*Movie Recommendation System*

Project Plan

Version 1.3

**TEAM NAME: Xplorers**

**TEAM MEMBERS: Purva Bansod**

**Priyanka Jadhav**

**Preetam Maske**

**Priyanka Mutha**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| Apr 2 2025 | 1.1 | Initial project submission | Priyanka Mutha |
| Apr 25 2025 | 1.2 | Added test plan and updated architecture | Purva Bansod, Priyanka Jadhav |
| May 5 2025 | 1.3 | Integrated frontend with backend and updated deliverables | Preetam Maske, Priyanka Mutha |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose of this document 4

1.2 Intended Audience 4

1.3 Scope 4

1.4 Definitions and acronyms 4

1.4.1 Definitions 4

1.4.2 Acronyms and abbreviations 5

1.5 References 5

2. Background and Objectives 5

3. Organization 5

*3.1* Project group 5

3.2 Customer 6

4. Development process 6

*5.* Deliverables 6

6. Project risks 6

7. Communication 7

7.1 Canvas 7

7.2 Git 7

8. Project plan 7

8.1 Time schedule 7

8.1.1 Remarks 7

8.2 Test plan 9

8.2.1 Testing Remarks 10

9. References 10

# Introduction

In the age of digital content overload, users often struggle to find personalized and relevant content. Recommendation systems have become essential in guiding users to discover items that align with their preferences. This project aims to develop a movie recommendation system that suggests films based on user preferences and historical data using machine learning techniques.

The system is designed with a modular architecture, consisting of a user-friendly frontend interface, a Python-based RESTful backend API, a structured database, and a recommendation engine that leverages collaborative and content-based filtering methods. The platform ensures a seamless user experience, delivering real-time recommendations and is deployable on scalable cloud infrastructure.

The primary objective of this project is to deliver an end-to-end recommendation system that is both accurate and responsive, showcasing the integration of data science, software development, and cloud deployment.

Our Movie Recommender System, built using Python and Natural Language Processing (NLP), offers a user-friendly way to discover your next favorite movie. We utilize the bag-of-words concept to generate movie recommendations based on factors like similarity, tags, genre, and production company. Additionally, users can access detailed information about each movie and its cast, enhancing both discovery and engagement.

## Purpose of this document

The purpose of this document is to provide a comprehensive project description for the application called **Movie Recommendation System**, which is designed to assist users in discovering movies that align with their preferences and viewing history. This document outlines key aspects of the project, including its objectives, organization structure, stakeholder roles, deliverables, identified risks, communication strategies, time schedules, and financial planning.

## Intended Audience

This document shall be used in all phases of the project as a guideline. Intended audiences of this project are all project stakeholders:

* project supervisor
* project leader
* team members

## Scope

This document defines the project plan of the Movie Recommendation System application. The overview includes objectives of the project, organization of the project team, development process that is going to be used during the project, assessment of possible risks, communication used between project stakeholders and project plan that includes time schedule and activity plan.

## Definitions

### Definitions

|  |  |
| --- | --- |
| **Keyword** | **Definitions** |
| Movie Recommendation System | The name of the project |
| Project Supervisor | A person in charge of supervising the project |
| Project Leader | A person in charge of organizing the team and communicating with the project supervisor |
| Team Member | An active member of the team responsible for making the job done |
| Milestone | A time in a project that marks the end of a project phase or the completion of an important deliverable. |
| Git | Version control system that will be used in this project |
| Scrum | An iterative and incremental agile software development method for managing software projects and product or application development |
| Kunagi | Web-based tool for integrated agile project management and collaboration based on Scrum |
| Scrum sprint | The basic unit of development in Scrum |
| Scrum master | Ensures the smooth working of the Scrum team and enforces Scrum practices |
| Product owner | Responsible for product management and its quality |

## References

Images reference used in Movie Recommendation System are:

* Google images, all the images used in the application are referenced from google.

# Background and Objectives

**Background**

With the exponential growth of digital content, users are often overwhelmed by the vast selection of movies available on various platforms. A personalized movie recommendation system can enhance user experience by suggesting relevant content based on their preferences, past behavior, and popular trends.

