# SW5E Bestiary Companion SRS

## Jeremy Harrington and Nazanal Laughlin

### 03/30/2024

### V0.1.2

# Section 1: Introduction

## Purpose

To provide an application which serves to make the vast amount of data and information available in print for a tabletop pen and paper game more readable and searchable.

## Scope

The Bestiary Companion will service the needs of users in the SW5E TTRPG system to find precise data on the myriad creatures that exist in the TTRPG system. This application will allow for filtering based on several possible criteria delineating one creature or group of creatures from another, noted in the Requirements section of this document.

## User Profiles

* Customer
  + The customer is Delta Squad, the group which manages the SW5E ecosystem. They need a mobile app to serve as a companion/searchable database of all the creature information that is a part of their game.
* Target Audience/Users
  + Users will be those who are organizing SW5E games as well as the players in those games who want easy access to statistics and information to make game play smoother.

## Workflows

The users will open the app and be on the home screen. They can either start a search using text only, go to the advanced search page to see a full list of filterable and start an advanced search, or go to the lists page, which allows them to load/export lists. Loading a list or starting a search brings up the results page for that action, which allows a user to select a creature to obtain a detailed list of all of its attributes, or return to the previous page. While on the details page, a user can add a creature to a list.

## Definitions

|  |  |
| --- | --- |
| System | The Application, as detailed in “Scope”. SW5E Bestiary Companion Application. |
| Game | SW5E Tabletop Roleplaying Game System. |
| Creature | An entity existing within the Game, composed of many Filterable and Non-Filterable traits and statistics. |
| Creature Data | All of the statistics, skills, abilities, and information relevant and related to a specific creature stored within the databse, including all Filterable, Non-Filterable, Numeric, and Non-Numeric data concerning the rules related to the creature in the System. |
| Filterable | Searchable information about a Creature that can be used to identify it and distinguish it from other Creatures in the DB. Filterables include Creature Name, Creature Type, Creature Size, and Challenge Rating. Filterables vary somewhat in how specific they are, with some being broad categories, others being specific. |
| Non-Filterable | Information about a creature stored within the database but that is not directly searchable through the Application. This information is still displayed within the app, as it is relevant to the System. |
| Creature Type | A broad Filterable, indicating whether a Creature is a “Droid”, a “Beast”, a “Humanoid”, or some other category. |
| Creature Size | A specific Filterable, indicating the approximate size of the Creature. Size options are Tiny, Small, Medium, Large, Huge, and Gargantuan. Each size category is approximately twice the size of the category immediately before it. Humans are considered Medium. |
| Challenge Rating | A numerical Filterable, indicating approximately how powerful players of the Game should be to handle a specific Creature. |
| DB | The Database representing all the Creatures and their respective data entries. |
| User | A User of the System, needing information about one or more Creature entities. |

# Section 2: Requirements

## Requirements

1. The System shall allow the User to search for Creatures by all Filterable traits.
2. The System shall query the DB to retrieve Creatures matching the search criteria of the User.
3. The System shall have a UI rated 8/10 or higher by 4/5 or more of users as being “Intuitive” and “Self Explanatory”
4. The System shall allow the User to select a Creature from the List.
5. The System shall allow the User to view all of the information (Numerical, Non-Numerical, Filterable and Non-Filterable) of a selected Creature.
6. The System shall present the data of a selected Creature in a format rated 8/10 or higher by 4/5 or more of users as being “Easy to Read” and “Accessible”
7. The System shall prevent the User from editing the DB.

## Stretch Requirements

1. The System could allow the User to add a Creature to a list of favorites.
2. The System could allow the User to select a Creature from the favorites list.
3. The System could allow the User to create additional lists of Creatures
4. The System could allow the User to export additional lists of Creatures to other Users.
5. The System could allow the User to import a received list of Creatures for use.
6. The System could display any loaded list to the User.

# Section 3: Overview

## Resources

This application will be using a client/server model.

* Server
  + The server will be an Ubuntu linux server
  + It will run
    - A MySQL database, for data storage/processing
    - A C# .NET service for internet connectivity, allowing the clients to connect to the server
  + It will be setup by us but owned and maintained by Delta Squad
* Client
  + There will be two different variations of the client
    - Android
      * The android version of the client will run on Kotlin and the Android standard libraries
      * It will be distributed using the Google Play store
    - iOS
      * The apple version of the client will run on Swift and the iOS standard libraries.
      * It will be distributed using the Apple app store.

