Requirements Analysis and Architecture Design

4+3 Solutions

Brandon Daryl Wanji [Leader]

Nathalie Atouba-Eyoune, Kenny Yang, Zihao Zhao,
Eric Kohler, Santiago Franco, Jiahui Xu, Xingzhi Chang

Contents

1 Introduction

MOBI is a platform to query a chatbot in order to receive timely responses about information regarding Brock University and the Niagara 2022 Games. The chatbot will scrape information off of both Brock University's and the Niagara 2022 Games official website and use that scraped information as the basis for answering queries.

This document follows the 4+1 view model and documents the architecture of the chatbot as well as the different views and components of the chatbot.

1.1 Purpose

The purpose of this document is to develop a blueprint of the chatbot after which the implementation would flow quickly. Also, the document provides a general architectural overview and bring the requirements of the system. Several different architectural views are used to illustrate different aspects of the system, both dynamic and static behavior of the system are described.

1.2 Scope

This software architecture document showcases an overhead view of the chatbot in regards to both the database as well as exactly how the chatbot chooses a response to a query. The 4+1 view model helps to showcase the chatbot in its entirety from its class structures to its use cases.

1.3 Overview

This document will detail the infrastructure of the chatbot.

2 Architectural Design

2.1 Scenarios

Audience – Stakeholders and end-users of the system

Area – The scenarios are represented by the use-cases of the chatbot through the multiple uses of the chatbot. These scenarios describe all the use cases of the chatbot.

2.2 Logical View

Audience – Designers, Programmers, Testing Staff

Area – The logical view will consist of the functional requirements of the chatbot specifically querying the chatbot will return an appropriate answer as well as the scraping of data into its proper categories.

This section will primarily describe the design details of the chatbot, primarily how it responds to queries as well as utilizes the scraping of data.

2.3 Process View

Audience – Designers, Programmers

Area – This area will consist of the non-functional requirements and primarily focus on the ordering of scraped data as well as the expected time it takes for a question to be answered.

This portion also goes through the expected run time, specifically of the chatbot from query to answer as well as the particulars of how the chatbot processes the categorized data and outputs a proper response.

2.4 Development View

Audience – Programmers, Designers

Area – This area will consist of the chatbot's specific decision-making process as it goes through the scraped data to pick an answer. This area will also feature the scraping of data and its placement within a JSON document inorder for it to be in a readable state.

2.5 Physical View

Audience – Programmers, Designers

Area – This area will be taking advantage of AWS Lambda to provide both the server and client side of the program. This area will also take advantage of DynamoDB for the storage of a database.

3 Architectural Goals and Constraints

3.1 Server Side

The chatbot will be hosted on the serverless computing platform, AWS Lambda. The chatbot will be a web based application that supports most common PC operating system (Windows, Linux, Apple). The chatbot will be using AWS API Gateway to have the chatbot connect to the database. A JSON file will be used for storing data scraped from Brock University's official website as well as for the chatbot to search for answers of queries. The JSON file will be stored using AWS DynamoDB.

3.2 Client Side

The chatbot will be accessible through a website. The chatbot is being developed as a website application first with plans of being integrated and updated into a phone application. The website will support users of the most common web browsers (Chrome, Safari, Firefox, Internet Explorer, etc.).

3.3 Security

The security features will be handled through AWS Lambda. As AWS Lambda handles data encryption, the team will mainly be responsible for what is already within the server. With the database being held within AWS DynamoDB The team will ensure that a small amount of people have access to the AWS Lambda account as well as using AWS CloudTrail in order to log activities.

3.4 Persistence

All data will be stored using AWS DynamoDB. The data on the server will be updated regularly either whenever changes are made on the scraped websites or on a regular basis with a set scheduled downtime reserved for updating the database.

3.5 Reliability and Availability

The system will be tested in several ways (Unit testing, Stress testing, system testing) before the final release to make sure the system meets all requirements. The serverless platform is likely to respond to concurrent users at any given time without losing the consistency, otherwise if issues are detected during testing, a queuing system or a limited amount of time with the chatbot could be implemented.

3.6 Performance

The chatbot will respond to queries made by users at a timely basis. More details on performance will be more apparent when the system is fully developed and tested.

3.7 Portability and Reuse

The first version of the chatbot will be a website application with plans of integrating it into an application suitable for Android or IOS. By utilizing AWS Amplify, the future versions would be able to easily translate over to an application. As an application taking advantage of the AWS Lambda all functions are adopting modularity design. Therefore, this chatbot is highly reusable that can be modified to answer any kind of information.

3.8 Development Tools

The project incorporates many development tools.

Programming language: Html, Css, Javascript, Python

Platform: AWS Lambda, AWS API Gateway, AWS Amplify

Database: JSON file, AWS DynamoDB

Schedule: Github Projects .

