Software Requirement Specification

CRATE

April 17th, 2018

Version [1.5]

**Presented To:**

**Donna DeMarco**

**Submitted By:**

Daniel Carman, Nina Schnyder, Dana Fidler, Roksana Ferkh

**Table of Contents**

[**Revision History**](#_vqzfpkca7atq) **3**

[**1.0 Introduction**](#_lhsbqlf26ymp) **3**

[**1.2 Scope**](#_wmvt2ngdgf9j) **4**

[**3.0 Requirements**](#_sag2mkni8uet) **5**

[**3.0.1 Punch List**](#_hn97l5umtbhv) **5**

[**3.1 Account Functions**](#_cp00yg4qrbp9) **6**

[**3.1.1 Customer Sign up**](#_qrpx7jodhdtg) **6**

[**3.1.2 Store Owner Sign up**](#_7qfg0hnfetjf) **7**

[**3.1.3 Sign in**](#_rh5ci8qkmm65) **8**

[**3.2 Customer Application Function**](#_cuzjnc506w7u) **9**

[**3.2.1 Search by Keyword**](#_9o9rzvsva3jv) **9**

[**3.2.2 Search by Location**](#_na2lfuzfckdp) **10**

[**3.2.3 Update WishList**](#_yy35voca91bb) **10**

[**3.2.4 Favorite Stores**](#_64rab5d0bb0k) **11**

[**3.2.5 Rate Stores**](#_weu2pwinip38) **11**

[**3.3 Store Owner Functions**](#_x175d6wpn8mf) **12**

[**3.3.1 Connect Social Media**](#_qfgt60xnbztq) **12**

[**3.3.2 Make Posts**](#_ut3q1bevucse) **13**

[**3.3.3 Adding Records to Inventory**](#_rf26xhz91zda) **13**

[**3.3.4 Updating Records in Inventory**](#_mi4hjb4xkjht) **14**

[**3.3.5 Removing Records in Inventory**](#_pf49ih33u7t2) **15**

[**3.4 Admin Functions**](#_61ohzur1azv0) **16**

[**3.4.1 Remove Account**](#_e5uvx9g3beo8) **16**

[**3.4.2 Updating Records in Inventory**](#_le0547rkd90p) **17**

[**3.4.3 Removing Records in Inventory**](#_jy35ue1u4msr) **18**

[**3.5 Social Media Functions**](#_253wa2emghqi) **19**

[**3.5.1 Pulling Information from Social Media**](#_7hm9p4yt7j56) **19**

[**4.0 Non Functional Requirements**](#_sqw06bahpxg) **20**

[**4.1 Other Systems**](#_vx2l5ip3q4qd) **20**

[**4.2 Security**](#_tmphvcs364ep) **20**

[**4.3 Budget**](#_8dn6kl6twwtu) **20**

[**4.4 Miscellaneous App Store Requirements**](#_7mcvhrm46t3w) **20**

[**4.5 Miscellaneous Google Play Requirements**](#_h93f5nhz8bq) **20**

[**5.0 Architecture**](#_jjurf4c773ts) **21**

[**5.1 Computers**](#_59b732nqsysl) **21**

[**5.2 Database Server**](#_xf48kto4b1bj) **22**

[**5.3 Web Server**](#_pe796ewyo127) **22**

[**5.3 Phones**](#_4yvqyzltjsw) **22**

[**Signature Page**](#_1t3h5sf) **23**

### **Revision History**

| **Date** | **Author** | **Distributed to** | **Version** |
| --- | --- | --- | --- |
| Mar 27, 2018 | All | class | 1.0 |
| Apr 3, 2018 | All | class | 1.1 |
| Apr 5, 2018 | All | class | 1.2 |
| Apr 10, 2018 | All | class | 1.3 |
| Apr 13, 2018 | All | Prof. Donna Demarco | 1.4 |
| Apr 17, 2018 | All | Prof. Donna Demarco | 1.5 |

