Final Report

Development of an Android App for a Local Startup (Fygo)

Group Members – DP34

Arielle Lasry - 260806019 - arielle.lasry@mail.mcgill.ca - ECSE 458D1
Julien Phillips - 260804197 - julien.phillips@mail.mcgill.ca - ECSE 458D1

Project Supervisors

External supervisor: Andrew Mackarous - Fygo CTO - Andrew@fygo.app

Internal supervisor: Professor Peter Caines - Department of Electrical and Computer Engineering - peterc@cim.mcgill.ca

Abstract

The visual aspect of food is more important than ever in the digital age. Users repost, share with friends, and post their own images of food online constantly. The main goal in creating Fygo's Android application is to give users the ability to discover restaurants and browse their menus with high quality images. In partnership with Fygo, restaurants will be able to upload their menus to the application where users will then be able to browse their complete menu and discover popular dishes. Restaurants will also be able to pay for features advertisements that appear on the homepage of the application. Over the course of the year, we established and completed all the requirements set out at the beginning of the semester. In order to achieve the creation of this application, we began by learning the Kotlin programming language while simultaneously learning the essentials of Android application architectural design. Learning Kotlin and seeing code samples helped us understand the practical implementation of the architectural pattern used. This learning was done mostly by reading the relevant documentation, watching tutorials, as well as consulting our external supervisor, who has years of experience in mobile app development. After extensive research, we devised a suitable architecture design, way to represent the data and completed the main pages of the user interface as well as its required functionalities.

Table of Contents

List of Abbreviations

MVVM – Model-View-ViewModel
UI – User Interface
SRS – Systems Requirement Specification
AVD – Android Virtual Device

Introduction

The objective of this project is to design and develop an Android application. Fygo's goal is to improve the restaurant and food industry by creating an app which allows a user to browse a restaurants' menu using photos of the menu items. In addition, it would allow users to discover new restaurants and meals based on location and via search results for food items and categories. This application aims to tackle the common issue of ordering from or arriving at a restaurant and being unsure of what to order. The Fygo Android application will allow a user to view a restaurants' entire menu in a visual way, making it easier to choose what to order as well as providing a more pleasant menu browsing experience. Overall, it aims to impact a large number of people as it will improve user experience, provide better ways for restaurants to display their offerings to customers, and most importantly help promote local restaurants. The expected outcome of the application which we decided on at the start of the project was to be a fully functioning Android application with two pages; the homepage, where a user can discover restaurants and browse their menus, as well as a search food item page, where users can search for food items.

Background

Before embarking on this project, there was a great deal of theory and concepts we had to further research and learn. We had to brush up on our software architecture design patterns as we had to decide which one to use in building our application since that is the very first step. We decided to go with the MVVM, which was recommended to us by Andrew and seems to be the prevailing best practice for Android architecture design. This architecture facilitates the separation of the development of the user interface from the development of the back-end logic, while also allowing for the views to update dynamically with any change in the backend data. In addition to this, Andrew recommended the development be done using Android Jetpack. Android Jetpack is a suite of libraries that help developers follow best practices, reduce boilerplate code, and allow for written code that works across Android versions and devices. Android Jetpack was new to us, therefore also required research and reading documentation on the necessary libraries we'd need to use. Furthermore, we decided to develop the application in Kotlin, a relatively new language that has gained a lot of popularity in Android application development. This language was new to the both of us, however, given its similarities to Java and our familiarity with a variety of other programming languages, learning Kotlin was not too challenging. Since Fygo's iOS counterpart is already using Firebase as the remote server, and we need to use the same remote data source, we had to learn about Firebase and how it works. Firebase is a platform developed by Google for creating mobile and web applications. It functions as a server, API and datastore. Since Firebase is a Google technology, just like Android is, integration between both was facilitated greatly. In the latter half of this project, extensive research was done on the practical implementation of Views and View Models since the frontend was the primary focus.

Requirements

Fygo's Android application aims to solve the problem of not having any way to see your food before you order it. This application serves as a medium for a consumer to discover new restaurants, especially those in the local area, as well as browse a restaurant's menu in an organized fashion with high definition images of every item on the menu. By providing an application that does this, we hope that consumers will have a better way to browse and discover menus from their favourite restaurants and restaurants they've never been to.

The project requirements include research, design and development. Research is a necessary task to ensure we move along successfully while understanding what we are doing. The design aspect is crucial

as it sets the tone for the development phase and gives us the build plans for the application. The development stage is where the practical implementation of the design takes place. At the start of the year, we outlined a minimum SRS with Fygo as a plan for the features which took the most priority and the features which could be omitted should time not permit their completion. The minimum SRS established is to have a fully functioning home page, which contains three horizontally scrolling card views: one for advertisements, one for local restaurants and one for popular dishes. Clicking on any of these cards brings you to the restaurants page in the application, where the user can view all menu items as well a receive information about that restaurant (hours, location, type, etc.). This homepage and the corresponding implementation of restaurant pages with their menus is the primary focus of this project. The secondary focus is to create the search page and corresponding search functionality, with search functionality being the last thing to be implemented if time permits. This minimum SRS outlines the work which should be completed by the end of the project.

The major constraint we believed to have with this project was time. Given that our schedules are busy, and we have other work to complete in the same time frame, finding time to complete this project the way we'd like to can be challenging. The focus was of course to implement the desired functionality, but we also hoped to do so with a compelling user interface that matches the design template of the iOS counterpart while also creating an application that feels native to Android. We were pleased to discover that we indeed had enough time to create and refine the user interface that closely matches the iOS counterpart. Throughout the year, we made sure to complete as much as possible as quickly as we could without sacrificing the quality of code produced.

Design and Results

Backend

The first steps before beginning any development were correctly modeling the data and devising a suitable application architecture. Modelling the data in this application was relatively easy, since the only two entities to model were restaurants and menu items. We were able to quickly come up with data models that closely resembled the final models, but as we went through the development of the application, we did need to add some data fields and change the data type for existing fields as well. The final data models can be seen in Figure 1 below.

