**Practical assessment III – Front-end Scripting**

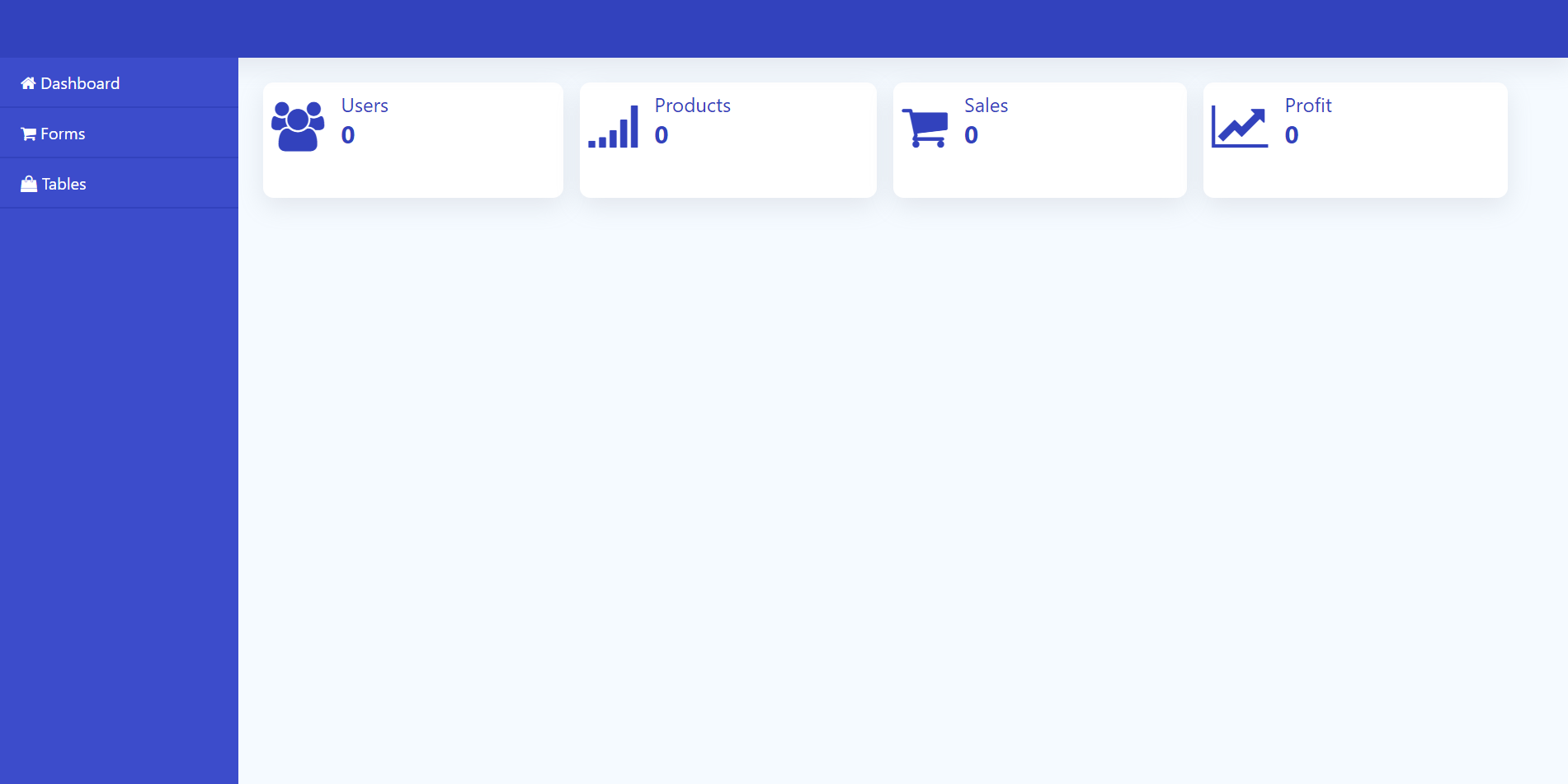
**Time: 11:30 – 13:30**

**Steps**

1. Download the base projects from GitHub. The project is the UI which you are going to build upon, here is the link <https://github.com/dmunthalli/practical-template.git>.
2. The second project is a dummy backend, it contains a server which we can interact with using a client like postman of any browser, but in this case its going to be the application you are going to develop. Here is the link <https://github.com/dmunthalli/AssessmentAPI.git> . Once downloaded, get into the root directory and run the command **npm run dev** in a terminal. You can interact with the API as follows.

