Dungeons and Datasheets

Asymptotically Challenged Monkeys

December 10, 2019

Jonathan Bacon Eva Moniz Brandon Pozil

Advised by Ike Quigley

We have abided by the UNCG Student Code of Conduct

Contents

1 Introduction

1.1 Purpose

Dungeons & Datasheets is a desktop application for all D&D players to use as a character sheet management system. A primary aim of this software is streamlining the experience of playing Dungeons & Dragons, allowing users more time to focus on the role-playing aspect of the game.

1.2 Document Conventions

- Dungeons & Dragons is often abbreviated to D&D.
- At various places in this document, a convention is used to describe dice rolls in the game of D&D, where a roll of a y-sided die repeated x times will be written as xDy. For example, if a player rolls a 6-sided die 2 times, this is written as 2D6.

1.3 Intended Audience

This document is intended for the developers of Dungeons & Datasheets and the project manager, Ike Quigley.

1.4 Definitions

Character Sheet The collection of data representing the totality of information describing a player's character in Dungeons and Dragons. This includes name, stats, background story, and more.

LAN Local Area Network.

1.5 Project Scope

Dungeons & Datasheets is intended to be used maintain character sheet information, allow simulation of dice rolls, and allow basic chat between users on the same LAN. It is not intended to provide gameplay simulation, map-making tools, or complex inventory and non-player-character management.

1.6 Technical Challenges

The software must provide an easy-to-use system for chat over LAN. Care must be taken in deciding how this is to be implemented.

Working with API's proved difficult as each API was built syntax agnostic so as to be usable by many different languages, This caused issues which forced us to use both the GSON and JSON libraries independent of each other when we would have liked to use one for any and all operations.

1.7 References

Ike Quigley, Teacher Extraordinaire

2 Overall Description

2.1 Project Features

The software will have the following features:

- A character sheet management system
- A chat system
- A dice rolling simulator
- A monster generation system

2.2 User Characteristics

This software is intended for experienced players of Dungeons & Dragons. Users are expected to understand how a character sheet works, as the program only provides the functionality for data entry of character sheet information.

2.3 Operating Environment

So long as the host device can run a Java 8 SDK, this can run on any operating system.

2.4 Assumptions and Dependencies

This software targets a system with at least Java 8 and at least 2 GB of RAM. The program requires the Google GSON library for its persistent data storage, as well as the JSON library for parsing monster data.

3 Functional Requirements

3.1 Primary Requirements

- The software shall provide the ability for the user to create a new character sheet.
- The software shall provide the ability for the user to modify an existing character sheet. This modification includes all stats, and textual elements like name, race, alignment, backstory, etc.
- The software shall provide the user with a process of saving and loading character sheets to/from the file system.

3.2 Secondary Requirements

These features should be implemented if time permits.

- The software shall provide functionality to simulate dice rolls of all of the standard die sizes of D&D: D4, D6, D8, D10, D12, and D20.
- The software shall provide a chat system, whereby multiple users on the same local area network can join a chatroom and communicate with each other. This chat system should include sending information about simulated dice rolls.
- The software shall provide a basic inventory management system, allowing the user to keep track of items stored in their character's inventory.

