LemmeDoItForU

LemmeCook
Software Architecture Document

Version <1.1>

Revision History

Date	Version	Description	Author
<28/07/2024>	<1.0>	First version of SAD	Bùi Tá Phát
<05/08/2024	<1.1>	Revised version of SAD	LemmeDoItForU Team

Table of Contents

Software Architecture Document

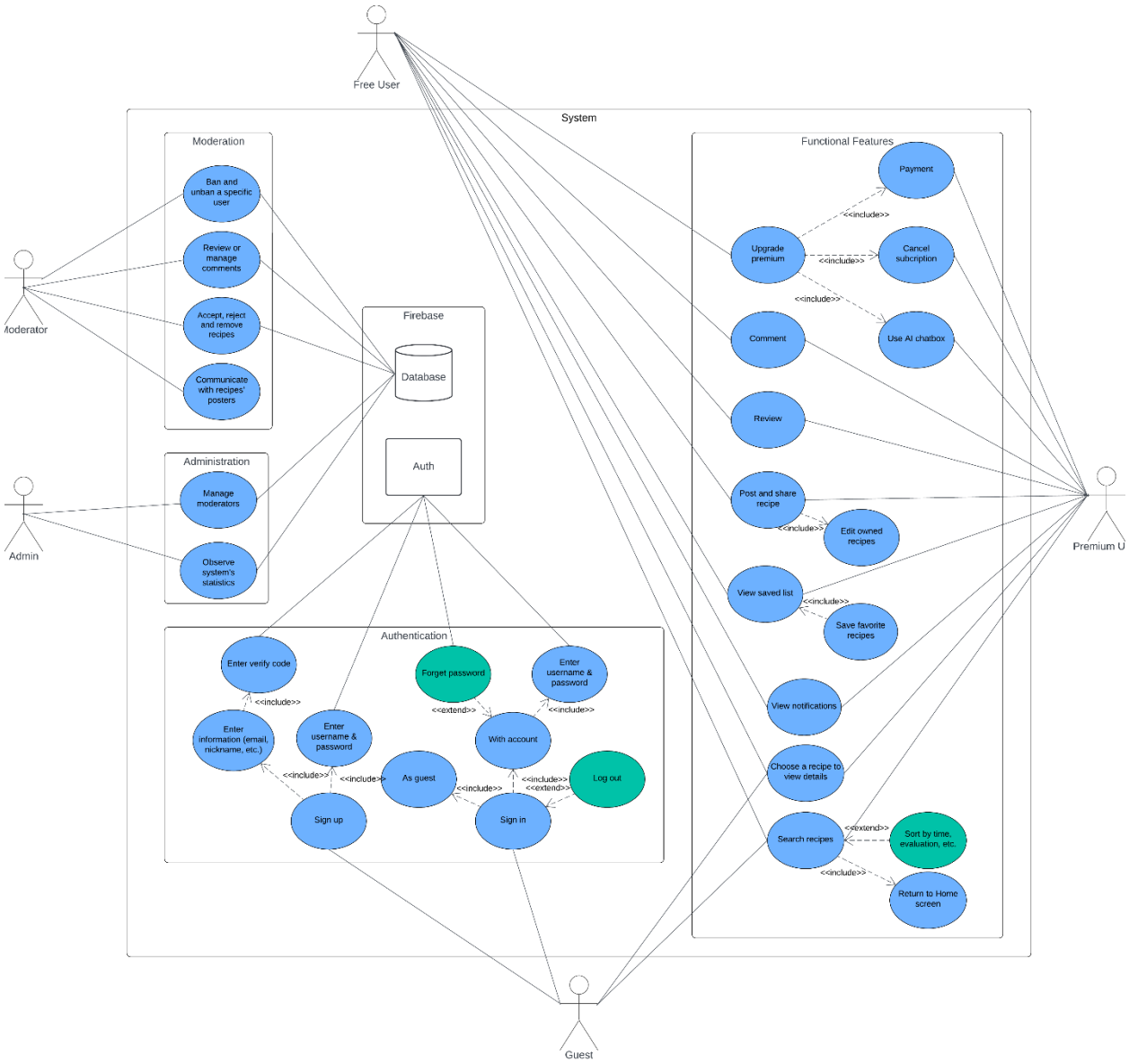
1. Introduction

The purpose of this Software Architecture Document (SAD) is to provide a comprehensive overview of the architecture for the LemmeCook application. The LemmeCook system aims to address the lack of a convenient, unified platform for discovering, sharing, and discussing recipes with an AI assistant that understands dietary needs and preferences. This document outlines the key architectural goals, constraints, and logical components of the system.