**Objectives**

* To design and develop a system that provides accurate movie recommendations using user input and historical data.
* To implement machine learning models that analyze patterns in user preferences and movie attributes.
* To create an intuitive user interface for browsing, rating, and receiving personalized movie suggestions.
* To ensure scalability and efficiency in handling large datasets and real-time user queries.

# Architecture & High Level Design

The system architecture of the Movie Recommendation System integrates key components that work together to provide users with personalized movie suggestions. The architecture includes:

* **Frontend**:  
  A user-friendly web interface developed using streamlit enabling users to input preferences and view recommended movies with detailed descriptions and cast information.
* **Backend**:  
  A **Python-based RESTful API** built using frameworks streamlit. It handles incoming user requests, communicates with the database, applies recommendation logic, and returns results to the frontend.
* **Natural Language Processing (NLP) Module**:  
  Implements **NLP techniques**, such as **bag-of-words** and **TF-IDF**, to process movie overviews, tags, and metadata. This module enhances content-based filtering by capturing semantic similarities between movies.
* **Database**:  
  User data, movie metadata, ratings, and preprocessed features are stored in a **relational database** hosted on **Amazon RDS** for high availability and scalability. Movie information is sourced from CSV datasets and APIs like TMDb.
* **Recommendation Engine**:  
  Combines **collaborative filtering** and **content-based filtering** (enhanced with NLP) to generate accurate movie suggestions tailored to individual users.
* **Deployment**:  
  The complete system is **deployed on cloud infrastructure** (e.g., AWS EC2 or Heroku) for high availability, ease of access, and scalability.
* **CI/CD Pipeline (CircleCI)**:  
  Utilizes **CircleCI** for **continuous integration and deployment**. CircleCI automates testing, ensures code quality, and deploys new features seamlessly from the GitHub repository, supporting agile development workflows.

# Organization

San Jose state University- Master of Science In Software Engineering -Extended

## Project group

|  |  |
| --- | --- |
| **Name** | **Responsibility (roles)** |
| Purva Bansod | Project Leader, Backend Developer |
| Priyanka Angad Jadhav | Backend Developer |
| Preetam Maske | Frontend Developer |
| Priyanka Ajit Mutha | Frontend Developer |

## Customer

The target customers are listed below:

* Movie enthusiasts looking for personalized recommendations
* Streaming platform users
* Media platforms aiming to improve user engagement

# Development process

The project will use the **Agile development methodology** with weekly sprints and continuous integration. Each sprint will focus on implementing key features, followed by testing and review. Tools like Trello, GitHub, and Slack will be used for collaboration and tracking.

# Deliverables

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **To** | **Output** | **Planned week** | **Promised week** | **Late +/-** | **Delivered week** | **Notes** |
| Team | Dataset collection & preprocessing | Week 1 | Week 1 | 0 | Week 1 | CSV and TMDb data |
| Team | Backend API development | Week 2 | Week 2 | 0 | Week 2 | RESTful API implemented |
| Team | Recommendation algorithm | Week 4 | Week 4 | 0 | Week 4 | Collaborative filtering done |
| Team | Frontend UI/UX | Week 5 | Week 5 | 0 | Week 5 | HTML/CSS/JS interfaces |
| Team | Integration and testing | Week 6 | Week 6 | 0 | Week 6 | End-to-end functionality |
| Team | Final report and documentation | Week 8 | Week 8 | 0 | Week 8 | Project documentation |

# Project risks

|  |  |  |
| --- | --- | --- |
| **Possibility** | **Risk** | **Preventive action** |
| High | Delay in integrating backend and frontend | Early communication and weekly sync-ups between dev sub-teams |
| Medium | Data quality issues from external sources | Pre-cleaning and fallback CSV datasets from reliable sources |
| Medium | Algorithm performance not meeting accuracy | Test multiple ML models, fallback to content-based filtering |
| Low | Deployment issues on cloud | Used containerization and tested deployment early on |

# Communication

Effective communication is essential for the successful completion of the project. The team adopted a structured communication approach to ensure all members are aligned, tasks are tracked, and progress is transparent throughout the development cycle.