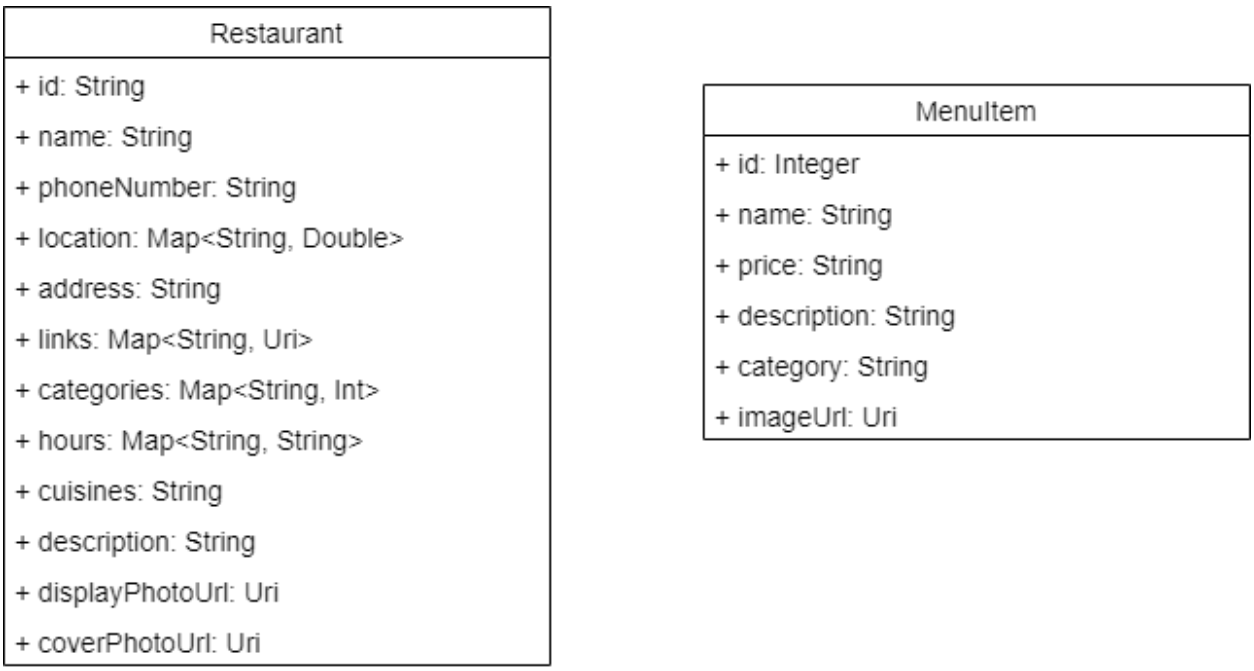

Figure 1: Restaurant and Menu Item Class Fields

Deciding which route to take architecturally was made simple for us given that MVVM is regarded as the best practice for Android applications and given that Andrew suggested we use this architectural pattern. Understanding this architectural pattern and its benefits however wasn't quite as simple. MVVM allows the frontend of the application to be separated from the backend. In addition to this, each View relies on its corresponding ViewModel to get data, and all data being shown in a View is simply observed from the ViewModel. This allows the data in the application to update automatically when any change occurs in the backend. The initial architecture design is shown in Figure 2 below and implements the MVVM pattern. In this design the repository acts as the single source of truth in the application and selects whether to obtain data from the local database or the server. This ends up being convenient from the frontend perspective as there is only a single data source to deal with.

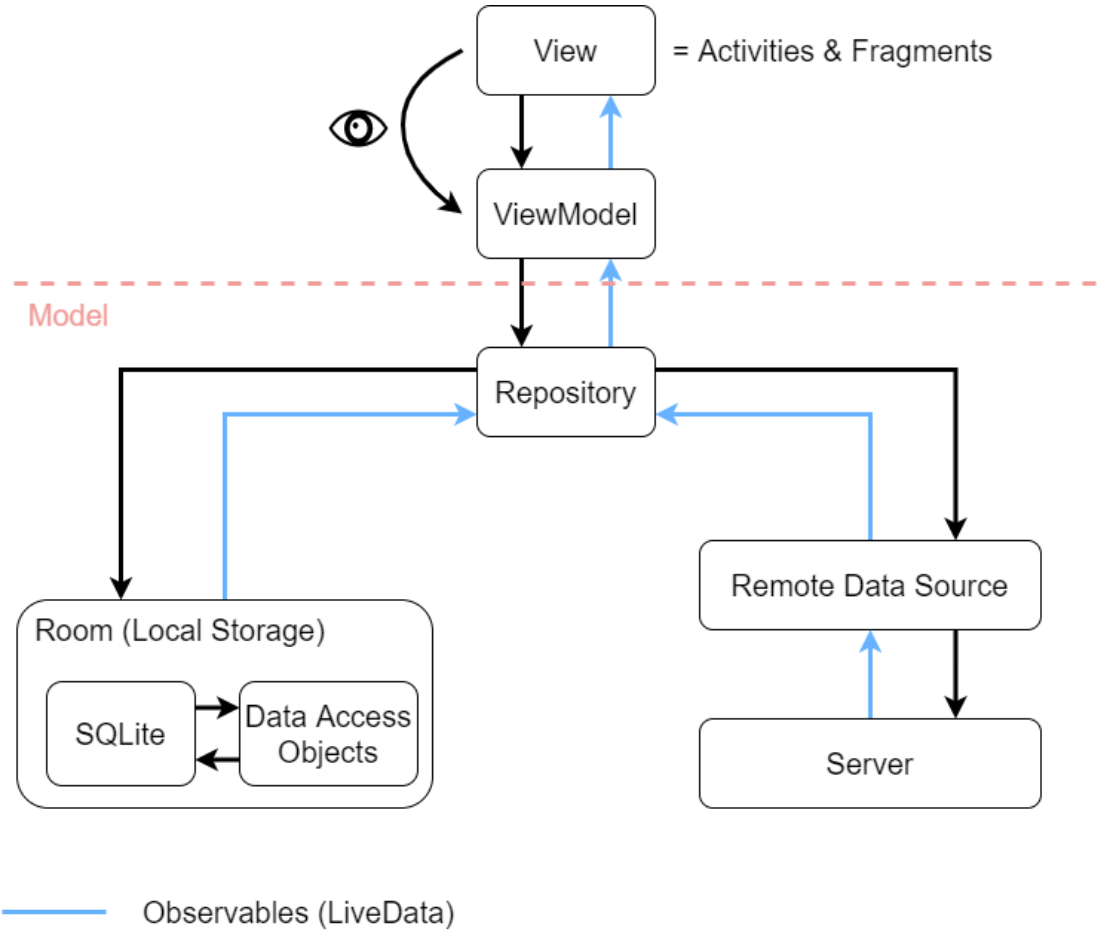

Figure 2: Architectural Diagram

When integrating the remote data source into the code base, we came to the realization that Firebase, the remote data source, had caching enabled by default. This meant that the work we had completed for the local storage part of the backend was no longer necessary. In response to this, we decided to revise the application's architecture as we believed it would simplify development of the application. Doing so allowed us to avoid having to create a repository that needed to choose from the local or remote data source but brought some of its own challenges as well. The revised application architecture can be seen below in Figure 3.