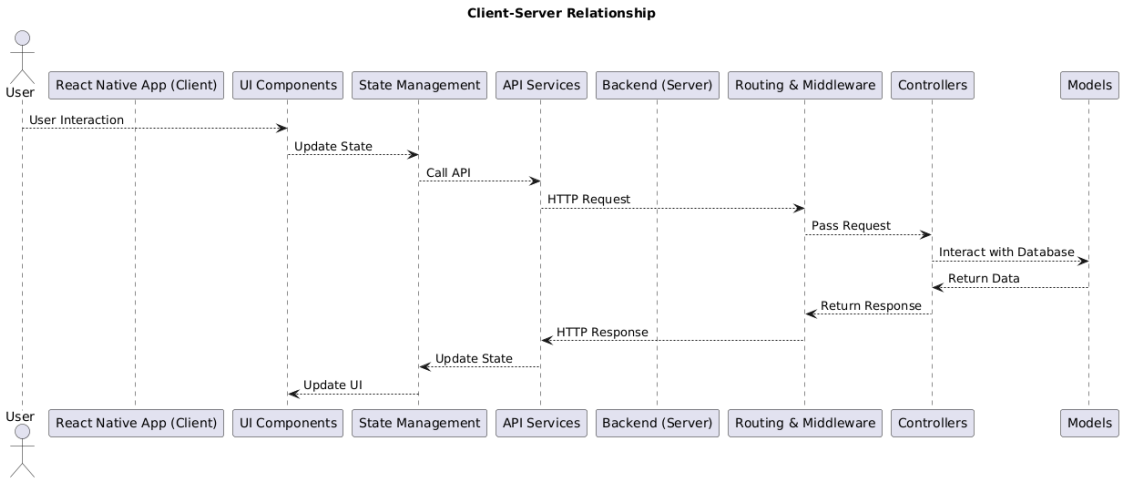
2. Architectural Goals and Constraints

- Security: Ensure the application adheres to security best practices to protect user data and privacy.
- Scalability: Design the system to handle a growing number of users and recipes efficiently. Require Node.js's event-driven architecture.
- Performance: Optimize the application for fast response times and smooth user experience.
- Portability: Ensure the application can be easily deployed on Android devices, and iOS if could.
- Reusability: Design components that can be reused across different parts of the application. Require Typescript programming languages for clarity.
- Usability: Create an intuitive and user-friendly interface.

