Pre-Task: Visual Analysis of Image Data with Generative Models

**1. Objective**

The goal of this project was to create a React application with two main features:

1. A **Home Page** to display a dynamic image grid.
2. A **Statistics Page** with interactive bar charts that allow users to filter data and view results.

**2. Tools and Technologies Used**

* **React.js**: Used to build the user interface of the application.
* **React Router**: Enabled navigation between the Home and Statistics pages.
* **D3.js**: Used to create interactive and visually appealing bar charts.
* **CSS**: Styled the application for responsiveness and interactivity.
* **JSON**: Simulated backend data for statistics and filtering.
* **Node.js & npm**: Set up the environment for React development.
* **Visual Studio Code**: Used as the code editor for writing and managing the project files.

**3. Installation Steps**

#### **A. Installing Node.js**

1. **Download Node.js**
   * Visit the official [Node.js website](https://nodejs.org/) and download the LTS version.
2. **Install Node.js**
   * Follow the installation wizard and complete the setup.
3. **Verify Installation**
   * Open the terminal and run the following commands:

node -v

npm -v

Ensure both Node.js and npm versions are displayed correctly.

#### **B. Setting Up Flask**

1. **Install Python**
   * Ensure Python is installed. If not, download and install Python from the [official Python website](https://www.python.org/).
   * Verify installation:

python --version

pip --version

1. **Install Flask**
   * Use pip to install Flask:

pip install flask

1. **Install Required Flask Modules**
   * Install any additional Flask modules as needed, e.g., Flask-CORS for handling cross-origin requests:

pip install flask-cors

#### **C. Starting Flask Server**

1. Create a app.py file with the Flask app logic.
2. Run the server using:

Python app.py

1. Access the server via http://127.0.0.1:<5000>.

#### **D. Installing Required Node.js Modules**

1. Navigate to the project directory in the terminal.
2. Install the necessary modules:

npm install react-router-dom d3

1. Start the React development server.

npm start

The application should open at http://localhost:3000.

**4. Approach**

#### **A. Navigation Using React Router**

* Implemented React Router to create two routes: one for the Home page and another for the Statistics page.
* Set up navigation links so users could switch between the two pages seamlessly.

#### **B. Home Page**

* Designed a grid layout to display images.
* Loaded image data from a JSON file.
* Ensured responsiveness so the image grid adapts to various screen sizes.

#### **C. Statistics Page**

1. **Interactive Bar Chart**
   * Used D3.js to create bar charts.
   * Allowed users to click on individual bars to filter data.
2. **Highlighting Selected Bar**
   * Highlighted the bar that was clicked to indicate it was selected.
   * Used dynamic styles to reset previously selected bars when a new one was selected.
3. **Displaying Data Counts**
   * Added labels on the bars to show the count of each category.
4. **Dynamic Filtering**
   * Clicking a bar dynamically filtered the data and updated the displayed content.

#### **D. Responsive Layout**

* Applied Flexbox for the layout of bar charts, ensuring they aligned horizontally and adapted well to different screen sizes.

**5.Challenges and Solutions**

#### **Challenge 1: Navigation Issues**

* Initial navigation setup didn’t work as expected.
* **Solution**: Used BrowserRouter and structured routes correctly to ensure proper navigation.

#### **Challenge 2: Highlighting Selected Bar**

* Clicking on a bar did not visually distinguish it from others.
* **Solution**: Used state management to track the selected bar and updated the styles dynamically.

#### **Challenge 3: Bar Chart Layout**

* The bar charts were stacked vertically instead of appearing in a horizontal layout.
* **Solution**: Utilized Flexbox in CSS to arrange the charts horizontally and ensured responsiveness.

#### **Challenge 4: Dynamic Data Updates**

* Filtering data based on user interaction was not updating the content dynamically.
* **Solution**: Implemented event handlers to capture user actions and update the state accordingly.

#### **Challenge 5**. **OpenSSL Issues in Node.js**

* **Challenge**: Encountered OpenSSL errors during Node.js installation and execution. Errors included:
  + Error: error:0308010C:digital envelope routines::unsupported
  + Missing or outdated OpenSSL libraries.
* **Solution**:
  + Updated Node.js to the latest LTS version.
  + Configured the OpenSSL legacy provider using:

In Power shell $env:NODE\_OPTIONS="--openssl-legacy-provider"

In command prompt set NODE\_OPTIONS=--openssl-legacy-provider

Rebuilt Node.js dependencies using npm rebuild to resolve potential module issues.

**6. User Interaction Features**

1. **Navigation**
   * Users can navigate between the Home and Statistics pages using the navigation bar.
2. **Bar Chart Interaction**
   * Clicking on a bar highlights it and filters the data dynamically.
   * The bar chart shows data counts on the bars for better clarity.
3. **Responsive Design**
   * The application is fully responsive, ensuring usability on different devices.

**7. Styling and Design**

* Used CSS for styling navigation, grid layouts, and bar charts.
* Focused on clean design with user-friendly colors and hover effects.
* Ensured selected bars had distinct colors to make interactions clear.