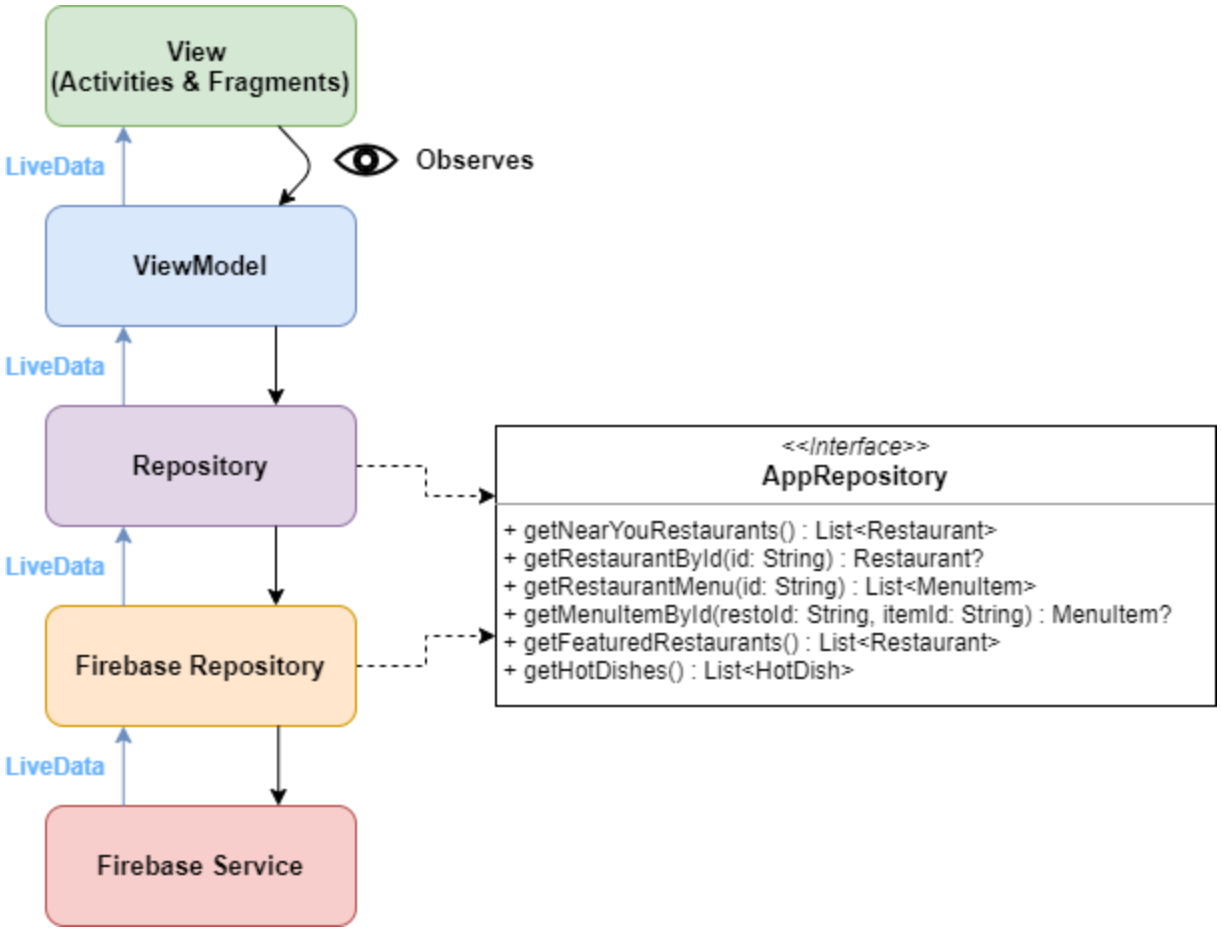

Figure 3: Revised Application Architecture

The revised application architecture is very similar to the old architecture, except without a dedicated part for the local storage, since it is built in automatically. The interaction from the frontend to the backend remains identical in that Views continue to observe data contained in their corresponding ViewModel. In addition, ViewModels get their data via the Repository class, the single source of data from the frontend's perspective. Since the top-level Repository class is used within the ViewModels to make function calls and receive data, we wanted the make the top-level Repository as general and non data source specific as possible. Making the top-level repository non data source specific was a suggestion given to us by Andrew, who told us to consider the fact that if Fygo were ever to change their databases from Firebase to another location, that the repository referenced within the ViewModels would need to be changed. Since we want to keep the backend and frontend independent of each other, we decided to have two levels of repository. The top-level repository is the one referenced within all the ViewModels, and simply calls its equivalent function in the lower-level repository. The lower-level repository is one that is linked to a specific data source (in our case Firebase). In order to enforce that the top-level repository can call an equivalent function in the lower-level repositories, we made the repositories implement a common interface, which dictates the functions that must be present in both. What makes this dual layer repository architecture convenient is that should Fygo ever move away from Firebase, all that would be required would be to implement a new lower-level repository specific to the new data source, and then change the top-level repository to call the equivalent function in the new lower-level repository. This also allows for a mixed data source backend, where the top-level repository can call functions from as many different

lower-level repositories as there are, pulling data from many different places without needing to make any frontend changes.

Frontend

The frontend part of this application was well defined because the minimum SRS was defined in terms of what a user should be able to do, which directly references the user interfaces necessary. In addition we were also able to base ourselves off the iOS version of Fygo's application. We have a "Hot" page, which functions as the app's home page, and a search page, both of which are accessed via a bottom navigation bar. The home page has three horizontally scrolling views on vertically stacked. Each horizontally scrolling view contain cards that are clickable, and lead to a corresponding restaurant page when clicked. The restaurant page displays information about the restaurant, as well as its menu in another horizontally scrolling view. Since the frontend was heavily dependent on multiple horizontally scrolling views containing cards that display some data, this part of the code was reused many times and adapted in small ways to suit the needs of that view. The built-in android RecyclerView element was used to achieve these horizontally scrolling views. Figure 4 below shows the main elements involved in getting a RecyclerView working in Android.