|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Data |
| <http://localhost:3000/sold_products> | GET | Pie chart data |
| <http://localhost:3000/products/> | POST | name, price, description, quantity |
| <http://localhost:3000/products/> | PUT | name, price, description, quantity |
| <http://localhost:3000/products/id> | GET | Gets a product with the id |
| <http://localhost:3000/products> | GET | Table data |
| <http://localhost:3000/dash_stats> | GET | Dashboard cards data |
| <http://localhost:3000/top_products> | GET | Bar chart data |
| <http://localhost:3000/visitors> | GET | Line chart data |
| <http://localhost:3000/sales> | GET | Scatter chart data |

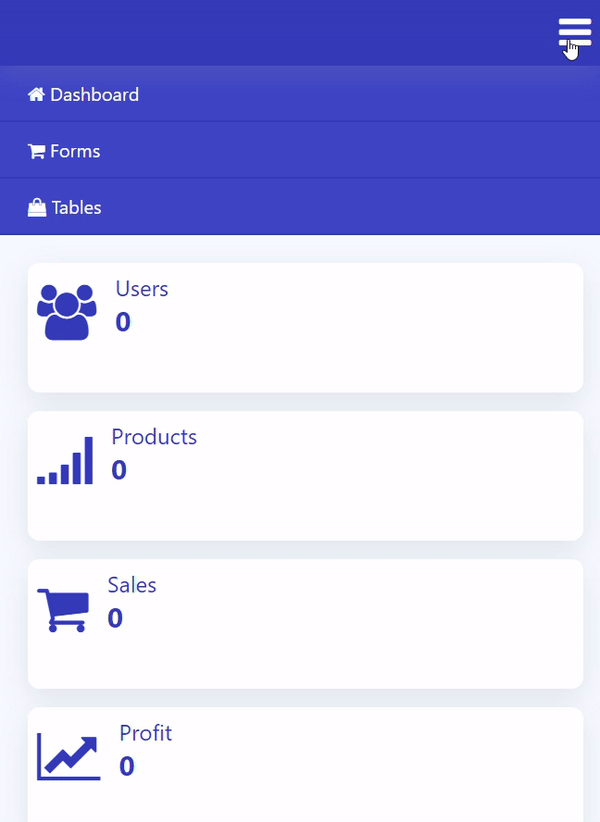
1. Initialise the project as a git repository (the base project).
2. Create a public repository on GitHub and link it with the local repository.
3. Secondly add each other as collaborators. If each member is able to make commits to the projects that will be a real reflection of collaboration, that means more points.
4. Explore the first project on your computer, you should have something similar to the following diagram.