All of the raw data for the project and database will be provided by Delta Squad, though parsed and converted to a digital format by us.

## Data at Rest

There will be two storage locations for data. The first will be the searchable database stored on Delta Squad’s servers. These are Ubuntu linux machines running the MySQL database. It will store all of the creature data that is to be displayed in the app.

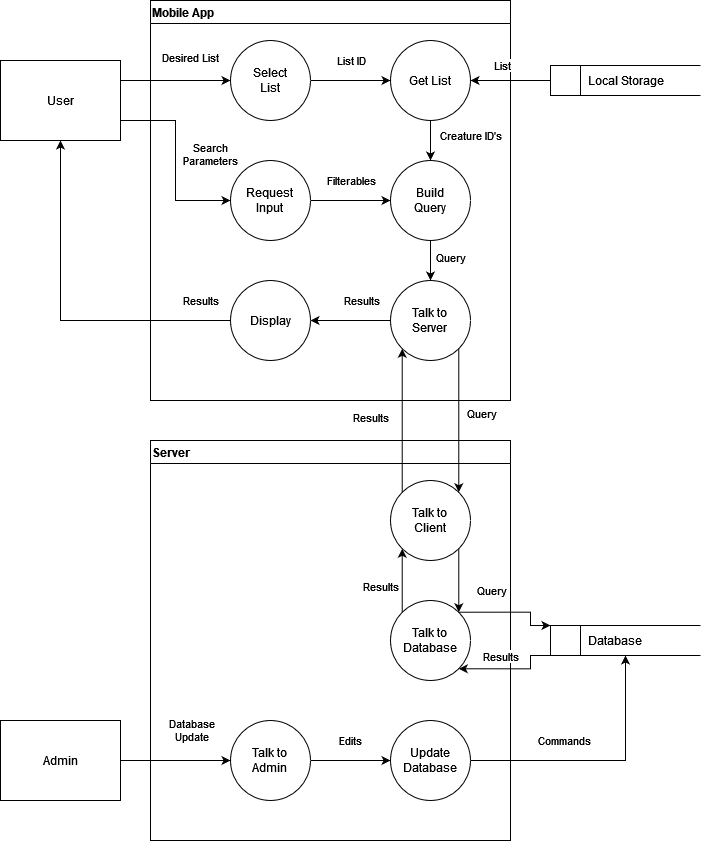
The second is local storage on the user’s device of their custom lists. These will be stored in CSV files containing the Primary Key IDs of all creatures they have added to that list, so a query can be easily built that selects all of them.

## Data in Motion

The search requests will be transmitted using standard MySQL connection practices, setting up the PHP service on the Delta Squad server and using the built in libraries for Swift/Kotlin to connect over the internet and query the database.

The processing will be split between the users device, which translates the user’s selected filterables and search criteria into a usable query, and the server, which executes the query on the database and returns the results.

## Data State



## User Workflow

# Section 4: Verification

## Testing

Requirement 1: Verify that a selected Creature Page shows all data associated with that creature

Requirement 2: Verify that all Filterables produce matching result lists

Requirement 3: Verify that all Creatures retrieved match search criteria

Requirement 4: Verify that the retrieved list is clean and easy to use

Requirement 5: Verify that the User can select a retrieved Creature

Requirement 6: Verify that all data for a selected Creature is present and easy to read

Requirement 7: Verify that the DB is secure

Stretch Requirement 8: Verify that a selected Creature is correctly added to a Favorites List

Stretch Requirement 9: Verify that a Creature may be selected from a Favorites List

Stretch Requirement 10: Verify that the User can create additional lists

Stretch Requirement 11: Verify that selected lists can be exported

Stretch Requirement 12: Verify that selected lists can be imported

Stretch Requirement 13: Verify that any selected list is properly displayed to the User

## Demo

For demonstration purposes, a number of scenarios will be examined. For searching, the following strategies will be employed: demonstrate completion of a search that, if successful, would produce a single Creature or a single, adequately specific group of Creatures; demonstrate completion of searches that are built around one of the several Filterables noted previously; and lastly, a demonstrated completion of a search that would fail to produce any Creatures, due to invalid criteria. For List functionality, a complete List for export/import testing, as well as for testing ability to add and access List items.