Reflection Report on Grocery Spending Tracker

Team 1, JARS
Jason Nam
Allan Fang
Ryan Yeh
Sawyer Tang

1 Changes in Response to Feedback

1.1 SRS and Hazard Analysis

For the SRS, several changes were made based on TA feedback. In the User Business section of the document, we added statistics and citations to make the project's purpose more compelling. Introductions were also added to all sections in order to give readers an overview of what information would be covered. Furthermore, product use case numbers were added in the Individual Product Use Cases section for better traceability. Changes were also made to the Tasks section of the document. Here, we added timelines and priorities for each development phase based on feedback. Lastly, formatting issues were addressed such as changing the image formats of all images to pdf in order to improve their quality. After the Revision 0 demo, we also made some changes due to internal decisions. In particular, we had to reduce the scope of the project due to time constraints and as a result, some requirements were removed.

From TA feedback, the critical assumptions made in the Hazard Analysis were updated. In particular, assumptions in regards to the hosting service we would be using for the backend and the user's camera functionality, both of which are largely out of our control. We also removed the weak assumption we originally had in the section. Additionally, we were given some additional suggestions to prevent the log in hazards we had identified, so those were included and a new security requirement was created. From peer feedback, we included the critical assumption that the user's phone would be functioning properly since that is out of our control as well.

1.2 Design and Design Documentation

For the Module Guide, several changes were made based on TA feedback. One such change was including traceability in the Module Development timelines. To accomplish this, references were placed into the timeline tables such that readers can quickly view the contents of each module. Additionally, there were

some implementation details regarding specific technology that should not have been present in the document. As a result, these were removed. Lastly, we were informed that we should be more specific in regards to the Hardware Hiding Module. Therefore, we added more details to that module section including specific hardware devices such as a camera. From peer feedback, a minor change was made to update the name of M2 to be consistent with the name it has in other sections of the MG and MIS.

In the Module Interface Specification, we were informed via TA feedback that some of our syntax was incorrect. Specifically, syntax regarding types on declared variables and their definitions. There was also some improper text formatting problems with multi-character variables, all of which have since been resolved. Additionally, modules in the MIS were missing their Module Types so a new section was added for all modules to define their module types. Based on peer review feedback, we added definitions for some additional notations (i.e. Optional<Object> and $\mathbb{N}^2$) we used throughout the document. We also clarified an unclear assumption made in Section 7.4 where the other team did not understand what "H" meant in the context of a receipt input.

1.3 VnV Plan and Report

Based on TA feedback, we added symbolic constants to our VnV Plan. For instance, rather than using a fixed number such as 70% in our test descriptions, we changed it to a constant such as PASSING_RESPONSE_RATE. Additionally, we were asked to provide more details on some tests. Specifically, what constitutes as a pass and some other ambiguity regarding survey usage. Lastly, the Implementation Verification Plan was missing some details so those were added as well. Based on peer review feedback, some test cases were updated to include missing information such as an initial state, input, and expected output. There was also feedback to add some more detail to the Design Verification Plan and alphabetize the Symbols, Abbreviations, and Acronyms table, both of which were addressed. Finally, based on internal decision making after the Revision 0 demo, we decided to reduce scope and had to remove some test cases that were no longer relevant to the application.

For the VnV Report, many of the TA feedback changes that were addressed had to do with formatting and appearance of the document. This included alphabetizing the Symbols, Abbreviations and Acronyms table, changing some of the table appearances, and adding table rows with short test descriptions to better show what each test is validating. There were also some minor changes requested such as GitHub stylizing and explicitly mentioning table names instead of something like "the above table". Some peer review feedback was also addressed in the VnV Report, such as some ambiguity regarding one of our performance graphs and some missing details on specific tests. We were also asked to explain who the external users we tested with were which we resolved by pointing readers to the Stakeholders section of our SRS document.

2 Design Iteration (LO11)

The biggest design change in the application had to do with the receipt scanning portion of the application. Initially, the vision for the application was to have our own optical character recognition (OCR) system as part of the application. As such, the original implementation of the OCR involved the use of various Python scripts using Tesseract OCR and OpenCV. This was demonstrated as part of the proof of concept demo, however soon afterwards, we realized we were having issues with robustness, accuracy, and speed. Given that this was meant to be a consumer application, usability was a very important trait for the app and these issues were significant roadblocks for us. After talking to Dr. Smith, he referred us to Dr. Anand who gave us feedback regarding our implementation. He suggested us to use an existing OCR implementation given that there was already existing technology that solved what we were trying to accomplish. Additionally, in the interest of time, we did not have the resources to create our own, optimized OCR implementation in addition to a full mobile app. As a result, we decided to take Dr. Anand's advice and use Google MLKit as our final OCR implementation. This design change provided significant improvements to the app's usability, with robustness, accuracy, and speed all being positively impacted. This also allowed us to dedicate resources to other parts of the application resulting in a more complete app than what otherwise would have likely been created. This design change had the biggest impact on the final outcome of the project and we believe the current implementation is far better than the original implementation we had originally planned.

