Dev Team 11 (Shelby Wilson, Dominick Dellecave, Matthew Garvey)

DEV-4 Beta Prototype

Home View Model (Matthew):

Within the project I was responsible for implementing the main page of our application. This is the page where restaurants are displayed to the user, and they have the ability to swipe left or right (or indulged) on them. The key task that this view model was responsible for handling was being able to swipe through restaurants in order to find a place to eat. As far as that task is concerned, it did not change much at all in the development process. If anything, this task was improved upon for accessibility purposes. Originally, the intention was that the user would simply swipe left or right on the given restaurant and/or click on ‘indulged’ if they had already been there. In the coding phase of the project, I decided to both support swiping and button tapping. Therefore, the main page features a button for left (no), right (yes), and indulged (been there), and it supports swiping left, right, and up. The restaurant is displayed essentially exactly as detailed in our wireframes, and they contain all of the information that would be necessary for the user to make their decision. Thus, the primary key task of allowing user to swipe through restaurants to find a new place to eat was accomplished according to plan.

One thing that did change was the information component of the home page. This did not impact the key task, however it is a slight deviation from the wireframes. Originally, the main page was intended to feature a downward arrow that would take a user to more information about the displayed restaurant. However, the wireframes nor the storyboards had any design for what that information page would look like. Once time came to develop the main page, that information tab did not feel necessary from the home page, as it would only serve to slow down the process of finding a place to eat, which would be counter to our app's motivations.

Otherwise, the rest of the main page works as intended. With that being said, some backend implementation can be added to make the homepage work 100% as expected. As it stands the backend serves to move restaurants between the main view model and into the list view model. However, the backend does not currently support truly utilizing the user's preferences, distance, allergies, and cost desire. With that being said, I do intend to flush this out for the sake of our repository, to allow all of us to showcase a complete front and backend application. The existing backend would pretty simply facilitate these improvements to the preference utilization, however my own uncertainty with MVVM rules and regulations has given me pause in handling this. I would plan to have the home page update every time preferences are modified, but that all occurs within the view model for the profile, and I am uncertain if adding in any sort of "updateHome()" in that position would be a red flag for the sake of this course.

One major lesson I learned from this is that when working within a team, if everyone is able to communicate effectively early and often, the project will go by quite seamlessly. Our group was consistent in collaborating and staying ahead of deadlines, and consequently we felt like a well oiled machine, excuse my cliche. When some changes were made to the code, the group would immediately be informed, and if there were any issues that arose they were handled quickly. Another lesson I learned was that separation of tasks, when done properly, makes integrating code much easier, we rarely had merge conflicts or bugs, and as mentioned, those that did happen were handled swiftly. Independently, I was very unfamiliar with MVVM and Flutter coming into this class, and I can confidently say that I was able to grow those skills and understanding to a level where I can be confident with them. Moving on, as I mentioned, I did not proceed with any sort of “more information” area on the home page. This was where I learned that sometimes there is a cost benefit that needs to be done when developing, and sometimes that also works to benefit the user. Although I personally would have liked to see that page manifest, it would have required a large effort for something that was just going to be a cool addition to the app, and not serve to actually complete our key tasks. The user’s ultimate goal was to find somewhere faster than scrolling through Google, and that page would have just moved the google search into our app. This helped me grasp a more nuanced understanding of the cost-benefit analysis of engineering effort, as I was considering bot my personal time and the benefit to the user.

Lists and Reviews (Shelby):

I was responsible for implementing the Lists and Reviews pages within the Indulge app. These pages help contribute to the functionality of key tasks 2, 3, and 4 in our original design proposal:

- #2) Reviewing food places you visit and saving them for you to look back on.
- #3) Maintaining a list of places you have been.
- #4) Maintaining a list of places you want to visit.

Key Task #2 has not changed much since the start of our project, however, there is one key difference from DEV-0. In DEV-0, we mentioned that this task would support the use case: “reviewing food locations to remember how it was and to provide that info to other users”. We quickly realized that sharing reviews between users would add an extra layer of complexity to the app (possibly dealing with roles and permissions), so we decided to make reviews completely personal. Our final Reviews page implementation only shows the user’s own reviews. We changed this because we realized a social media aspect would be too complex for the scope of the course. We wanted to keep personal reviews because our personas are all people who go out to eat a lot. The personal reviews give them quick reminders of what they absolutely loved or hated about a restaurant. This can be anything from vibes, service quality, to actual enjoyment of the food. One of our personas, Jayden, is also a photographer, so the personal reviews section will allow him to jot down information about the location for his business.

Key Task #3 is also almost identical to how it was described in DEV-0, but we added a small change to make the app more functional. When a user clicks the "Indulged" button to show that they have already been to a restaurant, it gets directly added to a "Been There" list on the Lists page. In our application, the "Been There" list is a special list that is always at the top of a user's lists to keep track of places they have already visited. These decisions are all the same from the original design. However, the one difference is that we added "Indulged" buttons within all other lists so users can mark them as "Been There" after visiting. With our original proposal, there would be no way for users to mark that they visited a new restaurant in the app (thus defeating a lot of the app's purpose). With the Indulged buttons in other lists next to restaurants that the user has not been to before, users can easily mark new places as visited and write reviews after they have been there.

