Reflection Report on Flick Picker

Team 7, 7eam
Talha Asif - asift
Jarrod Colwell - colwellj
Madhi Nagarajan - nagarajm
Andrew Carvalino - carvalia
Ali Tabar - sahraeia

Contents

Table 1: Revision History

Date	**Developer(s)**	**Change**
April 3	Talha	Adding Section 2, 3, and 4
April 4	Talha	Adding Section 5

This document covers the 7eam's reflection based on the year-long capstone project. It contains everyone's thoughts summarized in a digestible manner. In addition, the report will cover the changes we made in response to feedback while being assessed on the Canadian Engineering Accreditation Board Learning Outcomes.

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]

1.1 SRS and Hazard Analysis

1.2 Design and Design Documentation

With regards to Design Documentation, there were not any significant or unforeseen changes to our project's Software Architecture and modules. As a result, our implementation closely follows the plans described within our Design Documentation. Revision changes within the documents were mostly minor changes, which include improvement of diagrams, the addition of a UML diagram, and other miscellaneous changes.

1.3 VnV Plan and Report

2 Design Iteration (LO11)

The final design and implementation contain two pieces: the front-end, which is the client, and the back-end, which is the server. As the client is user-facing and directly interacts with it, the design is swayed by feedback and how the application "feels" to use. Whereas the server is not directly interacted with by the users, they only see the results of it. Given this, the client was changed more than the server.

2.1 Client Design and Implementation

Going deeper into the client's design, we, as developers, are also users of the application, where we had a decent idea of how the client should look and the overall flow of it, which is what we started with. Using Figma, 7eam could display their thoughts as pictures, where each component should be, and what pages are needed. Going from Figma to the actual initial implementation had a few colour changes and made the application feel smooth. However, after

publishing the application and collecting feedback, more changes had to be made:

- Additional preferences were added
- History functionality was added
- Layout was modified on some pages
- Improve friends feature

Making the difference between us as developers and users abundantly clear. We were extremely comfortable as we had been looking at the pages endlessly, but a user coming in for the first time had to learn the layout. 7eam wanted to make sure that the users were able to learn how to use the application in a fast and simple way. Implementing and designing the client is a constant reiterative process. Users can provide more detailed feedback on what they expect as they use the application more. The client's current iteration cannot be considered final and is expected to evolve continually. Still, the abovementioned changes are how the client changed from the first iteration to the current one.

As an aside, authorization is also handled by the client as it is delegated to Firebase, which did not undergo any changes from the initial plan. There are many benefits to using Firebase, Flick Picker is more secure, user data more secure, and it makes server design easier.

2.2 Server Design and Implementation

Looking at the server instead, compared to the client, it had a much more straightforward design and implementation process, with fewer changes. At the start, 7eam knew what tasks the server had to accomplish, and since we created both the client and server, the API responses will be used consistently across the application. First, a plan for the objects was made, then implemented into TypeScript objects while being tested in the database on how the response is structured. The minor changes made to the server up to the current implementation was just adding a few more objects in various API calls, namely User IDs, to make linking between database documents easier. Overall though, this does not mean the server will undergo zero changes. Again, as users get more comfortable, their feedback might need to be implemented on the server, not just on the client.

As an aside, since Flick Picker uses two different external API endpoints to collect information about the movies and tv shows verses the animes, they are not unified, so we needed to unify them and create a intermediate step so that the client could use them efficiently, which is also in the final iteration of the server.

3 Design Decisions (LO12)

Flick Picker was split into seven modules, each designed separately and then pieced together to create the application. Breaking the application up makes sense, as the results of each module interact with one another, but the path to get to that result is unique to that specific module. For example, the "Groups" implementation is entirely separate from the "Friends" implementation, however "Groups" uses the result of "Friends" to add users to a group, meaning "Groups" does not care how "Friends" is implemented, it just needs the result. Following this design decision made developing the application more organized and something we could use in the industry while working on other projects.

We had to consider some limitations, though, authentication being one of them. Since we are using Firebase to store user application data, it was only natural to delegate user authentication and oauthentication to Firebase since it provides those services, which is also why we picked Firebase as a database. Knowing that we designed the client and server differently than we would have if we could not use Firebase's services. Furthermore, using two different external APIs for movies/tv shows and animes creates a limitation and a constrain as they do not have the same response structure, so we need an intermediate step in the server to convert the response into something the client can use, which was something we had to take into consideration designing the server and overcome the constraint.

One of the most prominent assumptions we made while creating the client is that users will have an idea of the show being recommended to them, which means that we provide little detail of the show itself in the application whether this is correct or incorrect needs to be determined by collecting more user feedback.

These limitations, assumptions, and constraints went into the initial design of the application, and if user feedback coming pushes back against these considerations, then 7eam has to go back and find ways around them. There is already a potential example of including the recommended show's synopsis, which means we would have to restructure how the external APIs are used, which is all part of an iterative design.

