SynapShare Frontend

Complete Technical Guide

Table of Contents

---

1. Technologies Used

- **React.js**: Main framework for building the user interface as a Single Page Application (SPA).
- **React Router DOM**: Handles client-side routing for navigation between pages.
- **Tailwind CSS**: Utility-first CSS framework for rapid and responsive UI styling.
- **Framer Motion**: Adds modern, smooth animations to UI components.
- **Axios**: Makes HTTP requests to the backend API.
- **Firebase (Client SDK)**: Handles authentication (Google Sign-In, JWT management).
- **Context API / useState / useEffect**: For state management and side effects.

---

2. Project Structure

```
synapshare-frontend/
├── public/
│   └── index.html
├── src/
│   ├── components/      # Reusable UI components (e.g., Footer, Navbar)
│   ├── pages/           # Main page components (Notes, Discussions, Nodes, etc.)
│   ├── App.js           # Main app component (routing, auth state)
│   ├── index.js         # Entry point
│   ├── firebase.js      # Firebase config and initialization
│   └── ...other files
├── package.json
└── tailwind.config.js
```

---

3. App Initialization and Routing

index.js

This is the entry point for the React app that renders the `<App />` component in the root div.

```javascript
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

App.js

This file sets up all routes for the app using React Router, handles authentication state via Firebase, and fetches and stores the username from the backend after login.

```
javascript
```

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import { useState, useEffect } from "react";
import { auth } from "./firebase";
import Notes from "./pages/Notes";
import Discussions from "./pages/Discussions";
import Nodes from "./pages/Nodes";
import Home from "./pages/Home";
import Login from "./pages/Login";
import Admin from "./pages/Admin";
import News from "./pages/News";
import Search from "./pages/Search";

function App() {
  const [user, setUser] = useState(null);
  const [username, setUsername] = useState("");

  useEffect(() => {
    const unsubscribe = auth.onAuthStateChanged(async (firebaseUser) => {
      setUser(firebaseUser);
      if (firebaseUser) {
        // Fetch username from backend
        const token = await firebaseUser.getIdToken();
        const res = await fetch(`/api/user/${firebaseUser.uid}`, {
          headers: { Authorization: `Bearer ${token}` },
        });
        if (res.ok) {
          const data = await res.json();
          setUsername(data.username);
        }
      } else {
        setUsername("");
      }
    });
    return () => unsubscribe();
  }, []);

  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home user={user} username={username} />} />
        <Route path="/login" element={<Login />} />
        <Route path="/notes" element={<Notes user={user} username={username} />} />
        <Route path="/discussions" element={<Discussions user={user} username={username} />} /
        <Route path="/nodes" element={<Nodes user={user} username={username} />} />
        <Route path="/admin" element={<Admin user={user} />} />
        <Route path="/news" element={<News />} />
        <Route path="/search" element={<Search />} />
        {/* ...other routes */}
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

---

4. Firebase Authentication

firebase.js

Initializes Firebase for use in the frontend and exports auth and googleProvider for use in login and authentication flows.

```javascript
import { initializeApp } from "firebase/app";
import { getAuth, GoogleAuthProvider } from "firebase/auth";

const firebaseConfig = {
  apiKey: "YOUR_API_KEY",
  authDomain: "YOUR_DOMAIN",
  // ...other config
};

const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);
export const googleProvider = new GoogleAuthProvider();
```

Login.js

Handles Google login using Firebase. After login, the user state in App.js is updated, and the JWT is used for backend API requests.

javascript

```javascript
import { auth, googleProvider } from "../firebase";
import { signInWithPopup } from "firebase/auth";

function Login() {
  const handleGoogleLogin = async () => {
    try {
      await signInWithPopup(auth, googleProvider);
      // User is now authenticated
    } catch (err) {
      alert("Google sign-in failed.");
    }
  };

  return (
    <div>
      <button onClick={handleGoogleLogin}>Sign in with Google</button>
    </div>
  );
}

export default Login;
```

---

5. Main Pages

Notes

This component fetches notes from the backend, allows authenticated users to upload new notes (with optional file), and displays a list of notes with download links for files.

