**“Dungeons And Dragons Character Creator”**

Software Requirements Specification

Version .1

2/10/2020

Team Leader: Ryan Tyniec

Team Designer: Tyler Dagge

Team Programmers: Stephen Galvin, Franklin Schneider

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 1/30/20 | 0.1 | Ryan Tyniec | Created a Web API project, configured API with user authentication, used entity framework generated code base for user authentication. |
| 2/10/20 | 0.1 | Ryan Tyniec | Final draft of SRS doc, for class grading. |
|  |  |  |  |
|  |  |  |  |

**Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

**1. Introduction** 3

1.1 Purpose 3

1.2 Scope 3

1.3 Definitions, Acronyms, and Abbreviations 3

1.4 Overview 4

**2. General Description** 4

2.1 Product Perspective 4

2.2 Product Functions 4

2.3 Users and Characteristics 5

2.4 General Constraints 5

2.5 Assumptions and Dependencies 5

2.6 Operating Environment 6

**3. Specific Requirements** 6

3.1 Functional Requirements 6

*3.1.1 <Functional Requirement #1>* 6

*3.1.2 <Functional Requirement #2> 6*

*3.1.3 <Functional Requirement #3> 7*

*3.1.4 <Functional Requirement #4> 8*

*3.1.5 <Functional Requirement #5> 9*

*3.1.6 <Functional Requirement #6> 9*

*3.1.7 <Functional Requirement #7> 10*

*3.1.8 <Functional Requirement #8> 10*

*3.1.9 <Functional Requirement #9> 11*

*3.1.10 <Functional Requirement #10> 12*

3.2 Use Cases 12

*3.2.1 Use Case #1 12*

*3.2.2 Use Case #2 13*

*3.2.3 Use Case #3 13*

*3.2.4 Use Case #4 14*

*3.2.5 Use Case #5 14*

*3.2.6 Use Case #6 15*

*3.2.7 Use Case #7 15*

*3.2.8 Use Case #8 16*

*3.2.9 Use Case #9 16*

*3.2.10 Use Case #10 17*

3.4 Non-Functional Requirements 17

*3.5.1 Performance 17*

*3.5.2 Reliability 17*

*3.5.3 Availability 17*

*3.5.4 Security 18*

*3.5.5 Maintainability 18*

*3.5.6 Portability 18*

3.4 Design Constraints 18

3.5 Logical Database Requirements 18

3.6 Other Requirements 18

4.Sequence Diagrams 18

4.1 Figure 1: Character Stat Generation 19

4.2 Figure 2: Character Creation or Viewing a Previous Sheet 20

4.3 Figure 3: Character Hit Point Generation 20

4.4 Figure 4: Character Name and Information Generation 21

4.5 Figure 5: Character Skill, Ability, and Feat Selection 22

**5. Change Management Process 22**

**List of Tables**

Use Case #1 12

Use Case #2 13

Use Case #3 13

Use Case #4 14

Use Case #5 14

Use Case #6 15

Use Case #7 15

Use Case #8 16

Use Case #9 16

Use Case #10 17

**List of Figures**

[Figure 1:](#_1idq7dh) Character Stat Generation 19

[Figure 2: Character Creation or Viewing a Previous Sheet](#_42ddq1a) 20

[Figure 3: Character Hit Point Generation](#_42ddq1a) 20

[Figure 4: Character Name and Information Generation](#_42ddq1a) 21

[Figure 5: Character Skill, Ability, and Feat Selection](#_42ddq1a) 22

# 1. Introduction

The introduction to the Software Requirement Specification (SRS) document should provide an overview of the complete SRS document. While writing this document please remember that this document should contain all of the information needed by a software engineer to adequately design and implement the software product described by the requirements listed in this document.

## 1.1 Purpose

The purpose of this document is to provide a thorough overview of the “Dungeons and Dragons Character Creator” application.. The document will primarily be used for both the developers of the application, as well as for any potential clients that are requesting the software. The document will include a general description of the project, as well as the overall scope, goals, and limitations of the project.

## 1.2 Scope

(1) **Our Project Name:** Dungeons & Dragons Character Builder

(2) **What is the goal of this application?:** The application will be designed to allow players of Dungeons and Dragons to be able to create, edit, and view character builds for games. The application will allow users to create a character profile and store all of their builds.

(3) Our goal for this software is to create and manage a service based Windows application for desktops. To build the application, we are using Microsoft’s .NET Framework as a platform of development. We plan to implement a comprehensive WPF front end that allows users to view their character build sheets, as well as create new ones. Due to the nature of Dungeons and Dragons, a primary goal for the team is to allow character creation for all players to be as simple and intuitive as possible. To achieve this, the project should have an emphasis on creating an intuitive user interface that is easy to use.

Since this application is highly reliant on data, we also plan to incorporate a local database for the application. Due to the nature of software development, we plan to use programming methods that ensure there is as little dependency as possible between our front end design and back end local database to allow expansion of the software to be more efficient after release.

