Software Engineering Assignment 3
CS-400-01
Software Design and Development
Fall 2021

Developing a Software Architecture

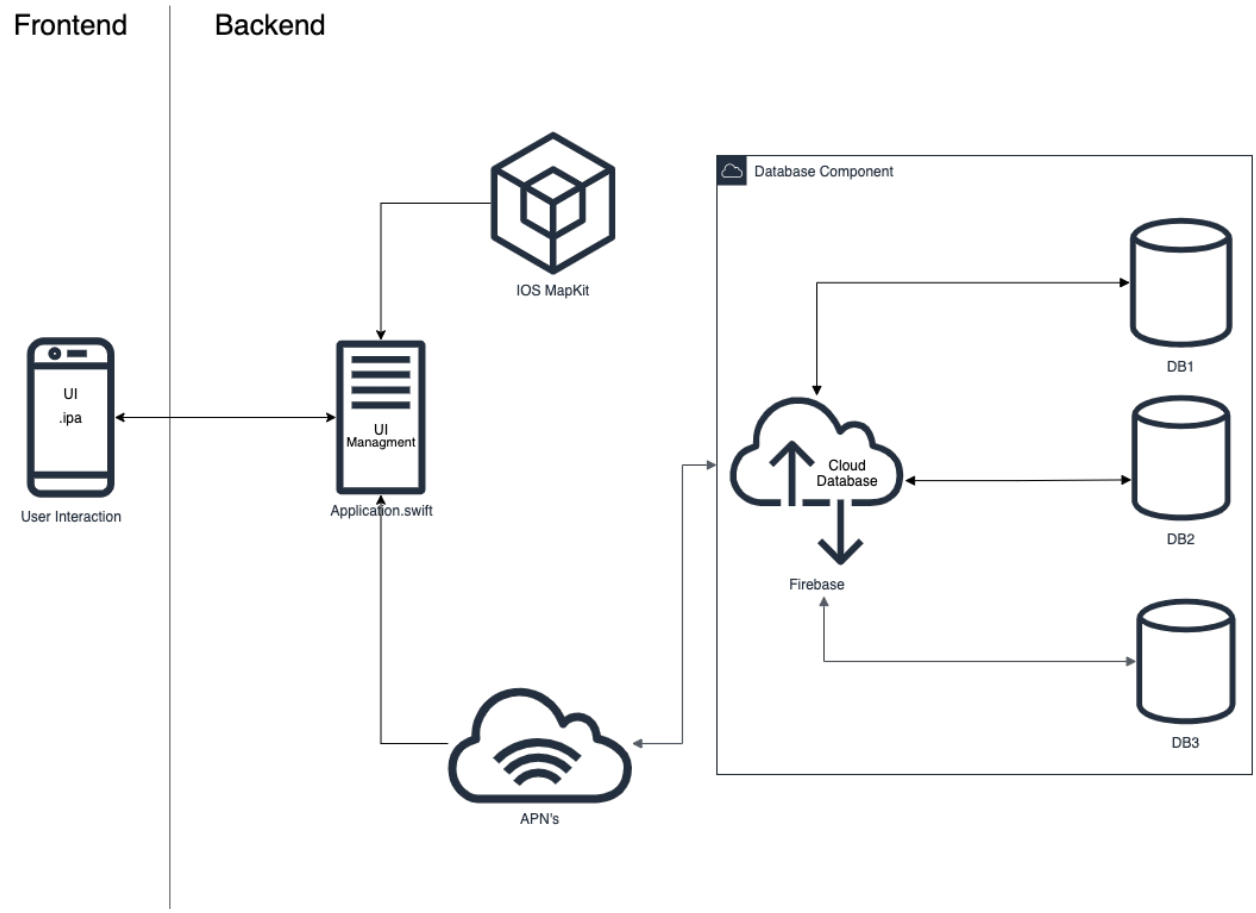
Introduction

In software development, there are many ways of designing architecture. The system architecture will affect many aspects of the software, including responsiveness, reliability, availability, security, usability, maintainability, and resilience. However, some of these aspects oppose each other, such as usability and security. A secure application implements security layers, which lowers the ease of usability, a user must pass through multiple layers of authentications to be able to interact with the application. On the other hand, a more accessible application, would come at the cost of security, yet easily susceptible to security exploits. These types of trade-offs are an important part of software development as the implementation of balanced applications requires the implementation of architecture, which is stable, quick, and secure.