3.3 Use Case Diagrams and Wireframes

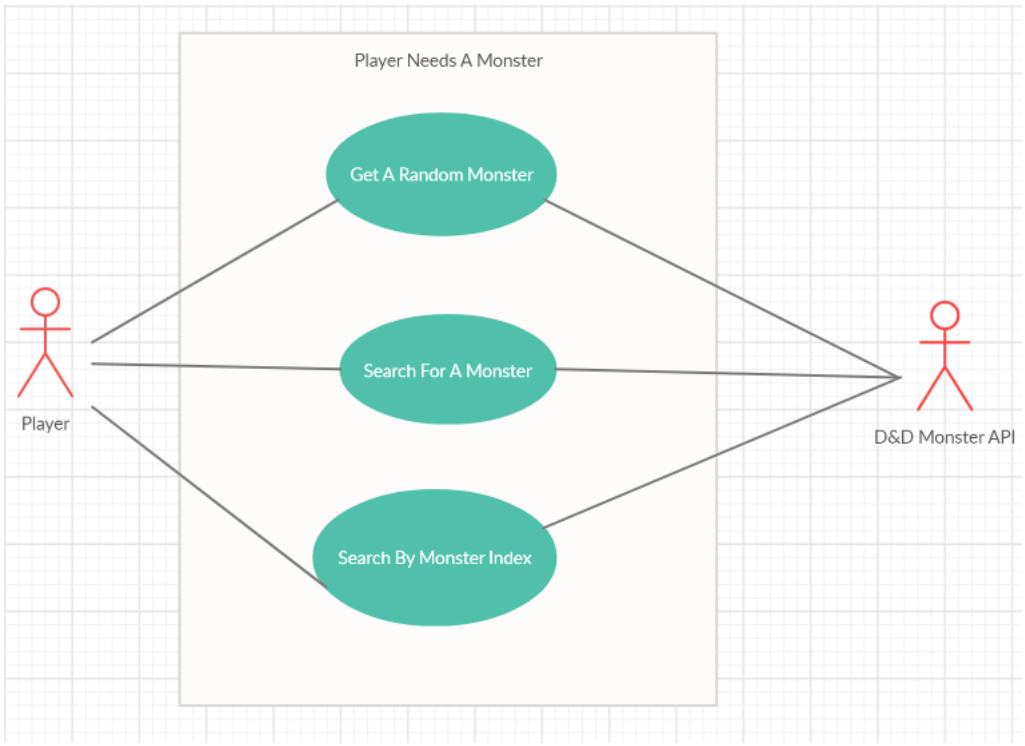

Figure 1: Monster generation use case

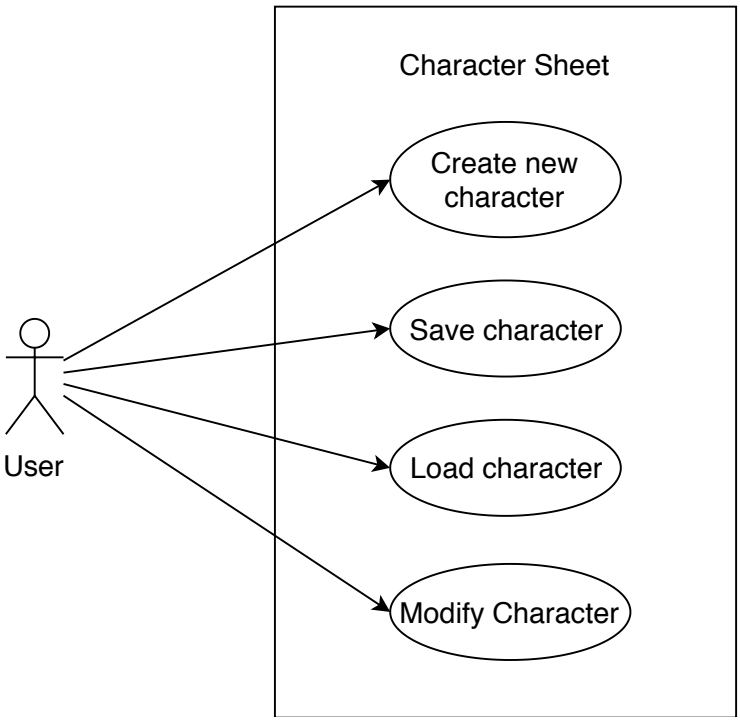

