

CHATBOT

Major project report submitted in partial fulfillment of the requirement

for the degree of Bachelor of Technology

in

Information Technology

By

ARPIT SOOD (191513)

UNDER THE SUPERVISION OF

Dr . NISHANT SHARMA



Department of Computer Science & Engineering and Information

Technology

Jaypee University Of Information Technology, Wakhnaghat,

173234, Himachal Pradesh, IND

Candidate's Declaration

I hereby declare that the work presented in this report entitled "Android Chatbot " in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering/Information Technology submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from Feb 2023 to May 2023 under the supervision of Dr. Nishant Sharma Assistant Professor(Grade-2)Department of Computer Science And Engineering . I also authenticate that I have carried out the above mentioned project work under the proficiency stream Data Science.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

ARPIT SOOD

Arpit Sood(191513)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)

Dr. Nishant Sharma

Assistant Professor(Grade-2)

Department of Computer Science And Engineering

Dated:

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date:

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: _____ Department: _____ Enrolment No _____

Contact No. _____ E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> • All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String 		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by

Name & Signature

.....

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully.

I really grateful and wish my profound my indebtedness to Supervisor Dr NISHANT SHARMA. Assistant Professor (Grade-2), Department of CSE Jaypee University of Information Technology. Wagnaghat. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to Dr NISHANT SHARMA, Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straight forwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Arpit Sood

TABLE OF CONTENT

Sr. No	Title	Page No
1	Candidate's Declaration	I
2	Plagiarism Certificate	II
3	Acknowledgement	III
4	Table of Content	IV
6	List of Figures	V
7	List of Tables	VI
8	Abstract	VII
9	Chapter-1 Introduction	1-12
10	Chapter-2 Literature Survey	13-16
11	Chapter-3 System Development	17-47
12	Chapter-4 Performance Analysis	48-62
13	Chapter-5 Conclusions	63-64
14	References	65

List of Figures

Fig No.	Title	Page No.
1.1	Incremental Model	6
1.2.	Gantt chart	12
2.1	System texture	15
3.1	Architectural Design	17
3.2	Architectural Design	18
3.3	Architectural Design	18
3.4	intent invoked	20
3.5	Database Model	21
3.6	UML class diagram	22
3.7	UML activity diagram	24
3.8	UML activity Diagram	25
3.9	UML activity diagram	25
3.10	UML activity diagram	26
3.11	Sequence Diagram	27
3.12	Sequence Diagram	28
3.13	Sequence Diagram	28
3.14	Banking Dialog	32
3.15	Web base Chatbot	33
3.16	Web base chatbot	33
3.17	Scanner for app	34
3.18	Banking bot	34
3.19	API model	43
3.20	API model	43
4.1	User chatbot	60
4.2	Conversion flow	61

4.3	Conversion flow	62
4.4	Chatbot Conversion Context	62

List of Tables

Table No.	Title	Page No.
3.1	CUI Design Principles	30
3.2	CUI Design Patterns	31
4.1	Simulated User Phrases: Test Results	50
4.2	Simulated User Phrases: Test Results	50
4.3	Simulated Device Testing	53-55
4.4	Overall accuracy comparison of Devices	55
4.5	Conversation exit rate	57
4.6	Subjective & objective measurement	59

Abstract

Today the users face a lot of problem regarding booking of the hotels in any android application because in most of the cases the user gets result more than what he expected or he gets results which are not according to his convenience .This paper focuses on automating the process of communication by use of chat-bot and it also focuses on providing customized results to the user which makes the process of hotel booking convenient and user friendly for him. An extensive research done on existing systems gave us an insight into their shortcomings which this system attempts to overcome by creating a chat-bot using Artificial Intelligence Markup Language and using various algorithms such as Keyword Matching , String Similarity , Spell Checker and Natural language parser. The implementation of this system has resulted in better resource utilization and increased responsiveness of user behaviour. This system has been implemented to integrate with any Hotel Management Android Application to ease the process of hotel booking.

A chatbot is a conversational agent where a computer program is designed to simulate an intelligent conversation. It can take user input in many formats like text, voice, sentiments, etc. For this purpose, many open-source platforms are available. Artificial Intelligence Markup Language (AIML) is derived from Extensible Markup Language (XML) which is used to build up a conversational agent (chatbot) artificially. In this paper, we use „programo“ which is an AIML interpreter for the generation of the responses of users' input. We have used this method for developing an Android application chatbot that will interact with users using text and voice responses.

CHAPTER 01

1.1 INTRODUCTION

Once it has a thorough awareness of the user's needs, an AI bot built on Android can engage with the user. As a result, the AI in the bot will act like a human and respond to the user's questions rather than one human on one side and another human on the other. AI Bot makes use of "neural networks," which are sizable machine networks that resemble the network of neurons in the human brain. It is a piece of software that use the Internet to carry out automatic operations. A chatbot typically works more faster than a human alone can to complete activities that are straightforward and fundamentally repetitive. An interactive agent, or "bot," is a computer programme created to mimic an intelligent speech-based communication with one or more human users.

Chatbots have become increasingly important as a human-computer interface in recent years. An interpreter, a knowledge base, and a user interface are the three main parts that make up a chatbot. According to Laven [6], a chatbot is a computer programme that tries to mimic a written dialogue with the intention of briefly fooling a human into thinking they are speaking to someone else. A chatbot is essentially a conversational agent that can have a natural conversation with the user about a particular topic. Online chatbots have been widely used for customer support, education, counselling, and amusement. Famous chatbots now in use include ALICE [2], Sim Simi, and Clever Bot. Extensible markup language (XML) is the ancestor of Artificial Intelligence Markup Language (AIML). which is applied to synthesis a conversational agent. AIML-based chatbots are well-known for being inexpensive, lightweight, and simple to set up. The AIML object class of data objects is used to explain how computer programmes behave. Program-o [1], an open source PHP-based AIML engine, was used in our article. It is an AIML chatbot script interpreter. Information about the chatbot is kept in a MySQL database. We also store all AIML scripts in a

database. When a user submits a message to a chatbot programme, the chatbot programme formulates and sends the user a response in accordance with the AIML response that corresponds to their message. Under the terms of the GNU General Public Licence, it may be immediately installed on a local server. Internet-based chatbots can be programmed to respond to text, speech, and emotional cues. We took user input for this post in the form of text and voice. Because the user may check the input to see if there are any problems, text input and output is comparatively efficient. Text entry, however, takes time. The introduction of a voice interface with speech recognition technology is the answer. These techniques greatly enhance the chatbot application's ability to communicate with the user.

The way people engage with one another and with enterprises has been completely transformed by digitization and the advent of mobile and internet-connected devices (Eeuwen, M.V., 2017). As technology companies integrate artificial intelligence (AI) into the products they sell, such as Google Assistant, Google Home, and Amazon Alexa, millennials are adopting and supporting new technologies into the routine of their everyday lives. Businesses anticipate that the current and forthcoming generations will be crucial, game-changing clients. They demand smooth interactions, responses in a matter of seconds rather than minutes, and more intelligent self-service alternatives. (2017) (Teller Vision). Among the first businesses to use the technology was the banking and financial services sector. This integration has significantly expanded and assists banks in connecting with a larger consumer base, allowing.

1.2 Problem Statement :

Technology called an AI chatbot makes it possible for people and machines to communicate naturally. We discovered from the literature that chatbots typically operate like a standard search engine. The essential process flow remains the same even though the chatbot only provided one output rather than several outputs or results: a new search will be run whenever an input is entered. Nothing to do with the earlier output. The goal of this project is

to develop a chatbot that can process subsequent searches in light of earlier ones. This feature will enhance the chatbot's capacity to process input in the context of a chatbot. Approach: We redesigned the chatbot using a relational database model strategy in an effort to expand the conventional chatbot process mechanism.

Users today have numerous difficulties while attempting to book hotels through any android app since, in the majority of situations, they receive results that are either unsatisfactory or exceed their expectations. In order to make the customer's hotel booking experience convenient and user-friendly, this document focuses on automating the communication process using a chat-bot and also emphasises giving the user with personalised results. In-depth analysis of existing systems revealed their flaws, which this system aims to fix by building a chatbot using an AI markup language and a variety of algorithms like keyword matching, string similarity, spell checking, and a natural language parser. The adoption of this technology improved resource management and increased responsiveness.

1.3 Objectives :

- A chatbot's primary objective is to comprehend the user's needs and provide the necessary information in response.
- Chatbots enable companies to interact personally with customers without having to hire human personnel.
- They serve as online assistants who aid with curriculum updates, paper grading, student and graduate data retrieval, and admissions process coordination.
- The main objective of the "College Inquiry Chatbot" is to keep users fully informed about ongoing and upcoming college events, reduce the workload of the school office staff, save the user's time and energy, and minimise the time needed to resolve the user's inquiries.

- Understanding the user's needs and responding with the appropriate information is a chatbot's main goal.
- Chatbots allow businesses to engage in personal client interaction without needing to hire human staff.
- They work as online assistants who support curriculum changes, paper grading, student and graduate data retrieval, and management of the admissions process.

1.4 Methodology :

To guarantee that a realistic time period is created for each step of the project and that requirements are properly specified, choosing a suitable methodology is crucial to the overall development of any software programme. The creation and design of this programme will take into account a variety of development approaches. The development methodology that is most suited for this project will be highlighted in this section.

1.4.1 Waterfall Methodology :

When you initially learn about software development, you are typically introduced to this pretty traditional methodology. The requirements collecting, analysis, design, implementation, and testing phases make up the five phases of the waterfall model, a highly predictable method for developing software. The stages are finished one after another. The fundamental drawback of the waterfall approach is that because the project is broken up into phases, it is quite rigid. To adhere to the overall project schedule, each phase has a deadline to provide an output. Project deliverables, design documentation, and test plans are used to gauge the success and progress of a project. It is challenging since each project phase is described early in the project life cycle and goals have been specified.

1.4.2 Incremental Model :

The waterfall concept gave rise to this software methodology. Iterative incremental build phases are used to design, create, and test the application. Each build will result in the creation of a subsystem or element. As new requirements are likely to be identified and included in each incremental release, expanding on the functionality from the previous build, the project will become more complex overall. Software is frequently delivered in phases, thus it's crucial to manage the versions of the software's components throughout its lifecycle using version control technologies like GitHub. It will only take a few weeks for each build to complete the app's foundation. It is possible to offer criticism.

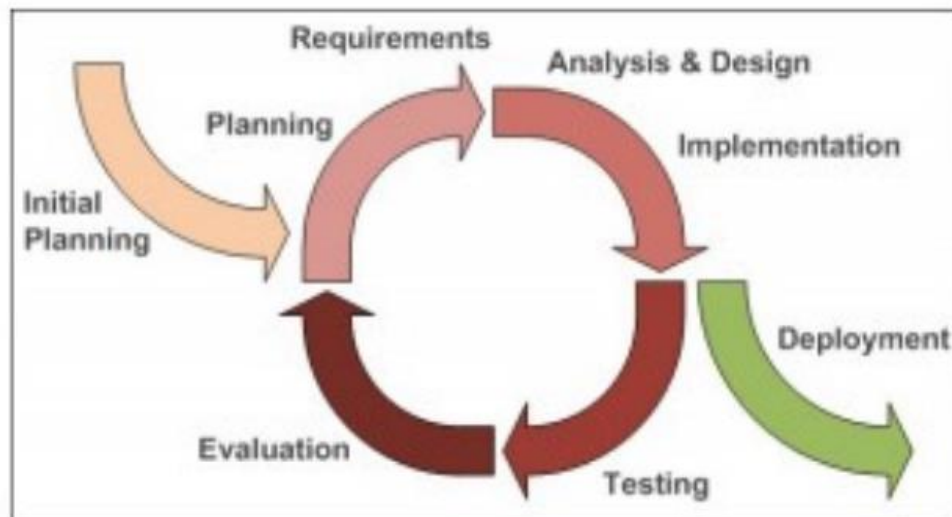


Figure 1.1 Incre model

Bugs are found early in short, controllable cycles, making testing and troubleshooting simpler. As new requirements are easily incorporated into each build and an updated version is produced, this process is adaptable during implementation (Khan et al., 2011).

1.4.3 Selected methodology:

The best development methodology for this project is the incremental model. Since additional requirements are likely to be discovered at later

stages of development and since each iterative build makes it simpler to implement new requirements throughout the development process, the incremental model's flexibility makes it the best choice for this project.

1.4.4 System Need:

Manually communicating with potential users is a labor-intensive process. The chatbot and user's one-on-one communication is highly prized. However, controlling talks between two persons is typically not possible due to the tens of thousands of applications that are made each year. It would take a lot of effort for an Android application administrator to track down suitable solutions and inform each user. Reducing this time-consuming process is beneficial. By creating an engaging chatbot, the initiative hopes to lessen the load on the user initiating the conversation and perhaps other users as well. Both the user and the bot can communicate via a chat interface.