The other large design change for the application had to do with the user interface (UI). In our Revision 0 demo, we had a very preliminary UI that simply served to navigate to the different features we wanted to show. At the time, we had put very limited thought into the actual user experience and visual appeal of the app due to prioritization of features, however part of the feedback we got from the demo was finalizing a UI design since we would have to do usability testing as part of the following deliverable. This resulted in us looking back to the data we had originally elicited from McMaster students during the SRS creation. The general consensus from the original user feedback was that they wanted an application that felt familiar. As a result, we referenced several existing applications during this development period and settled on the final design which takes inspiration from apps such as Instagram, McDonald's, and Apple Health. We ended up finalizing a colour scheme and polishing some of the components to make everything look more cohesive as well. After completion, this final UI design was very positively received by the McMaster students we showed it to and during usability tests, these same students were able to figure out general functionality within 15 minutes. As a result of putting more emphasis on user feedback, we were able to create a UI design that was more usable and appealing than the original implementation we had demonstrated.

Based on feedback during the Revision 0 demo, we also made changes to the budgets. For example, we displayed the user's total budget allocated to each goal, rather than only showing a bar of progress as was initially done.

Additionally, based on user feedback, tap focus was added to the receipt capture in order to improve the user experience.

3 Design Decisions (LO12)

Due to limited time, some of the design decisions we made involved scoping down or changing the project. For instance, as previously mentioned, we used Google MLKit for our OCR implementation rather than making our own. This was due to Dr. Anand's recommendation but also due to the fact that there was more to our capstone project than just the OCR portion. As a result, we used an existing implementation in order to focus efforts on other portions of the app. We also had to scope down some of the features we had originally planned. We had initially intended to have location support in order to give recommendations based on user and store location, however, we found that we had overcommitted and did not have enough time to implement this functionality. As a result, it had to be removed from the current scope of the project.

Another design decision that had to be changed during the development period pertains to the Classification Engine. Originally, we had planned to construct a database of products in order to classify items from different grocery stores. This, however, was not feasible as after contacting different stores, we were not able to acquire a list of product data. As a result, we had to use web scraping and approximate string matching to construct our own database as users input receipts into the system.

Finally, we had planned for iOS and Android support in the final version of the application however the final version is only supported by Android devices. This was a result of limited time, but also restrictions placed on developers by Apple themselves. Apple only allows users with Apple devices to develop iOS applications. Furthermore, only two members of the team had Mac computers/iPhones while the other two had Android and Windows devices. This meant all developers could work on Android support but only two developers could handle iOS support. Since not everyone was able to work on it, we decided that iOS support would be out of scope for the current implementation. In other words, iOS support had to be pushed due to constraints put on iOS development by Apple.

4 Economic Considerations (LO23)

Since this product was developed with the low-income public as the primary stakeholders and users, we feel strongly that there is a substantial demand and potential users (essentially the whole low-income population) for the services our product provides. The estimated cost to fully develop and produce a version that we feel has enough functionality to be suitable for the market would likely be around $100,000, primarily paying for the salaries of the development team for 6 months. Due to the charitable nature and motives for developing this product,

we are highly motivated to provide the services of the product for free. Taking this into account, the team will need to consider alternative methods to generate revenue, such as a freemium business model, in-app advertising, or the sale of user data (ethically of course). Marketing the product and attracting users will likely begin with low-cost marketing efforts such as social media accounts, network marketing, and working with non-profit organizations at McMaster University. At this point, the team feels it is not likely that the product will be able to generate significant or break-even revenues unless the product can be bought out by a larger organization that can operate the product at a loss or can find other ways to monetize.

5 Reflection on Project Management (LO24)

5.1 How Does Your Project Management Compare to Your Development Plan

In terms of the team meeting and team communication plan, both were followed closely as documented. Meetings took place weekly on Wednesdays and Sundays with certain exceptions such as exam periods. Roles were also followed closely for each meeting and were consistent throughout the project's duration. Moreover, we used Discord, SMS, and GitHub for communication as was described.

For the team member roles documented, they were generally followed with one change. Jason Nam and Sawyer Tang followed their roles as described however Allan Fang's and Ryan Yeh's roles slightly changed due to workload capacity. Allan Fang led frontend development as written but due to extra capacity, Ryan Yeh took on the lead for OCR development in addition to his other roles. Besides that, the team roles were followed as written.

Regarding the workflow plan, Git and GitHub were used as described in the document with issue tracking and the version control process being followed.

Finally, the technologies described were followed throughout the project with the exception of the usage of node-tesseract-ocr. This library was included due to the assumption we would be implementing our own OCR, however we ended up using Google MLKit instead to handle the receipt capture process.

