Design Document

Technologies Used

Frameworks**:**

- SpringBoot2
- SpringMongoDB
- Hystrix
- Junit
- Mockito

Infrastructure related:

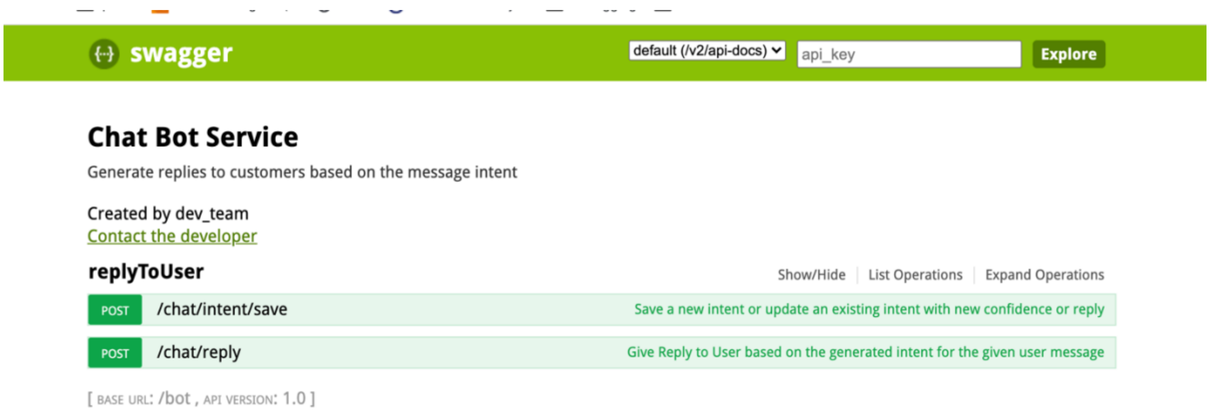
- Docker
- Maven

Programming Languages

- Java 11
- Bash

Swagger:

Swagger endpoint:
http://localhost:8080/bot/swagger-ui.html



Chat Bot Service

Generate replies to customers based on the message intent

Created by dev_team
Contact the developer

replyToUser

/chat/intent/save

Save a new intent or update an existing intent with new confidence or reply

/chat/reply

Give Reply to User based on the generated intent for the given user message

[BASE URL: /bot , API VERSION: 1.0]

Initial setup :

- As the spring boot application starts up, a json file with data corresponding to intents, replies and the confidence thresholds is uploaded to the mongo DB as a single collection. This is solely for the purpose of this assignment in order for the data to be available.

Endpoints supported:

