**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* | 9/11/2024 |
| Alex Flinchum | QA Leader | *Alex Flinchum* | 9/11/2024 |
| James Zheng | Security Leader | *James Zheng* | 9/11/2024 |
| Rekik Mengstu | Requirement Leader | *Rekik Mengstu* | 9/11/2024 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| 1.0 | Joshua Shilts | **9/11** | **Creation** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[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

(Please give an overview of your project. It should include the motivation, the purpose and the potential users of the proposed software system, the basic functionality of the proposed software system and the possible technology stack to be used. )

Team Blockbuster is creating a Data Science based application called Top Movies API. The Top Movies API will provide the ability to query and display movie data based on a variety of data points. The motivation behind this is that people are very interested in understanding the trends behind who is seeing what movies and for what reasons. The potential users of this could be advisors, movies producers, or streaming companies whose purpose would be to help them determine how well a new movie idea is going to work. The application will be a microservice based application built using the Python Flask framework. It will utilize ReactJs for the frontend while we use python for the backend and SQLite for the database.

# 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)

1. **Release date:**

As a user I want to submit a movie release date so that the system can filter movies released on the given day.

As a developer I want to ensure the release date field is validated with the correct format so that only appropriate values get sent to database

As a user I want to receive a confirmation message upon submission so that the user knows submission was successful.

1. **Genres**:

As a user I want to submit a movie's genre so that the system can filter movies by genre.

As a developer I want to ensure a movie's genre field is validated with the correct format so that only appropriate values get sent to database

As a user I want to receive a confirmation message upon submission so that the user knows submission was successful.

1. **Cast**:

As a user I want to submit an actor's name so that the system can return a list of all the movies that the actor was in.

As a developer I want to create an automatic filter for unintended duplicates so that the system only returns a unique list of movies.

As a user I want to filter by an actor's name and movie genre so that the system can return a list of movies that the given actor was in but for only that given genre.

* + 1. Desirable Features (the nice features that you really want to have too):
       1. As a user I want to submit an amount of money so that the system can filter the movies by budget.
       2. As a user I want to filter by both budget and box office so that the system can show which movies performed the best monetarily.
       3. As a developer I want to be able to easily update field names like certificate to more user understandable language like rating, and rating to ranking so that the system results are more human readable.
    2. Optional Features (additional cool features that you want to have if there is time):
       1. **Graphics:**  
          As a user I want to select a graphical option so that the system can plot Charts for the frontend of the stack to display data visually.

* 1. Nonfunctional Requirements
     1. Security requirements

# Management Plan

## Objectives and Priorities

Goals:

* Write requirements
* Provide a MVP as soon as possible to unblock other team members from building up the deployment functionality.
* Create Unit test
* Integrate Unit Tests into pipeline and deployment

## Risk Management (need to be updated constantly)

One of the main risks the team identified was the amount of time we have to accomplish this work and become familiar with a framework and software deployment. To counteract this we have decided to meet 3 times a week to improve communication and collaboration as well as swarm of the initial functionality to unblock the pipeline and deployment work from not having the executable required to run the pipeline.

**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 | All user input fields and buttons | 0 | 8 |
| 2 | End to end integration | 0 | 10 |
| 3 | Testing and Verification | 0 | 20 |

# Configuration Management Plan

## Tools

* + - Google Drive
    - Google Docs
    - GitHub
    - PyCharm
    - GitHubDesktop
    - Discord
    - Pivotal Tracker
    - Python (v3.12)
    - Pip (v24.2)
      * Python Flask
      * PyLint
    - Docker (v25.0.6)
    - Docker Compose (27.1.0)
    - nodeJS (v20.17.0)
    - SQLite (v3)
  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

TBD

# 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

Pip 8 coding standards

## 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 Pip 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)

# Glossary