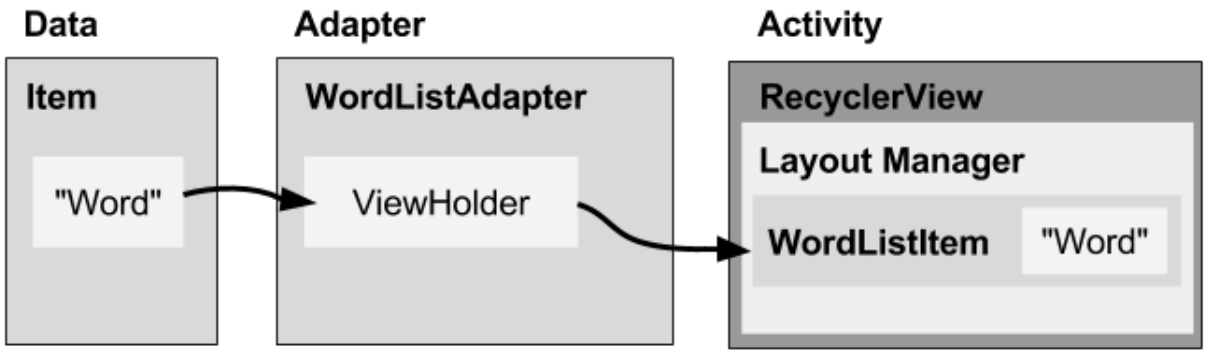

Figure 4: Functioning of RecyclerView, LayoutManager, Adapter, ViewHolder and Data

Each RecyclerView needs a corresponding LayoutManager and Adapter. Data is passed into the Adapter, and the Adapter deals with how to map the data objects to their corresponding view. That view is then displayed in the RecyclerView. When the user scrolls and elements "move off" the screen, what really happens is the same views are reused but repopulated with new data in the Adapter. This makes sure that the application does not overuse resources generating an individual view for each piece of data, but instead repopulates existing views with new data when it needs to.

Since many files were used to compose the frontend and there is quite of bit of nesting of Fragments, Figure 5 below shows its basic architecture, along with which ViewModels and Adapters each Fragment uses if they use one.

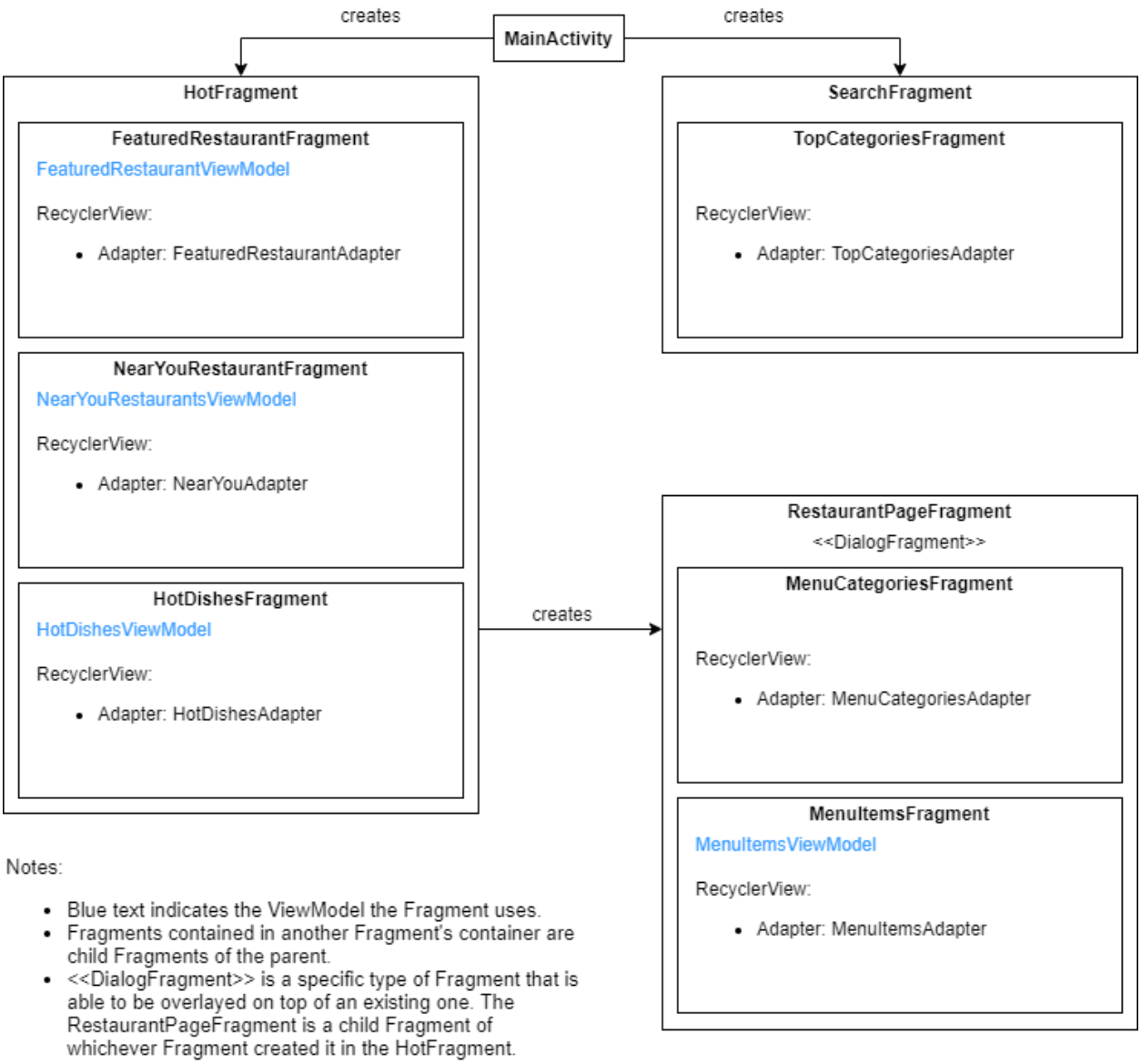

Figure 5: Frontend Architecture

Images of the final user interface of the application alongside the blueprints created at the start are shown in the Appendix. Note that the final product and the initial blueprints do not correspond exactly, as some changes were made along the way to ensure we had time to deliver on all the functionality of the application.