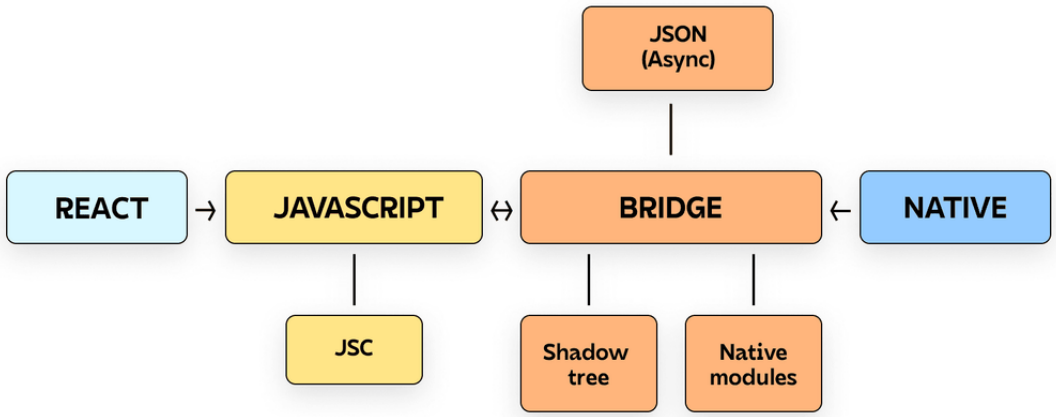
3. Use-Case Model



4. Logical View



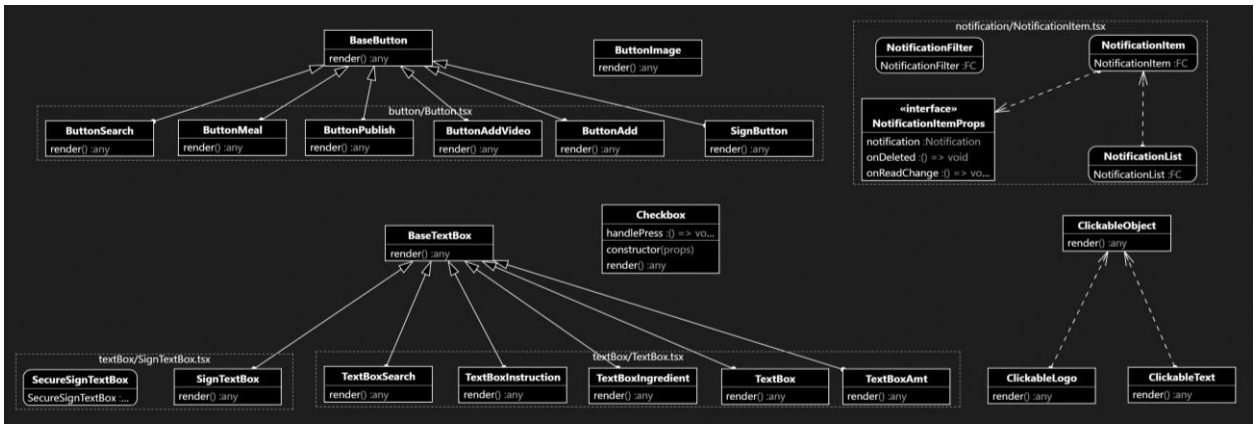
React Native Expo App Client-Server Communication



NodeJS React Native Server-Side Logic

4.1 Component: UI Components

- Frontend component (Client-side React Native).
- These are components extended from React Native components (e.g., <View>, <Text>, <Button>) that make up the user interface.

