UNIVERSITY INSTITUTE OF COMPUTING

DIVISION- MCA/BCA/BSc(CS)

A Final Project Report

On

College Enquiry Chatbot

Submitted in partial fulfillment of the requirements for the

award of the degree of

Bachelor of Computer Applications (BCA)

in

Session: 2019-22

By

STUDENT1, REG ID: RITIKA RANA 19BCA1126

STUDENT 2, REG ID: ATHARV SINGH 19BCA1132

STUDENT 3, REG ID: PRABHAT Kr. CHAUDHARY 19BCA1172

Supervised By

Mr. Deepak Vats

Assistant Professor

University Institute of Computing

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI,

PUNJAB,140413

Project Coordinator **Head-UIC**

Acknowledgment

"It is not possible to prepare a project report without the assistance and guidance of the other people. This one is certainly no exception".

At the start of the report, I would like to extend my sincere respect towards all the notable people who had helped me throughout and my group members in this endeavor without their guidance and uninterrupted support, It wouldn't be possible without them.

I am ineffably indebted to Mr. Deepak Vats (Assistant Professor) of the UIC department, for his meticulous guidance and encouragement to accomplish this project. I am extremely thankful to Dr. Manisha Malhotra, (HOD) of the UIC department, Chandigarh University for her valuable guidance and support in completing this project on time. I extend my gratitude to Chandigarh University for giving me this opportunity to perform this project and giving me a chance to learn something new.

I also acknowledge with a deep sense of reverence and my gratitude towards my family members, who had always supported me positively and economically.

At last but not least gratitude goes to all my friends who directly or indirectly supported me and my team members for completing this project report.

UNIVERSITY INSTITUTE OF COMPUTING

DIVISION- MCA/BCA/BSc(CS)

Declaration

We hereby declare that the project work entitled "CHAT-BOT" submitted to The Chandigarh University, is a record of an original work done by us under the guidance of Mr. Deepak Vats, Assis. Professor of University Institute of Computing, Chandigarh University. This project work is submitted in partial fulfillment of the requirements for the award of the degree of Bachelors in Computer Applications for the session 2019-2022.

We also further certify that

- The work contained in the report is original and has been done by us under the supervision of our supervisor.
- The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or the any other University of India or abroad.
- We have followed the guidelines provided by the university in writing the report.
- Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and given their details in the references.
- The Android App has been designed by us only and its function and behavior are developed by our team.

Team Members

Member Name	UID	Signature
Prabhat Kr. Chaudhary	19BCA1172	*prabhat*
Atharv Singh	19BCA1132	*atharv*
Ritika Rana	19BCA1126	*ritika*

UNIVERSITY INSTITUTE OF COMPUTING

DIVISION- MCA/BCA/BSc(CS)

Training Letter

Dear Students,

This letter is to confirm that the University Institute of Computing is offering you to work as a trainee for the Minor Project of 5th Semester. Your training project will be approx. three months duration and will begin on 15th of August and end on 5th of November.

This program will serve as Academic training following your Bachelor of Computer Applications (BCA) degree in UIC at Chandigarh University. Through this training program, you will gain experience in the field of College Enquiry Chatbot.

Your training supervisor will be Mr. Deepak Vats. Your training supervisor's contact information is as follows: Mob: +918198911043.

Sincerely,

Project Coordinator

UIC

UNIVERSITY INSTITUTE OF COMPUTING

DIVISION- MCA/BCA/BSc(CS)

Certificate by the Supervisor

This is to certify that the work incorporated in the project report entitled "***College Enquiry Chatbot***" is a record of work carried out by **Ritika Rana (19BCA1126), Atharv Singh (19BCA32), Prabhat Kumar Chaudhary (19BCA1172)** under my guidance and supervision for the award of Degree of **Bachelor of Computer Applications** in the faculty of Department of University Institute of Computing of Chandigarh University, Mohali, Punjab, India.

To the best of my/our knowledge and belief the project report:

- Embodies the work of the candidates themselves
- Has duly been completed
- Fulfills the requirement of the Ordinance relating to the Bachelor's degree of the University
- All help received by them from various sources have been duly acknowledged
- No part of this report has been submitted elsewhere for the award of any other degree and
- Is up to the desired standard both in respect of contents and language for being referred to the examiners.

MR. DEEPAK VATS

Supervisor

Assistant Professor

The project work as mentioned above is hereby being recommended and forwarded for examination and evaluation.

Project Coordinator **Head-UIC**

UNIVERSITY INSTITUTE OF COMPUTING

DIVISION- MCA/BCA/BSc(CS)

Abstract

Individuals collaborate with frameworks increasingly more through voice colleagues and chatbots. The times of exclusively captivating with assistance through a console are finished. These new methods of client communication are supported part by progressions in Artificial Intelligence and Machine Learning innovation.

This task is expected to carry out an electronic chatbot to help with web-based banking, utilizing devices that uncover man-made brainpower strategies, for example, regular language understanding. Permitting clients to connect with the chatbot utilizing regular language input and to prepare the chatbot utilizing suitable techniques so it will want to produce a reaction. The chatbot will permit clients to see all their financial data from inside the chatbot.

The created model was observed to be an exceptionally valuable device to legitimize the requirement for an advanced strategy for an association to be coordinated inside many administrations presented by banks and monetary organizations. In an industry with low client fulfillment rates and restricted innovation to expand openness. The chatbot conquers the difficulties banks face to build the utilization of their administrations and gain a strategic advantage over driving contenders.

Many individuals embrace Smart Assistant Devices like Google Home or Amazon's Alexa. The chatbot was tried across a scope of gadgets, for example, Google Home and Assistant on android gadgets to lay out the critical contrasts between the two methods of association, spoken and text exchange. These tests were completed to recognize the worth of coordinating such innovation encompassing the new interest in chatbots and conversational interfaces. Demonstrating chatbots can be applied to a particular area to improve availability, reaffirming that they are something beyond a passing trend and have a reasonable use.

UNIVERSITY INSTITUTE OF COMPUTING

DIVISION- MCA/BCA/BSc(CS)

Preface

This Project Report has been prepared in fulfillment of the requirement for the Subject. Studies of the program B.C.A. in (Sem V) in the academic year 2019-2022. This project gives an insight into Chat-Bot. It's currently being used in numerous fields like online customer support, phone interactions, information retrieval, or assisting with online commerce or tech support.

Because it is easy to deploy, companies find them a great first use case for AI within their organization. This project will help to know the growing demands for Chat-Bot be it an organization or any other spear and also how things work smoothly without any hindrance of multiple websites. Information within minutes.

UNIVERSITY INSTITUTE OF COMPUTING

DIVISION- MCA/BCA/BSc(CS)

Table of Contents

Table of Figures

Table of Tables

Chapter 1: Introduction

Conversational bots have a wide range of applications, and in recent years they have been developed for use in education. The ability to handle tens of thousands of questions and answers at the same time provides an alternative to teachers and lecturers who are facing a shortage of time and expertise. Chatbots have become more prevalent in recent years, and different types have emerged for different purposes. Non-deterministic or evolving (dynamic) chatbots have a learning component that allows them to take past interactions and outcomes and alter their behavior when they meet the same situation. Dialogflow may be a tongue-understanding platform that creates it easy to style and integrate a conversational interface into your mobile app, web application, device, bot, interactive voice response system, and so on. Using Dialogflow, you'll provide new and interesting ways for users to interact together with your product. Dialogflow can analyze multiple sorts of input from your customers, including text or audio inputs (like from a phone or voice recording). It also can answer your customers in a few ways, either through text or with synthetic speech.

Background

Dialogflow provides the user with a web user interface known as Dialogflow CX Console. A user can use this console to create, build and test different CX agents.

The CX console is quite similar to the ES console, but the main difference is the user interface which is more visual in the CCX console. It uses different types of graphs that represent each flow as a conversational state machine diagram, which helps the user to understand complex agents easier to understand.

The Dialogflow CX console is quite different from the Google Cloud Platform console. these both have their different uses as Dialogflow CX console manages Dialogflow CX agents, on the other hand, Google Cloud Platform console manages GCP-specific Dialogflow CX settings as billing and other GCP resources.

Most users should prefer using the Dialogflow CX console to build agents, and as per the higher or advanced requirements, users can also use the Dialogflow CX API to build agents.

QuickStart, concept and how-to guides walk you through the console's operations. This guide gives a high-level overview of the console.

