Frontend Developer Assignment

Objective

Develop a basic ad builder application using the Polotno library. The application should feature a "Feed" tab displaying static data (images and product headers) and a builder's side view that allows users to edit selected images or product headers.

Assignment Details

1. Project Setup

Initialize a New React Project
npx create-react-app ad-builder
cd ad-builder

Install Polotno Library
npm install polotno

- Obtain a free API key from the Polotno Cabinet for development purposes.

---

2. Implementing the "Feed" Tab

Purpose

- The "Feed" tab serves as a repository of static assets (images and product headers) that users can select for editing.

Steps

- **Create a Tab Interface:**
  - Implement a tabbed navigation using a library like Material-UI or build a custom tab component.
  - Refer to this tutorial for guidance on creating tabs.
- **Design the "Feed" Tab:**
  - Display a list or grid of static images and product headers.
  - Store these assets locally in your project (e.g., in a `public/assets` directory).
- **Enable Selection:**

    - Allow users to click on an asset, which will then load into the builder's side view for editing.

---

3. Developing the Builder's Side View

Purpose

- This area provides users with tools to edit the selected asset from the "Feed."

Steps

- **Integrate Polotno Components:**
    - Set up the Polotno workspace within the builder's side view.
    - Refer to the Polotno Documentation for integration details.
- **Load Selected Assets:**
    - When a user selects an asset from the "Feed," load it into the Polotno workspace for editing.
- **Implement Editing Features:**
    - Provide basic editing functionalities such as adding text overlays, resizing, and applying filters.
    - Utilize Polotno's built-in tools to facilitate these features.

---

4. User Interface and Experience

Design Considerations

- Ensure the application is responsive and intuitive.
- Maintain a clean layout with clear distinctions between the "Feed" and the builder's side view.

Navigation

- Implement smooth transitions between selecting an asset in the "Feed" and editing it in the builder.

---

5. Technical Specifications

Framework/Libraries

- React.js for building the user interface.
- Polotno for the canvas editor functionality.

Styling

- Use CSS frameworks like Bootstrap or Tailwind CSS for consistent styling.

State Management

- Utilize React's built-in state management (e.g., `useState`, `useContext`) or a state management library of your choice.

Version Control

- Use Git for version control.
- Regularly commit your code with clear messages to a public Git repository.

---

6. Deployment

Purpose

- Deploy the application to provide a live demo link for evaluation.

Steps

- **Deploying to Vercel:**

```
npm install -g vercel
vercel login
vercel
```

- Follow the prompts to link your project and deploy.
- Refer to Vercel's Deployment Documentation for detailed instructions.

---

7. Submission Guidelines

Repository

- Fork the provided GitHub repository [insert repository link].
- Push your code to your forked repository regularly.

README File

- Include setup instructions detailing how to run the project locally.
- Describe your approach to implementing the features.
- Note any assumptions or decisions made during development.
- Provide the live deployment link to the working application.

---

8. Evaluation Criteria

- **Functionality:**
    - The application meets all specified requirements and handles edge cases gracefully.
- **User Experience:**
    - The interface is intuitive, responsive, and visually appealing.
- **Code Quality:**
    - Code is clean, well-structured, and adheres to best practices.
- **Problem-Solving:**
    - Demonstrates effective problem-solving skills in implementing features.