# 

# **1.0 Introduction**

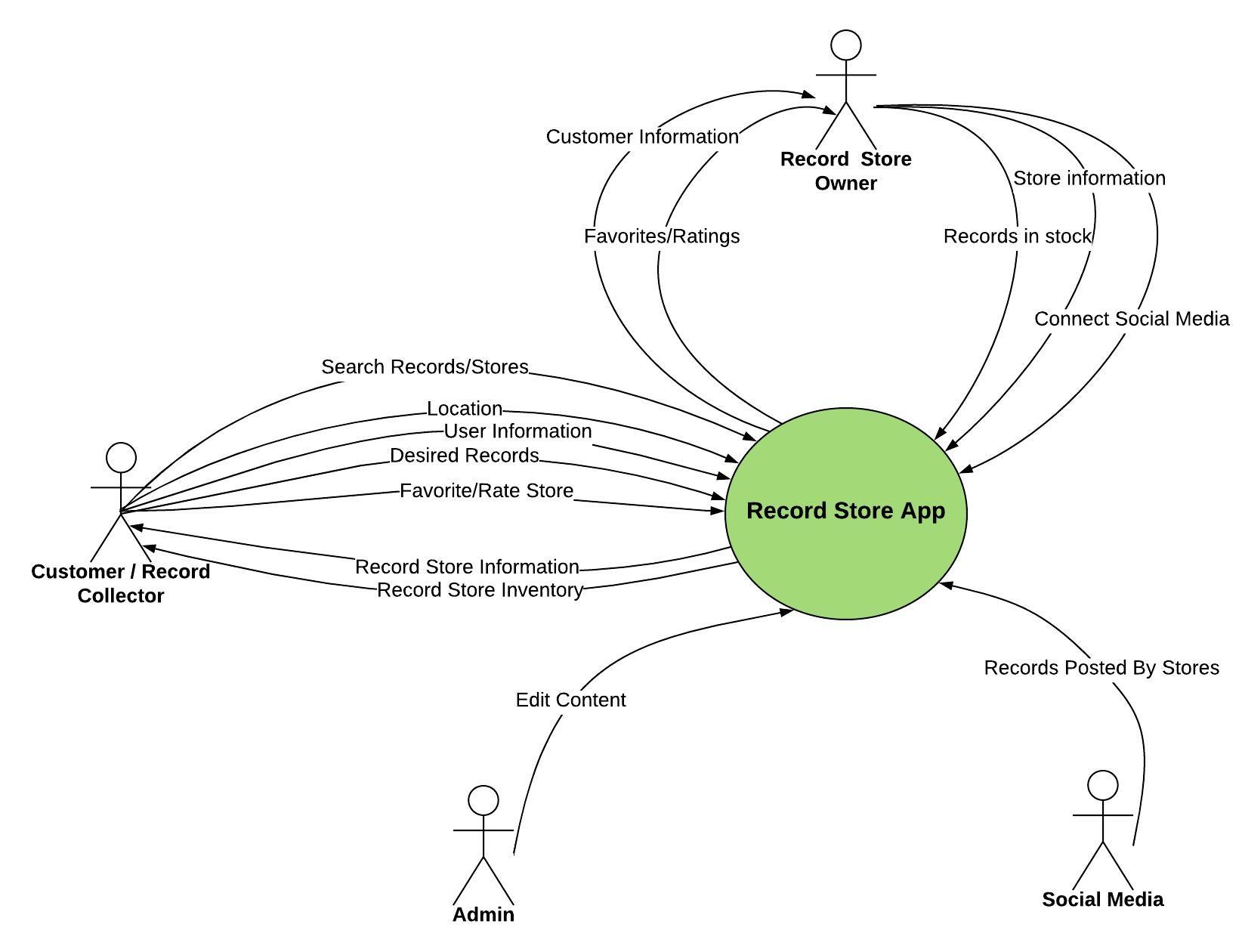
Finding record sellers in a local area can sometimes be a challenge, especially considering some places that sell them are not specifically record stores. On top of that, knowing what these stores have in stock is largely impossible unless you follow every single social media post they make. Our goal is to make an application that bonds record store collectors to record store owners to make it easier to find certain albums users are interested in purchasing.

The potential audience for this app would be anyone who has a passion for records and collecting and wants to have easier access to record shops and their location. Our audience can create an account depending if they are a record store owner or just a collector. If the user wishes to create a store owner account, they can upload records from their shop and connect with collectors who are interested.

If a user wishes to create a collector account they can connect with other users and create a wishlist on the app. They then can be matched to local records stores that have similar inventory as the user has on their wishlist. The collector can also browse pictures, updates, and any deals/sales posted by record stores.

The goal of our project is to develop a record store application which will allow users to collect and have access to record stores locations. The record store app should allow the record stores to create account and upload information about the records they have and allow the users to create account and have the ability to add the records they like to their wishlist. The record store app should reach out to a wide range of users with the reliable information about the records that the records stores will provide.

# **1.2 Scope**



**Store Owners:** Users who are going to post information about the records in stock and general information about their business.

**Users/Customers:** Users who are interested in collecting records and finding new record stores in their area.

**Admin:** A user verifies the record store information to provide reliable data for the rest of the users.

**Social Media:** The social media account of the record store where they can get specific record information from.

**2.0 Related Documents**

[Software Development Plan link](https://drive.google.com/open?id=1da10ZOwGRPlHTIO06fhbYOuV0TBioyY-sqNClwaC4qc)

[Requirements Traceability matrix link](https://docs.google.com/spreadsheets/d/1p9GvC-WUyqMa4jHuWd21f7b-LDsmxnp8gulYSWjS_7Y/edit#gid=0)

# **3.0 Requirements**

As a Customer I want to…

* Create Account
* Login
* Search for:
  + Stores
  + Albums
  + Artists
  + Labels
  + Genres
* Favorite Stores
* Update Wishlist

As a Store Owner I want to...

* Create Account
* Login
* Edit Inventory
* Create Posts on Crate homepage regarding sales, new stock, etc.
* Connect Social Media

As an Admin I want to...

