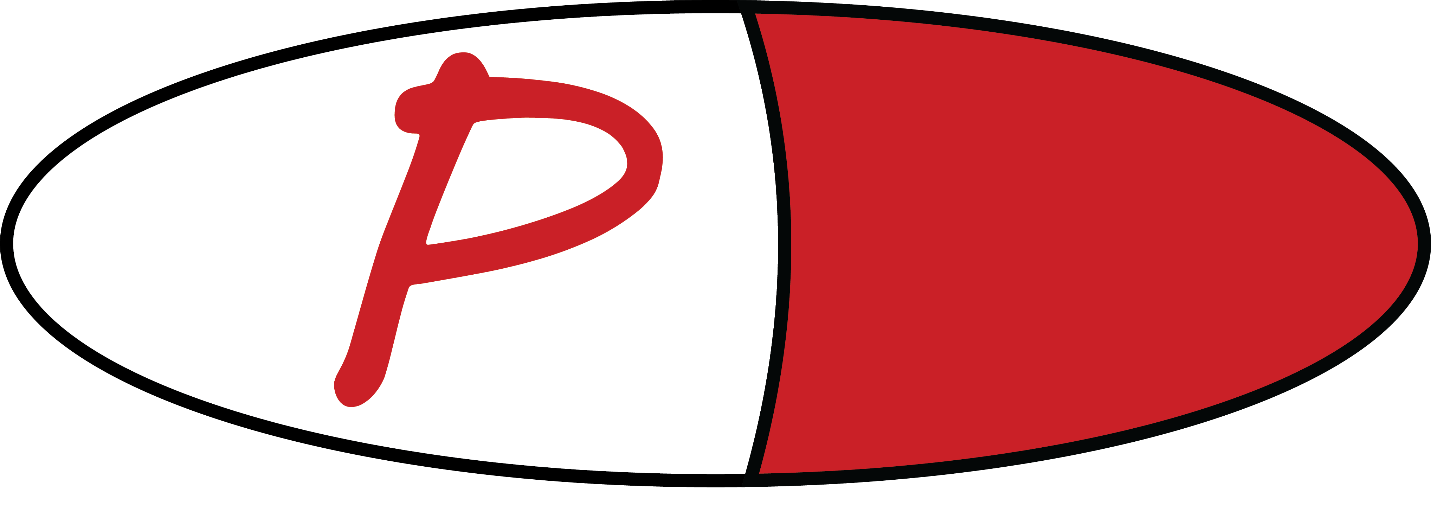
**Requirements Engineering Report**

**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. Requirements Class Models
   6. Data Directory
   7. Limitations & Constraints
   8. Non-functional Requirements
4. Design Models
   1. Graphical Model of Architectural Design Identifying Subsystems
   2. Textual Descriptions of Architecture and Subsystems
   3. Subsystem Design
      1. UML Class Diagrams
      2. User Interface Design
5. Test Plan
6. Major Software Functions
   1. Implemented
   2. Not Implemented
7. Problems Encountered
8. Bibliography
9. 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 will 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
* Spotify
* Classmates
* Professor Guinn

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Information | Simple FP | Average FP | Complex FP | Total  FP |
| User Input | 9 | 4 | 6 | 19 |
| User Output | - | 10 | - | 10 |
| User Online  Queries | - | 8 | - | 8 |
| Logical Files | 7 | - | - | 7 |
| External Interface | - | 21 | - | 21 |
| Total Function Points | - | - | - | 65 |

|  |  |
| --- | --- |
| Reliable Backup and Recovery | 2 |
| Specialized Data Communication | 4 |
| Distributed Processing Functions | 0 |
| Critical Performance | 1 |
| Operation in Heavily Utilized Operational Environment | 0 |
| Online Data Entry | 3 |
| Online Data Entry via multiple screens | 0 |
| ILFs updated online | 2 |
| Inputs, Outputs, Files, or Inquiries Complexity | 5 |
| Internal Processing Complexity | 4 |
| Reusable Code Design | 3 |
| Conversion and Installation Inclusion | 0 |
| Designed for Multiple Installations in Different Organizations | 0 |
| Facilitate Change and Ease of Use | 5 |
| **Total** | 29 |

Total Function Points = FP = 65 X [0.65 + 0.01 X 29] = 61.5

* + 1. Lines of Code Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| Functions | Good LOC Est. | Average LOC Est. | Bad LOC Est. |
| GUI | 250 | 500 | 750 |
| Database/API  Queries | 150 | 300 | 450 |
| Backend | 500 | 700 | 900 |
| Log-in Support | 200 | 400 | 600 |
| Estimated LOC  (\*Based on Average Case\*) |  | 1900 |  |

With the average productivity rate of 620 LOC/person-month, we can complete 1,860 LOC/month. Assuming $8,000 per person-month of labor the cost using LOC estimation is $24,000 a month. This equates to roughly $12.90 per LOC, with this the total cost of the project will be $24,510.