1. Chat Reply Endpoint

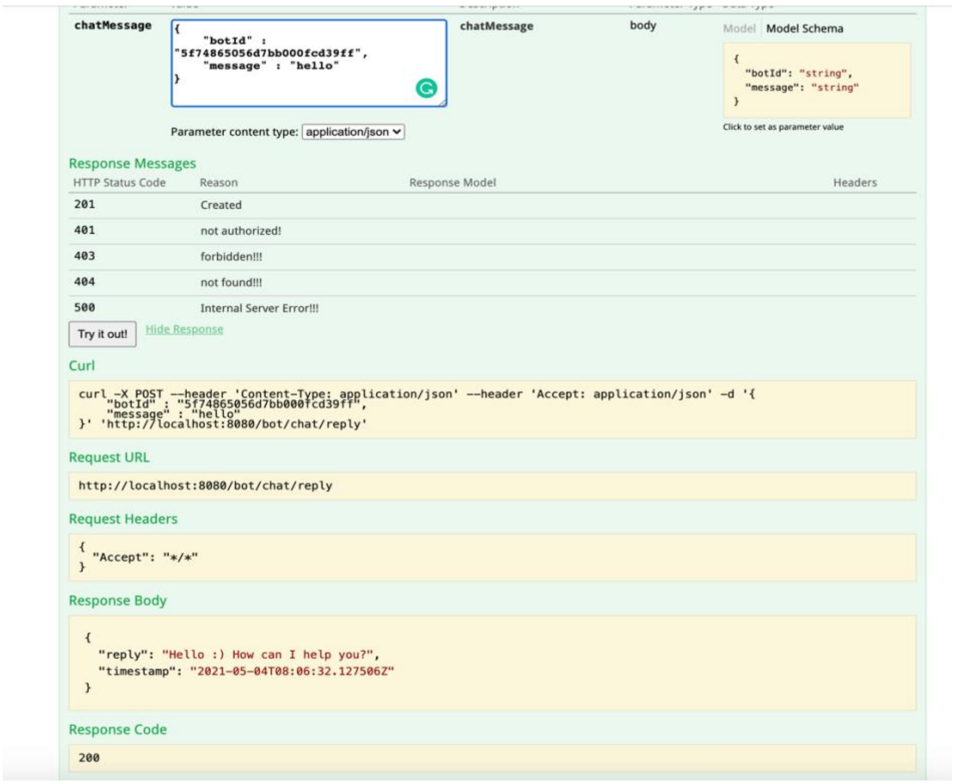
- This is a POST endpoint which will receive bot identifier and user's message as input payload.
- This input is then sent to an external AI endpoint to fetch a list of intents along with corresponding confidence levels that match the user's message.
- The intent with the highest confidence is selected and the corresponding reply is fetched from Mongo collection of intents.
- If the intent has lower confidence than what is stored in the mongo collection then a default reply is given to the user.

Edge cases handled:

- If the external AI API that is used to fetch a list of intents and confidences takes a very long time (more than 20 seconds) then a hystrix circuit breaker will break the request and send a defaulted reply to the user.
- If the input payload has an incorrect bot identifier or if it is empty then 400 client error is thrown from the endpoint.
- All possible exceptions are wrapped into user friendly exceptions with appropriate messages.

Further improvements that can be made to the service:

- Application monitoring can be introduced to monitor the response time of all the APIs and the performance.
- Caching can be introduced to avoid repetitive get calls to the AI intent API to fetch the intents for the same messages over a certain period of time. Hence, Redis cache with appropriate TTL value for each key can be used.
- Currently logging is done in a very simple fashion and the total response time taken for each endpoint has been captured. This can be further enhanced by adding more information.
- There is one more json file called as training data which has all the user typed messages that are mapped to an intent as per the AI API response. This data can be imported as a separate training collection into mongo. So, when the above endpoint is used to giver user reply for a message then the user's message can be **asynchronously** stored in this collection.
- This training data can be used for feedback purposes that will help improve the AI performance and also for reporting to analyze and fix issues in the system.



chatMessage	{ "botId" : "5f74865056d7bb000fcd39ff", "message" : "hello" }	chatMessage	body	Model | Model Schema: { "botId": "string", "message": "string" }

Parameter content type: application/json

Click to set as parameter value

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	not authorized!		
403	forbidden!!!		
404	not found!!!		
500	Internal Server Error!!!		

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{
    "botId" : "5f74865056d7bb000fcd39ff",
    "message" : "hello"
}' 'http://localhost:8080/bot/chat/reply'
```

Request URL

```
http://localhost:8080/bot/chat/reply
```

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body

