The SRS is graded using the following point distribution.

* 1. Cover Page, Revision History, and Table of Contents – 10 points
  2. Section 1 Introduction – 15 points
  3. Section 2 General Description – 20 points
  4. Section 3 Functional Requirements – 40 points
  5. Section 4 Non-Functional Requirements – 15 points

Each section will be graded as:

* exceeds expectation(100%)
* meets expectation(90%)
* needs some improvements(80%)
* needs major improvements(60%)

**1. Introduction**

**1. Purpose**

**2. Scope of the System Specified**

**3. Definitions, Acronyms, and Abbreviations**

**4. References to Supporting Documents**

**5. Overview of rest of SRS**

**2. General Description**

**1. Product Perspective**

**2. Product Functions**

**3. User Characteristics**

**4. General Constraints**

**5. Assumptions and Dependencies**

**3. Functional Requirements**

**4. Non-functional Requirements**

**1. Introduction**

**1. Purpose** - This SRS document describes The Botler v1.0 in regards to the following components: External Interface, Data Modeling, and Backend Architecture.

**2. Scope of the System Specified** - Discord is a highly dynamic social media platform designed to enhance the user’s experience beyond text messaging. Servers provide several interactive mediums for users to interact with each other. As such, it’s necessary for each server to employ the use of an admin team to moderate the chat environment.

Bots are Discord systems that automate several administrative and entertainment functions. The Botler will allow admins to automate common moderation tasks and manage tiers of leadership roles. The Botler will also allow users to enjoy various functions such as trivia games, music streaming services, and random cat pictures.

The Botler is an Open Source system which can function as a standalone system or enhanced with additional functionality. It can be determined by the admin team which, if any, additional functions should be added.

***3. Intended Audience and Reading Suggestions* -** This documentation serves as a guide for developers, validation teams, and documentation writers. This documentation serves as a resource for stakeholders, marketing staff, project managers, and users.

This document is organized as follows:

1. Section 1 provides an overview of this system.
2. Section 2 details the functionality, implementation, and documentation of this system.
3. Section 3 covers the interfaces used by this system.
4. Section 4 describes the functional requirements of this system.
5. Section 5 reviews nonfunctional and miscellaneous requirements of this system.

It’s recommended to begin by reviewing Sections 1 and 3 of this document. Developers should review Section 2. Project Managers should review Sections 4 and 5. All other readers should peruse relevant sections at their discretion.

**4. *References to Supporting Documents***-<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

Any external document, specifications? if not, say “None.”

*-Link to manpages & website GUI?*

**2. General Description**

**1. Product Perspective -** It lives on a server and interfaces with Discord (the chat application) and talks to the server (Discord and computer server).

**2. Product Functions -**

-Trivia

-Moderation

-Entertainment

-Plays you ad-free music from YouTube

-Shows you cat pictures

**3. User Characteristics** -

-Gamers

-People who want to chat with their friends

-Moderators on Discord servers

-Need to know how to use a computer or a mobile device and the internet

**4. General Constraints**

**-**Discord constraints

-Can’t be in two voice channels at once

-We don’t have a beefy server

-Limited time

-We don’t know all the computer languages

- express such things as host hardware limitations, interfaces, and implementation language requirements. Provide a general description of any other items that will limit the developer's options for designing the system.

**5. Assumptions and Dependencies**

**-**Discord will continue to work

-Discord will continue to allow unlicensed bots/homebrew

-People know what Botlers are

-Having access to the server(s)

-Access to the web panel/documentation

-Internet continues to exist

**-** List and describe each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but any changes to them can affect the requirements in the SRS. For example, an assumption might be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change.

**6. User Documentation -**

-Man pages available online and on GitHub

- Help command

List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.

Explain what you think should be provided. We won’t actually create these, but for a real product, what would you include?

**3. Functional Requirements**

The functional requirements are going to be written in narrative form identified with numbers. Each requirement is something that the system SHALL do. Thus, it has a common name of a *shall* list. You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.

-Node.js  
-Discord

**\*\*\* Cat Photo Man \*\*\***

**Objective**:

* Look at cat photos

**What is Source**:

* Sad, needs cats

**Trying to Achieve**:

* Happiness and meowvelous things

**Priority**:

* High

**Who is Source**:

* Dominic Trentadue

**Actors**:

* Users/Moderators

**Flow of Events**:

* Basic - User accesses Discord server > User sends cat pic command > Bot returns cat pic
* Alternative - User accesses Discord server > User sends invalid command > No response from bot
* Exception - User accesses Discord server > User sends cat pic command > Bot returns message with no cat pic embedded due to Cat API site being down or some other error

**Includes**:

* N/A

**Preconditions**:

* User must be logged into Discord
* The Botler must be on the Discord server in question

**Postconditions**:

* Cat picture returned to server

**Notes/Issues**:

* Currently (04-12-2021), no way to test or resolve if Cat API website goes down

**\*\*\* Moderator \*\*\***

**Objective**:

* Monitor for offensive content in the chat and remove it
* Option to automate user bans based on number of warns
* Log specific Discord server events in an audit log
* Automate assignment and removal of emoji roles based on user reacts in chat

