Spoiled Tomatillos - Final Report

TEAM-51**:** Ian Anderson, Patrick Milne, Maddie Seraphin, Emily Trinh, Tina Wu

Overview of the Problem:

Our client is a stealth startup that is interested in bridging the gap between Netflix/Amazon and Facebook by creating a social recommendation system for movies. The company has an in depth go to market business plan, solid funding, and backed by multiple fortune 500 companies. The company has a robust pricing model focused on selling ads, providing links to movie purchasing and movie theater sites, and selling user data. This is important to the client as they feel there is a large untapped market for a social site that recommends movies and TV shows to users and provides easy access to methods and discounts for purchasing movie tickets, downloads, or streaming services. This service could be especially helpful for helping users decide which streaming service to subscribe to based on the one that has the most movies that the system recommends to the user. The client would like to be first to market with this app, gain significant market share, and then be acquired by 2019.

Currently, the client is operating externally of any organization, though it does have multiple anonymous partners. To our knowledge the company has no other product or business model.

The system the client is looking for allows users to sign up and have access to an IMDB like database at their fingertips. Users can rate movies that they have seen, which automatically updates the average ratings of the movies and contributes to future movie recommendations. Users will also be able to search and add friends to their friends list. Adding friends will allow for a social aspect of the site and improve the movie recommendations they will receive. What's more, users can send recommendations to their friends of movies they should plan to see together or just as a friendly recommendation.

The client is looking for a movie recommendation model that provides recommendations based on critic reviews, the average rating on the site, and based on the ratings of other people who have rated the same movies the most similar to the person receiving the recommendation.

Our target market is social media users who love movies and TV shows. We believe this is a large and growing market as streaming services become more popular and as more TV shows are created over time. Many people have resorted to solely using streaming services instead of buying cable. Thus, we can capitalize on this growth and have our project provide a social media element to streaming films. On regular movies sites now, such as IMDB and Rotten Tomatoes, users can only read about the movies that they search. However, our project will enable end users to bookmark movies for later, recommend movies to friends, and provide links/discounts to purchasing films.

Overview of the Result:
The final Spoiled Tomatillos application completed all critical use cases and needs that were specified by the client. In terms of user management, the user is able to create a Spoiled Tomatillos account with a unique email address that is not already associated with an account. Further, the user can use their account credentials to login and logout of the system. When logged in to the application, a user is able to: search for a movie modeled from TMDB, rate a movie, review a movie, add a friend, and recommend a movie to a friend. Once a user has rated at least one movie, the system will recommend movies to that user based on the genre of the movies that user liked. Users can see these recommendations on the homepage, as well as movies that are popularly rated on Spoiled Tomatillos. Users can navigate to their profile to see notifications that they have been sent, such as friend requests that need to be approved or received movie recommendations. Admin users have the ability to elevate another end user's privileges to become an Admin and can remove users from the database.

Backlog Statistics:

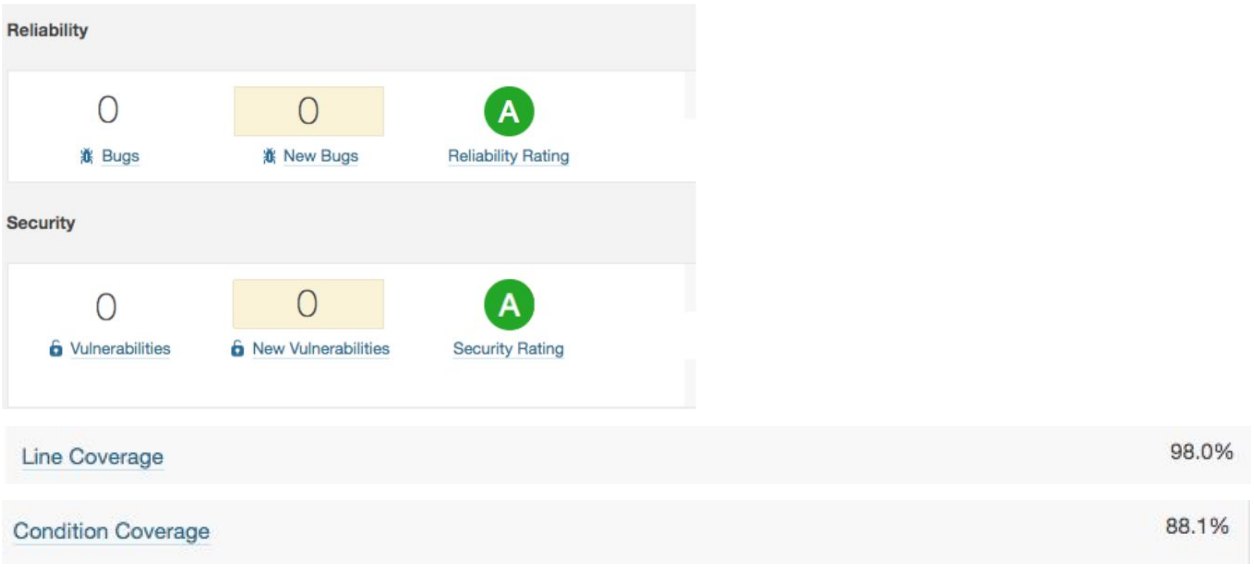
	Low	Medium	High	TOTAL
Open	0	0	0	**0**
In Development	0	2	0	**2**
Closed	2	75	14	**91**
TOTAL:	**2**	**77**	**14**	

The team was able to complete 91 tasks and subtasks that were created on Jira. The two tasks that are labeled as "In Development" are associated with the front end integration of session management. JWT was fully integrated on the backend of the system. However, the authorization header would not appear properly on the front end. After consulting a plethora of online resources, all resources suggested that we alter the settings of the server to allow CORS. However, we are not in charge of the server and this solution was not feasible. We worked relentlessly with our TA, Raghu, over two sprints to integrate session management and were unable to do so. However, all other critical use cases and demands of the client were met.

	# of Completed Items
Sprint 1	12
Sprint 2	15
Sprint 3	18
Sprint 4	22
Sprint 5	25

There was also a constant increase in the amount sprint tasks/subtasks that were completed for each sprint. This shows how the team developed stronger time management and collaboration skills over time.

Quality Claims:



Spoiled Tomatillos follows all best practices in terms of reliability, security, and testing. There are 0 bugs and 0 vulnerabilities in the code, which asserts an “A” rating for reliability and security. In addition, there is 98% line coverage and 88.1% condition coverage from the 247 tests that were created for the code. Lastly, all code is commented with javadoc to ensure that other developers and the client can thoroughly understand the system.

Team’s Development Process:

When we began this project, knowing that none of us really had any web development experience, we were quite intimidated. We knew that we had to do everything we could to make things as painless as possible. Based upon our past experiences/skillsets, we decided upon roles for each member of the team at the beginning of the project to make the development process as easy as possible. The roles are as follows:

Ian and Maddie – Front-end (HTML, CSS, ReactJS code)
Pat – Back-end (Database functionality)
Emily and Tina – Back-end (DevOps and Testing)

These roles remained fairly constant throughout the entire project, although we were all willing to step outside of these defined roles if needed for the project to succeed. We all were able to build our skills in this role over the course of the semester. Ian and Maddie completely taught themselves ReactJS for this project, and the back-end members of the team learned how to create and maintain a REST API to be used by any front-end implementation.

At the beginning of each sprint we would assign a Scrum leader who would facilitate the assignment of tasks to all the members of the team at our initial sprint meeting. This person would hold everybody on the team accountable for getting their tasks completed, and check in on the status of each team member.

Throughout each sprint, we would update the others on our team daily via our group iMessage and make sure everybody else knew what we were working on. We would also let anybody know if there were any blockers from other team members that were preventing us from completing our tasks.

As far as technology goes, we made effective use of JIRA to track which tasks we had completed and which tasks were yet to be completed, transitioning these tasks via smart commits. We were also able to instantly deploy our master codebase to our project instance via Jenkins (although we did not have this fully set up until later in the project than we would have liked.)