Further Improvements

After having completed this project, there are a select few things we would have liked to implement given the extra time. We think it would have been interesting for a click on the restaurant's location lead to their location in Google Maps, as well as clicking their telephone number lead to a call being placed. We also think it would be possible to further refine the UI by adjusting colours, text size, and spacing on some of

the UI elements. Lastly, given the extra time, we believe it would be helpful to have implemented a robust test suite to make sure that the app's behaviour is as expected.

Engineering Tools

The engineering tools used throughout the project have remained unchanged. Android Studio is still the development environment and there have been no issues with it so far. It is the recommended IDE for development on Android applications. The code is written in Kotlin and Kotlin has recently become one of the official Android development languages, so works very well with Android Studio.

Impact on Society and on the Environment

Use of Non-Renewable Resources

Throughout this project our team has been communicating using emails as well as other online communication means. Although previously unaware of this, these technologies indeed do have a carbon footprint. An email containing an attachment can cause approximately 50g of CO_2 emissions. Therefore, continual correspondences between team members and supervisors can certainly add up [2]. Hence, the greatest environmental impact will be the carbon produced by technology such as emailing, internet usage and sharing and editing documents. The manufacturing, distribution, use by consumers and recycling aspect of our application is not very applicable since it is a software-based project and these stages do not have an environmental effect.

Environmental Benefits

There does not seem to be any direct environmental benefits, however, the lack of use of non-renewable resources is a benefit. Another indirect environmental benefit is that Fygo will support local restaurants, and shopping locally reduces your environmental impact. In general, locally owned businesses are more likely to make local purchases, requiring less transportation of food items [2]. In addition, we hope that promoting local restaurants will help people stay in their local areas instead of going elsewhere to get food, again reducing the use of personal transportation.

Safety and Risk

Although out of our control as we are only building the application, the safety and risk of the application is directly linked to the restaurants that Fygo features. As the application is promoting certain restaurants and dishes, the main point of contention is the food being promoted. The featured restaurants should be hygienic and strictly follow health regulations.

In our opinion, the main risk is that a visual menu creates a temptation for so many things. It leads us to "eat with eyes," possibly exaggerating and making unhealthy decisions. More specifically, this is something called visual hunger. The way in which a food is plated impacts people's flavour perception and can modify people's food choices, specifically their consumption. We know that advertising increases the consumers' wanting for food as well so "visual hunger" can cause those behaviours that are associated with food consumption. Although indulging once in a while causes no harm, this is a risk that should still be considered given the obesity rate in Canada is on the rise with Montreal at 21.5% [3]. However, the application can also decide to promote healthy choices and help fix this potential issue.

Benefits to Society

There are both quality of life benefits as well as economic benefits that this product offers. One benefit of this application is the convenience it provides its users with. It is simplifying decision making and enabling an enhanced user experience when it comes to ordering food. It lets users scroll through dishes and restaurants without having to imagine if the dish is something they would like, rather than knowing it is something they would like. It eliminates the uncertainty and disappointment that often comes when ordering a dish. It allows users to step out of their comfort zone and order something new without being worried about what they will receive.

Additionally, a major benefit to society is the promotion and support of local restaurants. By supporting local businesses, you are supporting your local economy. Significantly more money stays in a community when purchases are made at locally owned businesses, rather than nationally owned ones. Considering the current state of restaurants and the government shut down, our local businesses are struggling immensely, and this application can help put struggling restaurants back on their feet.

Report on Teamwork

Throughout this project we have managed to work together, split tasks between us, and meet virtually on a regular basis. To make efficient use of our time, we assigned work based on our strengths and have also taken on work to unburden the other teammate when their schedule becomes busy. As a team we collaborate very well and have not encountered any major difficulties. Being a small team also makes it much easier to communicate with each other since there is only one other person to update on the progress of tasks.

The following is a general explanation of member specific roles and has remained consistent throughout the entirety of the project.

Arielle Lasry:

Arielle has taken on the primary managerial role while taking on a secondary development role. This entails planning, putting together the reports and presentations, organizing meeting times and making developments in the codebase where assigned.

Julien Phillips:

Julien has taken the lead developer role. He has more experience with app development and software technologies. Julien splits up development tasks between us and conducts the bulk of the code reviews with Andrew. Julien also helps with reports, particularly with the technical documentation.

Conclusion

This project has been a great learning experience. Towards the beginning, we were overwhelmed with the number of things to learn before beginning development. We conducted tons of research both design and implementation related before even starting to make sure we had a complete understanding of the way we would need to approach this project. We also took extra time at the start make sure we had clear requirements, since without establishing proper requirements, architecture and technologies, an application cannot be built successfully. We figured that taking extra time towards the start was justified and would help pave the way for easier time developing the rest of the application. After gaining the

knowledge we needed to begin coding we began with the backend of the application. Needing to change architectures halfway through the project also took plenty of research to make sure the new architecture would be built with best practices in mind. Establishing a good backend made the frontend of the application much simpler to implement. Though the frontend of the application was less complex to understand, it was the much more time-consuming aspect of the development. However, this was alleviated due to lots of code reuse among similar frontend components. The final product is an application that meets all the main requirements set out at the beginning of the project.

References

[1] E. Richards, “The Carbon Cost of an Email,” *carbonliteracy.com*, Feb. 18, 2018. [Online]. Available: https://carbonliteracy.com/the-carbon-cost-of-an-email/. [Accessed Nov. 19, 2020].

[2] C. Gill, “10 reasons why it’s important to shop local,” *boody.com*, Aug. 21, 2019. [Online]. Available: https://www.boody.com.au/blogs/eco/10-reasons-why-its-important-to-shop-local# . [Accessed Nov. 20, 2020].

[3] S. C. Government of Canada, “Health at a Glance,” *Adjusting the scales: Obesity in the Canadian population after correcting for respondent bias*, 2015. [Online]. Available: https://www150.statcan.gc.ca/n1/pub/82-624-x/2014001/article/11922-eng.htm. [Accessed: Nov. 20, 2020].