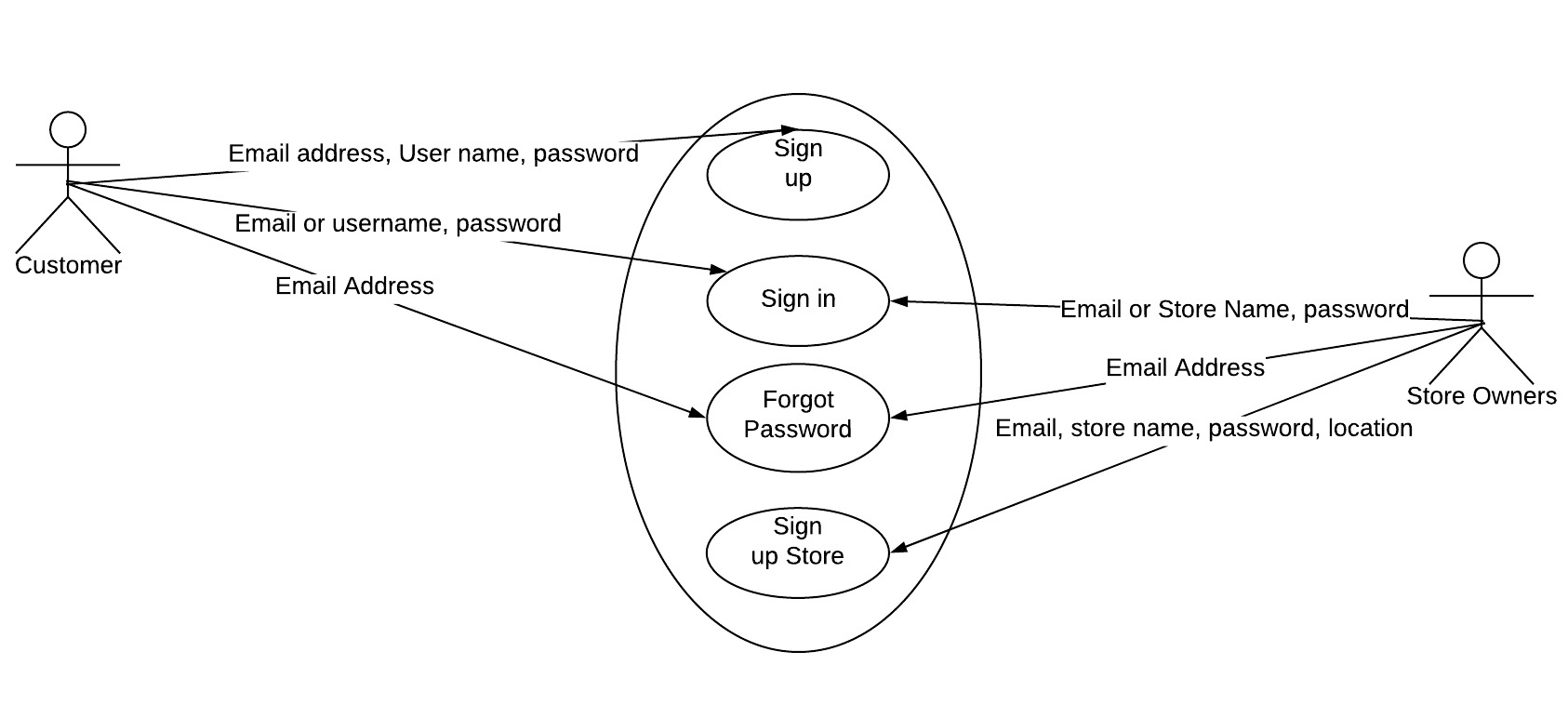
* Edit Store Inventory
* Edit created stores
* Ban Users

## 3.0.1 Punch List

The following list are the items still to be resolved:

1. approve trusted users? who could also be moderators or have other privileges in the future

# 3.1 Account Functions



## 3.1.1 Customer Sign up

**Description:** In order to use the search, create a wishlist, favorite stores etc., customers must sign up for a free customer account in the Crate app. Displayed on the sign in page are multiple forms that must be filled out to create the account. The forms are: Email, Verify Email, Username, Password, Verify Password. There are also two options, one to submit a completed form, and one to go back to the previous page.

**Use Case**:

1. **Name**: Customer Sign up
2. **Participating Actor(s)**: Customers
3. **Entry**: App is downloaded
4. **Exit**: Account is created
5. **Flow**:
   1. Customer enters data
      1. Email Address
      2. Username
      3. Password
   2. Account is created
6. **Special Requirements**: If data is invalid, display error message and try again as many times as they want.

## 3.1.2 Store Owner Sign up

**Description:** To sign up for a store owner account, the user must enter information into a form (Email Address, Store name, Password, Store location). Once completed, the form is submitted to be validated. If validated the account is created, if not, an error message is displayed and the user may try again.

**Use Case:**

1. **Name:** Store Owner Sign up
2. **Participating Actor(s):** Store Owners
3. **Entry:** App is downloaded
4. **Exit:** Account is created
5. **Flow:**
   1. Store owner enters data
      1. Email Address
      2. Store Name
      3. Password
      4. Store Location
         1. Address
         2. Town
         3. Description
   2. Account is created

i. Confirmation email

1. **Special Requirements:** If data is invalid, display error message and try again as many times as the user wants.

## 3.1.3 Sign in

**Description:** To log in to a previously created account, users must provide the correct login information. If the login information is correct, the user is signed in and sent to their home screen. If the information is incorrect, an error message is displayed prompting the user to try again, or sign up for a new account if they are a new user.

