TableSetter Software Requirements Specification

# 1. Introduction

## 1.1 Purpose

This SRS details the software requirements of the initial release of the TableSetter mobile application, such as an overall description of the TableSetter application, the software needed to create the application, and the different use cases for the application. This document is intended to be used by the TableSetter development team as an organized reference and guide for the prioritization of application features for the first version of TableSetter.

## 1.3 Project Scope

TableSetter is a mobile application that allows users to manage and sort a database of board games. More information can be found in the *TableSetter Vision and Scope Document* [1].

## 1.4 References

1. Araujo, C., Jamerlan, M.J., & Vaughn, E. (2019) [*TableSetter Vision and Scope Document*](https://docs.google.com/document/d/13vNVAsAh-Ewdhii-yZWLjD26L2J2y7djo7GU6y7Y7Ts/edit?usp=sharing)
2. Araujo, C., Jamerlan, M.J., & Vaughn, E. (2019) *TableSetter Functional Design Document*

# 2. Overall Description

## 2.1 Product Perspective

This project is a mobile application that will be able to quickly file and sort different catalogs of board games. The reason for this is the lack of tools and assets in this area. TableSetter is expected to add additional functionality in later releases of the application.

## 2.2 User Classes and Characteristics

|  |  |
| --- | --- |
| **User Class** | **Description** |
| Application User (Favored Class) | This user will be able to access the database of board games and search for a board game they wish to add to their catalog. The user will also be sort the board games in their own catalog in several ways and will be able to tag them using predefined tags or through user-defined tags. |

## 2.3 Operating Environment

OE-1: TableSetter is a mobile application currently set to release on the Android platform.

OE-2: Table Setter has all database features hosted locally by the user.

# 3. System Features

## 3.1 Add a Board Game to the Catalog

|  |  |  |  |
| --- | --- | --- | --- |
| **ID:** | Use Case 1 | **Name:** | Add a Board Game |
| **Created By:** | Eryk Vaughn | **Date Created:** | 2/11/19 |
| **Primary Actor:** | User | **Secondary Actors:** | None |
| **Description:** | The User wants to add a board game to their catalog. | | |
| **Trigger:** | The User clicks the “Add to Catalog” button. | | |
| **Preconditions:** | Pre-1: User has opened the application  Pre-2: User has navigated to the tab containing the “Add to Catalog” button. | | |
| **Postconditions:** | Post-1: User was able to add to the local database of board games.  Post-2: User was able to add notes and tags to the game before it was added to database. | | |
| **Normal Flow:** | **1.0.) Create a blank template**   1. Allow the user to add the name of the board game. 2. Add a picture of board game. 3. Add or create tags for the board game. 4. Add notes for the board game. 5. Click “Submit”. | | |
| **Alternative Flows:** | NONE | | |
| **Exceptions:** | **1.0.E1.) Board Game Name is Taken**   1. Error message comes up telling you name is taken. 2. Asks you to resubmit new name.   **1.0.E2.) Board Game Name is Invalid**   1. Error message comes up telling you the name is invalid. 2. Asks you to resubmit new name. | | |
| **Priority:** | High | | |
| **Frequency of Use:** | Common | | |
| **Business Rules:** | NONE | | |
| **Other Information:** | Once the user submits an entry it can be edited at another time. | | |
| **Assumptions:** | NONE | | |

## 3.2. Sort The Catalog of Board Games

|  |  |  |  |
| --- | --- | --- | --- |
| **ID:** | Use Case 2 | **Name:** | Sort the Catalog |
| **Created By:** | Marc Julian Jamerlan | **Date Created:** | 2/11/19 |
| **Primary Actor:** | User | **Secondary Actors:** | None |
| **Description:** | The User wants to sort their catalog of board games in a certain way. | | |
| **Trigger:** | The User chooses one of the options in the “Sort Catalog” tab. | | |
| **Preconditions:** | Pre-1: User has opened the application.  Pre-2: User has navigated to the “Sort Catalog” tab. | | |
| **Postconditions:** | Post-1: Catalog will be sorted in an order specified by the user.  Post-2: The sorted catalog will be displayed to the user. | | |
| **Normal Flow:** | **2.0.) Sort the catalog of board games**   1. User is shown a menu containing options for sorting the catalog. 2. User chooses an option from the menu. 3. Database of catalog entries is sorted using the sort function chosen. 4. Catalog is displayed to the user. | | |
| **Alternative Flows:** | NONE | | |
| **Exceptions:** | **2.0.E1.) Catalog is Empty**   1. Error message: “The catalog is empty!”   **2.0.E2.) Sort Fails Unexpectedly**   1. Error message: “Failed to sort the catalog, catalog is unchanged.” 2. Catalog will not be sorted. | | |
| **Priority:** | High | | |
| **Frequency of Use:** | Fairly Common | | |
| **Business Rules:** | None | | |
| **Other Information:** | None | | |
| **Assumptions:** | An entry exists in the catalog. | | |