Key Task #4 is probably the most changed task from our original description in DEV-0. Rather than having a single "To-Visit" list of places the user wants to visit, we have different lists depending on the cuisine type, pricing, and dietary restrictions of a restaurant. Since the point of our app is to help users find new food locations, we decided that all lists besides the "Been There" would be considered "To-Visit" lists. We also opted to have the restaurants organized in different categories to help speed up the time it would take to find a location in the app. We decided to remove the To-Visit list because it would just add clutter for our users. We think that our personas would prefer to have everything organized as much as possible to fuel efficiency during their busy days. Rather than scrolling through To-Visit, users can quickly find restaurants in a specific category.

The designs of both the Reviews and Lists pages have remained similar to what was included in our wireframes, but there are some changes to discuss. One change is the addition of a + button at the bottom right of the Reviews page to create a new review. In our original wireframes, this was omitted, so there was seemingly no way to create a new review. So, we made sure to add it to the real app. We also originally had no difference between the view used to create a new review vs. the view used to read an already created review. We designed it this way because we wanted reviews to be editable, but we later removed this to avoid too much complexity in the UI. We also figured our personas/users would be writing very quick reviews, so they wouldn't spend too much time editing them to make them perfect. We separated the Create Review view and the Review detail view so that the Create review view has an editable textbox and star rating widget, while the review detail view has read-only versions of these widgets and a delete button.

<u>Lessons Learned Overall:</u> There are some lessons learned sprinkled throughout the previous paragraphs, but I'd like to describe more general lessons I learned here. I learned a lot from this class and project. At the beginning of the semester, I had no experience with Flutter, let alone mobile development. I had also never used the MVVM pattern before and had little experience with declarative programming. However, now that we are at the end, I feel much more comfortable coding in Flutter and the MVVM architecture. Although our setup still might not be perfect, we follow the architecture pattern to the best of our ability. We learned to focus on the separation of concerns in our application, which helped make integration seamless. We also

learned good ways to separate our code in the repository. Once development picked up, we realized we needed separate branches for each of the key features. Having these separate branches and pull requests to merge changes was much easier than working on the same branch. We utilized GitHub issues, milestones, and branches to keep us on track, which proved very helpful in the integration process.

Profile View Model (Dominick):

I was responsible for the user info related pages of our application, which means that I handled all code and functionality to user data, accounts, and onboarding. These features provide data to the rest of our application, allowing for our key features to work and update seamlessly as the user might change his/her preferences. As such, these features should be thought of as "core tasks", which are necessary for the whole of the app to function correctly. Within the application, my front-end/view work consisted of the login screen, onboarding process, and user profile tab. Unsurprisingly, as is common when learning front-end frameworks, the view portion of my work and a significant learning curve. In the beginning of my time working on Indulge, I was often in a manner of trial and error, attempting to figure out the syntax and multitude of packages included in Flutter. Over time however, I became much more confident in my Flutter skills, even whipping up completely new views and/or widgets in a matter of minutes.

On the back-end/data storage side of things, I initially struggled with the concepts of MVVM and state management initially. Specifically, learning to use Flutter's "provider" feature required me to follow along with a lot of tutorials and shift my thought process when considering how to represent my data behind the scenes. Along with this, our group, using an SQLite database, was able to implement a fully functional back-end in the closing stages of our beta development. This is something I am very proud of, as I had only minimal prior SQL experience, and so, by talking through the architecture of the database with my team and utilizing Dart's sqlite3 package, I feel I have a much better grasp on SQL and relational databases in general now.

In my part of the application, there were no deviations from the original design. Honestly, our original design depicted a very simple user account back-end, which I successfully implemented and then expanded upon while keeping my group up to speed. For example, the onboarding process is something I got around to that the team hadn't planned for originally. Also, there are a significant number of data interactions utilizing my view model that we had not originally planned for, like storing and displaying user statistics such as how many food spots they have saved and reviewed. I feel I was able to go above and beyond in this manner because I forced myself to sit down and learn state management within Flutter on a deeper level, which allowed me to quickly implement these fleshed out features in our app.

Finally, I'll summarize the various lessons, skills, and tools I learned throughout the course of this semester. First, the obvious one is Flutter. Like I mentioned above, my previous experience with front-end frameworks was fairly limited, and so a lot of the concepts such as providers, MVVM, and state management were new to me. Next, I got even more comfortable with using Git/GitHub with a team to complete a project. Lastly, the entire process of the Software Development Lifecycle from concept to design to implementation, always with a focus on the user experience, was something I had never gotten to participate in before. I really enjoyed the user-focused aspect of the course, as it allowed for us to get real-world, helpful feedback both before and during development. I have always been fascinated by providing top-notch user experiences, so I really enjoyed this process and think it will prepare me to be even more effective on a team throughout my career.