Google Summer of Code

GSoC'24 PROPOSAL - The Mifos Initiative

Functional Enhancements to Mobile Wallet for G2P Use Cases

Organization: Mifos Initiative

Project Name: Functional Enhancements to Mobile Wallet for G2P Use Cases

Candidate Name: Pratyush Singh

Expected Project Size: 350 hours

Mentors:

- Rajan Maurya
- Avinash Vijayvargiya

Contents

1. Project Idea

Abstract

The project focuses on the enhancement and refinement of a feature-rich and secure Mobile Wallet application tailored for Government to Person (G2P) payments, demonstrating a potent tool for fintechs and financial institutions. Originating as a Google Summer of Code initiative from 2017-2023, the project delivered the initial mobile wallet framework, evolving into a sophisticated reference application with two primary apps: PixieCollect and MifosPay, later consolidating focus on MifosPay. The developmental trajectory of the project saw significant milestones: integration with Mojaloop transaction flows, user experience enhancements, support for Kotlin, and implementation of standing instructions and merchant transactions. Further advancements include integration with Fineract CN, multi-theme support, migration to Kotlin, and a transition towards a multiplatform approach using Kotlin multi-platform. The project aims, in 2024, to finalize G2P functionalities, ensuring production-readiness, improving security, and adopting modern development practices such as Jetpack Compose and leveraging Mifos' Android SDK. It underscores a commitment to evolving into a generic wallet management system through the Mifos X framework, incorporating cutting-edge features like the latest Payment Hub EE version, a new notifications framework, and a redesigned architecture to align with the Open Wallet Foundation's principles. This ambitious endeavor promises to redefine the landscape of mobile wallets, emphasizing security, extensibility, and a seamless user experience for G2P payments and beyond.

2. Implementation Details

2.1 Integrate latest version of Payment Hub EE

2.2 Integrate Mifos' notifications framework

2.3 Incorporate Mifos' new design library

- Currently, our Android projects display a variety of user interfaces, with even individual applications experiencing inconsistencies in UI elements. To address this issue and ensure uniformity both within and across our apps, we need to develop a new design library. This will establish a consistent UI framework for all our projects.

- At the time of drafting this proposal, Mifos is utilizing an outdated version of the Mifos UI Library that relies on XML. It's important to acknowledge that this will become obsolete once we transition our project to Compose. Currently, we are defining our reusable components within the :core:ui module of the project.

- The integration of a new UI library will be contingent upon the nature and structure of this library. Should it contain components common across all our Android projects, we could seamlessly replace the existing definitions with those provided by the library. For instance, consider a **Login Screen** that appears in all our projects. By defining this screen within our new library and deploying it, we can ensure consistent UI across all projects.

2.4 Migrate XML to Jetpack compose

- Converting the project to jetpack compose is of utmost priority for this years GSoC. Currently we are using Fragment + Compose approach to migrate our project. We are designing compose UI and then setting the content in our fragments. This typically looks like the following

```
    setContent {
        AppTheme {
            ComposeScreen()
        }
    }
```

- This method though not wrong, leads you to a code that is bloated and outside of the Compose mentality. Hence we need to start introducing the **Navigation Compose**. The way it will work in our project is that each **NavController** will be associated with a single NavHost composable. The **NavHost** would link the NavController with a navigation graph that specifies the composable destinations that we should be able to navigate between. As we navigate between composables, the content of the NavHost is automatically recomposed. Each composable destination in our navigation graph is associated with a route which is basically a String that defines the path to your composable

- We will need to define a file lets say `AppNavigation.kt` to define screen names and routes for Navigation. This would typically look like:

```
    enum class Screen {
        Screen1, // this will be HomeScreen in our project
        Screen2, // this will be PaymentsScreen in our project
        ... // other screens
    }
    sealed class NavigationItem(val route: String) {
        object Screen1 : NavigationItem(Screen.Screen1.name)
        object Screen2 : NavigationItem(Screen.Screen2.name)
        ... // other screens
    }
```

- We then need to define **NavHost** with our screens in `AppNavHost.kt`. A demo implementation would like the following:

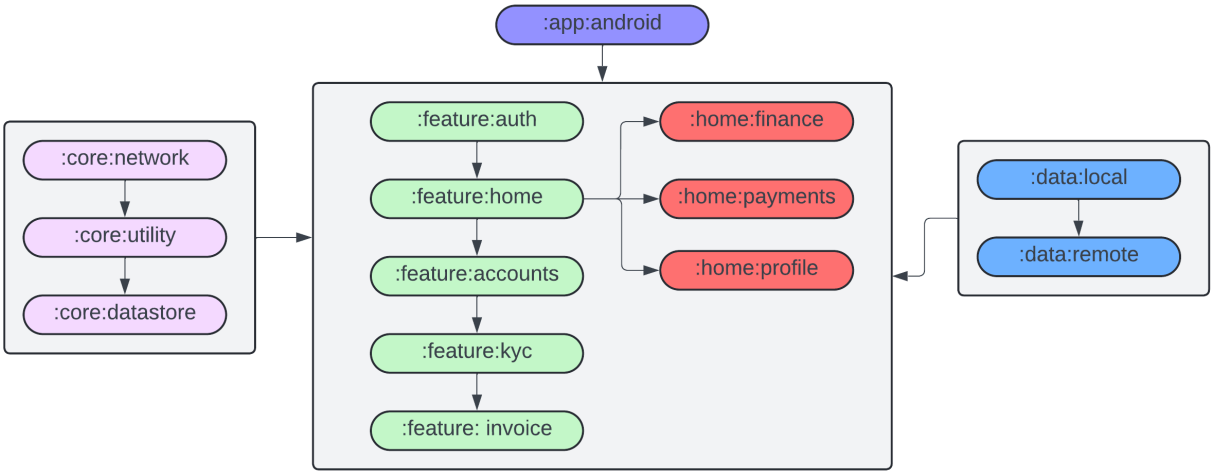
```
    @Composable
    fun AppNavHost(
        modifier: Modifier = Modifier,
        navController: NavHostController,
        startDestination: String = NavigationItem.Splash.route,
        ... // other parameters
    ) {
        NavHost(
            modifier = modifier,
            navController = navController,
            startDestination = startDestination
        ) {
            composable(NavigationItem.Splash.route) {
                SplashScreen(navController)
            }
```

```
            composable(NavigationItem.Screen1.route) {
                Screen1(navController)
            }
            ... // other composables
    }
```

- After this is done we will call *AppNavHostinside* inside our `MainActivity.kt` file.This will lay the foundation for initial setup and will be crucial when we refactor the **Bottom Navigation** to compose.
- Since we already have a lot of resusable components in our project, migrating the project to compose while integrating NavigationCompose shouldn't take a lot of time. I will communicate any UI revamp ideas with my mentor and will implement the same if he gives me a nod