4 Economic Considerations (LO23)

Determining if there is a market for Flick Picker at a large scale requires deeper market analysis. However, we can use a naive approach to determine if the application is helpful by collecting feedback and seeing the usage of the application itself. For example, viewing the analytics that Firebase provides, we can see the application is being used for little outreach the application has. In general, any application that combines lists of shows to watch will be used now and then,

just when someone is looking for something new to watch, and it is common to talk about shows someone is watching within a friend group. From just human nature and analytic data, there is a market for Flick Picker.

Marketing an application is more straightforward than physical products, as potential users can immediately navigate the website, create an account, and start using it. We must get eyes from advertisements and social media outreach, generating website traffic. Flick Picker would collaborate directly with show providers and show an ad in the pre-roll to get users to find more shows and, ideally, end up back on the website they were watching the original show. This marketing method also targets the correct demographics since we need people interested in discovering new shows. Moreover, 7eam would not charge anything for the product, nor would we need to sell any to make money. It would instead need to host non-intrusive and safe advertisements from vendors 7eam deems appropriate to monetize the application.

As the application is an open-source project, there are many routes for attracting users, a few being discussed above. More local outreach for the application is also essential. Word-of-mouth and getting the application into friend groups can cause huge cascading effects. Ideally, every group member makes an account, then shares it with their other friend groups, which repeats forever. At this point, potential users would be hard to estimate as more outreach and word-of-mouth are needed. Our friend group has ten users, each with other friends of two to ten people, with little overlap. So, the scale at which users come into using the application could grow incredibly large, even if there are only twenty-two active users in the database currently.

5 Reflection on Project Management (LO24)

Flick Picker used various different technologies together for project management, GitHub for code and code reviews, Jira for issue tracking, and Discord for the majority of the communication taken place, alongside a Google Calendar to keep track of dates and deadlines for the project.

5.1 How Does Your Project Management Compare to Your Development Plan

The team meeting plan, team communication plan, team member roles, and workflow plan were all followed from the beginning of the year. Nearly all the major technology 7eam initially discussed was used, with a few libraries added for the client and server, but those are relatively minor while looking at the big picture. There was a plan to implement dependency checking through Snyk or an equivalent, but it has yet to be implemented due to time constraints and other deadlines being more critical. Synk would have automatically flagged libraries used in Flick Picker if they have security vulnerabilities. Since there

were not too many dependencies being used, the urgency of Synk was reduced as we could check if the current library versions have security vulnerabilities manually.

5.2 What Went Well?

7eam's plans for team meetings, communication, and dividing roles since the start went well. Following the capstone deadlines made planning around them easier without explicitly stating they would be followed, as it was only natural. Everyone on the team hopped on and completed their fair share of all the documentation and the code. Splitting team member roles into specific areas allowed us to learn the ins and outs of the section we assigned ourselves efficiently. The minor drawback is that the client developers do not know how the server works, and vice versa. It is only a minor drawback, as this is how the industry is expected to function. It takes work to be an expert at everything, which we would have needed more time for throughout the year.

Most of the workflow plan also went well. Explicitly creating a protected branch and forcing two code reviews on a pull request from the developers ensured high-quality code was merged into "develop" and no redundant code was pushed up. Forcing a pull request description was also great for quickly getting a sense of the changes being requested. Finally, pinging on Discord automatically when a pull request is opened in the repository also allowed every developer to be up to date with the application's state.

Finally, all the technology mentioned was used efficiently as well as linting, however there might be room for improvement in the external APIs, further discussed below.

5.3 What Went Wrong?

Compared to what went right, only a few things went wrong regarding process and technology. In the workflow, we said we would use Jira for issue tracking, an incredibly effective website for large-scale plans. It could have been clearer for Flick Picker 's use case. Since the development is linear initially, the need for Jira drops and everything follows a clear pattern. If there had been more horizontal development in various parts of the application, Jira would have been the correct route. Using GitHub issue tracking to be fully organized would have been better.

Going back to the vertical development, the workflow plan for the pipeline was also rather lacking, there was just no tests or anything initially to ensure a green pipeline. This is both due to the structure of the capstone course and how development was required on the application, tests were written at the end, so the pipeline was always going to be green. However, this plan was important to have as further development can get verified through the pipeline.

Finally, from the assumption 7eam made regarding the information required for the shows, if that assumption is incorrect based off the user feedback then there may have been better external APIs to use in place of the ones currently being used.

5.4 What Would you Do Differently Next Time?

Overall, the project went smoothly, and there is little we could have used differently regarding the process and technology tools. Using GitHub for issue tracking, creating tests earlier, and preliminary research on what users would like would have resolved all the abovementioned issues. This is the most significant takeaway; 7eam should have involved users since the beginning. With the answers provided by those initial alpha users, there could have been a smoother transition between the development and release of the application. So, including users from the start would have been the most important for any application of this kind.