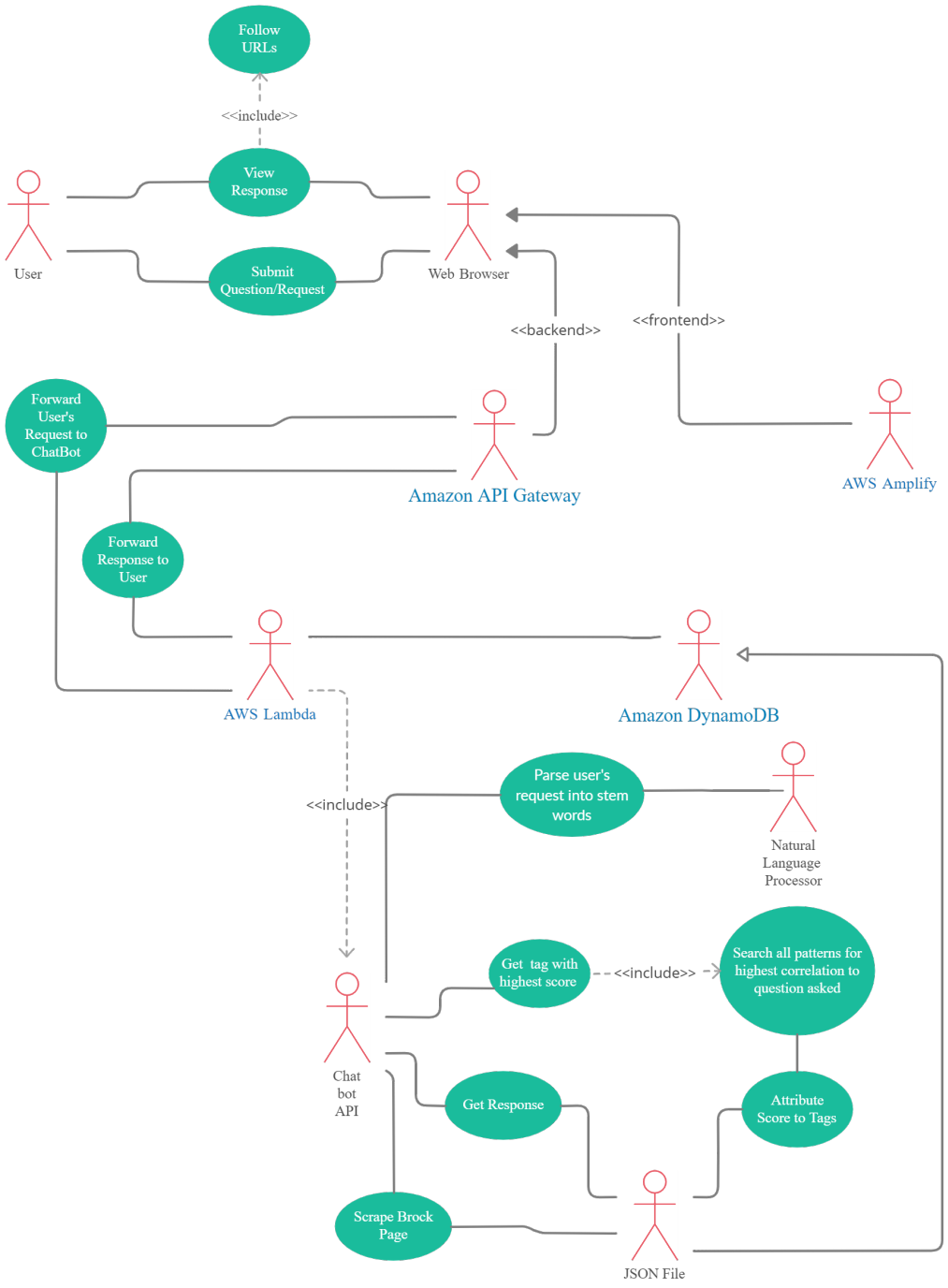
Communication: Discord, Microsoft Teams

3.9 Schedule

This project is designed using the agile method Scrum. In general, the development will have three phases. In the first phase, the general requirements and the design of the software architecture will be established. Following the initial phase, there are series of sprint cycles where each sprint cycle develops a segment of the system. The final phase will integrate all the project and complete all the documentation.

4 Use case view

4.1 Use-case Diagrams



4.2 Use-case Realizations

Use case name	Chatbot provides accurate answer to user.
Scenario	The response provided by the chatbot to user, is accurate and user is satisfied.
Triggering event	A question is submitted via the messaging platform.
Brief description	User types in a question in the messaging platform and the chatbot provides an accurate response to the user.
Actors	User, Messaging Platform and Chatbot, JSON File, Natural Language Processor.
Related use cases	None
Preconditions	• The messaging platform should successfully transmit the user's question to the chatbot. • The intent specified by the user's question should exist in JSON file.
Post conditions	The messaging platform should successfully transmit the chatbot's response to the user.
Flow of events	• User types a question in messaging app. • User clicks submit option. • Messaging app transmit submitted question to chatbot. • Chatbot uses Natural Language Processor to parse user's request into stem words. • Chatbot searches all patterns in JSON file of intents dictionary to find highest correlations to the question asked. • Chatbot attributes score to each tag; higher scores indicate more correlation. • Chatbot selects tag with highest score. • Chatbot randomly selects predefined response from selected tag. • Chatbot forwards selected response to user via the messaging platform.
Exception conditions	Connectivity problem / system is down.