1.4.5 Existing system:

ELIZA was the first chatbot to be created. Joseph Weizenbaum created it using the keyword matching method. The concept was to read the user's input, search for particular patterns, and load the response if the pattern was discovered. If the keyword is not identified, ELIZA attempts to elicit further information from the user in accordance with the established rules in order to maintain contact with them.

Artificial Intelligence Mark-up Language (AIML) files are used by ALICE to store data and employ pattern matching. Like ELIZA, it is a chatbot whose primary use is to converse with the user. It is mostly employed for sporadic user interactions.

- Natasha is an Android hiking companion who responds to your

1.4.6 Problems in the existing system:

- ELIZA is speaking in a language she does not comprehend. It only offers outcomes that follow pre-established rules.
- ALICE can only get information from its database because it lacks the ability to learn.

Natasha is solely used to speak with users and provide them with information that is accessible online. The objective of this system is not to make hotel reservations.

Users have trouble switching between transactions in popular hotel booking software. In order to complete the transaction, you must navigate through numerous screens.

1.4.7 The proposed system:

This system interacts with users using a chat application that offers insightful comments and guidance for obtaining the data required to reserve hotel rooms. The user receives precise output from the system, even in the case of small spelling mistakes. Additionally, parsing prevents transmitting words to the system that don't fit into patterns. Frequent travellers can maximise the system's potential.

1.4.7.1 GPS tracking:

The Global Positioning System is used by GPS tracking to pinpoint and follow the precise location of a user's device. The application begins by following the user's current position, which is then saved in the database.

When reserving a hotel room in the same city or when the user forgets to specify where they want the accommodation, this location is helpful.

1.4.7.2 Calculation of AI:

A chatbot that is programmed in AIML does AI computation. The first step is for the user to start chatting and provide input. The chatbot is now determining whether the user's input is unsuitable, inadequate, complete, or conversational. The user is informed that their contribution was unsuitable if it is inappropriate. The user is prompted to enter the missing parameters if the input is insufficient. If the input is conversational, the chatbot engages the user in a casual conversation. The user receives the exact output if the input is complete.

1.4.7.3 Hotel reservation:

Users receive personalised recommendations before beginning a hotel reservation. The user then selects a hotel from the list provided above. The user is now given a payment link, and when they click it, they are taken to the payment gateway page. The consumer then pays for the hotel of their choice. The chat interface is now used to notify the user of the details of their booking.

1.4.8 Functional requirements:

- Enable users who are not logged in to register and store their data to the database.
- Send SMS and email confirmation notifications.
- Once the user's credentials have been uploaded to the database, a QR code using Google's two-factor authentication will be displayed to them, and a special code will be generated and delivered to the user's mobile device.
- Users must be able to examine information about the accounts they have access to, such as savings, loans, and checking accounts.

The chat bot will send users a transaction statement via email so they may view their transactions.

- The chatbot's integration of the TrueLayer Starling API returns data on the user's bank account.
- Users will be able to communicate with the chatbot via voice or text instructions, and it will be able to understand what they are saying thanks to Dialogflow API integration's natural language understanding processing.
- When the context from earlier messages and chats may be unclear, a chatbot should be able to retain a conversational state.
- Offer text- and audio-based answers.

1.4.9 Non-functional requirements:

- The chatbot must be quick to answer, taking no more than 5 seconds to do so in response to a user's message, for example.
- The database needs to be extensible to handle the increasing number of users. • The chatbot needs to be dependable and nearly bug- or error-free.
- As sensitive data is used, the chatbot needs to be secure; Google two-factor authentication will be added as an additional security measure.
- Adhere to data protection regulations as the Privacy Act 1198
- The chatbot may be communicated with using natural language, which promotes human-computer engagement.
- Give accurate comments and responses for the EEE521 Final Year Project 2017–2018 B00659303 19 input. • If feasible, handle unexpected inputs effectively and alert the user in the right way.

.

1.4.10 Software and Hardware Specifications:

The following is a list of the hardware and software requirements for implementing a chatbot.

1.4.11 Software specifications:

- Tramp
- Laravel Homestead
- Nginx server
- MySQL DB 11
- PHP 7.1
- JavaScript
- Oracle VM VirtualBox
- Dialogue flow
- Laravel 5.6
- Sublime Text 2
- Ngrok

.

1.4.12 Hardware Requirements:

A Linux/Ubuntu PC that serves as a local chatbot server Android devices;
Intel Core i5 processor

1.4.13 Project planning:

A collection of project milestones and deliverables is developed to show how the project is broken down so that a management plan can be created and followed throughout the project life cycle. Minor adjustments could happen in each iteration, though, because some requirements might need to be modified. To ensure that all project deliverables are met, the steps necessary in each iteration are specified. Make a risk management strategy and project schedule with project tasks for each semester displayed on a Gantt chart.

.

1.4.14 Risk management:

A critical step in minimising the influence of hazards on the overall success of the project is identifying and managing the risks and repercussions related to any project. Early risk identification enables the implementation of preventative measures to reduce the possibility of dangers arising. The following steps will be performed to lower risks in this project: periodically backing up the project report and critical documents; using Github to back up code and prevent loss; using your own initiative to keep track of the project timeline.

1.4.15 Project Management:

Figure 5.4.4.1, a Gantt chart prepared in MS Project, illustrates the project timeline and shows when milestones should be reached in order to deliver the project on schedule. The tasks' breakdown, the sequence in which they must be completed, and the time allotted for each are mentioned below.

CHAPTER 02

2.1 LITERATURE SURVEY

For the advantage of its consumers, the banking sector distributes technological assets and services through a variety of electronic distribution channels. According to Ajimon and G.G.S. Gireesh K. (George and Kumar, 2013), both the financial services and banking sectors consider online banking to be a business commodity. Many of our services have entered the digital era thanks to technological advancements, and banking is one of the main industries that is utilising these advancements to enhance its offerings. In the UK, there are now two paradigms for online banking. One of these is an integrated Internet bank with an online presence but that also uses branches to do business. The second independent Internet bank, which is totally autonomous and derives all of its revenue from the Internet alone

Banks use technology to increase their processing speed, attract more clients, and broaden the range of services they may provide (Consoli, 2005). Over the past ten years, there has been a huge increase in demand for internet banking. "15% of branch customers use online banking once a day, 59% once a week, 77% at least once a month, and 53% were confident that they could do the best part of their banking online" (Barty, J. and Recketts, T. BBA, 2014). Because consumers can manage their finances while on the road, online banking has grown in popularity as a way to meet the demands of modern living and eliminate the necessity for in-person branch visits. This is clear from the fact that branchless banks are now emerging .

The majority of internet banking service providers, according to a recent study by Ling et al. (2016), are attempting to attract many of their clients to use their services. They view the primary factor as a lack of user satisfaction with online banking services. Customer satisfaction is

primarily influenced by "service quality, web design and content, security, privacy, speed, and convenience" (Ling et al., 2016). This indicates that there isn't enough technology to raise client satisfaction. A chatbot might be integrated into the online banking experience to offer quick, convenient, and customised services.

The first chatbot to use a pattern matching algorithm is thought to be Eliza. Joseph Weizenbaum created it in 1964 [2]. A rule-based chatbot built on the AIML (Artificial Intelligence Markup Language) standard is called ALICE [3]. It comprises more than 40,000 categories, each of which combines a pattern with a corresponding reaction. A summary of chatbot applications created with AIML scripts was presented by Md. Shahriare Satu and Shamim-AI-Mamun [7]. They claimed that chatbots built using AIML are simple to use, portable, and effective. Their research offers comprehensive details on numerous chatbot applications. A chatbot based on AIML and LSA was created by Thomas N. T. and Amrita Vishwa [4] for customer support on e-commerce platforms. Their strategy demonstrates how we might enhance the chatbot's functionality.

2.2 SYSTEM ARCHITECTURE :

The following figure gives the information of our system architecture.

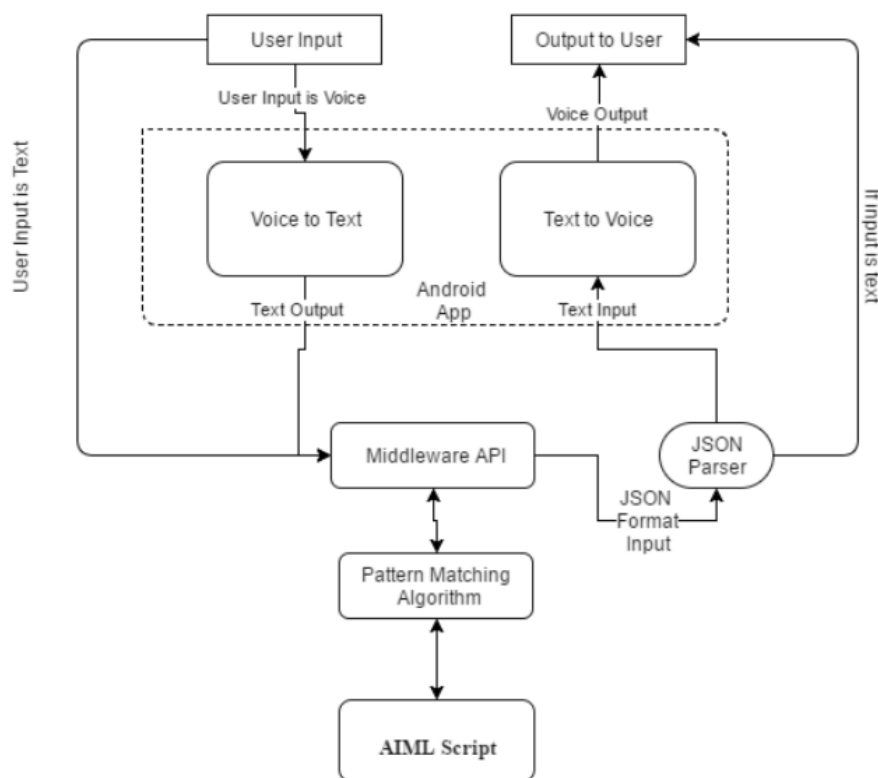


Figure 2.1 System Texture

The system operates in both text and voice modes. The first mode is turned on when the user enters data in text format. The middleware API receives user input and returns a response. The second mode, on the other hand, is engaged when the user speaks. To submit the speech data to the middleware API in this voice mode, we first transform the voice to text.

The mechanism known as middleware is what links AIML scripts to our Android application. User input is delivered to an algorithm for pattern matching through AIML scripts when it is received across middleware. A pattern matching method is used in this procedure to find a match between a legitimate response and one of the available AIML scripts. The middleware receives the matching template when the pattern matches. After that, the middleware converts the template to JSON format and replies to the Android app. The application decodes the JSON answer after receiving it and provides it to the user.

CHAPTER 03

SYSTEM DEVELOPMENT

Design

3.1 Architectural Design:

The proposed chatbot's abstract architectural design is shown in Figure in a simple and understandable manner.