Figure 2: Character sheet use case

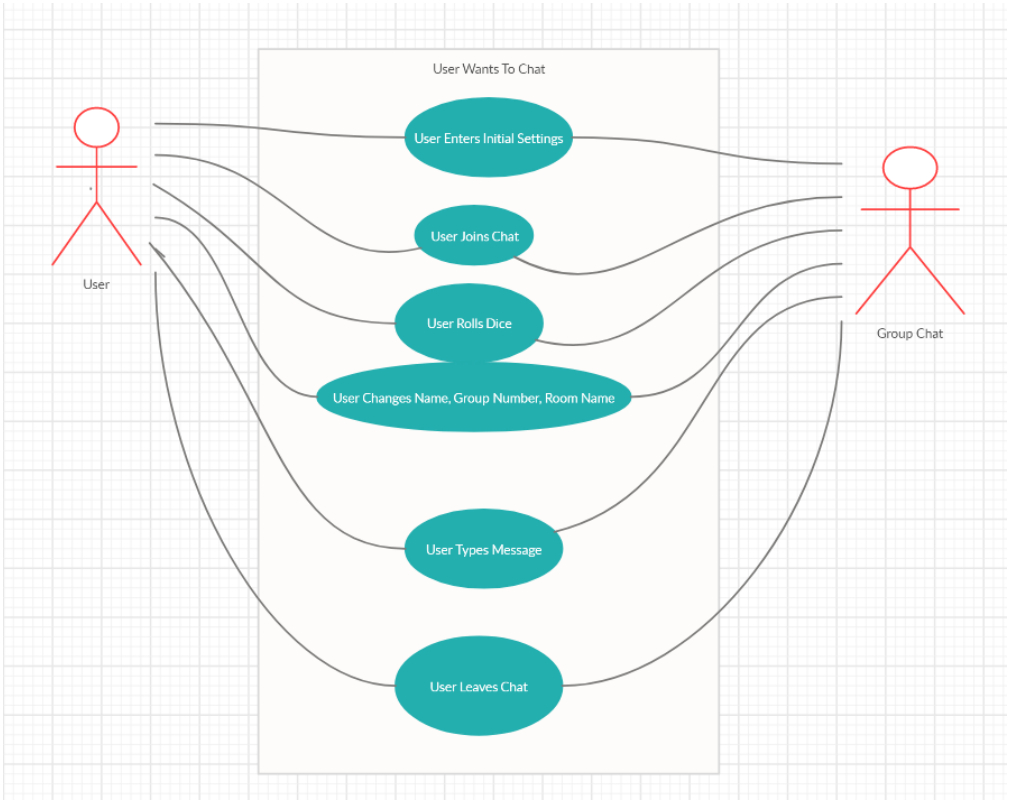

Figure 3: Chat use case

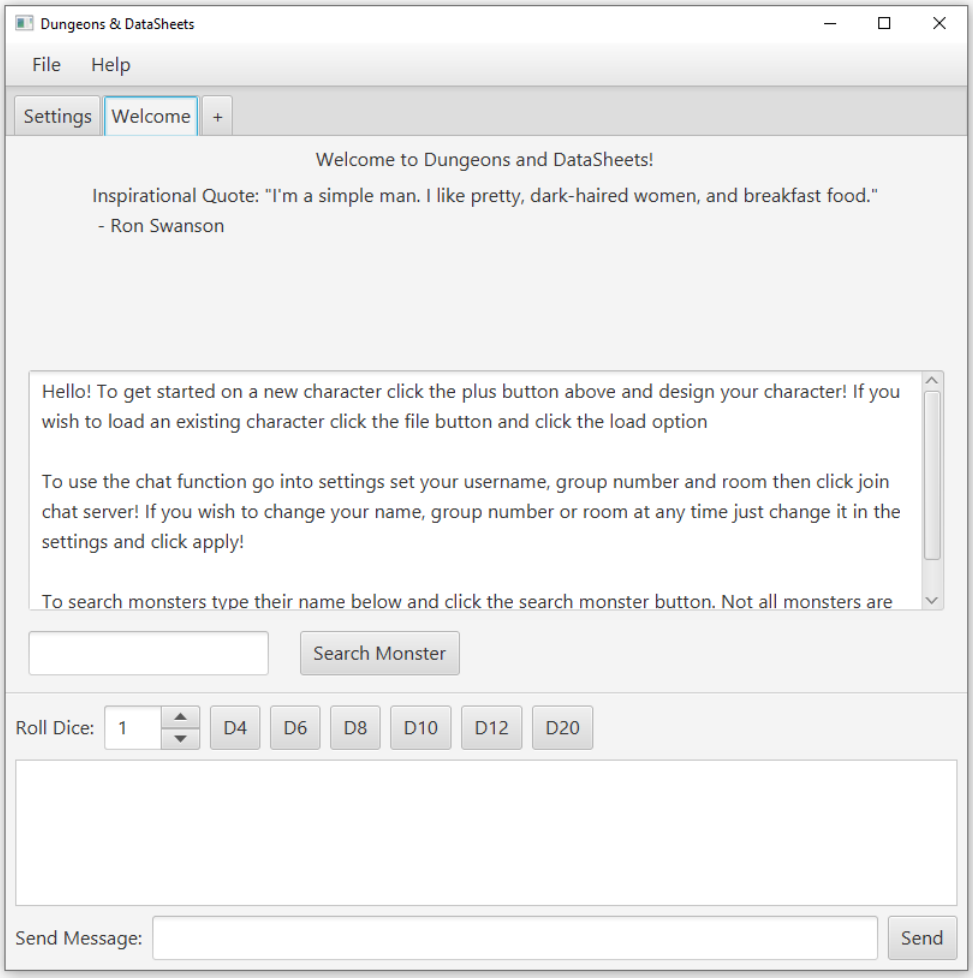

Figure 4: Welcome Page, Chat, and Dice controls

