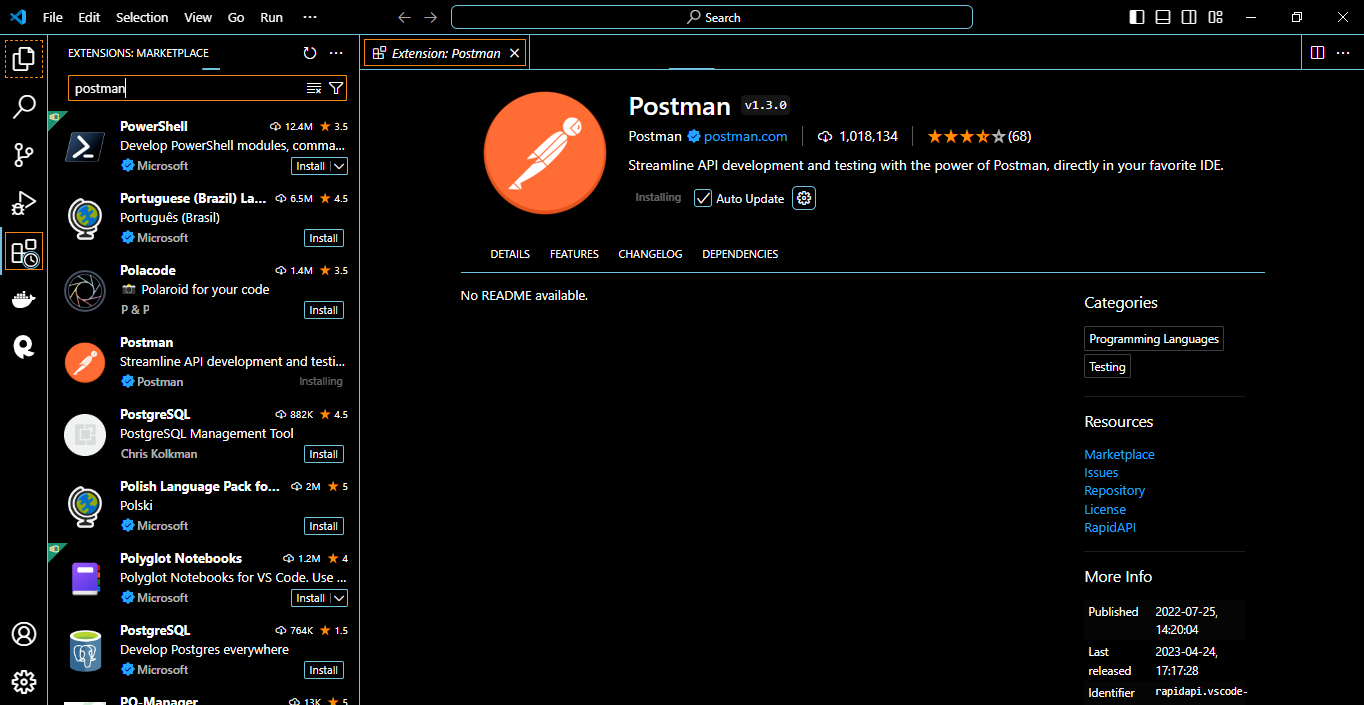
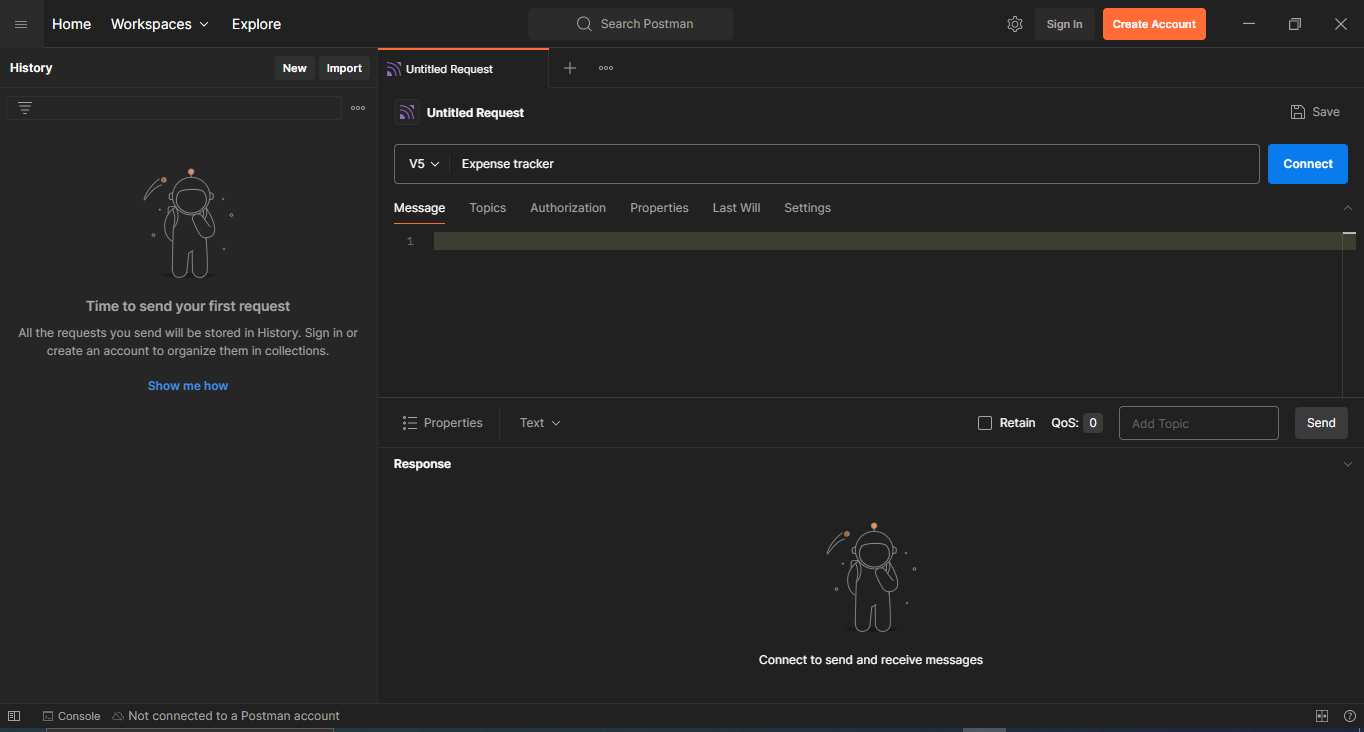
**Testing Backend API Endpoints**