Objectives

The objective is to facilitate the student and the upcoming generation knowing about the best college or the university without going through many different websites to get everything on one platform.

Looking onto today's framework and the confusion raised in the mind of students regarding the best colleges or universities for their respective courses this chat-bot come website has been designed to overcome this problem or the confusion from the mind of the upcoming engineers, CA, and many more.

Another important objective to develop the College Enquiry Chatbot is to avoid going through many websites this will take the user directly to the right page or the official website of the college or university.

Purpose

As of now, we all know technology is rapidly increasing, and chatbots will only become more popular, as they can solve the problem much faster compared to a user. The chatbot is a way to communicate better between people and services, enhancing user experience. And simultaneously they interact more with the customer and give the companies new chances to enhance the user-friendly experience from the feedback and engagements provided by the customer to the chatbots.

The most common problems faced by a user is-

- Navigating things on a website is hard if you are new.
- Most of the time it happens we don't get answers to simple questions.
- Facing problem to contact to admin or getting address or phone numbers.

These problems can be easily resolved with the help of a chatbot, some of the main purposes of using chatbot are listed below-

- Getting relatively quick and on-the-point answers without wasting time.
- Solving a complaint faster as most of the time chatbot automatically contacts the admin regarding this.
- We get detailed explanations for our queries
- If you want to contact the admin, you can also do that using it as it will notify the admin much faster.

The chatbots ought to be not difficult to utilize, react in an opportune style, and be generally easy to understand. The bots should make the client's cooperation as simple and quick as conceivable to guarantee that the client's time isn't squandered and that they get what they need with no trouble or misjudging from the bot. The discussion should stream and consistently keep the client in charge of the discussion. Clients should leave their experience having gained the chatbot and think that it was a fun, simple to utilize, and clear connection that would urge them to return decisively.

With informing stages being the most utilized sort of use on the planet, organizations will be hoping to exploit this and begin to create their chatbots to work alongside their web-based media pages. For instance, an individual calling an eatery to perceive what time they open at for sure is unique today, the client can just message the page on Facebook and the bot will react likewise. This saves time for genuine representatives to accomplish other work and permits the chatbot to deal with straightforward undertakings. Since clients will as of now have an informing application introduced on their cell phone, there is no compelling reason to download a different application to utilize the chatbot. This can turn plenty of clients away as these days there is plenty of utilizations accessible and most clients will be tired of downloading an application that they may just utilize on more than one occasion.

Scope

The college inquiry chat-bot project is based on algorithms that analyze and interpret user queries and messages.

The program responds to the student's question with the assistance of algorithms.

To facilitate the student and the upcoming generation knowing about the best college or the university without going through many different websites getting everything on one platform.

Looking onto today's framework and the confusion raised in the mind of students regarding the best colleges or universities for their respective courses this chat-bot

come website has been designed to overcome this problem or the confusion from the mind of the upcoming engineers, CA, and many more.

Another important objective to develop the College Enquiry Chatbot is to avoid going through many websites this will take the user directly to the right page or the official website of the college or university.

Applicability

Some various types where a chatbot can apply are as follows -

- **Chatbots offers better Service**

  It allows users to interact more and is an all-in-one single interface where users can provide feedback, get help, contact admin, and much more.

- **24/7 support**

  Unlike human beings, a chatbot can work tirelessly and help a user regardless of time. It can be used at many places where a user can be tired but a chatbot can work.

- **Lower operational costs**

Deploying a chatbot is also cost-effective, mainly the cost of a chatbot is for maintenance only and if we use a human being for the work that a chatbot can also do it will be more expensive.

- **Better customer experience**

  AI-powered chatbots **are often** designed **to know** the intent of the customer and automatically trigger **the proper** conversational flow **to supply** a proactive and complete customer service experience.

Organization of Report

1. **Google Maps**
   - It is a web mapping platform where customer applications are offered by Google.
   - It offers satellite imagery, aerial photography, street maps, 360° interactive panoramic views of streets (Street View), real-time traffic conditions, and route planning for traveling by foot, car, air (in beta), and public transportation.

2. **Google Translator**

- It will help people of different regions, countries to get the information in their preferred language
- One of the bonuses of using Translate is that it not only shows the direct meaning but pulls out some other details as well
- This is a pretty useful feature and might help use a word more appropriately

3. **Chat-Bot**

- It also works on Google Assistant as it is kept in mind for the usage of clients
- It will Control your devices and your smart home.

- One can access information from the calendar and other personal information as well
- Find information online, from restaurant bookings to directions, weather, and news, Control your music, etc.
- content on your Chromecast or other compatible devices.
- Run timers and reminders

4. Website functionality

- One can use a website in case they using a desktop or laptop
- Online Presence 24/7
- the website helps customers to find anything, anytime and anywhere
- Information Exchange
- Credibility
- It Cuts Costs
- Market Expansion
- Consumer Insights
- Advertising
- Competitors Online

Chapter 2: System Analysis

Identification of Need

It is obvious from the examination completed in the writing survey that cutting-edge monetary administrations are continually trying to grow their advances, both to further develop user-friendly and increment conveyance of administrations through the progressions in innovation. This is to acquire a strategic advantage over different universities and colleges for monetary query and to extend its user's base. An area explicit chatbot will be carried out to help the user with their queries. To conquer the user fulfillment issues related to an update regarding admissions in the respective universities or colleges. The chatbot will give an individual and proficient correspondence between the user and their admin to deal with their questionaries and get help when required, for example, taking note of any inquiries and booking courses of action. The chatbot will permit the user to feel sure and agreeable when utilizing this help paying little heed to the user's PC education because of the normal language utilized in messages. It likewise gives an entirely open and productive help as all connections will occur inside the one talk discussion refuting the requirement for the user to explore through a site.

Preliminary Investigation

This segment will cover the foundation research that I have directed into chatbots and a portion of the advances we've investigated before settling down into the undertaking.

CHATBOT:

A chatbot is a computer-based intelligence specialist that can partake in a discussion with a client. Most are furnished with a courier-type interface with input from a client and a yield from the chatbot. The chatbot processes the client's feedback and yields an answer depending on what the client has quite recently sent. It very well may be a hello, discussion subject, or even a picture.

Most essential chatbots work by coordinating with a client's contribution with a predefined set of discourse.

For instance, a client saying "Bless your heart" will result in the chatbot saying "The pleasure is all mine". The predefined set of exchanges can be set up to mirror a typical discussion between two individuals. Issues can emerge when a client says something the chatbot doesn't perceive, a model could be the client significance to say "Thank you", yet rather saying "Much appreciated", this can befuddle the chatbot as it will be hoping to coordinate the "Bless your heart" contribution with "Welcome". This prompts a great deal of manual work by attempting to characterize each mix of a client saying "Much appreciated".

Present-day chatbots are more complicated and component normal language handling that can gain from client inputs. They can get to APIs to get data from clients like news, climate, time, and so forth They can even handle requests and make appointments altogether through a chatbot interface.

Chatbots are appropriate for cell phones as informing is at the core of a cell phone. Informing has made considerable progress since SMS messages became advocated in the 2000s and are presently on the decrease. From the years 2011-2015, the utilization of SMS in Ireland has dropped by 44%. 3 billion texts in 2011 contrasted with 1.7 billion texts in 2015.

[John Hargan, killbiller.com 2015] even though SMS is encountering a decrease, this doesn't imply that individuals aren't sending messages any longer, it simply implies they are utilizing various administrations. It is quickly turning into the standard where a chatbot is simpler to use than an application, and organizations are exploiting this.

API:

API is the acronym for exercise Programming Interface, which is a software buffer that allows two exercises to orate to each different. Each moment you employ an app like Facebook, ship an immediate communication or agree on the rainfall on your phone, you're operating an API.

Over the years, "APIs" often describe any type of generic connection interface to an application. However, these days, modern APIs employ some properties that are very valuable and useful:

- The latest APIs are easy for developers to use, easily accessible, and comply with commonly understood standards (usually HTTP and REST)
- . They are treated like products, not codes. They are designed for use by specific target groups (such as mobile developers) and are documented and versioned to give users certain expectations regarding maintenance and lifecycle.
- They are much more standardized, so there is much more discipline regarding security and governance, as well as performance and scalability monitoring and management
- Like any productive software, modern APIs have their own software development life cycle (SDLC) for design, testing, building, managing, and versioning. The latest API for
- consumption and version control is also well documented.