**Use Case:**

1. **Name:** Sign in
2. **Participating Actor(s)**: Customers and Store Owners
3. **Entry**: Account has been created
4. **Exit**: User is sent to their home screen
5. **Flow**:
   1. Login info is entered
      1. Customer
         1. Email or Username
         2. Password
      2. Store Owner
         1. Email or Store Name
         2. Password
   2. User is logged in and sent to home screen
6. **Special Requirements**: If information is invalid, display error message prompting user to try again/sign up for a new account. Signing in to one account is limited to 5 attempts before the account is locked and the user must unlock using their email.

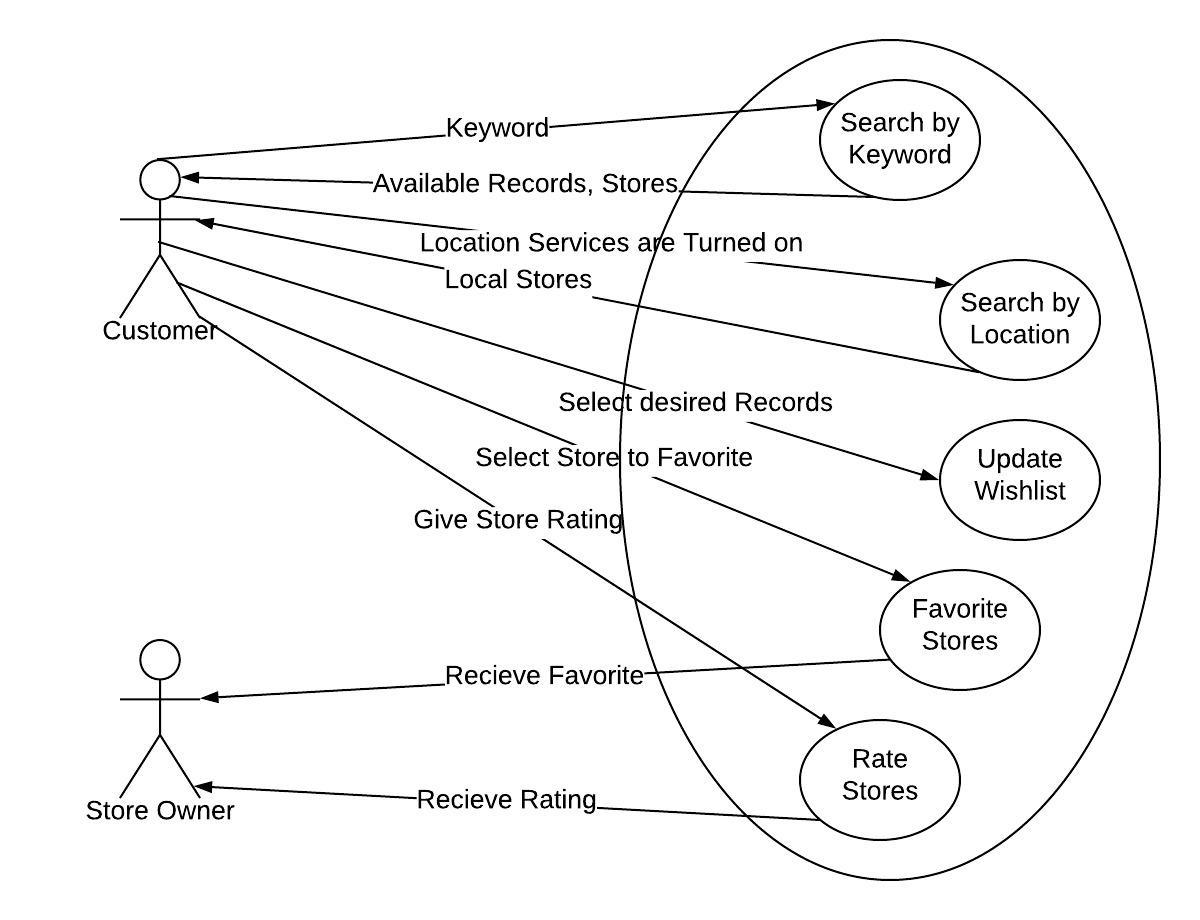
3.1.4 Forgot Password

**Description:** In the case that a user has forgotten their password for their account, they can request a password reset to be sent to their email they signed up for the app with.

**Use Case:**

1. **Name:** Forgot Password
2. **Participating Actor(s)**: Customers and Store Owners
3. **Entry:** Account exists and user has forgotten password
4. **Exit:** User’s account has a new password
5. **Flow:**
   1. User navigates to forgot password form
   2. User selects for email to be sent
   3. User clicks link in email
   4. User creates new password
6. **Special Requirements:** None

# 3.2 Customer Application Function



## 3.2.1 Search by Keyword

**Description:** The users can search for many things on the app, including artists, albums, genres, labels, years, and stores.

**Use Case**:

1. **Name**: Search by Keyword
2. **Participating Actor(s)**: Customers
3. **Entry**: User is logged in to a customer account
4. **Exit**: Search results are returned
5. **Flow**:

a. Customer enters terms key to search for

* + 1. Store name
    2. Album
    3. Artist
    4. Genre
    5. Label
    6. Year
  1. Term is validated
  2. List of results is returned

1. **Special Requirements**: If data is invalid, display error message and try again as many times as they want.

