Sprint Review Report Template

CrossFilmz

Sprint Review Report

Sprint 2

Version 1.0

6/29/2020

Anthony Gruber, Avery Peiffer, Samir Sherlaker, Carter Smith

Software Engineers

Prepared for

COE 1530 Section 1010

Summer 2020

University of Pittsburgh

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 6/25/20 | Version 1 | Anthony Gruber, Avery Peiffer, Samir Sherlaker, Carter Smith | First Revision |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Anthony Gruber | Software Eng. | 6/29/2020 |
|  | Avery Peiffer | Software Eng. | 6/29/2020 |
|  | Samir Sherlaker | Software Eng. | 6/29/2020 |
|  | Carter Smith | Software Eng. | 6/29/2020 |

**Table of Contents**

**Revision History ii**

**Document Approval ii**

**1. Introduction 1**

**2. Specific Goals 1**

**3. Analytics 2**

**4. Conclusion 3**

# 1. Introduction

The purpose of this sprint was to begin implementing features for the platform after its initial skeleton was designed in Sprint 1. These features centered around the database operations necessary for the platform’s functionality, with the ideal goal being to connect those operations to the platform’s frontend . The user stories and points planned for this sprint include: setting up the movie database using the Utelly, Guidebox, and/or OMDb APIs (3); getting recommendations (3); viewing movies as a dynamic grid (2); creating a REST endpoint to filter genres (2); creating a REST endpoint to retrieve user recommendations (2); implementing login with OAuth (2); setting up basic database operations (2); adding ratings (1); updating ratings (1); and displaying each movie within the grid using a card layout (1). This is a total of ten user stories valued at nineteen story points.

All of the above user stories are being or have been implemented except for setting up the movie database using various APIs. We applied for an API key for the Guidebox API, which seems like it will be the easiest to use for populating the database. However, we have not yet received a response and so will likely be saving this user story for the next sprint. This user story also has a relatively low priority; it is more important to verify that the database operations are working using test values than populating the database with real movies. If we still do not receive a response from Guidebox by early next sprint, we will look into manually entering a small battery of movies into the database for demonstration purposes.

The project seems to be progressing relatively smoothly. Though there are many tasks that still need to be completed before we reach a minimum viable product, it feels like we are on pace to complete these tasks. The most challenging aspect of the project has been adapting to technologies which might not be familiar to all of us. However, we have each gravitated towards the aspects of the project with which we feel most comfortable, which has resulted in a healthy balance among the group.

# 2. Specific Goals

***2.1 #8: Set up basic database operations***

***2.1.1 Story Description:***

*Build the necessary tables in an SQLite database using DB Browser and add the basic functions necessary to retrieve and store items in the database.*

***2.1.2 Story Acceptance Criterion***

*Users and movies can be added to the database.*

***2.1.3 Story Dependencies***

*None*

***2.1.4 Story Challenges***

*It is not obvious as to how all of the relevant data should be stored in the database tables, so this needs to be worked out.*

***2.1.5 Story Assigned to***

*Avery*

***2.1.6 Story Points***

*2*

***2.1.7 Status: Completed or not***

*Completed*

***2.2 #9: Add a user’s rating***

***2.2.1 Story Description***

*Implement the backend code necessary to add a user’s rating to the database and create the API endpoint to communicate with the frontend.*

***2.2.2 Story Acceptance Criterion***

*Movie ratings can be added to the database from within the user class (no frontend integration yet)*

***2.2.3 Story Dependencies***

*#8*

***2.2.4 Story Challenges***

*Best way to insert data into the database table such that it can be retrieved in a readable form*

***2.2.5 Story Assigned to***

*Avery/Samir*

***2.2.6 Story Points***

*1*

***2.2.7 Status: Completed or not***

*Completed*

***2.3 #10: Update a user’s rating***

***2.3.1 Story Description***

*Implement the backend code necessary to update a user’s rating in the database and create the API endpoint to communicate with the frontend.*

***2.3.2 Story Acceptance Criterion***

*Movie ratings can be updated in the database from within the user class (no frontend integration yet)*

***2.3.3 Story Dependencies***

*#9*

***2.3.4 Story Challenges***

*Best way to insert data into the database table such that it can be retrieved in a readable form*

***2.3.5 Story Assigned to***

*Avery/Samir*

***2.3.6 Story Points***

*1*

***2.3.7 Status: Completed or not***

*Completed*

***2.4 #11: Card layout for each movie within a grid***

***2.4.1 Story Description***

*Create an aesthetically-pleasing card view that movies will be displayed as on the main grid.*

***2.4.2 Story Acceptance Criterion***

*A card is visible and informative, containing the space to hold all necessary information about a movie.*

***2.4.3 Story Dependencies***

*None*

***2.4.4 Story Challenges***

*Figuring out the exact data that will be displayed in the movie card so it can be built in to the card’s design*

***2.4.5 Story Assigned to***

*Samir*

***2.4.6 Story Points***

*1*

***2.4.7 Status: Completed or not***

*Completed*

***2.5 #12: Get all movies endpoint***

***2.5.1 Story Description***

*Create an API endpoint in the Flask project to return all movies from the database.*

***2.5.2 Story Acceptance Criterion***

*All movies can be retrieved from the database using the Flask interpreter (no frontend integration yet).*

***2.5.3 Story Dependencies***

*#8*

***2.5.4 Story Challenges***

*Structuring the endpoint so that it can be useful to the frontend when future features are implemented.*