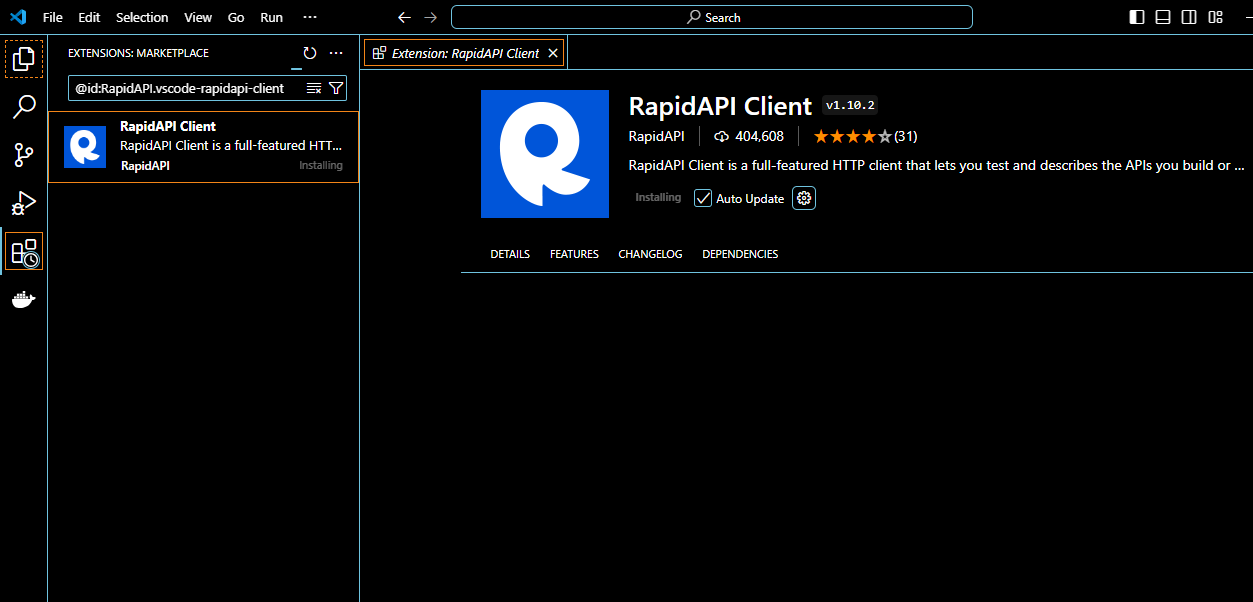
Install and set up Postman for testing backend API endpoints.