## 3.2.2 Search by Location

**Description:** The user may also search for stores around them by choosing what type of store they want and the radius around them they wish to search in. This feature would give the user stores in their area based on the mile radius around them they had selected.

**Use Case**:

1. **Name:** Search by Location
2. **Participating Actor(s):** Customer
3. **Entry:** User is logged into a customer account
4. **Exit:** Search results are returned
5. **Flow:**
   1. Customer chooses what type of store(s) to search for
      1. Dedicated Record Store
      2. Thrift Shop
      3. Antique Store
      4. Other
   2. Customer chooses the radius around them they wish to search in
   3. List of results are returned
6. **Special Requirements:** None

## 3.2.3 Update WishList

**Description:** Users can find the items that they are looking for and add it to the wishlist or remove items from the wishlist. By doing so they are able to be notified if there is a record in their location that matches their record list.

**Use Case**:

1. **Name**: Update wishList
2. **Participating Actor(s)**: Customers
3. **Entry**: User is logged in to a customer account
4. **Exit**: Wish list is updated
5. **Flow**:
   1. Customer finds item they want to add to list
      1. Select add to wishlist
   2. Item is added to account’s wish list
   3. Customer removes items from wishlist
      1. Select Remove item
6. **Special Requirements**: No special conditions

## 3.2.4 Favorite Stores

**Description:** The users can search for their favorite stores on the app and add it to favorites.

**Use Case**:

1. **Name**: Favorite Stores
2. **Participating Actor(s)**: Customers, Store Owner
3. **Entry**: User is logged in to a customer account
4. **Exit**: Store is added to favorites
5. **Flow**:
6. Customer finds store they wish to add to favorites
7. Store name

b. Store name exist

c. Store is added to account’s favorites

1. **Special Requirements**: If store is already in favorites, user has option to unfavorite and remove from the account’s favorites.