2.5 Basic implementation of multi-platform

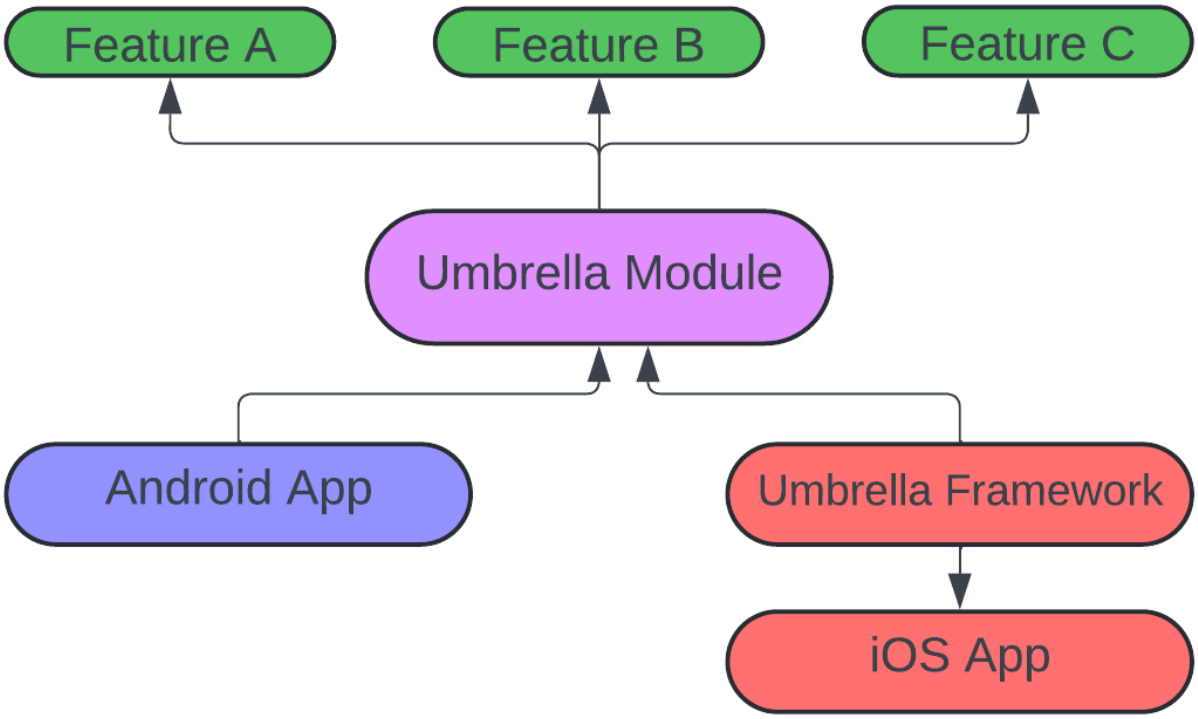
- Implementing *Kotlin Multiplatform* (KMP) is another huge undertaking in this years GSoC. Once implemented, it will upgrade our native android application to a cross platform application
- Our project is currently broken down into a few modules and I believe we can break it down further based on different features. Each feature module handles its own scope of responsibility, usually covering the functionality of a single screen or a closely related group of screens. Right now we only have `:feature:auth` module and we can extend it to different such features. We can create a *database module* inside of core to find all the necessary classes for our local database
- We can be ambitious and take `:data` module out of core and treat it as a separate module. It will handle the data structure serving the feature modules in a well-separated manner.A data module can be associated with multiple feature modules to ensure that each feature module has complete access to all the necessary data.Every data module includes the necessary modules for **Koin**, module-specific data classes, and the repository interface and implementation



- To ensure cross platform application KMP shares the business logic between the platforms and it is of utmost importance that we migrate anything java related to kotlin. That includes migrating *SharedPreference* to **Data Store**, *Retrofit* to **Ktor**, *Hilt* to **Koin** and *DbFlow* to **SqlDelight**. We also have the option to choose between Koin & Kotlin-Injection and also between SqlDelight & Realm. The end

purpose of their alternatives are the same and hence I will discuss the better alternative with my mentor before I start implementing any of them

- Until this point we have divided our project into different features and now we have to share this with the iOS platform.The iOS app can depend on one framework generated by the Kotlin Multiplatform module. When you use several modules, you need to add an extra module depending on all of the modules you're using, called an umbrella module, and then you need to configure a framework containing all of the modules, called an umbrella framework.



2.6 Implement Playstore release github action pipeline

2.7 Update wallet framework to be make use of Mifos' Android SDK

2.8 Complete migration of Java code to Kotlin

In the process of migrating to Kotlin, we significantly enhance our codebase by adopting Kotlin's more concise syntax, which notably reduces boilerplate code. Furthermore, Kotlin's built-in null safety and powerful features like coroutines and flows add substantial value to our development process. However, our ultimate objective is to transition our project to **Kotlin Multiplatform** (KMM). To achieve this, it's imperative that our entire codebase is converted to Kotlin. At the time of drafting this proposal, **only 1.6%** of the project remains to be migrated. When I first started contributing to this project, a considerable portion needed to be transitioned from Java to Kotlin. Therefore, I initiated the conversion process on a package-by-package basis, systematically working towards our goal of a fully Kotlin-based codebase.

