**CS673 Software Engineering** 

**Team 6 - Blockbuster**

**Project Proposal and Planning**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Joshua Shilts | Leader | *Joshua Shilts* | 9/11/2024 |
| Elizabeth Tyree | Design and Implementation Leader | *Elizabeth Tyree* | 9/11/2024 |
| Ricky Zheng | Configuration Leader | *Ricky Zheng* | 9/22/2024 |
| Alex Flinchum | QA Leader | *Alex Flinchum* | 9/11/2024 |
| James Zheng | Security Leader | *James Zheng* | 9/11/2024 |
| Elizabeth Tyree | Requirement Leader | *Elizabeth Tyree* | 9/11/2024 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| 1.0 | Joshua Shilts | **9/11** | **Creation** |
| 1.1 | Joshua Shilts | **9/22** | **Update** |
| 1.2 | Joshua and Alex | **10/7** | **Update** |
|  |  |  |  |
|  |  |  |  |

[Overview](#_heading=h.gjdgxs)

[Related Work](#_heading=h.30j0zll)

[Proposed High level Requirements](#_heading=h.1fob9te)

[Management Plan](#_heading=h.3znysh7)

[Objectives and Priorities](#_heading=h.2et92p0)

[Risk Management (need to be updated constantly)](#_heading=h.tyjcwt)

[Timeline (need to be updated at the end of each iteration)](#_heading=h.3dy6vkm)

[Configuration Management Plan](#_heading=h.1t3h5sf)

[Tools](#_heading=h.4d34og8)

[Deployment Plan if applicable](#_heading=h.2s8eyo1)

[Quality Assurance Plan](#_heading=h.17dp8vu)

[Metrics](#_heading=h.3rdcrjn)

[Code Review Process](#_heading=h.26in1rg)

[Testing](#_heading=h.lnxbz9)

[Defect Management](#_heading=h.35nkun2)

[References](#_heading=h.1ksv4uv)

[Glossary](#_heading=h.44sinio)

# Overview

Team Blockbuster is creating a web based application where a user can view a static webpage that will submit a query to a backend that will respond with the requested data. The application utilizes a React frontend, a Python flask backend, and a Sqlite database designed in a microservice architecture that will provide asynchronous form submission. The motivation behind the project is that many companies want to understand trends from the top grossing movies across the industry. The Application will provide the user the ability to filter and query a dataset of 250 pre-ranked videos to determine trends and make insights into what movies should be produced given the current trend.

# Related Work

N/A

# Proposed High level Requirements

* 1. Functional Requirements  
     (For each functional requirement, please give a feature title and a brief description using the following format: As (a role), I want to (action), so that (value).)
     1. Essential Features (the core features that you definitely need to finish):

(For each essential features, please give a rough estimation in terms of person hours or an range of person hours)

Feature/Chore Title: [Create Frontend MVP](https://www.pivotaltracker.com/story/show/188277925)

As a user I want to be able to input and select data in the webpage and be able to submit using a button to find trends about certain movies I am interested in, so that I can find and discover new patterns and trends based on a given set of criteria.

Estimated hours: 30

* Test case ID, name:
  + 1; Frontend | UI | Create Frontend MVP | Build the Application | Successful

Feature/Chore Title: [Create Backend MVP](https://www.pivotaltracker.com/story/show/188277924)

As a user I want there to be a Python backend that will respond to REST API calls so that when the backend receives a request it can respond to that request with data in proper format to the correct location

Estimated hours: 30

* Test case ID, name
  + 2; Backend | UI | Create Backend MVP | Run the Application | Successful

Feature/Chore Title: [Write Front-end Requirements](https://www.pivotaltracker.com/story/show/188238798)

As a dev I need there to be some frontend requirements so that what the frontend needs to do can be clearly defined in order to reduce confusion, and rework.

Given the need that software need requirements

when this ticket is complete

then a document will exist that contains all the software requirements for the front end.

This doc may need to be in github.

Feature/Chore Title: [Map out Database Architecture](https://www.pivotaltracker.com/story/show/188254491)

As a developer, I want to create a database structure for the Kaggle top 250 movie dataset in a database, so that requirements for the docker container can be outlined and the microservice can communicate with the python backend.