As of now, the application’s scope is limited to the desktop it is being used on. Therefore, besides the initial install, the application will not directly require internet access or external connections to databases.

## 1.3 Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| User | Person who is using the application when it is released. |
| D&D | Abbreviation for the game Dungeons and Dragons. |
| WPF | Windows Presentation Foundation - the platform we will be using to create the user interface of the application. |
| Front End | The part of the application that users will interact with. |
| Back End | The part of the application that works with the front end; this handles the initialization of the application and handles data transmission. |

## 1.4 Overview

The remainder of this document is comprised of three additional chapters. The remaining chapters will follow the

same naming conventions and numbering precedence. The remaining chapters will further detail the functions of

the project and limitations of the application.

# 2. General Description

This project serves a tool intended for people who play the wildly popular roleplaying game Dungeons and Dragons. The concept of this application is a character building tool, so that people who are unfamiliar or not used to the character building process may actually develop functional and wonderfully created characters.

## 2.1 Product Perspective

This product is a new self-contained product that is utilizing the Dungeons and Dragons roleplaying system to make a character creation software that aims to provide benefits to both new players and returning players. This product is designed to be used to supplement a player intending to create a new character, and provides that service so that people may play their own game of dungeons and dragons. The larger “system” of dungeons and dragons does not require this product. However in our team's observation of the player base of this game, a character builder that focuses on user choice would find a strong home inside the community of people involved with the game.

## 2.2 Product Functions

Major Functions:

* Accessible as a standard Windows application.
* Create a character for Dungeons and Dragons.
* Display character sheet upon finishing the creation of a character.
* An intuitive user interface for the user to be guided through character creation.
* The program must let the user make decisions in regards to building the character.
* The program must explain what options the user has and how they affect their stats.

## 2.3 Users and Characteristics

**User Classes in order of importance**

1. **New D&D Players**: Inexperienced, need the step-by-step creation process, need character features explained to them.
2. **Experienced D&D Players**: Are already knowledgeable about D&D and would use the software just for convenience.
   1. **People Trying to Get Others Into D&D**: This group is a subset of the above players, they are generally experienced and want to get more people into their hobby/playgroup, and intend to use this product to make the skill gap a smaller one for people to jump.
3. **The Development Team:** This group will be the developers as we intend to test, use, and expand upon this project overtime, and will need to dedicate time where we as a team put ourselves into the shoes of the other user types.

## 2.4 General Constraints

Below are several constraints that we believe will have a significant impact on the project:

* **Time**: Due to this project being tasked as a class project in a single semester, we inherently have limited time to define, implement, and release our application.
* **Skill/Ability**: Our team is new to using .NET Framework and the technologies that go along with it. As a result, we will have to place a significant amount of time to learn and practice with the environment.
* **Group Members:** Our entire team is comprised of 4 people. Therefore, there is a limit as to how much we can complete due to our small size.

## 2.5 Assumptions and Dependencies

As of the creation of this SRS document, our current plan is to develop a Windows based application. This, therefore, implies that the client machine must meet the minimum Windows 10 hardware requirements in order to support the operating system that our software will use. The machine will also need a suitable internet connection to be able to access and download the files of the application. This machine will also need to have sufficient graphics processing power in order to properly display our WPF based front end, and be able to run and maintain a locally created SQL database. As we are planning to build the application using Microsoft’s latest development platform .NET Framework version 4.7.2, there might be requirements for the machine to install the necessary .NET Framework packages to be able to run the software.

In order to develop the application itself, it is necessary that all of our developers have access to a machine that is running either the latest Windows operating system (Windows 10) or a late release of Mac OS. These machines will need to be capable of effectively running Visual Studio and a variety of software to ensure the developers have access to all of the tools they need to work efficiently (such as a web browser, Git, etc…). These machines will also need to be able to install and support the necessary dependencies the project will need via an internet connection. For project storage during development, we will be utilizing a publicly accessible GitHub repository to both commit, retrieve, and document our project as it is developed.

As the project is in development, it is inevitable that the project can change and evolve, hopefully for the better. As a result, we might have to develop different solutions, such as migrating from a locally based database to utilizing an API to grab, edit, and view data.

## 2.6 Operating Environment

Windows desktop based software which will be used with a file system that will sit with the product that contains the user data for character creation.

# 3. Specific Requirements

## 3.1 Functional Requirements

### 3.1.1 FR 1

### TITLE: A Desktop Based Application

DESC: The user should be able to download this application onto their Windows computer, through an online distribution (such as a Google Drive folder), it will be free to download.

RAT: A method for the user to get access to the files of the application.

PRIO: 9 (Largest priority, the basis of all other requirements falls into having an actual application.)

RISK: 2 (Low risk: using Google as our cloud storage is one of the most accessible solutions we have to distribute the software. Google’s cloud services are used worldwide for consumers and businesses to store data, proving it is incredibly reliable. Therefore, if there are any problems with the storage solution, we are reliant on Google to provide a solution and make sure our repository is live.)

