# **SWE 3313**

**Software Requirements Specification**

**Document**

**Project Name: ARIA Active Responsive Integrated AI**

Team #4

Jordyn Jones, Lewis Dante, Will Martin, Jackson Nuckles, Joshua Onuoha, Phillip Pham, Miguel Ramirez

# **Team Members:**

A: All team members contributed to the document.

SWE 3313

***Version: Beta 0.0***

***Date: 10/08/2023***

SWE 3313

# **TABLE OF CONTENTS**

1. Introduction …………………………………………………………………………………. 4

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms, and Abbreviations

1.4 References

2. Specific Requirements ……………………………………………………………………. 5

2.1 External Interfaces

2.2 Functional Requirements

2.3 Non-Functional Requirements

2.3.1 Performance

2.3.2 Reliability

2.3.3 Availability

2.3.4 Security

2.3.5 Maintainability

2.3.6 Portability

2.4 Database Requirements

2.5 Design Constraints

3. Use Case Models ………………………………………..…………………………….. 7

3.1 Use Case Diagrams

3.2 Use Case Descriptions

**1. Introduction**

The Software Requirements Specifications (SRS) document outlines the functional and non-functional requirements for the development of a system designed to facilitate the seamless exchange of content between various platforms. The primary goal of this system is to enable users to utilize Python code to send images from Discord, along with text generated from Chat GPT, to the Instagram API.

### **1.1 Purpose**

The purpose of this Software Requirements Specifications (SRS) document is to lay out both functional and non-functional components of our automated Instagram bot, which is designed to read direct messages (DMs) from Instagram and respond in real-time using the ChatGPT API. This SRS serves to define features, constraints, and applications of our software for whoever may read this document.

The intended audience for our SRS would be the software development team to give them a clear and concise overview of what needs to be included, as well as team managers to track project progress.

### **1.2 Scope**

The software to be produced here is ARIA (Active Responsive Integrated AI). The software should be able to read Instagram DMs and, using the ChatGPT API, write a response in real-time to interact with other Instagram users. The bot should be able to create Instagram posts either using the Midjourney API or from a database of images. The end goal of the bot is to facilitate seamless communication and content sharing across these platforms.

**1.3 Definitions, Acronyms, and Abbreviations**

DMs: Direct Messages, referring to private messages in our context, specifically Instagram Direct Messages.

### **1.4 References**

(1) ChatGPT API Documentation, OpenAI, Version 2.0, July 2023.

(2) Instagram API Documentation, Facebook, Version 3.5, June 2023.

(3) Discord API Documentation, Discord, Version 196.19, Sept 2023.

## **2. Specific Requirements**

**2.1 External Interfaces**

Inputs:

User Messages: Users can send messages to the Instagram chat bot.

User Login: User login credentials for Instagram.

Instagram API: Access to Instagram's API for communication.

Chat GPT API: Access to the Chat GPT API for generating text responses.

Image Data: Data for creating Instagram posts.

Outputs:

Bot Responses: Text responses generated by the chat bot.

Instagram Posts: Newly created and published Instagram posts.

Read Messages: Incoming direct messages from users.

Processed User Requests: Responses to user requests.

System Status: Information regarding the bot's operational status.

### **2.2 Functional Requirements**

Send and Receive Messages:

Users send messages to the bot.

Bot receives and processes messages.

Generate Responses:

Bot generates text responses using Chat GPT API.

Responses based on user's message.

Create Instagram Post:

Users create and publish Instagram posts through the bot.

Read Direct Messages:

Bot reads incoming direct messages from users.

Manage User Requests:

Bot handles user requests promptly.

### **2.3 Non-Functional Requirements**

**2.3.1 Performance**

- The system should be able to respond to user messages within seconds under normal operating conditions.

- The bot should be able to reply to users 90% of the time and respond to 51%+ of messages as a baseline within a few seconds.

- The system should support an infinite number of users and terminals simultaneously.

#### **2.3.2 Reliability**

- The bot should be responsive to users within hours of operation.

- The bot should be able to handle error handling, securely storing data, and perform reliably under load.

**2.3.3 Availability**

- The system should run 24/7, with occasional downtimes for maintenance or Instagram outages.

- If the bot fails to respond to 3 or more prompts, it should be restarted automatically.