* + 1. Cost Estimation

**COCOMO** **II**

|  |  |  |  |
| --- | --- | --- | --- |
| Objects | Quantity | Weight | Total |
| Screens | 4 | 4 | 16 |
| Reports | 3 | 6 | 18 |
| Components | 4 | 10 | 40 |
|  |  | Total OP | 74 |

NOP = (OP) X [(100 - %Reusable Code)/100]

NOP = 74 x [(100-10)/100] = 66.6

PROD = NOP/Person-Month

Effort = 66.6/13 = 5.12 Person-Months

Assuming a labor rate of $8,000 per person-month, the COCOMO II model estimates that this project will cost $40,960.

* 1. Risk Plan

**Impact Levels**

**1**-Negligible

**2**-Moderate

**3**-Severe

**4**-Catastrophic

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Probability | Impact | Mitigation |
| Team member Leaving | 15% | 4 | Redistribute workload. Possible loss of minor features. |
| Falling behind | 25% | 2-3 | Meet and re-planwork schedule to accommodate work load. |
| Lack of skill | 25% | 2-3 | Find help, watch videos, ask someone who can help us. |
| Spotify Servers Going Down | 10% | 4 | Integrate a different api (Last.fm, Pandora, etc.) |

* 1. 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. Server Functions (No Agent)

4.1 Login Authentication

4.2 Search Caching

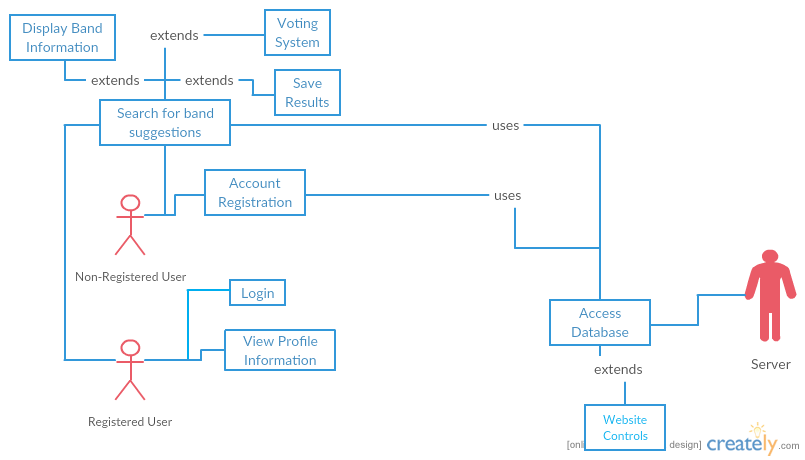
4.3 Information Retrieval

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 in the database.

