Designing 100x Micro blogging platform

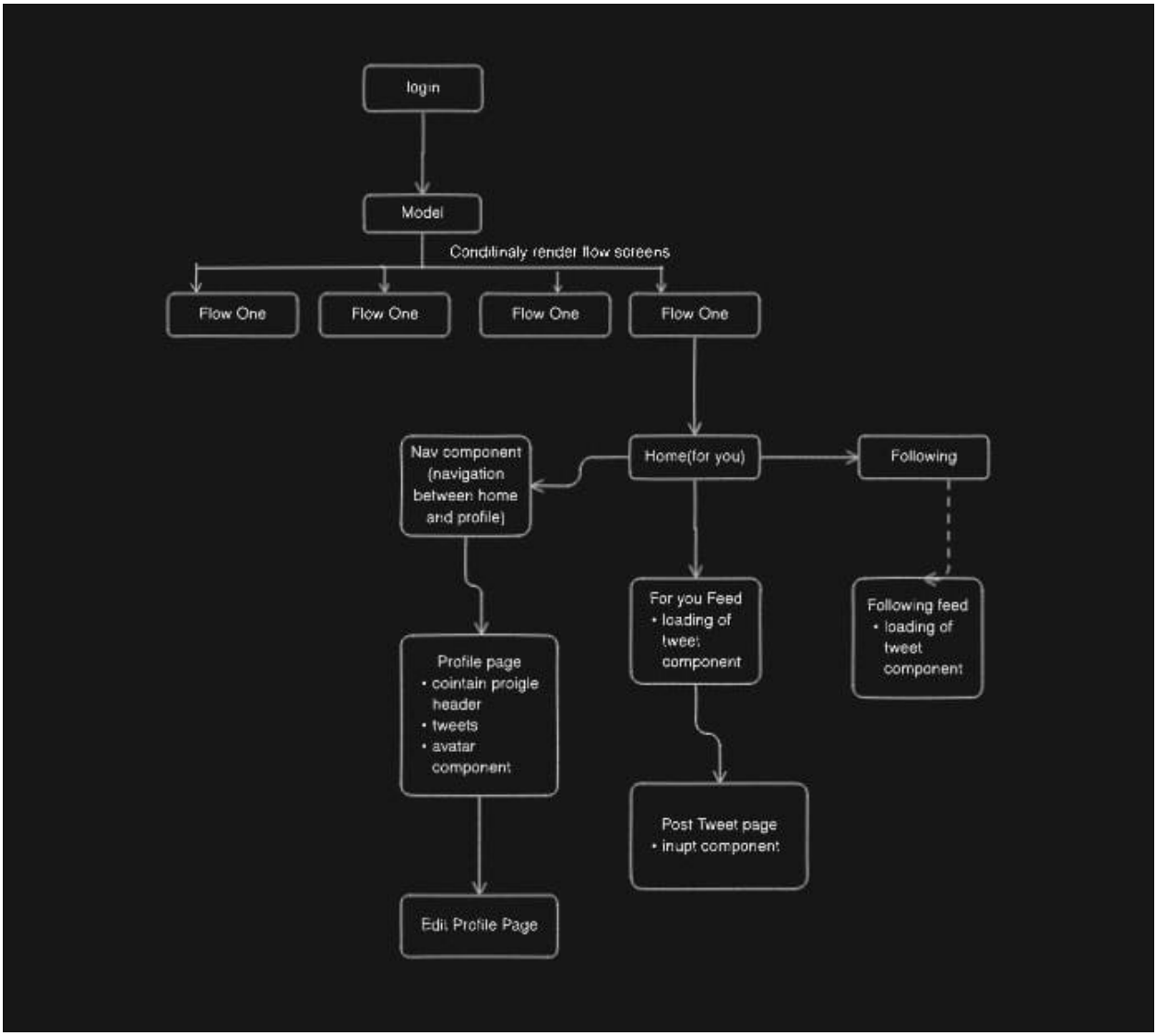
Name:	Vijayabaskar
House:	Debian

Understanding Requirements:

- we are build a twitter clone with basic functionalities, that will allow us to create account, post tweet's , and all other basic featue's of twitter. (note: we are not including image, or video to tweet, we only posting the tweets as of now)
- user registration,
  - User 's can create account , or login if they already have a account
  - edit the account, add profile, header image, handler, and all other info's
  - user can post their tweets and other user's can view that tweets in the feed,
  - user's can intereact with others tweets via likes, commands,retweeet
  - retweeting is a feature that enable us to susbscribe to other's idea(post) and show it in our tweet's collections
- posting micro blogs
  - posting micro blogs have the typical twitter constrain of 280 character and make it work.
  - each tweet that is posted by a particular user is belongs to that user, and have a attribute's like timestamp , device is attract to it.
  - each post contian its own like, share comments and reach count's
- following users
  - Each user can follow many user
  - user's can their followed person tweets in the following tab of the homepage

- Timeline
  - A profile timeline displays the latest Tweets ordered from newest to oldest from a specific public Twitter account.
- searching,
  - user can search other user's and follow them
- mobile responsiveness and accessibility
  - get the navation to the bottom, and move the trending to other page
  - keep it simple, to the mobile user and module the component into different page's that is easy to navigate