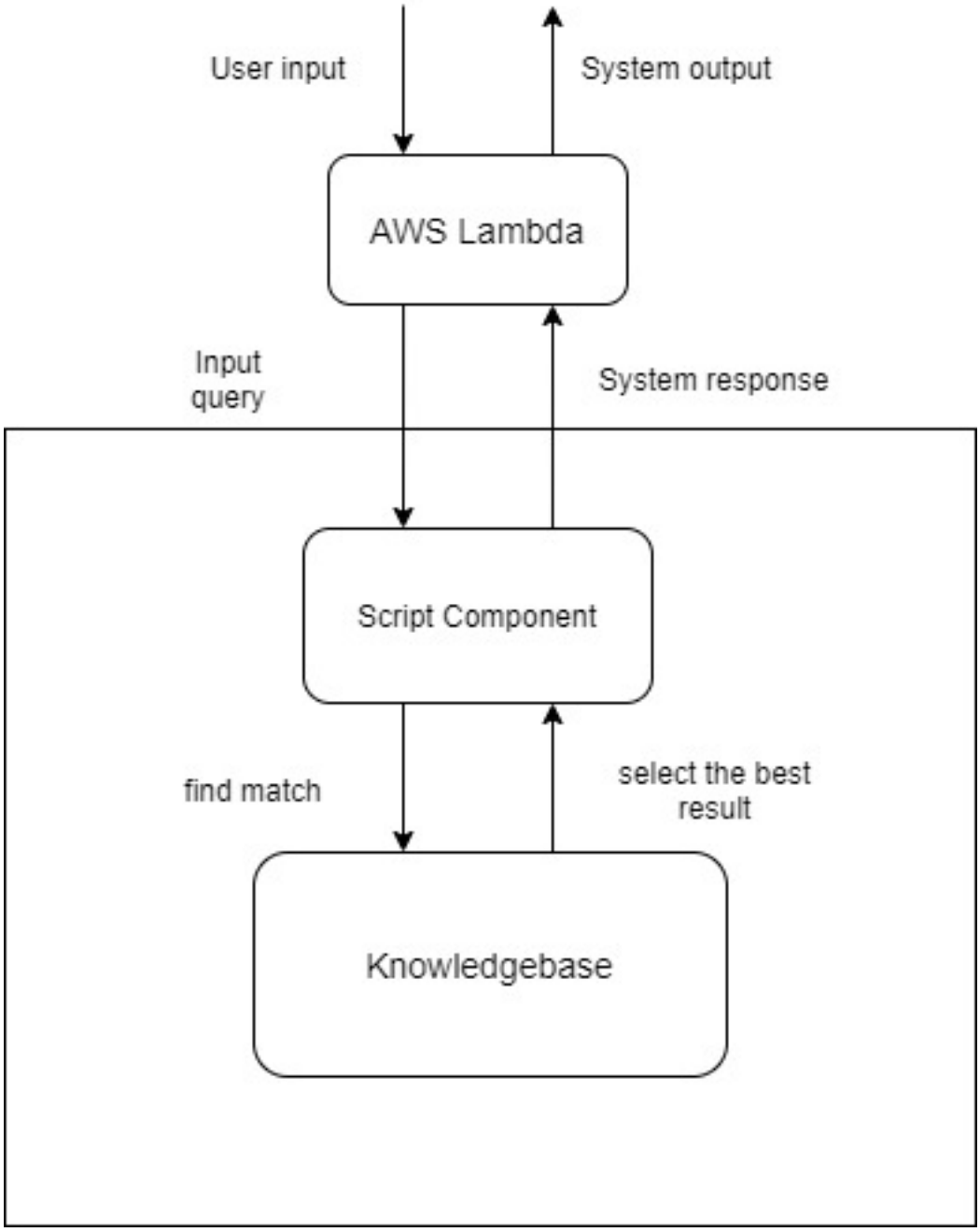
Use case name	Chatbot provides inaccurate answer to the user.
Scenario	The response provided by the chatbot to user, is inaccurate and user is not satisfied.
Triggering event	A question is submitted via the messaging platform.
Brief description	User types in a question in the messaging platform and the chatbot provides an inaccurate response to the user.
Actors	User, Messaging Platform and Chatbot, JSON File, Natural Language Processor.
Related use cases	Chatbot scrapes reference brock page.
Preconditions	The messaging platform should successfully transmit the user's question to the chatbot.
Preconditions	• The messaging platform should successfully transmit the chatbot's response to the user. • Chatbot should either provide an accurate response or Brock contact info.
Flow of events	• User types a question in messaging app. • User clicks submit option. • Messaging app transmit submitted question to chatbot. • Chatbot uses Natural Language Processor to parse user's request into stem words. • Chatbot searches all patterns in JSON file of intents dictionary to find highest correlations to the question asked. • Chatbot attributes score to each tag; higher scores indicate more correlation. • Chatbot selects tag with highest score, but tag does not reflect user's intent. • Chatbot randomly selects predefined response from selected tag. But response do not provide accurate response to user's question. • Chatbot forwards selected inaccurate response to user via the messaging platform. • User views inaccurate response and is not satisfied. • User submits another question (could be the same as before). • Chatbot scrapes reference brock page.
Exception conditions	Connectivity problem / system is down.

Use case name	Chatbot has no valid response to user's question.
Scenario	The chatbot did not find a response to the user's question in the JSON file.
Triggering event	A question is submitted via the messaging platform.
Brief description	User types in a question in the messaging platform and the chatbot does not have a response to provide to the user.
Actors	User, Messaging Platform and Chatbot, JSON File, Natural Language Processor.
Related use cases	Chatbot scrapes reference brock page.
Preconditions	• The messaging platform should successfully transmit the user's question to the chatbot. • JSON file should not contain the user's intent.
Post conditions	• The messaging platform should successfully transmit the chatbot's response to the user. • Chatbot should either provide an accurate response or Brock contact info.
Flow of events	• User types a question in messaging app. • User clicks submit option. • Messaging app transmit submitted question to chatbot. • Chatbot uses Natural Language Processor to parse user's request into stem words. • Chatbot searches all patterns in JSON file of intents dictionary to find highest correlations to the question asked. • Chatbot attributes score of zero or less then acceptable threshold to each tag. • Chatbot selects tag with highest score. • Chatbot scrapes reference brock page.
Exception conditions	Connectivity problem / system is down.

Use case name	Chatbot scrapes reference Brock page.
Scenario	Chatbots does not have a response that will satisfy user's question and must scrape brock page for useful link.
Triggering event	A question is submitted via the messaging platform.
Brief description	User types in a question in the messaging platform and the chatbot provides an inaccurate response to the user.
Actors	User, Messaging Platform and Chatbot, JSON File, Natural Language Processor.
Related use cases	• Chatbot has no valid response to user's question. • Chatbot provides inaccurate answer to the user.
Preconditions	All tags have score under acceptable threshold or user submitted message indicating they are not satisfied with chatbot's response.
Post conditions	• The messaging platform should successfully transmit the chatbot's response to the user. • Chatbot should either provide an accurate response or Brock contact info.
Flow of events	• Chatbot scrapes Brock website for user's intent and returns useful link. • Or user's intent not found and chatbot return Brock contact info.
Exception conditions	Connectivity problem / system is down.