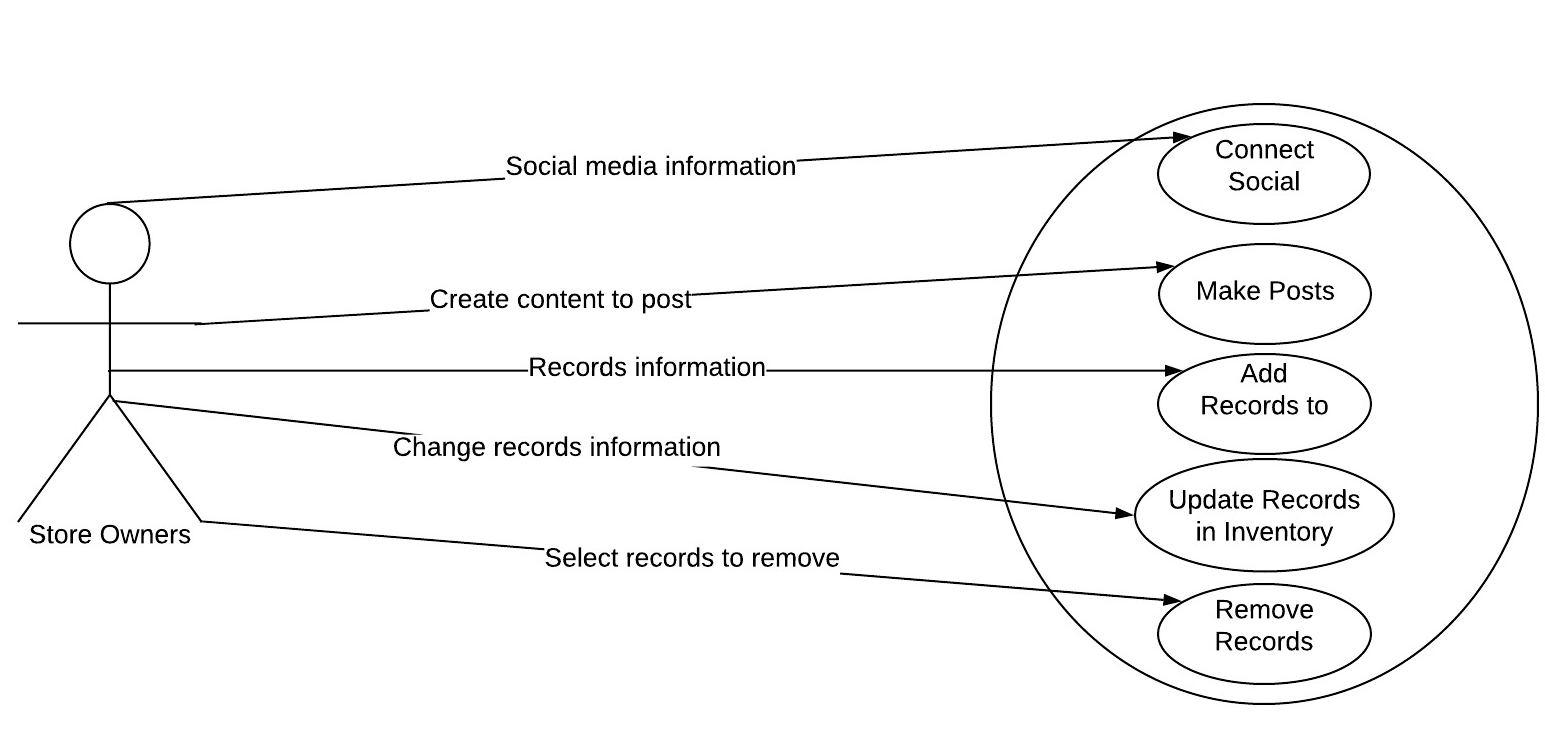
## 3.2.5 Rate Stores

**Description:** The user can rate the stores they have visited based on their experience.

**Use Case**:

1. **Name**: Rate Stores
2. **Participating Actor(s)**: Customers, Store Owner
3. **Entry**: User is logged in to a customer account
4. **Exit**: Store is rated.
5. **Flow**:
   1. Customer find store they want to rate
      1. Store name
   2. Store name exists
   3. Store is rated
6. **Special Requirements**: If the user has already rated the store there is opportunity to change rating or delete rating.

# 3.3 Store Owner Functions



## 3.3.1 Connect Social Media

**Description:** Store owners can connect their social media in order to have the app scrape(take information from) their posts and update their inventory appropriately

**Use Case:**

1. **Name**: Connect Social Media
2. **Participating Actor(s):** Store Owners
3. **Entry:** Store owner account exists
4. **Exit:** Store owner account and social media account are connected
5. **Flow:**
   1. Store Owner navigates to the connect social media page
   2. Store owner chooses social media account they wish to connect
      1. Facebook
      2. Instagram
      3. Twitter
      4. Tumblr
   3. Store owner is redirected to the sign in page for their chosen social media
   4. Store owner provides login information for social media account
      1. Email or Username
      2. Password
6. **Special Requirements:** If the account sign in information is invalid, the user is shown an error message and has the option to try again.

## 3.3.2 Make Posts

**Description:** Store owners can make posts to the app when they have a new records, new sale, and any business update.

**Use Case**:

1. **Name**: Make Posts
2. **Participating Actor(s)**: Store owners
3. **Entry**: User is logged in to a store owner account
4. **Exit**: New post is added to feed
5. **Flow**:
   1. Post information is added
      1. Text
      2. Pictures
      3. Hyperlinks
   2. Post is added to account’s store page
6. **Special Requirements**: No special conditions

## 3.3.3 Adding Records to Inventory

**Description:** Store Owners have the opportunity to keep a virtual inventory for other users to see. They can add contents to keep a running list of what they have available in their store.

**Use Case:**

1. **Name:** Adding record information to the store owner’s inventory
2. **Participating Actor(s)**: Store Owners
3. **Entry**: Store Owner has created and logged into their account
4. **Exit**: Item has been added to the user’s inventory
5. **Flow**:
   1. Store owner enters inventory record to be added
      1. Artist
      2. Album name
      3. Year
      4. Condition
      5. Price
      6. Label
      7. Type
      8. Picture of Cover
   2. Store owner reviews record’s contents
   3. Store owner submits record to owner’s virtual inventory.
   4. Store owner’s record is submitted to virtual inventory.
6. **Special Requirements**: If user tries to enter a record without the required information, an error will occur.

## 3.3.4 Updating Records in Inventory

**Description:** Store Owners have the opportunity to keep a virtual inventory for other users to see. They can update its contents to keep a running list of what they have available in their store.

**Use Case:**

1. **Name:** Updating record information to the store owner’s inventory
2. **Participating Actor(s)**: Store Owners
3. **Entry**: Store Owner has created and logged into their account and a record is successfully added in their inventory.
4. **Exit**: Record is updated and re-added to virtual inventory.
5. **Flow**:
   1. Store owner searches for record they wish to update.
   2. Store owner selects record they wish to update.
   3. Store owner updates record information on..
      1. Artist
      2. Album name
      3. Year
      4. Condition
      5. Price
      6. Label
      7. Type
      8. Picture of Cover
   4. Store owner reviews new record’s contents
   5. Store owner submits record to owner’s virtual inventory.
   6. Store owner’s record is submitted to virtual inventory.
6. **Special Requirements**: If user tries to update a non existent item, error message is returned and user can try again.

