Northeastern University
CS 4500 – Software Development

Spoiled Tomatillos Final Report

Yiwen Dong, Diana Regalbuto, Fiona Tran,
Jiameng Wang, Xi Zhang
Team 24

April 2018

Overview of the Problem

❖ Our Client

Our software development team has been contracted by a stealth startup (who we will call our client) with a new product called “Spoiled Tomatillos”. Spoiled Tomatillos will bring aspects of online movie databases like IMDB and Amazon into the social media space, creating a new platform with the potential for seamless monetization and expansion into new markets. Their team has big ideas but no technical expertise, so they have asked us to build a Phase 1 prototype of their product that will be ready to show to investors by April 2018. Our team’s prototype will prove the viability of this concept and allow our client to market Spoiled Tomatillos to potential buyers and collaborators.

❖ What is our client trying to achieve?

The problem our client company wants to solve is how to show movie watchers good, individualized recommendations for films they have not seen yet and aren’t sure if they’ll enjoy. Our clients believe they can solve this problem through the power of social networks, which allow people to connect with friends and like-minded strangers to talk about shared interests and collaborate in new ways. Our clients have conceptualized their product “Spoiled Tomatillos” as a hybrid of an online movie database and a social network. Friends should be able to share film ratings and recommendations with each other, and make plans to see popular films that are playing locally. In order to give users recommendations for films their friends haven’t seen, our clients want to use machine learning to create a predictive recommendation system based on user-to-user collaborative filtering.

The basic service of Spoiled Tomatillos will be a searchable collection of movies with information like plot, cast, awards, showtimes, like the info on sites like IMDB. Like IMDB, users can create free accounts and and assign ratings to films they’ve seen. From the social perspective, Spoiled Tomatillos needs to let users find their friends on the platform and suggest movies to each other. Finally, the system will produce personalized movie recommendations based on average critic ratings, average user ratings on the site, and User-User collaborative filtering.

❖ Why is it important for us to solve this problem?

By displaying movies that are available from different streaming platforms on one review site, and showing users personalized recommendations, our product will offer an enticing combination of features that has never been seen before.

Overview of the Result

❖ Quantity

Our software development team has been able to complete the basis and general functionality of the Spoiled Tomatillos prototype. We have completed 69 Jira tickets over six sprints, as well as 34 Jira tickets for user stories and system setup before the first sprint. These tickets cover system setup, user registration, movie information, movie reviews, movie recommendations, groups, and friends. With this functionality in place, the Spoiled Tomatillos product supports the basic needs of the client and is in a good place to be further developed with more features.

❖ Quality

We assure that the product has been developed in high quality. We have achieved 99.1% code coverage through 96 unit tests. The final program has 0 bugs and 0 vulnerabilities as determined by SonarQube. Wherever possible, the program has been generalized so that new features can be added on the back-end without disrupting the quality of the product.

Overview of the Team's Development Process

❖ General Process

Our team used an Agile process to develop the project. This Agile process involved multiple sprints, sprint reviews, and stand ups. The sprints ran bi-weekly showing the quick turnaround that the an Agile process requires. At the beginning of each sprint, the team met to plan according to the sprint expectations and each of our availabilities for the upcoming weeks. This planning and allocation was recorded and documented in JIRA. When creating the use cases and tasks for the sprint, we made sure to organize them by the larger functionalities and the front or back end. We wanted to ensure that tasks were contributing to a broader use case and that we were covering all sides of the build. Using JIRA allowed us to point out dependencies and observe our progress throughout the sprint. We were able to create automated progress reports that would depict our progress throughout the whole project from sprint to sprint. We remained in contact throughout the sprint using Slack. Using Slack allowed us to hold daily stand-ups to communicate our progress and issues. The application was also integrated with GitHub to send pull request notifications which allowed quick turnarounds for merging into master.

❖ What Worked

We used Jira effectively to organize our user stories, tasks and bugs for each sprint. At the end of each sprint, we were able to reflect on the tasks that we accomplished as well as tasks that were returned to the backlog. Our use of Slack was also successful because it allowed team members to communicate with each other and conduct daily stand-ups which helped to resolve problems early in development. Once we integrated GitHub with Slack, the ability to view pull requests in their own Slack channel made it easy to see open requests and talk to the team about them in the same program. The team also used GitHub successfully for version control, and we worked together to develop a consistent system of making branches for new features and making small, incremental merges so that the team could always access the most current stable build of the app.

