Arcane Library

System Requirements Specification

#### **Table of Contents**

Page

## 1. Introduction

* 1. Purpose of This Document
  2. References
  3. Purpose of the Product
  4. Product Scope

2. Functional Requirements

3. Non-Functional Requirements

3.1 Customer Constraints

3.2 External Interfaces

3.3 Other

4. Deliverables

5. Open Issues

## Appendix A – Agreement Between Customer and Contractor

Appendix B – Team Review Sign-off

Appendix C – Document Contributions

## **Document Versioning Control**

| **Version Number** | **Date** | **Changes from Previous Version** |
| --- | --- | --- |
| 1.0 | 12/16 | N/A |

## 1. **Introduction**

1.1 Purpose of This Document

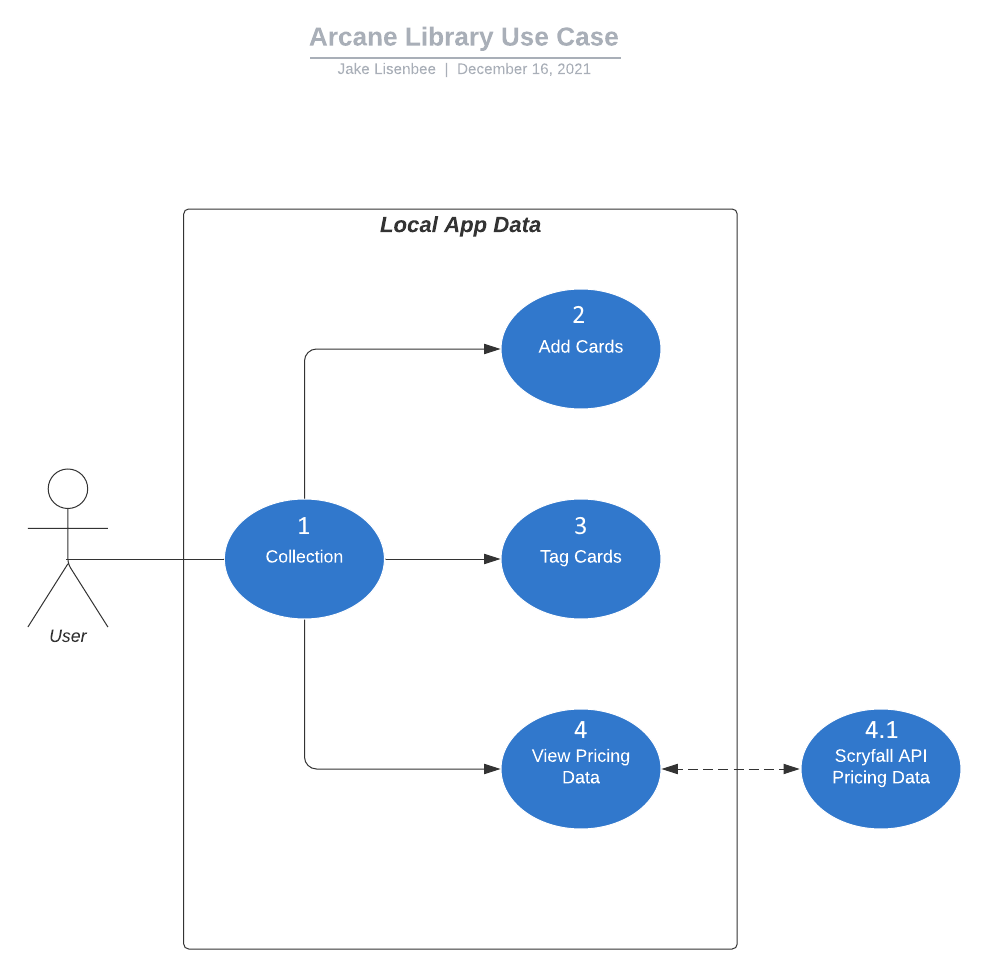
This document outlines to any future or current developers and to the customer the functional expectations of the Arcane Library trading card organizer. In addition to bare functionality, scope, non-functional constraints, and visual aesthetic will be outlined.

1.2 References

N/A

1.3 Purpose of the Product

The purpose of this software is to provide a lightweight and easy to use “at a glance” view of one’s collection of trading cards without needing to sort through each card in one’s collection in case such a situation is unavailable. Most other similar software available is cumbersome and unfocused, or not focused on organization but rather card searching as a database.

* 1. Product Scope

The system is wholly contained on a user’s mobile device with only occasional and optional calls to an online database. The program is lightweight and simple and focuses on simply keeping a record of the cards in a user’s collection.

The intended use is that a User enters the program looking at a collection (1). From there, a user can add cards to that collection (2), Tag existing cards (3), or view the pricing data of cards in that collection (4).