## 3.3 Delete Board Game

|  |  |  |  |
| --- | --- | --- | --- |
| **ID:** | Use Case 3 | **Name:** | Delete Board Game |
| **Created By:** | Eryk Vaughn | **Date Created:** | 2/13/19 |
| **Primary Actor:** | User | **Secondary Actors:** | None |
| **Description:** | The User wants to delete a board game from the catalog. | | |
| **Trigger:** | The User clicks the “Delete from Catalog” button. | | |
| **Preconditions:** | Pre-1: User has selected a board game in the list.  Pre-2: User clicks the “Delete Entry” button. | | |
| **Postconditions:** | Post-1: Selected entry is removed from the catalog. | | |
| **Normal Flow:** | **3.0.) Delete the entry**   1. All values corresponding to the entry are cleared and deallocated. 2. The entry from the database is removed. | | |
| **Alternative Flows:** | NONE | | |
| **Exceptions:** | **3.0.E1.) Deletion Fails Unexpectedly**   1. Error message: “Failed to delete the selected entry, please try again.” 2. The entry will not be deleted. | | |
| **Priority:** | High | | |
| **Frequency of Use:** | Uncommon | | |
| **Business Rules:** | NONE | | |
| **Other Information:** | NONE | | |
| **Assumptions:** | An entry exists in the catalog. | | |

## 3.4 Edit An Entry in the Catalog

|  |  |  |  |
| --- | --- | --- | --- |
| **ID:** | Use Case 4 | **Name:** | Edit Catalog Entry |
| **Created By:** | Christopher Araujo | **Date Created:** | 2/13/19 |
| **Primary Actor:** | User | **Secondary Actors:** | None |
| **Description:** | The User wants to edit an entry in the catalog of board games. | | |
| **Trigger:** | The User clicks the “Edit Entry” button. | | |
| **Preconditions:** | Pre-1: User has selected a board game in the list.  Pre-2: User selects the “Edit Entry” button. | | |
| **Postconditions:** | Post-1: User was able to change the details of a board game within the local database. | | |
| **Normal Flow:** | **4.0.) Re-open the entry to be edited**   1. Allows the user to change the name of the board game. 2. Change the picture of board game. 3. Edit, delete, or add tags to the board game. 4. Edit, delete, or add notes to the board game. 5. Click “Submit”. | | |
| **Alternative Flows:** | NONE | | |
| **Exceptions:** | **4.0.E1.) Edit Fails Unexpectedly**   1. Error message: “Failed to edit the entry, please try again.” 2. The entry’s information will remain unchanged.   **4.0.E2.) An Invalid Edit is Made**   1. Error message: “One or more of the edited fields is invalid.” 2. The entry’s information will remain unchanged. | | |
| **Priority:** | High | | |
| **Frequency of Use:** | Uncommon | | |
| **Business Rules:** | NONE | | |
| **Other Information:** | Once someone submits an entry it can be edited afterwards. There are no limits to how often the entry can be edited. | | |
| **Assumptions:** | An entry exists in the catalog. | | |

