**E-KMC System Design**

1. **High-Level System Design**
   1. **Android App Architecture (Front End)**

UI Layer

Presentation Layer

Domain Layer

Data Layer

Dependency Injection

Dependency Injection

Dependency Injection

Using the Clean Architecture

Directory Structure (Feature Based)

com.xxx.ekmc

|--- common/

| |--- helper/

| |--- base/

| |--- network/

|

|--- feature/

| |--- func/

| | |--- auth/

| | | |--- domain/

| | | | |--- usecase

| | | | |--- model

| | | |--- data/

| | | | |--- local

| | | | |--- remote

| | | | |--- repository

| | | |---- di

| | |

| | |--- kmc/

| | |--- xxxfunc

| |

| |--- presentation/

| | |--- login/

| | | |--- LoginViewModel.kt

| | | |--- LoginScreen.kt

| | |

| | |--- home/

| | |--- xxxscreen

* 1. **Components**

**UI Layer**

* Using Jetpack Compose
* Observes the observable data from ViewModel
* Communicates with ViewModel

**Presentation Layer**

* Uses ViewModel
* Handles screen logics
* Exposes data using StateFlow or SharedFlow
* Access domain layer or data layer directly

**Domain Layer**

* Contains business logics
* Access the data layer

**Data Layer**

* Read or write data sources (local or remote)
  + Local data: Room Database (for KMC data), DataStore (for user settings)
  + Remote data: Read/write the Backend using Retrofit & OkHttp
* Data sources is provided through data repository
* Periodic Synchronization using WorkManager, Polling
  1. **Backend Services (Microservices)**

|  |  |  |
| --- | --- | --- |
| **Service** | **Functionality** | **API Example** |
| Auth Service | Login, token refresh | /auth/login |
| User Profile | Get/update user info | /users/{id} |
| KMC data submission | Post the KMC data | /submit/kmc |
|  |  |  |

1. **Security**

* Frontend – Backend authentication using JWT
* AES encrypted DataStore, SQLChiper encrypted Room DB
* Uses HTTPS connection with TLS 1.2 or above

1. **Testing Strategy**

* Using TDD approach

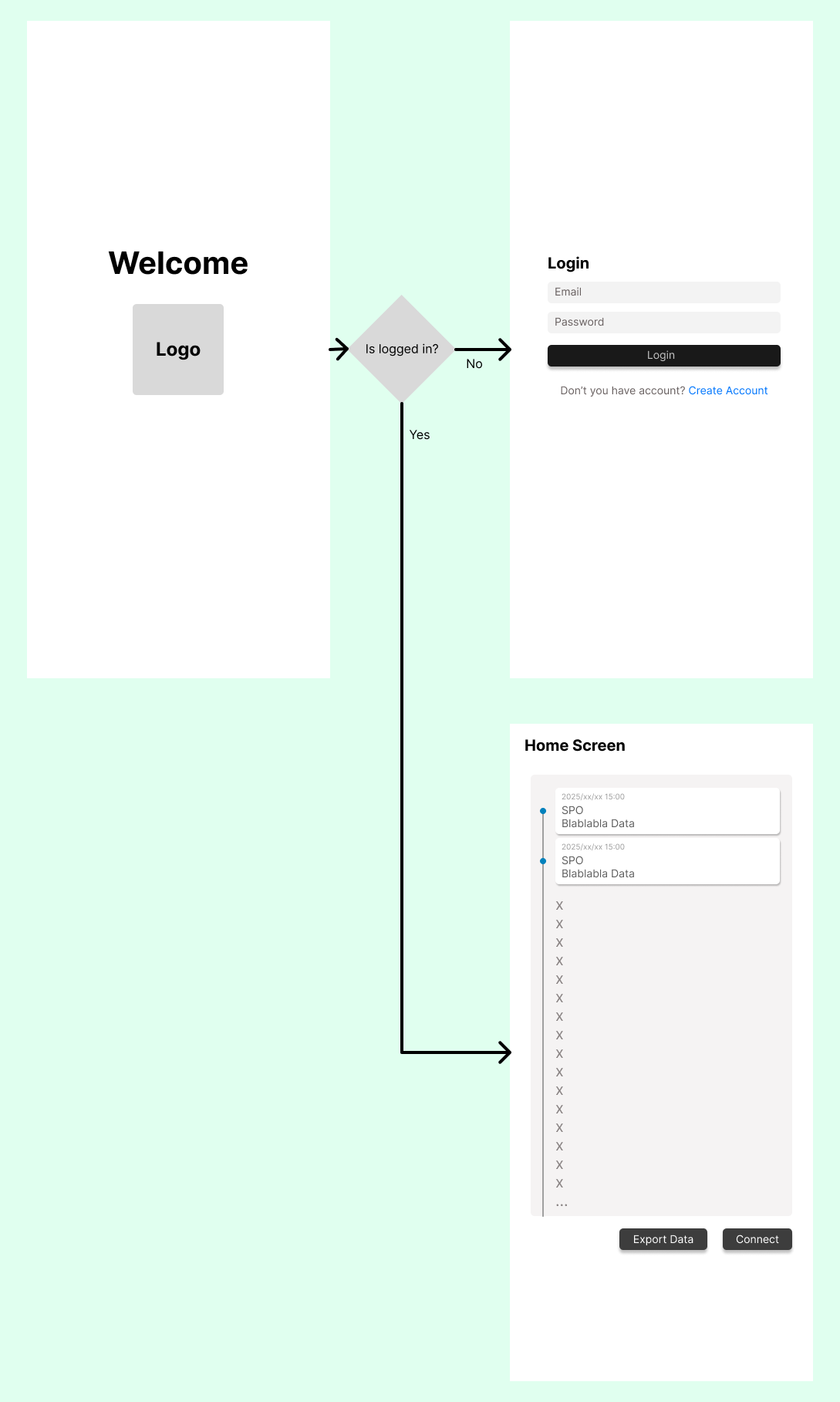
|  |  |  |
| --- | --- | --- |
| **Layer** | **Test Type** | **Tool** |
| UI | UI Tests | Jetpack Compose Test, UI Automator |
| ViewModel | Unit, UI logic | JUnit, MockK, CoroutineTest |
| Domain | Unit Tests | JUnit, MockK, CoroutineTest |
| Data | Unit, Integration tests | JUnit, MockK, MockWebServer |
|  |  |  |

* Testing Type
  + Happy Path
  + Edge Cases
  + Performance
  + Security

1. **Considered Aspects**

* Scalability (Feature based architecture)
* Testability (Clean separation of concerns)
* Maintainability (Independent layers)
* Security (Data encryption, HTTPS connection)

1. **User Journey**



For temporary, the Login Screen will not be implemented yet

1. **Fee**

Working in progress