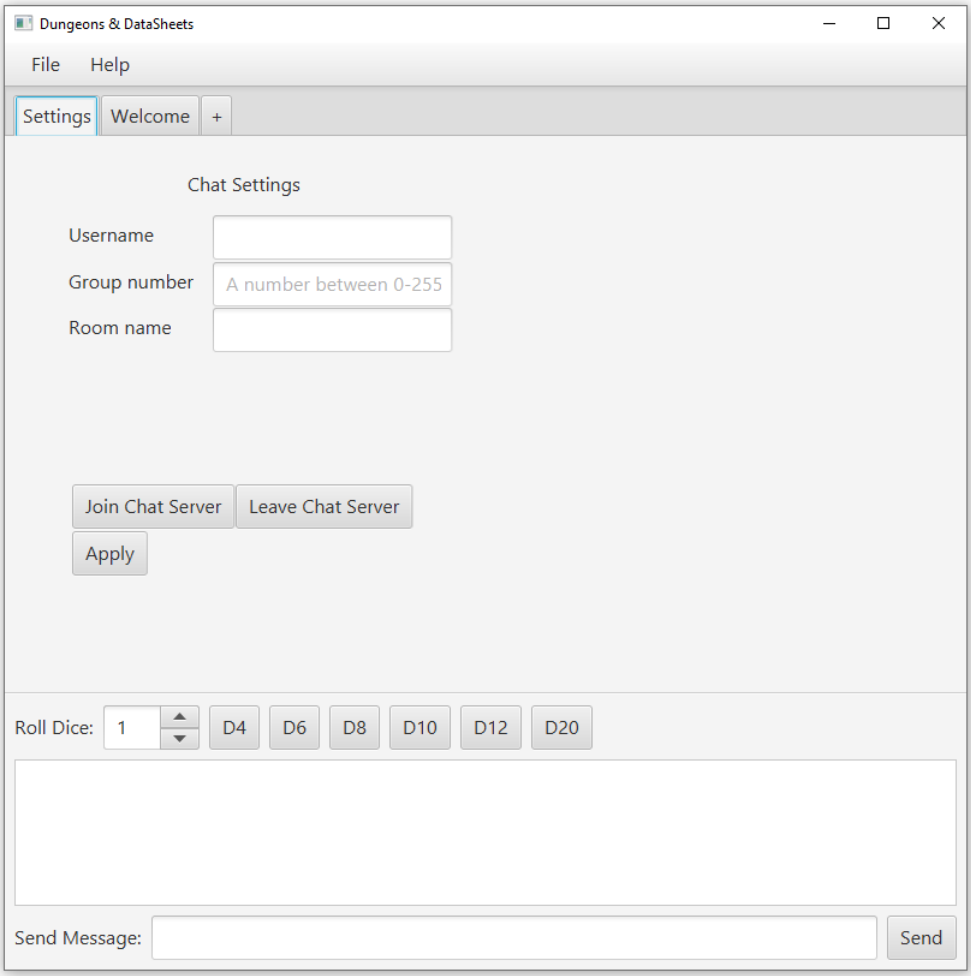

Figure 5: Settings Page

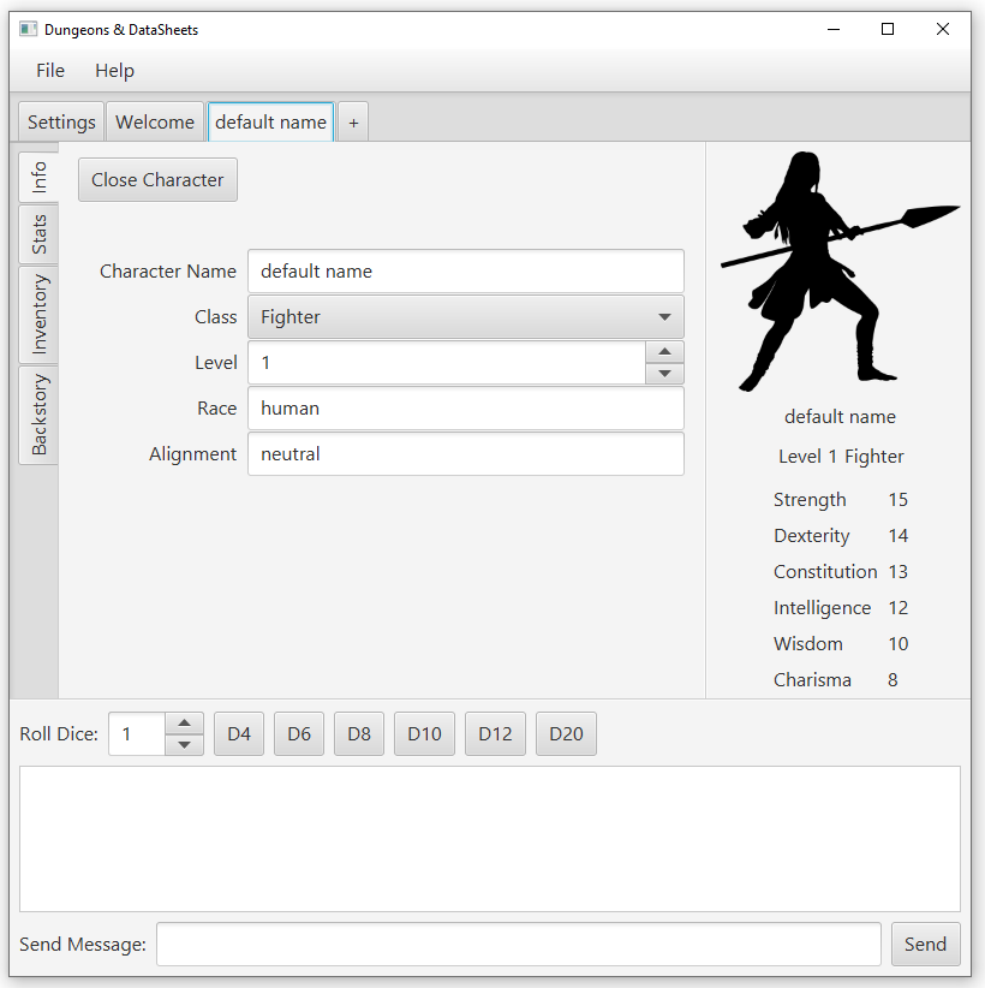

Figure 6: Character Sheet

