Reflection Report on Sayyara

Team 31
SFWRENG 4G06
Christopher Andrade
Alyssa Tunney
Kai Zhu
Ethan Vince-Budan
Collin Kan
Harsh Gupta

April 6, 2023

1 Changes in Response to Feedback

1.1 SRS and Hazard Analysis

1.1.1 SRS

The SRS document was missing lists of tables and figures that would make for easier navigation through the document, so these were added. Additionally, we also went through and removed any requirements that were no longer being used for the project. This included things that were outside of the scope of the MVP, such as payment information for a shop owner, mobile application requirements as the application did not need to be deployed for mobile use for revision 1, and additional features such as work orders and vendors. Requirements kept in the document were modified if they were not specific enough, and were also modified to remove any features that we did not need to implement for revision 1, such as the mobile application. We also added additional requirements that were originally not part of revision 0, namely section 10.1.6, the Shop Lookup Requirements section. Tables such as the traceability matrices were modified to remove any requirements that are no longer part of the project and to add any new requirements we missed originally. We also added which functional requirement corresponds to which revision of the project, with revision 2 taking place post-capstone for any team member that would like to continue working on the application. All requirements were also given a priority listing of high, medium, or low.
Additionally, we modified the functional decomposition diagrams to make them more simplistic and to get a better picture of what was expected for the entire project, including post-MVP features. All sections of the diagrams have been

organized by major feature, and diagrams are separated by the users and their intended accessible features.

1.1.2 Hazard Analysis

In the Hazard Analysis document, we needed to reformulate our concept of what the hazard analysis is and what the purpose of the document is. These changes were made, and the document was also scanned for any requirements that were no longer part of the project. We also ensured we had a list of tables, and did a final spelling and grammar check for revision 1.

1.2 Design and Design Documentation

The System Design Document was given a grammar and spelling check for revision 1. Additionally, system variables have been added to this document as they were not present in revision 0. This includes the sections of monitored variables, controlled variables, and constant variables.

Both the Module Guide (MG) and Module Interface Specification (MIS) were updated to remove any trace of requirements that are no longer required for revision 1, and were modified for grammar and spelling for revision 1. Additionally, the MIS was updated to be slightly more consistent with the syntax between applicable modules.

1.3 VnV Plan and Report

1.3.1 VnV Plan

Any planned test cases that were going to be used to test requirements that are not present in the revision 1 source code have been removed, such as testing for mobile compatibility. These tests are not necessary for revision 1 and since their related requirements have been removed from the SRS document, their respective planned tests have been removed from the VnV Plan. Planned test case FR-ApR2-1 was removed as customers cannot currently schedule appointments for themselves as of revision 1. Any mention of customers scheduling appointments has been removed from planned tests as well. Planned test cases FR-ApR3-3 and FR-ApR3-4 were removed for this reason as well. Mobile compatibility testing NFR-OER2-1 was removed due to no longer being a requirement of revision 1. NFR-MSR1-1 was removed for this reason as well.

The planned unit tests have been added to the revision 1 update of the VnV Plan document. These were performed for the results of the VnV Report, but needed to be added to the VnV Plan as well.

1.3.2 VnV Report

The VnV Report was relatively well done by the team and there was a grammar and spelling pass performed for revision 1. Additionally, any test cases that were used to test requirements that are not in the revision 1 source code have been removed, such as testing for mobile compatibility. These tests are not necessary for revision 1 and since their related requirements have been removed from the SRS document, their respective tests have been removed from the VnV Report. All planned test cases outlined in the Vnv Plan changes were removed due to not being needed for revision 1. We also removed any mention of the application needing to be deployed, as this is no longer a requirement of revision 1. Any tests related to this feature have been removed as well.

2 Design Iteration (LO11)

Around the time of development of the proof of concept for Sayyara, the team realized that some of the features presented by our supervisor Nabil were not part of the MVP (minimum viable product) and were considerations to include if we had time or for future iterations of the application. This caused us to reevaluate the features we had planned on implementing in the SRS document. One of the features that we had planned for the application but had not implemented yet was the Vendors feature, allowing shops to order parts used for servicing from vendors directly from the website. Once we learned this was not part of the MVP, we decided to keep it in mind but take it off of the list of features to develop for the time being due to potential time constraints of implementing the rest of the more important features of the application.