5 Logical View

5.1 Overview

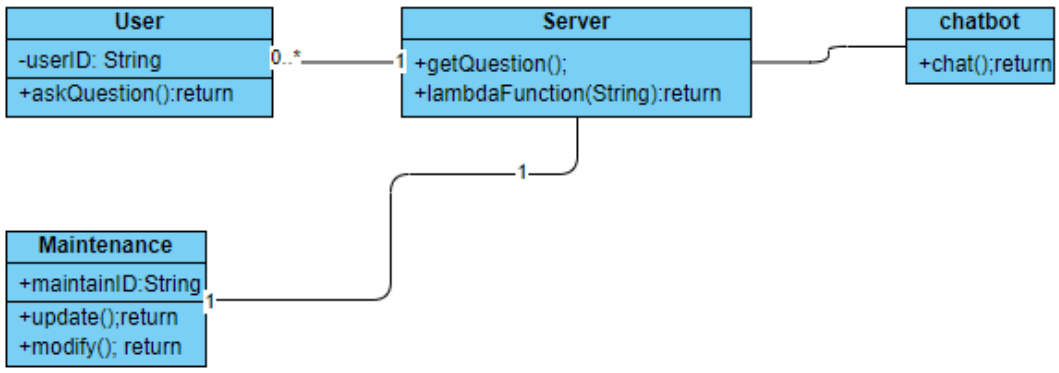


As shown in the figure,the message is initiated by the user which consists of a set of strings in order to request some relevant data. The input is sent from users to a web interface which here is the AWS lambda. once the user input is parsed, the query is forwarded to the appropriate component

and find the most precise information as the best match of the query. Then, the result is returned as a system response to the interface and received by the user.

5.2 Architecturally Significant Design Packages

5.2.1 class diagram



The Class diagram describes the structure of the system with classes, attributes and the relationship between the objects. The user sends request while the server transfers the query to the chatbot which then responds to the request. The chatbot class acts as a service provider, which takes a query and outputs a response. The Maintenance class updates the information.

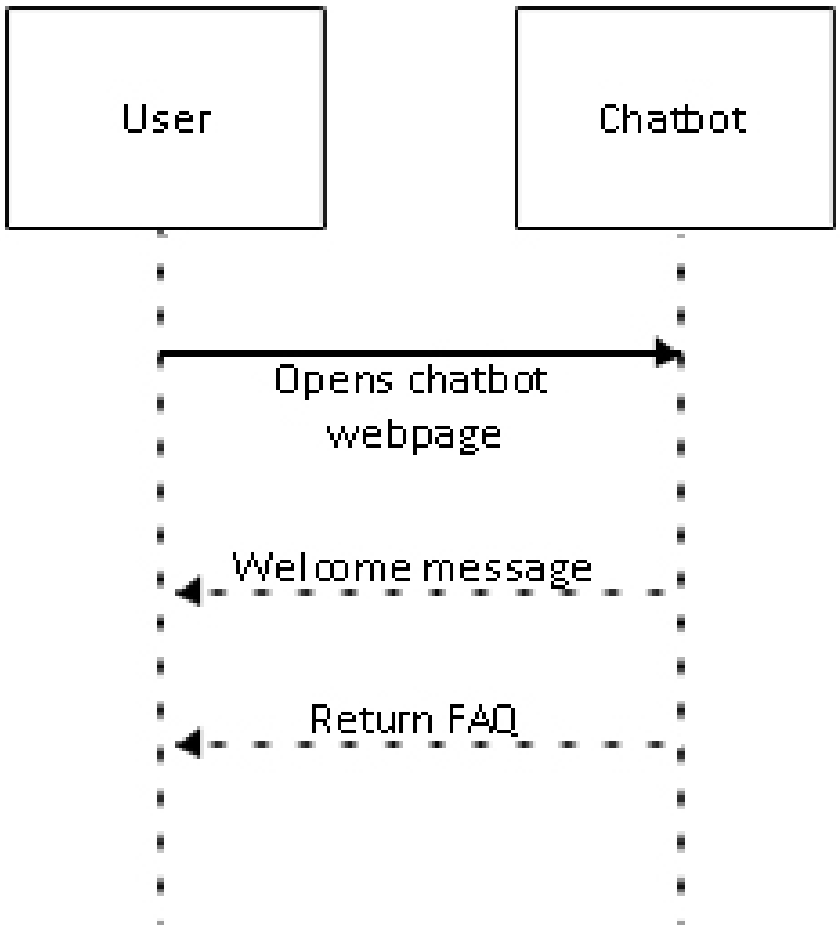
6 Process View

6.1 System Sequence Diagram

The system sequence diagrams for the chatbot's responses to user input are below.

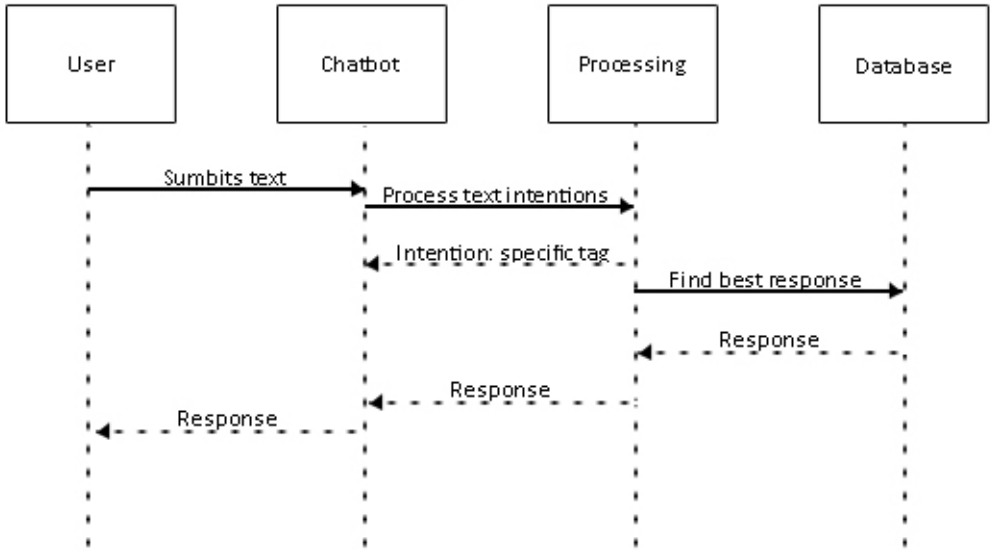
6.1.1 User enters the chatbot webpage.

