Spoiled Tomatillos Final Report

cs4500, Team 42, Spring 2018

Arianna Tang, Ben Campbell, John Martin, Lisa Oakley, Peter Battinelli

Problem Overview

The client is an external entity to our organization. They are contracting with our firm to develop a product to bridge the gap between media consumption and social interactions in the online sphere. The client is a startup who came to us to make a “Phase 1 prototype” of this product and convey to potential buyers that this product is feasible, works correctly, and has interest from a user base.

This organization consists of a CEO, Michael Weintraub, and a team of co-founders/CTOs in the form of other professors and teaching assistants. They are the core team of a startup which is motivated by the possibility of acquisition in FY19. They have determined that there is a market and a need for this sort of product, and feel that this is a feasible goal. Over the course of development, our direct contact, Amit Raul, provided extensive insight into the ideas of the organization to shape the development of features for the product. Through bi-weekly demos, our contact provided excellent feedback and acted as a sounding board for potential features and ideas as well as providing productive criticism and solutions.

The interest in the movie-recommendation with social media platform stems from the idea that there is a market for the marriage of media consumption and media sharing. The goal is to create a one-stop-shop for not only watching movies, but forming and filtering social circles based on shared interests in various media. After determining that there are two separate spheres that have yet to be bridged by any large competitor, the client desires to build a platform where individuals can connect and bond over shared interests. Eventually they hope to go as far as to create an experience of virtual camaraderie when users consume and discuss opinions on different forms of media. They are starting with movies but hope to eventually expand into other domains such as books and music.

For this “Phase 1 prototype,” the goal is not to achieve all these initiatives, but to show that, for a concise user group, sharing and connecting over movie preference is desired and feasible. To achieve this, there is an expectation that what we present will be a Minimum Viable Product (MVP). The MVP we are ready to deliver provides a unique user experience combining our proprietary movie discoverability algorithms with social media interaction. On the social front, users are able to discover and follow each other as well as recommend movies to their followers through “prods”. The client is able to experience movie discoverability that includes user-to-user movie recommendations and generated playlists based on a user’s favorite genre alongside a simple movie searching feature. For future development, the MVP is designed with developer portability in mind with a strong test suite, high code quality standards, and proper logging for easy traceability of errors.

The delivery of this MVP is important to the client because they hope to be acquired and feel that this proof of concept will be a first step toward that goal.

Results Overview

We accomplished almost every use case we originally set out to accomplish. We originally scoped out several goals for the system, including:

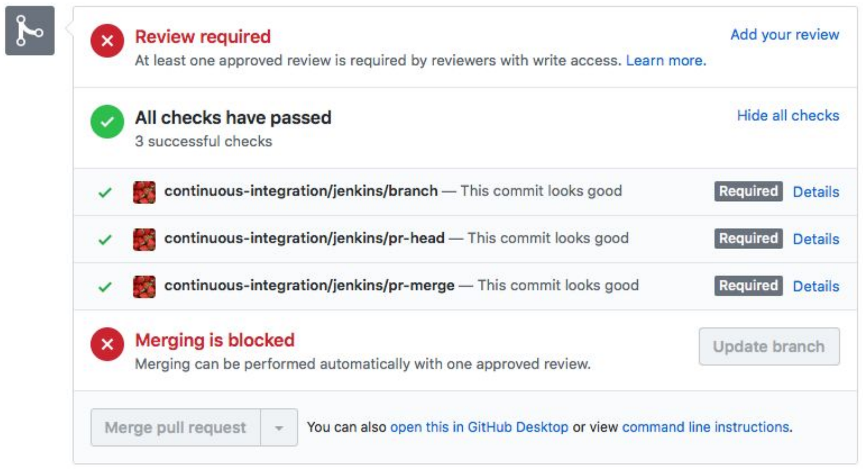
- Users able to sign up for and access accounts
- Users able to search for movies and glean basic information
- Users able to rate and review movies, which will lead to recommendations
- Users able to identify and connect with friends
- Users able to recommend movies to friends
- Users able to be grouped by the system based on connections and interests
- Admin users exist and have more capabilities than end users
- Algorithms exists to smartly suggest movies based on user information

From this, we derived a handful of use cases that served as the basis for our development, including:

1. Sign up
2. Log in
3. Log out
4. Reset password
5. Delete account
6. View profiles
7. Edit profile
8. Identify friends
9. Prod friends (recommend movies)
10. View recommendations
11. Search movies
12. Read movie details
13. Rate and review movies

Of these, we accomplished all but one: password reset. Password reset was decided to be slightly out of scope for the task of creating a system of fundamental features. It also proved to be difficult to implement well without adding an emailing system. We called into question whether account deletion was necessary as well. In light of recent industry events involving using privacy and data rights, as well as the upcoming deployment of the General Data Protection Regulation (GDPR) in Europe, which, among other things, guarantees that European citizens be able to request that applications delete their information upon request. In addition to this list of original use cases, we added system logging feature for administrators. This was added as a convenience for debugging and passing the system off to future developers.