The team also had to include system setup and learning as part of the development process. We did not have experience in web development before beginning work on Spoiled Tomatillos, so several group members taught themselves React.js in order to build a working front-end for the app. For the back-end, we were able to leverage our previous lab work and set up the Spring MVC infrastructure with Hibernate to store our data. This learning experience was valuable and generally successful, although it did slow us down at the beginning of the project.

❖ What Did Not Work

We ran out of free space on our Jenkins instance several times over the course of development. This required lengthy system restarts and in one case a full reboot of the EC2 instance. At other times, Jenkins was very slow to build and test our project with Maven. This slowed down our overall development process because we were unable to merge our code into the master branch until the Jenkins checks were complete. We found ourselves having to compromise between making frequent merge requests and waiting to merge until more progress had been made because Jenkins was such a bottleneck. If we could go back and improve this process, we would have allocated a larger EC2 instance and Java heap size to Jenkins and customized the repository scanning settings so that Jenkins built each branch less often and more quickly.

Retrospective

❖ Best Experiences

It is a great experience to be able to create a collaborative project and see it come to life. There's no greater joy than to see our creation and hard work come to life. To communicate as a group and to manage our projects is something no other classes at Northeastern had given us. We enjoyed being able to use some of our experiences from our previous co-ops and gaining more skill for our future co-op and/or careers. Setting up Jenkins, managing Jira, complying with Sonarqube have brought valuable experience we are sure everyone can appreciate. Being able to see our apps come alive on AWS is just the cherry on top.

❖ Worst Experiences

There have been a lot of difficulties trying to finish this project. This class is set up to closely mimic a real working environment, and while that is great, it caused many difficulties in planning and working. Most of us take 4 classes during the semester and this is just one of those 4 classes, thus it becomes difficult to commit time in order to meet with members of the group with the limited time that we have and the limited space there is on campus. There are often very few places to get together where it is both quiet enough to work but accommodating enough for us to talk freely.

Beside the logistics problem, there were also technical ones. This is a software development class, but yet there's been more time spent on learning frameworks such as React and Hibernate. This class felt less focused as a result. The class lectures have done a great job preparing us for the Agile workflow and process of writing good code, but yet we were largely unable to implement these best practices due to deadlines getting in the way.

Finally, the systems such as Jira and SonarQube were sometimes very frustrating. Setting up some of these systems took up days and they tended to break at the worst of times. More support would certainly be appreciated because troubleshooting these systems took a huge amount of time. Jenkins especially was breaking under the weight our projects due to the sheer size taking up most of the spaces allocated by default to AWS instance and was taking a long time to finish running.

❖ What We Learned

Agile process in software development is very different from our previous experiences with projects from course work. At the beginning of the course, we had the impressions that we would be given detailed instructions on how to proceed with the process. However, we realized that in the industry, clients don't know what features they want until they see it. Our learning process started from understanding our users and requirement elicitation. Continuous communication and integration is also an important part of the project because client requirements could change over time. The feedback from bi-weekly sprint reviews not only helps us to constantly communicate with clients about what they want to see and what our progress is, which is the essence of agile process, but also being professional in the workplace. Another valuable experience we gained is flexibility in the project and teamwork. A group project is never an individual work. It is important to find out what works the best for each individual, but also for everyone in the team. At the beginning of the project, we simply put tasks on Jira and did what came to mind first. However, as we moved forward, we realized that being insightful of what requirements may come later in the project and plan ahead of time accordingly is beneficial to both our teamwork and the project. Some groups work better when the team work on front and back end separately; while the other groups prefer to divide system feature to individuals.

❖ What should Change in the Course

In general, this course should talk more about current solutions to software development. The focus of now is the history of development model, and design. I agree with the importance of these concepts, however, guiding students to grow in their study path is also significant. Sharing more coding experience can help students success in the project. Additionally, we felt that there was a lack of support during the later stages of the project. Once the lectures ended and the course shifted to focus on the project, it was difficult for our group to attend office hours to get help. It would have made more sense for the lecture time to be devoted to group work. However, for this to be effective the group members would all have to attend the same lecture. Organizing the groups by lecture as well as by lab section would have made it easier for us to meet since our schedules would be more aligned.