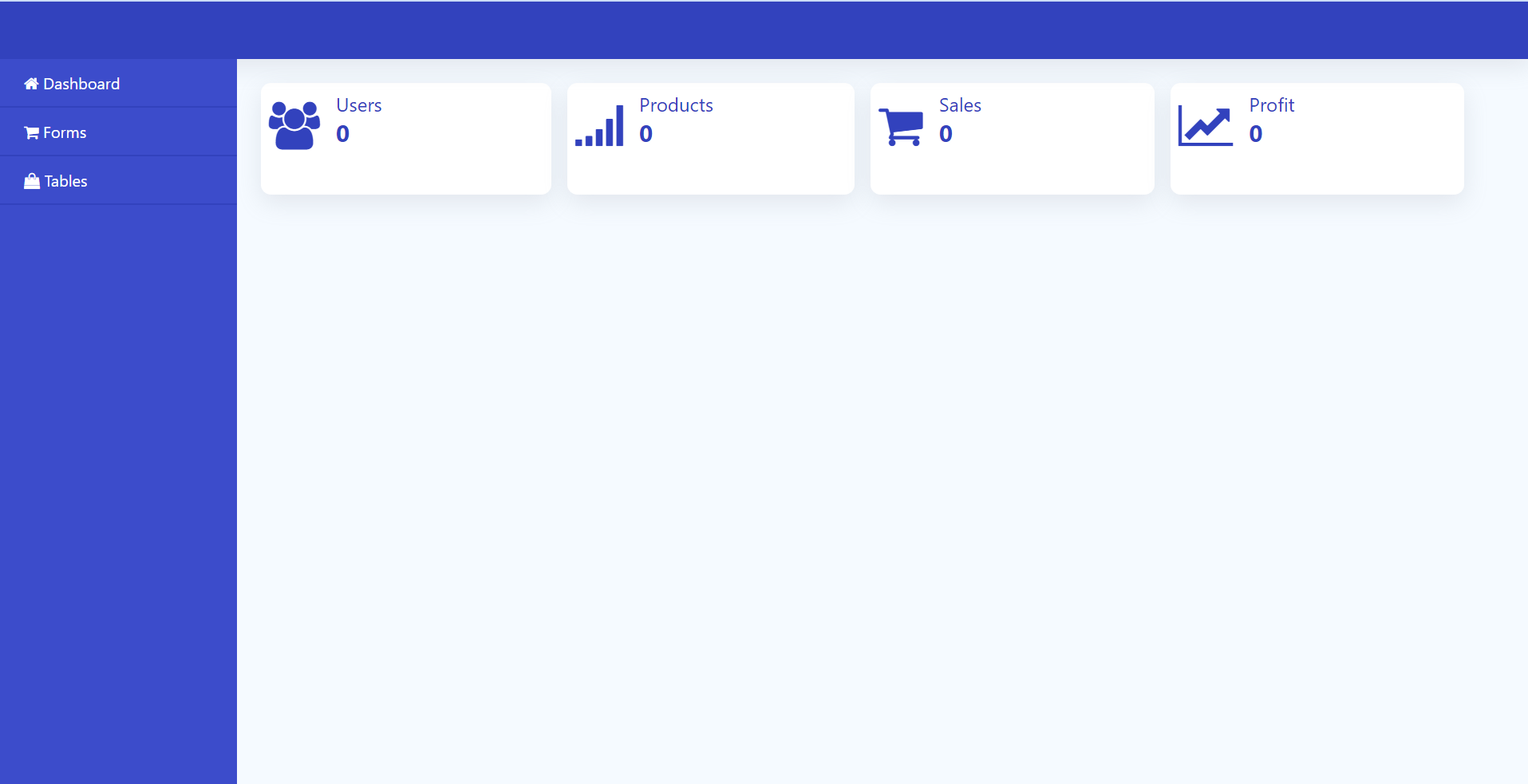
1. For each of the side navigation link there is just a header, each link has a separate file that you need to add content.

**Task 1- Showing and hiding the navigation menu**. **[5 marks]**

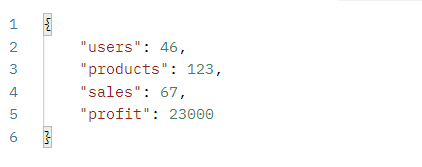
Provide an implementation that enables the user to show and hide the menu using the drawer button which gets shown on small screen devices as follows.