AIML:

Alice, short for Counterfeit Phonetic Web PC Element, is a chatbot created by Richard Wallace utilizing AIML (Man-made reasoning Markup Language).

At the point when I was first exploring chatbots and how to foster them I went over AIML (Man-made reasoning Markup Language). It's an XML pattern used to make shrewd chatbots. One of the world's most known chatbots; A.L.I.C.E, short for Counterfeit Phonetic

Web PC Element was created utilizing AIML. Right away, it was by all accounts the best system to foster my task on. There were loads of material online for me to peruse and will see how everything functioned. I began work on my undertaking utilizing this language and gained some headway however deplorably, it simply wasn't going to work out. It was an old language at first delivered in 2001 and wasn't truly adaptable. Essential discussions were not difficult to get going however getting to APIs and getting them facilitated via web-based media stages caused extraordinary trouble. Its antiquated design didn't face the present current stages. I rejected all of my work on AIML and moved to Microsoft Bot Structure which I stayed with for the rest of the improvement of the task. Looking back, I should've continued before.

FRONT END:

The part of the website that users interact with directly is called the front end. This is also known as the client-side of the application. It includes everything that you experience directly, including text colors and styles, images, graphics and tables, buttons, colors, navigation menus, and more. HTML, CSS, and JavaScript are the languages used for front-end development. The structure, design, behavior, and content of what you see on your browser screen when you open a website, web application, or mobile app is implemented by the front-end developer. Responsiveness and performance are the two main goals of the front end. Developers need to make sure that their website is responsive. NS. It will display

correctly on devices of all sizes. No part of the website should behave abnormally, regardless of screen size.

Some Frontend languages are:

- HTML: HTML stands for Hypertext Markup Language. It is used to design the front-end part of a website using the markup language. HTML is a combination of hypertext and markup languages. Hypertext defines links between web pages. Markup languages are used to define text documents within tags that define the structure of web pages.
- CSS: Lovingly known as CSS, Cascade Stylesheets is a simply designed language aimed at simplifying the process of displaying web pages. You can use CSS to apply styles to your web pages. More importantly, you can do this using CSS, regardless of the HTML of each web page.
- JavaScript is a well-known scripting language used to spell websites to make them interactive to users. It is used to improve the functionality of websites and run cool games and web-based software.

BACKEND:

The back end is also known as the server-side of the website. Save and organize your data and make sure everything is working properly on the client-side of your website. It's that part of the website that you can't see or interact with. This is part of the software that does not come into direct contact with the user. Parts and features developed by the back-end designer are not directly accessed by the user through the front-end application. Activities such as creating APIs, creating libraries, and working with system components that do not use the user interface or scientific programming system are also included in the backend.

Some Backend languages are:

- PHP: PHP is a server-side scripting language specially designed for web development. PHP code runs on the server-side, so it is known as a server-side scripting language.
- C ++: This is a universal programming language and is now widely used in competitive programming. It is also known as a back-end language.
- Java: Java is one of the most famous and widely used programming languages and platforms. It's very scalable. Java components are always ready to use.

Python: Python is a programming language that allows you to work quickly and integrate your system more efficiently.

JavaScript can be used as both a front-end and back-end programming language.

- Node.js: Node.js is an open-source, cross-platform runtime environment for executing JavaScript code outside the browser. It's important to remember that NodeJS is not a framework or programming language. Most people are confused and understand that it is a framework or programming language. We often use Node.js to create back-end services such as APIs for web and mobile apps. Used in the production of large companies such as Paypal, Uber, Netflix, and Walmart.

Feasibility Study

- To eliminate this problem of confusion in the minds of upcoming engineers, CAs, and many more, this website has been designed to act like a chatbot.
- By getting everything on one website, you won't need to visit many different websites.

- The user will be taken directly to the university or college's official website so that they won't have to navigate through several different websites.
- To eliminate this problem of confusion in the minds of upcoming engineers, CAs, and many more, this website has been designed to act like a chatbot.
- By getting everything on one website, you won't need to visit many different websites.
- The user will be taken directly to the university or college's official website so that they won't have to navigate through several different websites
- It is easy to use no prior knowledge is required
- Interactive User Interface
- Smooth functioning with woe.
- 24hrs service
- Good customer service
- Answer to most commonly asked questions
- Getting instant responses
- Friendliness and approach-ability
- Easy communication

Project Planning

In the 21st century as we can see the competition has increased in all the sectors – private or public sector as many students get confused about the best choice of colleges and universities. This project will resolve issues they face and give a better solution.

• It includes – campus, placements, fees, etc.

• The chatbot will include the top–best colleges and universities in India.

• We have built a website that will support this chatbot.

• Saves time – search for the college and get all the information you need in one place

• Easy to operate – Chatbot is there for you for any queries.

• Minor details about the colleges are included

• If someone asks for some detailed info about any colleges they are provided by the official links of particular colleges.

Project Scheduling (PERT Chart and Gantt Chart Both)

S.No.	Task Name	Duration (Days)	Week										
			1	2	3	4	5	6	7	8	9	10	11
1	Strategy	7											
2	Analysis & Planning	7											
3	UI/UX	15											
4	Information Architecture & Workflow	7											
5	Software Development	40											
6	Testing	7											
7	Modifications	4											
8	Deployment and Finalizing	4											

Figure 1 Gantt Chart

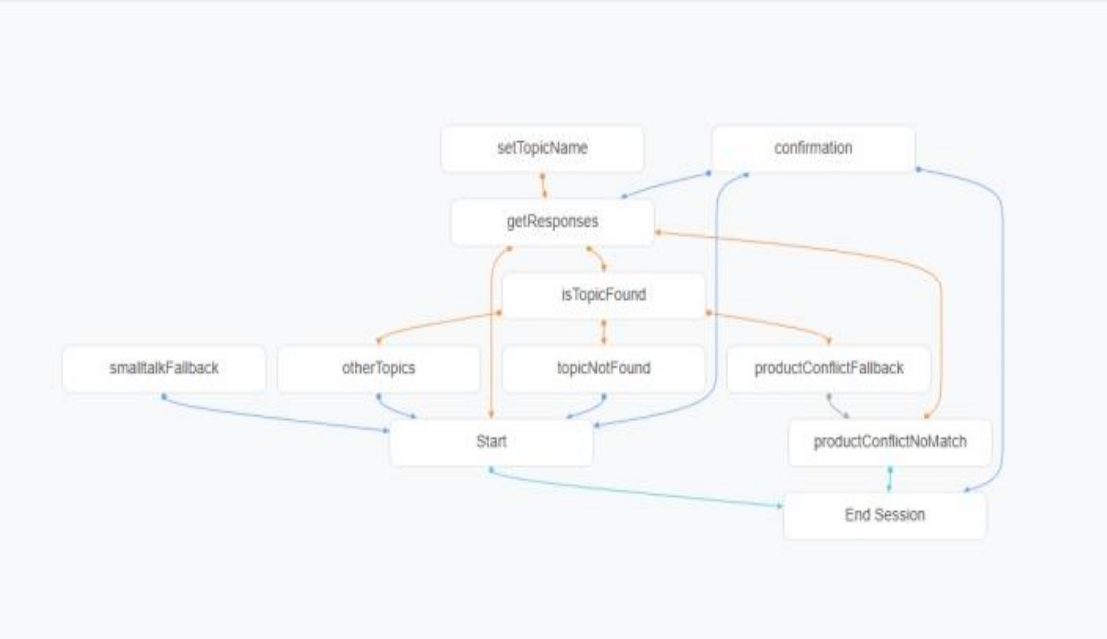

Figure 2 PERT Chart

OBJECTIVE	Days Required (maximum)
Strategy	7 days
Analysis and planning	7 days
UI/UX design	15 days
Information Architecture and Workflow	7 days
Software development (coding, designing)	40 days
Testing (website and chatbot combined working)	7 days
Modifications (if needed)	4 days
Deployment and finalizing	4 days
TOTAL DAYS	91 days

Table 1 Project Scheduling

Software Requirement Specifications (SRS)

Introduction & Purpose:

This document will provide all of the wants for the project Drexel Chabot.

It will function as a reference for developers and customers during the event of the final version of the system.