Estimated hours: 25

Feature/Chore Title: [Write out Backend Requirements](https://www.pivotaltracker.com/story/show/188277933)

As a developer, I want to create frontend requirements, so that what the frontend needs to do can be clearly defined in order to reduce confusion, and rework.

Estimate hours: 20

Feature/Chore Title: [UI: Implement welcome screen](https://www.pivotaltracker.com/story/show/188313845)

As a User, I want to be able to see a welcome page,

so that I know what webpage I am on and what to expect.

Acceptance Tests:

Given a link to a webpage or website, when I click on the link, then I am brought to a welcome page with text and a button.

Given a button on the welcome page, when I click on it then I will be brought to a search form page.

Estimate hours: 6

Feature/Chore Title: [UI: Implement data input fields and drop downs](https://www.pivotaltracker.com/story/show/188328806)

As a user, I want to give data within the webpage, so that I can search for data that matches the data given.

Given the need to request criteria from the user.

When this ticket is complete

The frontend will have an area with options for the user to give criteria.

Given the need to package this request and send it to the backend

when this ticket is complete

then the frontend will wrap all data points in a JSON object and send it to the backend.

Estimate hours: 6

Feature/Chore Title: [UI: Implement Submit page structure](https://www.pivotaltracker.com/story/show/188328771)

As a User, I want to have a clean and organized search page, so that I have a search area and a display area that displays what I have searched.

Given the task of implementing a submit page, when the submit page has been finalized and completed, then there will be multiple drop-down options:

-genre

-director

-runtime

-year

Estimate hours: 7

Feature/Chore Title: [UI: Create Clear Button Functionality](https://www.pivotaltracker.com/story/show/188382198)

As a developer, I want to create a clear button in the submit form, so that the user can clear all data that they have added.

Given the need to clear data from text fields from previous  
when this ticket is complete  
then a button will exist that allows a user to clear their results from a previous or current query.

Given the need to not have the clear button functionality crash the page  
when this ticket is complete  
then the webpage will not crash if there are NULL values in the text fields

Estimate hours: 6

* + 1. Desirable Features (the nice features that you really want to have too):
       1. As a user after submitting a query with or without filters I would like to see a graph to visualize the query results because I want to visualize the data to provide better analysis in determining new trends among moviegoers.
    2. Optional Features (additional cool features that you want to have if there is time):

* 1. Nonfunctional Requirements
     1. Security requirements
        1. None yet

# Management Plan

## Objectives and Priorities

Goals:

* Create deployment
* Expand UI Functionality
* Create end to end system functionality
* Integrate SCA
* Integrate PyLint to database code
* Address a security concern

## Risk Management (need to be updated constantly)

The main risk the team is currently experiencing is a lack of time to finish out desired features that would really make our application shine. There is a lot of new technology on the way as well that the team is going to have to learn and come up to speed on, which have all been documented in our updated Rick management sheet.

**Risk Management Sheet Link:** https://docs.google.com/spreadsheets/d/1JGyigLyJNwMd2LBrxEZ32dsJVCDcR5YC/edit?usp=sharing&ouid=113994091918424032168&rtpof=true&sd=true

## Timeline (this section should be filled in iteration 0 and updated at the end of each later iteration)

| Iteration | Functional Requirements(Essential/Disable/Option) | Tasks (Cross requirements tasks) | Estimated/real person hours |
| --- | --- | --- | --- |
| 1 | Created MVP for UI Created MVP for backend  Create Pipeline | 1 | 40 |
| 2 | [UI: Create Clear Button Functionality](https://www.pivotaltracker.com/story/show/188382198)  [UI: Implement Submit page structure](https://www.pivotaltracker.com/story/show/188328771)   [UI: Implement data input fields and drop downs](https://www.pivotaltracker.com/story/show/188328806)  [UI: Implement welcome screen](https://www.pivotaltracker.com/story/show/188313845) | 4 | 25 |
| 3 | Deployment UI Features More testing | 1 | 20 |

