**ASPIRING MINDS**

**CHATBOT**

YOKAMURUGAN.K

Table of Contents

[1. Introduction 3](#_Toc61034002)

[2. Technology Stack 3](#_Toc61034003)

[3. Application Framework 4](#_Toc61034004)

[4. Sequence Diagram 4](#_Toc61034005)

[5. Chat History 5](#_Toc61034006)

[6. Test Case 5](#_Toc61034007)

[**6.1** **Screenshots** 6](#_Toc61034008)

[7. Collection 9](#_Toc61034009)

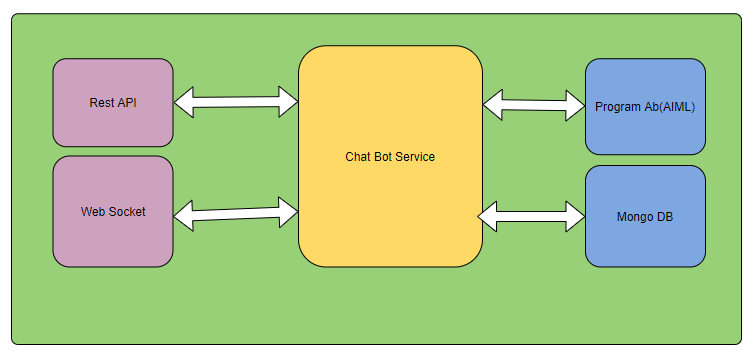
1. Introduction

Chabot is a software that can simulate a conversation (or a chat) with a user in natural language through messaging applications. Here, we have implemented to respond to basic user queries like interview application status and feedback.

1. Technology Stack

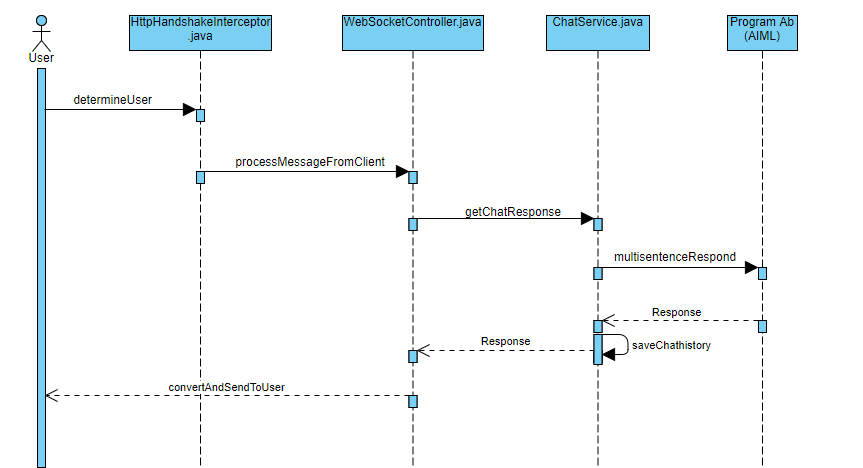
* **Spring boot -** Spring Boot is an open-source micro-framework maintained by a company called Pivotal. Spring module that provides the RAD (Rapid Application Development) feature to the spring framework. It provides Java developers with a platform to get started with an auto configurable production-grade Spring application. With it, developers can get started quickly without losing time on preparing and configuring their spring application.
* **Spring web socket -** The Web Socket protocol is one of the ways to make your application handle real-time messages. Application that sends messages back and forth between a browser and a server. It is a thin, lightweight layer above TCP. This makes it suitable for using “subprotocols” to embed messages.
* **Mongo DB -** Mongo DB is an open-source NoSQL database management program. NoSQL is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. Mongo DB is a tool that can manage document-oriented information, store or retrieve information. It supports various forms of data.
* **Program Ab (AIML) -** Program AB is the reference implementation of the AIML (**Artificial Intelligence Markup Language)** 2.0 draft specification. AIML is a widely adopted standard for creating chat bots and mobile virtual assistants.
* **Frontend (HTML. CSS & JavaScript) -**
* **HTML** (Hypertext Markup Language) code ensures the proper formatting of text and images for your internet browser. Without HTML, a browser would not know how to display text as elements or load images or other elements. HTML also provides a basic structure of the page, upon which **CSS** (Cascading Style Sheets) overlaid to change its appearance. One could think of HTML as the bones (structure) of a web page and CSS as its skin (appearance).
* **JavaScript** is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages.

1. Application Framework



Rest API is used to create applicant and job details. First, the Rest API calls the chatbot service and saves the data in Mongo DB. Once the details are created, run the application using localhost: 8080 and connect with a chatbot. The web socket connection will automatically create in the page loader. Once all configuration has been created, it will send a request from Web Socket through chat service to program AB using the pattern match (AIML) the response has been generated to return.

1. Sequence Diagram



1. Chat History

We are maintaining the chat history in mongo DB with random uuid per client.