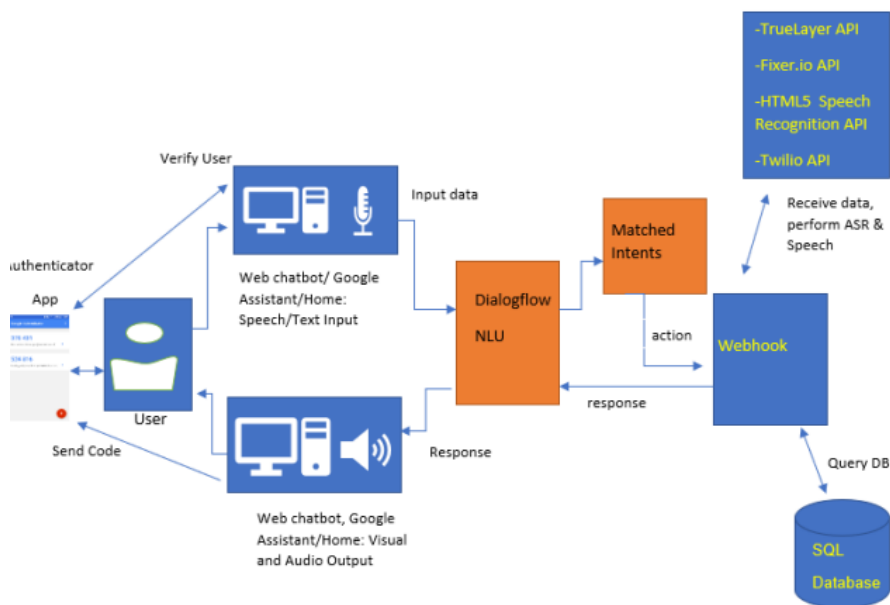


Figure 3.1

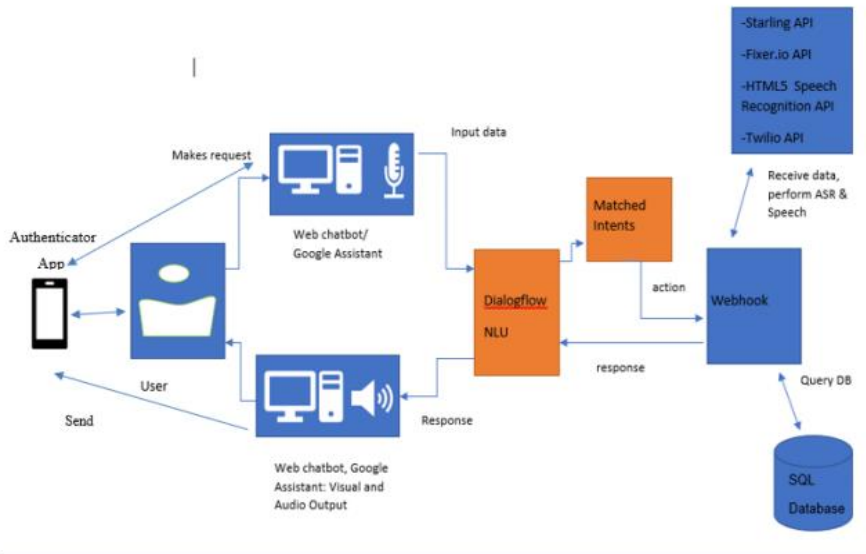


Figure 3.2

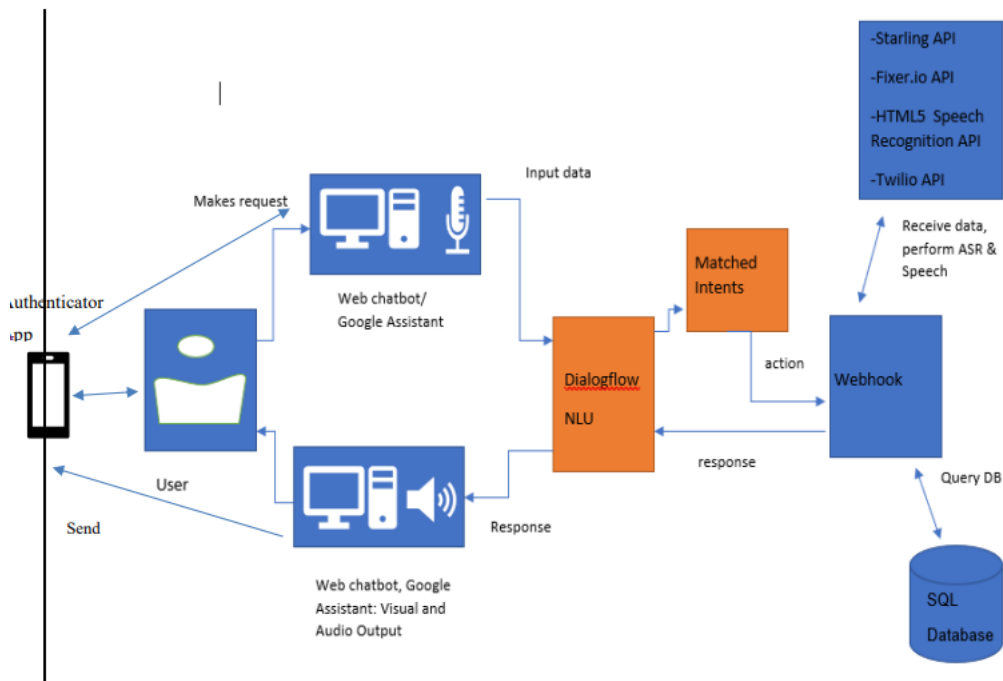


Figure 3.3

From the online client and the Google Assistant app for Android devices, users can communicate with the chatbot. Users will communicate with the chatbot using natural language text or voice phrases. Users will find it simpler to use and interact with the chatbot thanks to the Google Assistant

integration because there will be less typing and effort necessary when using voice requests and rich responses like graphics and cards. The chatbot may simply be expanded with Google Home thanks to the integration of Google Assistant. The client side of the application is created with Laravel Blade, HTML5, CSS, and JavaScript. The Laravel framework has a templating engine that adheres to the MVC architectural design pattern. Blade for Laravel compiles and caches

Blade employs template inheritance, which enables views to dynamically modify and reuse UI components without changing the code of the view from which they inherit, as well as derive the fundamental layout of another view. For control structures that may be necessary in specific view files, such as the "@foreach" annotation, Blade additionally has its own annotations (Laravel, 2018). In order for users to readily identify where they are spending the majority of their money, the client side also makes use of the chart.js CDN (Content Delivery Service).

Additionally, a web client and Webhook that retrieve JSON data from NLU via HTTPS POST requests are implemented using the Laravel framework. As a controller class, Webhook is created, allowing the associated web route to where the Webhook receives the payload in real time from the NLU. When the NLU determines that a user has activated a client-side intent, Google Assistant device, or Home device, the webhook receives data from the NLU.

An API uses a RESTful design approach and asks data to be obtained, but a webhook accepts data given to it from another service whenever an update or event occurs (elastic.io, 2018).

The chatbot can be trained to recognise entities and intents in user speech using Dialogflow's natural language understanding (NLU) engine. Intent mapping on the Dialogflow console will be used in the design of Chabot to direct user utterances to a set of phrases.

The diagram shows the data flow as a user invokes an intent: By instructing Dialogflow to recognise a set of common user expressions, it may determine whether an intent has been activated. The fulfilment webhook then receives this data. A custom response is generated and returned to the user from the webhook either visually or verbally when the required entities and parameters are extracted along with the action. The webhook decodes a generated JSON response and sends it back to the user for text and voice interaction after parsing and validating the JSON from the received payload.

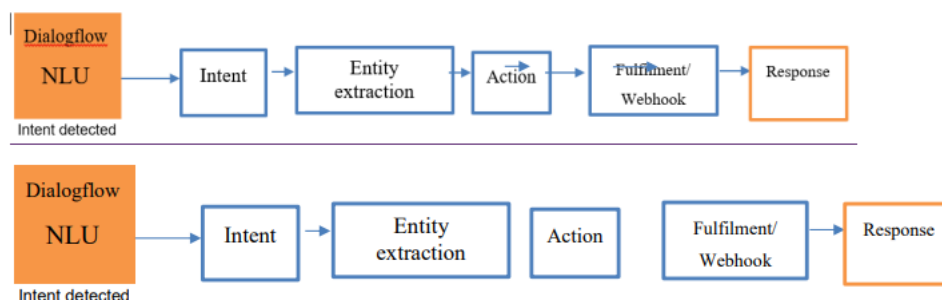


Figure 3.4 Intent Invoked

3.2 Database design:

Eloquent ORM, which "provides an Active Record implementation for working with database objects" (Laravel, 2018), is another component of Laravel. A small scale MySQL Workbench database is used to store the data locally in order to conserve memory. Each table in a database is represented by a "model" that enables the definition and manipulation of the relationships and logic between database items.

Figure 83.2.1 depicts the database model for the proposed chatbot and defines multiplicity. The users table contains details about application users, including whether or not they have Google 2 Factor Authentication enabled. The appointments table contains information on a user's appointments, including "topic". Contains details provided by the user

about the purpose of the meeting. Whether the user already has an appointment is indicated by the boolean value set for the 'booked' element. Data pertaining to a user's bank accounts are contained in the Accounts and Transactions table, for instance, a user may have several different types of accounts with a single bank, such as checking and savings

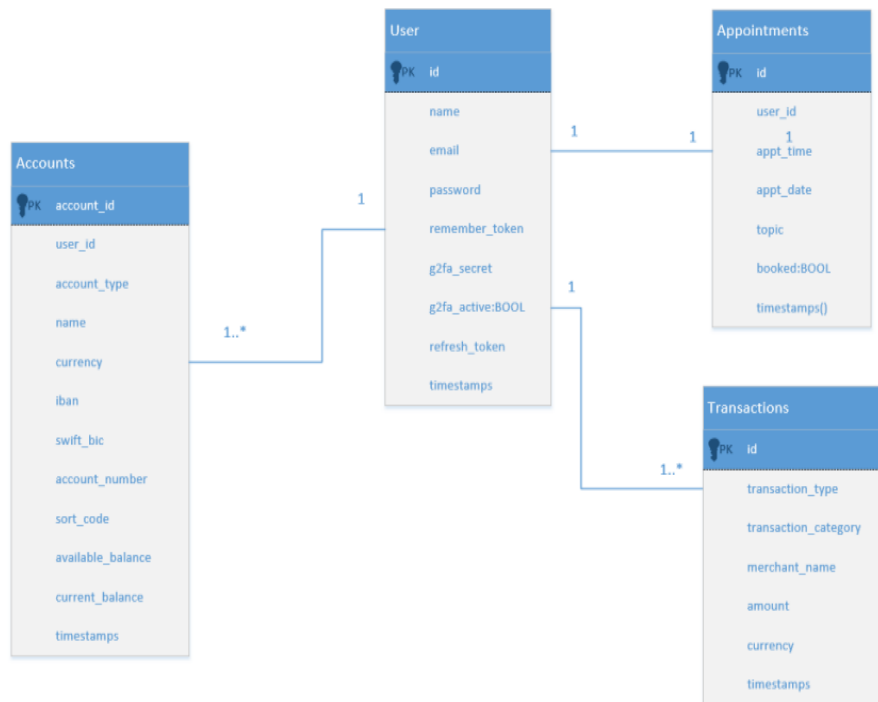


Figure 3.5 Database model

3.2 Class Diagram :

Classes required to create a chatbot are depicted in the class diagram in Figure 9.3.1. Although additional classes may be found or altered throughout development to satisfy the requirements, this will serve as a general framework for the chatbot implementation.

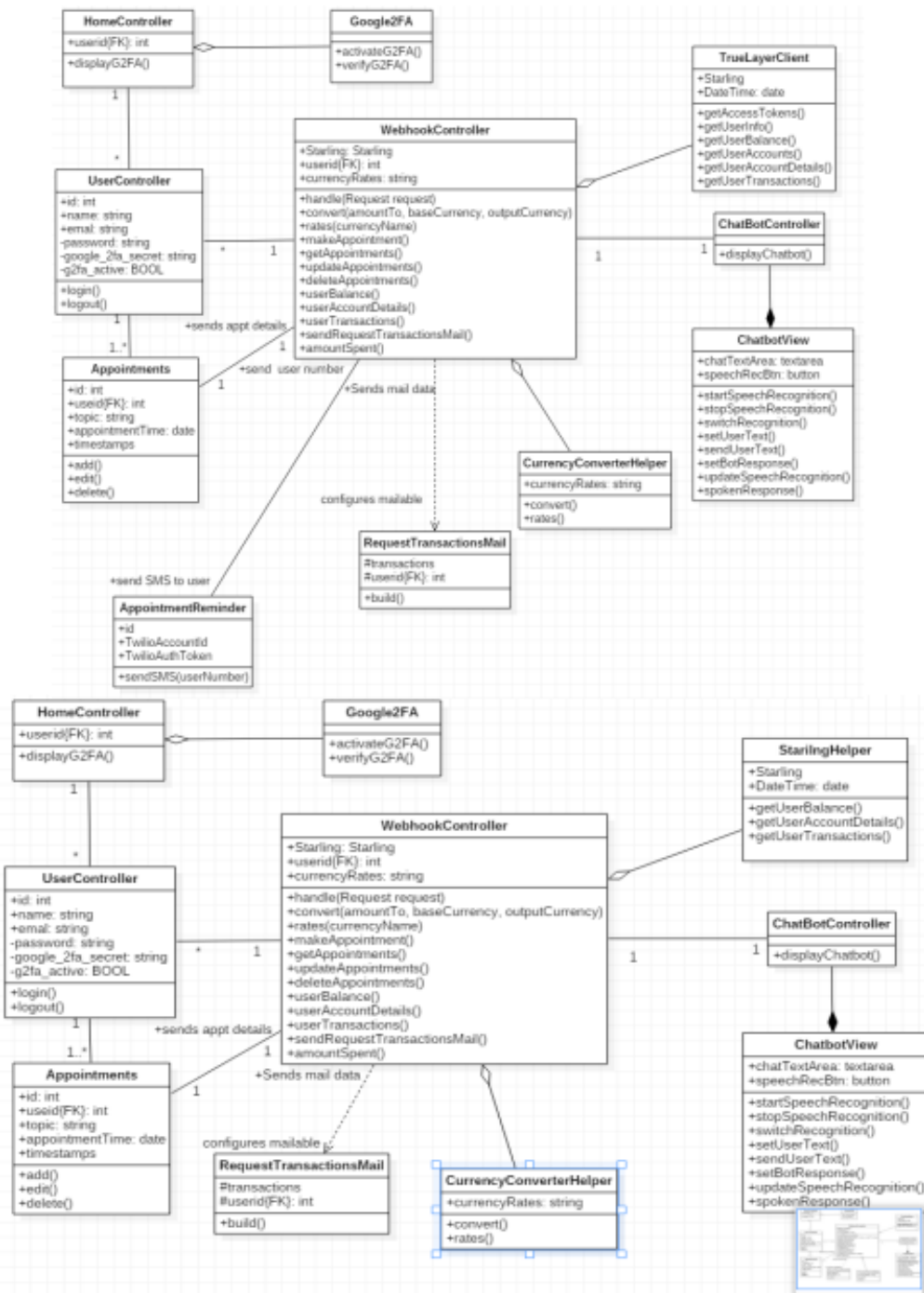
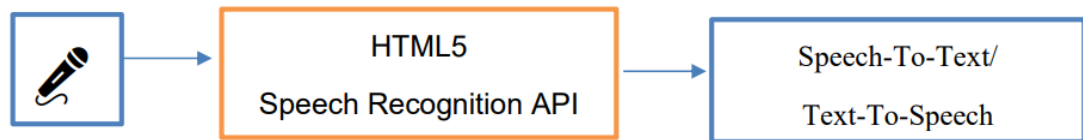


Figure 3.6 UML

3.3 Speech Recognition :

By speaking to the chatbot through the built-in microphone of the chatbot device, users can communicate with the chatbot using voice. Web

applications use the HTML5 Speech Recognition API for speech recognition and communication.