The team also continued to evolve Sayyara based on the constraint that it was to have a minimalist user interface. This was a consistent specification from the first version and has continued to be a factor through the final version. This constraint was important to us because the application needs to be accessible by users who may not be very tech savvy and will need to be able to navigate quickly and easily to the various features of the application. All features and pages of the application are designed with this constraint in mind.

The team also took inspiration from other existing applications that offered similar services (OpenBay, RepairPal, YourMechanic, to name a few). We examined their workflow processes and forms they presented to the user and determined what worked and what did not in terms of ease-of-use, understandability, and navigability. We took inspiration from what worked on these applications and developed our own methods to ensure the application met design constraints. These were present starting in the revision 0 version of the application and were improved for the final version.

Once the team completed the revision 0 version of Sayyara, we had the opportunity to demo the application to our supervisor Nabil. He offered some valuable feedback and said we were on the right track overall. There were a few minor user input adjustments that he suggested that would align with our constraint of the minimalist, easy-to-navigate user interface. This included making the shop registration page more consistent with the rest of the pages on the application, since visually it appeared a bit different from the rest of the forms on the page, and allowing the user to pre-select a service when creating a quote request rather than having to type out a description of the service they wanted. Overall, we had implemented all of the main features of the application requested by Nabil, we just needed to make minor adjustments to these features to ensure they met the requirements outlined in our SRS. The final version reflects these final improvements.

3 Design Decisions (LO12)

Because the Sayyara application is meant to be a simple application that needed to contain features that would be easy to use and navigate for users with various different technological backgrounds, we needed to keep this in mind when developing our design decisions for the application. Since the team knew what features we needed to implement, we decided to take a look at similar existing applications (Openbay, RepairPal, YourMechanic, to name a few) to see what to do and what not to do when it came to the visual aspects of the application. Some of these applications appear cluttered and a bit overwhelming and users who need to actually use the applications would likely feel the same way. We wanted to ensure the user was able to follow the flow of the application starting with this registration process. We took these decisions into account and the final version of Sayyara is the result of that; all features have a minimalist, easy-to-navigate user interface from start to finish. Another constraint of the application that we kept in consideration during development was a simple colour palette. The application has a solid dark-coloured background with light-coloured text that stands out that does not overwhelm the user. Along with this, we needed to ensure the layout of each page was simplistic. Each feature page has minimal components with individual cards separating each piece of information so the information is easy to digest and navigate. We strived to design the frontend components of the application with the final user in mind.

Demonstrating the revision 0 version of Sayyara to Nabil allowed us to gather some final feedback on the application before the final version was completed. We were on the right track with the application and our design decisions up until this point were the result of that. One piece of information we received from Nabil was that the Sayyara application did not need to be deployed onto a website for the MVP by the end of the term. Because of this, the team decided that since the application would not to be accessible on mobile devices, that we would instead focus on fine-tuning the desktop version of the application to

ensure that it met all design constraints and goals. We have succeeded in this goal and the application is fully compatible with desktop devices using the most popular web browsers.

4 Economic Considerations (LO23)

Currently, there is no application in the automotive industry that offers every service the Sayyara application offers for both customers and shops. There are several existing applications that offer a few of the services, but there is no single application that does it all. We believe there is a market for this type of product as this offers a single destination for shop owners to advertise their shop, store all their shop's information, and overall automates processes that they might be performing manually or have a separate employee handling. This application is also beneficial for customers, as it eliminates the process of having to Google search for several different shops in their area, calling these shops, and potentially booking several appointments with these shops just to receive a quote. The Sayyara application allows them to easily navigate local shops in their area, filtering by services they may need, and allows them to create a quote request and chat with these shops without having to bring their vehicle to the shop.