**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 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:**

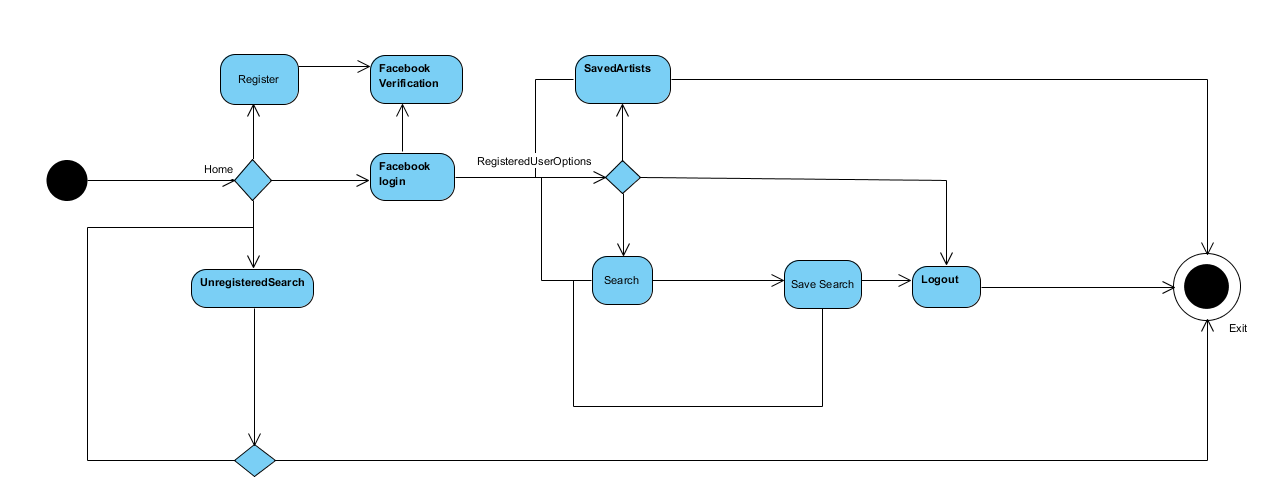
1. The user clicks a profile button that will take them to their account page
2. Server brings user to that page and displays the user’s information

**Exceptions:**

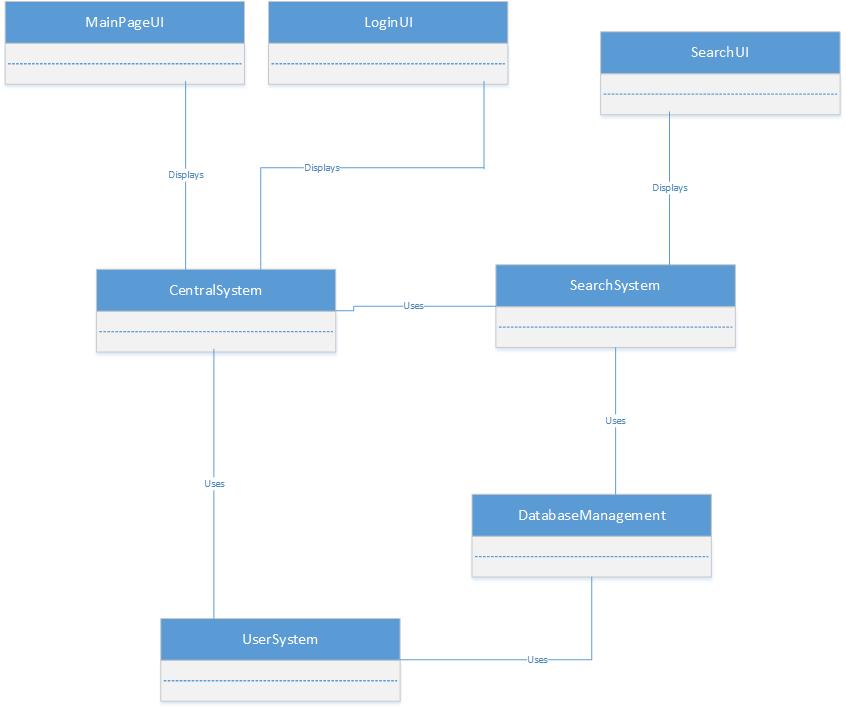
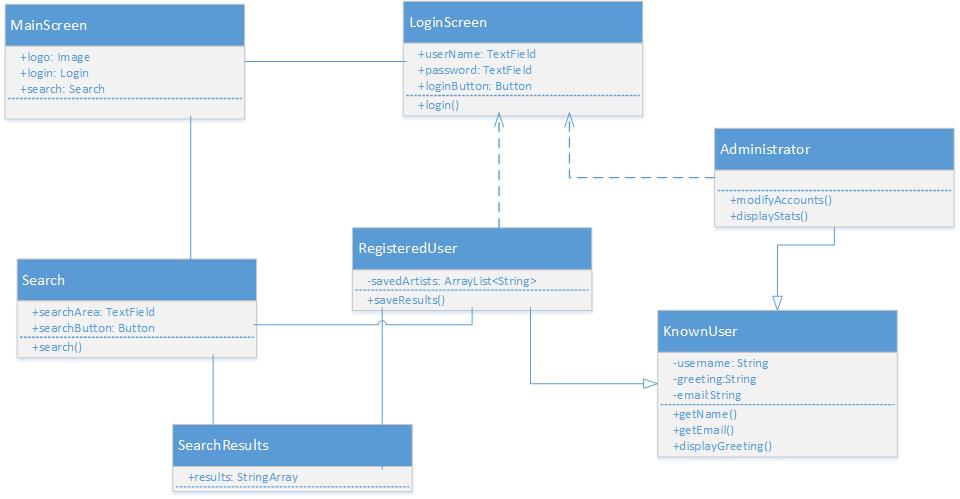
* The user does not view account information.
* The user is not logged int.