The output voice is then transformed by Dialogflow into JSON objects that can be read as input. A match score, sometimes referred to as the trust score, is given to the JSON answer. According to DialogFlow (2015), this rating demonstrates how successfully the NLU engine matches user input to the destination indicated in the console. 1 is a real match, and scores range from 0 to 1. The JSON response is described in more detail below.

"Who are you?" the user says.

```
{  "fulfilment": {
    "speech": "I am your smart banking bot, use me to get your account balance quickly, view transactions, debits, account details and receive currency conversions.",
    "messages": [
      {
        "type": 0,
        "speech": "I am your smart banking bot, use me to get your account balance quickly, view transactions, debits, account details and receive currency conversions."
      }
    ]
  },  "score": 1 - Exact Match
```

3.4 Activity Diagrams :

Activity diagrams are created to outline the processes involved in a chatbot and to specify the data flow. They depict how consumers behave while

dealing with chatbots and how the bots react to those behaviours through procedures. These will interfere with the chatbot's many features. The logic used by the chatbot when a user asks a currency conversion is shown in Figure 10.3.5.1. The user statement is sent to the Fulfilment webhook once the NLU analyses the user input and establishes the context. The NLU chatbot's JSON responses are accepted via the webhook, which employs POST endpoints. The accompanying intended action, which was predefined in the Dialogflow console during training, is then determined by the webhook and is mapped to the user-entered statement. If the behaviour

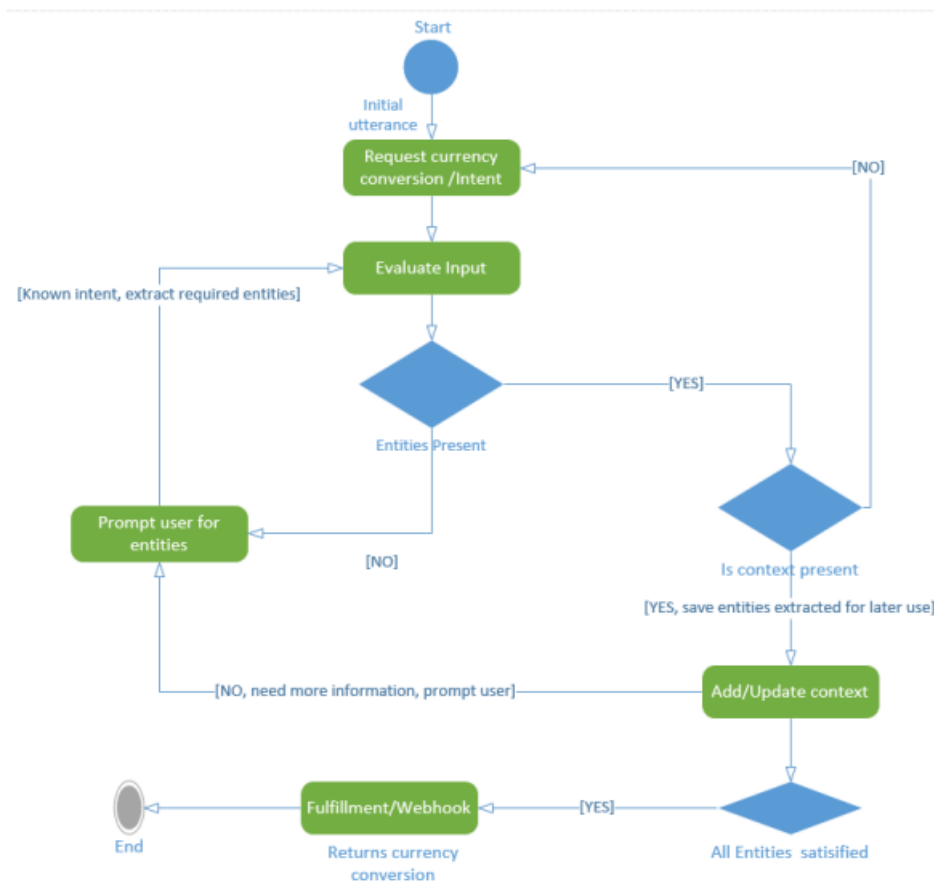


Figure 3.7 UML activity diagram

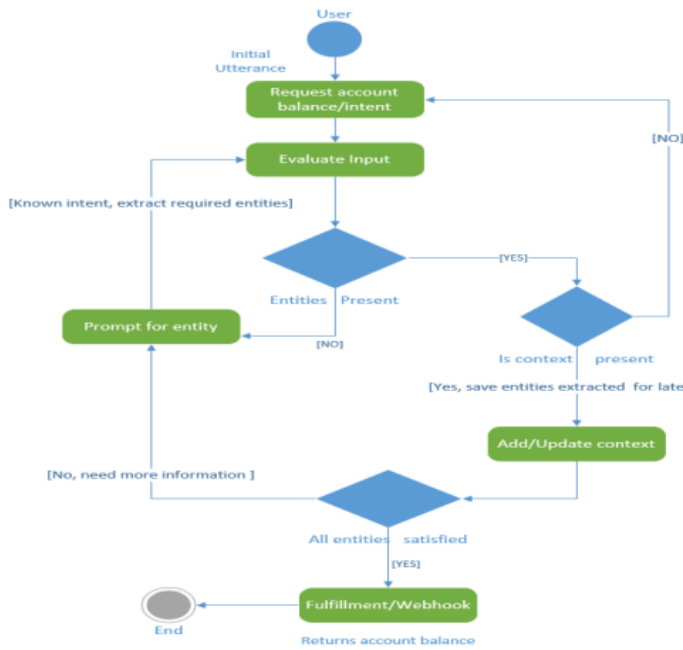


Figure 3.8 UML activity diagram

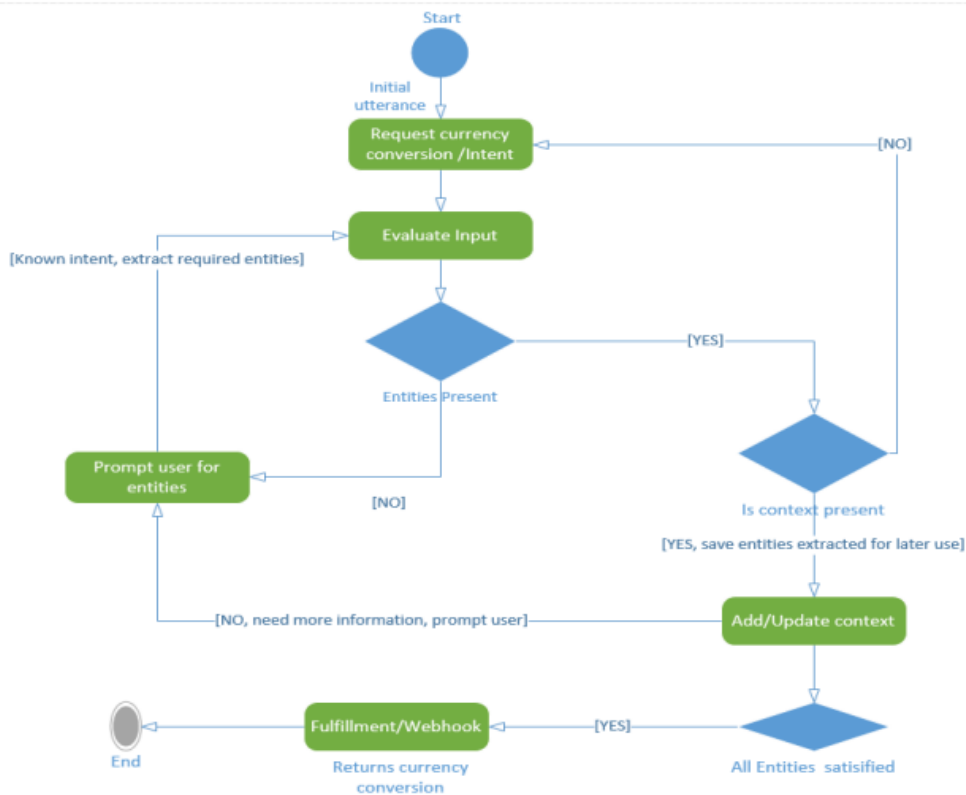


Figure 3.9 UML activity diagram

Figure 1 depicts the data flow when a user asks for their balance. NLU takes the user's feedback into account while presenting the intended product. The user's input is subsequently sent to the webhook in JSON format.

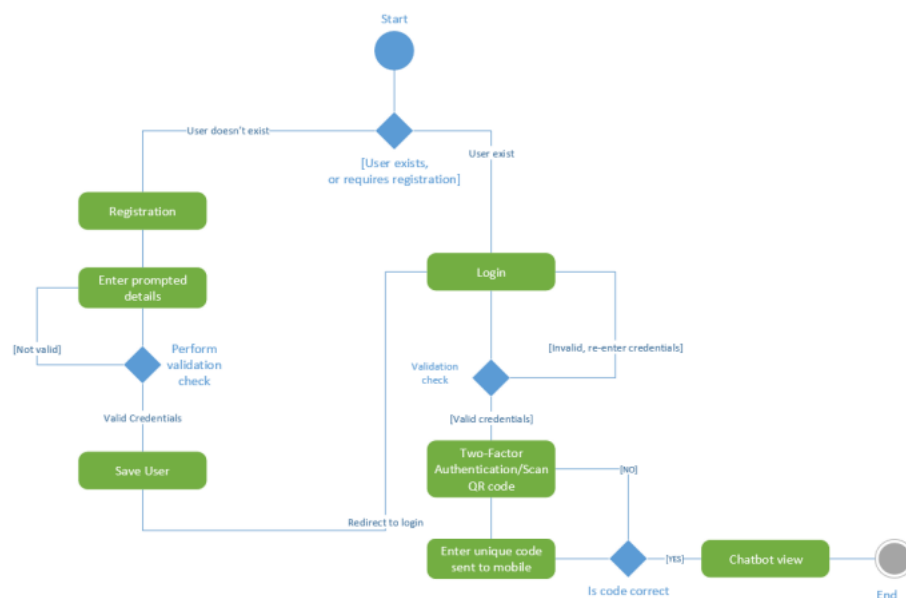


Figure 3.10 UML activity diagram

By removing the action from the intent that they receive and determining if it is contained in the intent, a webhook is utilised to offer users with appropriate bespoke fulfilment. Depending on the current action, it then calls the relevant function. The `getBalance()` function is invoked if the action is active. This method uses REST endpoints to access the True Layer API. The NLU chatbot is then delivered the response, which has been encoded in JSON format. An authentication package is included with the Laravel framework to handle application authentication. The procedure for logging in and registering is shown in Figure 123.5.3. Users must register in the application first in order to create an account. The user will not be permitted access to the site if they attempt to log in before registering.

3.5 Sequence diagrams:

The flowchart in Figure 13 3.6.1 displays a user's behaviour as they ask a chatbot for their banking information and describes the fundamental steps taken. The Chatbot NLU (Dialogflow) engine sends user input to the

webhook in JSON format. By interpreting the JSON it gets, the webhook decides what to do with the defined intent published from the NLU. The webhook will contact the appropriate method in the TrueLayerClientStarlingHelper class, which will call the True Layer Starling API and return data to the webhook to generate a response if the action fits the defined bank procedure. When the webhook receives the information returned by the API, it encodes the information into JSON, which appears as a response to the NLU.