DEP: Google Cloud Services, internet access, available storage on the destination machine.

#### 3.1.1.2 Stimulus/Response Sequences

The user will need to navigate to our yet to be determined Google Drive repository. The user will then download the application files and make sure they allocate space for the files (the operating systems this will be installed on will automatically inform the user if they are low on drive space). The user will then run the application launching a window that greets them and prompts them for input.

#### 3.1.1.3 Functional Requirements

The software capabilities would be a program that compiles and holds the rest of the software together, this requirement is the base level upon which all others will be built from; a functional application that can be downloaded upon the users computer. The anticipated errors range from the standard failure to design the application in a manner that allows it to function, alongside other issues that may arise from downloading and setting up the software. This may include scenarios such as failure to fully install the files from the Drive repository, attempting to run the software on unsupported operating systems and devices, failure to provide necessary account permissions if required, insufficient hardware to run the application, and more.

##### 3.1.1.3.1 REQ-1:

The application requires a method to install the application and then have a functional interface that allows for movement into the other functional requirements. This is the base level upon which others will draw from.

### 3.1.2 FR 2

TITLE: A User Interface

DESC: There should be a user interface that allows the user to interact with the software in a clear and concise manner. The interface should be functional and not be poorly designed or strain the user in its use.

RAT: A method for the user to interact with the application.

PRIO: 8 (Large priority, this application needs to have user interaction, and this will be done through a user interface.)

RISK: 5 (Moderate risk: The user interface is what the end user will be working with for the entire lifetime of the application. Creating a useful yet intuitive user interface will ensure that customers will enjoy using the application and continue to use it. Therefore, great care must be taken to create a UI that is useful and effective.)

DEP: FR 1

#### 3.1.2.2 Stimulus/Response Sequences

The user will run the application and be greeted with a prompt to start character creation or to recover a previous character sheet to be displayed. Upon selecting the start creation they will be taken through the process listed in FR X, this creation process is detailed there. If choosing the second option, it will be detailed in FR 3. The user interface should be clear and designed so that players going through either selection will have a clear line of choices to make, and not be searching for a button to continue the process.

#### 3.1.2.3 Functional Requirements

The U.I should be tasked with guiding the users throughout the program. Functionality dictates that the user can move about the interface without becoming confused with too many choices, rather it should be presented to them in several smaller clumping of choices, with short explanations accompanying them.

##### 3.1.2.3.1 REQ-1:

##### The U.I. needs to have buttons programmed into it, that allow the user to enter other portions of the application, and these buttons need to be visually identifiable for the user.

##### 3.1.2.3.2 REQ-2:

The U.I will be present throughout the entire application, because of this reason the U.I needs to interact and be layered with each other functional requirements that will be getting user viewership.

##### 3.1.2.3.3 REQ-3:

Window in Window Functionality: Most WPF based user interfaces are built using the XAML markup language and feature graphically represented windows. The application will be comprised of these windows, including one larger "shell" window, which will encapsulate all other windows which will display inputs and data to the user.

##### 3.1.2.3.4 REQ-4:

Adaptive U.I.- To create an application that is useful for a wide variety of players, we plan to have an advanced view (all character traits are shown) and a beginner view (each section of data is split into "bite-sized chunks"). Partitioning the process, as well as providing additional notes for new users will help make the interface useful for new and old players alike.

### 3.1.3 FR 3

### TITLE: Displaying character sheet

### DESC: A user should be able to display their finalized D&D character sheet.

RAT: To allow the user to view the information they have submitted for their character.

PRIO: 8 (Large priority, the user must be able to view the information they have entered, this is one of the main purposes of the program.)

RISK: 4 (Low risk, if this requirement was not functioning properly it would be easily noticeable by the programming team. The method of implementation may be difficult as many variables are being utilized to display the character sheet.)

DEP: FR 1, FR 2, FR 4, FR 5, FR 6, FR 7, FR 8, FR 9, FR 10 (This is essentially the end result of all other functions, it is the last point most users will interact with.)

#### 3.1.3.2 Stimulus/Response Sequences

After the required information has been entered into the program, the user would be able to click a button to display their character sheet. The user will also be able to display a character sheet that has been created previously by loading the character sheet through the program.

#### 3.1.3.3 Functional Requirements

The function must take the previously entered data and output it in a manner that is readable to the user. This data

will be filled into sections that are clearly outlined in the application. The manner it will do this is using data from

user input

##### 3.1.3.3.1 REQ-1:

##### Take the user previously entered data, across all the sections.

##### 3.1.3.3.2 REQ-2:

Output the data in a clearly set out format, and display of the values previously gathered.

##### 3.1.3.3.3 REQ-3:

Request appropriate parameters for the character sheet by using a local database.