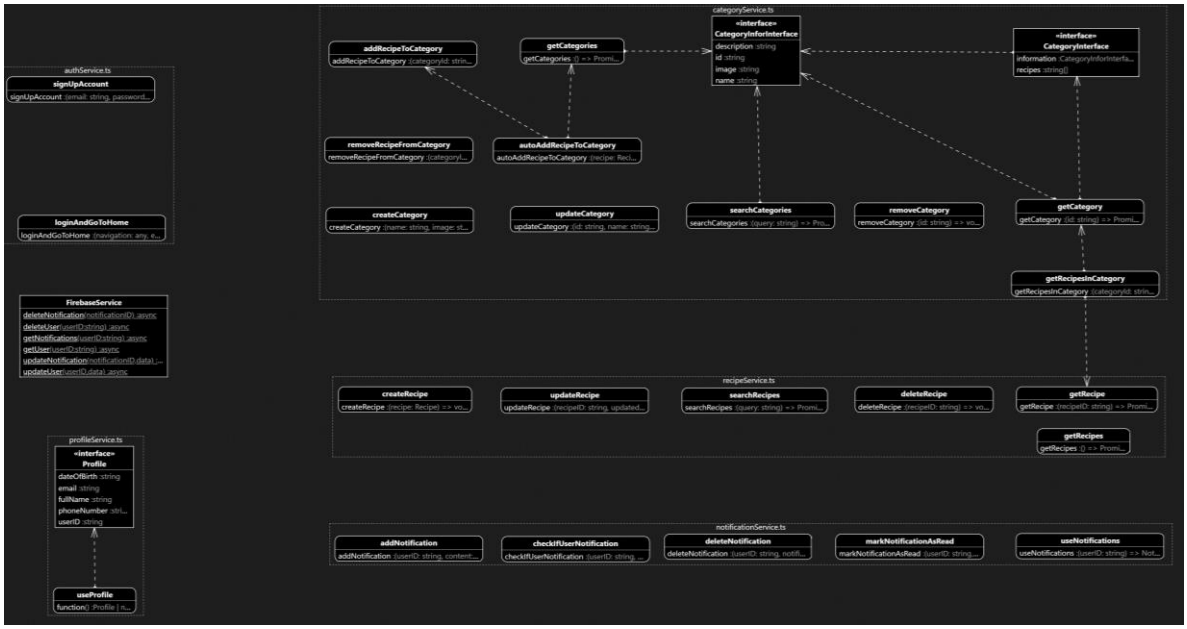


4.2 Component: State Management

- Frontend component (Client-side React Native).
- Manages the state of the application with tools like React's useState, useEffect.

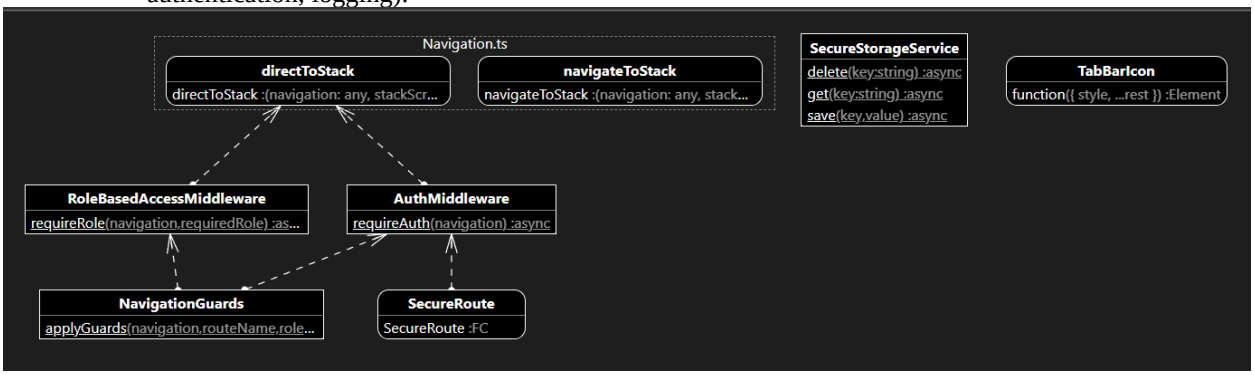
4.3 Component: API Services

- Frontend component (Client-side React Native).
- JavaScript functions that handle network requests from frontend (client-side app).
    - Exception: FirebaseService is service for model to interact with database through Firebase SDK, not API services



4.4 Component: Routing & Middleware

- Backend component (Server-side React Native and NodeJS)
- Manages incoming requests and passes them through various middleware functions (e.g., authentication, logging).

