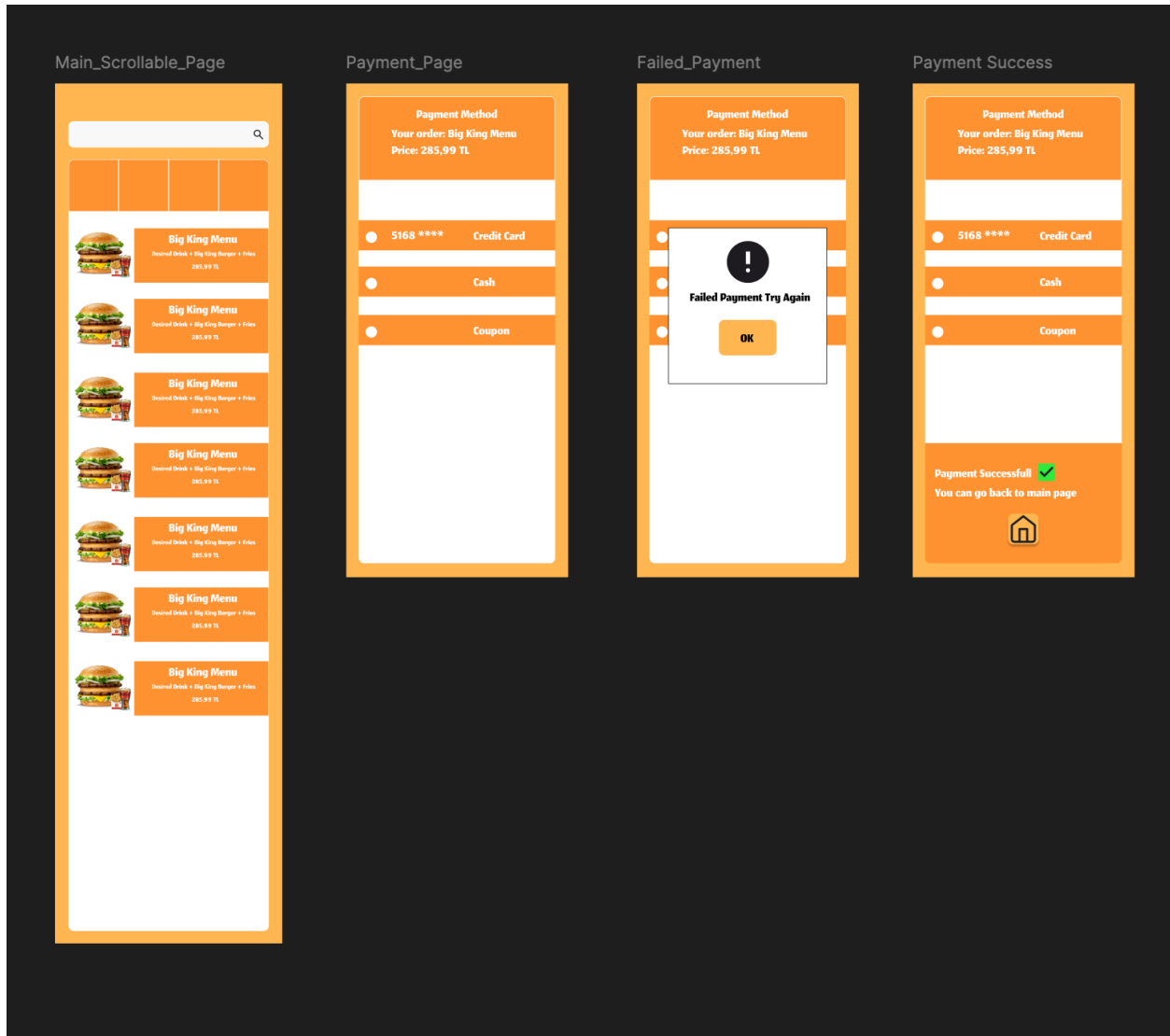


Week-1

Day-1

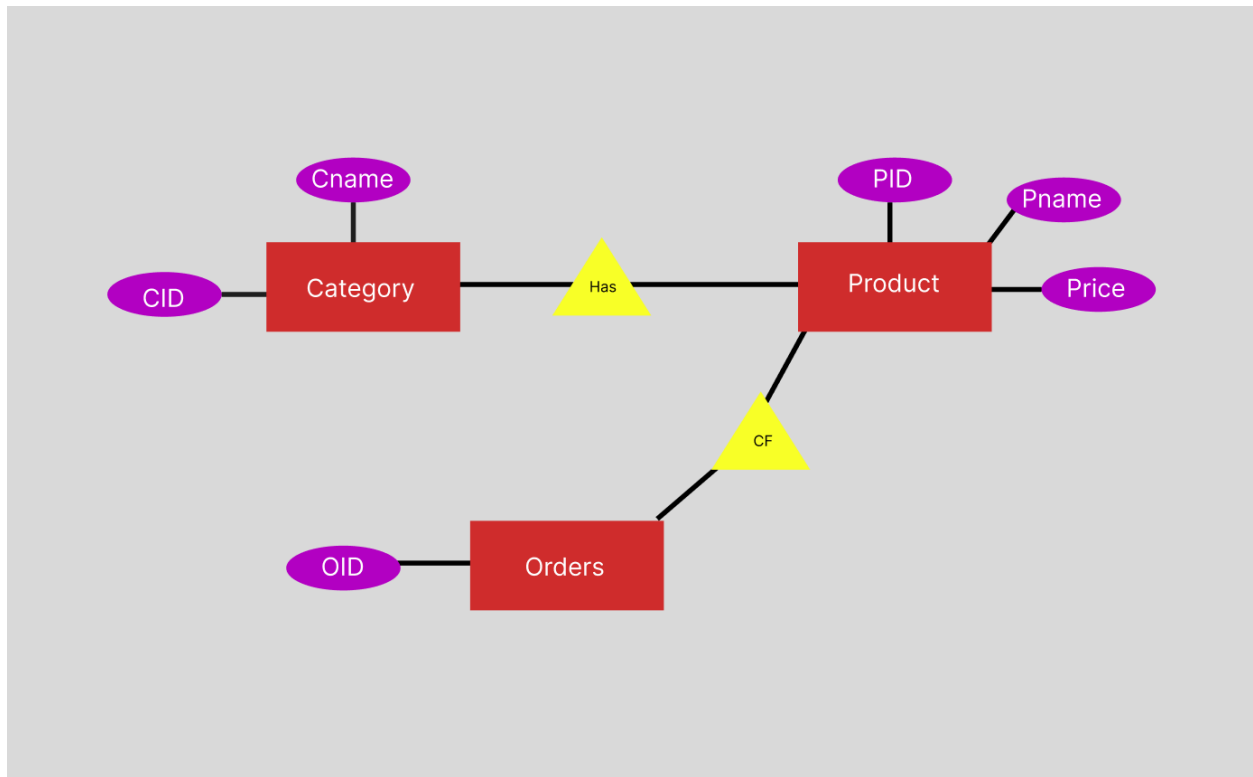
I designed the overall layout of the project on Figma.

https://www.figma.com/design/fnHCgcXtSB6N3x3ogGptQX/app_design?t=IDKzonSNWRm6lsxO-0



After the design I decided on the DB structure and created a Relational Model Diagram.

https://www.figma.com/design/xfttypk7pkmZrzDZFaBcmT/Relational_Model_Diagram?t=DKzonSNWRm6lsxO-0



Finally at the end of day 1, I created a Mobile App structure without the web service and SQL. Therefore I used data classes on Kotlin and used Mock Data.

I opened an Empty Project on Android studio and added 3 files and classes.