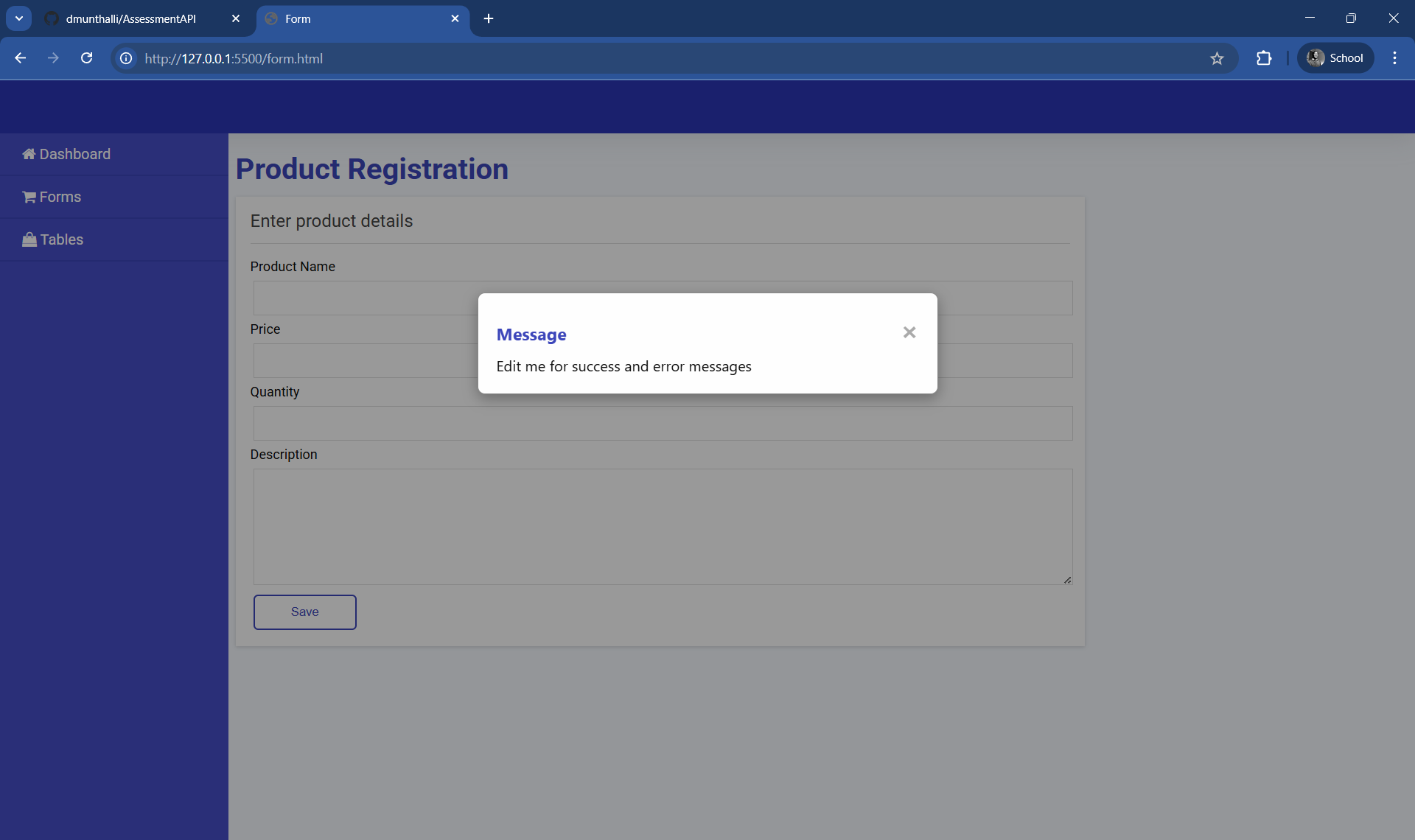
**Task 2 – Adding stats data to the cards [8 marks]**



In this task, load appropriate data from the API endpoint and update the values in the cards (initially they are set to 0). The end point is returning JSON data as follows.



**Task 3 – Submitting and validating the form**



In this task you are required to implement three things.

1. **Form validation**- implement validation on the data that can be entered by users in the various inputs here are some constraints.