### 3.1.4 FR 4

### TITLE: Stat Generators

### DESC: A user’s stats will be generated, through either the random creation method or through the standard array.

RAT: To create the user’s stats, by random generation or through standard pick, needed for most character creation

actions.

PRIO: 6 (important due to character creation requiring this data throughout the rest of the process.)

RISK: 3 (Low difficulty due to it being either a set list being used to populate data, or it is several dice rolls.)

DEP: FR 1

#### 3.1.4.2 Stimulus/Response Sequences

This will either be a user choice of the basic set array of data, or choosing the other button that randomly creates the numbers, the next stimulus and response would be them choosing which stat that number goes into.

#### 3.1.4.3 Functional Requirements

This will be handled by either random number generation, or the user will pull from a hard set pool of numbers, then the user will decide which value goes where into the stats. This will be important for other sections of the project, it being random or the standard array does not actually matter much in the process.

##### 3.1.4.3.1 REQ-1:

##### Offer a choice of random number values or hard set standard array.

##### 3.1.4.3.2 REQ-2:

Slot each of the stats in turn from the pool of numbers per the user request.

### 3.1.5 FR 5

### TITLE: Skill/Ability Population

### DESC: This requirement is the generation of skills found on every character sheet. These skills are drawn from class, race, and stats previously generated.

RAT: This is the bulk of player number crunching, it draws the previously generated data and adds it up to the final data returned to the user.

PRIO: 8 (Very Important as these stats are used very often in D&D and need to be properly generated.)

RISK: 7 (Most amount of data handling, it will take time to make sure all the values are being generated correctly.)

DEP: FR 1, FR 2, FR 4, FR 10

#### 3.1.5.2 Stimulus/Response Sequences

The user will be told what the base of their skills are, then given the choices they could then further expand those skills/abilities. This section will fall after Stat Generation as those values are needed to correct populate these values. Which options the user will be able to select will vary depending on the choices found in their other sections.

#### 3.1.5.3 Functional Requirements

This requirement inherently requires access to the data generated in other sections, taking that and beginning to populate some sections, but then giving the player choices on what they need to do otherwise.

##### 3.1.5.3.1 REQ-1:

The values of previous sections will be gathered and begin population of skills and abilities.

##### 3.1.5.3.2 REQ-2:

The user will be prompted with some choices they may specialize in that will receive additional points assigned to.

##### 3.1.5.3.3 REQ-3:

##### The user will then be told their saving throw values, their specialized skills, and their passive perception score.

### 3.1.6 FR 6

### TITLE: Character Background Story Creation

### DESC: This section is where the player will type out their character’s backstory. This does not have any bearing on actual ability, and is purely for roleplaying reasoning.

RAT: This is needed, as D&D is a roleplaying game, allowing a user to add their own flair of character to the sheet is very important, and a central point of this is adding character description taken from the user.

PRIO: 2 (One of last features needing to be implemented, this is purely cosmetic in terms of coding, no other data draws from it.)

RISK: 3 (Low risk as it is taking user input strings and then displaying on the final output.)

DEP: FR 1, FR 2

#### 3.1.6.2 Stimulus/Response Sequences

The user will be prompted to fill in their characters backstory, they do not need to fill this data in as it is purely cosmetic and personal to their character. There are no values needed for the rest of the application in this section and therefore it is purely due to D&D being a role playing game that this section is relevant to the project.

#### 3.1.6.3 Functional Requirements

Be able to take user entered data and convey it to the display requirement when it is needed.

##### 3.1.6.3.1 REQ-1:

Take user input data.

##### 3.1.6.3.2 REQ-2:

Place it into the specified section and eventually allow it to be displayed in the end result.

### 3.1.7 FR 7

### TITLE: Character Hit Point Generation

### DESC: This is important for the concept of the game, as it tells how alive/injured a player character is. This requirement needs to properly generate the users hit point number.

RAT: This requirement will need data from the characters class, constitution score, and any other mitigating factors. Then the system will roll a die to generate a value that is added to the other data, and then it will output that data to the end result.

PRIO: 5 (Important to character creation, but very straightforward in implementation. It will be built using data that is previously input.)

RISK: 5 (Some amount of data management, but it falls behind the other dependent data that we will be generating.)

DEP: FR 1, FR 2, FR 4

#### 3.1.7.2 Stimulus/Response Sequences

This section will take place after the skill population, it is similar in that it takes some of those values but must be done independently as it also takes value from which class the user is playing then depending on class and constitution score the user will be given their hit point count.

#### 3.1.7.3 Functional Requirements

Take the user constitution score and the user class type, and then return the randomly generated value that would be in the correct range for the character.

##### 3.1.7.3.1 REQ-1:

Take in the data from the previous sections, compare the class to the proper random factor.

##### 3.1.7.3.2 REQ-2:

Randomly generate the number based on the constitution score and dice used, then return it to the final product at the end.

