

Overview of the Problem

The objective of this project was to create a semi social media platform where users are able to search for movies, save what they like, and share what they want with other users on the platform. Our client believes the current movie recommendation systems are inadequate in providing users with the complete experience. The issue with existing platforms like Rotten Tomatoes or IMDB is that they lack the ability for users interact each other. Our team was thus assigned to fill this gap. We focused on integrating the social media aspect into a movie database platform. As a result, the product shares similar functionalities as existing platforms, such as the ability to find detailed information about movies, rating movies, and leaving reviews. On top of the social aspect, we wanted to create a more accurate recommendation system by taking advantage of the information extracted from user activities. We hope that the more users and traffic the application generates, the more accurate our recommendations will become.

Our primary goal to was to create a well designed system. We also kept in mind ways to make the product profitable. We want to maintain our integrity regarding user data and making sure that privacy of our users are not violated in anyway. Our stream of income will be from partnerships with third party vendors and sponsors. The challenge is to extend our product so that users will be able to purchase movie tickets (and hopefully in the future, books, music, etc) from third party vendors with us getting a cut of the profit.

Overview of the Result

At the end of the semester, our team completed a total of 21 use cases, which covered most of the core functionalities that were absolutely necessary for the application. As shown in the picture below, most of our tasks were completed, missing only minor future implementations. These include details such as searching for movie tickets from nearby theatres and having

group forums for users to discuss topics in. If given more time, we would continue working on these features as the next steps for the development of this application. Due to some early mistakes, we still have test review story that we did not finish in time. However, not only does our app provide users basic functionalities like searching for a movie, adding friends, recommending movies to friends and more, users also receive extensive recommendations and have can be granted admin privileges which gives access to functionalities such as banning users and removing reviews from movie pages. We dedicated a good amount of time on the look and feel of our application during the development process. As a result the UX/UI of the website is decently fluid and intuitive for a new user.

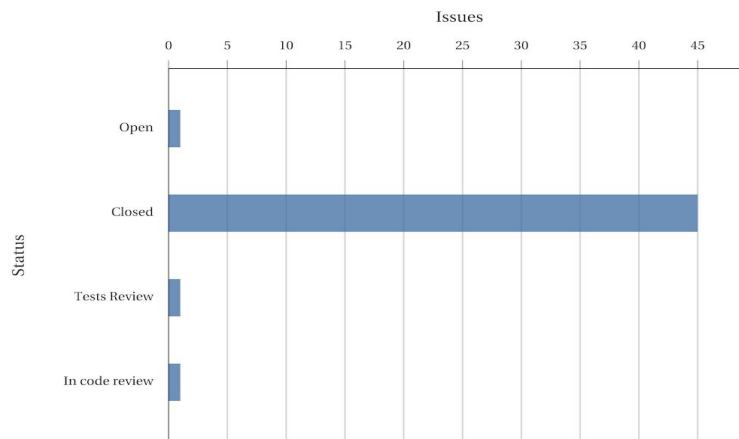
Team 47 Completion Report

Summary

This report shows data from the project team-47-spring18, broken down by Status.

Total Issues	Average Issues	Max Issues	Min Issues
48	12.00	45	1




Results



Results Data

Status	# of Issues
OPEN	1
CLOSED	45
TESTS REVIEW	1
Status	# of Issues

LocalSQLConnectServiceTest (Apr 13, 2018 3:09:10 PM)

Element	Coverage	Covered Instructions
▼  cs4500-spring2018-team47	 28.2 %	2,653
▼  src/main/java	 20.7 %	1,701
▼  edu.northeastern.cs4500.model.services	 30.2 %	1,522

Since we did not follow Test Driven Development from the start, we were very delayed in our test developments. We drastically improved our habits towards Sprint 4 and 5, but by that point we were simply unable to finish the tests on time. Regardless, our tests covered most of the API and database behavior, so we could have some guarantees about the data we are receiving from both TheMovieDB and our Local Database.

Overview of The Development Process

Our team aimed to follow the best practices we learned in class throughout the duration of the project. We've had both successes and failures up to the delivery of our final product. One of our biggest successes is that as a team we and maintained effective communication after struggling to do so in the beginning. In the early development process, we were working individually on our assigned tasks and met only before the sprint review. We realized that it was difficult to come together right before and piece things together. There were confusions about dependent functionalities and building upon each other's code. This was apparent after going through receiving feedback from our first sprint review and realized that not only did we forgot to push our UML diagram to our repository, none of the members were aware of this issue. We decided that in order to ensure workflow we had to spend more time working in the same place where any questions or confusions could be clarified immediately. We met and worked as a group more frequently, leading to better communication and team cohesion and as a result

higher sprint grades. We used slack to substitute daily stand-ups and although we sometimes default to other tools out of habit, each member was expected to provide the the team with accomplishments, goals, and questions if any. Eventually, our ability to be very transparent and available for communication became one of our biggest strong points in this entire project, as at least one person was always available to help whenever help was needed. One of the struggles that we've had is getting our test coverage up to what we want because of some internal as well as external factors. One of them being having troubles getting sonarqube to recognize our tests. Sonarqube also required us to put all of our code that involves executing sql statements in try catch blocks. The catch blocks were unreachable from our tests and therefore we could not get 100% branch coverage for our tests. We've made some poor design choices by having more logic than we'd like in our controllers that were difficult to test. If we were to do it again, we would abstract all of the logic in controllers to our services. Overall, we struggled to actually implement test driven development during our process due to our unfamiliarity with the technique coupled with the fast pace of the class, as well as our desire to have working implementations of use cases. It was only towards the end where we understood the importance of developing around the tests rather than writing tests after we code.

We learned very quickly what everyone's strength and weaknesses were, and quickly applied them to whatever was needed. As a result, people were often tasked with what they were most comfortable with. However, that did not stop the team from taking on uncomfortable tasks, as a goal for every one of us is to fully understand every aspect of our codebase, as well as learn new technologies along the way. The improved communication from Sprint 1 has helped a lot in transferring knowledge from one to another.

A Brief Retrospective

Overall, our team had a good experience working on the project. We took away lessons on navigating the software development cycle, working with changing requirements, and overcoming challenges as a team. We learned new technologies while gaining deeper understanding of ones that we know. Almost all of us learned the stack we used from scratch and are now familiar enough with it to know that we could work on another project using the same stack. Being able to simulate the entire software development process and build a product that we can ship was also great learning experience for all of us. There are a couple of things that we think could change about the course. We spent a good amount of time trying to follow instructions. Clearer guidelines and instructions would have helped us utilize our time more effectively. One of our members in charge of devOps was charged over \$600 attempting a reach goal for a sprint because we were not warned of AWS's billing policies(this is also AWS's fault for not being clear). We noticed spelling mistakes in most of the assignment instructions and that could be fixed easily. Some of the assignments such as the ones on deployment involving jenkins and AWS were rushed and we found it difficult to retain the knowledge since all we did was follow given instructions. For us, Sonarqube made it extremely difficult in getting test coverage and did not help us write better code. There were also a lack of instructions on using sonarqube. If given more time, we would have liked to experiment first with different stacks to test out which one would suit our needs the best. We also have other features that we want to implement such as giving users the ability to customize their profile page to provide a more personalized experience. A big takeaway for us is to never let tech debt pile on. We knew not to do it but only after experiencing the effects of our design choices in the beginning we learned how difficult tech debt is to deal with later on.