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

* 1. Activity Diagrams



* 1. Requirements Class Models



* 1. Data Directory

Administrator – Project members will have access to unique controls allowing them access to the user and data databases as well as control over the website’s interface.

Non Registered User – Any person who is using the website’s functions but is not logged in.

Registered User - Any person using the website’s functions while being logged into an account.

Saved Searches – Searches that a registered user has saved to their account for later viewing.

Rating System – User is giving their input on the relevancy of the search results.

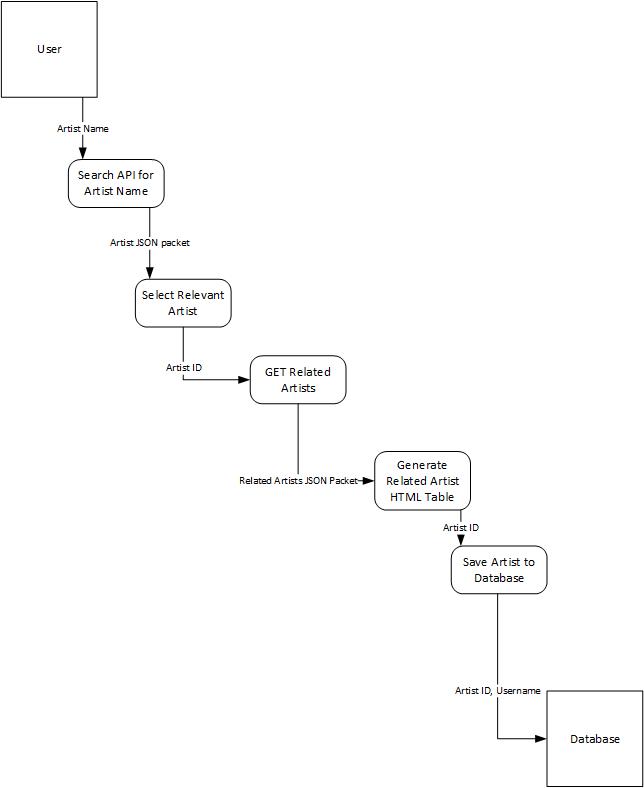
* 1. Limitations & Constraints

**Constraints**: Users’ must have an up-to-date internet browser to use all the of website’s functions.

* 1. Non-functional Requirements

Security is an important thing to take into consideration in this project as there are options given only to registered users. Our solution to this is to use Facebook integration allowing Facebook to handle the user’s information. As for usability we would like the user to be able to understand what options they have and how to use all the functions of our website without having to read too much. To do this we have designed a minimalistic interface that only shows important features to the user such as the search bar, login button and the logo. We also want to make sure our website is reliable, although some things are out of our control we have made sure that all aspects of the website are available and will respond within a few seconds by using the Spotify API and integrating Facebook’s login features.

1. Design Models
   1. Graphical Model of Architectural Design Identifying Subsystems



* 1. Textual Descriptions of Architecture and Subsystems

The focal point of the Prescribe software is the transformation of an artist name that the user types into a search bar into a dynamically generated table on the web page containing a list of Artist recommendations based on their input. This is accomplished through a series of calls and relays to the Spotify API. Due to the data-transforming nature of the problem, the architecture of the software is best realized using the Data-Flow architecture model.