When the user enters the chatbot web page the chatbot will first respond with a welcome message to the user and then supply the user with some frequently asked questions and their subsequent answers.



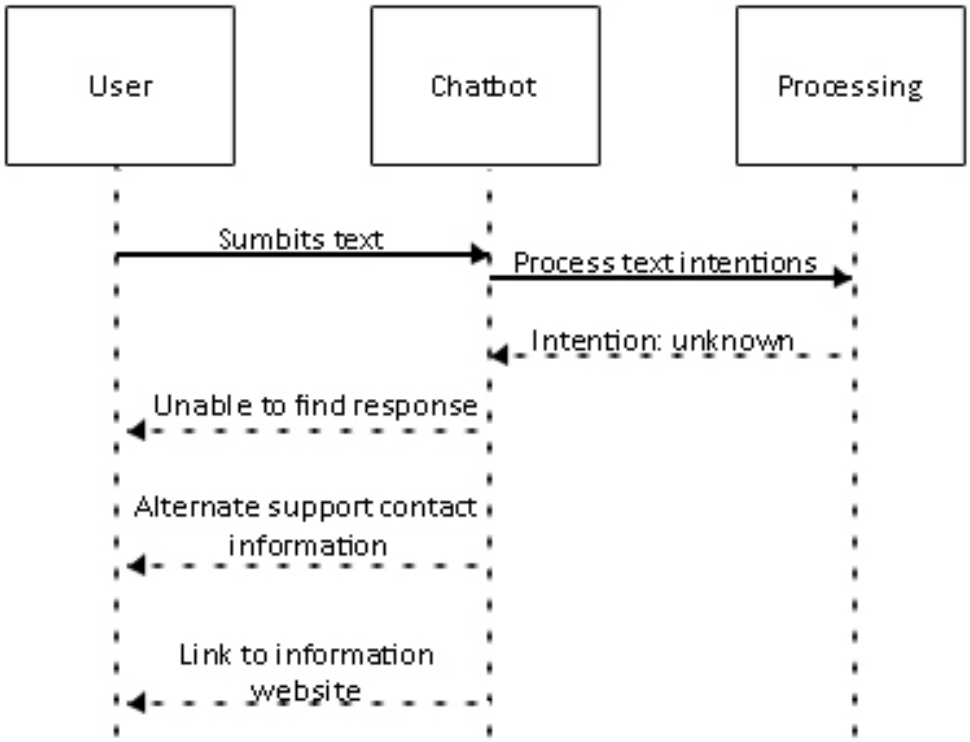
6.1.2 User inputs and submits text to the chatbot

When the user inputs and submits text to the chatbot, the chatbot will interpret the intentions of the text, whether it is a question, a greeting, a response to the chatbot, a goodbye, etc. Each intention is a specific tag which the database then uses to locate all the responses to choose from. The best response is chosen based on the rating of its correlation to the user's input. The chosen response is then returned to the user.



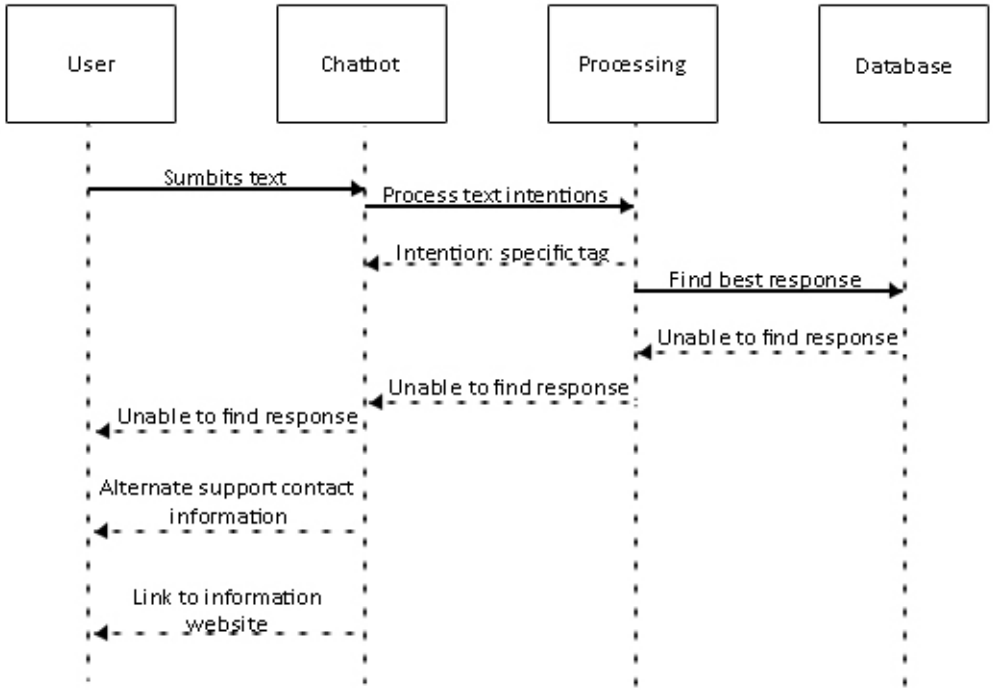
6.1.3 User submits text which cannot be interpreted

If the user submits text which the system is unable to interpret as it has no tags that can associate with the user's input, the chatbot responds with a message saying it was unable to find a response. Further information will be supplied to the user such as alternative support to contact by phone or messaging, along with a link to the website which the bot gets its information from.