What Worked:

What we lacked in web development experience, we made up for with team chemistry and a willingness to learn. We were in constant communication with each other at the start of each sprint to ensure that each of us understood what role we were playing in the coming weeks. From there, the front-end team members and back-end team members worked together to complete all of their tasks, with the scrum leader checking in with everyone individually to make sure no one felt too overwhelmed. We were able to tackle the many problems we faced by working together to find a solution. When we faced challenges we had never before seen, we got help from our TAs and online resources to point us in the right direction. The key to our successes was our flexibility and communication among teammates.

What Didn't Work:

In the few instances when our communication was lacking, our team faced problems. If some team members had updated the master branch but forgot to tell the rest of the team to pull, that could sometimes create extra work with merge conflicts later on. Sometimes the back-end team would change the name of a function and not think to alert the front-end team, which would cause problems in our product until we worked together to sort out the differences. Sometimes the front-end team would run into problems that broke the application without alerting the back-end team, who would be confused when suddenly the site didn't run. When our team failed to operate as a cohesive unit, our application failed as well.

Retrospective of Project:

Some of the best parts of this project were learning about the different aspects of building a web application, learning how to capitalize every team member's individual strengths, and working with very talented and smart people on the team. It was both satisfying and frustrating during the entire development process. Our team is very inexperienced in the realms of web development, full stack development, and DevOps. For every sprint, we learned about at least one integral part of web development - whether it be security with path variables, continuous integration with Jenkins, or full-coverage testing with SonarQube. Our team definitely learned a lot of skills throughout the semester and learned how to work with each other seamlessly. Every member has strengths and weaknesses and we all contributed equally towards the success of the project. We communicated thoroughly when a team member is blocked on a certain feature and we are all very present and supportive when a team member needs additional help for a specific story.

Some of the worst parts of this project revolves around the vague requirements and the loose interpretation of what constitutes as a use case. For our initial set of use cases, we came up with very detailed use cases; however, during the sprints we were told that certain use cases only counted as one but were not given prior notice. There were a lot of skills this project entails but most were not taught during lectures so our team had to spend a lot of time outside of class with our TA to learn and adjust. With the huge learning curve our team has to endure, it was extremely stressful and frustrating trying to meet the sprint expectations on time.

Throughout the course of the project, the team all learned a lot in terms of best software development practices, working with a team of different developers, and managing priorities and deadlines. We learned about the iterative testing process - releasing new simple features, testing the code thoroughly to make sure it is both secure and functional, then build off those simple features into more advanced functionalities. We understood early on to split up specific tasks based on everyone's expertise and interests so it was easy to ask for help on certain topics because we all know what everyone is working on. We meet at least twice a week to keep track of everyone's progress and check in on the rest of the team in terms of deadlines and responsibilities. As a result, we all learned a lot at the end of the project whether that be tangible technology and programming skills or improving our teamwork and communication.

Course Reflection:

It has been reiterated at multiple points in this project report and during every sprint that our team lacked a lot of the basic necessary skills to adequately complete many of the requirements on time. We had to meet with our TA for hours every week to learn about how to use Spring, how to create endpoints and pull API from an external movie database, how to set up Jenkins for deploying, how to test our endpoints without modifying our database, and how to write a REST controller. We struggled so much juggling responsibilities in completing the requirements of the sprint and learning and debugging technology/software-specific problems.

One of the biggest things we hope for the course to improve on is to be more cognizant of students who may not have web development experience prior to taking this course. The only prerequisite of this course is Object-Oriented Design (CS3500) - OOD did not teach about REST controllers, front-end development, database design, DevOps, or full-stack development. There was a lot of assumption starting this project that students have some experience in web development so the lectures were all very high-level - which was understandable because technologies and softwares vary in every company and every project. However, my team and perhaps other teams who were as inexperienced as us, would have benefited a lot with a more guided approach for this project. In regards to the software development life cycle and continuous integration, we practiced the SDLC and agile scrum development practice with each sprint which was an excellent learning strategy but we feel that a more detailed lecture on continuous integration would be extremely helpful as well to learn about the importance of automation.