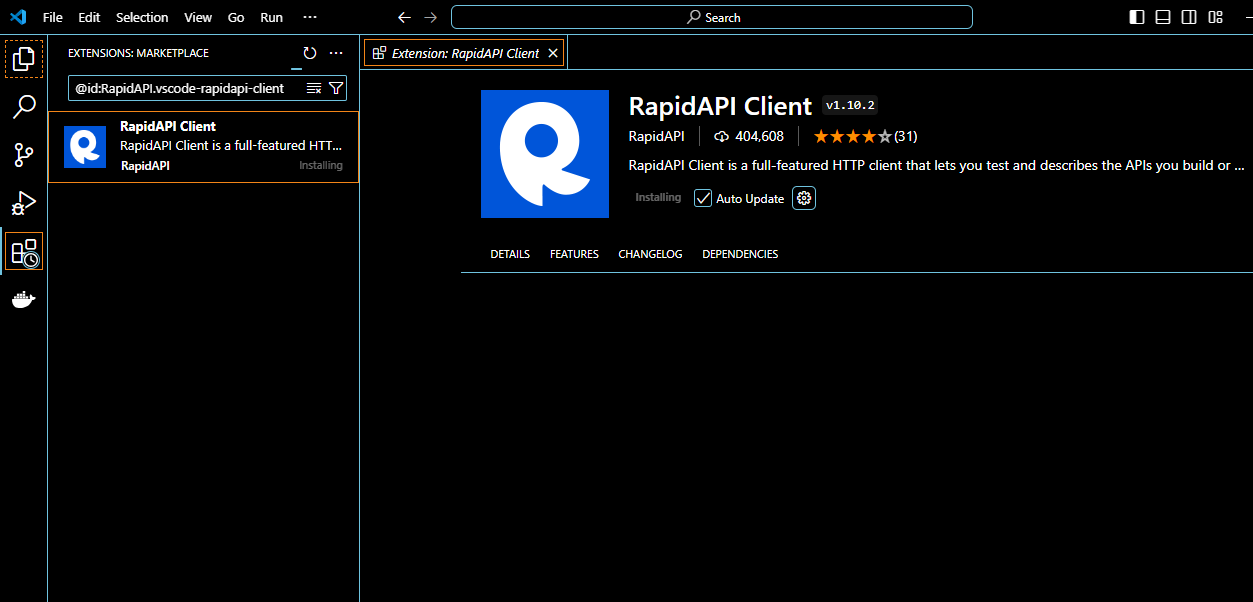
I installed both the postman and the postman extension for VsCode.



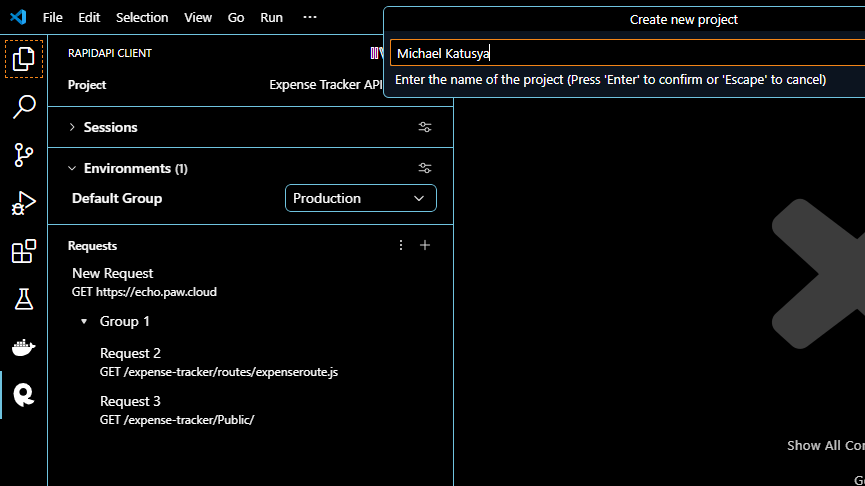


I also installed and mostly used the RapidAPI Client extension for VsCode.