2. **Functional Requirements**

| **Number** | 1 | |
| --- | --- | --- |
| **Name** | View Collection | |
| **Summary** | When a user opens the app, they can view their collection to manipulate it. | |
| **Priority** | 5 | |
| **Preconditions** | The app must be opened | |
| **Postconditions** | The app will display a list of cards in the user’s collection | |
| **Primary Actor(s)** | User | |
| **Secondary Actor(s)** | N/A | |
| **Trigger** | The app is opened | |
| **Main Scenario** | **Step** | **Action** |
|  | Step 1 | The App opens |
|  | Step 2 | The app loads the saved data from the collection |
|  | Step 3 | The App displays the collection |
| **Extensions** | **Step** | **Branching Action** |
|  | Step 3.1 | If loading fails, app will retry 2 times then display an error and ask if you would like to make a new collection |
| **Open Issues** | Issue # | N/A |

| **Number** | 2 | |
| --- | --- | --- |
| Name | Add Cards | |
| **Summary** | A user can add cards to their collection with a search | |
| **Priority** | 4 | |
| **Preconditions** | A collection must be successfully loaded | |
| **Postconditions** | A new card with quantity 1 will be added to the collection | |
| **Primary Actor(s)** | User | |
| **Secondary Actor(s)** | N/A | |
| **Trigger** | The “add card” button is pressed | |
| **Main Scenario** | **Step** | **Action** |
|  | Step 1 | User enters part of a card name into the search bar. |
|  | Step 2 | User selects desired card out of list of predicted requested cards |
|  | Step 3 | Card is added to Collection |
| **Extensions** | **Step** | **Branching Action** |
|  | Step 3.1 | If card is not found with search snippet, no predictive cards will be displayed and no card can be added. |
| **Open Issues** | Issue # | N/A |

| **Number** | 3 | |
| --- | --- | --- |
| Name | Tag Cards | |
| **Summary** | User adds custom tags and a quantity of cards to tag to a card listing | |
| **Priority** | 2 | |
| **Preconditions** | A card listing must be present in a collection | |
| **Postconditions** | A card listing will have a new tag associated with it | |
| **Primary Actor(s)** | User | |
| **Secondary Actor(s)** | N/A | |
| **Trigger** | User Long-Presses on a card listing | |
| **Main Scenario** | **Step** | **Action** |
|  | Step 1 | User long-presses on a card listing |
|  | Step 2 | System pulls up a menu of existing tags and option to add a new one |
|  | Step 3 | User updates an old tag’s quantity, name, or color, or adds a new tag. |
| **Extensions** | **Step** | **Branching Action** |
|  | Step # | N/A |
| **Open Issues** | Issue # | N/A |

| **Number** | 4 | |
| --- | --- | --- |
| Name | Request Pricing | |
| **Summary** | User can opt to request pricing data from the internet | |
| **Priority** | 1 | |
| **Preconditions** | A collection must be successfully loaded | |
| **Postconditions** | Card listings in the collection will have price listed also | |
| **Primary Actor(s)** | User | |
| **Secondary Actor(s)** | Scryfall API | |
| **Trigger** | When the user saves the collection, Scryfall API is polled for pricing | |
| **Main Scenario** | **Step** | **Action** |
|  | Step 1 | User saves their collection with the “save” button |
|  | Step 2 | Scryfall API is polled for pricing for cards in collection |
|  | Step 3 | Card listings pricing is updated. |
| **Extensions** | **Step** | **Branching Action** |
|  | Step 3.1 | If there is no internet connection, API call will fail and nothing will happen |
| **Open Issues** | Issue # | N/A |

3. **Non-Functional Requirements**

3.1 Customer Constraints

* This program should be developed for access on Android and IOS enabled devices
* Program should pull less than a MB of data per request to external APIs
* Program should not request external data twice within any given 5 second window

3.2 External Interfaces

## .json files from the Scryfall API containing pricing data

## File access from the OS of the user.

4. **Deliverables**

The following will be delivered to the Customer in a zipped file folder:

* Software Design Document: Word File Format
* Software Requirement Document: Word File Format
* Testing Document: Word File Format
* User Interface Document: Word File Format
* UML: Delivered in PNG format
* Compiled Software Package: .apk format and .ipk format for both operating systems.

5. **Open Issues**

So far the only apparent issue is that sometimes, due to latency in the operating system, the app will fail to load the collection in under 3 tries. The current temporary solution is to attempt to open the app again. So far, a second try has always worked if a first does not.

**Appendix A – Agreement Between Customer and Contractor**

By signing this document, the customer agrees that this document is to be used as a description of the functionality and overview of use for the Arcane Library mobile app. They also agree that this document is subject to change at any time by request of either the AL development team or the customer based on the specifications below

Changes made to this document will require another round of sign-offs by all party members, the proposed schedule will be adjusted by an adequate amount as determined by the AL development team, and compensation renegotiated up to within 20% of the previously agreed amount. Changes to this document will not be reflected in the project planning until a maximum of 7 days after re-signing has occurred, during which period the changes may be negotiated without another signing.

| **Signer** | **Signature** |
| --- | --- |
| AL Development Team |  |
| Customer |  |

**Appendix B – Team Review Sign-off**

The AL Development Team has reviewed this document and found it sufficient to communicate to the customer the needs and requirements of the development team and customer.

| **Signer** | **Signature** |
| --- | --- |
| AL Development Team |  |
| Customer |  |