Overall Description:

Most of the search engines today, like Google, use a system to rank different sites. When a user enters a question, the query is interpreted as keywords and therefore the system returns an inventory of the highest-ranked sites which may have the solution to the query. Then the user must undergo the list of web pages to find the solution they're trying to find. Drexel Chabot, however, will attempt to understand the query and supply a definitive answer.

There will be four main units to the system working together to know the

question and return an appropriate answer:

● Generic question construction - capable of taking a tongue question and making it more generic.

● Generic answer construction - capable of taking a generic question and providing a generic answer.

● Generic answer popular - capable of taking an answer

and populating it with information from the database.

● Information extraction - capable of extracting information through structured

or unstructured websites and storing that information during a database.

User Needs:

The two classes of users for this technique are described below:

a) **API users**
API users contain application developers who want to include Drexel Chabot API into other software applications.

b) **Mobile app/Web app/SMS users**
These users contain non-technical users who want to urge answers to their questions. These users ask questions and obtain answers with mobile, web, or text messaging interfaces. This class of users includes Drexel's current and expected students, teaching faculty, and staff.

Assumptions & Dependencies: -

Keras maybe a library for creating and using neural networks. It should provide us with all the functionality we'd like, however, if it's in some way deficient, then it'll get replaced with a special library. Beautiful Soup maybe a library for parsing HTML documents. It should be all we'd like to extract text from a webpage, but could also be replaced if necessary. We'll develop the project using Python and MySQL databases.

Software Engineering Paradigm Applied

Investigation (Analysis)

This part will incorporate different programming strategies; quantifiable criticism will be gathered from surveys which will be dissected to recognize the prerequisites of the chatbot. The inward and outside equipment and framework prerequisites are

illustrated. A Gantt diagram will be given laying out the venture plan and when every achievement ought to be met.

Plan (Design)

This part will cover the general plan of the chatbot; UI charts/storyboards will be incorporated which detail how the GUI will appear to the user. The specialized engineering will be planned and shown in a graphical structure, for example, an action chart. It will lay out how every part cooperates with the other. The plan for the information base and the connections between tables will be given.

Execution and testing (Implementation and testing)

The segment will report the execution cycle to foster the arrangement, including; the code used to foster convincing components. Characterize what was realized during every cycle and report every model. When execution is finished the chatbot will be tried utilizing unit testing and fitting experiments. Any blunders or blames recognized will be fixed to guarantee the most extreme quality.

Assessment and Reflection (Evaluation and Reflection)

An assessment will be carried on the finished venture to decide its quality and worth. It will detail how the general venture was created. Reflecting upon the general insight, featuring any regions that could be improved for future work and what well all through the undertaking.

Data Models

(Like DFD, Control Flow diagrams, State Diagrams/ Sequence diagrams, Entity Relationship Model, Class Diagrams/ CRC Models/ Collaboration Diagrams/Use-case Diagrams/Activity) Diagrams depending upon our project requirements