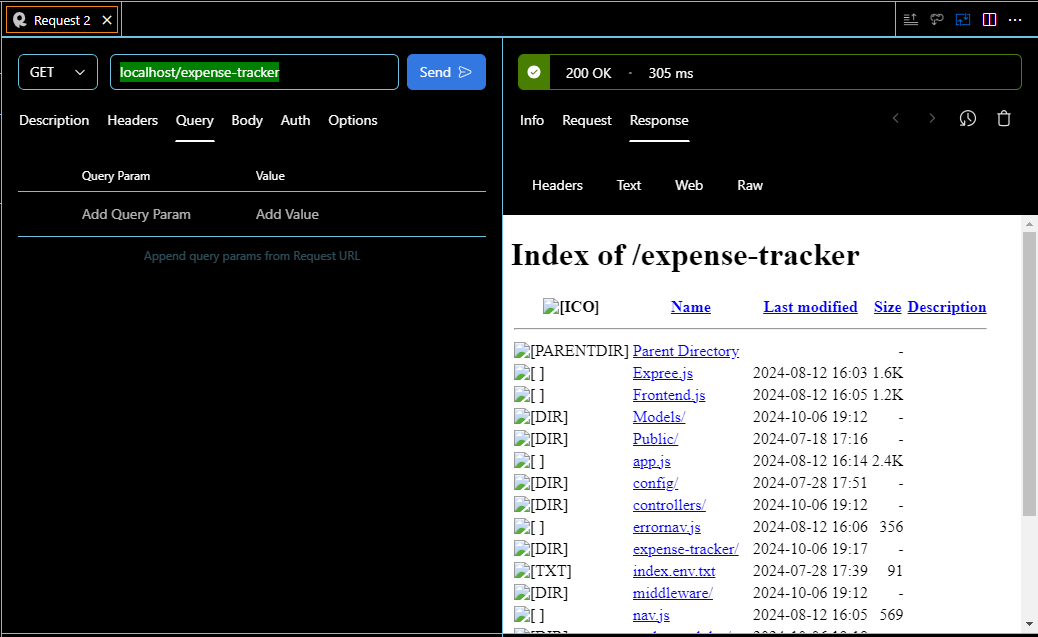


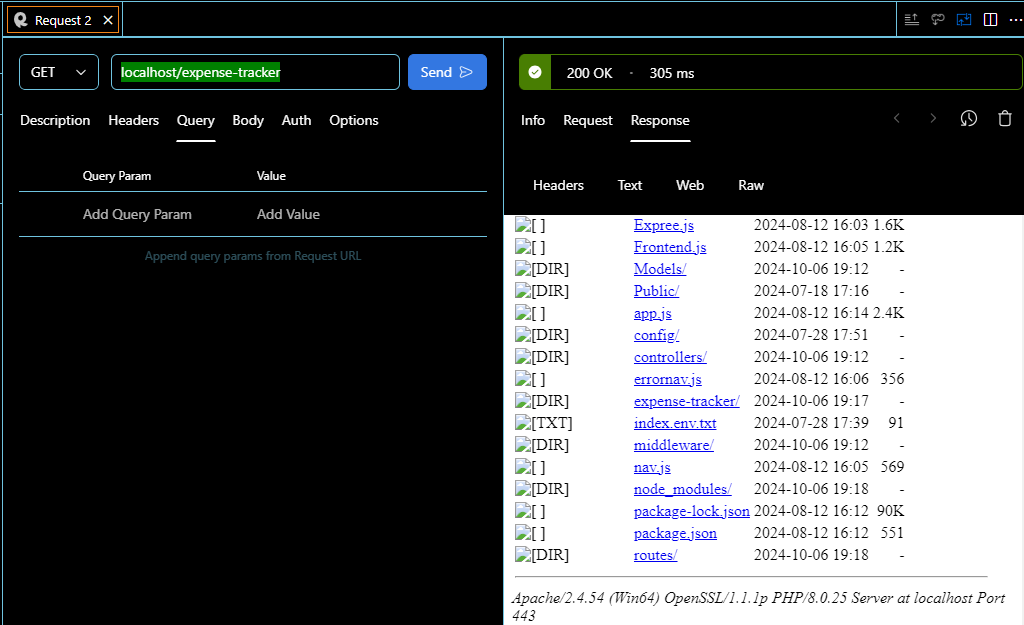


The start page of the rapidApi client, where I created a folder for the project I was supposed to work on.



The index page of the expense tracker, where all the assets were displayed





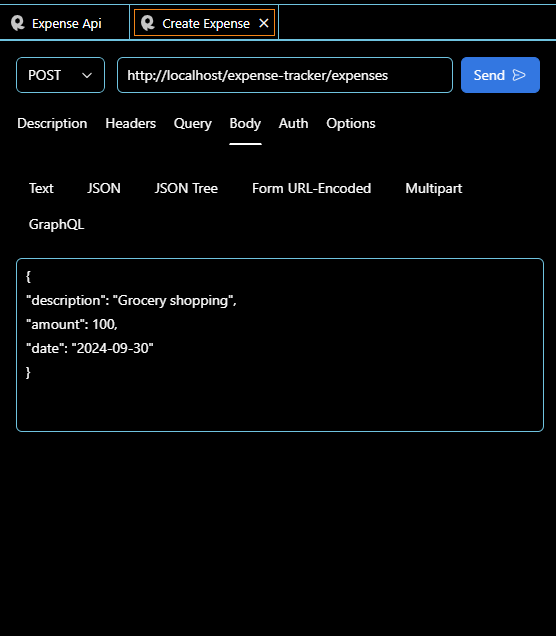
Creation of collections and requests in Postman to test various API endpoints in the expense tracking application

Test CRUD operations, authentication and error handling functionalities of the backend API.

CRUD stands for Create, Read, Update and Delete. These operations are fundamental to most APIs and testing them ensures that your API can handle the most common tasks.

