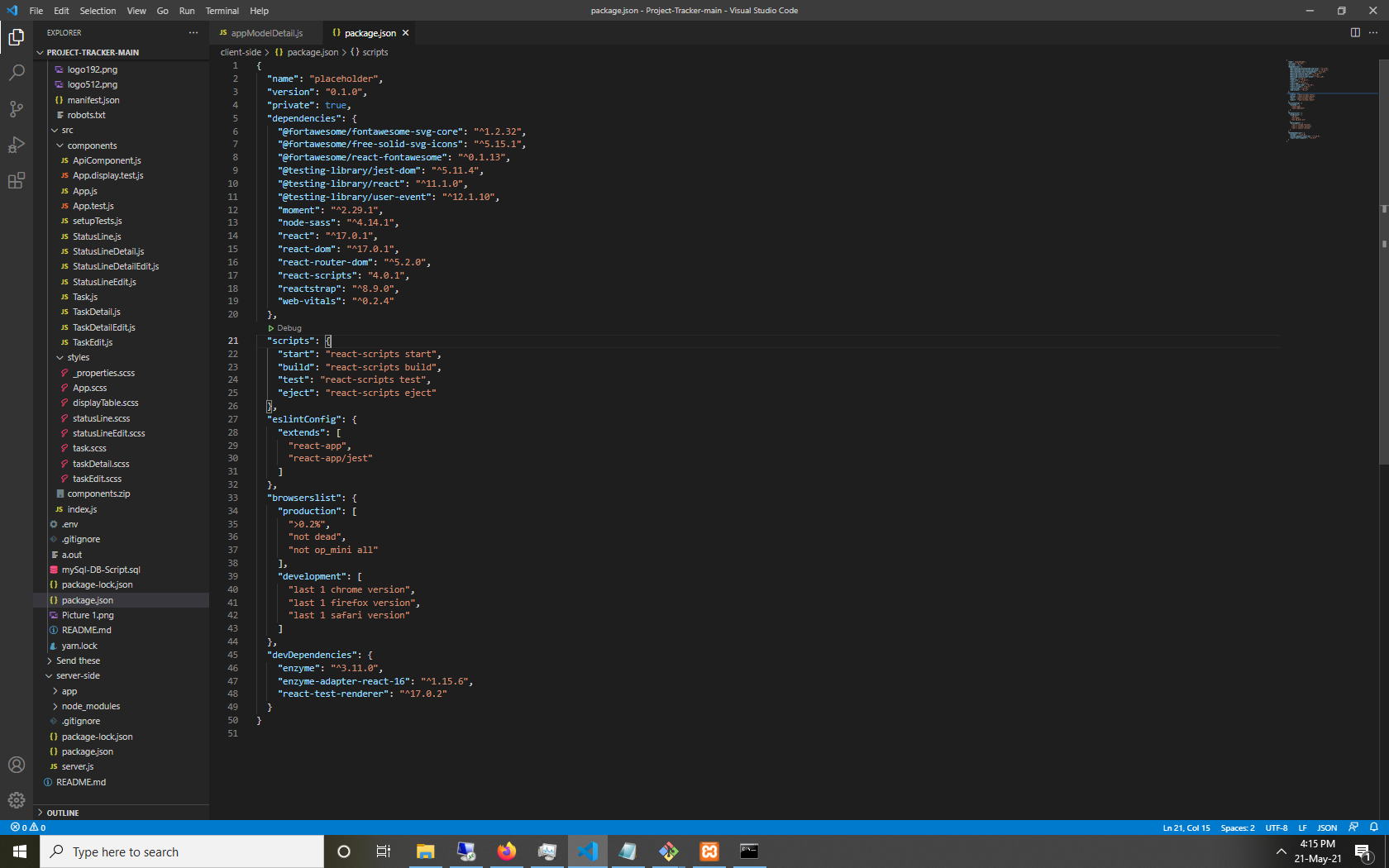
Now, for developing front end we will use Visual Studio Code for IDE, for front end we use React JS, and we use moment library for managing dates and calculation.



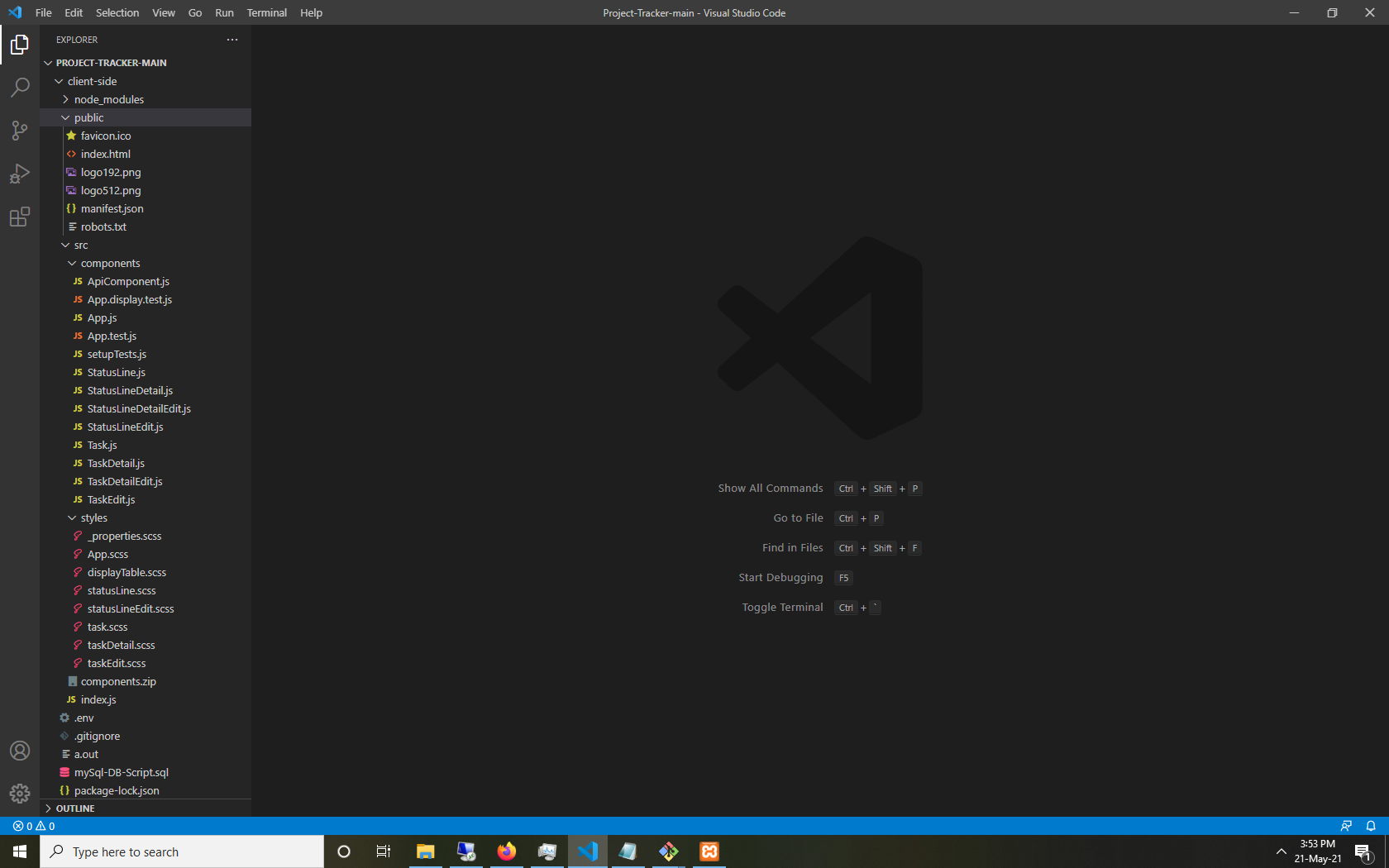
We start with the create-react-app syntax, this develops