Level 0 DFD

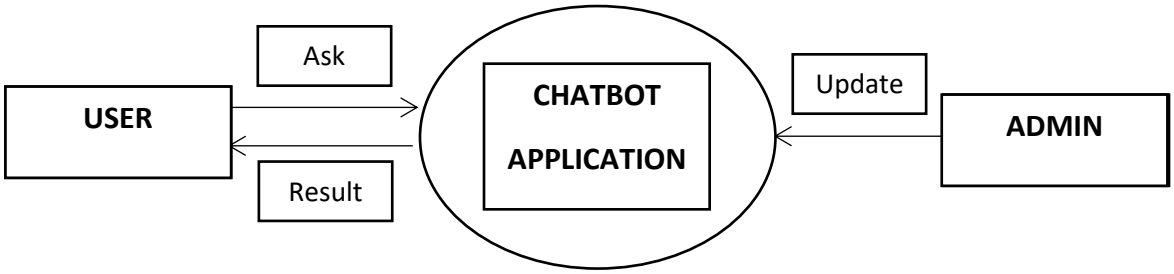

Figure 3 Level 0 DFD

Level 1 DFD

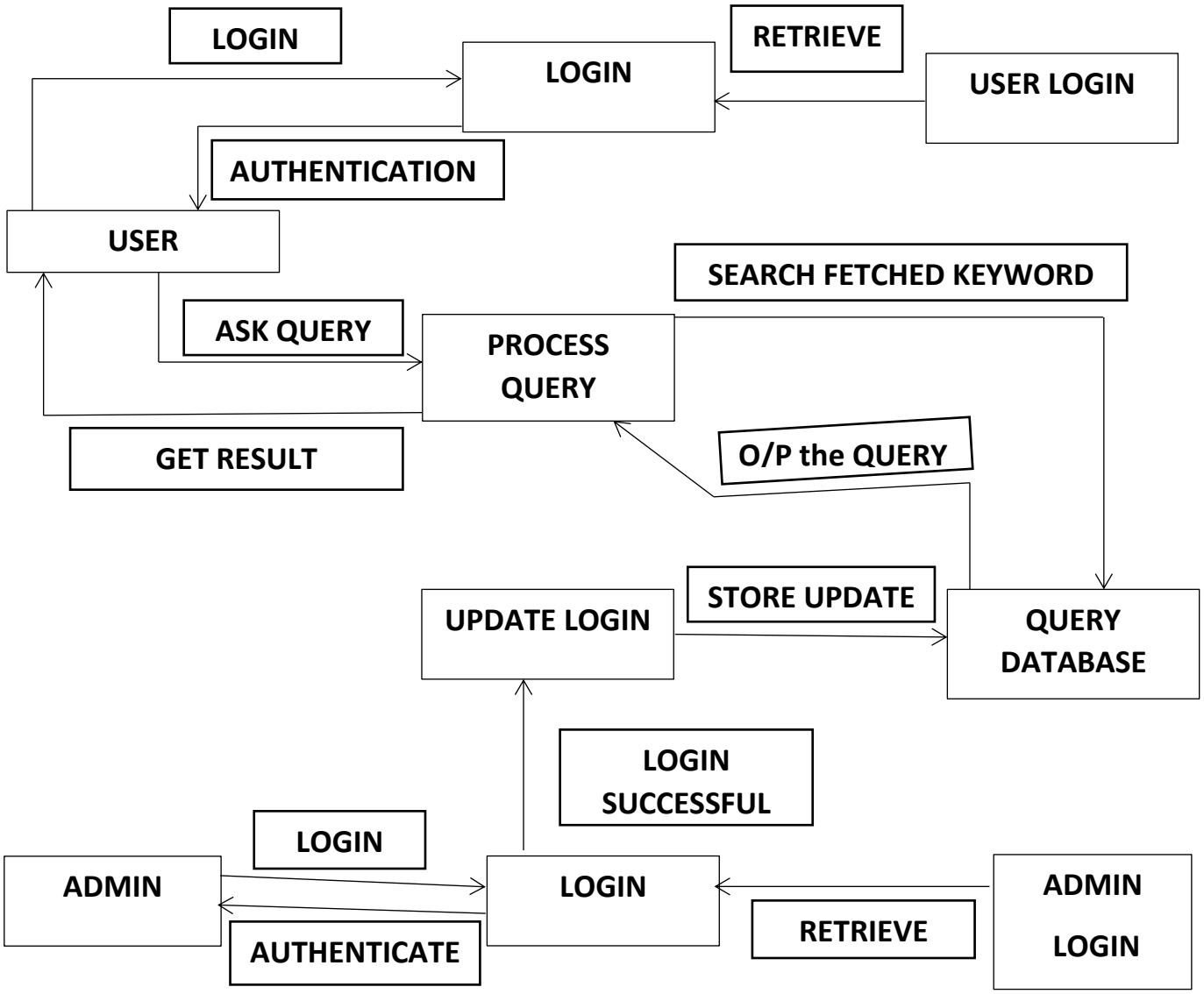

Figure 4 Level 1 DFD

Level 2 DFD

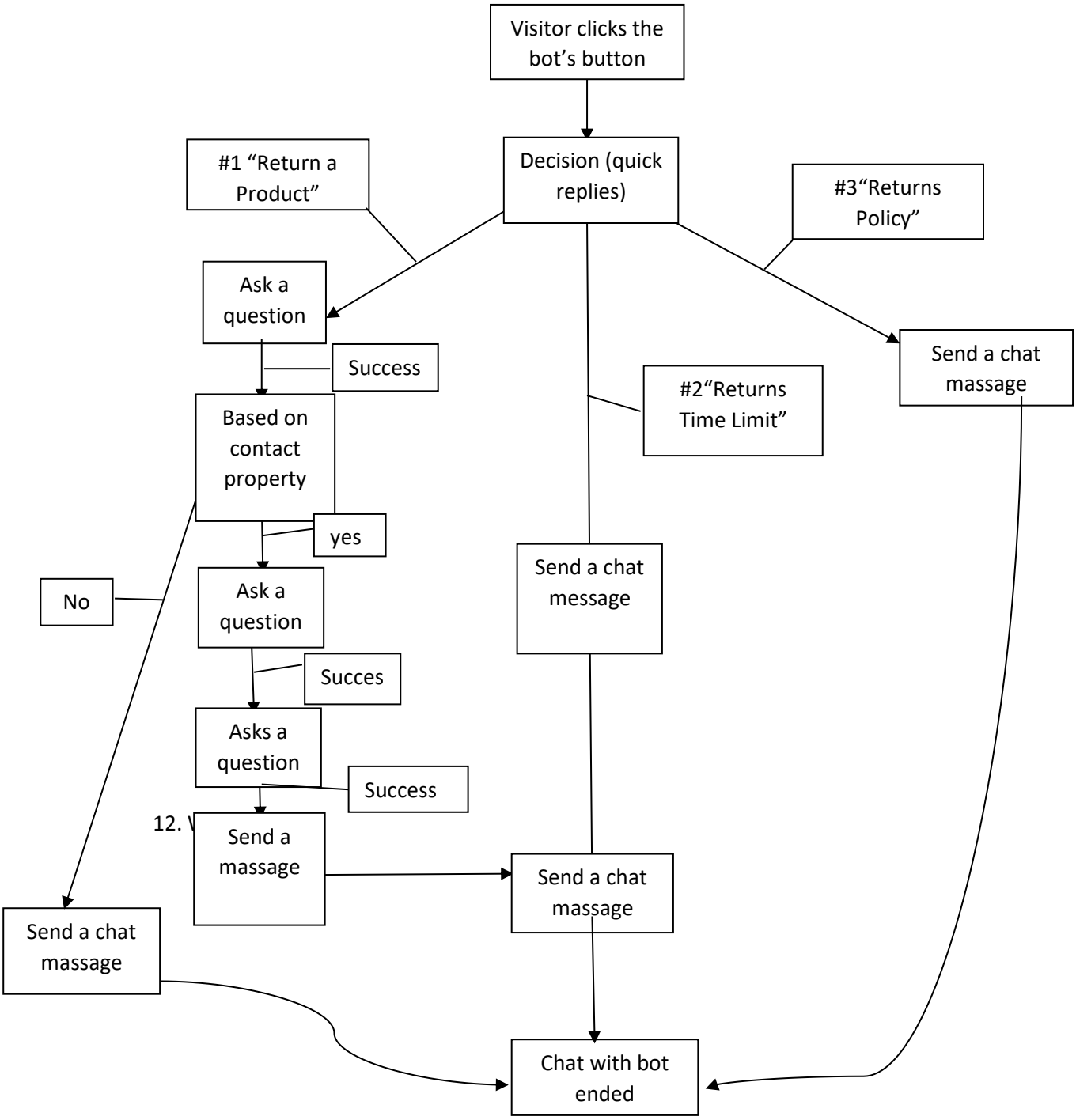

Figure 5 Level 2 DFD

ER DIAGRAM

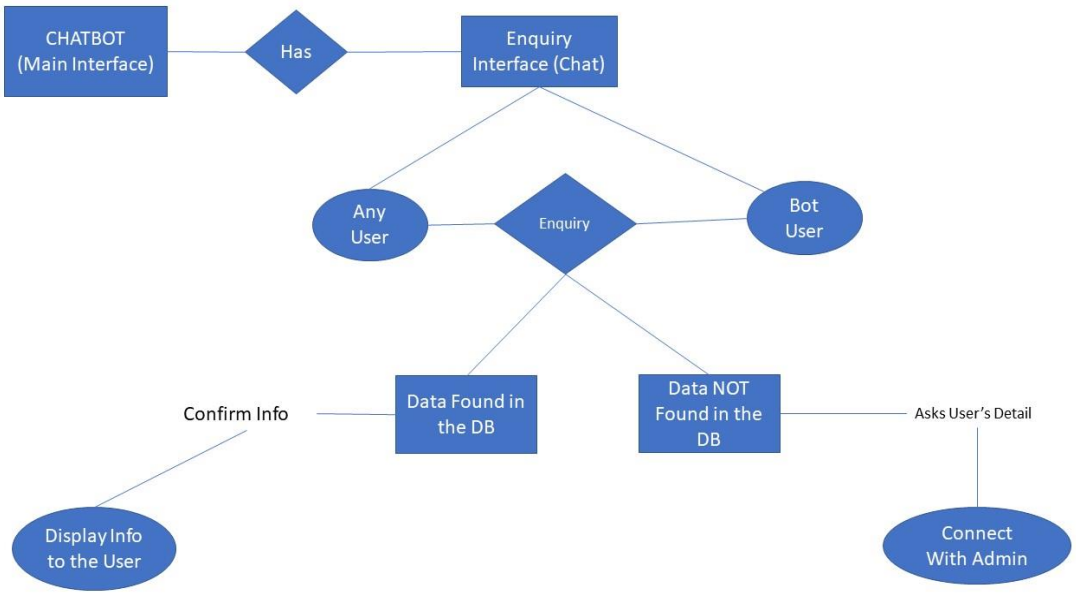

Figure 6 ER Diagram

The Chatbot is that system that guides you to the right path with the basic information as per the user requirement. The ER diagram represents the working algorithm of Chatbot.

The chatbot is the main Interface that carries the Enquiry Interface (Chat) from the user and checks whether the inquiry asked by the user matches the database or not. If the information is true and matches the database then the bot will display the information to the user else the bot asks for the user data which includes name, contact, email, etc., so that the admin can get in touch with the user over the platforms to clear the doubt and also to give the information regarding the Enquiry asked.

Chapter 3: System Design

Modularisation Details

1. **Chat User (User-Server)**:
   The suggested system is built on a user-server model. On the central server, all the data will be stored in an efficient database. Users can obtain this information using an Android application loaded on their cellphones (user machines). The user interface on each user PC will be upgraded.

2. **Chatbot**:
   A chatbot is a piece of software that allows users to access material and services through natural conversations. Chatbots are often in the form of a chat user that uses natural language processing to engage the user in a conversation. Chatbots direct conversation flow depending on the context of the user's request and respond with natural language words to deliver direct responses, request additional information or suggest other actions.

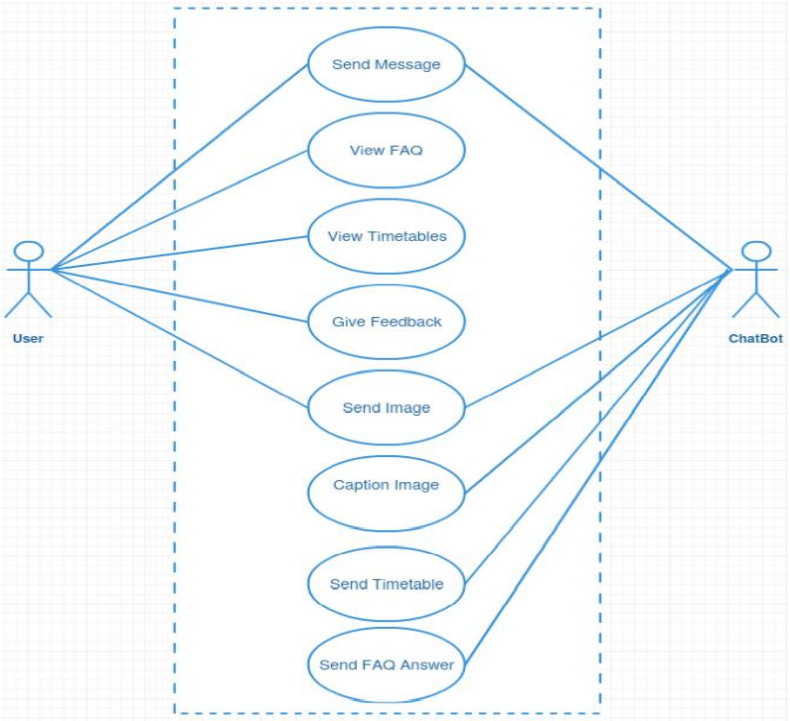

Figure 7 Working of CHATBOT

3. **Pattern matching**:

   A bot sends a comparison query to a machine. When a query matches a database, it is sent to data services.

