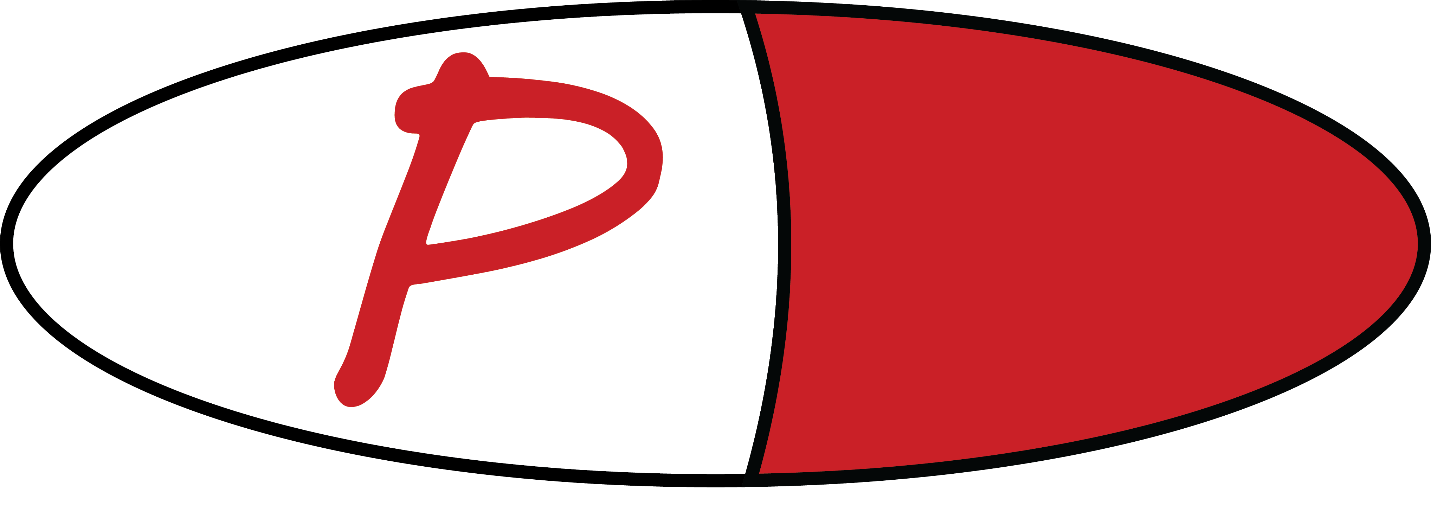
**Requirements Engineering Report**

**Date: February 17, 2016**

**Version v1.00**

**Software Project Name**

Prescribe



**By:** Team StarMony

***Project Manager:*** Jeremy Brown

***Quality Assurance:*** Brandyn Deffinbaugh

***Technical Lead:*** Mitchell Powell

**Table of Contents**

1. Introduction
   1. Objective
   2. Scope
   3. Success Criteria
   4. Collaboration with Stakeholders
2. Project Plan
   1. WBS
   2. Project Resources
   3. Responsibility Matrix
   4. Gantt Chart
   5. Pert Chart
   6. Cost Estimation
      1. Function Point Estimation
      2. LOC Estimation
      3. Cost Estimations
   7. Risk Plan
   8. Project Monitoring & Control Mechanisms
3. Requirements & Analysis Models
   1. Major Software Functions
   2. Use Case Diagrams
   3. Use Case Descriptions
   4. Activity Diagrams
   5. Sequence Diagrams
   6. Requirements Class Models
   7. Prototype Description
   8. Data Directory
   9. Limitations & Constraints
   10. Non-functional Requirements
4. Problems Encountered
5. Bibliography
6. Introduction
   1. Project Objective

The objective of our project is to provide a web application that can aid users in the discovery of new music based on their musical preferences.

* 1. Project Scope

The user will be able to input an artist or a band into search field which will provide a similar list of artists or bands. If the user creates an account using Facebook or Google+, they will be able to give feedback to train our model by up or down voting the search results. This will help grow the machine to give better feedback to our users. The user can also post their search results to their social media page. Search results will provide the user with a list of similar bands including the bands’ information such as their biography and discography. The user also be able to favorite a band for later reference. We also want to include a top artists section that users can view to see which artists have been given the most up votes for the week, month, or year.

* 1. Success Criteria
     1. Search Functionality
     2. Rating system integration
     3. Smooth Web GUI
     4. Icon graphics
  2. Collaboration with Stakeholders
* End-Users
* Facebook
* Google
* Spotify
* Pandora

1. Project Plan
   1. Work Breakdown Structure



* 1. Project Resources

|  |  |  |  |
| --- | --- | --- | --- |
| Resource Name | Cost | Description | Status |
| Eclipse IDE | Free | Internal Development Environment for Java programming language | Obtained |
| Domain Name | $20 / year | The domain name at which the web application will be hosted | In Progress |
| JavaScript Interpreter/Web Browser | Free | Interpreter for the JavaScript programming language, as the front end will be developed in JavaScript | Obtained |
| Labor | $9375 | 1.5 person/months at a rate of $6,250 per person month | In Progress |
| Web Hosting Server Space | $10 / month | Space on a server to host the data needs of the Prescribe project. | In Progress |

**Total:**

$9375 up front, $140 recurring annually

*This will certainly need to be updated after a little more analysis about what will be required for the project.*

* 1. Responsibility Matrix



* 1. Gantt Chart



* 1. Pert Chart



* 1. Cost Estimation
     1. Function Point Estimation
     2. Lines of Code Estimation
     3. Cost Estimation
  2. Risk Plan
  3. Project Monitoring & Control Mechanisms

There are several mechanisms that the Starmony group will utilize in order to ensure effective communication and proper maintenance of the code base. As a group, we will meet a minimum of once a week in a formal, face-to-face meeting to discuss the progress of the project, any issues that have arisen and for code review. In addition, the group will maintain regular group-wide electronic communication to ensure that all group members are kept up to date with what progress has been made, short term goals, and current issues. The group will maintain a “TODO” document on the Git repository to ensure that there is effective communication about what current issues are at hand.

