Reflection Report on Sayyara Automotive Matcher

Team 27, Kappastone
Tevis Doe, doet
Gilbert Cherrie, cherrieg
Rachel Johnson, johnsr12
Harkeerat Kanwal, kanwalh
Himanshu Aggarwal, aggarwah

[Reflection is an important component of getting the full benefits from a learning experience. Besides the intrinsic benefits of reflection, this document will be used to help the TAs grade how well your team responded to feedback. In addition, several CEAB (Canadian Engineering Accreditation Board) Learning Outcomes (LOs) will be assessed based on your reflections. —TPLT]

1 Changes in Response to Feedback

[Summarize the changes made over the course of the project in response to feedback from TAs, the instructor, teammates, other teams, the project supervisor (if present), and from user testers. —TPLT]

[For those teams with an external supervisor, please highlight how the feedback from the supervisor shaped your project. In particular, you should highlight the supervisor's response to your Rev 0 demonstration to them. —TPLT]

This section of the reflection report will cover the changes made in response to the feedback from our both our professor and the teaching assistants, as well as our supervisor. As we were an industry project, we did not receive feedback from other teams due to the NDA we needed to sign as part of development.

1.1 SRS and Hazard Analysis

The feedback received for the SRS was primarily from the teachers of the capstone course. The major piece of feedback that we tried to improve on was the how verifiable our non-functional requirements were. Our initial non-functional requirements were vague, such as T̈he product shall be aesthetically pleasing.; which is hard to verify. To fix this, in our revised SRS, we made the non-functional requirements more grounded overall, and split up many that were too broad to verify.

The functional requirements that the group chose largely based on the information that our supervisor provided to us directly through a confluence, and so no feedback was gathered from him.

The only feedback gathered for the Hazard Analysis was to add a list of tables and figures, which was added to the document in revision 1.

1.2 Design and Design Documentation

Feedback gathered for the design in general was taken from the demonstrations both as part of the course and to our supervisor directly. Our supervisor, however, did not have a lot of feedback to provide. A preliminary design was created on Figma to show our supervisor, who gave it the green light before development began. The rest of the design was critised by the TAs during the POC demos and the Rev 0 demos, where feedback on the ordering of screens was given. Mainly, feedback was given to consider the flow of a user as they travelled through the application. Through this feedback, the ordering of screens were changed, modifying the original design quite a bit.

Feedback for the design documentation was mainly in the form of missing requirements in traceability matrices and poor formatting. Both were fixed in revision 1 of the documents.

1.3 VnV Plan and Report

The VnV Plan and report were given feedback through avenue to learn, mainly covering tests that were either missing or incomplete. In response to feedback, results for tests were added, new tests in response to changes in the application were added, and the traceability matrices were updated to reflect these changes.

2 Design Iteration (LO11)

[Explain how you arrived at your final design and implementation. How did the design evolve from the first version to the final version? —TPLT]

The broad design for the implementation was decided well in advance through the use of Figma, and the team largely stuck to this design. That being said, some things did have to be changed compared to our original vision. As mentioned in the previous section, some user screens were re-ordered to make more sense for real use, and some screens were removed entirely. Over the course of the development of the implementation, we also found that we were missing screens that we may need, and so they were added as well.

The major changes that occurred to the implementation were in the back-end of the implementation. Initially, an ERD was created to show off the different models in the database and how they interacted with each other. The plan was to base the implementation on this ERD. The issue was, however, as the implementation progressed, we found that the diagram lacked many of the models and data that were needed for the front-end to function. As a result, there

are many, many new models that were not initially conceived as part of the database. And so, even though the design of the front-end changed in response to feedback from both the POC and Rev 0 demonstrations, the large changes from our initial vision were actually in what the TAs and professor could not see, the back-end.

3 Design Decisions (LO12)

[Reflect and justify your design decisions. How did limitations, assumptions, and constraints influence your decisions? —TPLT]

We made quite a few design decisions early on, before we started development on the project, mainly on the scope of the project. Our supervisor gave us a list of requirements they wanted for the minimum viable product, and we decided our assumptions and constraints around this. Unfortunately, however, we were not able to meet every requirement that our supervisor wanted, which was clear once we begun working on the project in earnest. This was mainly due to the time that we were actually able to spend on the project. Because of this, we had to narrow our scope further that what we originally intended, and our design decisions, mainly how the back-end was structured, had to be changed in response. The most important thing to note, however, is even if this scope was narrowed, most of our design decisions were made by our supervisor prior to meeting us. We were not able to choose the features we thought the product should have, the actions a user should be able to take, the difference between shop owners and customers, etc. Because of this, we were essentially constrained to his vision, even if, at times, we disagreed with his design decisions. This influenced our decision making throughout the project, as we were aiming to please our supervisor, and fulfill his requirements, rather than do something which made sense to us. Sometimes we couldn't justify his design decisions, and that's okay. We were still able to make something that we are proud of.