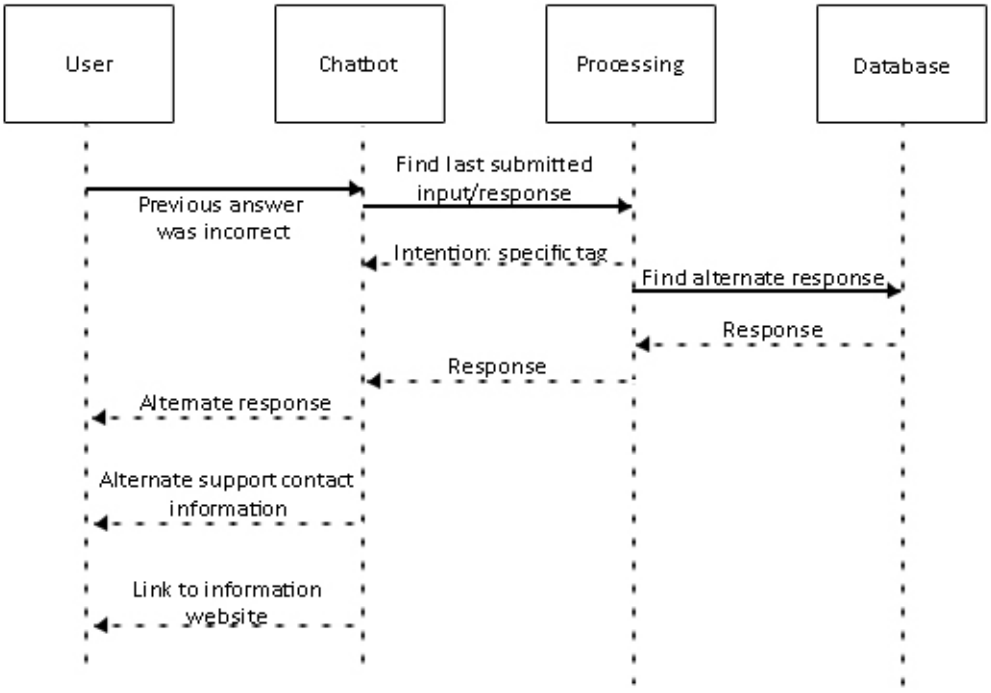
6.1.4 User submits text but no suitable response can be found in the database

If the user submits text however no suitable response can be found, then the chatbot will respond with a message saying it was unable to find a response. Further information will be supplied to the user such as alternative support to contact by phone or messaging, along with a link to the website which the bot gets its information from.



6.1.5 User notifies the chatbot that the previous response was incorrect

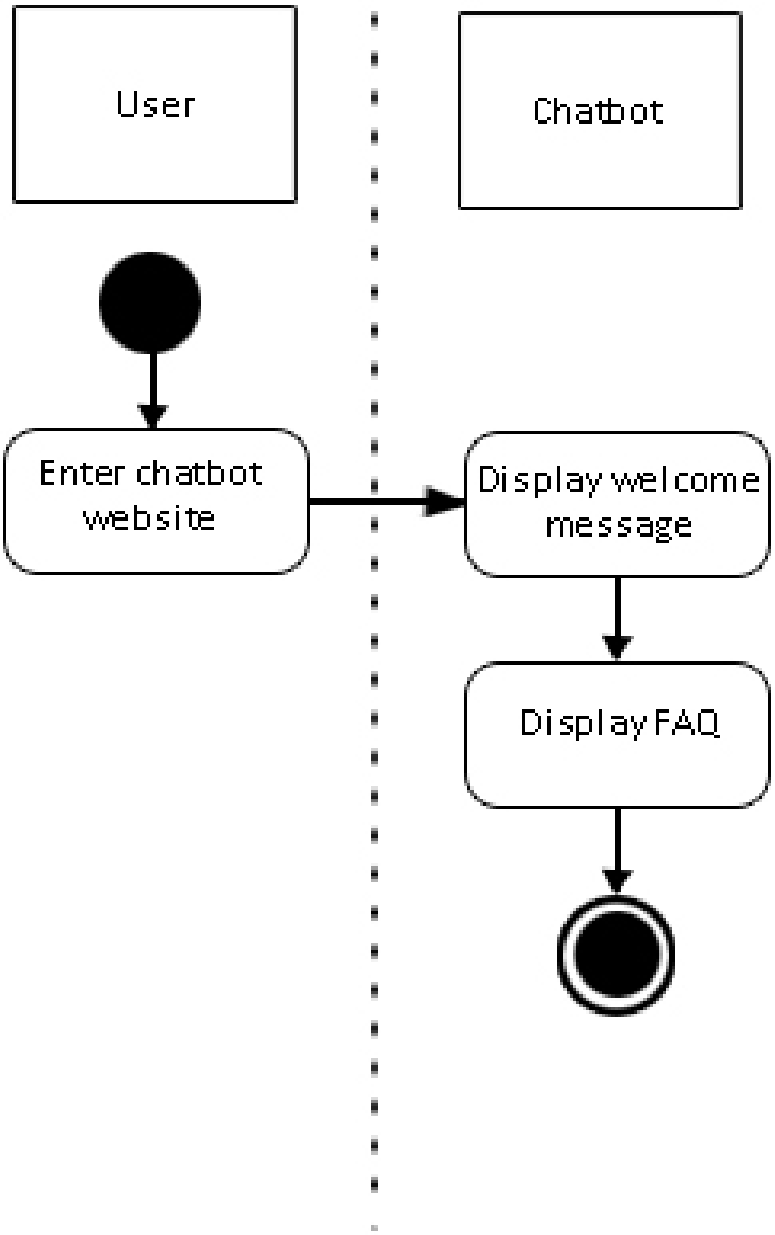
If the user notifies the chatbot that the previous response to their input was incorrect or unhelpful, the chatbot will find the user's last submitted input and calculate its intentions. It will also find the chatbot's last returned response to the user's input. From this the chatbot will find an alternative response in the database that is different from the previous response and return that new response to the user. Further information will be supplied to the user such as alternative support to contact by phone or messaging, along with a link to the website which the bot gets its information from.