It will be the responsibility of all group members to pull any changes from the remote repository whenever they work on their local branches so any integration issues can be caught early and fixed before they become a larger problem. In turn, it will also be the responsibility of each group member to push any completed changes or additions to the master branch of the remote repository as often as possible. Any works in progress should be maintained in a development branch to ensure a compiling codebase on the master branch at all times.

1. Requirements & Analysis Models
   1. Major Software Functions

1. Registered User Functions

1.1 Search

1.2 Login

1.3 Search Result Feedback

1.3.1 Upvote or Downvote

1.3.2 Save Artists

1.4 View Profile Info

1.5 View Artist Info

2. Non-Registered User Functions \*\*\*

2.1 Search

2.2 Account Registration

2.3 View Artist Information

3. Administrator Functions (Agent)

3.1 Login

3.2 User Account Control

3.3 Website Controls (Manual Override)

4. Server Functions (No Agent)

4.1 Login Authentication

4.2 Search Caching

4.3 Information Retrieval

4.4 Top Artist Info (Most upvotes, searches, etc...)

4.5 Email Confirmation

4.6 Account Removal (Via facebook, Google+

* 1. Use Case Diagram



* 1. Use Case Descriptions

**Use Case Number:** 01

**Use Case Name:** Search for Band Suggestions

**Primary Actor:** User (both registered and non-registered)

**Secondary Actor:** Server

**Goal:** Allow user to input artist and return suggestions

**Preconditions:** User must be on the website

**Trigger:** User wants to discover new artists

**Scenario:**

1. The user accesses the webapp.
2. The user inputs artist they wish for the search results to be based off of.
3. The server takes the input and delivers the search results.

**Exception:** The artist entered by the user is not a band or is not discoverable.

**Priority:** Essential, this is the main feature of our software.

**Use Case Number:** 02

**Use Case Name:** Account Registration

**Primary Actor:** Non-Registered User

**Secondary Actor:** Server

**Goal:** Allow a non-registered user to create an account

**Preconditions:**

* The system must be set up to allow user to link an account to our webapp
* User must have a Google or Facebook account

**Trigger:** User wants to create an account.

**Scenario:**

1. The user accesses the webapp.
2. The user links Google or Facebook account.
3. The server performs the necessary steps to connect account.
4. The server stores this information for future use.

**Exceptions:** The user does not have a Google are Facebook account

**Priority:** Moderate. This step will be necessary for us to implement our save and rating features.

**Use Case Number:** 03

**Use Case Name:** Login

**Primary Actor:** Registered User

**Secondary Actor:** Server

**Goal:** Allow user to login to their account.

**Preconditions:** User has already registered.

**Trigger:** User wants to access their account.

**Scenario:**

1. User goes to the login page
2. User enters username and password
3. Server confirms and logins in the user.

**Exceptions:**

* The user does not have an account
* The user enters the wrong information

**Priority:** Moderate. This step will be necessary for us to implement our save and rating features.

**Use Case Number:** 04

**Use Case Name:** Voting System

**Primary Actor:** Registered User

**Secondary Actor:** Server

**Goal:** Allow user to login to upvote or downvote a band in their search results.

**Preconditions:**

* User is logged in
* User has already searched for results

**Trigger:** User wants to vote on their results.

**Scenario:**

1. User searched for suggestions
2. Server has returned results
3. User selects whether they like a specific band (upvote) or if they do not (downvote)

**Exceptions:**

* The user does not provide a rating

**Priority:** Low. This step will help us implement our top artists of the week, month, and year page.

**Use Case Number:** 05

**Use Case Name:** Save Results

**Primary Actor:** Registered User

**Secondary Actor:** Server

**Goal:** Allow user to save a suggested band for future use

**Preconditions:**

* User is logged in
* User has already searched for results

**Trigger:** User wants to vote save their results.

**Scenario:**

1. The user has searched for suggestions
2. The server has provided results.
3. The user favorites an artist

**Exceptions:**

* The user does not save band

**Priority:** Moderate.

**Use Case Number:** 06

**Use Case Name:** Display Band Information

**Primary Actor:** User (both registered and non-registered)

**Secondary Actor:** Server

**Goal:** Allow user to view information on a suggested band

**Preconditions:**

* User is logged in
* User has already searched for results

**Trigger:** User wants to learn more about an artist

**Scenario:**

1. The user has searched for suggestions
2. The server has returned suggestions
3. The user clicks the display information button under an artist’s name
4. The webapp displays the artist’s information.

**Exceptions:**

* The user does not click the display information button

**Priority:** Low.

**Use Case Number:** 07

**Use Case Name:** View Profile Information

**Primary Actor:** Registered Information

**Secondary Actor:** Server

**Goal:** Allow user to view their personal information

**Preconditions:**

* User is logged in

**Trigger:** User wants to view their profile

**Scenario:**

1. The user clicks a profile button that will take them to their account page
2. Server brings user to that page and displays the user’s information

**Exceptions:**

* The user does not view account information.
* The user is not logged int.

**Priority:** Low. Must be implemented if the favorite system is created.

* 1. Activity Diagrams
  2. Requirements Class Models
  3. Prototype Description
  4. Data Directory
  5. Limitations & Constraints
  6. Non-functional Requirements

1. Problems Encountered
2. Bibliography

<https://creately.com>

http://musicmachinery.com/music-apis/