Our folders/files consist of the following:

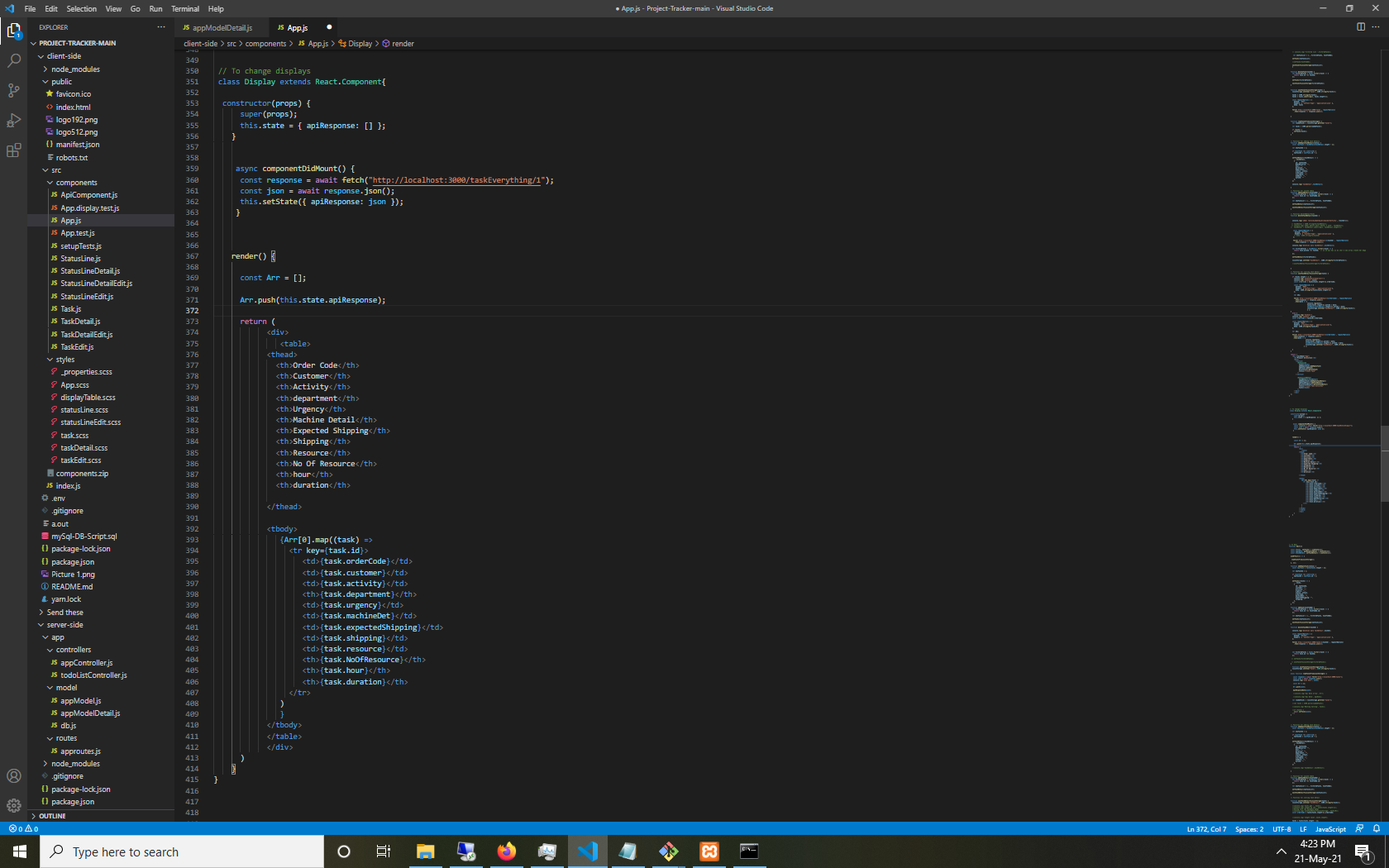
1. Root/Index.js:
2. Src
   1. Components
   2. Styles
3. Public
4. .env
5. Package.json