***2.5.5 Story Assigned to***

*Samir*

***2.5.6 Story Points***

*1*

***2.5.7 Status: Completed or not***

*Completed*

***2.6 #13: Get recommendations***

***2.6.1 Story Description***

*Create and implement a content-based recommendation algorithm that takes a user’s past ratings and returns recommendations for movies to watch in the future.*

***2.6.2 Story Acceptance Criterion***

*Recommendations for a user can be retrieved by calling this function from the user class.*

***2.6.3 Story Dependencies***

*#9*

***2.6.4 Story Challenges***

*Designing the algorithm for calculating the recommendations, figuring out how complex to make it*

***2.6.5 Story Assigned to***

*Avery*

***2.6.6 Story Points***

*3*

***2.6.7 Status: Completed or not***

*Completed*

***2.7 #14: Implement login with OAuth***

***2.7.1 Story Description***

*User should be able to click a login button and be taken to a Google OAuth screen where they can sign in with Google credentials. If verified, it brings them back to the home page.*

***2.7.2 Story Acceptance Criterion***

*User can successfully log in and log out of a profile using Google credentials and those credentials (username, email, avatar) can be accessed.*

***2.7.3 Story Dependencies***

*None*

***2.7.4 Story Challenges***

*Understanding the OAuth protocol and using the return token to pass to future endpoints*

***2.7.5 Story Assigned to***

*Samir/Carter*

***2.7.6 Story Points***

*1*

***2.7.7 Status: Completed or not***

*Completed*

***2.8 #15: View movies as a dynamic grid***

***2.8.1 Story Description***

*All Movies should be rendered in a dynamic and resizable grid of cards*

***2.8.2 Story Acceptance Criterion***

*The application can be viewed both on mobile and web, with multiple re sizing and the number of cards in any given row automatically adjusts*

***2.8.3 Story Dependencies***

*#17*

***2.8.4 Story Challenges***

*Figuring out how to store the state of all possible movies to be rendered in a performance efficient manner*

***2.8.5 Story Assigned to***

*Samir*

***2.8.6 Story Points***

*1*

***2.8.7 Status: Completed or not***

*Not completed - besides waiting for a response regarding the Guidebox API (detailed again below), we have spent more time this sprint working on our API endpoints. This user story will remain a priority for future sprints, though.*

***2.9 #16: REST endpoint to filter genre***

***2.9.1 Story Description***

*Returns a list of all movies that fit a certain genre*

***2.9.2 Story Acceptance Criterion***

*A SQL query that returns all possible matches and handles cases where there are no matches.*

***2.9.3 Story Dependencies***

*#17*

***2.9.4 Story Challenges***

*A larger database will be more representative of the various genres available for movies*

***2.9.5 Story Assigned to***

*Avery/Samir*

***2.9.6 Story Points***

*1*

***2.9.7 Status: Completed or not***

*Completed*

***2.10 #17: Set up movie database using Utelly, Guidebox, and/or OMDb APIs***

***2.10.1 Story Description***

*Retrieve all the movies available on the major streaming platforms and enter them into the movies database table.*

***2.10.2 Story Acceptance Criterion***

*Any movie can be retrieved from the movies database along with its availability across streaming platforms.*

***2.10.3 Story Dependencies***

*#8*

***2.10.4 Story Challenges***

*Getting access to APIs, writing scripts to put all the movies into the database*

***2.10.5 Story Assigned to***

*Avery*

***2.10.6 Story Points***

*3*

***2.10.7 Status: Completed or not***

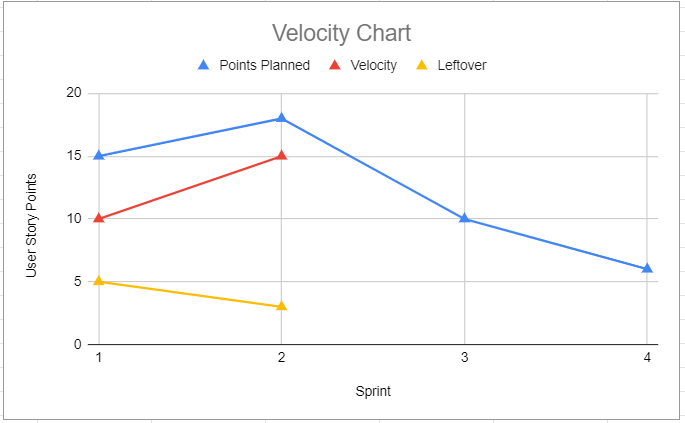
*Not completed - we are still waiting on a response from Guidebox for an API key. If there is no response in the near future, we will look into manually entering a smaller subset of movies into the database table.*

# 3. Analytics

## 3.1 Sprint/Product Burndown Chart

### 

## 3.2 Sprint Velocity (sample chart shown below)



***Average Velocity: 12.5 user story points/sprint***

# 4. Conclusion

In conclusion, our analytics show that we have completed a significant and stable amount of work over the first two sprints. The estimated metrics for the third and fourth sprints are subject to change as we discover new features that we need to implement, but it appears that we are on a reasonable pace to deliver a finished product by the project’s deadline. The first two sprints involved more tasks that could be easily accomplished because they concerned the project’s setup and initial features, whereas the tasks planned for the final two sprints are more about implementing the specific features that make our platform unique. Therefore, it could be necessary to raise the user story points for some of the tasks that we have planned for the final two sprints if they involve more detailed work than that required to complete the tasks in the first two sprints.