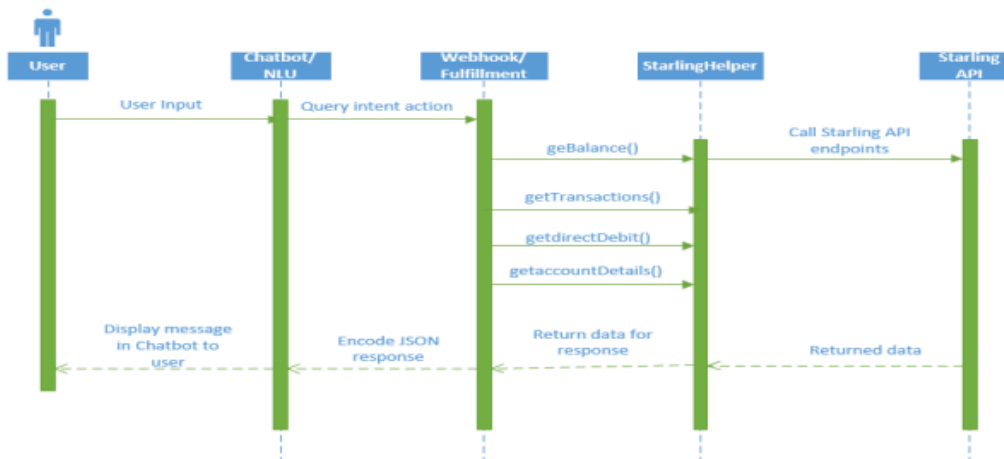


Figure 3.11

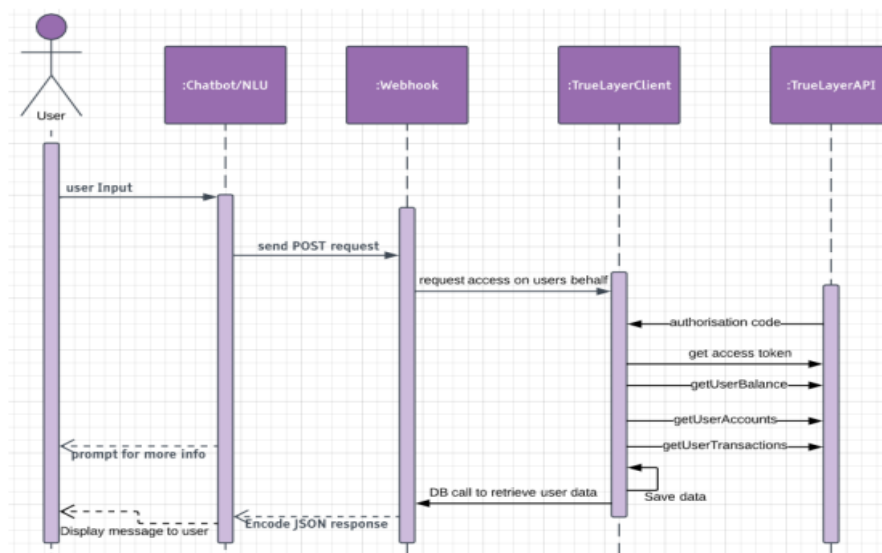


Figure 3.12

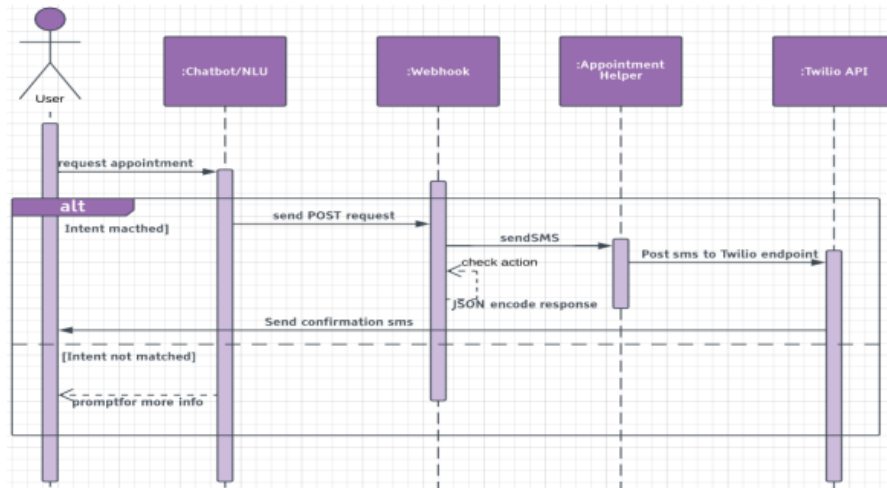


Figure 3.13

The audience that will utilise the chatbot must be carefully considered in order to implement it successfully. Because mobile banking is important to a wide range of age groups, the chatbot's target audience is made up of individuals from various age groups and technological backgrounds. The chatbot will provide simple, direct responses in order to appeal to all age groups. Developers can choose a voice for their bot that fits its personality and use rich answers with Google Assistant (Actions on Google, 2018). However, as the majority of users of a site like this currently engage online primarily through messaging services, using a platform like this should be simple (Interactions.acm.org, 2017).

3.5 Conversational User Interface Design:

The dialogue will take place inside the chatbot using natural language, and conversational user interfaces (CUIs) will be designed with this interaction in mind. According to Actions on Google (2018), "Conversation as a method of interaction is frequently referred to as the new user interface." 2018 (cs.hmc.edu) establish a set of design guidelines for conversational user interfaces.

PRINCIPLES		DESCRIPTION
1.	Design by personality	The bot personality should provide a natural and unique interaction experience for users. The personality should be designed to guide the users through the interaction which may include being aware of what the user may say or want. The personality is the style and manner the bot holds throughout the interaction.
2.	Flexibility in response	Flexible responses should be provided to user input, the bot should have various/random responses to the same user input.
3.	Text vs Custom Buttons	It should be clear to the user what input is expected to carry on in the conversation. A fallback response should be used in situations where the input is not recognised such as suggesting other operations that are available. Instead of solely relying on natural text commands from the user, a good UI should render structured options to the user in form of buttons or text.
4.	Simplicity in Interaction	Be concise and clear in response to questions. The conversation flow should be

		straight and not branch out to complex paths.
5.	Conversation Flow	The conversation flow is important to ensure that the user is not irritated in situations where the bot may not be able to provide an appropriate answer immediately. The user would then have to undergo a conversation in order to determine the answer to their query.
6.	Tasks and duty specifications	The bot should deliver explicit tasks for a particular domain.
7.	Rigid syntax, then NLP	People are prone to spelling mistakes or using colloquial phrases which may not be understood by the bot.
8.	Empathy and emotional state	It is important that the bot builds a rapport with the user and convey emotion in responses based on certain user input.
9.	Keep conversation short	Users want to interact with the bot to receive answers or reach a goal quickly. Avoid long winded conversations as it will make the interaction feel burdensome. This helps avoid ambiguity.
10.	Triggers and actions	The bot needs to persuade users and seek ways to encourage users to carry out specific actions through the bot.
11.	Predict and personalise	Bots can analyse previous input from users for important parameters, in order to anticipate a user's predisposition. It will then provide an answer that would be unique to that particular user.
12.	Fully/partially automated	NLP should be used for automated tasks.
13.	Providing a way out	Allow user to begin the conversation again or exit the conversation.
14.	User boredom	The conversation should engage the user throughout the interaction.

Table 3.1 CUI Design Principles

A precise list of chatbots, including those utilised in the financial sector, is recommended by the design process.

DESIGN PATTERNS		DESCRIPTION
1.	Follow-up questions	Follow-up questions should be specific in the context of the conversation. An example of this is when the user wishes to request a currency conversion. The chatbot would ask; "what is the amount and currency currency you wish to convert to?"
2.	Cognitive load	Chatbots should present coherent choices to users to reduce the amount of effort needed for the user to interact with it. This can be achieved through the use of buttons and images.

Table 3.2 CUI Design Patterns

3.5 Dialog Design

The design of the dialogue is an important factor to take into account when building a chatbot because the interaction is mostly between the user and the chatbot and occurs through natural language. To do this, the appropriate dialogue will be created using follow-up and fallback intents in the Dialogflow console. There will be a predetermined purpose that, when used by the chatbot user, will end the conversation in its entirety, independent of the circumstances. A dialogue box is a group of phrases and words that react to user input. There will be conversation.

User: "please book me an appointment"

be created with natural-sounding human utterances that respond to users. Plans for a fallback will be another design feature. The chatbot responds with a message indicating it does not match the intent input and asks the user to repeat the question or offer more information if the user statement cannot be matched to an intent. When communicating with a chatbot, dialogue is also created via follow-up intents, which promote a more natural flow of discourse. As an illustration, later intents gather more data from users as necessary; the user can state;

User: "Please Book me an appointment"

The chatbot utilises successive intents to ask the user for further information, such as the date, time, and phone number because it is aware that it needs this information to function properly. As this chatbot is a task-based interactive chatbot that includes banking domain knowledge, Dialogflow recommends designing a linear dialogue for interactions where the data present in the conversation is extracted to help users achieve their goal. A user statement may be expressed in a variety of ways:

User: "Show me my balance?"

User: "What is my balance?"

User: "Tell me my account balance?"

To return the response to the user in this example, the chatbot will remove the necessary "balance" and "account" inputs (Dialogflow, 2018).

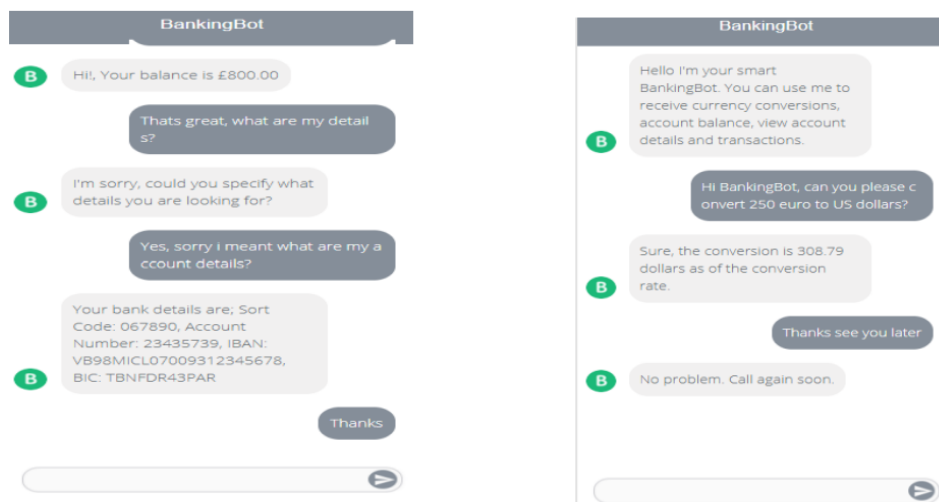


Figure 3.14 Banking Dialog

for Google Assistant, create. Compared to other traditional user interfaces, it has a more straightforward design that makes it more user-friendly and effective (DZone, 2018). By pressing the device's microphone or utilising the input area, users can immediately ask the chatbot a question. When the icon is clicked, the device's microphone is turned on, the user is made aware that a recording is being made, and the chatbot's reply is presented on the screen right away. For each message, Google Assistant employs a different colour scheme. To distinguish between different chatbot

responses, various colours will be employed. and user input.

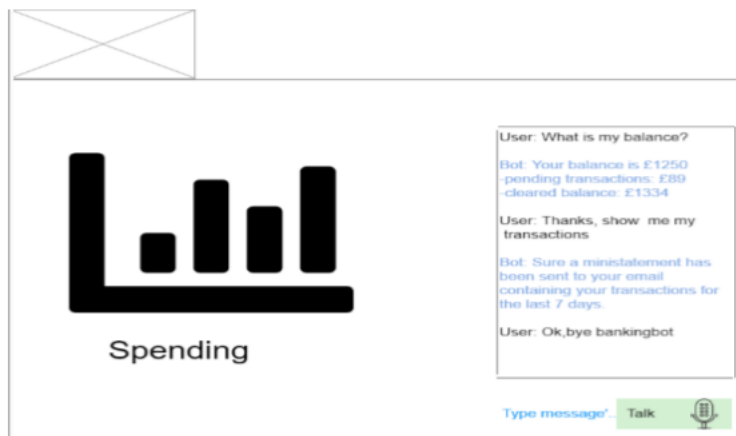


Figure 3.15 Web base chatbot

the design for the web-based chatbot view. Users can see how

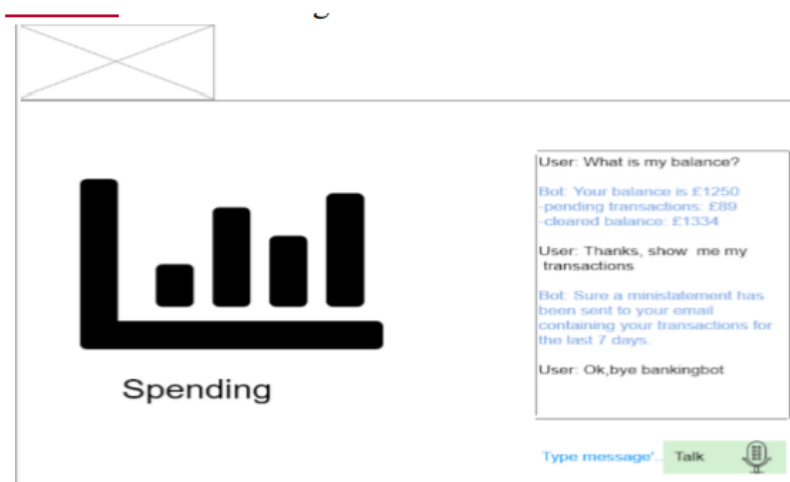


Figure 3.16 Web base chatbot

Users will be able to discover excessive spending by seeing how much they spend and what they spend their money on in a graphical style. For instance, they may realise they spend too much on coffee each week. Due to the fact that each message is given a name and colour depending on whether it was sent by the user or the bot, it is visually obvious to the user what the bot's response is.