At the time of drafting this proposal the following files are still haven't been migrated to kotlin:

- VerifyUser.java

- WrapContentHeightViewPager.java
- FetchAccounts.java
- FetchAccountTransaction.java
- RegisterUser.java
- DownloadTransactionReceipt.java

Although this task is not of immediate high priority, it plays a critical role in the grand scheme of our project objectives. Therefore, I plan to submit pull requests for these changes as promptly as possible. Should I encounter any challenges in completing this task before the Google Summer of Code timeframe, I am committed to resolving it before the conclusion of the first coding phase.

2.9 Improving the security framework

2.10 Exploring PoC Architecture for Open Wallet Foundation Alignment

3. Contributions to Mifos

Below are the links to my contributions at the time of submitting this proposal :

Merged Pull Requests

1. PR #1926: Modified Error message in case of wrong endpoint
2. PR #1927: Fixes the behaviour of filters in savings section
3. PR #1930: Fixes unresponsive nature of 'Change Passcode' in Settings
4. PR #1981: Fixes the white background in dark mode
5. PR #1984: Fixes the scrolling feature in landscape mode
6. PR #2017: Fixed the crashing of app when changing main mobile theme
7. PR #1910: Fixes App Crash when orientation is changed to landscape mode
8. PR #1904: Added Country Code Picker in the Signup form
9. PR #1899: Fixes crashing of app when selecting an empty Product ID

Open Pull Requests

1. PR #2034: Fingerprint Authentication in the Passcode Activity
2. PR #1929: Fixes background color of Passcode with the theme
3. PR #1912: Fixes Scrolling in landscape mode
4. PR #1908: Fixes duplication in Material Auto Complete Text View
5. PR #2049: Allows User to enter the correct passcode even after three unsuccessful tries
6. PR #1938: Offline Support in the home fragment
7. PR #2002: Fixed the app to remember language set by the user before logging out

- I intend to continue contributing to the codebase even after submitting my proposal and expect that there may be changes made to the Pull Requests that I have opened. Therefore, I am providing the links to those PRs here

Issues Reported

- I had opened a total of **15 issues** that mainly focused on bugs that were present in the codebase and also on the features that were absent from the mifos mobile at the time . Some of them are still open and have PRs either by me or from my fellow contributors
- All of my open and closed issues can be accessed from here

4. Week Wise Breakdown

4.1 Community Bonding Period (1 May - 26 May)

Week 1

- Get in touch with the developers and the mentor
- Introduction to the community, to the mentor and fix timings to communicate
- Discuss any suggestions and changes to the project. There could modifications, new additions or amendments; it would be better to go over these early

Week 2

- Go Through Mifos-Mobile codebase
- Take reference from android client and try to implement features that are currently absent in mifos-mobile
- Go through the Open Banking API and related documentations

Week 3

- Discuss the working of the existing application with the mentor. Discuss the implementations of the new features
- Go over the new design in detail and ask for changes and suggestions
- Discuss the changes that need to take place in the frontend to accommodate for the breaking changes

4.2 Phase 1 (27 May - 12 July)

Week 4

- I will start off gradually with the migration from MVP to MVVM architectural style
- The order that I will follow is :
    - Splash Activity
    - Login Activity
    - Registration Fragment
    - PassCode Activity
    - HomeOld Fragment
    - Home Fragment and so on
- This order is somewhat sequential and will be one of the first steps to be undertaken
- I would also take inputs from my mentors and if the need arises, change the order of the migration

Week 5

- I would dedicate this week to the implementation of the Navigation graph
- I would start off by grouping the fragments that are connected in a sequential flow and map out a graph for the same in the xml
- This would require the addition of FragmentContainerView in certain activities which will allow further FragmentTransaction operations on the FragmentContainerView and provide a consistent timing for lifecycle events.

