ReactJS Hackathon Machine Test

Problem Statement

You are required to build a **mini banking application** in **ReactJS**. This application will demonstrate a **multi-step registration wizard, KYC verification, OTP validation, and Google SSO login flow**. You will use a **JSON Server** to mimic backend APIs (no real backend or database connectivity is required).

The focus of this test is to evaluate your knowledge of:

- React fundamentals (hooks, state management, components, props, context, routing)
- Form handling & validations
- File upload & client-side validation
- Integration with JSON server (mimicking backend API calls)
- Google SSO login integration
- Code reusability, security, and clean folder structure

🔷 Requirements

1. Registration Wizard (3 Steps)

Build a **3-step wizard form** where users register for a new banking account.

Step 1: Basic Information

Fields:

- First Name (required, alphabets only, no special characters)
- Last Name (required, alphabets only, no special characters)
- Phone Number (required, 10 digits only, proper regex validation)
- Email ID (required, valid email format)

Step 2: KYC Verification

Fields:

- Upload Photo (only jpg/jpeg/png, max size 2MB)
- Upload ID Proof (only jpg/jpeg/png, max size 2MB)
- Aadhaar Number (12 digits, proper regex validation)

Step 3: OTP Verification

- User should receive a mock **OTP** (hardcoded, e.g., 123456) on both Phone and Email.
- Candidate should implement OTP input fields (6 digits).
- Validate the entered OTP against the mock OTP before submission.

✅ On successful completion → Save the user data in **JSON Server** (simulating backend).

---

2. Login Flow with Google SSO

- Implement a **Login Page** with only **"Login with Google"** button (no username/password fields).
- On login, use Google OAuth popup.
- Once authenticated, check if the email exists in JSON server:
  - ✅ If yes → Redirect to **Dashboard**
  - ❌ If not → Show **error message** ("User not registered. Please Sign Up first.")

---

3. Dashboard

After successful login, show a simple **Dashboard** with:

- User Profile Information (all the fields submitted during registration).
- Profile picture (uploaded image).
- Logout button.

🔷 Technical Instructions

1. Use **ReactJS (latest version)** – Functional components only (no class components).
2. Use **React Router** for navigation.
3. Use **React Hook Form / Formik** (optional, but recommended for form handling).
4. State management can be done using **React Context API** or custom hooks (Redux not required).
5. Use **JSON Server** for mocking APIs:
   - /users → Save and fetch user registration data.
6. Validations must be handled on client-side before saving to JSON Server.
7. File Upload: Validate file type and size before accepting.
8. OTP should be hardcoded in code (123456) but properly validated on submission.
9. Google SSO must be integrated using standard React packages (e.g., react-oauth/google).
10. Focus more on **functionality and code structure** than UI design.

🔷 Submission Guidelines

- Share your complete project in a **GitHub repository**.
- Include setup instructions in README.md.
- Must run with:

```
npm install
npm start
```

- JSON Server must run with:

```
npx json-server --watch db.json --port 5000
```

- Do **NOT** use ChatGPT/AI code generation tools. All code should be written by you.

🔷 Evaluation Criteria

- ✅ Functional correctness (wizard, validation, OTP, login, dashboard)
- ✅ Clean & modular code structure (components, hooks, context usage)
- ✅ Proper form validation & error handling
- ✅ JSON Server API integration (save & fetch user data)
- ✅ Google SSO integration correctness
- ✅ Reusable components (form inputs, buttons, etc.)
- ✅ Security considerations (no password stored in plain text, validation before saving data)
- ✅ UI/UX (basic, but should be neat and user-friendly)

Example User Flow

1. User clicks **Register → Step 1 (fills basic info) → Step 2 (uploads files + Aadhaar) → Step 3 (enters OTP)**.
2. After validation, user is **saved in JSON Server**.
3. User goes to **Login Page** → clicks **Login with Google** → if email exists in JSON server → redirect to dashboard.
4. Dashboard displays **user profile details**.

⚠️ **Note:** Design is not important, but functionality, validation, and clean code are mandatory.