Users can use an authenticator app on their device to scan the QR code in the image. Users will be required to enter a special code after each login,

which prevents users from seeing the chatbot view until they have verified their account. After the QR code is scanned, a special 6-digit number will be



Figure 3.17 scanner for app

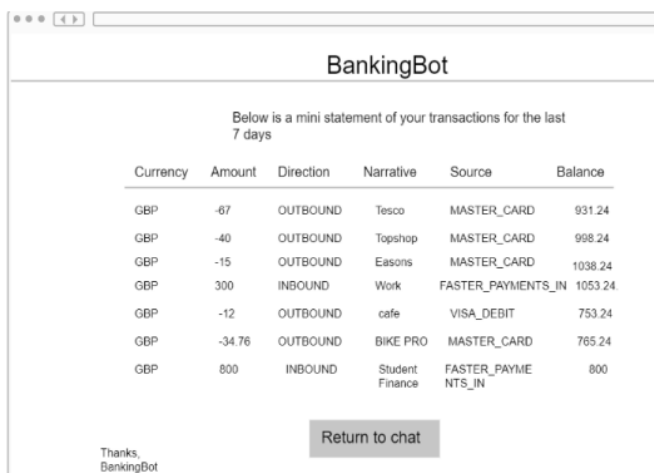


Figure 3.18 banking bot

Users won't need to switch between several websites to routinely see their transactions. An email will be sent to the user's registered email address after they request to examine their transactions.

Implementation:

We're going to create a straightforward app that includes a straightforward chatbot that will respond to queries we ask of it. See what we will be creating in this article in the video down below.

Step-by-step execution

Step 1: See How to create/start a new project in Android Studio for information on how to start a new project in Android Studio. Take note that Java is the programming language you have chosen.

Step 2: Add the dependency listed below to your build.gradle file

Add the following to the build.gradle file by going to Application > Gradle Scripts. dependency to it in dependencies section.

```
implementation 'com.android.volley:volley:1.1.1'
```

Go to the AndroidManifest.xml section after adding this dependency to your project and synchronising it.

Step 3: In the AndroidManifest.xml file, add the internet permission.

The code listed below should be added to AndroidManifest.xml under Application.

```
<!--permissions for internet-->  
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS_NETWORK_ST
```

Working with the activity_main.xml file in step 4

Add the following code to activity_main.xml by going to application > res > layout. The code for the activity_main.xml file is shown below.XML

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <!--recycler view to display our chats-->
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/idRVChats"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@id/idLLMessage" />

    <LinearLayout
        android:id="@+id/idLLMessage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:orientation="horizontal"
        android:weightSum="5">

```

```

    <!--edit text to enter message-->
    <EditText
        android:id="@+id/idEdtMessage"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="4"
        android:hint="Enter Message" />

    <!--button to send message-->
    <ImageButton
        android:id="@+id/idIBSend"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_weight="1"
        android:background="@color/purple_200"
        android:src="@android:drawable/ic_menu_send"
        android:tint="@color/white" />

</LinearLayout>
</RelativeLayout>

```

Create a modal class in step 5 to store our messages.

Go to application > java > package name of your programme > right-click on it > New > Java class and give it the name "MessageModal" before adding the following code. To make the code more comprehensible, comments are added. JAVA

```

public class MessageModal {

    // string to store our message and sender
    private String message;
    private String sender;

    // constructor.
    public MessageModal(String message, String sender) {
        this.message = message;
        this.sender = sender;
    }

    // getter and setter methods.
    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public String getSender() {
        return sender;
    }
}

```

```

    public void setSender(String sender) {
        this.sender = sender;
    }
}

```

Step 6: Make a user message layout file.

The drawing folder contains the icons used in this file. Navigate to app > res > layout, right-click on it, and choose "New > layout source file." Name the file "user_msg," and then enter the following code there.

Step 7: Design a message layout for the bot.

The drawable folder contains the characters used in this file. To create a new layout file, select New > Layout File from the context menu of the application, name it bot_msg, and then enter the following code.

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="start"
    android:layout_margin="5dp"
    android:elevation="8dp"
    app:cardCornerRadius="8dp">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <!--below widget is for image of bot-->
        <ImageView
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_margin="10dp"
            android:src="@drawable/ic_bot" />

        <!--below widget is for

```

```

        <!--below widget is for
            displaying message of bot-->
        <TextView
            android:id="@+id/idTVBot"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_margin="5dp"
            android:padding="3dp"
            android:text="Bot message"
            android:textColor="@color/black" />

    </LinearLayout>

</androidx.cardview.widget.CardView>

```

Working with the Adapter class in step 8

We must develop an Adapter class in order to set the data for our Chat RecyclerView objects. Navigate to application > java > package name of your application > right-click it > New > Java Class, give your class the name

MessageRVAdapter, and then add the following code to it. To make the code more comprehensible, comments are added.

```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class MessageRVAdapter extends RecyclerView.Adapter {

    // variable for our array list and context.
    private ArrayList<MessageModal> messageModalArrayList;
    private Context context;

    // constructor class.
    public MessageRVAdapter(ArrayList<MessageModal> messageModalArr
        this.messageModalArrayList = messageModalArrayList;
        this.context = context;
    }
}
```

```
@NonNull
@Override
public RecyclerView.ViewHolder onCreateViewHolder(@NonNull View
    View view;
    // below code is to switch our
    // layout type along with view holder.
    switch (viewType) {
        case 0:
            // below line we are inflating user message layout.
            view = LayoutInflater.from(parent.getContext()).inf
            return new UserViewHolder(view);
        case 1:
            // below line we are inflating bot message layout.
            view = LayoutInflater.from(parent.getContext()).inf
            return new BotViewHolder(view);
    }
    return null;
}
```

```

@Override
public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int position) {
    // this method is use to set data to our layout file.
    MessageModal modal = messageModalArrayList.get(position);
    switch (modal.getSender()) {
        case "user":
            // below line is to set the text to our text view o
            ((UserViewHolder) holder).userTV.setText(modal.getM
            break;
        case "bot":
            // below line is to set the text to our text view o
            ((BotViewHolder) holder).botTV.setText(modal.getMes
            break;
    }
}

@Override
public int getItemCount() {
    // return the size of array list
    return messageModalArrayList.size();
}

```

```

@Override
public int getItemCount() {
    // return the size of array list
    return messageModalArrayList.size();
}

@Override
public int getItemViewType(int position) {
    // below line of code is to set position.
    switch (messageModalArrayList.get(position).getSender()) {
        case "user":
            return 0;
        case "bot":
            return 1;
        default:
            return -1;
    }
}

public static class UserViewHolder extends RecyclerView.ViewHolde

    // creating a variable
    // for our text view.
    TextView userTV;

```



```

public ViewHolder(@NonNull View itemView) {
    super(itemView);
    // initializing with id.
    userTV = itemView.findViewById(R.id.idTVUser);
}
}

public static class BotViewHolder extends RecyclerView.ViewHolder

    // creating a variable
    // for our text view.
    TextView botTV;

    public BotViewHolder(@NonNull View itemView) {
        super(itemView);
        // initializing with id.
        botTV = itemView.findViewById(R.id.idTVBot);
    }
}

```

Create an API key in step 9 to access the chatbot service.

Check out [Brainshop.ai](https://brainshop.ai) With your username and password, you can easily establish an account. Just register for an account on our website. The screen below is what you will see after making a new account. You must input your email address after creating an account and select the Require password option to request a new password.

After entering your email address, you must update your account's password. We may now begin writing the API code.

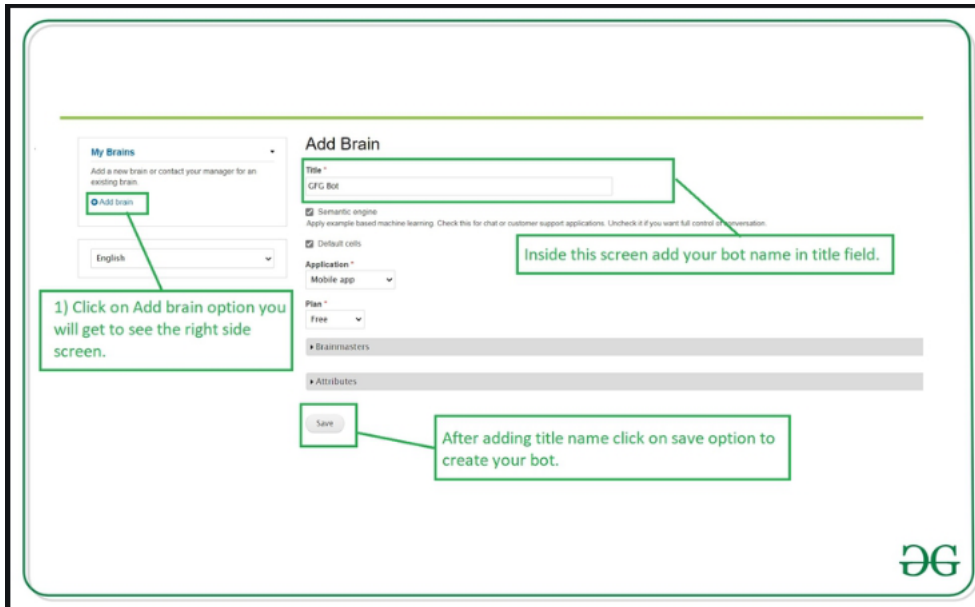


Figure 3.19

To build a new brain for your chatbot, adhere to the aforementioned instructions. Now that your bot has been created, let's retrieve the API URL for that brain. You can see the details of your bot as shown below by going to the settings page in your newly constructed brain.



Figure 3.20

Use the MainActivity.java file in step 10.

the code below into the MainActivity.java file. The code for the MainActivity.java file is shown below.

To make the code easier to read, comments were added.

JAVA:

```
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
```

```

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    // creating variables for our
    // widgets in xml file.
    private RecyclerView chatsRV;
    private ImageButton sendMsgIB;
    private EditText userMsgEdt;
    private final String USER_KEY = "user";
    private final String BOT_KEY = "bot";

    // creating a variable for
    // our volley request queue.
    private RequestQueue mRequestQueue;

    // creating a variable for array list and adapter class.
    private ArrayList<MessageModal> messageModalArrayList;
    private MessageRVAdapter messageRVAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

```

```

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // on below line we are initializing all our views.
        chatsRV = findViewById(R.id.idRVChats);
        sendMsgIB = findViewById(R.id.idIBSend);
        userMsgEdt = findViewById(R.id.idEdtMessage);

        // below line is to initialize our request queue.
        mRequestQueue = Volley.newRequestQueue(MainActivity.this);
        mRequestQueue.getCache().clear();

        // creating a new array list
        messageModalArrayList = new ArrayList<>();

        // adding on click listener for send message button.
        sendMsgIB.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // checking if the message entered
                // by user is empty or not

```

```

// by user is empty or not.
if (userMsgEdt.getText().toString().isEmpty()) {
    // if the edit text is empty display a toast message.
    Toast.makeText(MainActivity.this, "Please enter your
return;
}

// calling a method to send message
// to our bot to get response.
sendMessage(userMsgEdt.getText().toString());

// below line we are setting text in our edit text as emp
userMsgEdt.setText("");
}
});

// on below line we are initializing our adapter class and passin
messageRVAdapter = new MessageRVAdapter(messageModalArrayList, th

// below line we are creating a variable for our linear layout ma
LinearLayoutManager layoutManager = new LinearLayoutManager

// below line is to set layout
// manager to our recycler view.
chatsRV.setLayoutManager(layoutManager);

```

```

// adapter to our recycler view.
chatsRV.setAdapter(messageRVAdapter);
}

private void sendMessage(String userMsg) {
    // below line is to pass message to our
    // array list which is entered by the user.
    messageModalArrayList.add(new MessageModal(userMsg, USER_KEY));
    messageRVAdapter.notifyDataSetChanged();

    // url for our brain
    // make sure to add mshape for uid.
    // make sure to add your url.
    String url = "Enter you API URL here" + userMsg;

    // creating a variable for our request queue.
    RequestQueue queue = Volley.newRequestQueue(MainActivity.this);

    // on below line we are making a json object request for a get
    JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Rec
@Override
public void onResponse(JSONObject response) {
    try {

```

```

public void onResponse(JSONObject response) {
    try {
        // in on response method we are extracting data
        // from json response and adding this response to our array
        String botResponse = response.getString("cnt");
        messageModalArrayList.add(new MessageModal(botResponse,

        // notifying our adapter as data changed.
        messageRVAdapter.notifyDataSetChanged();
    } catch (JSONException e) {
        e.printStackTrace();

        // handling error response from bot.
        messageModalArrayList.add(new MessageModal("No response"
        messageRVAdapter.notifyDataSetChanged();
    }
}
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        // error handling.
        messageModalArrayList.add(new MessageModal("Sorry no respons
        Toast.makeText(MainActivity.this, "No response from the bot.
    }
});

```

```

    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        // error handling.
        messageModalArrayList.add(new MessageModal("Sorry n
        Toast.makeText(MainActivity.this, "No response from
    }
});

// at last adding json object
// request to our queue.
queue.add(jsonObjectRequest);
}
}

```

CHAPTER 04

EXPERIMENTS & RESULT ANALYSIS

4.1 Testing:

Testing is a crucial component of every software development life cycle. In order to comprehend the software's limitations, this requires carrying out specific activities and procedures. It is clear that specific problems and faults are recorded in test cases while testing application restrictions. This will raise the chatbot's general bar and level of excellence while also enhancing user experience. To assess the chatbot's overall performance, several testing techniques were used. The usefulness of the chatbot was tested through a dialogue, which involved gauging how effectively the chatbot could comprehend a user-supplied utterance even if you didn't type it. With average response times for voice and text exchanges, the response also indicates whether the intent was understood.