5.2 What Went Well?

The scheduled meetings were very useful to ensure everyone was on track for deadlines. It also gave transparency regarding any roadblocks and offered an opportunity for other members of the team to offer suggestions and problem solve together. Overall, it helped keep everyone accountable which ensured quality work was delivered on time. The GitHub issue tracking process was also very useful in tracking what was being worked on and what still needed to be worked on. The issues helped us to organize our work and isolate different tasks through the use of labels as well. This made the division and tracking of work very easy and more efficient overall. Finally, the CI/CD pipeline via GitHub

Actions was very helpful in terms of development. In particular, having unit tests automatically execute when new code was pushed helped us to ensure that high quality features were being produced and that there were no new problems being introduced.

5.3 What Went Wrong?

We did not have anything that explicitly went wrong however one problem that was encountered multiple times during the project's duration had to do with the workflow process and pull requests. Part of the workflow involved waiting for two reviewers to approve a pull request before a merge could take place. Often, pull requests would get "stuck" because reviews were not occurring as frequently as they should have. As a result, pull requests would pile up and merged in bulk rather than a constant stream of work being pushed through. This would often mean that before a deadline, everyone would go through a large number of pull requests to push everything that was required for a deliverable which was not ideal.

5.4 What Would you Do Differently Next Time?

For projects in the future, we would like to improve the overall planning. For this project, we had to change scope/design decisions several times which was to be expected, however better planning could have reduced the number of times we had to do it. Next time, we would like to create more detailed timelines for development with exact expectations documented for each period. This would help with deciding feasibility for features and allow us to set a better initial scope. More research would also help during the initial phases of the project. For instance, by researching more OCR approaches/libraries, we might have been able to settle on an off-the-shelf solution earlier rather than putting resources into something we would eventually scrap. Lastly, we think that speaking with experts/more knowledgeable individuals before starting a project would be very important as well. Our meeting with Dr. Anand was very insightful and if we had the opportunity to speak earlier, we likely would have been able to make better design decisions right away.

We also believe that communication could have been improved, especially in regards to pull requests. Something as simple as more frequent reminders could have improved the throughput of code merges and produced a more consistent stream of pull requests. As such, frequent communication and reminders would likely be beneficial for more efficient future projects.

6 Reflection on Capstone

6.1 Which Courses Were Relevant

In terms of documentation, SFWRENG 2AA4 was very helpful for this capstone project. This course was our first introduction to LaTeX and writing soft-

ware documentation. Working on documentation in that class greatly helped with writing and working with the documentation for this project. Similarly, SFWRENG 2DM3, SFWRENG 2FA3, and SFWRENG 3RA3 all helped with documentation as well. SFWRENG 2DM3 and SFWRENG 2FA3 were introductory courses to discrete mathematics which was needed for formalizing our requirements and design, such as in the MIS. SFWRENG 3RA3 taught software requirements and was the course that introduced us to different elicitation techniques. This course was leveraged for the different meetings we had with stakeholders to elicit various requirements. It also taught the different kinds of requirements (function and non-functional) which also used for this project, particularly in the SRS. Outside of documentation, SFWRENG 3DB3 introduced us to relational databases which we used in our final implementation. This course built the fundamentals which allowed us to plan and construct our own database for this application. Furthermore, SFWRENG 3A04 taught the basics of software architecture which greatly helped with the design process of our application. Referencing the different architectures covered, we were able to build out our own application for this project. Finally, SFWRENG 3S03 was important because it taught the basics of software testing and the different testing methodologies available. This course helped us to build our own testing library with the appropriate code coverage to ensure our application was reliable and functioning as intended.

6.2 Knowledge/Skills Outside of Courses

Much of the knowledge/skills related to specific technologies required learning and research outside of our coursework. In particular, Flutter/Dart was a new programming language for all of us so we had to do research on the documentation as well as watch some videos to familiarize ourselves with it. Likewise, not all of us had worked with NodeJS before or PostgreSQL so similar learning had to be done. This involved learning more than just the language as well; we tried to familiarize ourselves with the best coding practices and syntax too. Another skill we had to learn was using Azure and how to deploy an application on it. We decided to use Azure to host our backend and database, so there was a learning curve to configure everything properly. There was also a great deal of troubleshooting required which we had to utilize external resources to solve. CI/CD and specifically, CI/CD through GitHub Actions was also new for most of us. All of us understood CI/CD conceptually as it had been briefly covered in coursework, however using it in practice was something we had to teach ourselves. This involved learning the file structure for workflows, the syntax, and importing of existing actions to create our CI/CD pipeline. Overall, the knowledge/skills outside of courses were mostly technology specific and although some of the concepts were familiar, we still had to learn how to actually use them in practice.