```
javascript
```

```
import { useState, useEffect } from "react";
import axios from "axios";

function Notes({ user, username }) {
  const [notes, setNotes] = useState([]);
  const [title, setTitle] = useState("");
  const [subject, setSubject] = useState("");
  const [file, setFile] = useState(null);

  useEffect(() => {
    axios.get("/api/notes").then(res => setNotes(res.data));
  }, []);

  const handleUpload = async (e) => {
    e.preventDefault();
    if (!user) return alert("Please log in to upload notes.");
    const token = await user.getIdToken();
    const formData = new FormData();
    formData.append("title", title);
    formData.append("subject", subject);
    if (file) formData.append("file", file);

    const res = await axios.post("/api/notes", formData, {
      headers: { Authorization: `Bearer ${token}` },
    });
    setNotes([...notes, res.data]);
  };

  return (
    <div>
      <form onSubmit={handleUpload}>
        <input value={title} onChange={e => setTitle(e.target.value)} placeholder="Title" />
        <input value={subject} onChange={e => setSubject(e.target.value)} placeholder="Subject"
        <input type="file" onChange={e => setFile(e.target.files[0])} />
        <button type="submit">Upload Note</button>
      </form>
      <ul>
        {notes.map(note => (
          <li key={note._id}>
            {note.title} - {note.subject}
            {note.fileUrl && <a href={note.fileUrl} target="_blank" rel="noopener noreferrer">
          </li>
        ))}
      </ul>
    </div>
  );
}

export default Notes;
```

Discussions

This component fetches all discussions, allows users to post new discussions, and renders a list of all discussions.

javascript

```javascript
import { useState, useEffect } from "react";
import axios from "axios";

function Discussions({ user, username }) {
  const [discussions, setDiscussions] = useState([]);
  const [title, setTitle] = useState("");
  const [content, setContent] = useState("");

  useEffect(() => {
    axios.get("/api/discussions").then(res => setDiscussions(res.data));
  }, []);

  const handlePost = async (e) => {
    e.preventDefault();
    if (!user) return alert("Please log in to post.");
    const token = await user.getIdToken();
    const res = await axios.post("/api/discussions", { title, content }, {
      headers: { Authorization: `Bearer ${token}` },
    });
    setDiscussions([...discussions, res.data]);
  };

  return (
    <div>
      <form onSubmit={handlePost}>
        <input value={title} onChange={e => setTitle(e.target.value)} placeholder="Title" />
        <textarea value={content} onChange={e => setContent(e.target.value)} placeholder="Conte
        <button type="submit">Post Discussion</button>
      </form>
      <ul>
        {discussions.map(d => (
          <li key={d._id}>{d.title}: {d.content}</li>
        ))}
      </ul>
    </div>
  );
}

export default Discussions;
```

Nodes

This component fetches code nodes from backend, allows users to share code snippets, and displays all nodes with code formatting.

```
javascript
```

```
import { useState, useEffect } from "react";
import axios from "axios";

function Nodes({ user, username }) {
  const [nodes, setNodes] = useState([]);
  const [title, setTitle] = useState("");
  const [description, setDescription] = useState("");
  const [codeSnippet, setCodeSnippet] = useState("");

  useEffect(() => {
    axios.get("/api/nodes").then(res => setNodes(res.data));
  }, []);

  const handleCreate = async (e) => {
    e.preventDefault();
    if (!user) return alert("Please log in to post.");
    const token = await user.getIdToken();
    const res = await axios.post("/api/nodes", { title, description, codeSnippet }, {
      headers: { Authorization: `Bearer ${token}` },
    });
    setNodes([...nodes, res.data]);
  };

  return (
    <div>
      <form onSubmit={handleCreate}>
        <input value={title} onChange={e => setTitle(e.target.value)} placeholder="Title" />
        <input value={description} onChange={e => setDescription(e.target.value)} placeholder="
        <textarea value={codeSnippet} onChange={e => setCodeSnippet(e.target.value)} placeholde
        <button type="submit">Share Node</button>
      </form>
      <ul>
        {nodes.map(n => (
          <li key={n._id}>
            <strong>{n.title}</strong>
            <pre>{n.codeSnippet}</pre>
          </li>
        ))}
      </ul>
    </div>
  );
}
```