Appendices

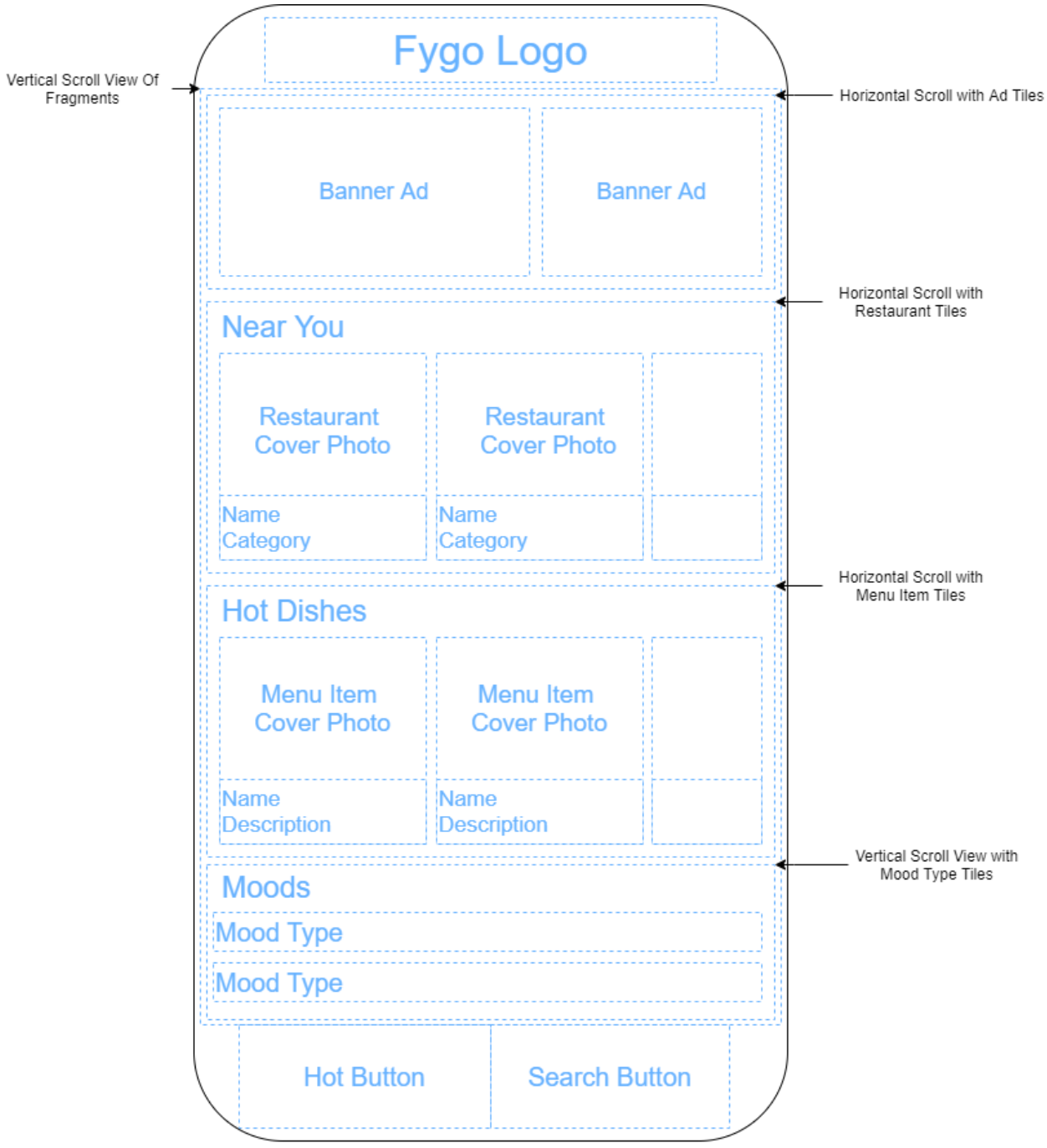

Figure 6: App Home Page Blueprint

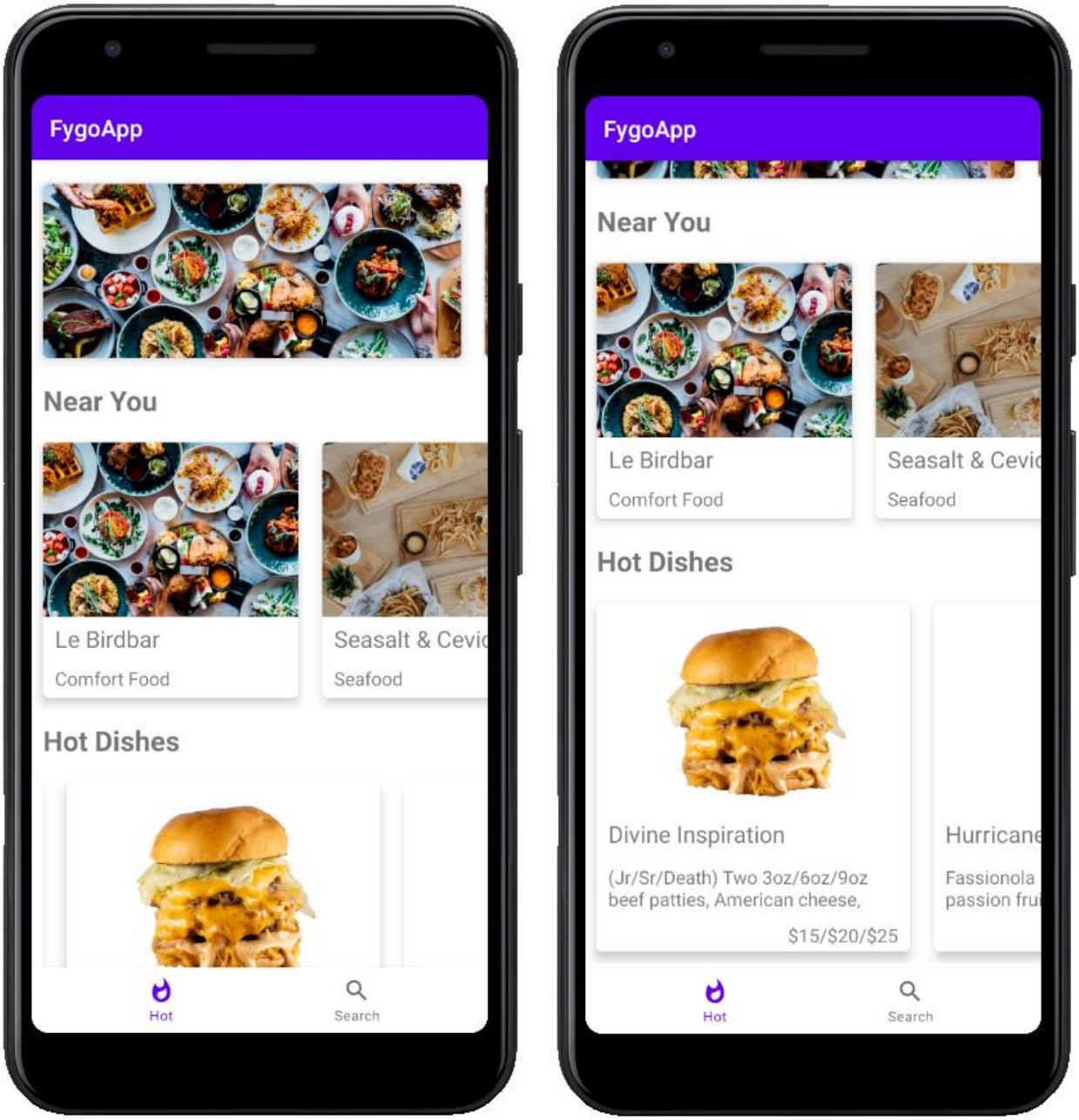

Figure 7: App Home Page Final Product