4 Economic Considerations (LO23)

[Is there a market for your product? What would be involved in marketing your product? What is your estimate of the cost to produce a version that you could sell? What would you charge for your product? How many units would you have to sell to make money? If your product isn't something that would be sold, like an open source project, how would you go about attracting users? How many potential users currently exist? —TPLT]

The idea for our project was initially conceived by our supervisor, who believed that there was a market for this type of product, and wanted us to create a minimum viable product in order to gather investors. After working on this product, we also agree that there would be a market for something like the application we've created.

Marketing the product would most likely involve initially getting various

shop owners to begin using the app in order to schedule their appointments and manage their shops. We would do this by giving them the application for free, or maybe even paying them to use it initially. This would give us a good user base, and allow shop owners themselves to see the benefits of the app and spread the word. Other ways to market the product would be fairly standard, using commercials, billboards, etc, likely focusing on one city to begin with before expanding.

The estimated cost to produce a version to sell wouldn't be that much higher than what we've currently put into a project. The main barriers are the lack of dedicated servers, and the lack of polish of the design. So long as the product began selling locally, dedicated servers would not cost a ton of money, and could be scaled up as needed. Something like Amazon DynamoDB, for example, allows you to only pay for the amount you actually use, and so until many users begin using the application, this would not be very costly.

The application would likely be sold to shop-owners using a subscription based model, ramping up the cost per month depending on the amount of customers it brings in among other things. Starting the application cheap, at something like $20.00 per month would allow there to be a low barrier of entry for small shops, while giving us room to charge more depending on the shop's characteristics. Perhaps if they want more services, more employees, more locations, they would have to pay more. Not charging customers is key here. It would be impossible for shop owners to get customers to use the new application if they had to pay to use it, and so by keeping it free for customers we can hopefully keep if profitable for shop owners.

As said earlier, there isn't really an amount of units we would need to sell to initially make profit. However, if the product grows in popularity beyond a small scale we would likely have to revisit how we're hosting our front-end, back-end, etc. and determine what we would need to sell in order to actually make a profit with a large-scale operation.

5 Reflection on Project Management (LO24)

[This question focuses on processes and tools used for project management. —TPLT]

The following sections detail the effectiveness of our project management plan.

5.1 How Does Your Project Management Compare to Your Development Plan

[Did you follow your Development plan, with respect to the team meeting plan, team communication plan, team member roles and workflow plan. Did you use the technology you planned on using? —TPLT]

Throughout the course of the project, we managed to continually follow majority of the plans outlined in our Development Plan.

In our team meeting plan, we committed to online weekly meetings on Discord. We were able to maintain this schedule throughout the whole project term, only missing one week due to exams and the holiday season. Although we did not have official meeting minutes, we had a standard flow of starting with a quick standup, then discussing next week's tasks, and finally an open discussion of issues, questions and blockers. Meeting notes were take every meeting and posted in the meeting-notes channel in the team discord.

Our team communication plan was also followed during the project. Both Discord and GitHub Issues were used to communicate about project-related topics. Discord was used as our main method of team communication, while GitHub Issues was used to raise new action items and keep track of progress.

Most team member roles were also followed. One change we made early on was to change the meeting arbiter from Harkeerat to Tevis. This was done since Tevis naturally led most of the meetings.

Finally, the development plan was also followed for the most part. The steps outlined in the development workflow section were followed by all developers for all tasks. Rules for issue, branch and pull request nomenclature and management were also followed in order to keep us organized. Specific GitHub features were used including issues and actions were used for this as well. Features that were not used to their full potential were GitHub Issues and Projects. As a team, we did not keep up with creating and managing milestones for our sprints. Instead, we used our weekly meeting notes posted in Discord to track progress.

5.2 What Went Well?

[What went well for your project management in terms of processes and technology? —TPLT]

The things that allowed our team to succeed were our consistent meetings, effective communication, and adherence to our development workflow. Since we met weekly and communicated daily through discord, we were better able to keep track of each other's progress, be on the same page when it came to implementation, and voice opinions and ideas without hesitation. Altogether, this allowed us to work very well as a team despite not knowing each other prior to beginning the project. This also allowed us to easily pick up the slack after losing a member of the team early in our development. Furthermore, being able to have a strict workflow to follow when developing, allowed for broken main branches, code errors, and conflicts.

5.3 What Went Wrong?

[What went wrong in terms of processes and technology? —TPLT]

Though we worked very well as a team over the course f our project, there were a few things that could have been improved to offer more efficiency. Firstly, we could have made better use of our GitHub project board and milestones. These GitHub features were abandoned very early, which led us to not have

very structured sprints. Using the board, would have ensured that we were more aware of due dates and the work needed before them. Furthermore, we did not review pull requests as quickly as we could have which delayed implementation. This is in part due to team members not requesting specific reviewers and also team members not receiving notifications for review requests. It would have been beneficial to have a development rule about reviewing pull requests and also having a dedicated section in the meeting to discuss open pull requests.

5.4 What Would you Do Differently Next Time?

[What will you do differently for your next project? —TPLT]

In the future, we would focus more on having a more structured project schedule and sprints. We will make sure to use the necessary tools to aid in organization of sprints and spend time effectively planning each sprint prior to the start of the sprint. This will allow us to keep better track of upcoming due dates and help us avoid working toward each deadline as they come up.