## Collaboration

**Our team maintained regular and effective collaboration through multiple channels."**

* Weekly team meetings via Zoom/Google Meet
* Daily task coordination using WhatsApp group
* Task tracking using Trello/Google Sheets
* Shared Google Drive for design drafts, notes, and documentation
* Peer reviews for code and reports to ensure quality

This multi-channel approach helped us stay aligned on goals and timelines.

## Git

All source code and finished documentation will be uploaded to Github repository.

Repository URL: <https://github.com/Preetam3620/Movie-Recommendation-System>

Deployed Website: http://54.183.152.225:8501/

# Project plan

## Time schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Id** | **Milestone**  **Description** | **Responsible Dept./Initials** | **Finished week Plan** | **Forecast Week** | **+/-** |
| 1 | Requirements gathering | All | Week 1 | Week 1 | 0 |
| 2 | Dataset collection & preprocessing | All | Week 2 | Week 2 | 0 |
| 3 | Backend API development | Backend Team | Week 3 | Week 3 | 0 |
| 4 | Recommendation algorithm | All | Week 5 | Week 5 | 0 |
| 5 | UI design & frontend setup | UI/UX | Week 5 | Week 5 | 0 |
| 6 | System integration & testing | All | Week 6 | Week 6 | 0 |
| 7 | Final documentation | All | Week 8 | Week 8 | 0 |

### Remarks

|  |  |
| --- | --- |
| **Remark Id** | **Description** |
| R1 | Dataset was partially cleaned manually due to inconsistencies in TMDb metadata. |
| R2 | Backend API was completed ahead of schedule, allowing for early integration tests. |
| R3 | Frontend revisions were based on early user feedback, requiring minor redesign. |

## Test plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No. | 001 | Phase: | 1 | Author: | Priyanka Jadhav, Purva Bansod | Date: 15 April 2025 |
| Test Category: | | **Functional** | | |  |  |
| Software Product: | | Recommendation System Backend | | | |  |
| Test Title: | | API Response Test | | | | |
| Test Purpose: | | To verify the API returns correct recommendations for given user input. | | | | |
| Test Setup: | | Localhost deployment of backend, sample user preferences sent via Postman. | | | | |
| Prerequisites: | | API server running, sample data preloaded. | | | | |
| Procedure: | | Send POST request with user ID and check for 200 response with valid movie list. | | | | |
| Checks: | | HTTP status, response time, and data schema validity. | | | | |
| Expected Results: | | 200 OK with a list of 5+ recommended movies. | | | | |
| Result: | | Passed | | | | |
| Reason for Failure: | | N/A | | | | |
| Remarks: | | Validated with test user and expected movie genres. | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No. | 002 | Phase: | 1 | Author: | Priyanka Mutha, Preetam Maske | Date: 30 April 2025 |
| Test Category: | | **Integration** | | |  |  |
| Software Product: | | Full Recommendation System | | | |  |
| Test Title: | | Frontend-Backend Integration Test | | | | |
| Test Purpose: | | To ensure seamless interaction between the frontend interface and backend API. | | | | |
| Test Setup: | | Frontend hosted locally or on test server, backend running on localhost or deployed endpoint. | | | | |
| Prerequisites: | | Backend API must be live and functioning; frontend should have input form connected to API. | | | | |
| Procedure: | | Launch the frontend interface.  Enter sample user preferences and submit the form.  Observe if the frontend correctly fetches and displays recommendations. | | | | |
| Checks: | | API is called successfully.  Data returned is rendered in the frontend.  Errors are handled gracefully. | | | | |
| Expected Results: | | Recommendations are displayed on the frontend based on API response. | | | | |
| Result: | | Pass | | | | |
| Reason for Failure: | | N/A | | | | |
| Remarks: | | UI successfully retrieved and displayed recommended movies for all test inputs. | | | | |

# References

<https://medium.com/@g.bharthvajan/building-movie-recommendation-system-with-aws-ml-services-869bf2b8486f>

https://madgicaltechdom.com/blog/how-to-build-a-movie-recommendation-system-with-aws/