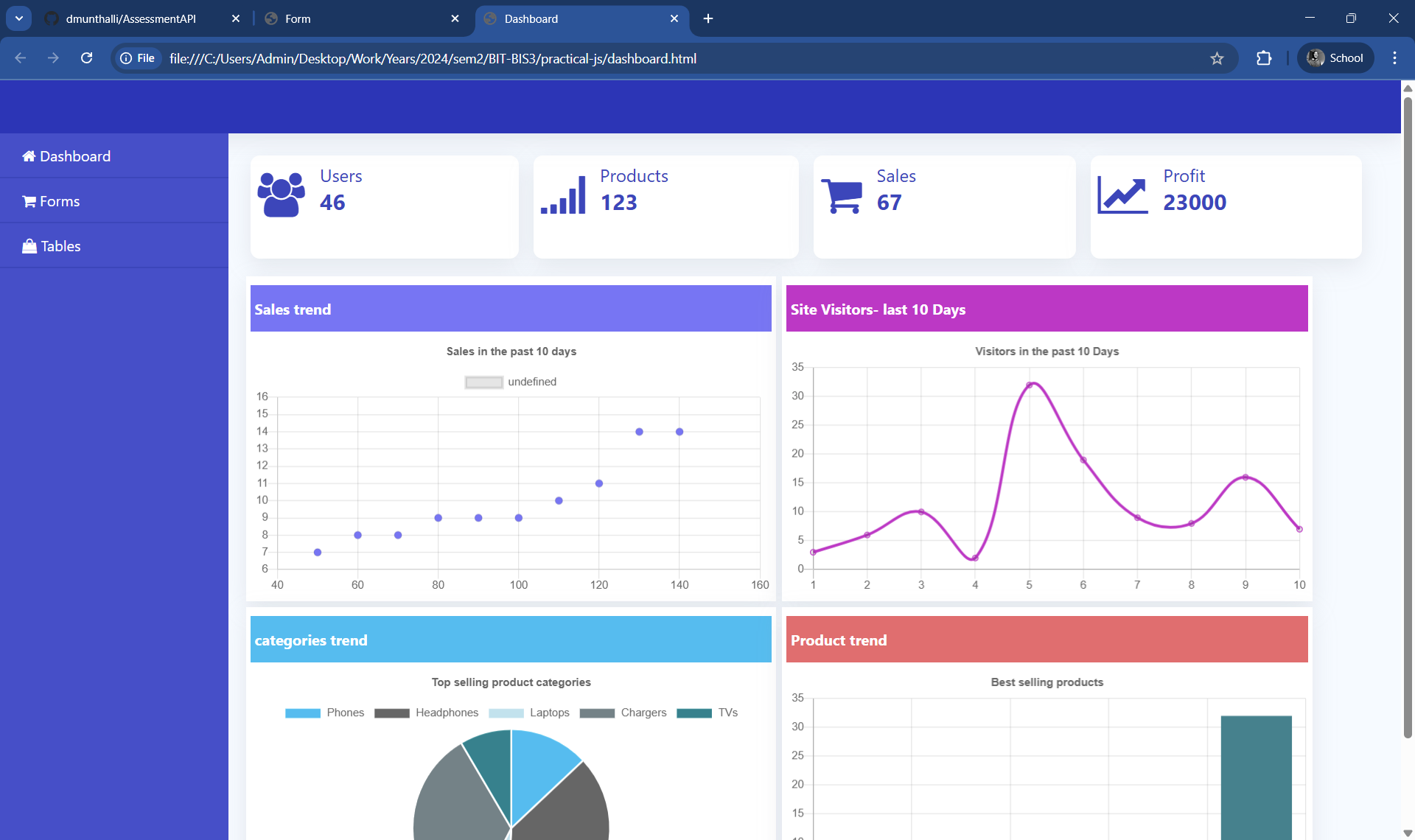
|  |  |
| --- | --- |
| **Input** | **Constraint** |
| Products | A string between 5 and 50 characters, not empty. |
| Price | An integer or float > 0, not empty |
| Quantity | An integer or float > 0, not empty |
| Description | A string between 5 and 200 characters, not empty. |

If the data entered by the user fails to meet the constraints, the users should not be allowed to submit the data. **[12 marks]**

1. **Submit the data**- implement code that when the user has entered the data and clicks the save button, the form sends the data to the given endpoint. When the data is sent successfully print the message from the server using a modal (more marks) or an alert. The entered data should reflect in the table.