4. **Data Services**:

   A machine receives a comparison question from a bot. A query is delivered to data services when it matches a database. As a result, all the modules stated above are completed in polynomial time sect, making this a P issue.

5. **UI:**

   The process of designing user interfaces in software or digital devices with a focus on appearance or style is known as user interface (UI) design. Designers try hard to design user interfaces that are more user-friendly and enjoyable to use. UI design encompasses both graphical and non-graphical user interfaces, such as voice-controlled interfaces.

Data Integrity and Constraint

It's easy to see why chatbots are the hottest new digital technology. Chatbots enable brands to communicate with customers more directly and frequently in ways that feel more real and personal by providing a user-friendly conversational experience across numerous messaging channels. As a result, chatbots are on the verge of disrupting several industries, including retail banking.

While it's true that security is crucial for any technology, most of the fear around chatbot security is overstated. Chatbots may be the most recent technology in terms of

user experience, but they run on basic, secure Internet protocols that have been in use for decades.

No matter how much advance a chatbot becomes but there are some limitations to it -

- Chatbots don't fully understand human language

  Although AI-powered smart-bots can comprehend the broad context, 40 out of 100 incidents are unrelated to it.

- They can't make their own decision

  Another drawback of chatbots is that they are incapable of making decisions. They lack the necessary expertise to distinguish between the excellent and the bad.

- Chatbots does not have emotions

  Without sentiment analysis information, chatbots will interact with customers in a specific way, regardless of the chat flow. As a result, some customers would rather end the conversation!

Database Design, Procedural Design/Object-Oriented Design

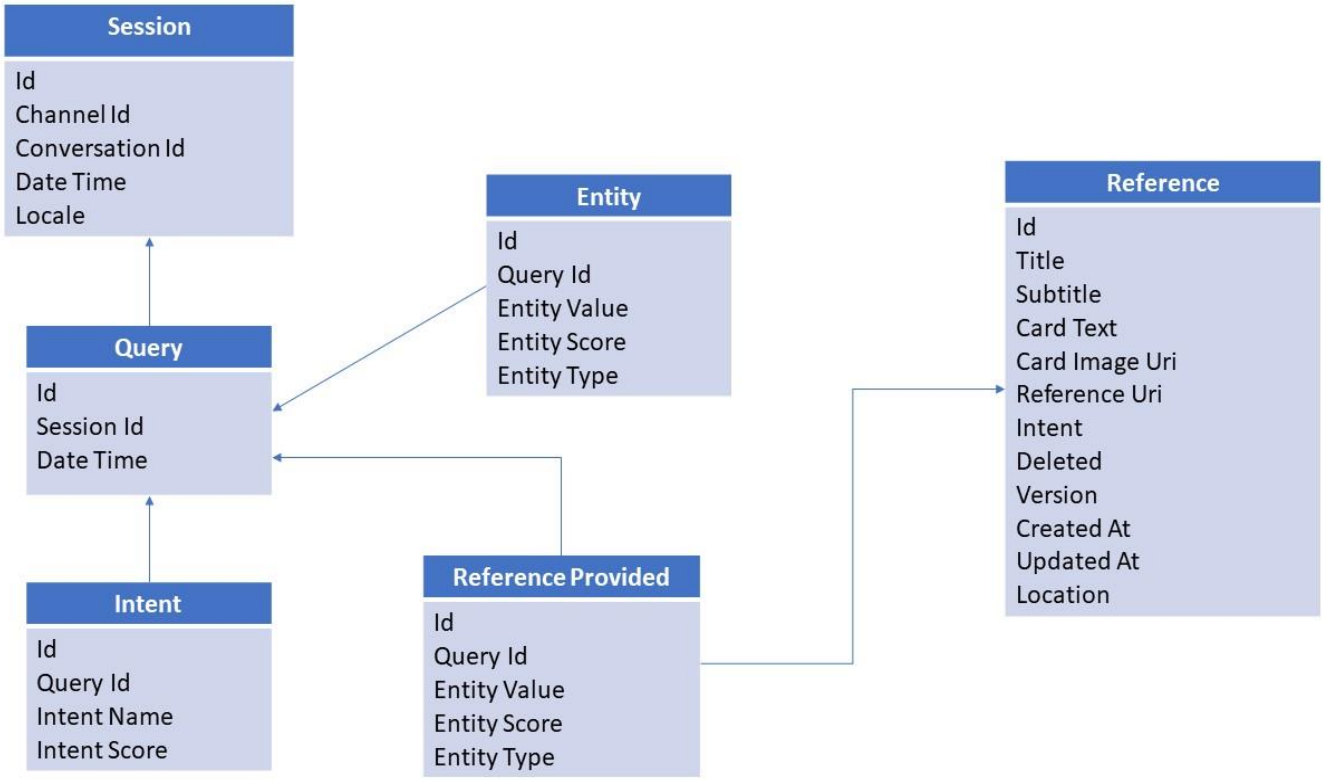

Figure 8 Database Design

User Interface Design

S. No.	Principal	Description
1.	Design by personality	The bot character ought to give a characteristic and remarkable intuitive experience for clients. The character ought to be intended to direct the clients through the communication which might incorporate

		monitoring what the client might say or need. The character is the style and way the bot holds all through the connection.
2.	Flexibility in response	Adaptable reactions ought to be given to client input, the bot ought to have different/arbitrary reactions to similar client input.
3.	Text vs Custom Buttons	It ought to be obvious to the client what information is generally anticipated to carry on in the discussion. A fallback reaction ought to be utilized in circumstances where the info isn't perceived, for example, proposing different tasks that are accessible. Rather than exclusively depending on normal text orders from the client, a decent UI should deliver organized choices to the client in the type of buttons or text.
4.	Simplicity in Interaction	Be brief and clear in light of inquiries. The discussion stream ought to be straight and not branch out into complex ways.
5.	Conversation Flow	The discussion stream is imperative to guarantee that the

		client isn't disturbed in circumstances where the bot will most likely be unable to give a proper answer right away. The client would then need to go through a discussion to decide the solution to their question.
6.	Tasks and duty specifications	The bot ought to convey unequivocal errands for a specific space.
7.	Rigid syntax, then NLP	Individuals are inclined to spell missteps or utilize conversational expressions which may not be perceived by the bot.
8.	Empathy and emotional state	It is significant that the bot fabricates compatibility with the client and pass on feeling in reactions dependent on specific client input.
9.	Keep conversation short	Clients need to connect with the bot to get answers or arrive at an objective rapidly. Stay away from wordy discussions as it will cause the collaboration to feel difficult. This stays away from equivocalness.

10.	Triggers and actions	The bot needs to convince clients and look for ways of empowering clients to bring out explicit activities through the bot.
11.	Predict and personalize	Bots can dissect past input from clients for significant boundaries, to expect a client's inclination. It will then, at that point, give an answer that would be extraordinary to that specific client.
12.	Fully/partially automated	NLP ought to be utilized for computerized assignments.
13.	Providing a way out	Permit the client to start the discussion again or leave the discussion.
14.	User boredom	The discussion ought to draw in the client all through the connection.

Figure 9 User Interface Design

Test Cases

- **Functionality Testing**

  Verified that there is no dead page or invalid redirects.

  Verified the working of the system.

  Verified data safety.

- **Interface testing**

  Performed to verify the interface and the data flow from one system to another like different windows and phones.

- **Performance testing**

  Load testing-checked by opening all the posts at the same time and it's working.

  Stress testing-checked the limits of the website till which it can handle stress.