### 3.1.8 FR 8

### TITLE: Character Skill/Ability/Feats Choices

### DESC: This requirement is the user being able to choose what character development options that they will use the most often in their time playing D&D.

RAT: This requirement will be involving the storage

PRIO: 7

RISK: 5 (Lots of data on the program end so that will be a lot of setting up, however, the actual use of the data will be simply added to the end result after the user has selected the options.

DEP: FR 1, FR 2, FR 5, FR 10

#### 3.1.8.2 Stimulus/Response Sequences

The character sheet will be mostly filled out, what will be offered is a selection of abilities depending on what class the user has selected. This is important as this will be most of the characters combat ability being selected so the user must be fairly aware of their options. Short descriptions of the choices offered will be provided, with more advanced data being referred to the player's handbook.

#### 3.1.8.3 Functional Requirements

The user’s specified class options will be presented from the storage file and then selected by the user, then placed in their section for final viewing.

##### 3.1.8.3.1 REQ-1:

The options will be pulled from the storage system and displayed to the user.

##### 3.1.8.3.2 REQ-2:

The user will select their options and then they will be moved to the final viewing so the character knows what abilities they have.

### 3.1.9 FR 9

### TITLE: Character Editing

### DESC: This requirement is regarding a final function tasked with allowing the user to go through the process and edit a value they previously had marked down.

RAT: This would be useful for the user to do so that they may have created a character they are fully satisfied with, and designed per their choices.

PRIO: 6 (This is important for the user, and for the project as a whole due to the functionality it provides.)

RISK: 7 (This gets into the process of editing data, this can be tricky especially in the data that can change other values down the line, so it is important to design without any stale data being able to be formed.)

DEP: FR 1, FR 2, FR 3, FR 4, FR 5, FR 6, FR 7, FR 8, FR 10 (It must interact with every other function if implemented properly as it must draw and edit values in those sections.)

#### 3.1.9.2 Stimulus/Response Sequences

The user upon completion of the different sections will be able to view their character sheet, then be given the choice of editing the sheet to better fit their specifications, that will take the user to a selection of choices over what section they would like to edit.

#### 3.1.9.3 Functional Requirements

It must be able to display the character sheet, drawing from FR 3, and then allow the user to select a section or previously went through requirement and edit that previous value.

##### 3.1.9.3.1 REQ-1:

Display the character data.

##### 3.1.9.3.2 REQ-2:

Allow the user to select certain sections to return to that requirement and process through it again, this could drastically change the data, so if options like changing race/class it would prompt them to restart character creation as large quantities of data would now be invalid.

##### 3.1.9.3.3 REQ-3:

##### The user must repeat the process they have selected, this will return them to the edit character menu tree.

### 3.1.10 FR 10

### TITLE: Character Name and General Information

### DESC: This requirement will be centered on taking the user’s character name, and high level factors that make up a character, race class, alignment.

RAT: This requirement will be handled by taking the user input for name, then offering options for them to choose what race/class/alignment they would like. These values are important for some data generated further into the application.

PRIO: 6 (Slightly high, it is a simple but good place to start in terms of development, and is quite important for the rest of the project, if only because it serves some mildly useful purpose later on.)

RISK: 4 (Takes user input, and serves as some of the first options offered to the user, this makes it more important than most of the other user based choices as this sets up a lot of the following options.

DEP: FR 1, FR 2

#### 3.1.10.2 Stimulus/Response Sequences

This is simple in that it takes the user selection for race/class, and some other less important values for the context of the application. The race/class are the only values here that actually will change other data, race (human, elf, gnome, and other fantasy races) have unique skills/values that can change some factors of character creation. In a similar way, the class option can change what values belong where or what final data values will be generated. The other values of name, alignment, and player name do not actually convey any bonus to point values.

#### 3.1.10.3 Functional Requirements

Take the user input data, and compare it to the list of possible options, race to race, class to class, the others are just strings for the user to personalize the sheet.

##### 3.1.10.3.1 REQ-1:

Take the user input value for race/class, and the other less important data.

##### 3.1.10.3.2 REQ-2:

Compare to the list of options upon matching it will prompt them with what it determines to be their choices and they will confirm or deny, if they deny they will be prompted to repeat the first input, and will be prompt until they correctly enter the data so that it can be transferred to other sections.

