Home Food Delivery - Coding Report

Group 2 - Soham Patel, Hari Seelam, Ish Soni, Krishna Bavana

Project Description

The Home Food Delivery mobile application will provide a way to deliver fresh homemade food to one's doorstep (like Uber or DoorDash). Cooks will prepare meals from their personal kitchen for customers who desire home-cooked meals, and drivers will deliver the meals to the customer's home.

Project Deliverables: First Release

This first scenario focused on user (customer or cook) account creation and login. Customers will be able to view the catalog of cooks available in their area and their menus. The customer will then be able to select menu items without any regards to purchasing. When the cooks sign up, they will be able to add their availability and menu which will be uploaded into our database.

Project Deliverables: Second Release

The second release was focused on Google Maps API, Weather API, and Ordering System. The ordering page UI contains many details like item name, the quantity of the item, descriptions, prices, the option to increase the quantity and decrease the quantity. All the information on the cooks' page is dynamically set based on the cooks' options and data is fetched from the database. We have used shared preferences to send the data of the customer's choice from the cook's order page to the final order page. Even on the final order page, the customer can change the item quantity and delete the item.

Project Deliverables: Comparison with the Original Group's Report

Our project was inspired by Group 26 of CS 440 Fall 2021 semester. I will be referencing section numbers from their final report throughout this analysis.

We kept their idea intact, as we have three distinct users of the app: customers, cooks, and drivers. All of these users have a similar UI structure when we compare our design to the final block diagram in section 25f of the report. There are slight differences in layout, but the structure is similar. A customer can order from a cook, and the cook will be able to view the order on their account. All of this is reflected on the Scenario Diagram in section 4a. For the database, we agreed with using SQL as suggested. Finally, the cooks are able to use the application offline as mentioned in section 17 since the app does not require Wi-Fi connection.

The major difference regarding the prototype and our project is the ideas for solution in section 35. Group 26 suggested we use Swift and XCode for iOS development and React Native for android development. Instead, we used Dart with the Flutter framework. This supports both iOS and Android, while writing on just one code base. This gives the app a consistent look no matter what device or emulator we use. We all used Visual Studio Code and Android Studio for development.

Testing and Inspection

We tested five main categories of the app: Cook orders page and final page, Google Maps API, Weather API, Login/Sign up Screen, and the Driver mode. Only people that did not develop the

code for that section were allowed to test. Pull requests were created, and only once the code reviewer tests the functionality to the fullest will the code be pushed to the master branch. These functionalities were tested on an Android emulator, IPhone emulator, and a physical device which has the latest version of the app built. The testers followed the inspection procedures that were outlined in the github pull request as well as tested any boundary cases they deemed necessary. At the end of the project, all of these major tests passed, deeming the app to be functional.

Project Retrospective

Having two branches for the git worked well (master and development), but it would have been better if we each had our own branch to work on and then merge to master rather than all of us using the development branch to do our work. Doing that caused problems with some pulls from the repository that would overwrite some of our work so we had to do more work to coordinate the files we were working on and our pull requests to the master branch. We could have put more research into the packages that are available in Flutter so certain aspects of the app would have been easier to develop. For example, we spent time creating a login screen only to find out there is a package that would have streamlined the process. Had we known that we would have spent more time on developing other features.

Waiting Room

An idea that we can implement later is having the cook set hours of operation for their business. We would have the cook set their business hours, and if the current time is within that time, then the customer would be able to view that cook when they are looking at the available cooks.This has a low cost of implantation with high benefits since development of this idea is not difficult for our development team.

Another idea is to have the cook be able to edit menu items. The reason a cook would want to do this is to either change the price, or to edit how the food item is being prepared. This would be a very low cost of implementation since it is one simple button that would run an update statement in the database, with medium benefits as menu items may not be edited often.

Another idea that we could implement later is a popup notification on the customer's device regarding the status of their order. There will be a notification regarding the order being cooked, out for delivery, and being delivered. On the driver side, they can get a notification when there is an order ready to be delivered within one mile of their current location. On the cook side, they can get notifications every time an order is placed to them as well as whenever they get any feedback from the customer. Implementing all of these notifications will have a high cost since we need to incorporate the Google Maps API with the driver notification. This will have a lot of benefit to the users since popup notifications on their device is a very easy way to communicate with the customer.

Finally, we can develop logic so that the customer can track the driver when the food they ordered is out for delivery. If the customer wants to know the status of the delivery, then they can go to the maps tab from the taskbar and view the route of the driver. This is a high cost of implantation since we need to use a GPS API and integrate it with our current Google Maps API.