4.2 Testing the simulated conversation:

The chatbot's comprehension of user speech for voice and text interactions, even when it is not written, was examined as part of the dialog's efficacy evaluation. With average response times for voice and text exchanges, the response also indicates whether the intent was understood. Performance data for the chatbot have been tracked through certain testing procedures.

A JavaScript command-line tool called dialogflow was installed in the project directory specifically for the purpose of running these tests.

Metrics were recorded for each simulated encounter, which was utilised to mimic how users would communicate with the chatbot. The tool instantiates the Dialogflow API on the command line, fires the welcome event for it, and then shows the response it returns as the command-line result to begin the simulated interaction for testing. The command line was used to test each sort of use case, and JSON formatted results were produced. Twenty simulated utterances in total were evaluated, and Table 5's findings are displayed there. Each remark tested was specifically word

Simulated User Utterance (including misspelt phrases)	Confidence Score	Matched Intent(s)	Unrecognised Intent(s)	Response Time (ms)
Hello bankingbot, what is my balance?	1	✓	✗	3480
Hello bankingbot, please show me my balance?	0.9999	✓	✗	3414
Display my blance	1	✗	✓	N/A
Show me my account details please?	1	✓	✗	3446
"account details" (account details)	1	✓	✗	2181
Can I see my account details	0.8700	✓	✗	3480
How much money have I spent this week?	0.8299	✓	✗	3649
What hove I spunt	0.3000	✓	✗	3308
Wat have I spnt this weke	0.3300	✓	✗	2504
Can you send me my transactions for the last seven _days?	1	✓	✗	7339
tamsucyions	1	✗	✓	N/A
how much is 5000 Canadian _dollars in Japanese yen	0.8900	✓	✗	3228
Convert 600 euro to Japanese yen	0.9499	✓	✗	3182
and 700 euro	1	✓	✗	1613
and 800 pounds	1	✓	✗	3381
what about 500 US dollar	1	✓	✗	3128
Can ypu please book me an appointment please?	0.6200	✓	✗	3380
buuk me apptmnt	1	✓	✗	3345

Table 4.1 Simulated User Phrases: Test Results

Book me an appointment for squivings on the 23/04/2018	1	✓	*	3322
Ok, my phone number is 089456787642	1	✓	*	4212

Table 4.2 Simulated User Phrases: Test Results

The outcomes of simulated "typical" chatbot conversations with the command line tool are displayed in the table. The dialogue for the chatbot was created during the design process using simulated phrases that were taken from user questionnaires and by watching people interact with the chatbot. Each statement is given a "Confidence Score" that ranges from 0.0 to 1, with 1 denoting a perfect match and 0 denoting a failure to match the statement to any intent. However, NLU additionally assigned a full confidence score of 1 to phrases or utterances where no purpose was discernible because this correlated with the default fallback intent to conveniently handle unforeseen user input. It is evident that reaction time varies depending on the fundamental features a user requires to accomplish a goal, such as

Only 10% of utterances do not fit the chatbot's intentions, which is a relatively high percentage. It also accounts for typographical errors. 80% of the 20 user phrases had successful matching intents overall. These findings show that the chatbot can comprehend the majority of sentences, including those with spelling mistakes, which speaks to the conversational quality. Although the phrase "what I spent this week" (which meaning "what I spent this week") only received a perfect 1 and a low confidence score of 0.33, the chatbot was nevertheless able to comprehend the statement's intent and identify the action, as seen in the figure.

typo.

```
wat have i spnt this weke
[2018-04-15 18:16:24.575] [DEBUG]
id: a962582a-973d-4d74-a116-e143c499fb2b
timestamp: 2018-04-15T18:16:20.988Z
lang: en
result:
  source: agent
  resolvedQuery: wat have i spnt this weke
  action: spent
  actionIncomplete: false
  parameters:
    date-period:
      spend:
        contexts: []
  metadata:
    intentId: 2194539a-bb3d-47f6-8b7b-e12fe1af8feb
    webhookUsed: true
    webhookForSlotFillingUsed: false
    webhookResponseTime: 2584
    intentName: bot-spent
  fulfillment:
    speech: You have spent -190.36 this week
    displayText: You have spent -190.36 this week
    messages:
      - type: 0
        speech: You have spent -190.36 this week
    score: 0.33000001311302185
  status:
    code: 200
    errorType: success
    webhookTimedOut: false
  sessionId: 56038af0-40d3-11e8-bb3e-57c75b419c46
You have spent -190.36 this week
```

<i>Average Understanding Score</i>	<i>Average Response Time</i>	<i>Dialog Duration</i>
17.7897/20= 0.8894	60594 (ms) 60.594(sec) / 20 = 3.02 sec	3 minutes and 26 seconds

The rounded average response time was 3 seconds, which is a reasonable amount of time. This implies that users will be able to complete their required activities through interaction with the chatbot quite fast, which is also represented in the total length of the dialogue, as shown in Table 76, which shows the overall interaction time. Instead of using simulated outcome phrases, the evaluation will employ user test results to further explain the results.

Through the Actions on Google (AoG) emulator in the AoG interface, Google Assistant and Home Agent were both evaluated. While Google Home was tested using voice interaction on the AoG simulator, Google Assistant integration was tested using text input through the simulator. If the utterance matched the intent phrase, the simulator delivers a JSON object with information on the chatbots' level of understanding. Similar language was used in all facilities to describe any differences in how NLU was understood.

<u>Intent</u>	<u>Google Assistant</u> <i>(Text)</i>	<u>Matched</u> <u>Intent(s)</u>	<u>Google Home</u> <i>(voice)</i>	<u>Matched</u> <u>Intent(s)</u>	<u>Web APP</u> <i>(Voice)</i>	<u>Matched</u> <u>Intent(s)</u>
<u>1.</u>	<u>Correctly spelt phrase:</u> <u>1.What is my balance</u>	✓	<u>Correctly pronounced phrase:</u> <u>1.What is my balance</u>	✓	<u>Correctly pronounced phrase:</u> <u>1.What is my balance</u>	✓
	<u>Incorrectly spelt phrases:</u> <u>1.What is my balan</u> <u>2. Wat is my bolonce</u>	x ✓	<u>Incorrectly pronounced phrases:</u> <u>1.What is my balan</u> <u>2.Wat is my bolonce</u> <u>Detected as:</u> <u>1.What is my balance</u> <u>2.What is my balance</u>	✓ ✓	<u>Incorrectly pronounced phrases:</u> <u>1.What is my balan</u> <u>2.Wat is my bolonce</u> <u>Detected as:</u> <u>1.What is my balance</u> <u>2.What is my bowl</u>	✓ x
<u>2.</u>	<u>Correctly spelt phrase:</u> <u>1.Show me my transactions</u>	✓	<u>Correctly pronounced phrase:</u> <u>1.Show me my transactions</u>	✓	<u>Correctly pronounced phrase:</u> <u>1.Show me my transactions</u>	✓
	<u>Incorrectly spelt phrases:</u> <u>1.Shw me mi tronsoctions</u>	x	<u>Incorrectly pronounced phrases:</u> <u>1.Show me my Tronsoctions</u> <u>Detected as:</u> <u>1.Show me my Transactions</u>	✓	<u>Incorrectly pronounced phrases:</u> <u>1.Show me my tronsoctions</u> <u>Detected as:</u> <u>1.Show me my Transactions</u>	✓
<u>3.</u>	<u>Correctly spelt phrase:</u> <u>1.Convert 200 Autralian dollars to pounds</u>	✓	<u>Correctly pronounced phrase:</u> <u>1.Convert 200 Autralian dollars to pounds</u>	✓	<u>Correctly pronounced phrase:</u> <u>1.Convert 200 Australian dollars to pounds</u>	✓

	<u>Incorrectly spelt phrases:</u> 1. Convert 200 pound to yoyo	x	<u>Incorrectly pronounced phrases:</u> Convert 200 pound to yoyo <u>Detected as:</u> Convert 200 pond to euro	✓	<u>Incorrectly pronounced phrases:</u> Convert 200 pound to yoyo <u>Detected as:</u> Convert 200 pound to euro	✓

Intent	Google Assistant (Text)	Matched Intent(s)	Google Home (Voice)	Matched Intent(s)	Web-APP (Voice)	Matched Intent(s)
1-	<u>Correctly spelt phrase:</u> 1. What is my balance	✓	<u>Correctly pronounced phrase:</u> 1. What is my balance	✓	<u>Correctly pronounced spelt phrase:</u> 1. What is my balance	✓
	<u>Incorrectly spelt phrases:</u> 1. What is my balan 2. Wat is my bolonce	* ✓	<u>Incorrectly pronounced spelt phrases:</u> 1. What is my balan 2. Wat is my bolonce <u>Detected as:</u> 1. What is my balance 2. What is my balance	✓ ✓ ✓	<u>Incorrectly pronounced spelt phrases:</u> 1. What is my balan 2. Wat is my bolonce <u>Detected as:</u> 1. What is my balance 2. What is my bow!	✓ * ✓ *
2-	<u>Correctly spelt phrase:</u> 1. Show me my transactions	✓	<u>Correctly pronounced spelt phrase:</u> 1. Show me my transactions	✓	<u>Correctly pronounced spelt phrase:</u> 1. Show me my transactions	✓

	<p>Incorrectly spelt phrases:</p> <p>1.Show me my transactions</p>	✗	<p>Incorrectly pronounced spelt phrases:</p> <p>1.Show me my Transactions</p> <p>Detected as:</p> <p><u>1.Show me my Transactions</u></p>	✓	<p>Incorrectly pronounced spelt phrases:</p> <p>1.Show me my transactions</p> <p>Detected as:</p> <p><u>1.Show me my Transactions</u></p>	✓
3-	<p>Correctly spelt phrase:</p> <p>1.Convert 200 Australian dollars to pounds</p>	✓	<p>Correctly pronounced spelt phrase:</p> <p>1.Convert 200 Australian dollars to pounds</p>	✓	<p>Correctly pronounced spelt phrase:</p> <p>1.Convert 200 Australian dollars to pounds</p>	✓
	<p>Incorrectly spelt phrases:</p> <p>1.Convert 200 pound to yoyo</p>	✗	<p>Incorrectly pronounced spelt phrases:</p> <p>Convert 200 pound to yoyo</p> <p>Detected as:</p> <p><u>Convert 200 pond to euro</u></p>	✓	<p>Incorrectly pronounced spelt phrases:</p> <p>Convert 200 pound to yoyo</p> <p>Detected as:</p> <p><u>Convert 200 pound to euro</u></p>	✓

Table 4.3 Simulated Device testing

	<i>Google Assistant</i>	<i>Google Home</i>	<i>Web App (HTML5 Speech API)</i>
<i>% Response Accuracy</i>			
% Matched Intent	<u>57.14%</u>	<u>100%</u>	<u>85.71%</u>

Table 4.4 Overall accuracy comparison of Devices

On numerous simulated devices, extensive testing was done. Google Home was discovered to be able to discern speech intent even when produced spoken words after testing on numerous simulated devices. In addition, it was discovered that Google Home could identify speech in which some words were missing or mispronounced, as demonstrated in Table 87. It was discovered that the text input given to Google Assistant on Android phones had stricter syntax than anticipated. the given input text to match the defined intent used in the training phrases. This is only true if the user provides the input text, though, as Google Assistant uses the same speech recognition and machine learning algorithms regardless of device .