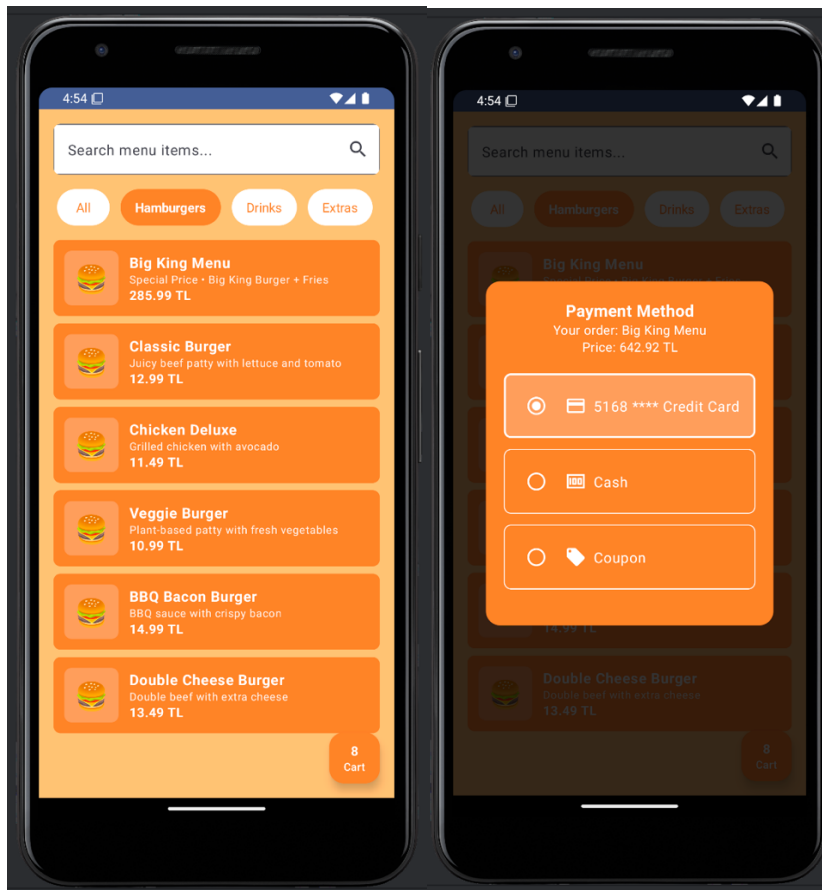
FoodModels.kt (which is under models package)

MainScreen.kt (under ui.theme)

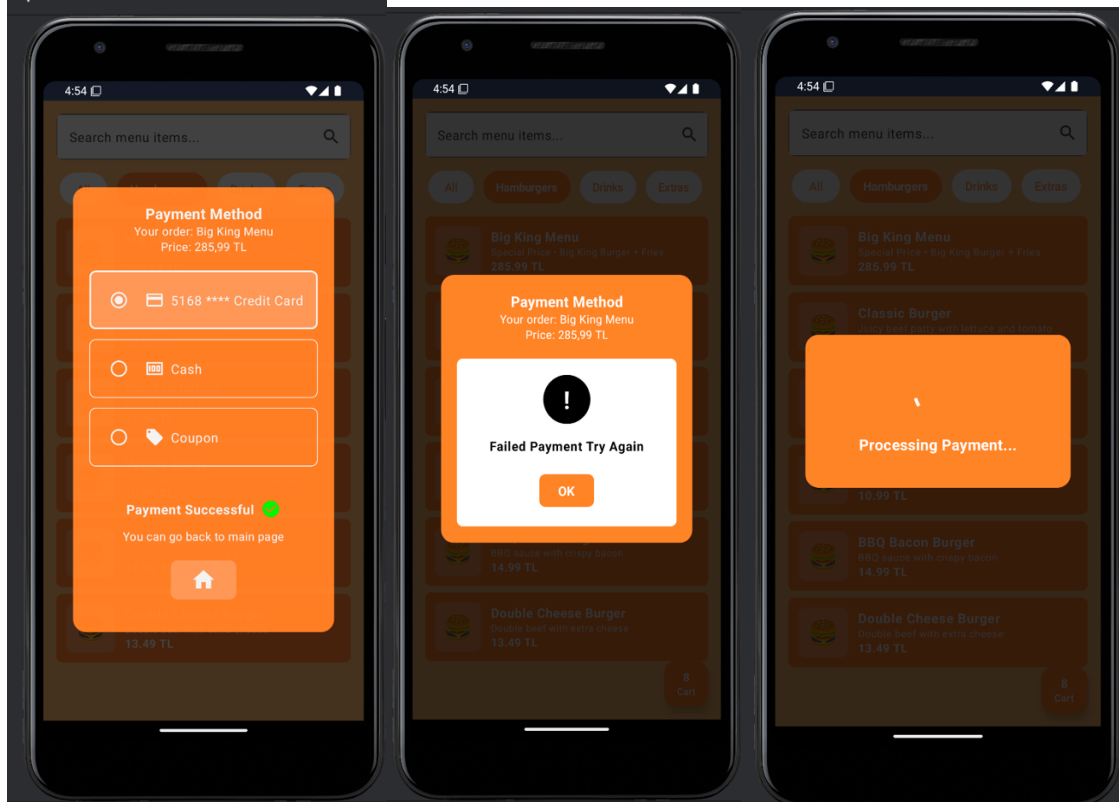
PaymentDialogs.kt (under ui.theme)