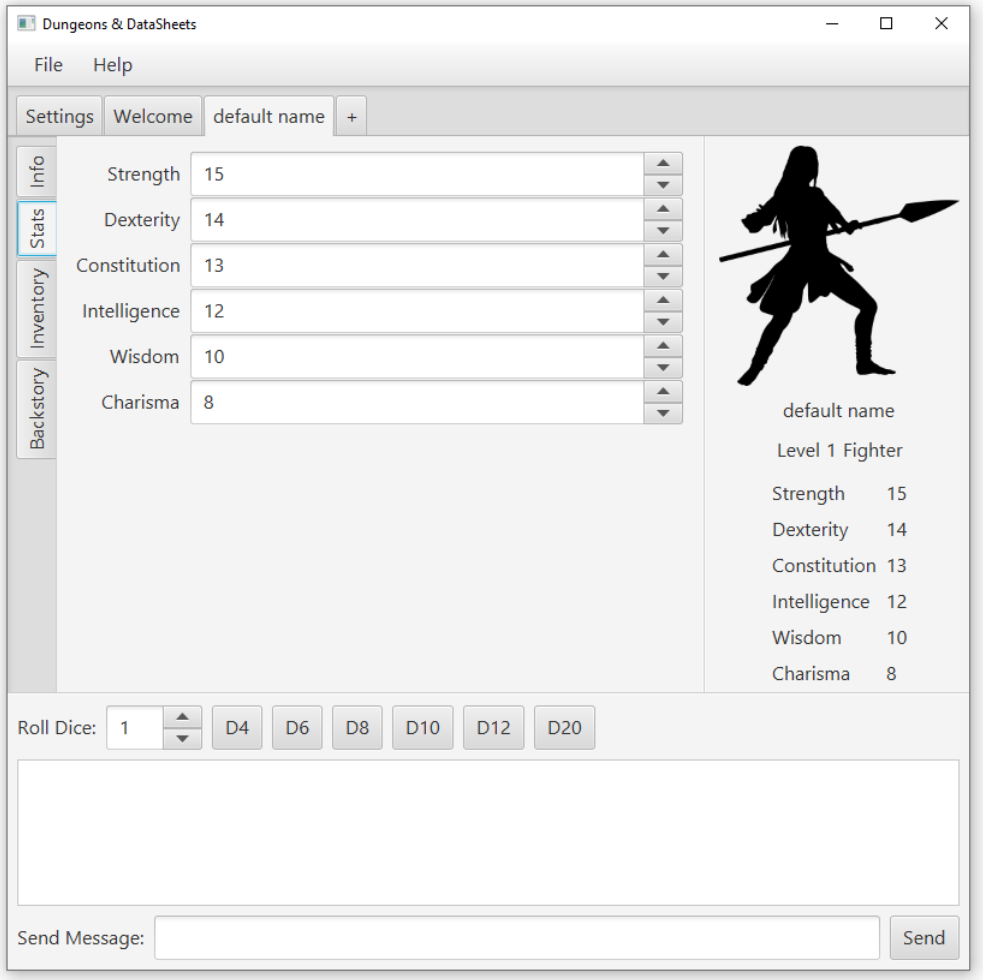

Figure 7: Character Sheet: stats page

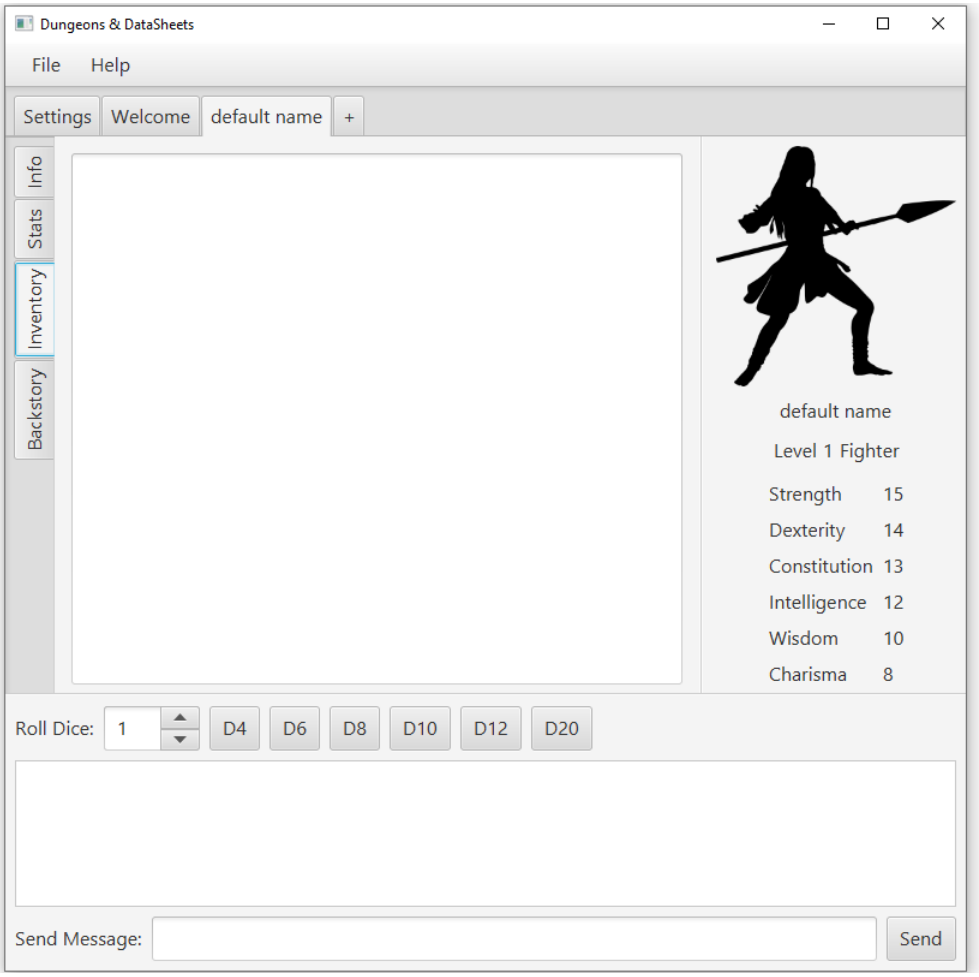

Figure 8: Character Sheet: inventory