**1.**

POST Request to Create an Expense

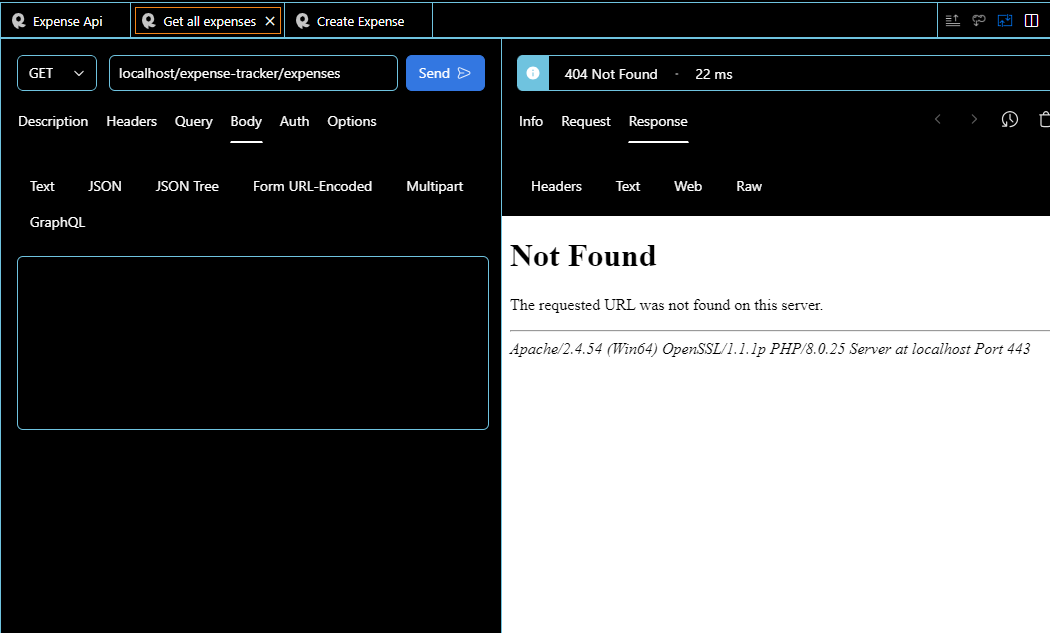


The above snippet shows how to post/create a new expense of **description** grocery shopping, **amount** 100 and **date** 2024-09-30. The expense is updated at where the last id is or you can specify the id on where to insert the expense.

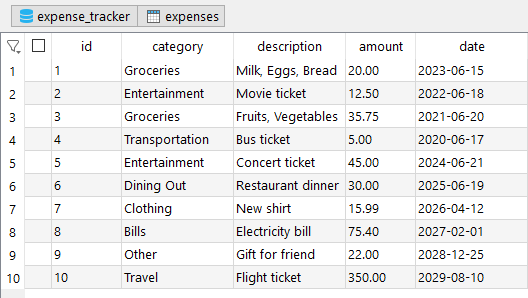
If successful, you should get a response with the message **new expense successfully added** to your database.

**2**

GET Request to Retrieve All Expenses



The GET request retrieves all expenses from your database and displays it. If it is not available the above code snippet is produced, but if successful then the data below should be retrieved from the database and be displayed.



**3.**

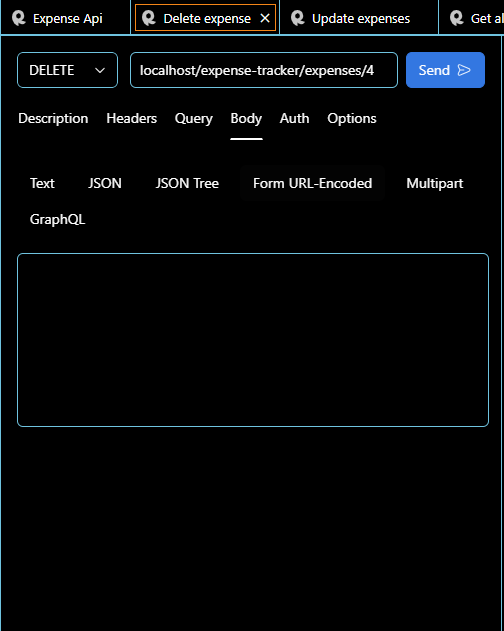
PUT Request to Update an Expense



This method will update the expense-tracker where the id is equal to 10 to the provided description and amount.

**4.**

DELETE Request to Remove an Expense



This will delete from the expense tracker where the id = 4

Debugging Frontend and Backend Code

**1.**

**Learn debugging techniques for frontend code using browser developer tools**

Browser developer tools e.g. Chrome DevTools offer powerful debugging features for frontend code, allowing you to inspect elements, view logs and trace through JavaScript code.

**Chrome DevTools**

Chrome DevTools provide several panels to help with debugging

* Elements - Inspect and modify HTML and CSS in real-time.
* Console - View errors, warnings and log output from your JavaScript code.
* Sources - Set breakpoints, step through code and inspect variables.
* Network - Monitor API calls, check status codes and view payloads.
* Performance - Analyze performance bottlenecks in your code.