**[5 marks]**

**Task 4 – Creating charts**

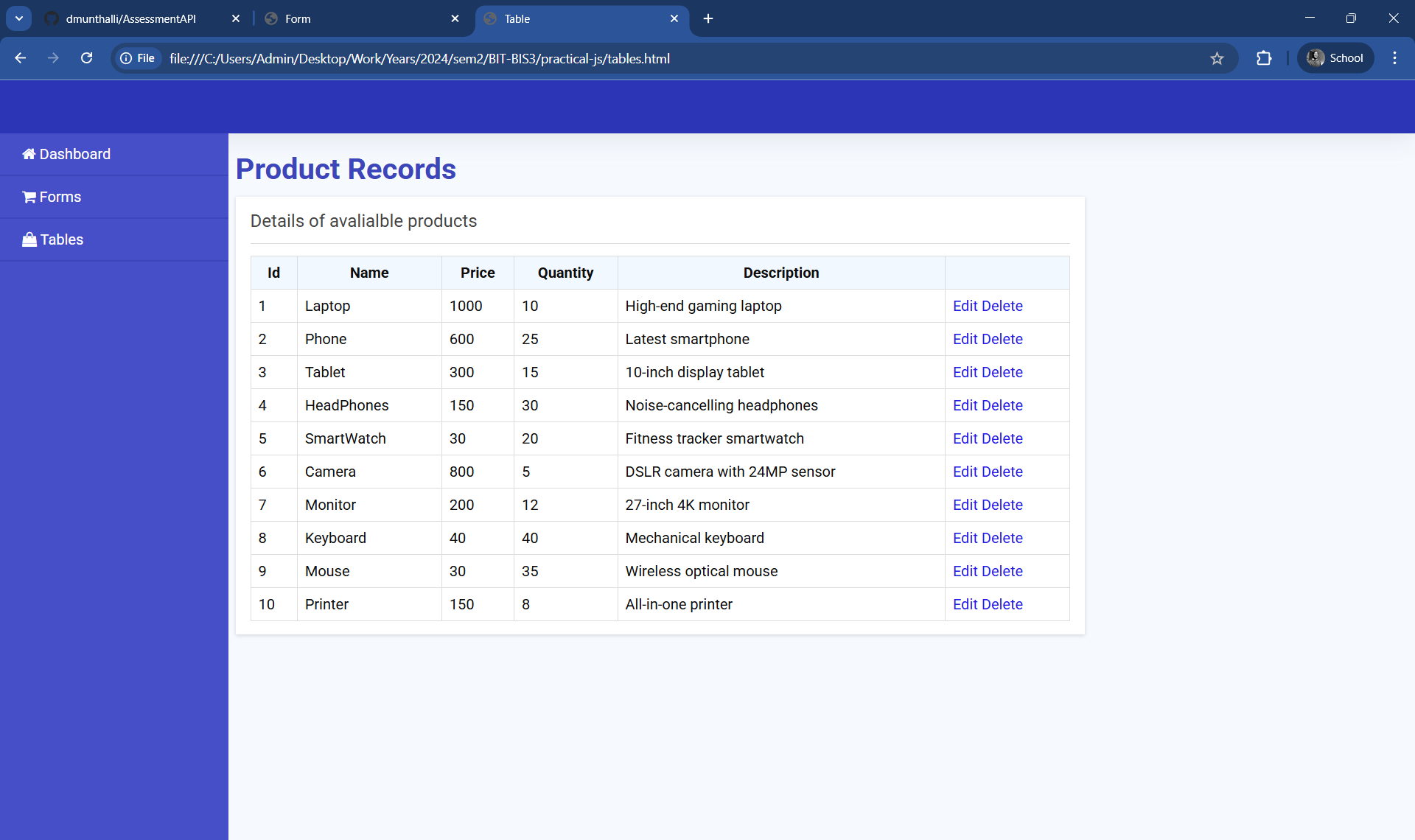


A screenshot of a graph

AI-generated content may be incorrect.

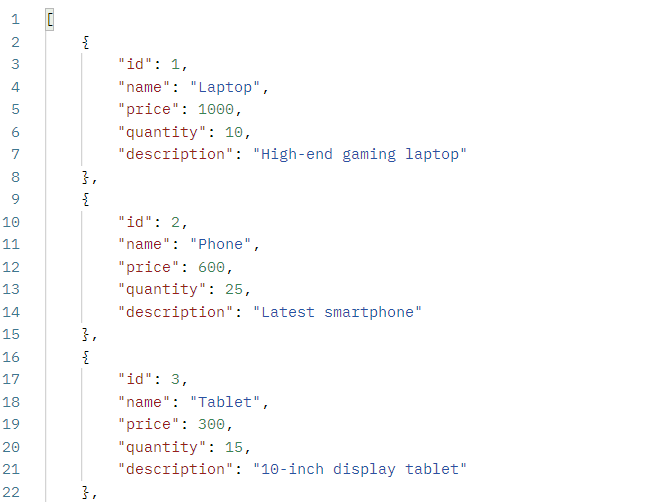
In this task you are required to implement the charts in the dashboard section. The are three different charts: scatter, line, pie and bar chart. Use the chart.js library and the data from the appropriate end points. **[20 marks]**

**Task 5 – Implement the table**



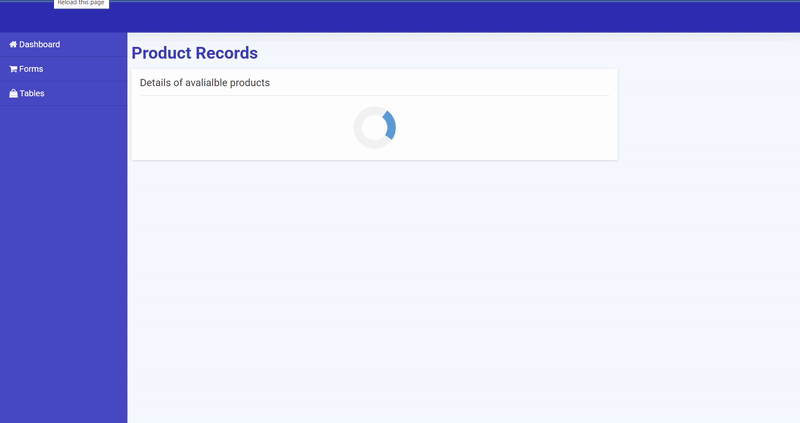
1. Pull data from the API endpoint and use it to create the table like in the figure. The end point is returning JSON data as follows. Note: the whole table structure has to be implemented using JavaScript (styles exist in the CSS file).

**[15 marks]**



1. Implement a loading animation which is displayed for 3 seconds before creating and loading the table to simulation a slow process. Consider the following demo.

**[10 marks]**



The loader CSS is already part of the CSS file you can see it by adding the following HTML.



1. Implement a delete operation. When a user clicks the delete button for a particular product, confirm the action with the user (using prompt() or a modal — modal gives more marks). If confirmed, send a request to the appropriate endpoint to delete the product. **[10 marks]**
2. Implement the edit operation. When a user clicks the edit button for a particular product, load the product’s data into a form for editing. After the user makes changes and clicks the update button, send the updated form data to the PUT endpoint to update the product.

**[15 marks]**