Chapter 4: Coding

Complete Project Coding

```
import sys
_b=sys.version_info[0]<3 and (lambda x:x) or (lambda x:x.encode('latin1'))
from google.protobuf import descriptor as _descriptor
from google.protobuf import message as _message
from google.protobuf import reflection as _reflection
from google.protobuf import symbol_database as _symbol_database
from google.protobuf import descriptor_pb2
# @@protoc_insertion_point(imports)

_sym_db = _symbol_database.Default()


from google.api import annotations_pb2 as google_dot_api_dot_annotations__pb2
from google.rpc import status_pb2 as google_dot_rpc_dot_status__pb2


DESCRIPTOR = _descriptor.FileDescriptor(
  name='google/assistant/embedded/v1alpha1/embedded_assistant.proto',
```

```
  package='google.assistant.embedded.v1alpha1',

  syntax='proto3',


serialized_pb=_b('\n;google/assistant/embedded/v1alpha1/embedded_assistant.proto\x
12\"google.assistant.embedded.v1alpha1\x1a\x1cgoogle/api/annotations.proto\x1a\x1
7google/rpc/status.proto\"\xbe\x02\n\x0e\x43onverseConfig\x12J\n\x0f\x61udio_in_c
onfig\x18\x01
\x01(\x0b\x32\x31.google.assistant.embedded.v1alpha1.AudioInConfig\x12L\n\x10\x
61udio_out_config\x18\x02
\x01(\x0b\x32\x32.google.assistant.embedded.v1alpha1.AudioOutConfig\x12I\n\x0e\
x63onverse_state\x18\x03
\x01(\x0b\x32\x31.google.assistant.embedded.v1alpha1.ConverseState\x12G\n\rdevic
e_config\x18\x04
\x01(\x0b\x32\x30.google.assistant.embedded.v1alpha1.DeviceConfig\"\xb6\x01\n\rA
udioInConfig\x12L\n\x08\x65ncoding\x18\x01
\x01(\x0e\x32:.google.assistant.embedded.v1alpha1.AudioInConfig.Encoding\x12\x1
9\n\x11sample_rate_hertz\x18\x02
\x01(\x05\"<\n\x08\x45ncoding\x12\x18\n\x14\x45NCODING_UNSPECIFIED\x10\
x00\x12\x0c\n\x08LINEAR16\x10\x01\x12\x08\n\x04\x46LAC\x10\x02\"\xe3\x01\n\
x0e\x41udioOutConfig\x12M\n\x08\x65ncoding\x18\x01
\x01(\x0e\x32;.google.assistant.embedded.v1alpha1.AudioOutConfig.Encoding\x12\x
19\n\x11sample_rate_hertz\x18\x02
\x01(\x05\x12\x19\n\x11volume_percentage\x18\x03
\x01(\x05\"L\n\x08\x45ncoding\x12\x18\n\x14\x45NCODING_UNSPECIFIED\x10\
x00\x12\x0c\n\x08LINEAR16\x10\x01\x12\x07\n\x03MP3\x10\x02\x12\x0f\n\x0bO
PUS_IN_OGG\x10\x03\"+\n\rConverseState\x12\x1a\n\x12\x63onversation_state\x1
8\x01 \x01(\x0c\"\x1e\n\x08\x41udioOut\x12\x12\n\naudio_data\x18\x01
\x01(\x0c\"\xbd\x02\n\x0e\x43onverseResult\x12\x1b\n\x13spoken_request_text\x18\
x01 \x01(\t\x12\x1c\n\x14spoken_response_text\x18\x02
\x01(\t\x12\x1a\n\x12\x63onversation_state\x18\x03
\x01(\x0c\x12Z\n\x0fmicrophone_mode\x18\x04
```

\x01(\x0e\x32\x41.google.assistant.embedded.v1alpha1.ConverseResult.MicrophoneMode\x12\x19\n\x11volume_percentage\x18\x05
\x01(\x05\"]\n\x0eMicrophoneMode\x12\x1f\n\x1bMICROPHONE_MODE_UNSPECIFIED\x10\x00\x12\x14\n\x10\x43LOSE_MICROPHONE\x10\x01\x12\x14\n\x10\x44IALOG_FOLLOW_ON\x10\x02\"\x7f\n\x0f\x43onverseRequest\x12\x44\n\x06\x63onfig\x18\x01
\x01(\x0b\x32\x32.google.assistant.embedded.v1alpha1.ConverseConfigH\x00\x12\x12\n\x08\x61udio_in\x18\x02
\x01(\x0cH\x00\x42\x12\n\x10\x63onverse_request\"\xb5\x03\n\x10\x43onverseResponse\x12#\n\x05\x65rror\x18\x01
\x01(\x0b\x32\x12.google.rpc.StatusH\x00\x12T\n\nevent_type\x18\x02
\x01(\x0e\x32>.google.assistant.embedded.v1alpha1.ConverseResponse.EventTypeH\x00\x12\x41\n\taudio_out\x18\x03
\x01(\x0b\x32,.google.assistant.embedded.v1alpha1.AudioOutH\x00\x12I\n\rdevice_action\x18\t
\x01(\x0b\x32\x30.google.assistant.embedded.v1alpha1.DeviceActionH\x00\x12\x44\n\x06result\x18\x05
\x01(\x0b\x32\x32.google.assistant.embedded.v1alpha1.ConverseResultH\x00\"=\n\tEventType\x12\x1a\n\x16\x45VENT_TYPE_UNSPECIFIED\x10\x00\x12\x14\n\x10\x45ND_OF_UTTERANCE\x10\x01\x42\x13\n\x11\x63onverse_response\":\n\x0c\x44\x65viceConfig\x12\x11\n\tdevice_id\x18\x01
\x01(\t\x12\x17\n\x0f\x64\x65vice_model_id\x18\x03
\x01(\t\"+\n\x0c\x44\x65viceAction\x12\x1b\n\x13\x64\x65vice_request_json\x18\x02
\x01(\t2\x8e\x01\n\x11\x45mbeddedAssistant\x12y\n\x08\x43onverse\x12\x33.google.assistant.embedded.v1alpha1.ConverseRequest\x1a\x34.google.assistant.embedded.v1alpha1.ConverseResponse(\x01\x30\x01\x42\x86\x01\n&com.google.assistant.embedded.v1alpha1B\x0e\x41ssistantProtoP\x01ZJgoogle.golang.org/genproto/googleapis/assistant/embedded/v1alpha1;embeddedb\x06proto3')