## 

## 3.5 Search For A Board Game in the Catalog

|  |  |  |  |
| --- | --- | --- | --- |
| **ID:** | Use Case 5 | **Name:** | Search For a Game |
| **Created By:** | Christopher Araujo | **Date Created:** | 2/13/19 |
| **Primary Actor:** | User | **Secondary Actors:** | None |
| **Description:** | The User wants to search within their own catalog for specific games. | | |
| **Trigger:** | The User clicks the “Search” button. | | |
| **Preconditions:** | Pre-1: User has opened their catalog. | | |
| **Postconditions:** | Post-1: User was able to search within their catalog for a specific game or a specific set of games. | | |
| **Normal Flow:** | **5.0.) Opens a search bar and prompts the keyboard to appear**   1. User enters their search query into the search bar.    1. May contain the name of the cataloged item.    2. May contain tags that pertain to the cataloged item. 2. A list of games that best pertain to the search query will appear. 3. Users can navigate to the catalog entry of a game returned by the search. | | |
| **Alternative Flows:** | NONE | | |
| **Exceptions:** | **5.0.E1.) No Such Items Found Within Database**   1. Page tells the user that no items matched the entered name or tags. | | |
| **Priority:** | High | | |
| **Frequency of Use:** | Common | | |
| **Business Rules:** | NONE | | |
| **Other Information:** | Once someone submits it can be edited and submitted. There are no limits on how often the entry can be edited. | | |
| **Assumptions:** | NONE | | |

# 

# 4. Data Requirements

# 4.1 Logical Data Model



## 4.2 Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Element** | **Description** | **Composition or Data Type** | **Length** | **Values** |
| **TAGS** | Is an object that stores a tags name and description | TAGS ID -   1. Tag Name 2. Tag Description |  |  |
| Tag name | The name of the tag. | String | 256 |  |
| Tag Description | The description of the tag and any other info. | String | 256 |  |
| **BOARD GAME** | This the board game itself. The board game holds the name of the game, its picture, and an array of tags. This is followed by a simple description. | Board game ID -   1. Game Name 2. Picture of Game 3. Array of Tags 4. Game Description |  |  |
| Game Name | This is the name of the game. | String | 256 |  |
| Picture of Game | An image of the game provided as either a link to an image or as image file. | String/image | 256 |  |
| Game Description | This is a short summary of the game, and a container for other notes about the game. | String | 256 |  |
| **CATALOG** | This is the storage center for the number of games. Keeping them in one central place. | CATALOG ID   1. Array of board games objects |  |  |

## 4.3 Reports

## 4.3.1. Search Results Report

|  |  |
| --- | --- |
| **Report ID** | Query-Return-1 |
| **Report Title** | Search Return |
| **Report Purpose** | User wants to search the list of board games. |
| **Decisions Made From This Report** | User will be able to view the selection of board games then select one for further info. |
| **Report Users** | The Sole User |
| **Data Sources** | This information is accessed from the CATALOG. |
| **Frequency and Disposition** | This report is generated whenever the user opens the app or refreshes the list. |
| **Latency** | Depending on the amount of board games, should take up to 5 seconds and no more. |
| **Visual Layout** | Portrait |
| **Header and Footer** | Report header: “Search Results”. |
| **Report body** | Shows a list of board games relevant to the search query:   * Title of the game. * List of the game’s tags. |
| **End of report indicator** | Number of search results returned. |
| **Interactivity** | Users will be able to click on a search result to navigate to it’s catalog entry. |
| **Security Access Restrictions** | None |

## 4.4 Data Acquisition, Integrity, Retention, and Disposal

DA-1: TableSetter will load the database and display the catalog upon starting the application.

DI-1: TableSetter will update the database to reflect the changes made upon the addition, deletion, and editing of an entry.

DD-1: All images, tags and notes within an entry will be deleted upon confirmation of the deletion of an entry.

# 5. External Interface Requirements

## 5.1 User Interfaces

For more information, see the *TableSetter Functional Design Document* [2].

## 5.2 Software Interfaces

TableSetter will use a database to store all the entries and in the catalog.

## 5.3 Hardware Interfaces

The system shall be developed to work on the Android platform, but specific hardware specifications are not required.

## 5.4 Communication Interfaces

The user shall communicate with the system’s database through electronic forms.

As the catalog will be hosted locally on the system there is no need for the user to connect online to utilise the application.

# 6. Quality Attributes

## 6.1 Usability

User shall be able to learn how to utilize Tablesetter’s various options in a matter of minutes to organize and store board game information within the system.

## 6.2 Performance

The system should take no more than 1-2 seconds to display each page.

# Appendix A: Glossary

Catalog - The database used to store information on board games.