Team 31
Amrita Vadhera, Amit Palkovic, Jean Paul Torre, Sajid Raihman, Thien Nguyen
April 18, 2018
CS 4500

Final Report

An overview of the problem

The expected client for this project is a stealth startup. Specifically, this is a start-up that avoids public attention almost completely in order to hide information from competitors. Employees will have to sign an NDA with the company and will be expected full confidently. Being a stealth startup will also lessen the pressure the project receives from the general public and investors, allowing the developers to create the project exactly the way intended. The client is internal to the organization, and basically represents the people from the organization who want this application to be created.

The project managers came to a conclusion that popular currently used platforms like Rotten Tomatoes and IMDB are too general, and don't serve the users as well as they could. The reviews presented on those platforms are too wide spread and impersonal which cause them to not be as meaningful or useful to the users. The idea behind this startup is that if the recommendation comes from someone that is actually connected to the users - whether it be a direct friend, family member, or a verified account - the users would be more likely to take that review seriously. These recommendations would be more meaningful when the people that are a part of person's actual network will contribute their thoughts in a public, easily-accessible way.

The client's goal is specifically to make it easier for people to get a more specific opinion on what people who are close to them enjoy watching. In addition to that, the webapp created will allow users to take a peek at what celebrities or athletes watch on a daily basis (and what they recommend watching) which is a perspective users are clearly already looking for (e.g. stories of celebrities on Snapchat/Instagram, and being able to follow celebrities on Twitter). Therefore, it seems to be the case that the client is seeking an interface that allows them to easily search for and click on a person to see their list of content. The client's overall goal here is to connect people in a way that currently is not being explored.

Although applications like Netflix collect data on what people's friends are watching, which contributes to their recommendations, this information is not public. In addition, websites like IMDB and Rotten Tomatoes do a good job of giving a general rating from critics and the general public. However, people do not feel a connection to either of those sources other than the fact that they provide data from reputable critics. The client hopes to enter a new niche quickly, and well enough that Netflix cannot simply add a new feature and remove them from the market. By being able to easily search a person's interests in this way, they hope that this product can bring the world together. Additionally, this project refers to the fact that those using the application should be able to view ratings in other countries besides the one they reside in. This

idea serves people so they can look at additional, international perspectives and by that to appreciate what other countries are enjoying.

Moreover, the more data that they have on what type of content people consume in groups, the more they can monetize it. By aggregating data about how people of specific demographics and zipcodes are using their data, the client hopes to sell this data for a profit. This is part of the monetization for the application, in addition to advertisements.

As a result, what we have created is a web-app that integrates IMDB/Netflix with the idea of social networking (Facebook, Twitter, etc). We are making this web-app available for two types of users- customers and admins. Admins have more options than users do. Some of these premium options include adding and removing users from a group. All customers of our app can log in, rate movies and discuss these ratings with their friends. Non-customers will have limited views and the option to sign up to the website.

An overview of the results

We have successfully created a web-app that integrates IMDB/Netflix with the idea of social networking (Facebook, Twitter, etc). We have made this web-app available for two types of users- customers and admins. Admins have more options than users do. Some of these premium options include adding and removing users from a group. All customers of our app can log in, rate movies and discuss these ratings with their friends. Non-customers will have limited views and the option to sign up to the website. Specifics of these are listed in grave detail in the use cases document.

Using statistical data, we can support these claims of completions. For example, according to Sonarqube, we have 95% coverage with 115 unit tests. We have A grade for both maintainability and security and B grade for reliability.

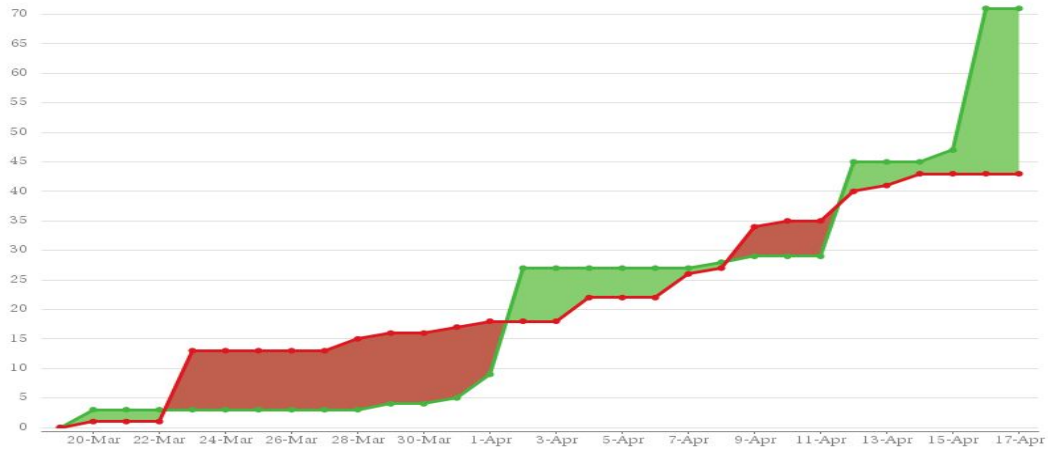
Below is an image also of the Jira report:

Search:

Status	Highest	High	Medium	Low	Lowest
OPEN	1	0	4	0	0
CLOSED	18	12	75	6	1
IN DEVELOPMENT	0	0	1	0	0
Status	Highest	High	Medium	Low	Lowest

Furthermore, the below chart generated via Jira shows the number of issues created vs. the number of issues resolved in the last 30 days. We believe this shows a good representation of our groups progression throughout the semester. There were time periods when our team went

through stretches that weren't implemented well followed by stretches that were implemented well, while all teammates were able to put in a fair amount of work into the project.



The Team's Development Process

Our team development process changed throughout the semester multiple times. Originally our team development process followed these steps:

- List item that have to be worked on by each sprint. What has to be done is defined at a high functional level (example: Build the search feature for the website)
- Weekly meetings
- "Daily scrum" via slack
- Commit / merge our individual branches (git) when team member feels like feature is complete
- No real dev rules, no real code review
- No test written

But following these steps, we had encountered some issues:

- Using the information from the "daily scrum" was hard because not everyone were developing at the same time/day
- Time at team meetings wasn't always spent productively
- Often miscommunication between teammates on the nature of meetings and whether they are necessary or not
- Code quality issue: discovering someone else's code is sometime tough (inline, variable+function+class names, spaces, comments..)
- Conflicts - changes in already existing classes (impact on someone else work)

- Responsibility of each task was sometimes unclear to some team members: after getting someone else code
- Team members were having problems communicating and scheduling time to meet with other team members

After facing those problems, we, as team, have decided to change our development process.

- We enforced strict code reviews - no such thing of just commiting your own branch
- We did not allow committing code that didn't have tests for it.
- We optimized our meetings - used the time more efficiently, with a clear head target to each meeting
- Team members were more available to respond to questions about their code - whether in person or online
- We decided to hold each individual accountable to their own code
- Merging earlier and often
- Have documentation to any code written

Even though we improved according our development problems throughout the semester, we still had some things we could improve upon. For example: were having issues doing the daily scrum routine, and still encountered some issues communicating at times to schedule meetings or when assigning tasks. I think a lot of the issues could have been solved by having some rotations of roles in our group like you would have in a regular company - we didn't have a person being a manager, a scrum master, a developer, Q&A etc. I think having a more defined role for each individual on the team on each sprint would have helped us be more organized and communicate better. Having roles could have also made it easier for us to hold each other accountable for each one's individual work.

A brief retrospective of the project

Overall, we learned a lot throughout the semester in this course. We believe that there are not many classes at Northeastern that put an emphasis on the process behind coding rather than the code itself. This class has forced us take a step back from only coding new features to our website and to look at how we are writing that code, and how we are interacting with our team members. This course taught us that in the real world a team would be combined of different people who would normally have different skills, and would be able to contribute in different ways.

The expectations for the sprint reviews for this course were very clearly aimed on seeing how we worked together as a group, how we communicated, and less about what specific

features we implemented. The TA reviewing our product also emphasized the importance of testing our code properly, which became a very important part of our development process.

All of us learned a lot about how to handle working with people who are different than who we were used to work with in the past, and we believe that that was a very important lesson. This class really took us out of our comfort zone because we got so much freedom (relatively) - we had to learn how to leverage all these new languages, libraries, and integration tools and really think for ourselves how we want our product to look like at the end, and figure out how to get there on our own. We felt like the class lectures for this course were more like a general guideline review of what the overall product should be similar to and which tools we should use, but we got a lot of freedom to just build in the things we believed would be most useful for the users.

This class also taught us how to handle problems in a way that we have never experienced before. In most coding classes you only have one partner, so everything is pretty easy to resolve, but software development we had to manage 4 other people. With that being said, it might be the case that the selection process could be more thought through for the groups, or picked by some categories and not in such a random way.

We think that some things would have been enforced earlier in the course to encourage good coding habits from the beginning. For example, test coverage was just a stretch for awhile, until it wasn't (for sprint 4) which caused a lot of backlogged untested code that could not have been committed and that really set us back. We also wish that there was a better way for the staff to monitor jenkins, because our team struggled for most of the semester with getting branches to pass jenkins, and to not have jenkins fail. I think having jenkins run properly earlier in the semester would have saved us a lot of time and internal arguing.

We also believe that sometimes the schedule for the deliverable due for this course was a little bit rushed, and we couldn't get all that we wanted done in time in combination with our personal homework assignments and the exams for this course (not to mention our other course load).

One last note that we have as a team for this course is that quicker feedback on the side of the faculty would could be helpful for teams in improving from sprint to sprint and by that achieving a better product.