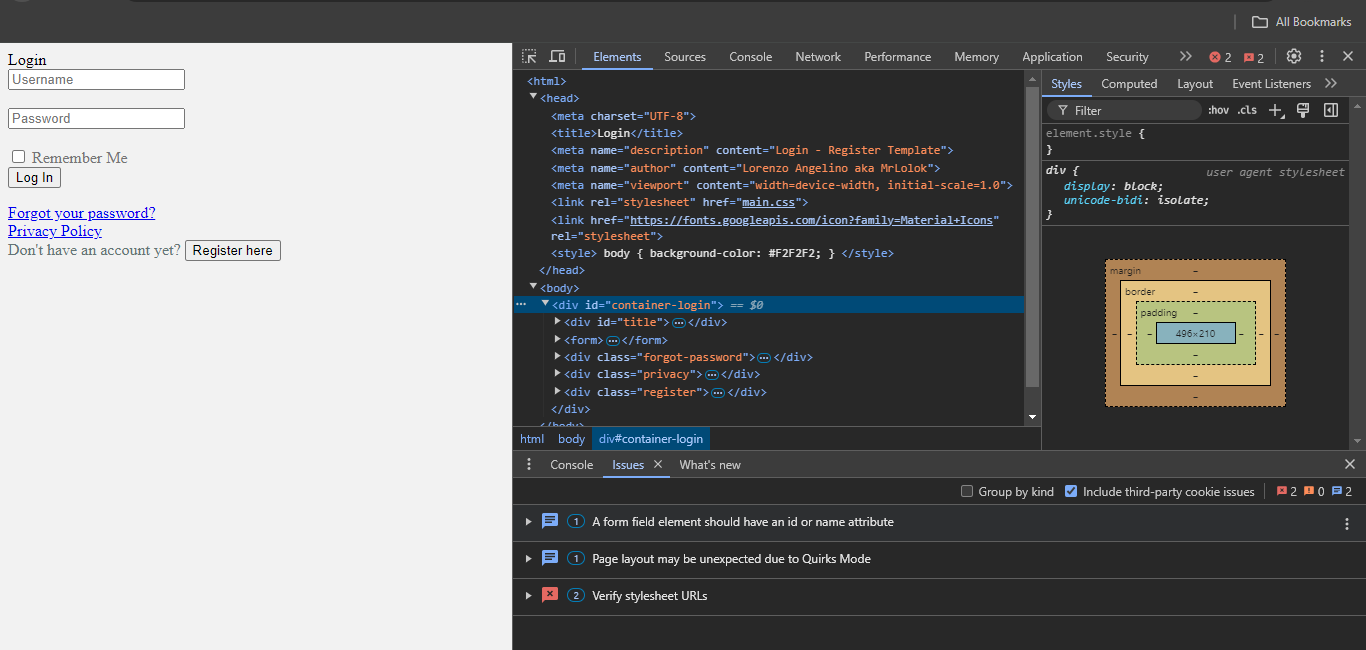
**Debugging Techniques**

Inspecting and Modifying HTML CSS in Real-Time

Open Chrome DevTools by clicking F12 or right-clicking on the web page then choosing Inspect.

In the elements panel, you can inspect and modify the structure of your HTML and the styles applied to each element.

Changes made here are temporary and useful for quickly testing CSS or HTML changes.



Sample inspect for a login page to the expense tracker

**Using Console for Logging**

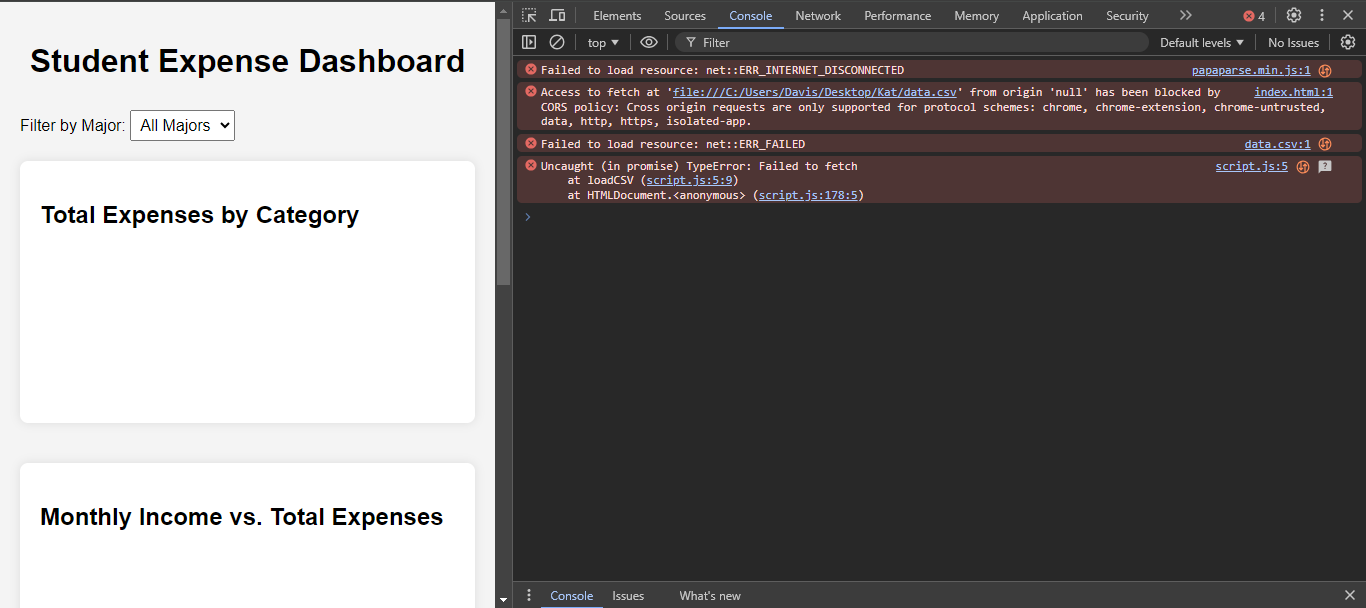
The console panel displays errors, warnings and logs. You can log information from your JavaScript code using console.log () to trace execution.