**What is Source**:

* Ensure security of the server by automating common moderation functions
* Ensure safe and fun chat environment

**Trying to Achieve**:

* Secure, safe, and fun chat environment
* Moderation made easy

**Priority**:

* High

**Who is Source**:

* Kelly and Rene

**Actors**:

* Moderators

**Flow of Events**:

* Basic - Moderator accesses server > Moderator sends help command > Moderator clicks link to Admin Menu > Moderator enters new banned word and clicks “Add Word”
* Alternative:
  + Moderator accesses server > Moderator sends help command > Moderator clicks link to Admin Menu > Moderator enters existing banned word and clicks “Remove Word”
  + Moderator accesses server > Moderator sends help command > Moderator clicks link to Admin Menu > Moderator clicks “Clear All Words”
  + Moderator accesses server > Moderator sends help command > Moderator clicks link to Admin Menu > Moderator clicks “Show All Words”
* Exception - N/A

**Includes**:

* N/A

**Preconditions**:

* Moderator must be logged into Discord
* The Botler must be on the Discord server in question

**Postconditions**:

* GUI updates JSON object storing array of banned words

**Notes/Issues**:

* N/A

**\*\*\* Trivia Lover \*\*\***

**Objective**:

* Play a game of trivia with friends

**What is Source**:

* Include a social interaction / entertainment function with The Botler

**Trying to Achieve**:

* Make The Botler fun

**Priority**:

* Medium

**Who is Source**:

* Kimmi

**Actors**:

* Users/Moderators

**Flow of Events**:

* Basic:
  + User accesses Discord server > User sends trivia command > Bot returns trivia info and question > Bot waits 30 seconds > Bot sends answer to question
  + User accesses Discord server > User has played a complete game of trivia > User sends prize command > Bot returns picture of a Dev Cat
* Alternative - User accesses Discord server > User sends trivia help command > Bot returns trivia tutorial info
* Exception - User accesses Discord server > User sends trivia command > Bot returns no data due to CSV being unavailable, not loading, or corrupted, or some other error

**Includes**:

* N/A

**Preconditions**:

* User must be logged into Discord
* The Botler must be on the Discord server in question

**Postconditions**:

* Users are able to play trivia

**Notes/Issues**:

* Currently (04-12-2021), no way to test or resolve if imgur website goes down

**\*\*\* Musician \*\*\***

**Objective**:

* Listen to music ad-free from YouTube or SoundCloud

**What is Source**:

* YouTube and SoundCloud have ads, so we want to provide an ad-free space
* Can listen to music with friends

**Trying to Achieve**:

* Enjoy music alone or with friends

**Priority**:

* Medium

**Who is Source**:

* Lat

**Actors**:

* Users/Moderators

**Flow of Events**:

* Basic - User accesses Discord server > User joins voice channel > User sends play command > Bot plays song > Then:
  + User could send pause command >
  + User could send shuffle command >
  + User could send loop command >
  + User could send search command >
  + User could send stop command >
  + User could send skip command >
  + User could send look at what’s playing now command >
  + User could send queue command >
  + Bot plays all songs in queue > Bot exits voice channel
* Alternative - User accesses Discord server > User joins voice channel > User sends play command > Bot returns message saying it could not find the song from YouTube or SoundCloud and advises the user to try searching for the song > User searches for song > Looped
* Exception - User accesses Discord server > User joins voice channel > User sends play command > Bot returns no response due to YouTube being down, SoundCloud being down, or some other error

**Includes**:

* N/A

**Preconditions**:

* User must be logged into Discord
* The Botler must be on the Discord server in question

**Postconditions**:

* Botler plays music in a voice channel

**Notes/Issues**:

* Currently (04-12-2021), no way to test or resolve if YouTube or SoundCloud go down

The overall use case diagram should be here.

The text description of each use case should follow.

3.1 Use case name and identifier

1. Objective - What is the ultimate objective of the use-case. What is it trying to achieve? What was the source of the use-case requirement?

2. Priority – The overall priority of this use-case (Low, Medium, High)

3. Source – Who is the main source of this use case. Who cares most about this functionality? This should be the one person you would ask if there is a question about this use-case. (Make up a name and cite their: John Smith (End-user) here.)

4. Actors - Who is involved in the use-case? Which actors/stakeholders?

5. Flow of Events

5.1. Basic Flow - flow of events normally executed in the use-case

5.2. Alternative Flow(s) - a secondary flow of events due to infrequent conditions

5.3. Exception Flow(s) - Exceptions that may happen during the execution of the use case

6. Includes - other use case IDs that are referenced in steps in the flow of events.

7. Preconditions - Any condition that must be satisfied before the use case begins. If the condition is “User is logged in”, then the first step of the use case is NOT “User logs in”. They are already logged in if that is a pre-condition!

8. Post conditions - The conditions that will be satisfied after the use case successfully completes

9. Notes/Issues - Any relevant notes or issues that need to be resolved

**4. Non-functional Requirements**

The same format as the functional requirements is to be used for the non-functional requirements. You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived. Performance Requirements, Safety Requirements, Security Requirements, S/W Quality Attributes

-Security requirements (authentication key)

-Follows the community guidelines (business rules)

-Open-source license

-Has to have a name and user picture