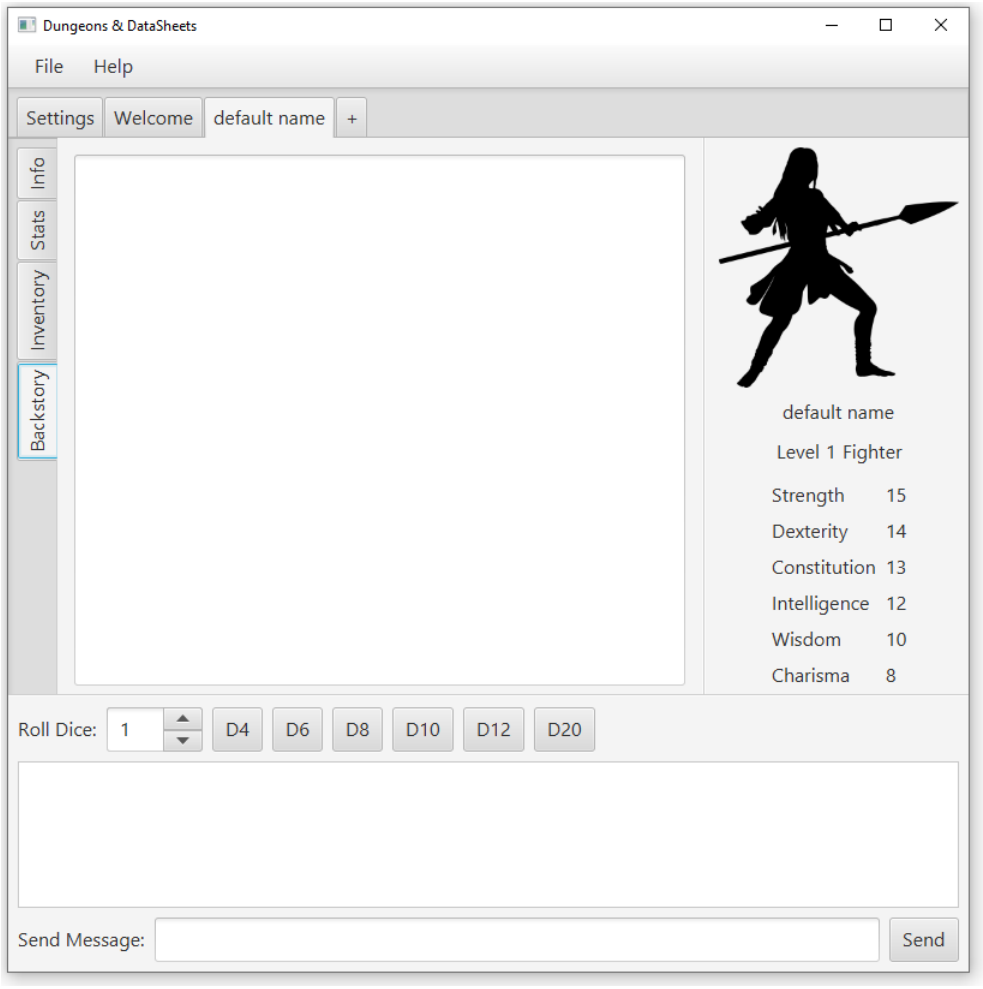

Figure 9: Character Sheet: backstory

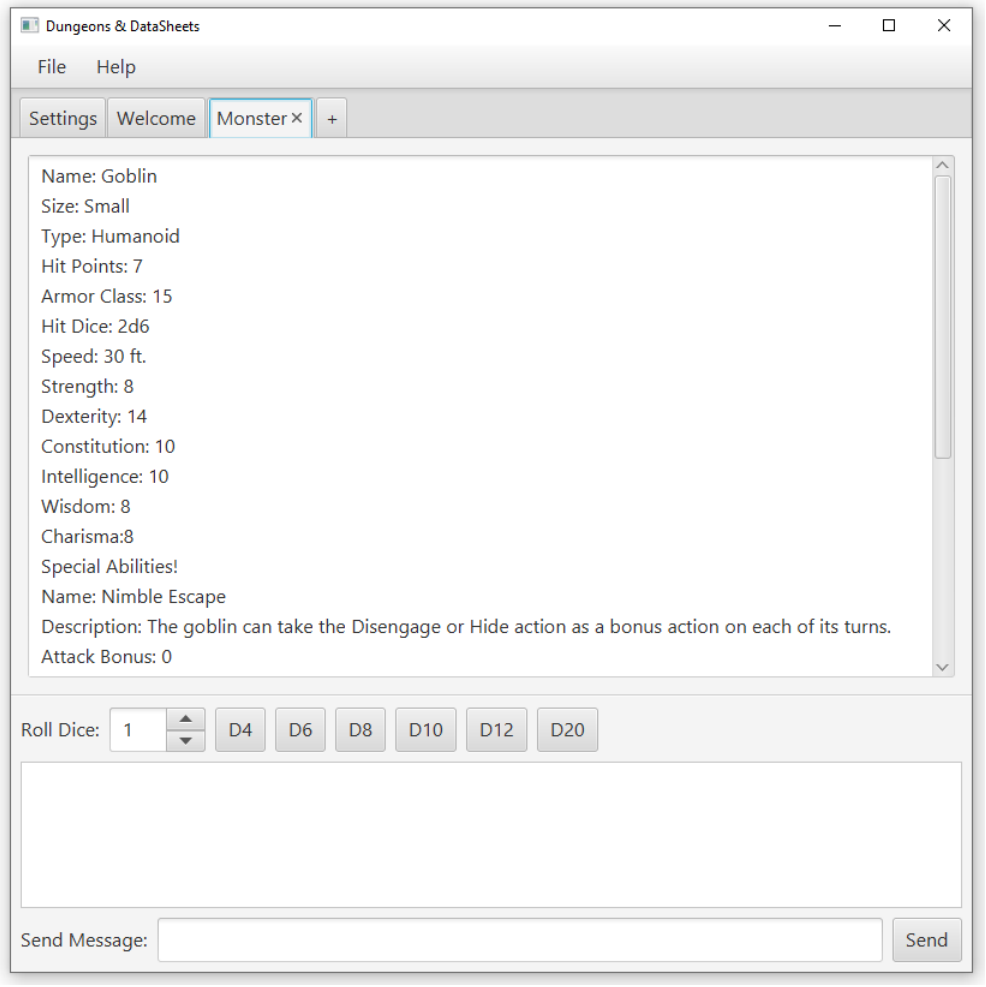

Figure 10: Monster information page

4 Technical Requirements

4.1 Operating Systems and Compatibility

The program will run on Windows 10 with Java 8 installed. Other operating systems may be usable, not explicitly supported.