Figure 8: App Search Page Blueprint

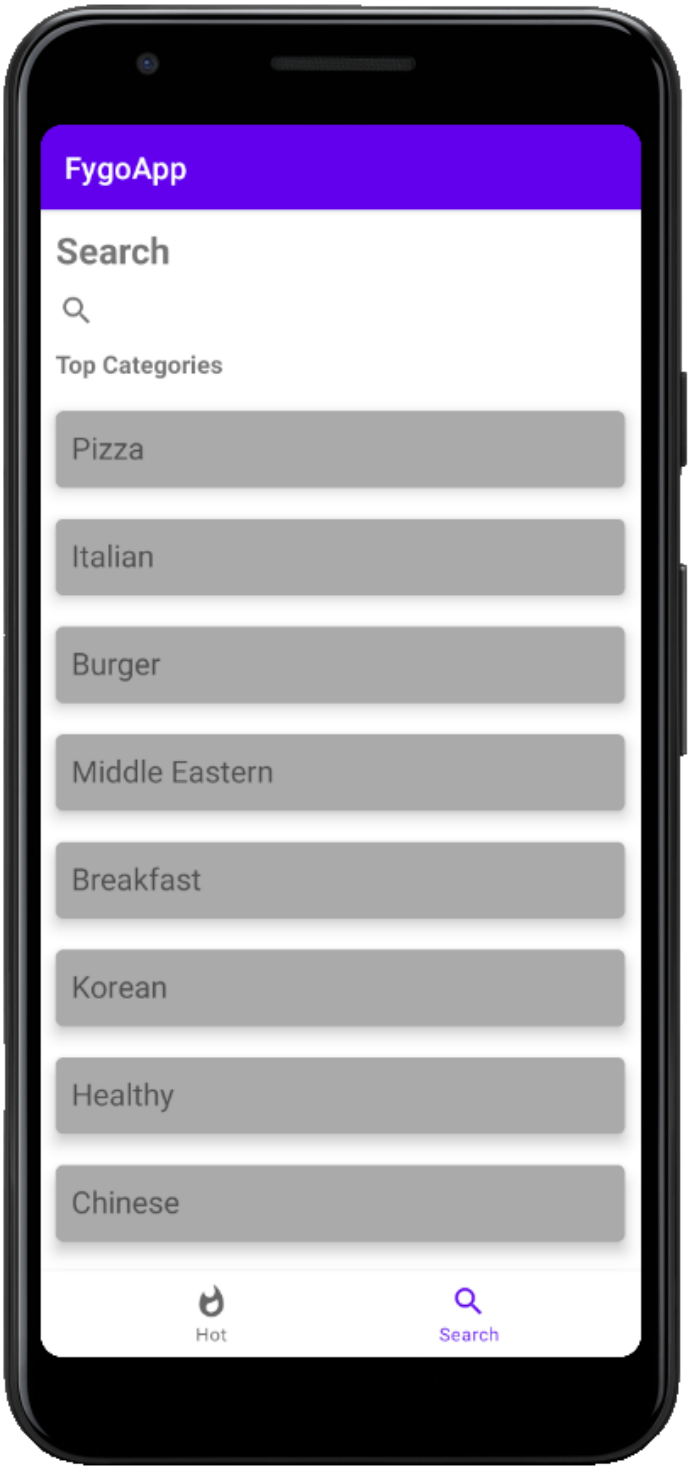

Figure 9: App Search Page Final Product