## 3.2 Use Cases

### 3.2.1 Use Case #1

|  |  |
| --- | --- |
| **Use Case Name** | Basic Application Functionality |
| **Reference** | Section |
| **Trigger** | Launching the application will bring basic functionality and selections |
| **Precondition** | The application must be installed on the user’s computer and be able to be launched. |
| **Basic Path** | 1. The user must launch the program. 2. The user can then select to either view a previous character sheet or create a new one. 3. If choosing to create a new character sheet begin the process in Use Case #10. |
| **Alternative Paths** | 1. If the user selected to view a previous character sheet the application will then refer to Use Case #2. |
| **Postcondition** | The user will end up going through character creation as outlined in other use cases or they will simply view the sheet. |
| **Exception Paths** | The user may end the program at any time. |
| **Other** | The options of character creation or viewing are the most efficient way to guide users through the application. |

*Table 1: Use case 1*

### 3.2.2 Use Case #2

|  |  |
| --- | --- |
| **Use Case Name** | Viewing Character Sheet |
| **Reference** | Section |
| **Trigger** | The user reaches the end of the creation process OR the user selects to view from the beginning of the application. |
| **Precondition** | The user must have completed the data sections of the character and stored the file correctly. |
| **Basic Path** | 1. The user will either complete the sheet during character creation and finish editing it, or they will simply come from the start of the application to a previously finished sheet.  2. The user will have the sheet displayed so they may observe their stats and data. |
| **Alternative Paths** | The alternative paths are found in how to get here, and are referenced in other use cases. |
| **Postcondition** | The user will have the character displayed before them until they return to the start of the application or they end the application. |
| **Exception Paths** | The user may end the application. |
| **Other** | The user may only view as printing the sheet is out of scope at this time. |

*Table 2: Use case 2*

### 3.2.3 Use Case #3

|  |  |
| --- | --- |
| **Use Case Name** | Merging of data |
| **Reference** | Section |
| **Trigger** | The final step before viewing/editing the character sheet near the end of creation. Alongside use case #9 #2 as it. |
| **Precondition** | Other sections have been completed and the values have all been generated. |
| **Basic Path** | 1. The user must be moving to the final viewing of their character.  2. The user will not see, but all the data will be moved to the correct zone when finally output and displayed.  3. The data must be accurate to the character creation process. |
| **Alternative Paths** | N/A |
| **Postcondition** | The user will being view the character sheet as in use case #2 |
| **Exception Paths** | Ending the application without saving or finally viewing the character. |
| **Other** | The user will not interact with the data being moved, but will select to go to the final viewing of the sheet. |

*Table 3: Use case 3*

### 3.2.4 Use Case #4

|  |  |
| --- | --- |
| **Use Case Name** | Stat Generators |
| **Reference** | Section |
| **Trigger** | After use case #10 and #6. |
| **Precondition** | The user has completed the use cases #6 and 10 |
| **Basic Path** | 1. The user will be prompt to either randomly generate the values or pick from the standard array of numbers.  2. The numbers will be randomly generated, and then displayed for the user.  3. The user will be prompt with each stat starting with strength, and ending with charisma.  4. The user will slot a value from the list into that value, until all are filled. |
| **Alternative Paths** | 1. The user will see the standard array of numbers from the player’s handbook.  2. The user will be prompt with each stat starting with strength, and ending with charisma.  3. The user will slot a value from the list into that value, until all are filled. |
| **Postcondition** | The user will see the values they entered, and told they are moving on to use case# 5. |
| **Exception Paths** | The user could end the application without completing the process, progress will not be stored. |
| **Other** | The user values will be generated in the system, or pulled from the standard array. Both methods are basically the same in the functionality, but it really does not matter as long as the data is in the slots for later points. |

*Table 4: Use case 4*

### 3.2.5 Use Case #5

|  |  |
| --- | --- |
| **Use Case Name** | Skill/Ability Population |
| **Reference** | Section |
| **Trigger** | After use case #4 |
| **Precondition** | The user will have completed the basic low level. |
| **Basic Path** | 1. The stat data will be shown.  2. The application will then generate the skill stores from the state modifiers.  3. The saving throws for the stats will also be generated.  4. Any additions that come from race/class will be added here.  5. The process will end with displaying the skills before any choices are made. |
| **Alternative Paths** | N/A |
| **Postcondition** | The user will see their skills, before being taken to choose their specialized |
| **Exception Paths** | The user can end the application, their data will not be saved. |
| **Other** | This section is |

*Table 5: Use case 5*

### 3.2.6 Use Case #6

|  |  |
| --- | --- |
| **Use Case Name** | Character Background Story Creation |
| **Reference** | Section |
| **Trigger** | After use case #10 |
| **Precondition** | Have competed the name and general information section, and now prompt with the entering of background factors for the character. |
| **Basic Path** | 1. The character will be asked for their background, this can be chosen from a list in the player’s handbook that will be displayed.  2. The user will be asked for character personality traits.  3. The user will be asked for character flaws.  4. The user will be asked for character ideals.  5. The user will be asked for character bonds.  6. The user will be asked for a description of the character. |
| **Alternative Paths** | N/A |
| **Postcondition** | The user will see the data they entered, and moved on to use case #4. |
| **Exception Paths** | The user can end the application, their data will not be saved. |
| **Other** | The majority of this data is not actually needed for other sections, but are needed for actually playing D&D, so it should be included for the user. |