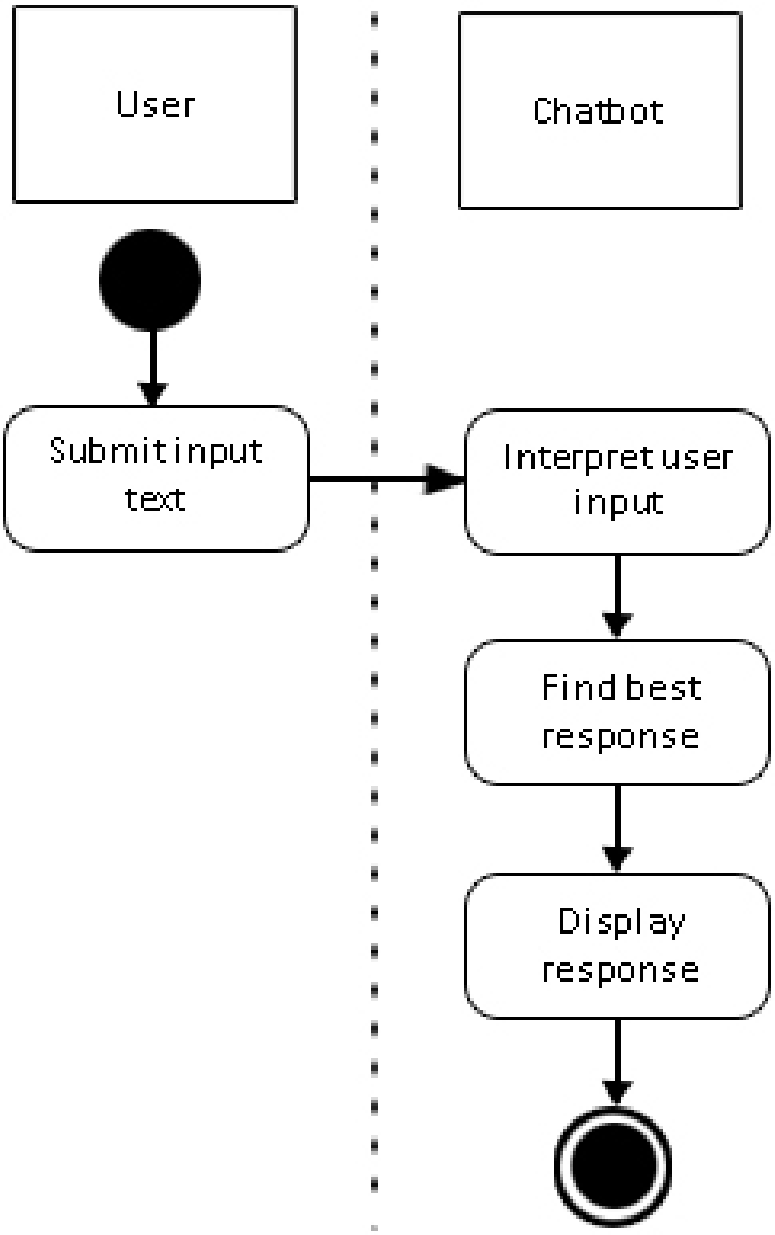
6.2 Activity Diagrams

The activity diagrams for the chatbot's responses to user input are below.