<https://github.com/ewok0116/internshiptemplate/tree/main/projects/Frontend/foodorderi ngapp>

Payment and Selection Page



Success, Failed Payment and Processing Pages



What I need to do in terms of Kotlin?

After DB processes are complete, I need to change the mock data and also on payment selection page only credit card is selected I also need to change that and depending on feedback the successful payment page might stay still instead of automatically leading to selection page.

Day-2

I got feedback on fixing and adding some features to the app

- Cart button should be bigger
- There should be a Dialog that shows products, and they should be deletable or incrementable
- There should be toast or toast like structure that appears after adding products to the bar (I used snackbar)

- More payment methods should be added
- Navigating back to home page should be manual
- The text of payment successful dialog should be fixed
- Build variant like theme structure should be made
- If wanted, a receipt should appear after the payment

I did fix the issues and added the wanted features to the application.
Only this one is still in progress

- Design patterns should be searched

In terms of Build Variant

Build Configuration (build.gradle)

```
flavorDimensions += "company"
productFlavors {
    create("companyA") {
        dimension = "company"
        applicationIdSuffix = ".companya"
        versionNameSuffix = "-companyA"
        manifestPlaceholders["appName"] = "Company A Food"
        buildConfigField("String", "COMPANY_THEME", "\"ORANGE\"")
    }

    create("companyB") {
        dimension = "company"
        applicationIdSuffix = ".companyb"
        versionNameSuffix = "-companyB"
        manifestPlaceholders["appName"] = "Company B Food"
        buildConfigField("String", "COMPANY_THEME", "\"GREEN\"")
    }
}
```

Theme Provider (Theme.kt)

```
val LocalAppTheme = compositionLocalOf { ThemeManager.getCurrentTheme() }
```

```
@Composable
```

```
fun FoodOrderingAppTheme(content: @Composable () -> Unit) {
    val currentTheme = ThemeManager.getCurrentTheme()
```

```
    CompositionLocalProvider(
        LocalAppTheme provides currentTheme
```

```

    ){
        MaterialTheme(content = content)
    }
}

```

Before (Hard-coded colors):

```

@Composable
fun ProductCard() {
    Card(
        colors = CardDefaults.cardColors(
            containerColor = Color(0xFFFFF8C42) // Hard-coded orange
        )
    ){ /* content */}
}

```

After (Dynamic colors):