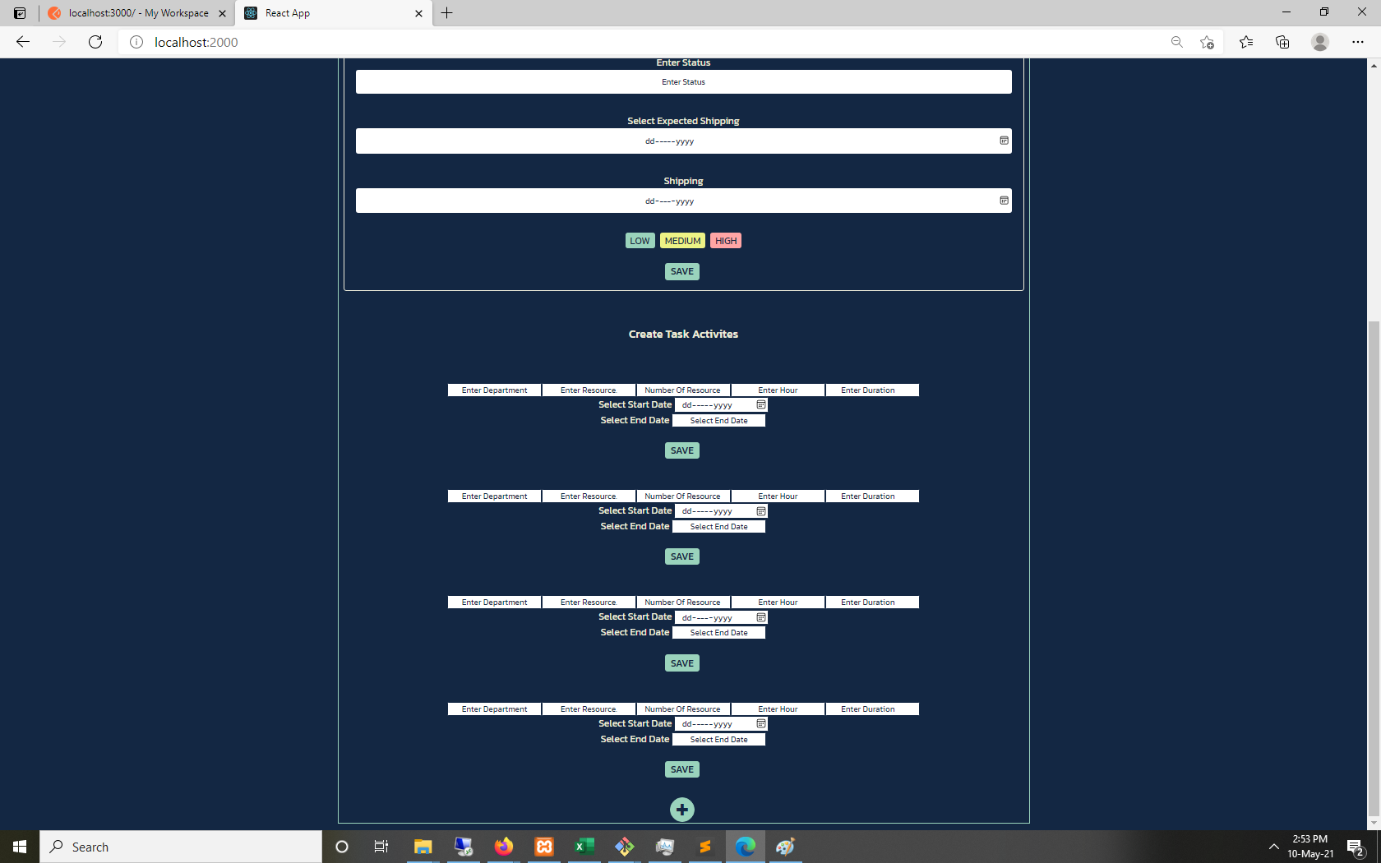
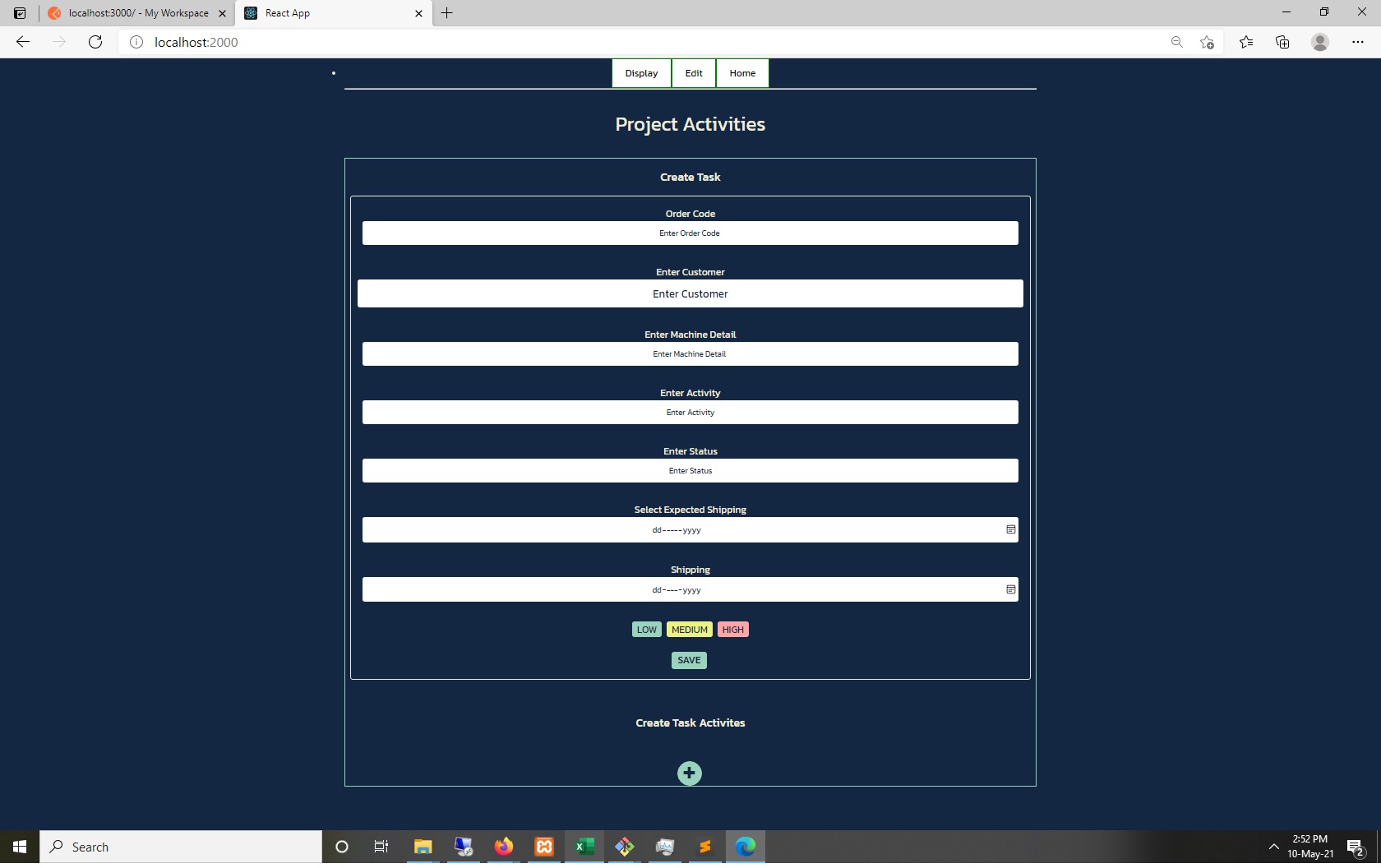
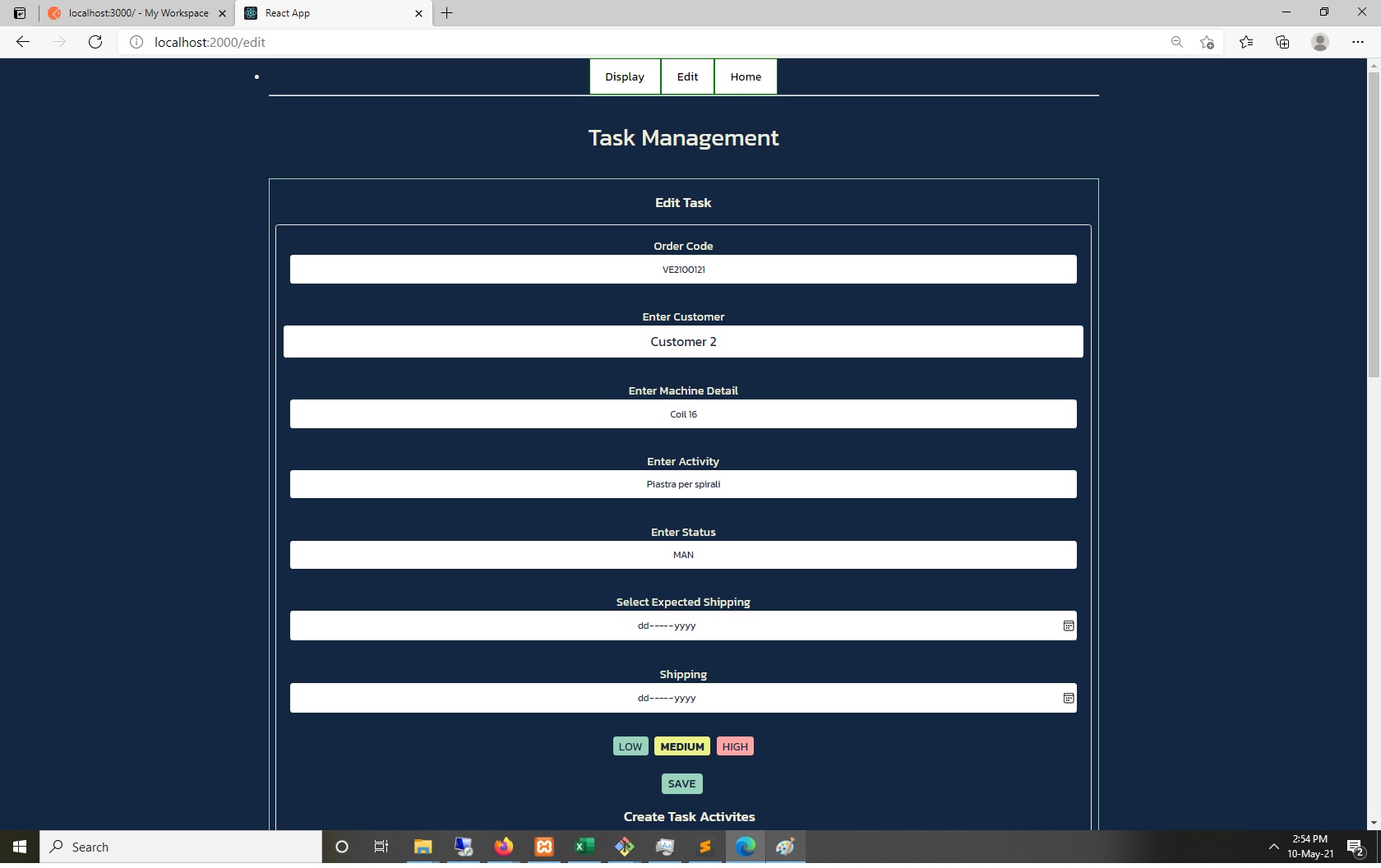
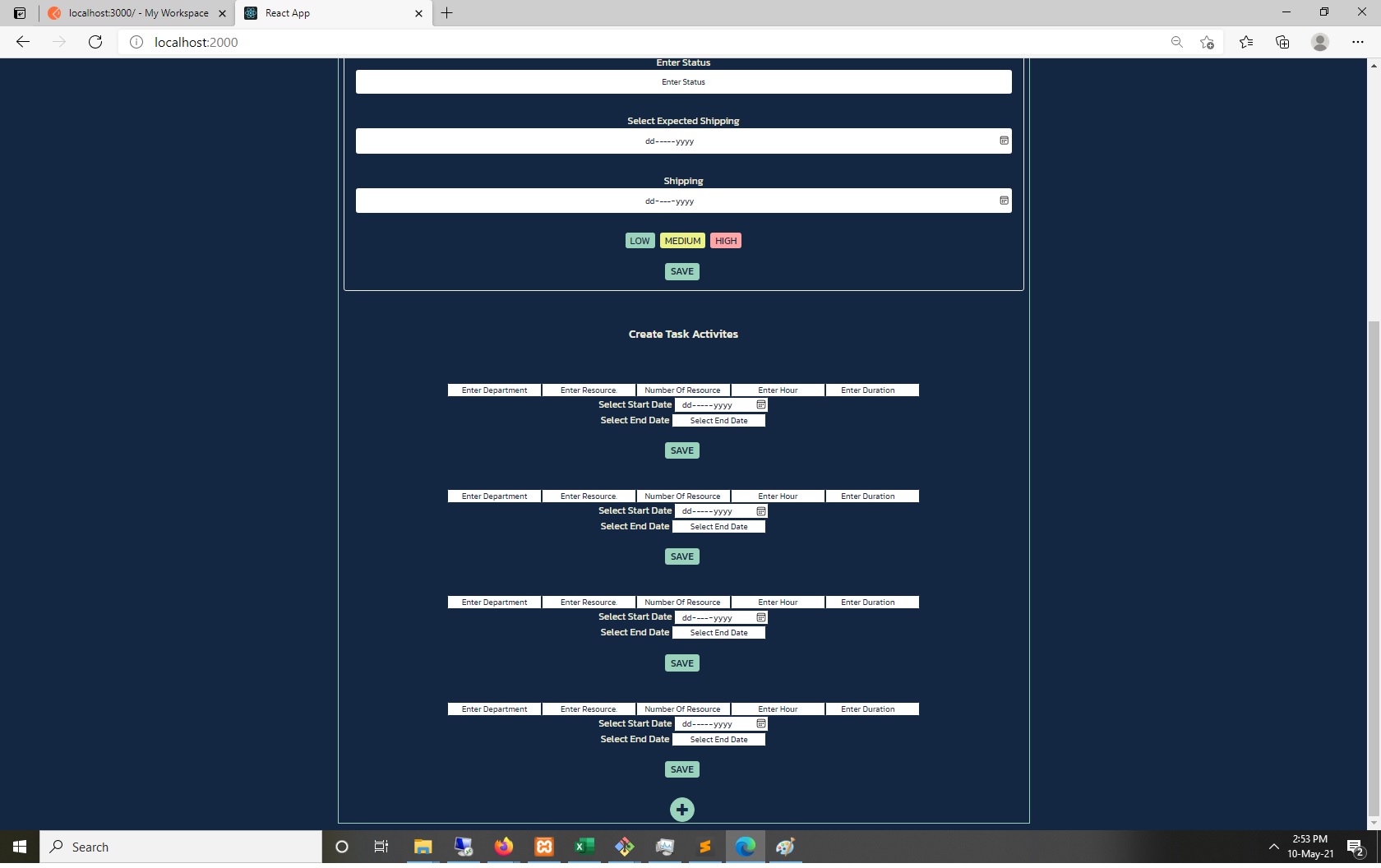