*Table 6: Use case 6*

### 3.2.7 Use Case #7

|  |  |
| --- | --- |
| **Use Case Name** | Character Hit Point Generation |
| **Reference** | Section |
| **Trigger** | After use case #5 |
| **Precondition** | The user must have correctly created the values either through random generation or standard array. |
| **Basic Path** | 1. The user will see their class, their constitution score, and what dice will be used to generate the random score.  2. The user will then have their hit point score generated between those values.  3. The user will see their hit point score. |
| **Alternative Paths** | N/A |
| **Postcondition** | The value given to the user |
| **Exception Paths** | The user can exit the application, this will not save their character. |
| **Other** | N/A |

*Table 7: Use case 7*

### 3.2.8 Use Case #8

|  |  |
| --- | --- |
| **Use Case Name** | Character Skill/Ability/Feats Choices |
| **Reference** | Section |
| **Trigger** | After use case #7 |
| **Precondition** | The user must have correctly created the values either through random generation or standard array and must have completed populating the data in skills and in hit point generation before this section. |
| **Basic Path** | 1. The user will be prompted with the basic number values for their skills.  2. The user will be prompt with which skills they could develop further.  3. The user will select their choices.  4. The user will see their choices and be moved to the next section.  5. The user will see the ability choices they can have, each with a brief description.  6. The user will select their ability choices.  7. The user will see the feat choices they may have, each with a brief description. |
| **Alternative Paths** | N/A |
| **Postcondition** | The user will see their selections, then be taken to the final decision of editing or viewing their completed sheet. |
| **Exception Paths** | The user can exit the application, this will not save their character. |
| **Other** | This use case will be pulling the most data that is stored regarding character creation, this data will be stored and pulled when requested. It will only display the data needed for that specific class/race option so that it does not bog the user down with to much random unneeded data. |

*Table 8: Use case 8*

### 3.2.9 Use Case #9

|  |  |
| --- | --- |
| **Use Case Name** | Character Editing |
| **Reference** | Section |
| **Trigger** | After use case #8, but before use case #2 |
| **Precondition** | Must have completed the other sections, |
| **Basic Path** | 1. The character data will be presented to the user.  2. The options of what section to edit will be displayed.  3. The user will select what section, and be returned to a version of that use case that will edit that section.  4. If they chose to edit Character Name and General Information they will be prompted with restarting character creation, as they will need to go through each section again because of the large changes that can occur if you edit that data.  5. After each edit they will be sent along to the menu of what to edit, when satisfied they can choose to complete the editing, and then be taken to use case #2. |
| **Alternative Paths** | N/A |
| **Postcondition** | The user will be in use case #2. |
| **Exception Paths** | The user can exit the application, this will not save their character. |
| **Other** | This use case can change a lot of data, and therefore if certain options are taken they will need to complete other sections of data, just in respect to how much data can become stale if not managed correctly. |

*Table 9: Use case 9*

### 3.2.10 Use Case #10

|  |  |
| --- | --- |
| **Use Case Name** | Character Name and General Information |
| **Reference** | Section |
| **Trigger** | After use case #1 |
| **Precondition** | Must have launched the program and this would be one of the first sections they interact with in the character creation process. |
| **Basic Path** | 1. The user will be prompt with entering the name of the user and the character.  2. Users will be prompt for which race their character is, from the player’s handbook.  3. Users will be prompt for which class their character is, from the player’s handbook.  4. Users will be prompt for what alignment their character is, of the few options allowed.  5. The data entered will be compared to possible choices, and then if valid will continue to use case #6 |
| **Alternative Paths** | N/A |
| **Postcondition** | The data entered will be displayed, before moving to the next section. |
| **Exception Paths** | Exiting the application will end the process. |
| **Other** | The data that is pertinent to the project is the race and class, the other are purely for flavor and character cosmetic. It is still important to store and return that data, as D&D is a roleplaying game and does require players to correctly add these little character bits of info alongside the important data. |

*Table 10: Use case 10*

## 3.3 Non-Functional Requirements

### 3.3.1 Performance

This project is not intended to be a severely demanding one. The project will have minimal graphical design, very small amounts of calculation, so its performance should be a very low burden for most computers.

### 3.3.2 Reliability

As long as the user satisfies the minimum requirements of the application found in section 2.5, the application will work as advertised when it is ready for deployment.

### 3.3.3 Availability

The program will be available on Windows. The program should be able to be downloaded and run locally, so there would be no availability hurdles.

### 3.3.4 Security

Concerning the installation of the software: Most modern browsers have basic antivirus checks when installing software online. As of now, the exact size of the application is unknown. However, it is more than likely that the project will be small enough for most web browsers to scan the files before completing a download. This is important, as our software will be publically available and can result in malicious clones of software if it is successful. Using Google as a repository grants our development team an impressive layer of security that will help from files being unknowingly tampered, removed, or modified from attackers.

As an offline application once installed, there should be no security concerns. If the user tried to open a file that wasn’t compatible with the application, the application would stop them. However, we cannot guarantee full functionality on an end user machine that is compromised with malicious software.