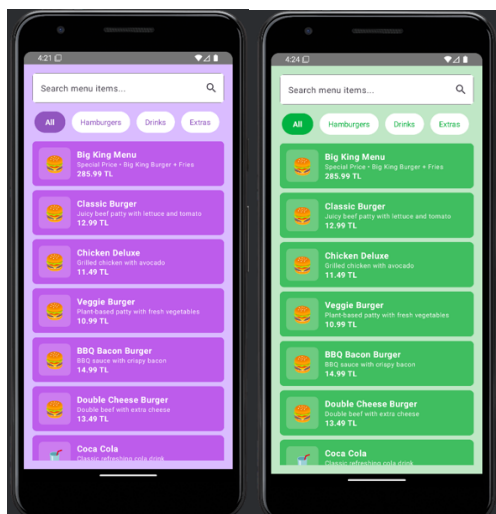
```

@Composable
fun ProductCard() {
    val theme = LocalAppTheme.current // Get current theme

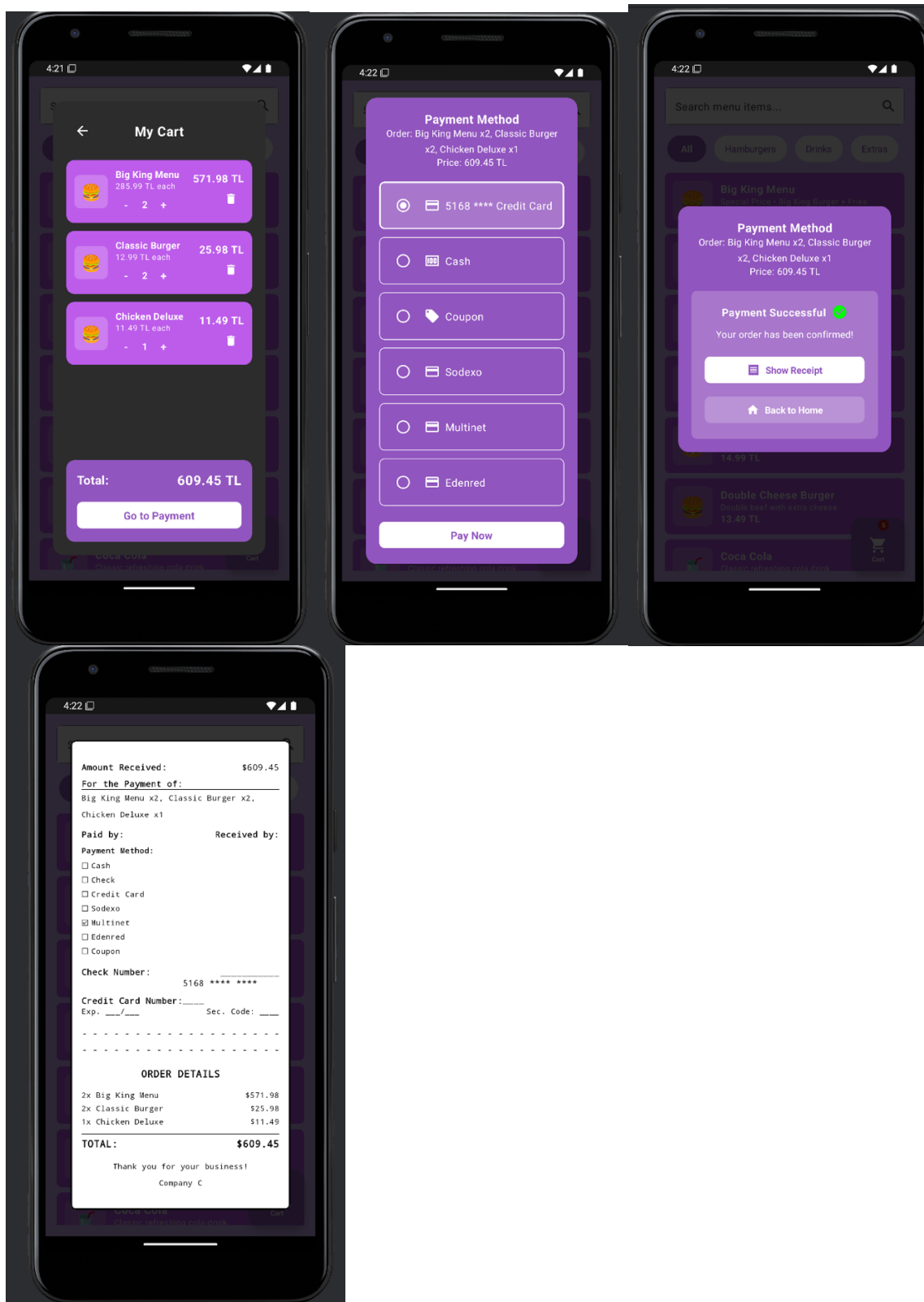
    Card(
        colors = CardDefaults.cardColors(
            containerColor = theme.cardColor // Dynamic color!
        )
    ){ /* content */}
}

```

And these are the new variants aside from orange



In addition, added receipt, cart dialog and more payment methods



Still I need to edit the receipt, and delete some of the texts. Moreover, need to study more about design patterns and the one I'm currently using which is MVC.