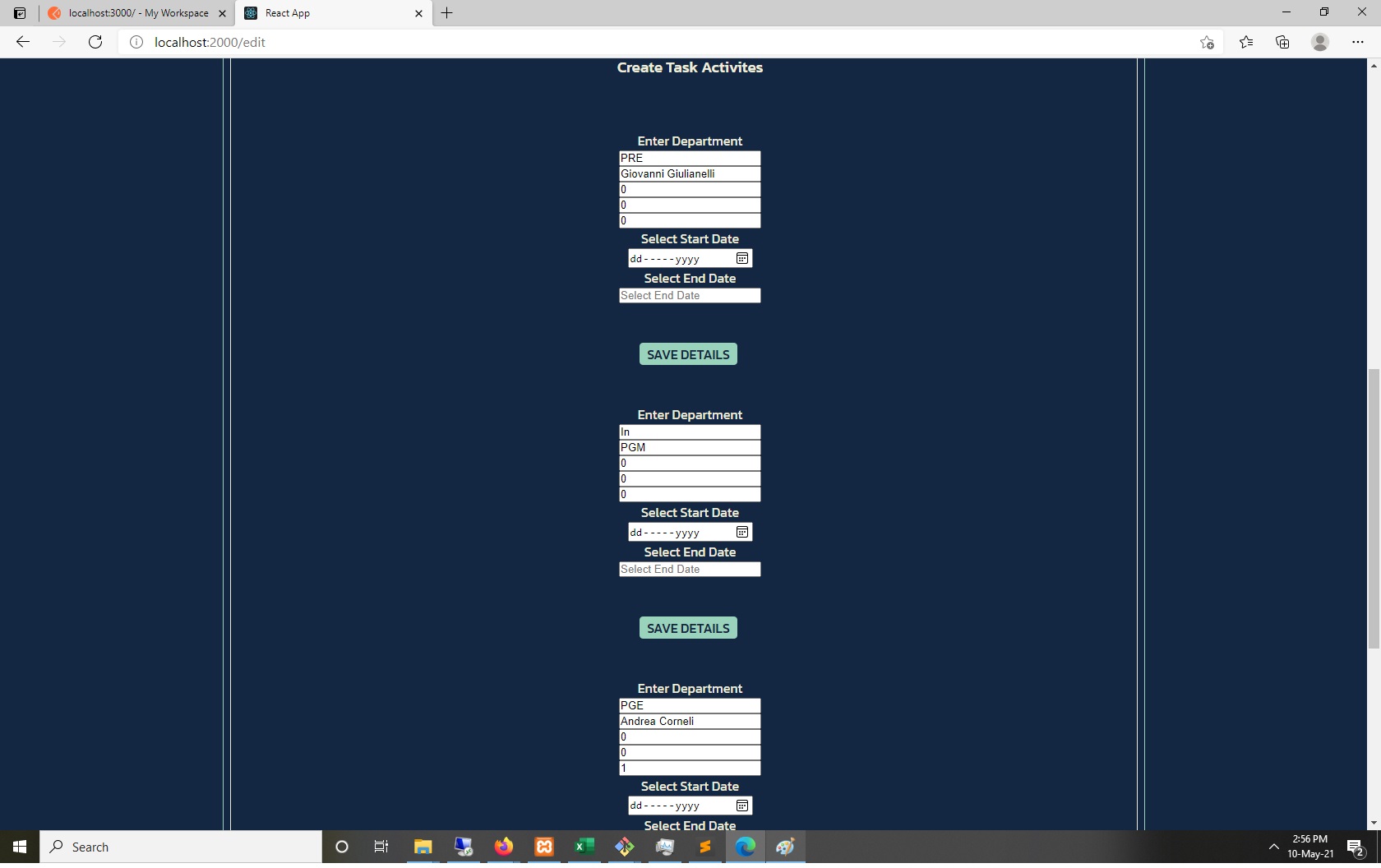
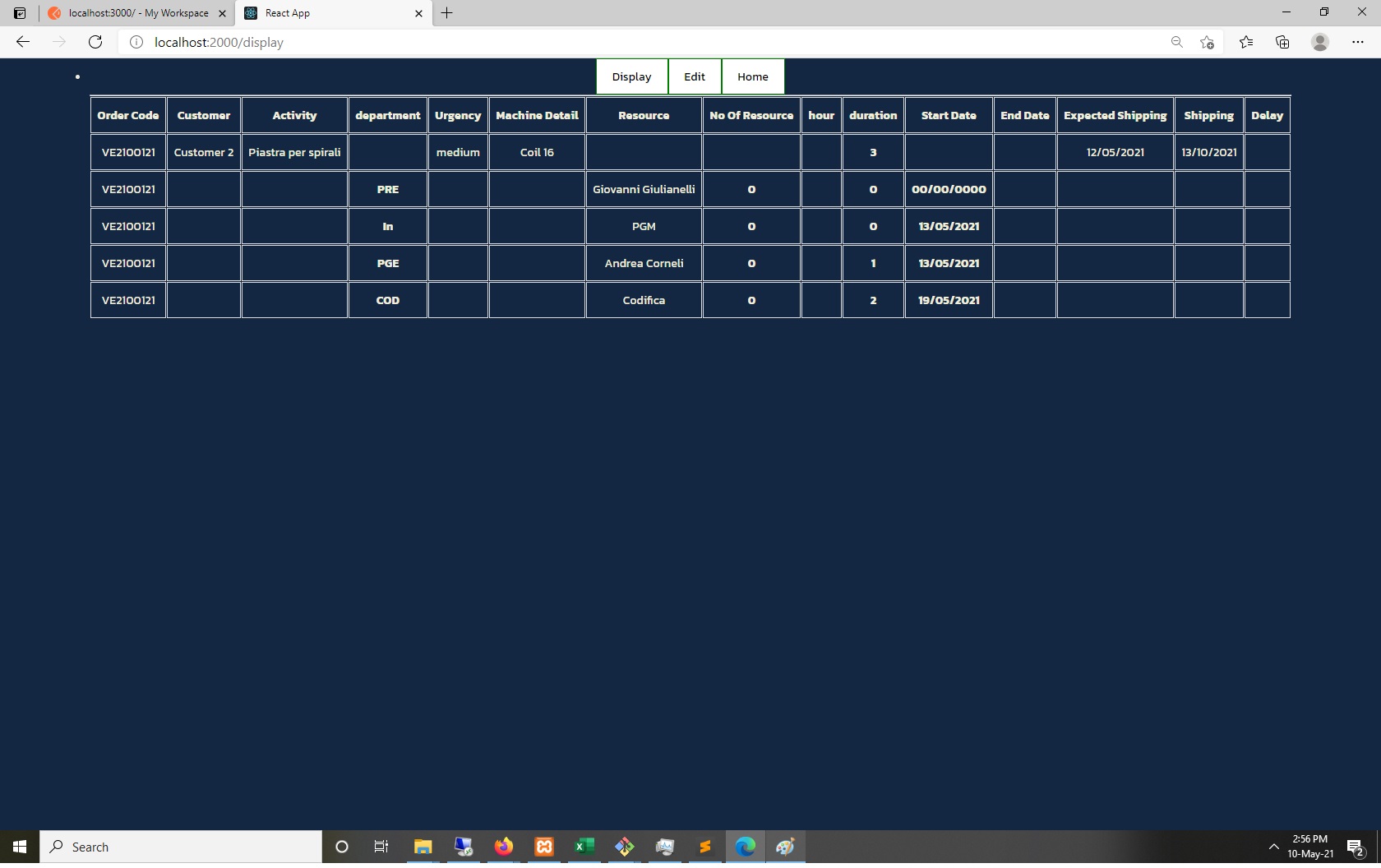
This is the package.json file for client side.