Week 6

- This week would be reserved for the migration from dagger to hilt & also for the integration of coroutines in the project
- Considering the fact that we need to write unit tests as well, Hilt will make it easier to write unit tests for the code by allowing us to easily swap out dependencies with mock objects
- Hence it will be better to get on with this at the earliest so that we can lay foundation for a better unit test phase

Week 7

- In this, week I will replace existing api layer from self service fineract to Open Banking
- We can take reference from the Open Banking App while implementing it in our project
- I will be communicating with the mentors and taking their valuable inputs especially when dealing with the endpoints that we have to hit

Week 8 and Rest of Phase 1

- *Buffer* for any pending tasks
- Prepare a report for evaluation
- Discuss brief plan for Phase 2 with the mentors

4.3 Phase 2 (12 July - 26 Aug)

Week 10

- Rocket chat will provide users with the support and facility of fast solutions to their queries. Rocket chat will provide users to communicate securely in real-time.

- Shivangi singh had already updated the callbacks in the library and had further added lifecycle to decline requests . The library can be found here
- Now all is left to be done is the integration into the project and I will integrate this once I get the updated *APIs* during the GSoC period

Week 11

- I will dedicate this week for the integration of mojaloop via the payment hub. Currently Mobile wallet already has this feature and the same can be taken as a reference while implementing it in mifos mobile
- I will consult with my mentor the API endpoints that needs to be hit for transactions, registrations, identification and so on

Week 12

- I shall be writing unit tests for the data layer in this week of GSoC
- Uptill here we would have migrated a certain portion of our project to MVVM and hence the tests can now be written keeping the new architecture in mind
- I shall be using Mockito framework to write the tests

Week 13

- This week will be dedicated to performing the UI tests using espresso whose dependencies are already present in the gradle
- We will have decent amount of changes to the UI uptill here and hence tests that cover all the screens must be put in place
- This will ensure that the views are displayed correctly and look the way they're supposed to.Besides it will ensure that the interactions are handled correctly.

Week 14

- After writing all the unit tests I will add appropriate jobs in the existing workflow to run these tests
- Running tests regularly and automating the testing process can save a lot of time and prevent issues from arising in the codebase

Week 15 and Rest of Phase 2

- If the mentors permit then I would like to implement my own ideas. Considering the ideas aren't that cumbersome it will work out even if its accomplished at the end of the project
- *Buffer* to complete any remaining tasks
- Prepare report for evaluation

   Priority Chart

4.4 Post Phase 2 (After Aug 26)

- Discuss the project's outcome with my mentors and devise a plan of action for future contributions
- Engage with members of the community to solicit feedback on project implementation and to identify possible add-on features

5. Why am I the right person ?

I have been doing Android Application Development for more than one year now. I have gained proficiency in it by doing multiple Internships, several Open Source Contributions as well as Hackathons.I am quite conversant with Android Architectural Components, MVVM, activities, fragments, support libraries, version control, Networking Services, Firebase, UI Development, Jetpack Compose etc.

6. Current area of study

I am pre final year student pursuing **Information Science and Engineering** at Dayananda Sagar College of Engineering.

- Over the course my area of study have included :
    - Learning the fundamentals of Java, Python and C++ along with Data Structures and Computer Architecture
    - Collecting, Storing and Analyzing data using tools like SQL and Python
    - Learning about the fundamentals of ML that included supervised and unsupervised learning,neural networks and deep learning
    - Learning about distributed systems, cloud computing, and virtualization technologies.

7. Contact Information

Name: Pratyush Singh

Email: aries.pratyush@gmail.com

LinkedIn: https://www.linkedin.com/in/Pratyush-Singh/

GitHub: https://github.com/PratyushSingh07

Gitter Id: Pratyush Singh

Mobile Number: +91 9693565684

Time Zone: Indian Standard Time (GMT+5:30)

8. Career Goals

As an android developer my first and foremost goal is to master the Android SDK,Multithreading and other related technologies and create a portfolio of projects that would include personal projects, open source contributions, or projects done as part of my studies or work. As I gain more experience , I would want to specialize in UI/UX design along with enterprise app development and seek out for leadership roles. I would sooner or later venture into AOSP and get a grasp of low level android as well and transition into a full stack mobile developer. I am self taught like many other developers out there and Open Source Projects have played an integral part in my growth and thus I would give back to this android community by sharing my knowledge and expertise through blog posts, tutorials,speaking engagements and of course by contributing to Open Source Projects.