```
{
  "reply": "Hello :) How can I help you?",
  "timestamp": "2021-05-04T08:06:32.127506Z"
}
```

Response Code

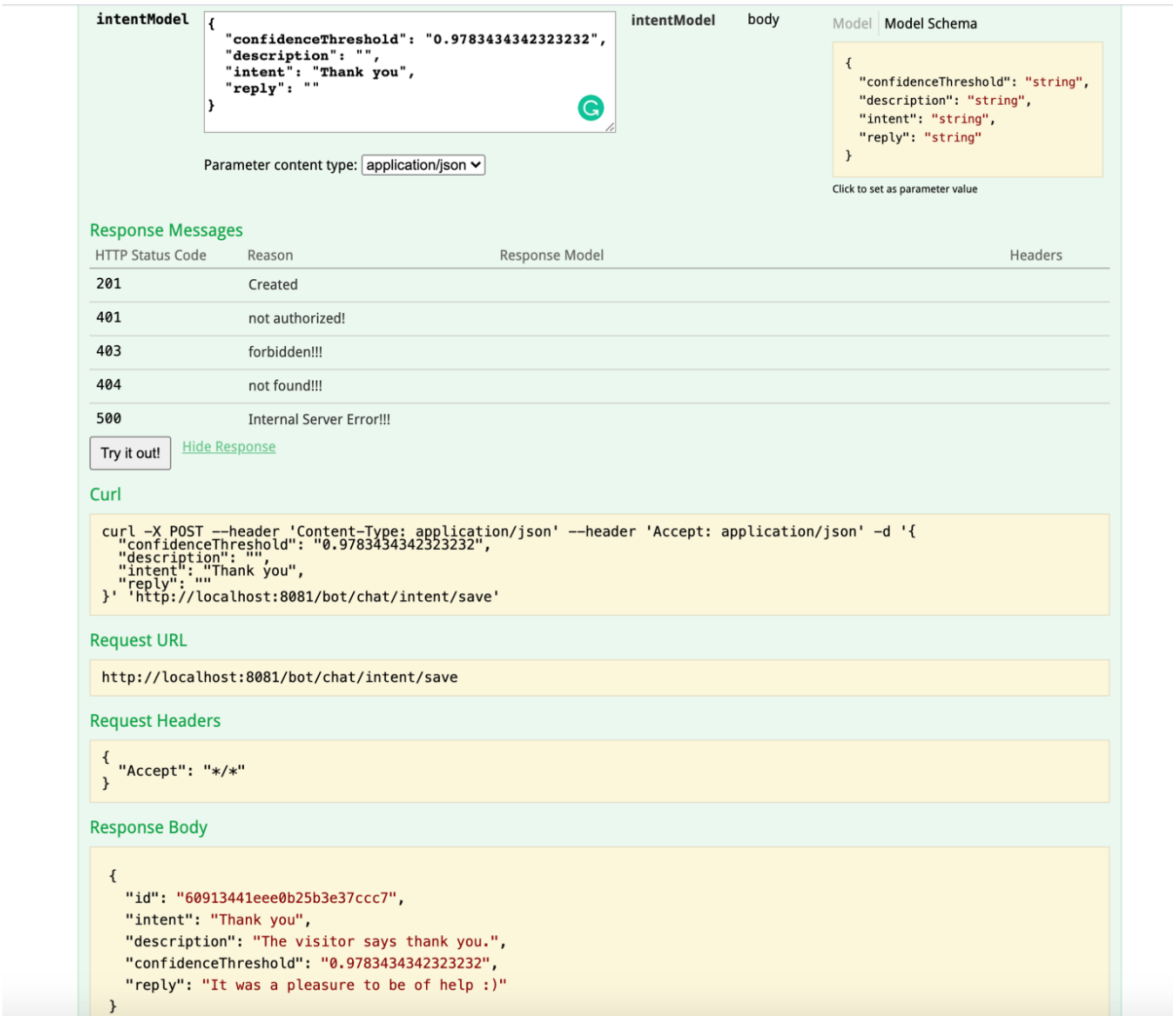
```
200
```

Response model for above endpoint:

- The response object ChatReply contains the chat reply in text and the timestamp.
- The chat reply will be default if no relevant intent is matched as per the confidence threshold.
- The timestamp is in human readable format, it sorts correctly and includes fractional seconds, which can help re-establish chronology. Most importantly it conforms to ISO 8601, which has been well-established internationally for more than a decade

2. Intent Save Endpoint

- This endpoint can be used to save a new intent into the mongo collection or update an existing intent with its confidence threshold or reply.
- This will be useful only when we need to alter the confidence threshold based on the analytics data.
- The confidenceThreshold value is accepted as string because mongo DB will internally store it as string even if we make the column as BigDecimal but I am using BigDecimal for interval validation.



intentModel