The main display function used to render the data from the backend api.



**Screenshots of User Interface are attached Below:**



In the Index JS file we just call the app component file from the component folder.

The App.js file is one of the most important files as it consists of all the function components which are used all over the projects and used by other files in the project. The Task.JS file is used for rendering the table 1 entry mode. Here we added id, orderCode, machineDet, activity, customer, urgency, status,

isCollapsed, expectedShipping, shipping and status. Here we define the functionalities in the file and give conditions, and in what condition which functions will be executed, the props are also included which has all the functions from app.js file.

We have another file called TaskDetail, here we have all the task details fields and the functions, the columns which we have are Enter Department, Enter Resource, No of Resource, Enter Hour, Enter Duration, Enter Title Input.

After saving data in Task Master the function component task detail is rendered and we can do the saving. For further we can go on Edit look up our saved data, modify or remove the record saved.

Saved Data can be viewed on Display on the button, populating from app.JS. Fetch API gets all the data and then renders it using the functional component Display. These are async functions using wait.

.ENV this is an environment variable folder which we use to save PORT on which client side server is started.

**Gestion de projet : comment vous êtes-vous organisé(e) ? Avez-vous fait un planning prévisionnel ? Si oui, quel était-il ?**

**Project management: how did you organize yourself? Did you make a schedule forecast? If so, what was it? ...**

We organize this project categorically in three sections, Entry/Home, Edit, Display. In home we have the entry for all the required data which will be master and transaction of task detail and respective activities. In edit we can edit all the task master and activities and also can remove it if we like or add more activities. In the third option which is display, here we display all the data.