9. My Projects

1. CardSwipeLibrary: *Source Code*

- Streamlined the development process for mobile applications by providing a ready-to-use library, reducing the time and effort required for implementing complex swipe gestures from scratch
- Empowered developers to create dynamic and interactive UI elements effortlessly, fostering innovation and creativity in mobile app design and development
- This Library was built entirely in compose and is also the very first library that I published

2. Cyclofit: *Source Code*

- Cyclofit is a safety and health monitoring system for cyclists
- It tracks the heart rate,calories burnt, distance covered and much more using the sensors integrated in a single device.Our device has a proximity sensor that can track any incoming vehicle by monitoring it's rate of change of speed. The data such as the heart rate are displayed into the app through an api
- Worked with firestore , firebase , Coroutines created a community section in this app that implements a real time database
- Visualized data in form of graphs using MPAndroidChart

3. NewsLive: *Source Code*

- Developed an Android application using the NewsAPI.org API, Retrofit, MVVM architecture, Room database, and Coroutines.
- Implemented background tasks and asynchronous operations using Coroutines, ensuring smooth app performance and user engagement.
- Integrated Room database to provide offline caching and storage of user preferences and data, reducing server requests and improving app speed and reliability
- Utilized MVVM architecture to separate concerns and increase code maintainability, as well as Retrofit to easily connect to the NewsAPI.org API and retrieve real-time news data.

10. Gitter Channel

Yes, I have visited all of mifos's gitter channel and my id was PratyushSingh07

11. Other Open Source Contributions

I have been contributing to Open Source for quite some time and here are some of my contributions:

1. Dare2Change :
    - It's an all in one android application to enhance your productivity and I got to work on this as a part of SLoP 2022
    - Revamped the complete UI of the application and introduced a dark mode theme
    - My contributions for Dare2Change: check here
2. Anki-Android :
    - There were certain @KotlinCleanup annotations in the codebase with messages such as *make data not null* and *simplify through scope functions*
    - I opened a few pull requests adhering to above mentioned messages
    - My contributions to Anki-Android : check here
3. Catroid :
    - It is written predominantly in JAVA and hence refactoring the existing codebase to Kotlin is a vital task
    - I refactored certain files to kotlin
    - My contributions for Catroid: check here

12. Experience with Angular/Java/Spring/Hibernate/MySQL/Android

Yes, I do have experience with Android and Java and have built projects centered around them. I have decent knowledge of MySQL and SQLite Databases.I have built a full stack application during the course of my internship by using Angular as the frontend and Spring Boot for developing RESTful APIs.

13. Other Commitments

I am fully committed to enhancing the Mobile Wallet platform during the upcoming summer as I do not have any other conflicting commitments.

14. What motivates me to work with Mifos for GSoC

Mifos Initiative is making a significant difference in the world by providing financial inclusion to people who would otherwise be excluded from the formal financial system. This mission is truly inspiring, and being a part of it through the Google Summer of Code program is a privilege. From a professional point of view mifos has a vibrant community of developers, volunteers, and supporters who are passionate about the mission of the organization. It's motivating to be a part of a community that is so dedicated to making a positive impact on the world.Besides Google Summer of Code is a fantastic opportunity for students to gain real-world experience working on open-source projects. Mifos Initiative's participation in this program shows that they are committed to helping the next generation of developers like me succeed and contribute to something that would impact lives of billions around the globe.The work that we as a students do during the Google Summer of Code program can have a lasting impact on the Mifos Initiative and the people it serves. Knowing that the work you do can make a real difference in people's lives is incredibly rewarding.

Mifos Initiative's commitment to open-source software is essential for the sustainability of the financial inclusion ecosystem. By making their software freely available, they are enabling other organizations to provide financial services to underserved communities.

15. Previous Participation in GSoC

Yes, I had participated in Google Summer of Code 2023 with this very organization and I would love to be a part this year as well

16. Application to multiple orgs

I will be applying only to Mobile Wallet for this years' GSoC