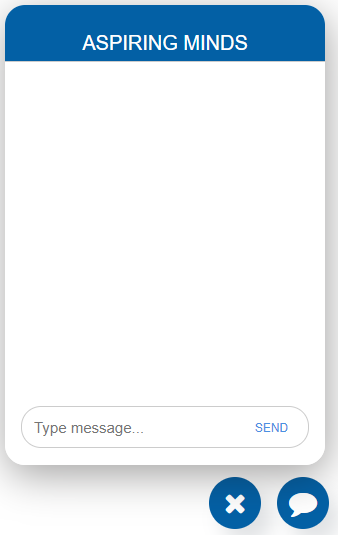


1. Test Case

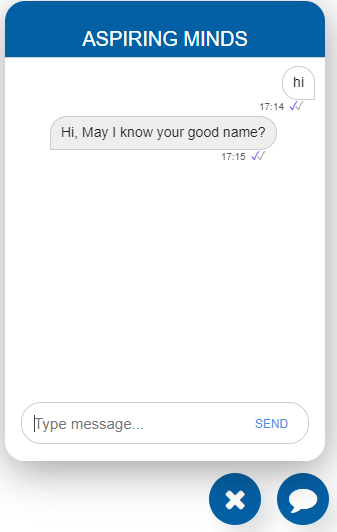
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| CB01 | To verify the initial Conversation | 1. Type message and press send button | Message = hi | User should get the response as “Hi, May I know your good name?” | As Expected | Pass |
| CB02 | To check name | 1. Type message and press send button | Message = I am Yoka | User should get the response as “Hello Yoka !! How can I help you ?” | As Expected | Pass |
| CB03 | To verify the user queries | 1. Type message and press send button | Message = I want to know my interview status | User should get the response as “Please input your email id to know the status” | As Expected | Pass |
| CB04 | To verify the status with valid email id | 1. Type valid email id and press send button | Message = yoka1@yopmail.com | User should get the response of applicant details | As Expected | Pass |
| CB05 | To verify the status with invalid email id | 1. Type invalid email id and press send button | Message = yoka@gmail | User should get the response as “Sorry, your email id is not matching. Please enter a valid email id” | As Expected | Pass |
| CB06 | To verify the status with invalid email id | 1. Type invalid email id and press send button | Message = yoka@gmail | User should get the response as “Sorry, your email id is not matching. Please enter a valid email id” | As Expected | Pass |

## **Screenshots**

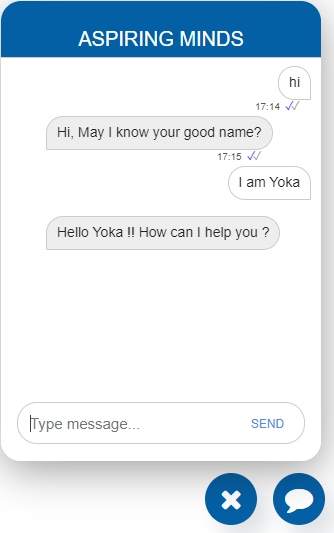
* When clicking on the chat icon the blank chat page will display to start the chat.



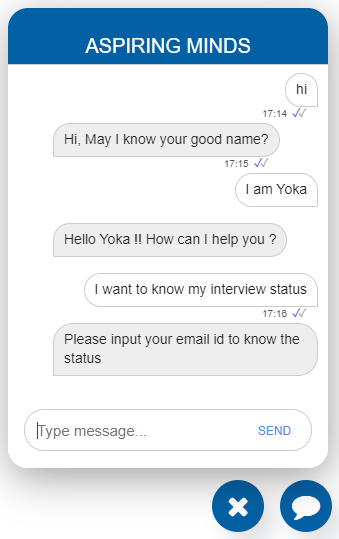
* **CB01 -** Initiate the conversation.



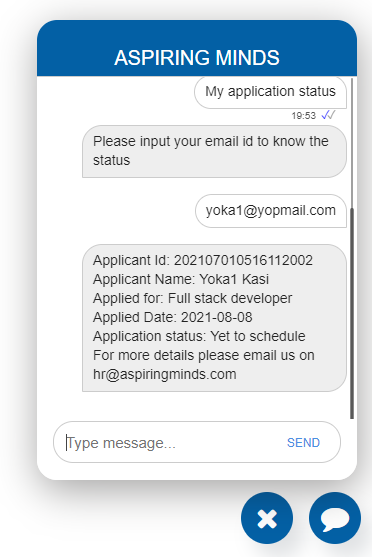
* **CB02 -** Next step is to provide your name.



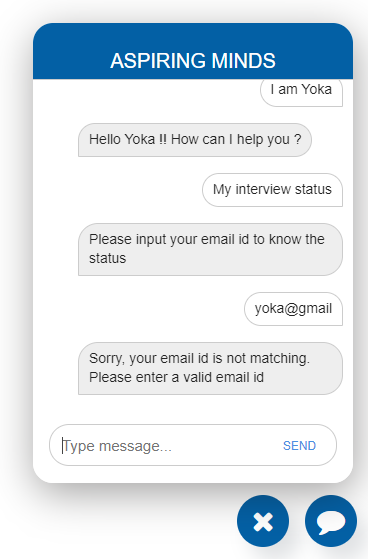
* **CB03 -** Here, we can ask the user queries like interview application status and feedback.



* **CB04 -** If we send a valid email id, then we will get the status of the applicant.



* **CB05 -** If we send an invalid email id, then we will get the error message.



1. Collection

To create sample data for applicants and job details collection.

* <http://localhost:8080/applicants>

{

    "applicantId": "202107010516112005",

    "firstName": "Applicant1",

    "lastName": "App",

    "email": "applicant1@yopmail.com",

    "phonenumber": "9597555836",

    "jobId": "1",

    "applicantStatus": "On Hold"

}

* <http://localhost:8080/jobdetails>

{

    "jobId": 2,

    "jobName": "Java Developer",

    "jobDescription": "Java 6 + years experience",

    "openPosition": 6

}