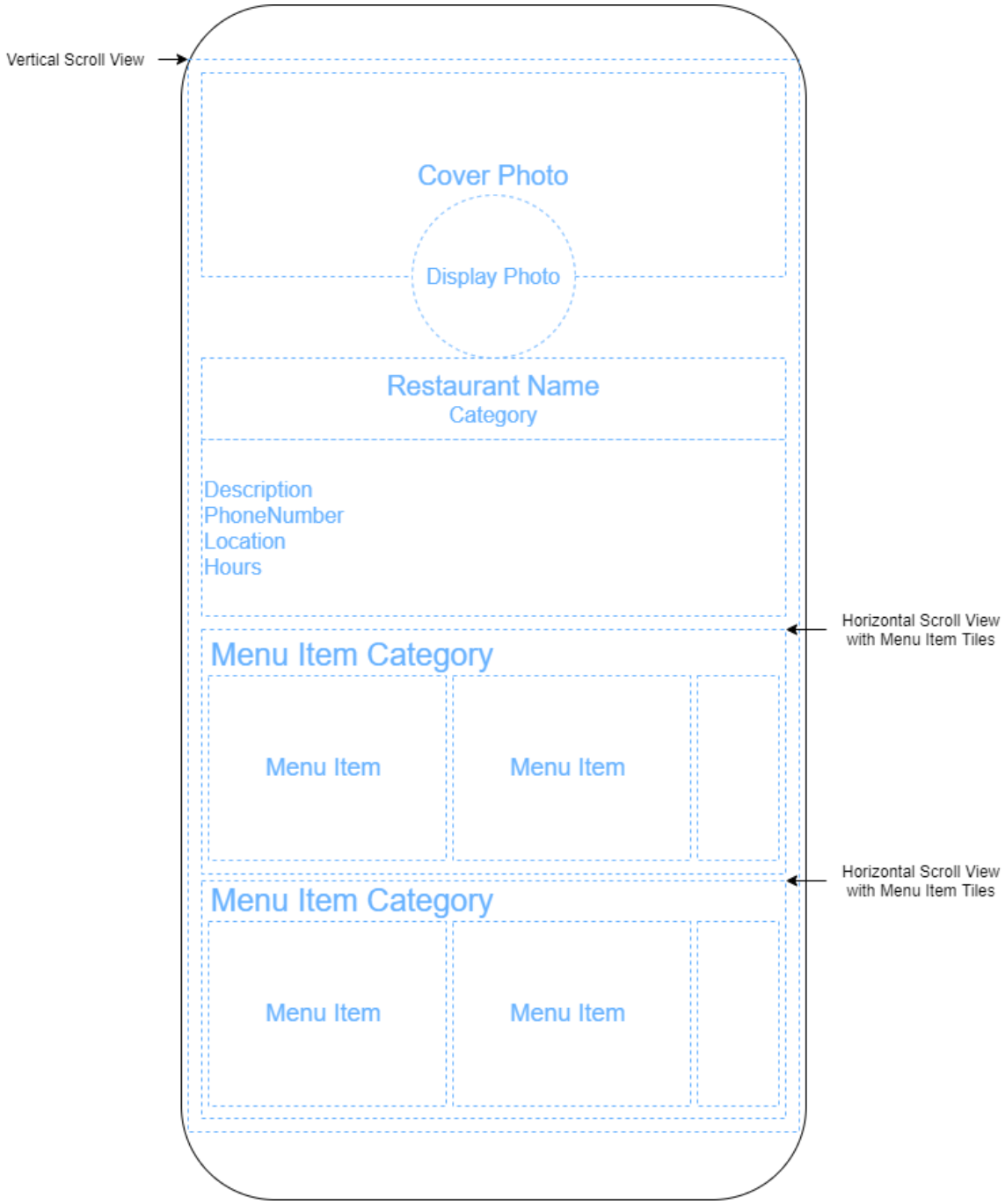

Figure 10: App Restaurant Page Blueprint

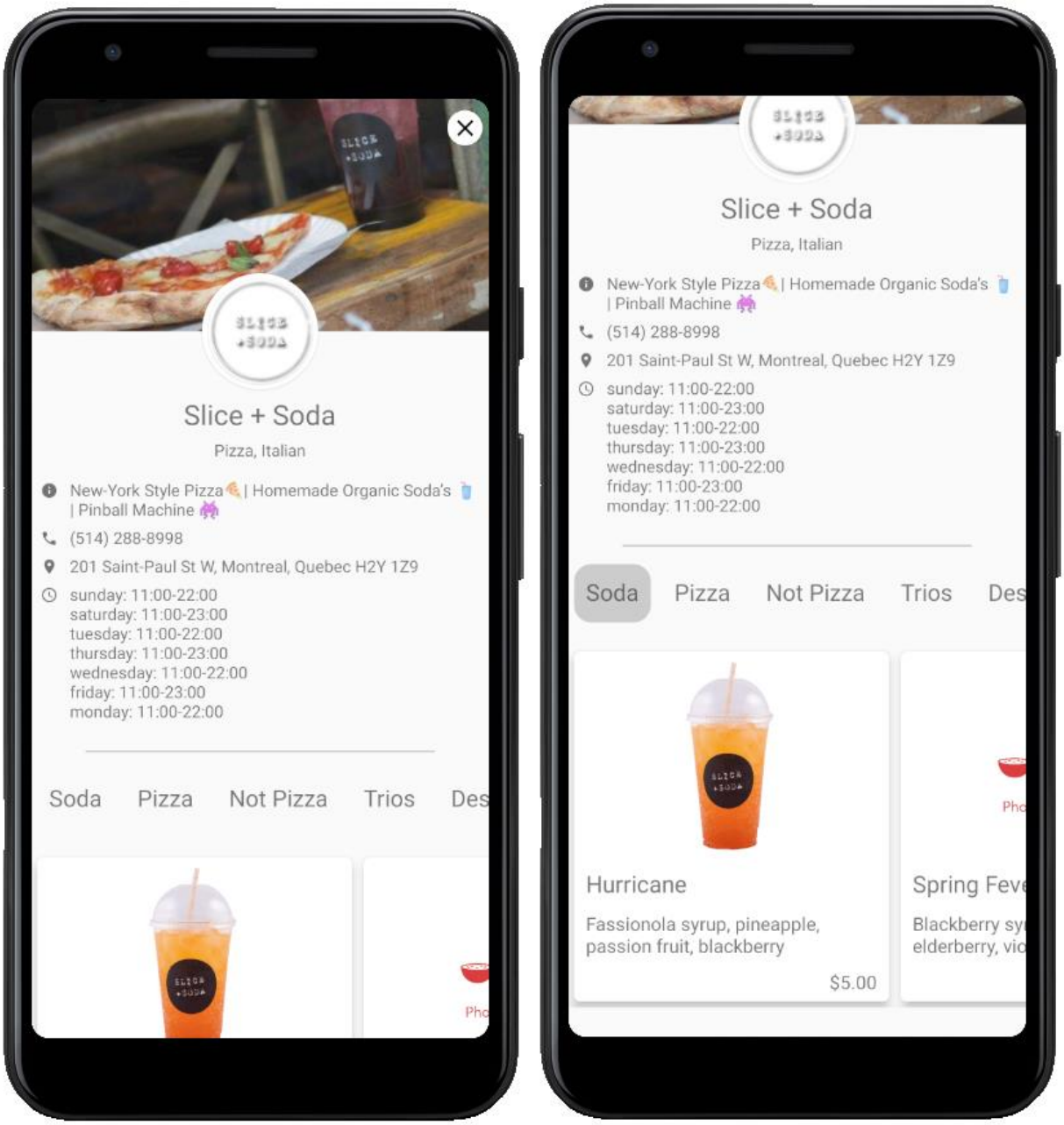

Figure 11: App Restaurant Page Final Product