Component Break Down


login
Model
Conditinaly render flow screens
Flow One
Flow One
Flow One
Flow One
Nav component (navigation between home and profile)
Home(for you)
Following
For you Feed
• loading of twee: component
Following feed
• loading of tweet component
Profile page
• cointain proigle header
• tweets
• avatar component
Post Tweet page
• inupt component
Edit Profile Page

```
├── public
├── README.md
├── src
│   ├── App.css
│   ├── App.jsx -> handle the Router part of the app, ifAuth ? gotoHOme : gotoLogin page
│   ├── assets -> contin all the asserts foler
│   ├── components
│   │   ├── Button.jsx -> button component with vaients of outline, base , bluebtn, outlineblack and image
│   │   ├── DropDOwn.jsx -> dropdown component (props: name and rest)
│   │   ├── Error.jsx -> Error pop-up component (prpos: onchange )
│   │   ├── Image.jsx ->Image component
│   │   ├── InputField.jsx -> Input Field with defined desing (props : name, type, errors, touched, ...rest )
│   │   ├── Search.jsx -> search Component with defined style (props; search params )
│   ├── index.css
│   ├── main.jsx -> main continer component for all the other component
│   ├── routes -> (folder) differet page and their repesitve component's
│   │   ├── AppFlow (folder)
│   │   │   ├── componenets (folder)
│   │   │   │   ├── Avatar.jsx -> user-profile image cotainer component
│   │   │   │   ├── ProfileHeader.jsx -> presented in the profiel tab , contains user info's
│   │   │   │   ├── ReactIcons.jsx -> like , shrare, cmd
│   │   │   │   ├── Search.jsx -> search bar compnent
│   │   │   │   ├── Trending.jsx -> trenditing component
│   │   │   │   └── TweetHeader.jsx -> tweetheader coponent
│   │   │   ├── EditProfile.jsx
│   │   │   ├── Home.jsx
│   │   │   ├── Nav.jsx
│   │   │   ├── PostTweet.jsx
│   │   │   ├── Profile.jsx
│   │   │   └── Tweet.jsx
```

```
        │ │ ├── context (folder)
        │ │ │ ├── AuthContext.jsx
        │ │ │ ├── AuthProvider.jsx
        │ │ │ ├── FlowNav.js
        │ │ │ ├── index.js
        │ │ │ ├── login.js
        │ │ │ ├── Theme.js
        │ │ │ └── Tweet.js
        │ │ ├── Error404.jsx
        │ │ └── loginFlow (folder)
        │ │ ├── FlowFour.jsx
        │ │ ├── FlowOne.jsx
        │ │ ├── FlowThree.jsx
        │ │ ├── FlowTwo.jsx
        │ │ ├── Modal.jsx
        │ │ └── WelcomePage.jsx
        │ └── serveces (folder)
        │ └── breakpoints.js
        ├── tailwind.config.js
        └── vite.config.js
```

State Management:

- I used context api to manage state across the app's
- Login context

```
import { useContext, createContext } from "react";

export const LoginContext = createContext({
  profile: [
    {
      name: "",
      email: "",
      dateOfBirth: "",
    },
  ],
  getProfileDetais: (profile) => {},
});
```

```
export const LoginProvider = LoginContext.Provider;

export function useProfile() {
  return useContext(LoginContext);
  }
```

- get the input's from the login form flow one and store it in a context to be able to access it across the app
- Tweet Context

```
import { useContext, createContext } from "react";

export const TweetContext = createContext({
  tweet: [
    {
      id: 0,
      userId: "",
      tweetText: "",
      time: "",
      comments: 0,
      retweet: 0,
      likes: 0,
    },
  ],
  postTweet: (tweet) => {},
  updateTweet: (id, tweet) => {},
  deleteTweet: (id, tweet) => {},
});

export const TweetProvider = TweetContext.Provider;

export function useTweet() {
  return useContext(TweetContext);
}
```

- Tweet context, that enable to share state across the app

Routing:

```
const route = createBrowserRouter(
  createRoutesFromElements(
    <>
      <Route
```

```
          path="/"
          element={<WelcomePage />}
          errorElement={createPortal(<Error />, document.body)}
        ><Route
          path="loginOne"
          element={<LoginFlowOne />}
          errorElement={<ErrorPage />}
        />
        {/* {background && <Route path="loginOne" element={<LoginFlowOne />} />} */}
        <Route
          path="loginTwo"
          element={<LoginFlowTwo />}
          errorElement={<ErrorPage />}
        />
        <Route
          path="loginThree"
          element={<LoginFlowThree />}
          errorElement={<ErrorPage />}
        />
        <Route
          path="loginFour"
          element={<LoginFlowFour />}
          errorElement={<ErrorPage />}
        />

        <Route path="home" element={<Nav />} errorElement={<ErrorPage />}>
          <Route
            path="foryou"
            element={<Home />}
            errorElement={<ErrorPage />}
          />
          <Route
            path="following"
            element={<Home />}
            errorElement={<ErrorPage />}
          />
        </Route>
        <Route
          path="profile"
          element={<Profile />}
          errorElement={<ErrorPage />}
        ></Route>
        <Route
          path="editprofile"
          element={<EditProfile />}
          errorElement={<ErrorPage />}
        />
        <Route
          path="postTweet"
          element={<PostTweet />}
          errorElement={<ErrorPage />}
        />
```

```
    </>
  )
);
```

- my entire login flow is handle via createcontext and conditional rendering ,
- and then my main app consist's of routing , and nested routing

Error Handling and User Feedback:

- Created a Error component that displays error
- it handle both the server error's and as well as routing errors

```
const navigator = useNavigate();
const error = useRouteError();

const handleclose = () => {
  if (error) navigator("/");
};
```

routing error is handled via useRouterError

Optimization:

- haven't implemented any optimization techniques up untill now, but going to implement it , and started to reading about optimization strategies, like lazy loading, interfaces, usereducer and useMemo.