## 3.3.5 Removing Records in Inventory

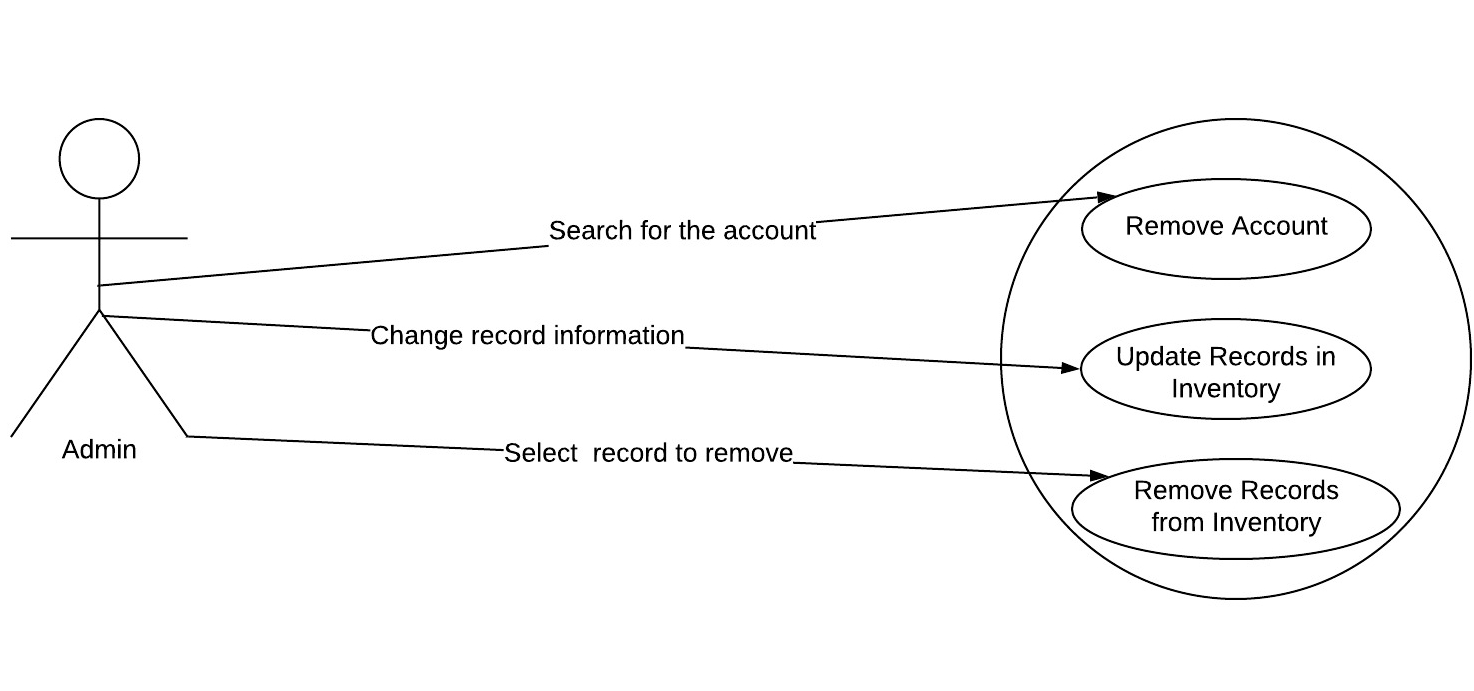
**Description:** Store Owners have the opportunity to keep a virtual inventory for other users to see. They can remove contents to keep a running list of what they have available in their store.

**Use Case:**

1. **Name:** Removing record information to the store owner’s inventory
2. **Participating Actor(s)**: Store Owners
3. **Entry**: Store Owner has created and logged into their account and a record is successfully added in their inventory.
4. **Exit**: Record is removed from virtual inventory.
5. **Flow**:
   1. Store owner searches for record they wish to remove.
   2. Store owner selects record they wish to remove.
   3. Store owner selects remove record from virtual memory
   4. Store owner is prompted if they wish to remove record.
   5. Store owner removes record from virtual memory.
6. **Special Requirements**: If user tries to remove/update a non existent item, error message is returned and user can try again.

# 

# 3.4 Admin Functions



## 3.4.1 Remove Account

**Description:** In some cases, an account may need to be deleted. This can be due to the user requesting deletion of the account, or because the user has been posting malicious/facetious content.

**Use Case:**

1. **Name:** Remove Account
2. **Participating** **Actor(s)**: Admin
3. **Entry:** User looking at user account is logged in to an admin account
4. **Exit:** User account is deleted
5. **Flow:**
   1. Admin navigates to user account page to be deleted
   2. Admin clicks delete account button
   3. Admin confirms they wish to delete the account
   4. The account is deleted
6. **Special Requirements:** none

## 3.4.2 Updating Records in Inventory

**Description:** Admin have the opportunity to keep a virtual inventory for other users to see. They can update its contents to keep a running list of what they have available in their store.

**Use Case:**

1. **Name:** Updating record information to the store owner’s inventory
2. **Participating Actor(s)**: Admin
3. **Entry**: Store Owner has created and logged into their account and a record is successfully added in their inventory.
4. **Exit**: Record is updated and re-added to virtual inventory.
5. **Flow**:
   1. Admin searches for record store that needs a record updated
   2. Admin searches for record they wish to update in store’s virtual inventory.
   3. Admin selects record they wish to update.
   4. Admin updates record information on..
      1. Artist
      2. Album name
      3. Year
      4. Condition
      5. Price
      6. Label
      7. Type
      8. Picture of Cover
   5. Admin reviews new record’s contents
   6. Admin submits record to owner’s virtual inventory.
   7. Store Owner’s record is submitted to virtual inventory.
6. **Special Requirements**: If admin tries to update a non existent item, error message is returned and user can try again.