4.2 Interface Requirements

4.2.1 User Interface

The user will interact with a graphical interface using a mouse and keyboard.

4.2.2 Hardware Interfaces

The software will indirectly use a wifi or ethernet interface on the user's computer to communicate with other computers on the network for the chat functionality. However, if these interfaces are not present, the program should continue functioning in its other areas.

4.2.3 Software Requirements

The program will use Google GSON to serialize and deserialize character sheet data for storage on the user's disk.

The program must execute its logic without hanging the GUI thread. This is especially important for the chat system, because it will run for an extended period of time and wait until input is received. This cannot possibly happen on the JavaFX Application thread. Therefore worker threads must be used.

4.2.4 Communications Interfaces

The chat communications interface is done through a multi-cast UDP system. This is to say that for our chat we set up a listener on a preassigned multi-cast port and when we desire to send a message to the other players we send out a UDP signal on that multi-cast address and port. This is done because other technologies exist which fill the space of internet based communications.

Multiple API calls are also made that require an internet connection to properly utilize. All of the API's are called via the getRequest() method. While some methods are universal, every API has things formatted just a little differently each time. This method reaches out, via an HTTPRequest and a URL, to grab a JSONObject that can be parsed. Thus, the only thing that can remain static for the API is the getRequest() method. Any string cleaning needs to be handled individually for each API call.

4.3 System Architecture

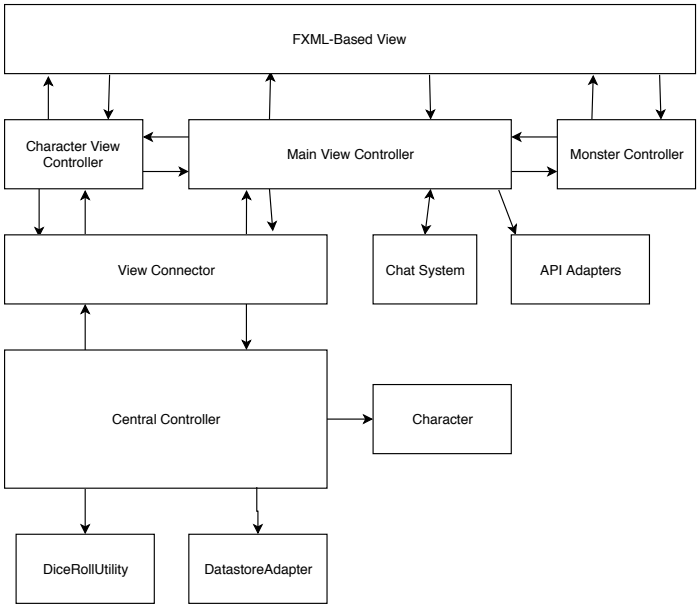

Figure 11: System Architecture

5 Nonfunctional Requirements

5.1 Performance Requirements

The program must execute its logic without freezing the GUI. This is discussed further in subsection 4.2.3.

5.2 Security Requirements

The software should not expose the user's computer to extreme security risks when performing its network communication.

5.3 Software Quality Attributes

5.3.1 Availability

This software will be available to anyone and will be open source.

5.3.2 Correctness

The software should accurately reflect the mechanics of D&D.

5.3.3 Maintainability

The software should be implemented using MVC and with appropriate connectors, adaptors, and encapsulation to increase maintainability.

5.3.4 Reusability

The software components should be adequately generalized and encapsulated, with minimal coupling in the model components, to increase reusability.

5.3.5 Portability

The software should run as-is on most machines running Java 8 or later with the appropriate interface hardware.

5.4 Process Requirements

All code should be written on a programmer's respective branch in the git repository located at https://github.com/PrenexNormalForm/dungeons-and-datasheets.

5.4.1 Development Process Used

The development process used was agile and scrum. We employed weekly meetings and code sprints to center development efforts.

5.4.2 Time Constraints

There were a few time constraints to make note of. Unfortunately, we ran out of time to implement the item search with the D&D API. While the getRandomMonster() and doesMonsterExist() methods work, with a little more time getRandomItem() and doesItemExist() could also be added. The other thing to note was that the private chat between two different users would have required more pipe-lining and a TCP connection to each individual user. This would have required far more overhead and management of user read and write threads. Lastly, there was talk about casting all the character information to an actual Character Sheet pdf file. This could be implemented later.

5.4.3 Cost and Delivery Date

The software will cost the price of tuition for 3 students taking a 3-credit hour class at UNCG. It must be ready by December 10, 2019 at 7:00 PM.