Folder Structure —

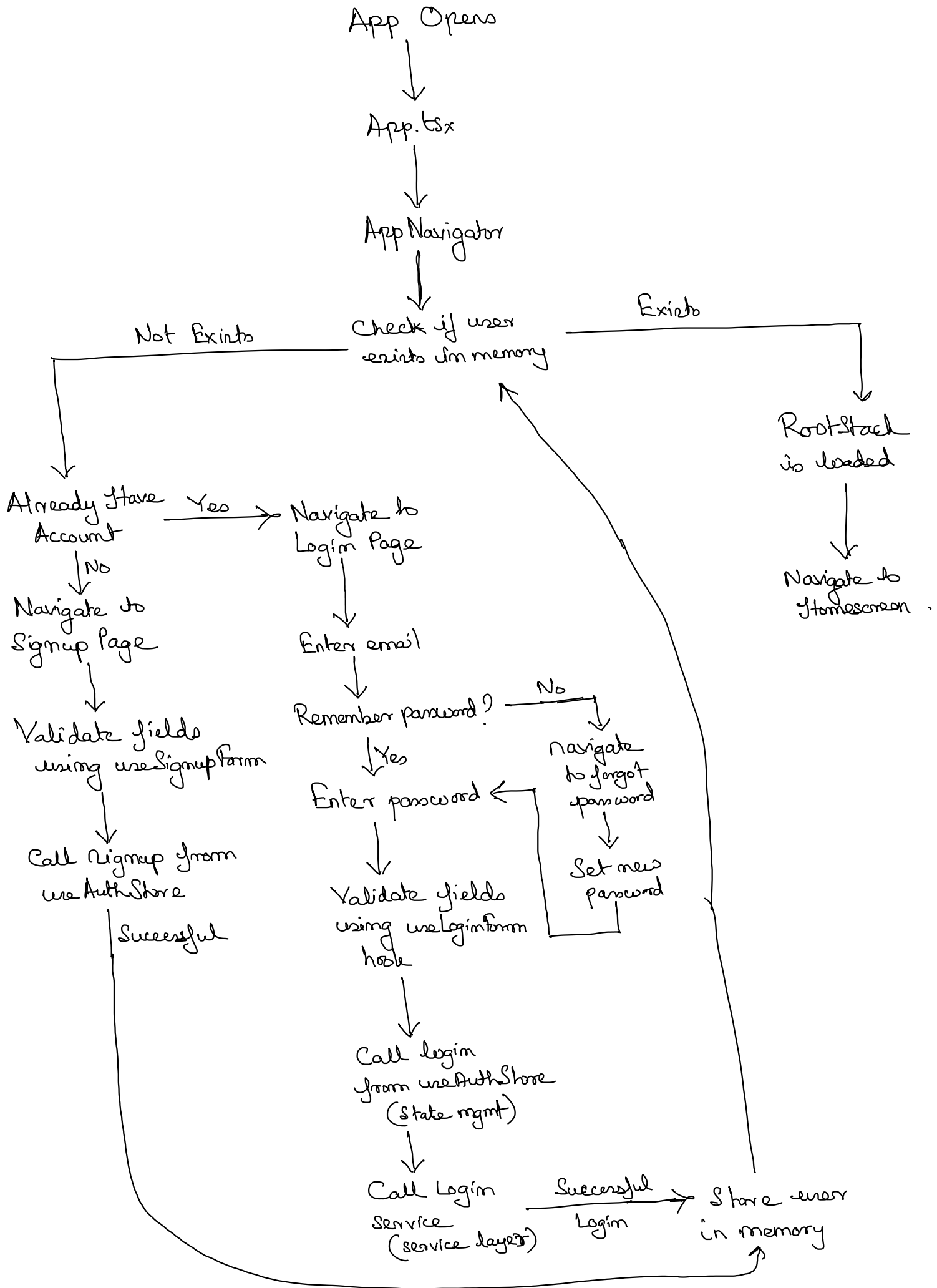## Feature based (related to particular feature)

→ Components (feature specific UI pieces)
→ Screens (Navigation screens)
→ hooks (feature logic hooks)
→ data (mock/static data) [for demo purpose only].
→ Store (state management)

Common

→ hooks (app-wide reusable hooks)

→ models (User, FoodItem, Category structure)

→ Navigator (App Navigation)
  → App Navigator (Takes care which navigation stack is required)
  → Auth Stack (Login, Signup, forgot password, onboarding pages)
  → Root Stack (Home page)

→ Services (Fetching data from APIs, other low level operations).

→ Store (State management (App Wide))

→ Theme (App's colors, typography, etc)

→ Utils (Utility files like validator etc)

App Flow

App Opens

↓

App.tsx

↓

App Navigator

↓

Check if user exists in memory

Not Exists → (left branch)

Exists → RootStack is loaded → Navigate to Homescreen

Not Exists branch

Already Have Account

— **Yes** → Navigate to Login Page

— **No** ↓

Navigate to Signup Page

↓

Validate fields using useSignupform

↓

Call Signup from useAuthStore

— **Successful** → (loops back to Store user in memory)

Login branch

Navigate to Login Page

↓

Enter email

↓

Remember password?

— **No** → Navigate to forgot password → Set new password → Enter password

— **Yes** ↓

Enter password

↓

Validate fields using useLoginform hook

↓

Call login from useAuthStore (State mgmt)

↓

Call Login Service (service layer)

— **Successful Login** → Store user in memory

Store user in memory → (loops back to Check if user exists in memory)

NOTE

* The bottom modal is only present in the homescreen and not in all the auth pages (as given in design) as it is incorrect. Because, the user will not have access to home, orders, support etc pages if he hasn't logged in

* The show/hide button in the passwordInput is text instead of icon due to lack of resources in the design.

* In the authService, we are returning a fixed user due to the lack of backend support/database (as it was mentioned to simulate the backend)