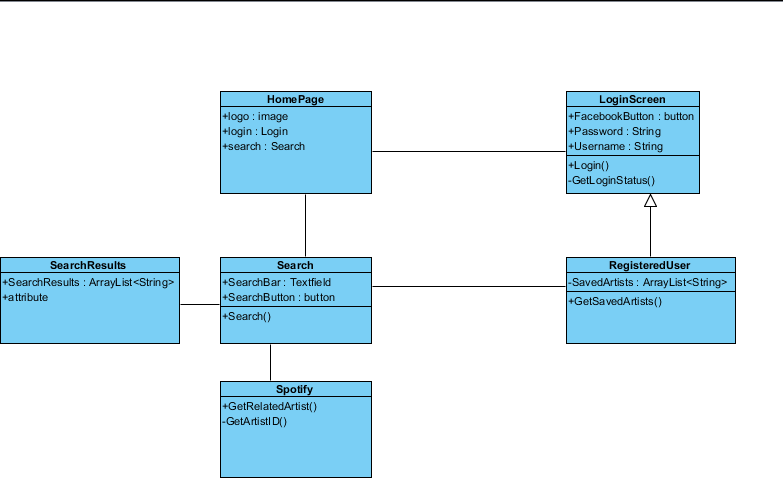
The Data-Flow process is initiated when the user hits the Enter key or the search button on the Prescribe home page. When the user submits their data, it is passed on to a JavaScript function which formats the string into a format where it can be used to search the API. The function then makes an Ajax (Asynchronous JavaScript And XML) GET call to the Spotify API which in turn will return a JSON formatted list of Artists that closely match the string the user entered. Each artist object that is returned by the API at this stage has three important pieces of information: Artist Name, Popularity, and ArtistID. First, Prescribe considers the Artist Names of the Artists returned in the search. If there is an Artist whose name is an exact match with the string the user inputted (ignoring case) then that artists ArtistID function is passed on to the next process. Should no artist names exactly match the user’s search criteria, then the Popularity field is given priority. The ArtistID of the most popular artist returned from the API will be passed to the next process (or the first artist whose popularity matches the maximum Popularity value of all Artists, should two have the same popularity).

The next process in the data-flow consists of taking an ArtistID (a unique identifier for a musician or band), formatting an appropriate URL, and then querying the API for a list of Artists recommended to the Artist with that ArtistID. There are not a lot of complications at this portion of the data-flow.

Next, the list of related Artists retrieved from the API in the previous step must be processed. There is a table defined in the HTML of the webpage with no elements. The related data about the related Artists is passed to a JavaScript function which dynamically builds the aforementioned table with data from the related artists JSON data. This table will include columns for artist picture, artist name, and a button which if pushed will create a modal pop-up displaying the biography of the artist in question.

If the user is registered and logged in, there is another step in the data-flow process that is available to them. They may choose to save a particular artist from the recommended artist table that interests them. Doing this will entail sending a POST-method call to the User-Artist database to save an entry with the user’s identifier, the artist’s ArtistID, and the date. This saved data can be retrieved later at the user’s convenience in a similar format to the original recommendation table.

* 1. Subsystem Design
     1. UML Class Diagrams



* + 1. User Interface Design

The main user bar of our site will have three buttons that perform single actions. These three buttons will be housed at the top of the page. Also, this bar will be displayed on every page in the app. One which will be labeled “Home”. This one will take the user to the home screen where they can search. The second, “About”, will take the user to a page with the team members’ names, their jobs, and a description of the webapp. The third will run a popup that allows them to sign in via Facebook. Once the user has signed into their profile, the login button will be replaced by a button that takes the user to their saved bands.

On the main page, there will be a search bar located in the middle of the page. A user is able to input any string of characters into this to search. There will also be a button labeled “Search” which will perform the search action.

Once a search has been completed, the user will be taken to a results page. On this page will be a list of related artists. This list will be displayed on a table that includes the name, the picture, a button to save the artist, and a link to the artist’s bio. If a user clicks the bio link, a popup will display over the main screen that displays a certain artist’s bio.

On the user’s page, a list of same bands will be displayed with the same format as the search result page. The “save artist” button will be replaced with a “remove artist” button.

1. Test Plan
2. Major Software Functions
   1. Implemented
      1. Artist Search
      2. Display Artist Biography
      3. Save Search Results (if logged in)
      4. Register
      5. Login
   2. Not Implemented
      1. Artist Rating System
3. Problems Encountered

* In calculating Function Points, it was unclear what to count at User inputs and what to count as user online queries. Additionally, it was not clear how to distinguish between External Interfaces and User Online Queries
* Unclear what Logical files are
* Cannot install Visio or project to laptop. Had to use online tools and Microsoft office
* Spotify removed the ability to get an artist’s biography through the Echo nest API and stopped issuing API key’s while they go through some transitions.

1. Bibliography
   1. Bootstrap
   2. Spotify API
   3. Wikipedia API
   4. Flask
   5. PeeWee