We broke down our projects into many smaller components so we easily build our projects easily and simply and then slowly combine them all to make it a seamless process.

We kept a small notebook on which noted down the ideas for UI and functionality. Our forecast was to complete the project within three weeks. We jotted down the rough timeline for that.

**Conception de la solution adoptée :**

**◼ Réalisation : modélisation, implémentation, choix techniques, algorithmes et fragments de code intéressants,**

**Design of the solution adopted:**

**◼ Realization: modeling, implementation, technical choices, algorithms and interesting code fragments.**

In the current software building we used the Waterfall model for developing the software as we knew all the details from the start and there was no expectation of changes and manipulations, so we knew we could use that. We started using the small bits of components we were using and then built our application on top of that. We didn't need to use any algorithm yet.

We implemented everything in bits and components. Deciding what technology stack benefits. Then working with how to define the definition of each functions and naming conventions

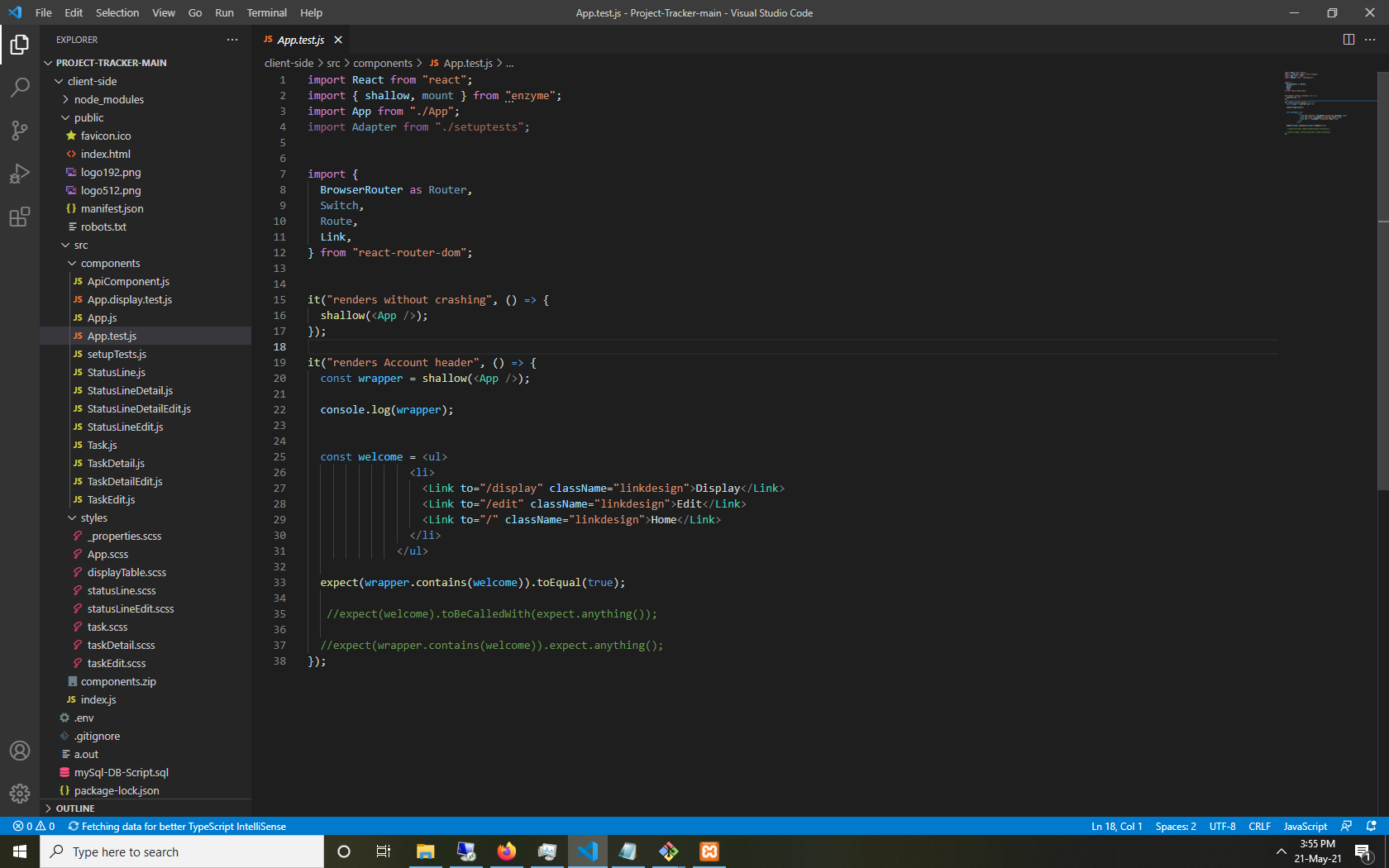
**Tests : planification, résultats, actions.**