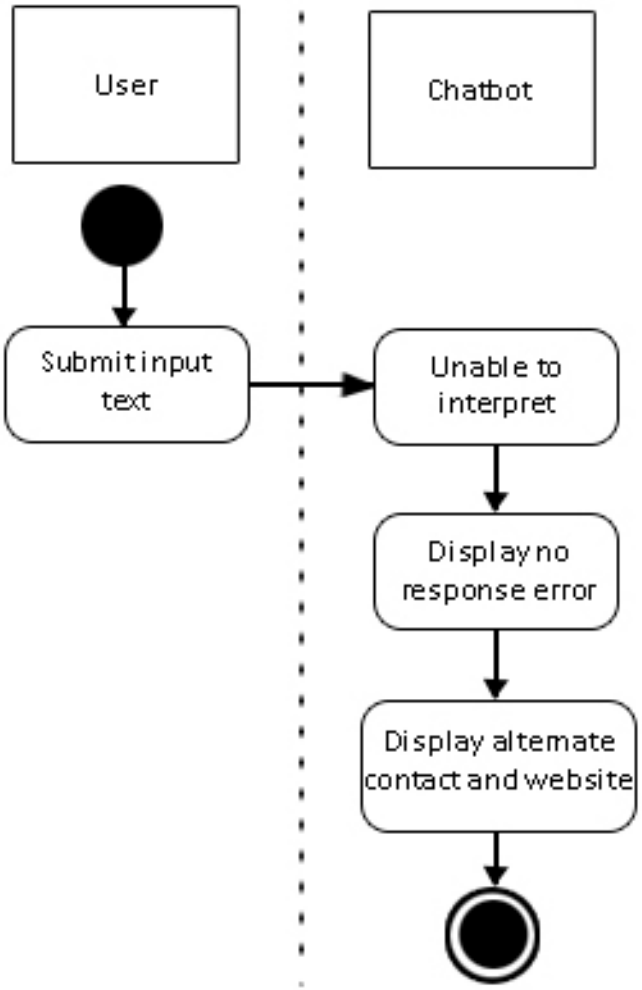
6.2.1 User enters the chatbot webpage



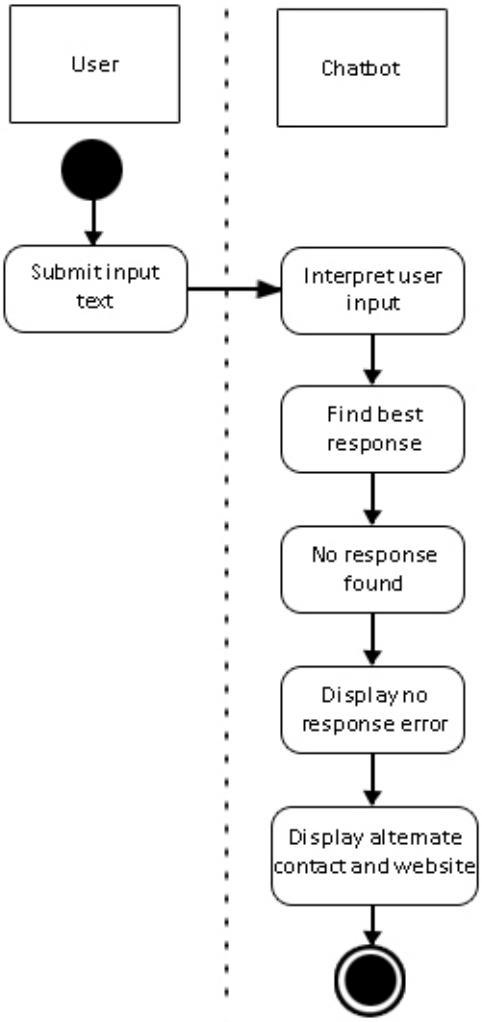
6.2.2 User inputs and submits text to the chatbot



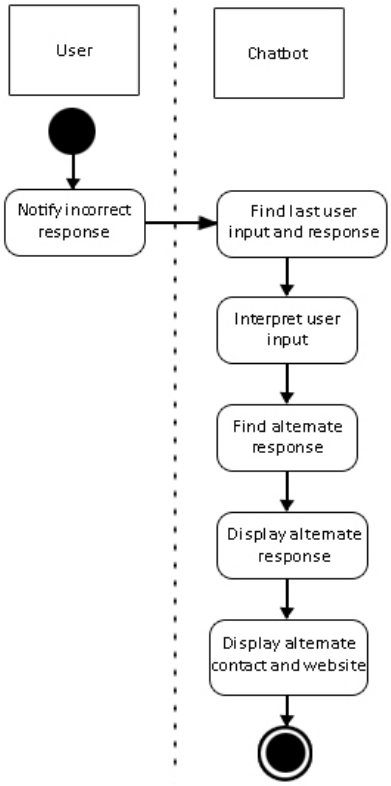
6.2.3 User submits text which cannot be interpreted



6.2.4 User submits text but no suitable response can be found in the database



6.2.5 User notifies the chatbot that the previous response was incorrect

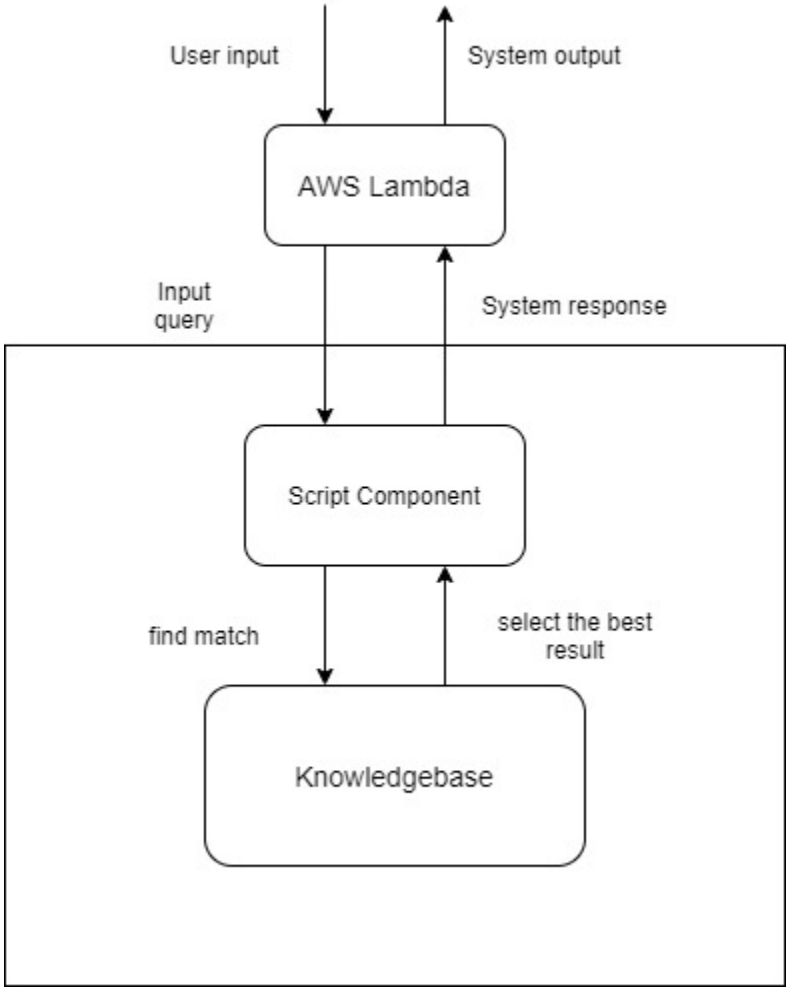


7 Deployment View

The first version of the chatbot will be a web application with plans of a client and server side to be hosted on AWS Lambda, as well as utilizing AWS DynamoDB in order to store scraped data from Brock University and the Niagara 2022 Games official websites. The UI/front-end will be handled through AWS Amplify and AWS Lambda will be communicating with AWS DynamoDB through the use of AWS API Gateway.

8 Implementation View

8.1 Overview



The first version of **MOBI** is planned to be a web application which receives a user's input and then outputs a corresponding response. The response will be sent to the chatbot through the usage of AWS API Gateway and a response that best corresponds to the query will be passed back to the user.

9 Size and Performance

The Application will begin on a web-based structure hosted on a serverless platform AWS Lambda with plans to be translated into a phone application. While web based the user will not have to install any components to their software system other than a web browser like google chrome.

The performance of the application is estimated to be O($N^2 + M^2$), where $M < N$

Using any web browser the User will access the system. All the functionality will be processed in the back-end. The front-end will be responsible for gathering data to and from the back-end to update the front-end with a response, therefore, requires very little computational power from the client side.