Attracting potential shops to the application could prove to be difficult due to the fact that shop owners are typically satisfied with the processes they already use to quote customers and service their vehicles. A way to attract shop owners to use the Sayyara application could be done through a demo of the application showing all of the features the application provides that could assist in automating the processes they already perform manually. We could ask for their feedback on the application; perhaps if there are any missing features that they could benefit from if they were added to the application. Having feedback from automotive shop owners would allow us to modify the application in ways that would make it more appealing to be used in the industry.

On the other hand, it could be difficult to attract new customers to the application due to them already being satisfied with the services their current automotive shop provides. By gathering feedback from users on what might get them to browse around, we could get a better idea on how to attract customers to the application. Whether that be a review system of shops on the application, price discounts, or just being able to do everything from the comfort of a single application, we would work with potential customers to determine what they would need to try out Sayyara.

When discussing with our supervisor Nabil, he mentioned that shop owners would pay a fee to have their shop housed on the Sayyara application. By charging shop owners a monthly subscription fee, say $10 to $20 a month, we

would need a significant number of shop owners to be signed up on the application in order to turn a profit decent enough to pay each developer. Around 1000 monthly subscriptions would still not be sufficient enough to make a living for each developer of the application, so perhaps taking a percentage of a service fee for each service a shop books through the application for a customer could be a source of profit as well. Overall, there would need to be a significant number of shops paying for the monthly Sayyara subscription with a significant number of services being booked in the application in order for the application to truly be profitable for each developer on the team. This is something to take into consideration for the future and how we can improve the profitability of the application is something to think about.

5 Reflection on Project Management (LO24)

5.1 How Does Your Project Management Compare to Your Development Plan

The team conducted weekly meetings to sync on current progress and next steps of development. When the team was unable to meet at the initially set meeting time of Friday at 2:30PM, accommodations were made and the team managed to sync at a different time during the week to ensure that the meeting still occurred. The team Discord was well-organized with different channels being used for different components of the project, and separate threads were opened to track issues that occurred during development. The Discord threads were used for lower-level issues and troubleshooting, while GitHub issues were opened to track the progress of tasks at a higher level. The GitHub issues were attached to pull requests once the related code was completed so team members were able to view the code for said issue before it was merged. Having the GitHub issues were helpful to track the bigger picture of the project and how the progress of overall development was going.

Team member roles differed from the roles documented in the Development Plan as everyone had the opportunity to work on little bit of everything. Once the features of the Sayyara application were fleshed out, each team member was assigned a section to work on, and would work on all components of that feature, from planning, to implementation, to testing. There was still room for overlap as team members would assist each other in sections that needed more attention.

The workflow plan was followed with the exception of implementing CI/CD in the GitHub repository. The workflow plan was still executed well otherwise, with the team being diligent and storing their changes in separate branches on the repository rather than committing everything to the main branch. Pull requests were performed once code was ready to be merged, and team members reviewed each other's code before pull requests were merged to the main branch. This eliminated the risk of anything breaking on the main branch if code was

to be merged there first.

Additionally, with the exception of GitHub CI/CD, the team used all technology that we planned to use. We originally had written Python as the backend technology as we were unsure about whether to use Django or Flask, but we kept the Python backend technology and decided on the Django framework for the project.

5.2 What Went Well?

The team made good efforts in writing well-organized documentation, especially the SRS document, that could in turn be used to efficiently develop the application source code. It is much easier to develop source code when the documentation is well-organized and laid out in a manner that is easy to understand. We believe this portion of the project went well for us and was beneficial for us as it reassured us that we were creating the product right, and we were creating the right product.

The team also made good efforts with Discord communication. There are several channels in the Discord server, used for general chat, meetings, resources, and issues. Each channel was utilized appropriately and offered a great way for the team to communicate about each topic. The issues channel was very important and every time a team member was experiencing an issue during development, they would post their issue in the channel and somebody was always there to assist in resolving the issue.
Along with the efficient Discord communication, the team took advantage of GitHub issue tracking and opened issues for each feature of Sayyara that needed to be implemented in source code. Issues were also opened for project deadlines and included checklists of items that needed to be completed before the deadline. Having the ability to assign pull requests to issues was also very helpful as we could close the issues as soon as the pull request for that issue was merged. GitHub issues were very helpful in outlining what the team needed to work on and who was working on what issue.