As a platform Kaffi will consist of an easy-to-use UI, which means optimized and smooth usability for all average users. Kaffi features are developed from potential users and persons who are on the lookout for great coffee shops yet prioritize quick and responsive applications, that like Google Maps. However, security must also be a priority for those who own businesses, so that their pages and data would not be suitable for security vulnerabilities. With the future implementation of additional features such as forums, data may need to be secured as well to prevent misuse of the user-to-user commination.

Part I: **Organization**

How should the system be organized as a set of architectural components, where each of these components provides a subset of the overall system functionality?



User Interface Component
User Interface Component is the frontend interface of the system architecture. It would be in the form of an iOS application which would be available for download from the Appstore. This is where all user interaction would happen, like login, map-manipulating, searching, and viewing business pages and the user's own profile.

User Interface Management Component
This would be the connecting component between the UI component and the rest of the backend architecture of the software. It would authenticate user login, apply the searches through the database component and import the map libraries to display for the user.

Map Component
This is the Apple Development kit library/libraries that would be imported to the User Interface management and then displayed in the UI. One of the packages which Kaffi will utilize is the iOS MapKit library, which would have access to iOS native maps to implement the map feature.

Database Component
The Database Component in this architectural design is developed with Firebase which is Google cloud-based database. The alternatives which are to be reviewed are MySQL and SQLite. In this architectural design, Firebase would work as the database manager, while housing multiple databases for a particular backend server needs and data storage. It would then contain the databases, as the best architectural design is unclear as of this moment Firebase provides a service to store the user data with iOS friendly interplay. This database will house business user sensitive information, general data, and eventually the forum's data. There are few routes for incorporating data, one is to generate a separate database or generate a relational database containing the different data in one database. Mapping the business page to the user data through a foreign key for example. Such a relational approach applies the same in Firebase, however with different implementation. These approaches to database component architecture are in consideration as the development progresses. During the development stage, the goal is to incorporate the database which will be the most functional with the Apple Push Notification Service APN's. A multi-tenant implementation is favorited rather than a multi-instance setup for Kaffi database.

Part II: **Architectural Components**

How should these architectural components be distributed and communicate with each other?

Each architectural component would communicate with each other through the main application program in swift, for example, the database will communicate with the UI management component through APN's. Multiple iOS libraries will be imported such as the earlier mentioned MapKit, another iOS kit that will likely be utilized is CoreLocations which allows retrieving the geographical location of the device. However, to maintain a maintain viable product only essential functionalities will be delivered with this prototype.

The MapKit would be imported as a library into the main application (UI management). The UI management component would work as the middleman between the UI and the backend stuff, like the databases and the maps lib, so all communications would go through that component.

Some aspects of the communication part and the exact way that it will be implemented, we are not sure yet, but the MapKit and firebase part we are fairly certain.

Part III: **Component Services**

What technologies should you use in building the system and what components should be reused?

Firebase:
As mentioned earlier, we would use firebase as a service for our database. It's a cloud service through google, so everything would be stored online. Here we would store user information, user preferences, business page information and forums information. It triggers backend code

that runs in a secure environment with cloud functions, and it sends notifications through cloud messaging.

iOS MapKit:
This is the maps library that we would reuse which will allow us to implement our map feature. We would implement our pins, that indicate a business location through this feature as well.

iPhone GPS tracker:
As this is being designed as an iOS app, the user will have an iPhone/iPad/iPod touch which means that it would have an integrated GPS tracker, we would use this technology to implement live tracking, together with the map library, we would have fully functional, live map.