```
export default Nodes;
```

Admin

This component is only accessible to admin users and fetches all data for management and moderation.

javascript

```javascript
import { useEffect, useState } from "react";
import axios from "axios";

function Admin({ user }) {
  const [allNotes, setAllNotes] = useState([]);
  useEffect(() => {
    if (user && user.email.endsWith("@admin.com")) {
      axios.get("/api/notes").then(res => setAllNotes(res.data));
      // Fetch other resources similarly
    }
  }, [user]);
  // Admin can delete any note, user, etc.
  return (
    <div>
      <h2>Admin Dashboard</h2>
      {/* Render controls for managing users, notes, discussions, etc. */}
    </div>
  );
}
export default Admin;
```

News

This component fetches and displays the latest technology news from the backend.

```javascript
import { useEffect, useState } from "react";
import axios from "axios";

function News() {
  const [articles, setArticles] = useState([]);
  useEffect(() => {
    axios.get("/api/news").then(res => setArticles(res.data));
  }, []);
  return (
    <div>
      <h2>Latest Tech News</h2>
      <ul>
        {articles.map((a, i) => (
          <li key={i}>
            <a href={a.url} target="_blank" rel="noopener noreferrer">{a.title}</a>
          </li>
        ))}
      </ul>
    </div>
  );
}
export default News;
```

Search

This component sends a search query to the backend and displays results for notes, discussions, and nodes.

```javascript
import { useState } from "react";
import axios from "axios";

function Search() {
  const [query, setQuery] = useState("");
  const [results, setResults] = useState({ notes: [], discussions: [], nodes: [] });

  const handleSearch = async () => {
    const res = await axios.get(`/api/search?q=${query}`);
    setResults(res.data);
  };

  return (
    <div>
      <input value={query} onChange={e => setQuery(e.target.value)} placeholder="Search..." />
      <button onClick={handleSearch}>Search</button>
      {/* Render results grouped by type */}
    </div>
  );
}
export default Search;
```

---

6. Styling and Animation

- **Tailwind CSS**: Used throughout for responsive, utility-based styling.
- **Framer Motion**: Adds smooth transitions and animations to UI elements.

Example of using Framer Motion:

```javascript
import { motion } from "framer-motion";

<motion.div animate={{ opacity: 1 }} initial={{ opacity: 0 }}>
  {/* Animated content */}
</motion.div>
```

---

7. User Experience Features

- **Dark Mode**: Managed via state and localStorage, toggles Tailwind dark classes.
- **Responsive Design**: Layouts adapt to all device sizes.
- **Error Handling**: All API errors are caught and displayed to users.
- **Accessibility**: Semantic HTML, keyboard navigation, and ARIA attributes.

---

8. API Communication

- Axios is used for all HTTP requests.
- Authenticated requests include the Firebase JWT in the Authorization header.
- All endpoints correspond to backend API routes.

---

9. Security

- All sensitive actions (posting, editing, deleting, voting) require authentication.
- Admin routes are protected by checking the user's role/email.
- File uploads are validated on both frontend and backend.

---

10. Summary

The SynapShare frontend is a modern, responsive React SPA. It uses Firebase for authentication, communicates with the backend using Axios, and provides a user-friendly interface for sharing notes, discussions, code nodes, searching, reading news, and admin management. Styling is handled with Tailwind CSS, and animations with Framer Motion. The code is modular, readable, and follows best practices.