**Example**

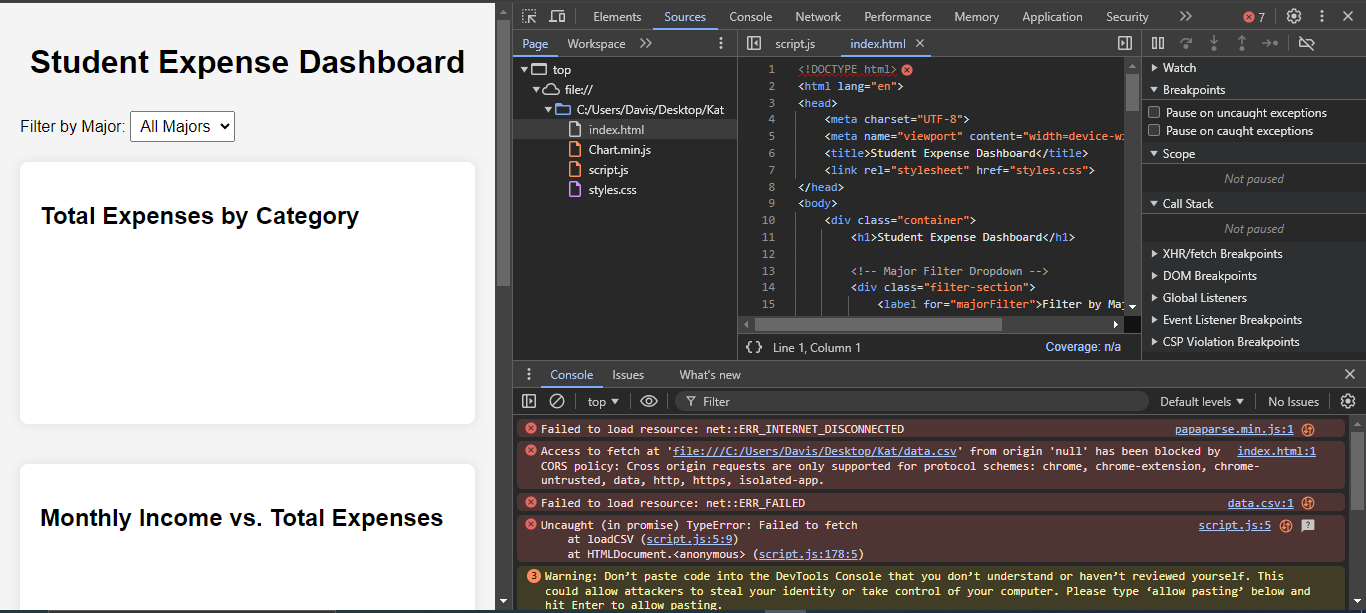
JavaScript

*console.log ('Form submitted:', formData);*

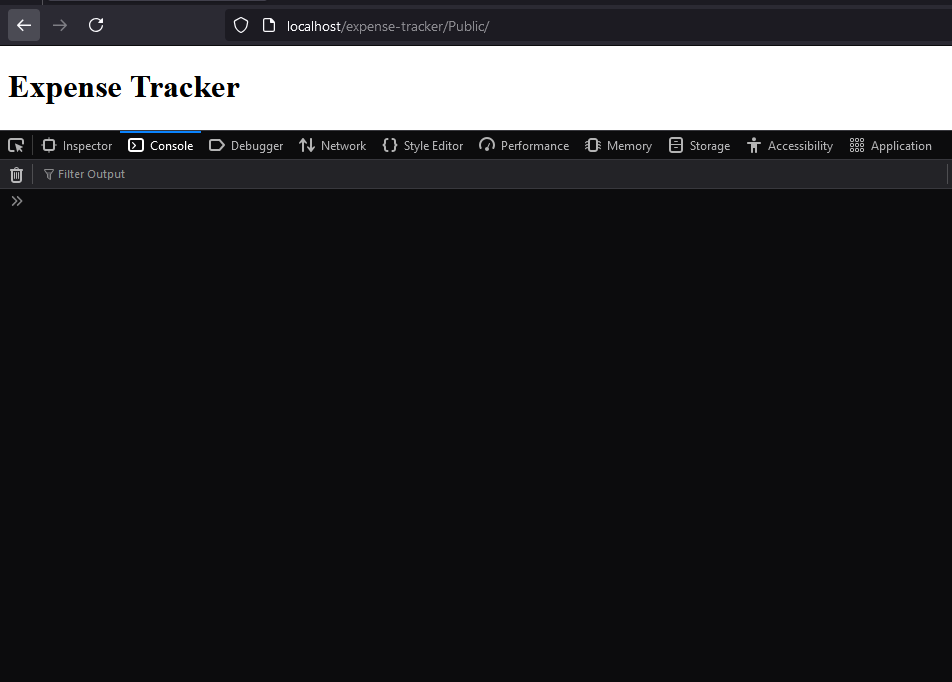
This allows you to inspect the values of variables and objects at different points in the code execution.



The snippet above shows issues in the code.