Although Google Assistant uses the same speech recognition and machine learning techniques regardless of the device it is installed into, Assistant adheres to a tighter NLP syntax when the user provides the input text. Even if the user's speech is inaccurate, software in Google and Assistant turns it into the correct input text before it is transmitted to the NLU. Consequently, Google Home Because the interface was solely voice instructions, Google Home was able to recognise and detect missing or incorrectly uttered words. These findings show that voice interaction is the best way to connect with a chatbot since it can recognise pronunciation problems and respond appropriately.

As can be seen from the findings in Table 9, the Google Home device was the primary means of communication with the chatbot since it was able to accurately match the intent of each delivered speech, earning a score of 100%. When compared to text-based interactions, conversational interfaces using spoken natural language performed better, with Google Assistant scoring 57.14%. The two conversational interfaces are very different from one another, indicating that speaking when interacting with natural language conversational interfaces is far more effective. With an average speech recognition rate of 85.71%, Google Assistant's speech recognition outperforms the HTML5 Speech API, suggesting that Google's platforms will function more effectively during actual user interactions.

Intent	(n) Turns Before Exit
1	3
2	4
3	4

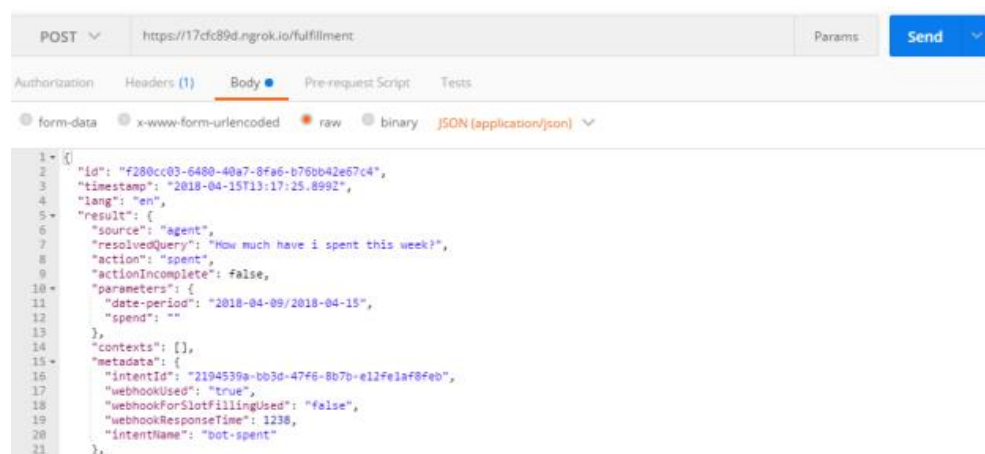
Table 4.5 Conversation exit rate

The table shows how many attempts the chatbot made to understand the meaning of the misspelt phrase defined in Table 87 7 before terminating the discussion. The chatbot provided backup responses at every opportunity, including "Sorry, I didn't quite get that, can you repeat that

please?" and "Sorry, I'm having trouble understanding what you said, can you say it again?" When the chatbot doesn't understand a specific statement or receives unexpected input, this functionality takes care of it. Even if the user statement was the identical each time, this also describes the use of various fallback intentions. Due to the user's frequent requests for the acceptable phrase and its absence, as well as its inability to, the chatbot left the conversation.

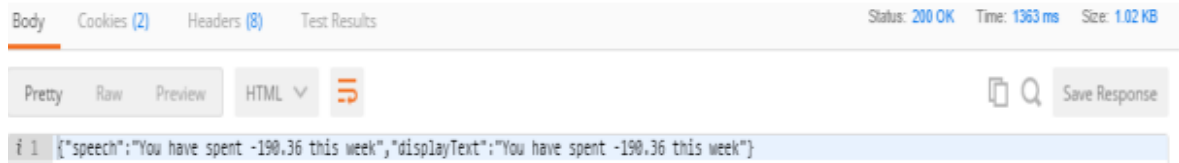
4.3 Testing the webhook:

A programme called Postman was used to send HTTPS POST requests to Webhook in order to verify the custom fulfilment offered by Webhook. This was done in order to test the Webhook's reaction to a specific user statement. In a POST request to the Webhook, the request body is retrieved from the Dialogflow developer console and sent there. The user phrase and other NLU-returned parameters are represented by a JSON object in the body, as seen in the picture below.



```
1 {
2   "id": "f280cc03-6480-40e7-8fa6-b76bb42e67c4",
3   "timestamp": "2018-04-15T13:17:25.899Z",
4   "lang": "en",
5   "result": {
6     "source": "agent",
7     "resolvedQuery": "How much have i spent this week?",
8     "action": "spent",
9     "actionIncomplete": false,
10    "parameters": {
11      "date-period": "2018-04-09/2018-04-15",
12      "spend": ""
13    }
14  },
15  "contexts": [],
16  "metadata": {
17    "intentId": "2194539e-bb3d-47f6-8b7b-e12fe1af8feb",
18    "webhookUsed": "true",
19    "webhookForSlotfillingUsed": "false",
20    "webhookResponseTime": 1238,
21    "intentName": "bot-spent"
22  }
23 }
```

Bugs can be fixed before the agent is given access to technologies like Google Assistant and Google Home or web-based chatbots since Postman is used to verify the operation of the Webhook in the distinct context in which it was constructed. Each time the response is returned to the event number, this will identify the network and log the issue. The picture below displays an example of a JSON response that the webhook returned.



The screenshot shows a web browser's developer console with the following elements:

- Top navigation: Body, Cookies (2), Headers (8), Test Results
- Status: 200 OK, Time: 1363 ms, Size: 1.02 KB
- View options: Pretty, Raw, Preview, HTML (dropdown), and a refresh icon.
- Search and Save Response buttons.
- Console output: A single log entry with index 1 and the JSON object: `{"speech": "You have spent -190.36 this week", "displayText": "You have spent -190.36 this week"}`

To determine whether it could successfully contact the TrueLayer API and determine any necessary parameters or authentication headers that had to be given with each subsequent request, the integration of other APIs implemented inside the project was also put to the test.

4.4 User Testing:

The Google Assistant, home, and web chatbots were chosen to be the three chatbot communication channels for user testing. As determined by the user's interaction experience, this will shed light on the chatbot's actual quality and general usefulness. A list of inquiries was developed to assess the chatbot during user testing. A group of users were given this questionnaire, which can be found in the report's appendix, in response to these inquiries.

4.5 Evaluation:

By outlining performance attributes and examining the outcomes, software evaluation aims to determine a chatbot's quality. Future work as well as a reflection of already finished work are desired. The user survey results shed a lot of light on the general response and accuracy rates. A user group comprising fifteen people with varied technical backgrounds received questionnaires. At the School of Engineering and Computing Open Day at Ulster University, these were filled out by parents and prospective students. This user base accurately represents the intended market for an app like this. Results from user surveys and simulated user interactions will be examined in order to distinguish between subjective and objective measures. Users were seen to record free-standing data regarding during the questionnaire.

Subjective metrics	(n)Users	Objective metrics	(n)Users
Naturalness	44.00%	Speech Recognition	86.67%80.00%
Likeability	93.33%	Response accuracy	86.67%
Ease of use	86.66%	Response rate	73.33%
Speech Recognition	80.00%		

Table 4.6 Subjective & objective measurement

Due to low user satisfaction when using their services, as was stated in Section 1.2, the majority of banks have trouble encouraging their consumers to adopt the technology. The incorporation of a chatbot into their financial services would greatly raise the degree of user happiness, according to the findings in Table 119. This is clear from the fact that compared to other users, 73.33% of all users believed the chatbot was "Extremely" fast. considerably quicker than the existing technologies their bank offers, such an online banking app. The results of the user testing phase confirmed that chatbots can successfully engage users and boost satisfaction: 86.66% of all respondents said that the chatbot was either "extremely very easy" or "moderately easy," easy to talk to, and achieved an overall likeability of 93.33 out of 93.33%, as indicated in table 119. The performance and quality of chatbots in actual contact are measured objectively using the questionnaire results. Overall, the chatbot's voice recognition rate was quite good; 80.00% of users said the chatbot understood them "very well." This metric shows the efficacy of simulated activities used in testing, as seen in Table 7. According to Table 6 of Section 5, the chatbot achieved an overall understanding score of 0.8894 (89.00%) on the simulated exams. The understanding scale's upper limit is reached with this outcome.

CHAPTER 05

5.1 CONCLUSIONS

In Section 1.3, Figure 1, the emergence and popularity of chatbots are obviously depicted. In light of this, the information gleaned from chatbot testing supports the recent increase in demand from businesses wishing to implement a chatbot. In comparison to conventional approaches, chatbots have been discovered to function at a very high level and offer users trustworthy and prompt replies. The chatbot offers consumers an effective way to handle their banking, thus users spend relatively little time engaging with it on average. Due to conversational user interfaces' high degree of comprehension and speech recognition, users may easily engage with chatbots to satisfy the demands of modern life thanks to their short interaction times. The chatbot has demonstrated its capacity to satisfy user needs for instant access to information and services.

Using sentiment analysis to include user sentiment. NLP methods might be used to examine a user's transactions to determine their spending patterns. It would be a smart idea to expand into this market as the Internet of Things continues to develop, as chatbots generate a significant quantity of data that is required for the success of IoT and Big Data ideas, which go hand in hand with further developments in the Internet of Things. Machine learning and AI. The Google Home line of products might be used to integrate IoT.

5.2 FUTURE ADVANCEMENT

One of the integration streams originally considered during development was to integrate the chatbot with Facebook Messenger, but during development this was revealed by recent media events such as Cambridge Analytic, a discrediting scandal that resulted in Facebook halting the

process of allowing other applications to integrate with its review of messenger apps, so this integration was not developed because Facebook stopped allowing developers to create new apps or chatbots through the service. Although Facebook has just recently reopened its app review process, it is now allowing developers to re-integrate with Messenger, allowing development of this integration feature for future prototypes (Facebook, 2018). Google authentication integration would be a beneficial feature for an implementation that allows users to link their Google account across multiple devices where the agent is distributed, such as Google Assistant, which improves the cross-platform experience for users.

A user can find out financial information about their account, such as how much they spent in a week. Based on this feature, it was thought that during the later stages of development, users should be able to directly query the chatbot to find out where and what they are spending their money on. Although users can view this information in a graphical format in a web application, it would also be useful if they could discover it through interaction with a chatbot.

The chatbot could also be developed to be multilingual. Support for multiple languages would help eliminate banking in developing countries and improve the overall availability of technology in the banking industry, while targeting a larger group of users.

From observing user interaction with the chatbot during user testing, there were many suggestions to integrate the chatbot into other popular platforms such as Amazon Echo or Dot, which would increase

REFERENCES

- [1] Ling, G., Fern, Y., Boon, L. and Huat, T. (2016). Understanding Customer Satisfaction of Internet Banking: A Case Study In Malacca. 'Procedia Economics and Finance', 37, pp.80-85.n (Accessed 11/10/2017)
- [2] Barty, J. and Recketts, T. (2014). 'Promoting Competition in the UK banking Industry', Promoting Competition in the UK banking Industry. BBACompetitionReport.,pp.4.Availableat:https://www.bba.org.uk/wpcontent/uploads/2014/06/BBA_Competition_Report_23.06_WEB_2.0.pdf (Accessed 10/10/2017).
- [3] Aburub, F., Odeh, M., & Beeson, I. (2007). 'Modelling non-functional requirements of business Processes'. Information and Software Technology, 49(1112),11621171.Availableat:<http://dx.doi.org/10.1016/j.infsof.2006.12.002> (Accessed 10/10/2017).
- [4] Ling, G. M., Fern, Y. S., Boon, L. K., & Huat, T. S. (2016). Understanding customer satisfaction of internet banking: A case study in malacca. Available at doi:[https://doi.org/10.1016/S2212-5671\(16\)30096-X](https://doi.org/10.1016/S2212-5671(16)30096-X)
- [5] Asif Irshad Khan, A.I., Rizwan Jameel Qurashi, R. J and Usman Ali Khan, U. A. (2011). 'A Comprehensive Study of Commonly Practiced Heavy and Light Weight Software Methodologies ', IJCSI International Journal of Computer Science Issues, 8(4), pp. [Online]. Available at: <https://arxiv.org/ftp/arxiv/papers/1111/1111.3001.pdf>(Accessed:08/11/2017)
- [6] Pauline M. McGuirk, P. M. Phillip and O'Neill, P. (2016). 'Using questionnaires in qualitative human geography', University of Wollongong, FacultyofSocialSciences,(),pp.11[Online].Availableat:<http://ro.uow.edu.au/cgi/viewcontent.cgi?article=3519&context=sspapers>(Accessed: 12/11/2017).