### 3.3.5 Maintainability

The program should be made in such a way that updating information and adding new options for character creation would be simple and non-conflicting. The biggest hurdle is our service as of now is completely offline. For updates we will have to rely on our consumers to check for newer versions to install. Most important is to ensure older builds are not wiped by potential updates. Luckily, the logic of the program is largely determined by the rules and guidelines of the Dungeons and Dragons game. As long as we strictly follow the guidelines, we can build a foundation for character data storage that will ensure the files will work for the lifetime of the application.

### 3.3.6 Portability

All the files needed for the application should be contained to one folder, allowing for program portability.

## 3.4 Design Constraints

Building an offline application means we as developers will have limited access to be able to update software after it is released. Our small team (comprised of 4 people) also means that we have a relatively small development team to build the project. As we have a deadline placed at the end of the semester, we will undoubtedly have to make compromises in order to make an application that fulfills our fundamental functional requirements.

## 3.5 Logical Database Requirements

A SQLite database file will be used to store organized data, but the data stored in the tables doesn't really interact with one another; and there is not so much data that size would become an issue.

## 3.6 Other Requirements

Focus on making the project one that is accessible to new D&D players, but also able to be a new tool for returning players that will not overburden with bogged down U.Is and confusing choice structure.

## 4. Sequence Diagrams

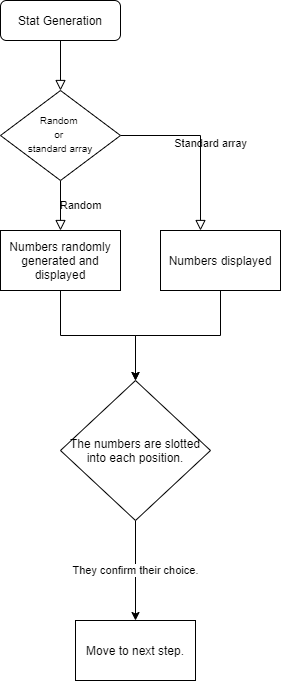


Figure 1: Character Stat Generation.

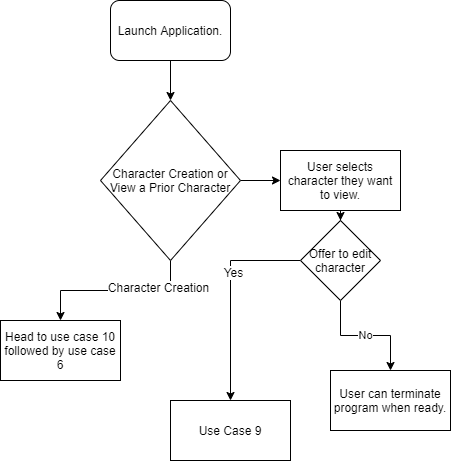


Figure 2 Character Creation or Viewing a Previous Sheet.

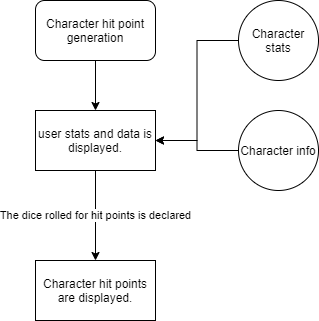


Figure 3 Character Hit Point Generation.

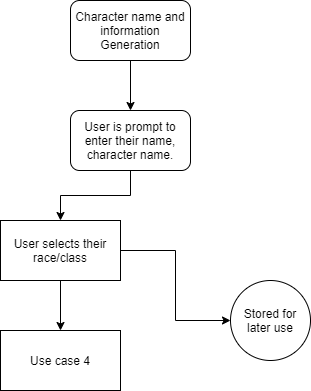


Figure 4 Character Name and Information Generation.

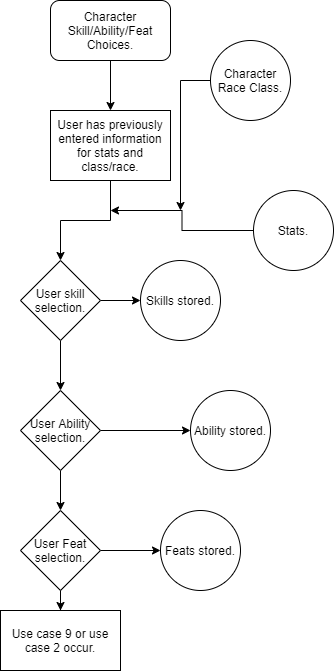


Figure 5 Character Skill, Ability, and Feat Selection.

# 5. Change Management Process

The process to update and change this document will be as follows:

1. Each proposed change must be brought before the team, to be discussed.
2. After brought forward the group will vote upon it being allowed into the document.
3. Upon ¾ people agreeing on the change, the one who brought the change forward will then amend the document.
4. All will examine the document after the change and sign off on it.