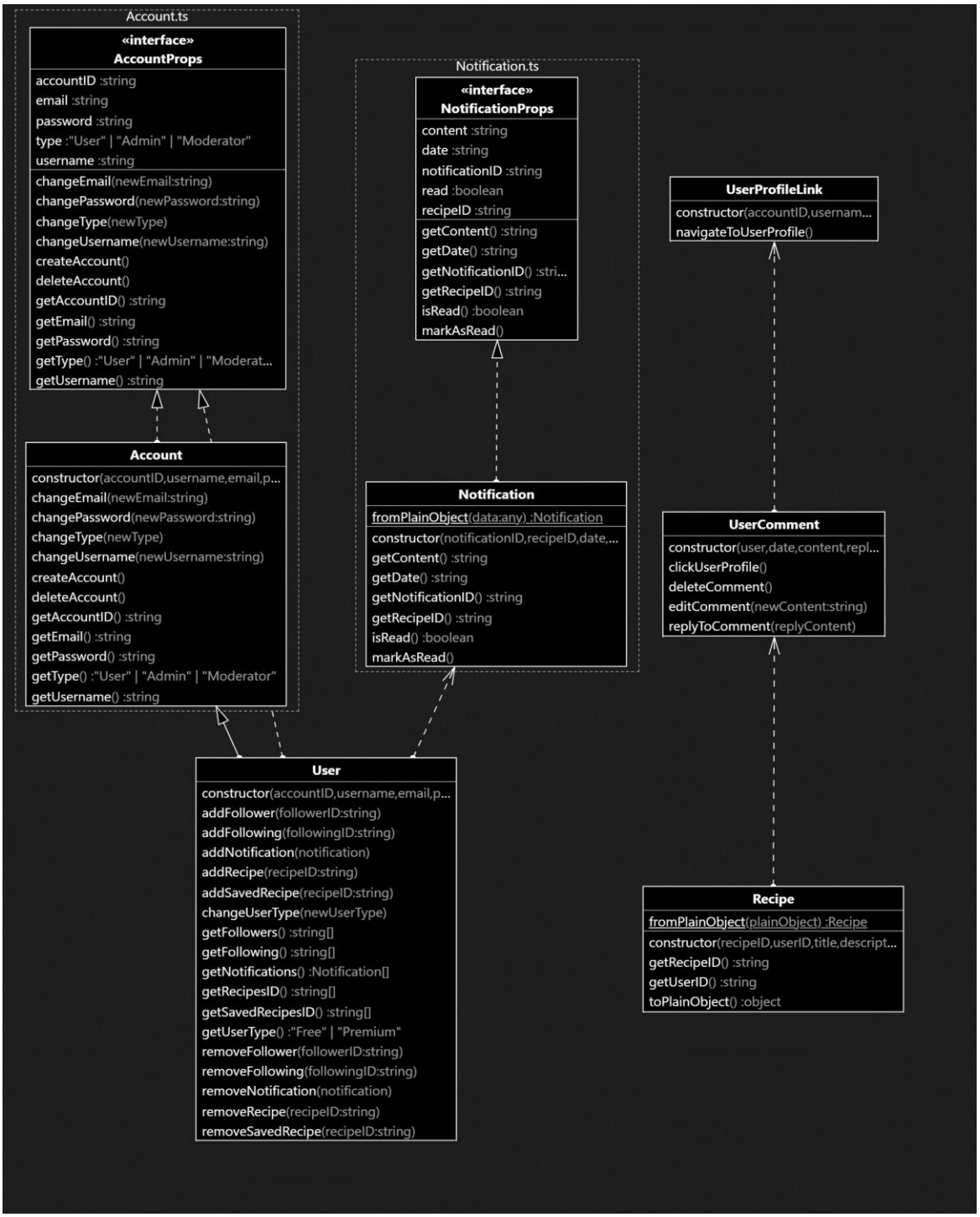


4.5 Component: Controllers

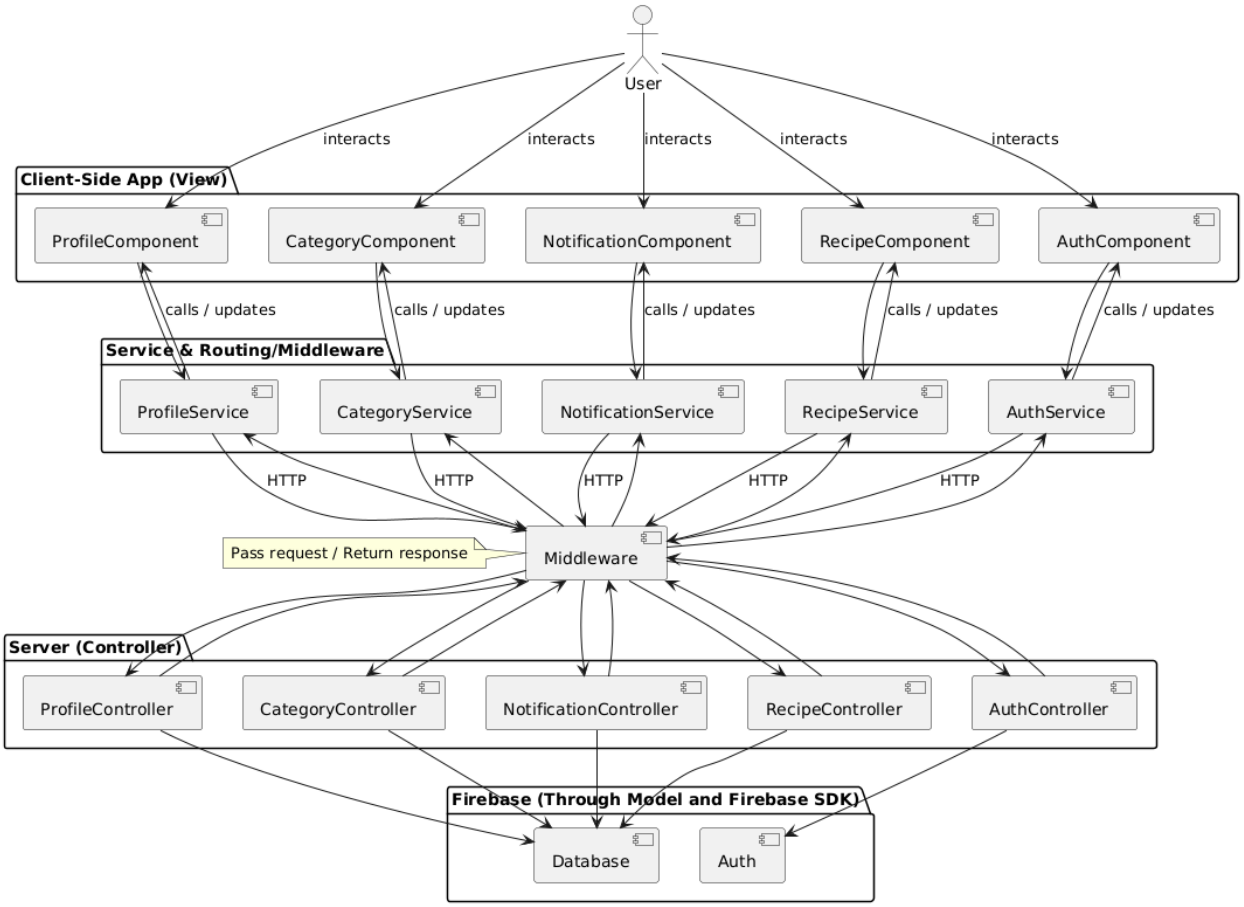
- Backend component (Server-side React Native and NodeJS)
- Handle the logic for different routes and interact with the models to process data.

4.6 Component: Models

- Backend component (Server-side React Native and NodeJS)
- Represent the data structures and handle database interactions.



5. Deployment



- User Interaction: The user interacts with the UI components in the React Native app.
- State Management: The app updates its state based on user input.
- Services Call: The app's API services make HTTP requests to the backend server.
- Middleware: The server's routing and middleware handle the incoming requests.
- Controller Logic: The server's controllers process the requests, interacting with the models to fetch or update data.
- Response Handling: The server sends back a response with the requested data or confirmation of an action.
- State Update: The React Native app updates its state based on the server's response.
- UI Update: The updated state triggers a re-render of the UI components to reflect the new data.

6. Implementation View

/LemmeCook

- **docs/**
- **pa/**
- **src/**
  - **android/**
  - **app/**
  - **assets/**
  - **components/**
    - **controllers/**
    - **models/**
    - **routingAndMiddle/**
    - **services/**
    - **stateManagement/**
    - **UI/**
  - **hooks/**
    - **useElapsedTime.ts**
    - **useToggle.ts**
  - **styles/**