We believe the team chose a good tech stack for the Sayyara application. All elements of the tech stack that were selected are frequently used in the industry for web development and we believe that this was great exposure and experience for potential future projects that we may develop. There was at least one element of the tech stack, whether that was Next.js, Django, NGINX, or Docker that each team member was unfamiliar with, so it was a great learning experience for everyone to be able to get exposure to a new type of technology.

Lastly, the team believes their choice of using Docker to containerize the Sayyara application was very efficient and allowed each team member to be able to easily run and test the application regardless of what computer operating system they were using or what specifications their computer had. This also allows anyone

outside of the team to run the application by simply installing Docker on their machine and running a shell script to start the container. The components in the container are easily modifiable and can be added to just as easily. This was a great introduction to Docker for a few members on the team as well.

5.3 What Went Wrong?

Some requirements that were initially outlined in the SRS document turned out to be no longer needed for the scope of the project. This should have been updated in the document immediately upon receiving this information, but the documentation was modified for the purposes of the final documentation revision. Having these requirements still present in the SRS meant that they cascaded through to other documents such as the Hazard Analysis and VnV Plan when in reality, they are not implemented in the final version of the application as they were not part of the MVP. It would have saved the team time if we had updated these requirements more frequently rather than having to now modify multiple documents to remove requirements that are no longer necessary.
Also related to documentation, some of the nonfunctional requirements were not specific enough in hindsight. Having requirements that are not specific enough can make them difficult to measure, and this in turn makes them difficult to test. Modifying these requirements in the SRS earlier could have resulted in easier testing implementations in the VnV Plan and VnV Report documents.

For some aspects of the application, the development of the code took place before the actual documentation was completed. Typically it is more efficient to plan and document the application before delving into the code development. This did not affect development or the eventual documentation, but this could be viewed as a bad practice as documentation should typically take precedence before development begins.

Most team members had frequent issues with merge conflicts when trying to merge main with their personal branch, or when they were trying to merge a pull request to main. While there aren't any ways to completely get rid of merge conflicts in Git, there are ways to mitigate having to deal with merge conflicts so often. The team should have merged their branches with main more frequently so they weren't so many commits behind main, and they should have made better efforts to create pull requests more frequently so instead of trying to merge one large change, several changes over several pull requests were performed instead. This would have improved efficiency of merging code when new features were implemented.

5.4 What Would you Do Differently Next Time?

For our next project, especially if it is a group project, we plan to ask questions to our peers more frequently. Several issues could have been resolved quickly

and efficiently if we had asked our peers for help with a certain section of the code or with documentation instead of spending time trying to find the answers on Google. Each team member brings a unique set of skills to the table and chances are we are able to help each other by simply just asking questions and not being afraid to ask for help if we need it. This issue only occurred a few times throughout the two terms of the capstone project, but could have been mitigated better and is something the team plans to learn from in the future.

For future projects with large scopes such as Sayyara, we plan to update and review our requirements for projects more frequently. Requirements tend to change throughout the development process of an application, and Sayyara was no exception as the project progressed. The team kept a condensed list of requirements, but did not update the SRS document as necessary, so as the requirements changed, the documentation did not. This goes for all documentation, not just the SRS. Reviewing the documentation every once in a while in a team meeting would be a great way to ensure that we are on the right track and that we ensure the documentation is being updated with the changes we make throughout the development process of the application.

Lastly, the team would like to incorporate the GitHub CI/CD feature for future projects. While separate testing was conducted on the Sayyara application for the VnV Plan and VnV Report documents, having the GitHub CI/CD feature automatically test code as it was pushed to the repository would have been a more efficient way to ensure that the code was working well with the code that was already in the repository. It would have also provided additional testing, and there is no such thing as too much testing. This feature will be explored for future projects.