Day-3

I fixed Receipt, added multiple payment methods, make the payment screen non-cancellable, and got rid of randomization of payment success or fail. After that I studied design patterns and look into MVVM. Since, I realized that I was not using the pattern MVC but I was using MVVM (Model View Viewmodel). It has two-way data binding and views don't directly manipulate the data. After that I searched whether there were any better patterns that I can use if the project got bigger and there was. MVI (Model View Intent)

The MVVM was usable when there are approximately 10 pages. MVI is used in bigger and more complex projects. In addition both can be combined for usage aswell. For my project, since it is a small project I will be using MVVM but later on if a customer want their app to have an advanced carts that has promotion or discount like features. MVI would be a better design approach.

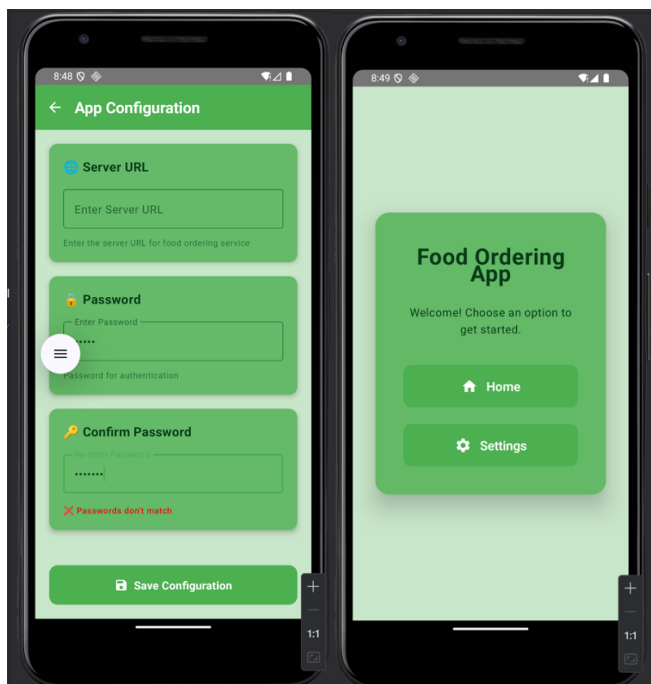
MVI Flow

User Action → Intent → ViewModel → New State → UI Update → User Action

MVVM Flow

User Action ↔ ViewModel ↔ UI (bidirectional)

Moreover, I finally fixed the GitHub branch error. In addition I did the homepage and config page.



Day-4

I started the day by learning .NET and watched some videos to get to know .NET
<https://youtu.be/9J1nJOivdyw?si=ZaFV2UjNUapBG8fv> : General What is Web Server

https://youtu.be/G5b1Zd367sA?si=0JfLxf_wM_QJjgR0 : Making WebSocket Server .NET 8

https://youtu.be/DeegkS0Vf9o?si=Y1zILKi_-F63JS43 : Connect DB C# in .NET

<https://youtu.be/OlkpRoE2mYQ?si=VH7EBdfmNLXebGX> : About ASP

Then learned that if I want to download SSMS I need a Windows computer since for Mac, I need to use Docker or other tools to make SSMS usage in Mac (I actually tried the docker way but it was not wanted so I dropped it during half way.) Then, I downloaded SSMS to a Windows device after that I created a DB which can be found in my projects Backend folder. After that, I downloaded .NET 8 to my Mac and on VScode I wrote a simple connection code then tried to connect to the Windows computer however, I failed on that part. Mainly because of authentication which I don't understand since even the Windows device was giving error when SQL auth. is chosen, and my Mac device cannot connect to the windows device when authentication is chosen as Windows auth. Therefore I decided to let that part go and look into .gitignore and found the files that should not be on my GitHub. removed those files from my internshiptemplate and added the .gitignore file.

Gradle build files

.gradle/
build/
app/build/
*/build/

Generated files

*.apk
*.aab
*.dex
*.class

Local properties

local.properties
gradle.properties

IDE files

.idea/
*.iml
.vscode/

OS files
.DS_Store
Thumbs.db

Logs
*.log

Keystore files (if you have any)
*.jks
*.keystore