# Configuration Management Plan

## Tools

* + - Google Drive
    - Google Docs
    - GitHub
      * GitHubs Actions
    - Ubuntu
    - PyCharm
    - GitHubDesktop
    - Discord
    - Pivotal Tracker
    - Python (v3.12)
    - Pip (v24.2)
      * Python Flask
      * PyLint
      * PyTest
      * Pandas
    - Docker (v25.0.6)
    - Podman
    - Docker Compose (27.1.0)
    - nodeJS (v20.17.0)
    - Jest
    - Babel
    - SQLite (v3)
    - VmWare Community 17
  1. Code Commit Guideline and Git Branching Strategy

To keep things simple we are going to have one branch per new feature that will get merged into a development branch that is not main. After each iteration we will merge into main where we may or may not use a tag to denote this. The pull request will be used extensively where each merge will require a review from several members of the team.

## Deployment Plan if applicable

Our plan is to host this on an amazon web service and use images to run our application. However we still need to do some research on how to accomplish this. This current plan is to look into AWS Elastic Beanstalk as a viable deployment option.

# Quality Assurance Plan

## Metrics

(Describe the metrics to be used in the project to measure the quality of your software. Each metric should be measurable and quantifiable. Examples of metrics include product complexity (LOC, # of files, # of classes, # methods, cyclomatic complexity, etc.) , defect rate (# of defect per KLOC), # of test cases, test case pass rate, cost (# of person hours used), # of user stories completed, etc. **The result of these metrics should be reported in the progress report/ iteration summary sheet.**)

| Metric Name | Description |
| --- | --- |
| Static Analysis | List number of vulnerabilities, code standard violations |
| Unit Test | Number of unit test |
| User Stories | Tracked time dedicated to hardening or improving codebase |

* 1. Coding Standard

[Pep 8](https://peps.python.org/pep-0008/) which is a coding standards guide for Python that enforces style rules on your source code.

## Code Review Process

For the code review process it was decided that the design and implementation leader needs to be included in the Pull request as well as two other developers. All need to approve any changes. When the pipeline is built the pipeline will also have to successfully deploy before any changes can be merged.  
What should be determined if Pull request is done in this case should be if the changes meet the acceptance criteria laid out in the description of the corresponding ticket in pivotal tracker. A reviewer's comments should be in line with the scope of the work laid out in the ticket as well as the Pep 8 coding standards.

## Testing

For manual testing:

We will build several small python scripts that send a Rest API response or request to one of the micro services to simulate behavior. The team will also build unit tests and conduct regular linting of the python code using PyLint.

For automated testing:

The goal is to get to a point where each developer's merge request has to pass all unit tests and integration tests in the pipeline before the new changes are merged into the code base.

## Defect Management

To manage defects we are going to use pivot tracker and record defects as we find them in the code base or through our testing processes. The personnel to manage defects will most likely be the person who found it or if required the implementation lead.

# References

Dataset: [IMDB Top 250 Movies Dataset (kaggle.com)](https://www.kaggle.com/datasets/rajugc/imdb-top-250-movies-dataset), By Chidambara Raju G (2022)

Database:

<https://www.sqlitetutorial.net/sqlite-python/creating-tables/>  
 https://www.sqlitetutorial.net/sqlite-python/creating-database/  
  
Deployment:  
 <https://github.com/docker/awesome-compose/tree/master/flask>

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/GettingStarted.html>

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html>  
  
CI/CD  
 Branch sanitization: <https://github.com/transferwise/sanitize-branch-name>

Development:  
 <https://peps.python.org/pep-0008/>

Pylint Integration: pass-fail functionality

<https://stackoverflow.com/questions/50944575/how-to-set-a-pylint-score-threshold>

Button Positioning in html:

<https://stackoverflow.com/questions/13998677/html-css-button-positioning>

Reset button to clear all criteria:

<https://www.geeksforgeeks.org/how-to-reset-all-form-values-using-a-button-in-html/>

Sonarqube setup video:

https://www.youtube.com/watch?v=O0yFLS30InY

# 

# Glossary

**API -** Application Programming Interface

**AWS -** Amazon Web Services

**CI/CD -** Continuous Integration Continuous Deployment

**IMDB -** Internet Movie Database

**JSON -** Java Script Object Notation

**MVP -** Minimum Viable Product

**REST API -** Representational State Transfer Application Interface

**SCA -** Static Code Analysis

**UI -** User Interface