We maintained very high code quality by putting merging restrictions on our github repositories. Unless we met 98% line coverage and 93% condition coverage, the pull requests were blocked. Additionally, peer reviews were required on each pull request with explicit approval from an engineer that was not involved in the code written for the particular pull request.


Review required
At least one approved review is required by reviewers with write access. Learn more.
Add your review

All checks have passed
3 successful checks
Hide all checks

continuous-integration/jenkins/branch — This commit looks good Required Details
continuous-integration/jenkins/pr-head — This commit looks good Required Details
continuous-integration/jenkins/pr-merge — This commit looks good Required Details

Merging is blocked
Merging can be performed automatically with one approved review.
Update branch

Merge pull request
You can also open this in GitHub Desktop or view command line instructions.

Figure 1: A github pull request where code coverage was sufficient, but the merge is blocked awaiting review.

Overall, the product is presented in a complete state, going above originally interpreted scope in some cases. It is presented with safety implementations including password salting and front-end session ids. And it is presented with high levels of testing coverage that guarantee system stability. We are confident that the system can meet its original expectations to serve as a MVP for Spoiled Tomatillos and has been designed in a sufficiently extensible manner.

Development Process

Our team used a lot of the suggested tools given by the teaching staff, which motivated a lot of the ways we stayed organized and communicated. We spoke primarily through slack, weekly labs, class meetings, and a few Saturday development sessions we organized throughout the project. We had a slack channel that was visible to our professors where we posted our standup updates and communicated with our TAs and professors. Because there were too many github integrations in the course slack, we created an additional slack for ourselves, which we ended up using for most of our internal communication.

This group slack facilitated a lot of really good communication. It was there that we balanced our workload, adjusted to unpredictable situations, and checked in with one another to discuss progress that could not be gleaned from Jira or GitHub. We had excellent and open communication. We frequently helped one another with problems and kept up to date on our progress on different parts of our stories.

In the first 3 sprints, we struggled to manage our time correctly, and had a few issues with getting features in by the deadline. We misunderstood a lot of the requirements and struggled to understand what were our priorities. However, we did an excellent job of opening conversation with our TA, learning from our mistakes, and approaching our last two sprints with a mentality of getting things done early and being sure of the work we were producing.

We chose off the bat to use a different stack than the majority of the class. This put us in the position of having to pave our own way when it came to anything provided by the professors. We had a lot more internal expertise in the stack we chose, which helped us to solve problems internally. However, especially for Jenkins and SonarQube, we had to make a lot of substitutions and adjustments to the advice provided to us by the professors. While requiring a fair amount of increased time investment toward the beginning of the course, we found that it allowed for quick, agile development later on in the process.

We focused on using test driven design whenever possible. Working in that manner allowed for good test coverage and reduced bugs. It did sometimes make it difficult for us to add necessary changes to existing functionality, but overall we found it to generate clean, simple code.

Early on we set up a system of team leads which helped us take ownership of different areas of the project. While we all had our hands in every part and were expected to understand how everything worked, this gave us an organizational structure which allowed for us to have a point person to ask about specific questions based on a technology. This facilitated the transfer of knowledge and learning easily through our team.

Project Retrospective

This course has a lot of potential for inspiring students to learn and participate in software development but at times it fell short of its goal. This course presented a lot of information but at times it felt like most of the topics presented would already be known to any computer science student who has gone on co-op. Often, progress of the final project was inhibited by the tedious homework assignments. Besides the issue of students having prior knowledge of the course materia, one big complaint of the project was lack of variation.

We know that in the past, groups were assigned to real companies who needed actual software development work to be done. This seems like it could help with our intrinsic investment. Most of us have worked on real projects, to working on a product that will never see the light of day can feel a little tedious. Secondly, the idea of every group creating the same project really took away from the creative drive that made software development a great 4000 level course. We recognize that finding real clients is not easy and grading unique apps leads to inconsistency so we have recommendations for future semesters. In order to solve the first issue, rather than changing the curriculum we suggest changing the timing of this course from a senior year class to a third year course.

Not only would this alleviate the frustration of relearning information already obtained from co-op it would also help first or second co-oping students better prepare for different work environments. While difficult, we cannot stress enough the value of unique projects. In fact, we all acknowledged that choosing our own stack was one of our favorite parts of the class. For future semesters we suggest having general requirements for the final project rather than having all group create the same site. Additionally, giving students to introduce their peers to unique technologies can be a great opportunity to learn what we already know at an even deeper level.

Additionally, we felt that the most impactful and useful class periods were the ones where we all interacted, sharing our experience, challenges, and solutions. Both for the project, and for our individual experiences, it seems like being able to gain from the knowledge of others would be incredibly useful in this course. If the lectures were modified so that the long slide presentations were replaced with prompts and discussion, we could get more out of the knowledge of our peers, as well as have the chance to ask specific questions to the professors to get a more tailored experience to the backgrounds that we each had.

I think we all agree that we learned a lot over the course of this class and the project, but most of that learning came from other students, other members of our team, and individual conversations with the teaching staff, rather than the lecture slides.