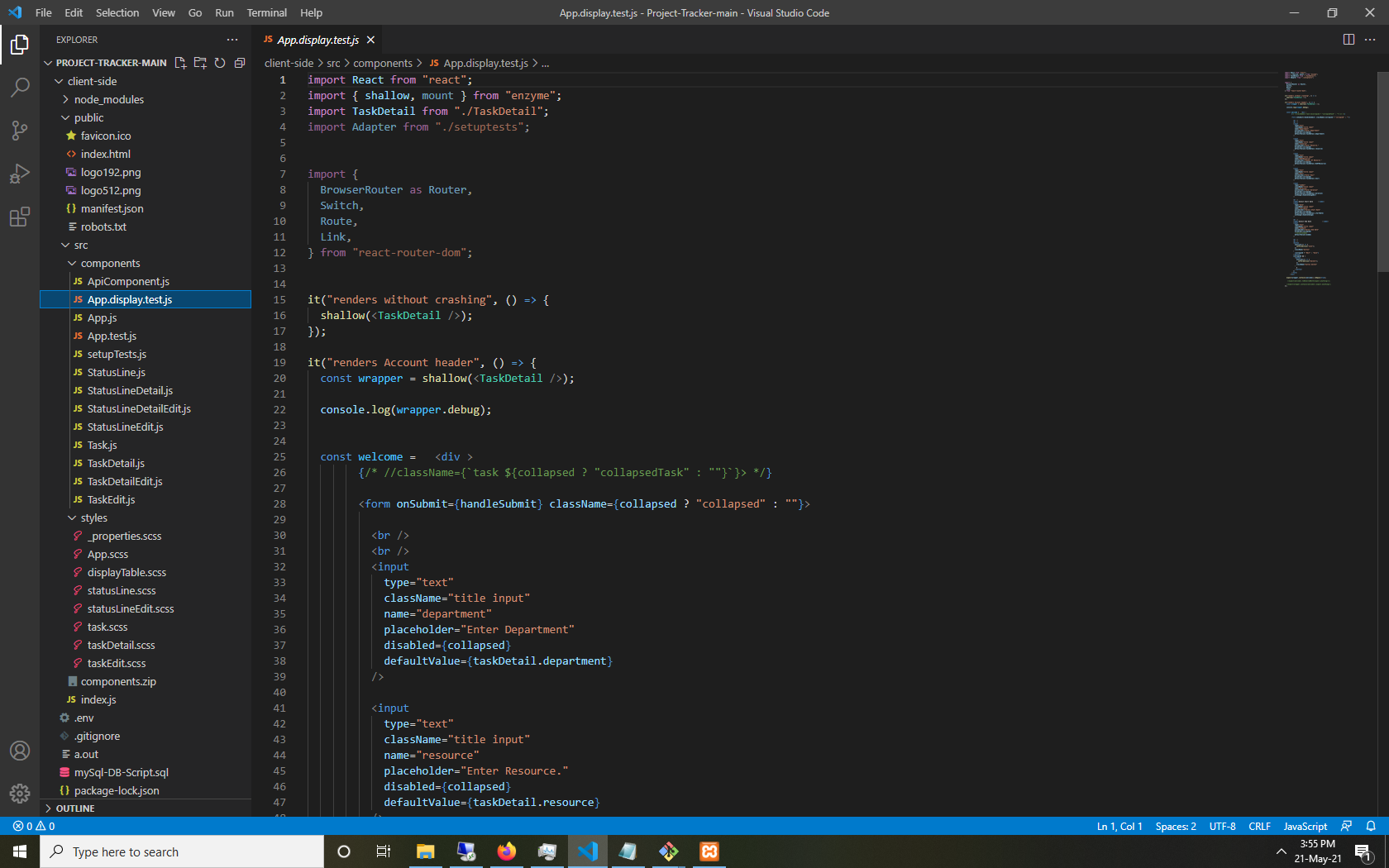
**Tests: planning, results, actions.**

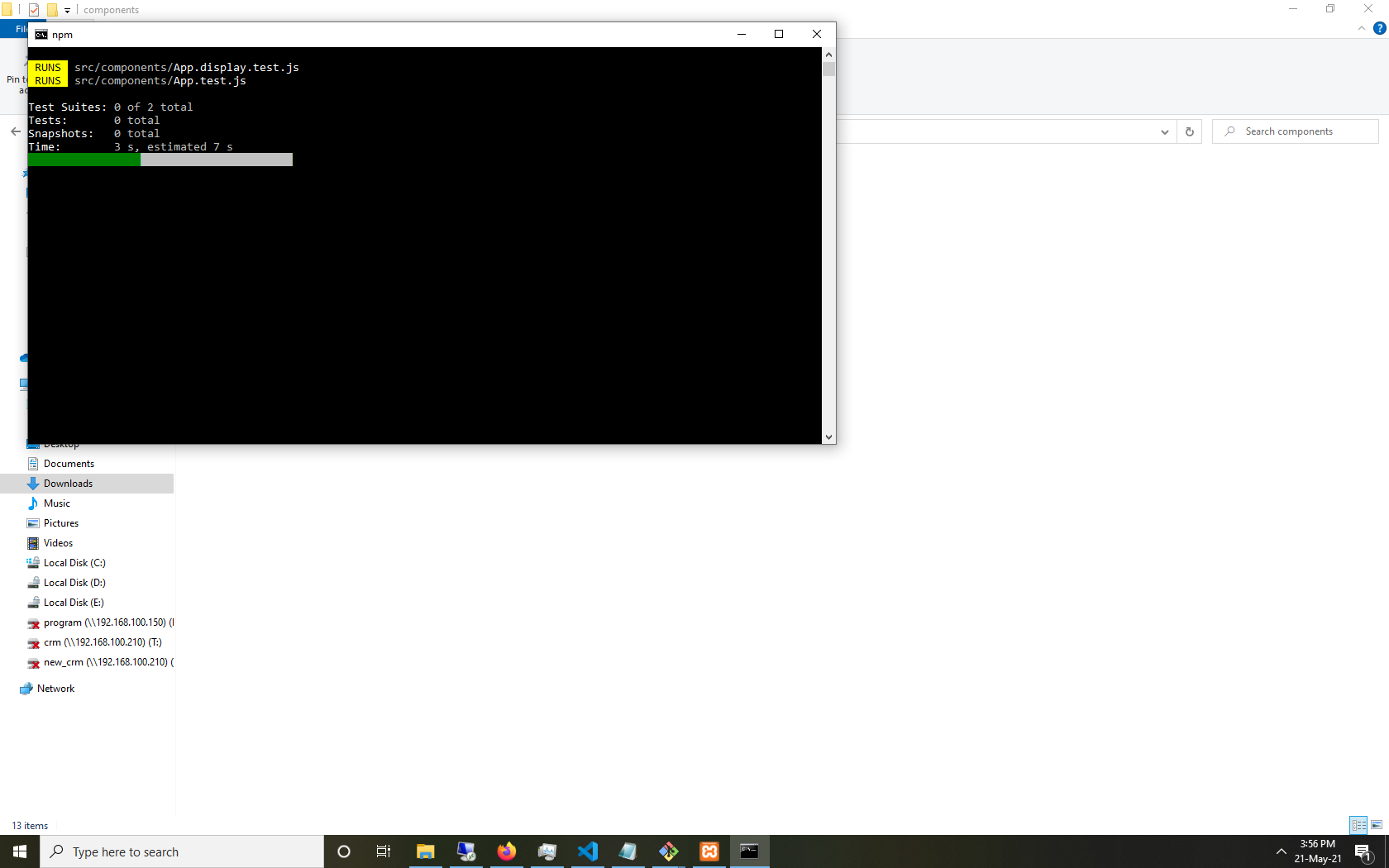
The first purpose of testing is to prevent regression. Regression is the reappearance of a bug that had previously been fixed. It makes a feature stop functioning as intended after a certain event occurs.

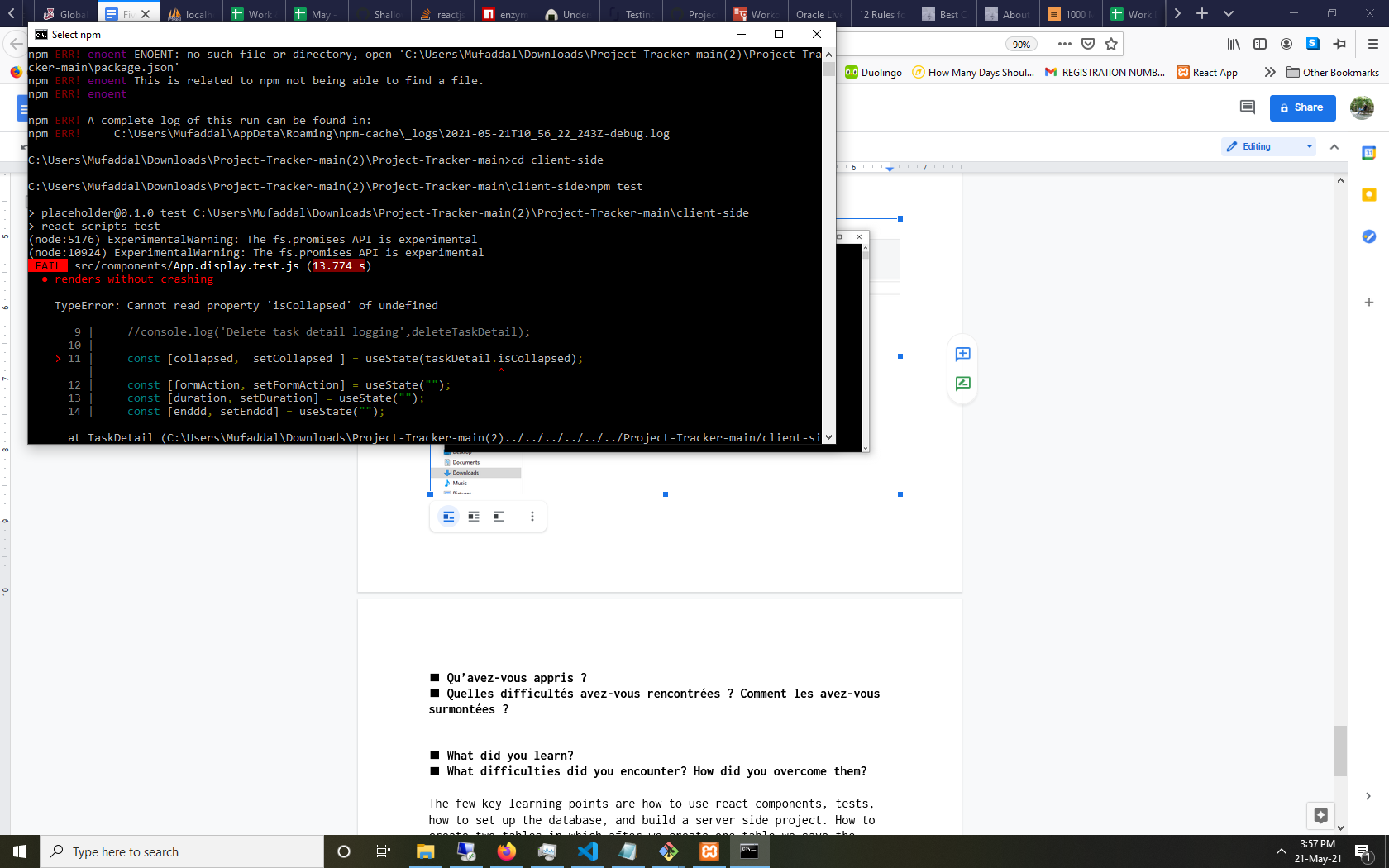
We use Jest with Enzyme for test scripts in react, following are the types we will use, Unit Testing, Integration Testing & End to End Testing. A little introduction is as follows:

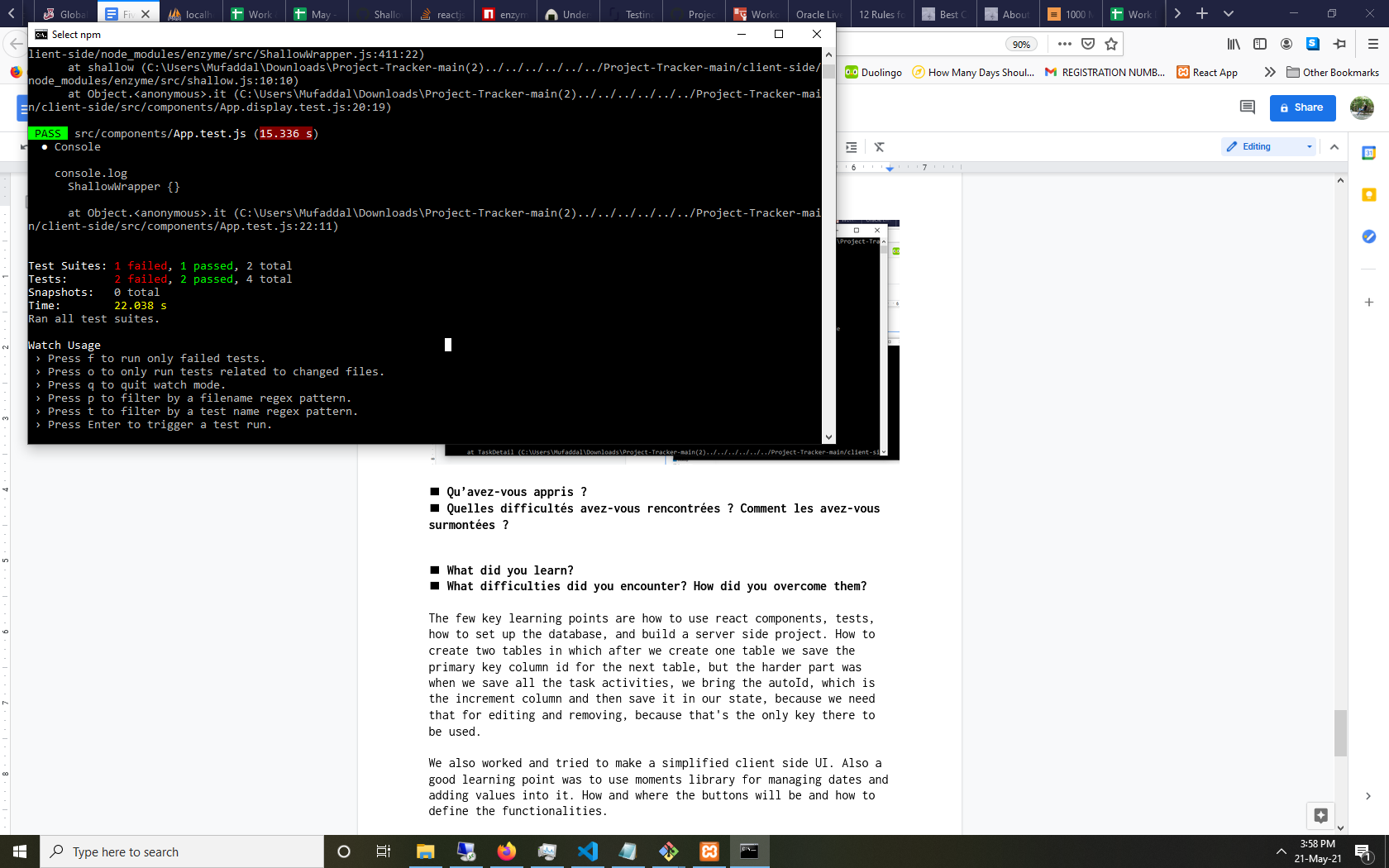
1. **unit testing**: testing an isolated part of your app, usually done in combination with shallow rendering. example: a component renders with the default props.
2. **integration testing:** testing if different parts work or integrate with each other. Usually done with mounting or rendering a component. example: test if a child component can update context state in a parent.
3. **e to e testing**: Stands for end to end. Usually a multi step test combining multiple unit and integration tests into one big test. Tests are done in a simulated browser, there may or may not be a UI while the test is running.

****

****







**◼ Qu’avez-vous appris ?**

**◼ Quelles difficultés avez-vous rencontrées ? Comment les avez-vous surmontées ?**

**◼ What did you learn?**

**◼ What difficulties did you encounter? How did you overcome them?**

The few key learning points are how to use react components, tests, how to set up the database, and build a server side project. How to create two tables in which after we create one table we save the primary key column id for the next table, but the harder part was when we save all the task activities, we bring the autoId, which is the increment column and then save it in our state, because we need that for editing and removing, because that's the only key there to be used.

We also worked and tried to make a simplified client side UI. Also a good learning point was to use moments library for managing dates and adding values into it. How and where the buttons will be and how to define the functionalities.

Few of the difficult side is to manage the multiple datatable and work them in a one good flow. How to seamless enter data in master activities and then save it into the task activities with all the references. Then the difficulty is to design and perform the summary columns setting in the display, how to populate and render the data in a good systematic way. We also used a moments library to add days in a single date and then how to format everything before storing it into the database and also fetching and calling back when rendering it in a screen of Task Edit.