**2.3.4 Security**

- The system should protect against unauthorized access, use, modification, destruction, or disclosure of data.

- It should utilize appropriate cryptographic techniques for securing data.

**2.3.5 Maintainability**

- The software should be designed for ease of maintenance, including modular design and clear code organization.

**2.3.6 Portability**

- The software should be portable to different host machines and operating systems.

**2.4 Database Requirements**

-The system will utilize a small database to store relevant information, including but not limited to the bot's name, age, birthday, and other pertinent details. This database will serve as a repository for maintaining essential information about the bot, ensuring efficient data management. The structure and design of this database will be aligned with the specific data requirements of the system to facilitate seamless access and retrieval of information as needed. Further details regarding the database schema and data models will be documented in accordance with the project's evolving requirements.

**2.5 Design Constraints**

-Skills and Experience: The development team consists of novice coders, which may impact the complexity and scale of the project. Efforts will be made to ensure the project remains within the team's skill level while providing opportunities for skill development.

-Hardware Limitations: Due to hardware limitations, the team may not have access to computers capable of running the software 24/7. This constraint could affect the system's availability and responsiveness.

-Monetary Limitations: The project budget is limited, and many of the tools and services required for development, such as cloud hosting or API subscriptions, may incur costs. Budget constraints will influence decisions related to software components and services.

**3. Use Case Models**

### **3.1 Use Case Diagrams**

Use Case Diagram: Content Exchange System



ACTORS 1,2,3



Actors:

User: Represents a user of the system.

Instagram Chat Bot: Represents the automated bot integrated with Instagram.

Chat GPT API: Represents the API for generating text responses.

Use Cases:

Send Direct Message

Actor: User

Description: Allows users to send a direct message to the Instagram chat bot.

Preconditions: User is logged into their Instagram account.

Postconditions: Message is sent to the bot for processing.

Generate Response

Actor: Instagram Chat Bot

Description: In response to a user's message, the chat bot generates a text response using the Chat GPT API.

Preconditions: Bot receives a message from a user.

Postconditions: A response message is generated and sent back to the user.

Create Instagram Post

Actor: User

Description: Allows users to create an Instagram post using the chat bot.

Preconditions: User is logged into their Instagram account.

Postconditions: A new Instagram post is created and published.

Read Direct Messages

Actor: Instagram Chat Bot

Description: The chat bot reads and processes incoming direct messages from users.

Preconditions: Bot is operational and connected to Instagram.

Postconditions: Messages are processed, and appropriate actions are taken.

Manage User Requests

Actor: Instagram Chat Bot

Description: Manages user requests, including responding to messages and creating posts.

Preconditions: Bot is operational and connected to Instagram.

Postconditions: User requests are handled effectively.

### **3.2 Use Case Descriptions**

*Requirements*: This use case must allow users to send a direct message to the Instagram chat bot, initiating a conversation with the bot.

*Pre-conditions*:

The user has an Instagram account.

The user is logged into their Instagram account.

The user is following the Instagram account associated with the chat bot.

*Post-conditions:*

The direct message is successfully delivered to the chat bot.

The chat bot generates a response within a reasonable time frame.

Scenarios:

*Normal Scenario:*

The user logs into their Instagram account.

The user navigates to their direct messages.

The user selects the chat with the Instagram chat bot.

The user types a message and clicks the send button.

The message is sent to the chat bot.

The chat bot receives the message.

The chat bot processes the message using the ChatGPT API.

The chat bot generates a response.

The response is sent back to the user's direct message thread.

The user receives and views the response.

*Alternate Scenario (Invalid Message):*

The user logs into their Instagram account.

The user navigates to their direct messages.

The user selects the chat with the Instagram chat bot.

The user types a message that includes invalid characters (e.g., emojis not supported).

The user clicks the send button.

The chat bot detects the invalid characters.

The chat bot sends an error message to the user, indicating that the message could not be processed.

*Alternate Scenario (Chat Bot Response Delay):*

The user logs into their Instagram account.

The user navigates to their direct messages.

The user selects the chat with the Instagram chat bot.

The user types a message and clicks the send button.

The message is sent to the chat bot.

The chat bot receives the message.

The chat bot processes the message using the ChatGPT API, which experiences a delay.

After a delay, the chat bot generates a response.

The response is sent back to the user's direct message thread.

The user receives and views the delayed response.