When clicking also on the console, you can also be redirected to where the specific issues is.

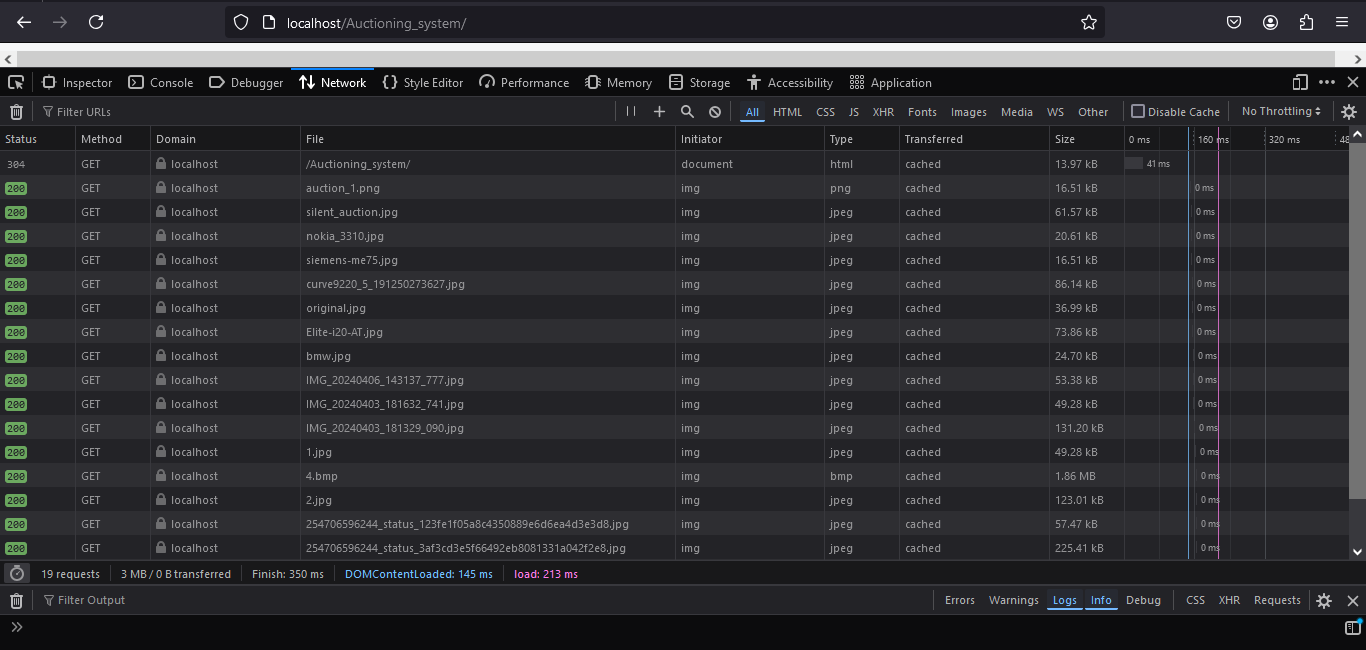


An example of a web page with no errors displayed on the console.

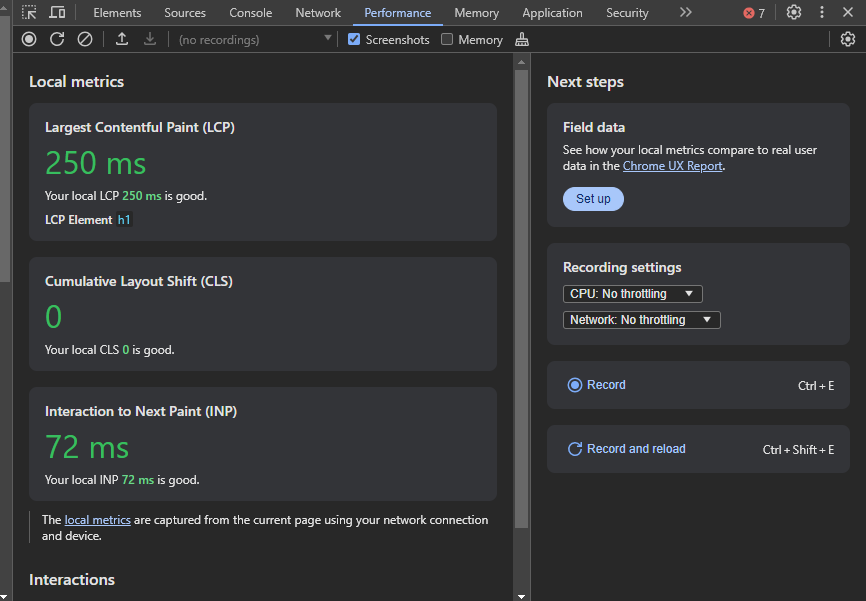
**Debugging Network Requests**

In the Network tab, you can view all HTTP requests made by your frontend. You can check the request headers, payload, response status and data.

This is useful for identifying issues with API calls, such as missing parameters, incorrect payloads, or server errors.



The figure above shows the network traffic passed on during the app usage.



The figure above shows performance.