## 

## 

## 3.4.3 Removing Records in Inventory

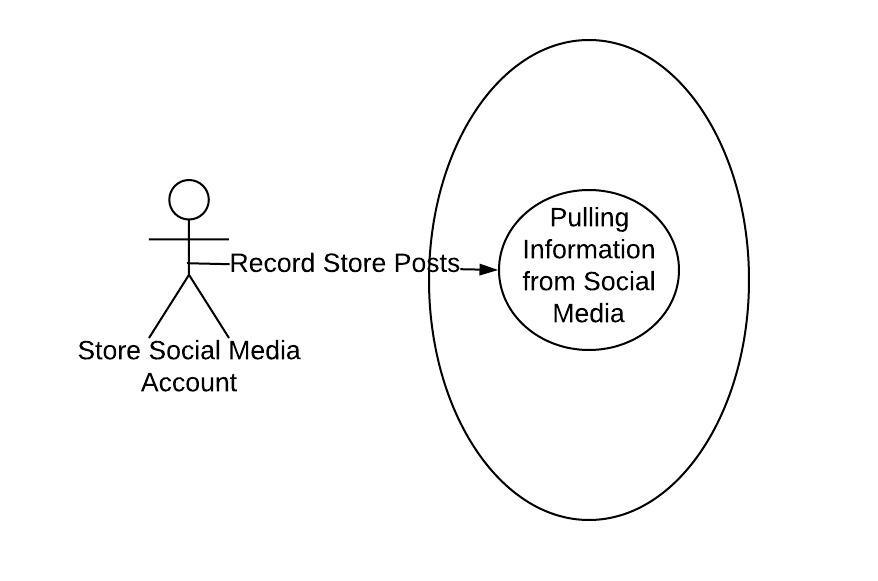
**Description:** Admin have the opportunity to keep a virtual inventory for other users to see. They can remove contents to keep a running list of what they have available in their store.

**Use Case:**

1. **Name:** Removing record information to the store owner’s inventory
2. **Participating Actor(s)**: Admin
3. **Entry**: User viewing inventory of store is logged in to an admin account.
4. **Exit**: Record is removed from virtual inventory.
5. **Flow**:
   1. Admin searches for record they wish to remove.
   2. Admin selects record they wish to remove.
   3. Admin selects remove record from virtual inventory
   4. Admin is prompted if they wish to remove record.
   5. Admin removes record from virtual inventory.
6. **Special Requirements**: If user tries to remove/update a non existent item, error message is returned and user can try again.

# 

# 3.5 Social Media Functions



## 3.5.1 Pulling Information from Social Media

**Description:** Once the store page is connected to a social media, the app will use technology to pull information from the account’s posts in order to add them to the store’s inventory in the app.

**Use Case:**

1. **Name**: Pulling Information from Social Media
2. **Participating Actor(s):** Store Social Media Account
3. **Entry**: Store owner account is connected to social media
4. **Exit:** Item is added to inventory
5. **Flow**:
   1. Social media post is made
   2. App gathers information from social media page
      1. Artist
      2. Album
      3. Genre
      4. Year
      5. Label
   3. Information is added to store’s inventory as a new item
6. **Special Requirement**: None

# **4.0 Non Functional Requirements**

## 4.1 Other Systems

Database Server to store information from app for access by users.

Navigation/location services to locate user and local stores around users.

Plugins from selected social media accounts to connect to store user accounts.

## 4.2 Security

The App Store, a place where Apple users can downloads apps submitted by developers, enforces a strict protocol regarding ATS, or App Transport Security. App Transport Security forces the app to connect to websites through HTTPS instead of HTTP, which encrypts users data while in Transit. ATS requires Transport Layer Security. The Google Play store also requires HTTPS over HTTP. There is no Play Store specific protocol comparable to ATS, just the requirement of HTTPS.

## 4.3 Budget

Costs for buying/renting and upkeeping the database server

## 4.4 Miscellaneous App Store Requirements

The app store requires all apps to have a way of detecting inappropriate/offensive content and blocking it. The banning of offenders is also required. Apps collecting information about the user and sharing it must post information about that in the privacy policy. Only public APIs may be used. App must be compatible with IPv6

## 4.5 Miscellaneous Google Play Requirements

User information must be transparently stored. Users must be informed of how/if their personal data is being used/shared. App must clearly outline what is not appropriate and have the ability to ban users for violating this policy.

# **5.0 Architecture**

## 

## 5.1 Computers

1. Developing the App
2. Creating user interface
3. Connecting to the database and web server
4. Downloading App to...
   1. iPhone
   2. Samsung

## 5.2 Database Server

1. Hold user Login info
2. Hold store inventories
3. Hold locations of stores
4. Deliver held information to web server when requested

## 5.3 Web Server

1. Interface between phone users and database
2. Allow users to navigate pages
3. Operate user functions

## 5.3 Phones

1. iPhone (IOS)
2. Samsung (Android)

### 

### 

### 

### 

### 

### **Signature Page**

AGREED TO:

**Customer?**

**Signature**



**Name:**



**Title:** 

**Date:**

AGREED TO:

**Team members**

**Signature Nina Schnyder**

**Name: Nina Schnyder**

**Title: Team Member**

**Date 04/10/2018**

**Signature Daniel Carman**

**Name: Daniel Carman**

**Title: Team Member**

**Date: 4/10/18**

**Signature Dana Fidler**

**Name: Dana Fidler**

**Title: Team Member**

**Date: 4/10/18**

**Signature Roksana Ferkh**

**Name: Roksana Ferkh**

**Title: Team Member**

**Date: 4/10/18**