```
{
  "confidenceThreshold": "0.9783434342323232",
  "description": "",
  "intent": "Thank you",
  "reply": ""
}
```

intentModel body

Model Model Schema

```
{
  "confidenceThreshold": "string",
  "description": "string",
  "intent": "string",
  "reply": "string"
}
```

Click to set as parameter value

Parameter content type: application/json

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	not authorized!		
403	forbidden!!!		
404	not found!!!		
500	Internal Server Error!!!		

Try it out! Hide Response

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{
  "confidenceThreshold": "0.9783434342323232",
  "description": "",
  "intent": "Thank you",
  "reply": ""
}' 'http://localhost:8081/bot/chat/intent/save'
```

Request URL

```
http://localhost:8081/bot/chat/intent/save
```

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body

```
{
  "id": "60913441eee0b25b3e37ccc7",
  "intent": "Thank you",
  "description": "The visitor says thank you.",
  "confidenceThreshold": "0.9783434342323232",
  "reply": "It was a pleasure to be of help :)"
}
```

Response model for above endpoint:

- The response object IntentReply contains the entire updated object which was saved as a document in the mongo collection.
- In the above example only confidence threshold was updated for the “Greeting” intent.
- If confidence threshold is missing or not in range (0-100) or not a number then we get BAD request error along with the relevant message.
- Additionally, if intent reply or confidence threshold is missing or we try to save a new intent without these values then we get a BAD request with invalid input message.

Steps to run the application:

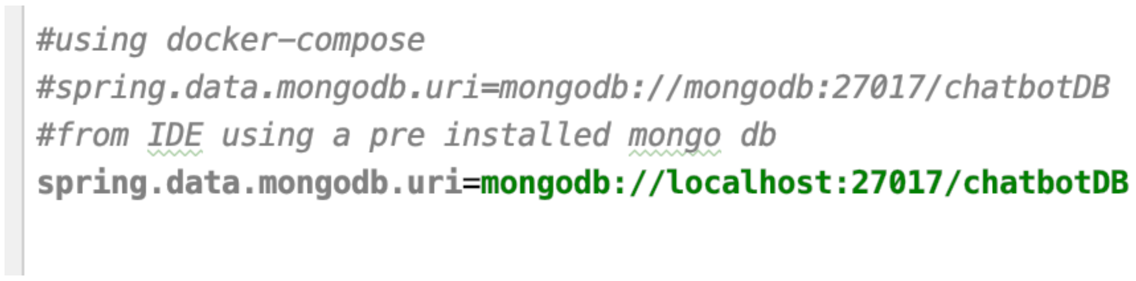
Note: Maven and Java 11 are the minimum requirements to run this Spring-boot application

There are two ways to run the application:

1. Using docker-compose that will create two containers linked with each other. One for the web server and the other one for mongo DB.
   After unzipping, Navigate inside the project directory and run the bash script file in the base directory.
   $ sh start.sh
   This will run all the tests, create a docker image of the project and create a container using this image linked with a separate mongo container.

2. Run it from an IDE like eclipse or IntelliJ once the project has been imported and built using maven. Also, this will require a mongo DB installed on the host machine which can be used by the application to connect to.
   Please comment out the mongo DB URL line for docker-compose in the application.properties as below and uncomment the line to use the URL with localhost/hostname where mongo db is installed.

spring.data.mongodb.uri=mongodb://localhost:27017/chatbotDB



```
#using docker-compose
#spring.data.mongodb.uri=mongodb://mongodb:27017/chatbotDB
#from IDE using a pre installed mongo db
spring.data.mongodb.uri=mongodb://localhost:27017/chatbotDB
```