,

```
dependencies=[google_dot_api_dot_annotations_pb2.DESCRIPTOR,google_dot_rpc
_dot_status_pb2.DESCRIPTOR,])



_AUDIOINCONFIG_ENCODING = _descriptor.EnumDescriptor(
  name='Encoding',
  full_name='google.assistant.embedded.v1alpha1.AudioInConfig.Encoding',
  filename=None,
  file=DESCRIPTOR,
  values=[
    _descriptor.EnumValueDescriptor(
      name='ENCODING_UNSPECIFIED', index=0, number=0,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='LINEAR16', index=1, number=1,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='FLAC', index=2, number=2,
      options=None,
      type=None),
```

```
  ],
  containing_type=None,
  options=None,
  serialized_start=598,
  serialized_end=658,
)
_sym_db.RegisterEnumDescriptor(_AUDIOINCONFIG_ENCODING)


_AUDIOOUTCONFIG_ENCODING = _descriptor.EnumDescriptor(
  name='Encoding',
  full_name='google.assistant.embedded.v1alpha1.AudioOutConfig.Encoding',
  filename=None,
  file=DESCRIPTOR,
  values=[
    _descriptor.EnumValueDescriptor(
      name='ENCODING_UNSPECIFIED', index=0, number=0,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='LINEAR16', index=1, number=1,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
```

```
      name='MP3', index=2, number=2,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='OPUS_IN_OGG', index=3, number=3,
      options=None,
      type=None),
  ],
  containing_type=None,
  options=None,
  serialized_start=812,
  serialized_end=888,
)
_sym_db.RegisterEnumDescriptor(_AUDIOOUTCONFIG_ENCODING)


_CONVERSERESULT_MICROPHONEMODE = _descriptor.EnumDescriptor(
  name='MicrophoneMode',
  full_name='google.assistant.embedded.v1alpha1.ConverseResult.MicrophoneMode',
  filename=None,
  file=DESCRIPTOR,
  values=[
    _descriptor.EnumValueDescriptor(
      name='MICROPHONE_MODE_UNSPECIFIED', index=0, number=0,
```

```
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='CLOSE_MICROPHONE', index=1, number=1,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='DIALOG_FOLLOW_ON', index=2, number=2,
      options=None,
      type=None),
  ],
  containing_type=None,
  options=None,
  serialized_start=1192,
  serialized_end=1285,
)
_sym_db.RegisterEnumDescriptor(_CONVERSERESULT_MICROPHONEMODE)


_CONVERSERESPONSE_EVENTTYPE = _descriptor.EnumDescriptor(
  name='EventType',
  full_name='google.assistant.embedded.v1alpha1.ConverseResponse.EventType',
  filename=None,
  file=DESCRIPTOR,
```

```
    values=[
      _descriptor.EnumValueDescriptor(
        name='EVENT_TYPE_UNSPECIFIED', index=0, number=0,
        options=None,
        type=None),
      _descriptor.EnumValueDescriptor(
        name='END_OF_UTTERANCE', index=1, number=1,
        options=None,
        type=None),
    ],
    containing_type=None,
    options=None,
    serialized_start=1772,
    serialized_end=1833,
)
_sym_db.RegisterEnumDescriptor(_CONVERSERESPONSE_EVENTTYPE)



_CONVERSECONFIG = _descriptor.Descriptor(
    name='ConverseConfig',
    full_name='google.assistant.embedded.v1alpha1.ConverseConfig',
    filename=None,
    file=DESCRIPTOR,
```

```
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='audio_in_config',
full_name='google.assistant.embedded.v1alpha1.ConverseConfig.audio_in_config',
index=0,
      number=1, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='audio_out_config',
full_name='google.assistant.embedded.v1alpha1.ConverseConfig.audio_out_config',
index=1,
      number=2, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='converse_state',
full_name='google.assistant.embedded.v1alpha1.ConverseConfig.converse_state',
index=2,
      number=3, type=11, cpp_type=10, label=1,
```

```
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='device_config',
full_name='google.assistant.embedded.v1alpha1.ConverseConfig.device_config',
index=3,
      number=4, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
```

```
  ],
  serialized_start=155,
  serialized_end=473,
)



_AUDIOINCONFIG = _descriptor.Descriptor(
  name='AudioInConfig',
  full_name='google.assistant.embedded.v1alpha1.AudioInConfig',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='encoding',
full_name='google.assistant.embedded.v1alpha1.AudioInConfig.encoding', index=0,
      number=1, type=14, cpp_type=8, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
```

```
      name='sample_rate_hertz',
full_name='google.assistant.embedded.v1alpha1.AudioInConfig.sample_rate_hertz',
index=1,
      number=2, type=5, cpp_type=1, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
    _AUDIOINCONFIG_ENCODING,
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=476,
  serialized_end=658,
)
```

```
_AUDIOOUTCONFIG = _descriptor.Descriptor(
  name='AudioOutConfig',
  full_name='google.assistant.embedded.v1alpha1.AudioOutConfig',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='encoding',
full_name='google.assistant.embedded.v1alpha1.AudioOutConfig.encoding', index=0,
      number=1, type=14, cpp_type=8, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='sample_rate_hertz',
full_name='google.assistant.embedded.v1alpha1.AudioOutConfig.sample_rate_hertz',
index=1,
      number=2, type=5, cpp_type=1, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
```

```
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='volume_percentage',
full_name='google.assistant.embedded.v1alpha1.AudioOutConfig.volume_percentage
', index=2,
      number=3, type=5, cpp_type=1, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
    _AUDIOOUTCONFIG_ENCODING,
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
```

```
  serialized_start=661,
  serialized_end=888,
)



_CONVERSESTATE = _descriptor.Descriptor(
  name='ConverseState',
  full_name='google.assistant.embedded.v1alpha1.ConverseState',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='conversation_state',
full_name='google.assistant.embedded.v1alpha1.ConverseState.conversation_state',
index=0,
      number=1, type=12, cpp_type=9, label=1,
      has_default_value=False, default_value=_b(""),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
  ],
  extensions=[
  ],
```

```
    nested_types=[],
    enum_types=[
    ],
    options=None,
    is_extendable=False,
    syntax='proto3',
    extension_ranges=[],
    oneofs=[
    ],
    serialized_start=890,
    serialized_end=933,
)



_AUDIOOUT = _descriptor.Descriptor(
    name='AudioOut',
    full_name='google.assistant.embedded.v1alpha1.AudioOut',
    filename=None,
    file=DESCRIPTOR,
    containing_type=None,
    fields=[
      _descriptor.FieldDescriptor(
```

```
      name='audio_data',
full_name='google.assistant.embedded.v1alpha1.AudioOut.audio_data', index=0,
      number=1, type=12, cpp_type=9, label=1,
      has_default_value=False, default_value=_b(""),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=935,
  serialized_end=965,
)
```

```
_CONVERSERESULT = _descriptor.Descriptor(
  name='ConverseResult',
  full_name='google.assistant.embedded.v1alpha1.ConverseResult',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='spoken_request_text',
full_name='google.assistant.embedded.v1alpha1.ConverseResult.spoken_request_text
', index=0,
      number=1, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='spoken_response_text',
full_name='google.assistant.embedded.v1alpha1.ConverseResult.spoken_response_te
xt', index=1,
      number=2, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
```

```
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='conversation_state',
full_name='google.assistant.embedded.v1alpha1.ConverseResult.conversation_state',
index=2,
      number=3, type=12, cpp_type=9, label=1,
      has_default_value=False, default_value=_b(""),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='microphone_mode',
full_name='google.assistant.embedded.v1alpha1.ConverseResult.microphone_mode',
index=3,
      number=4, type=14, cpp_type=8, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='volume_percentage',
full_name='google.assistant.embedded.v1alpha1.ConverseResult.volume_percentage',
index=4,
      number=5, type=5, cpp_type=1, label=1,
```

```
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
    _CONVERSERESULT_MICROPHONEMODE,
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=968,
  serialized_end=1285,
)



_CONVERSEREQUEST = _descriptor.Descriptor(
```

```
  name='ConverseRequest',
  full_name='google.assistant.embedded.v1alpha1.ConverseRequest',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='config',
full_name='google.assistant.embedded.v1alpha1.ConverseRequest.config', index=0,
      number=1, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='audio_in',
full_name='google.assistant.embedded.v1alpha1.ConverseRequest.audio_in',
index=1,
      number=2, type=12, cpp_type=9, label=1,
      has_default_value=False, default_value=_b(""),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
  ],
```

```
  extensions=[
  ],
  nested_types=[],
  enum_types=[
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
    _descriptor.OneofDescriptor(
      name='converse_request',
full_name='google.assistant.embedded.v1alpha1.ConverseRequest.converse_request',
      index=0, containing_type=None, fields=[]),
  ],
  serialized_start=1287,
  serialized_end=1414,
)



_CONVERSERESPONSE = _descriptor.Descriptor(
  name='ConverseResponse',
  full_name='google.assistant.embedded.v1alpha1.ConverseResponse',
```

```
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='error',
full_name='google.assistant.embedded.v1alpha1.ConverseResponse.error', index=0,
      number=1, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='event_type',
full_name='google.assistant.embedded.v1alpha1.ConverseResponse.event_type',
index=1,
      number=2, type=14, cpp_type=8, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
```

```
      name='audio_out',
full_name='google.assistant.embedded.v1alpha1.ConverseResponse.audio_out',
index=2,
      number=3, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='device_action',
full_name='google.assistant.embedded.v1alpha1.ConverseResponse.device_action',
index=3,
      number=9, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='result',
full_name='google.assistant.embedded.v1alpha1.ConverseResponse.result', index=4,
      number=5, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
```

```
    options=None),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
   _CONVERSERESPONSE_EVENTTYPE,
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
   _descriptor.OneofDescriptor(
    name='converse_response',
full_name='google.assistant.embedded.v1alpha1.ConverseResponse.converse_respon
se',
    index=0, containing_type=None, fields=[]),
  ],
  serialized_start=1417,
  serialized_end=1854,
)
```

```
_DEVICECONFIG = _descriptor.Descriptor(
  name='DeviceConfig',
  full_name='google.assistant.embedded.v1alpha1.DeviceConfig',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='device_id',
full_name='google.assistant.embedded.v1alpha1.DeviceConfig.device_id', index=0,
      number=1, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
    _descriptor.FieldDescriptor(
      name='device_model_id',
full_name='google.assistant.embedded.v1alpha1.DeviceConfig.device_model_id',
index=1,
      number=3, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None),
```

```
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=1856,
  serialized_end=1914,
)



_DEVICEACTION = _descriptor.Descriptor(
  name='DeviceAction',
  full_name='google.assistant.embedded.v1alpha1.DeviceAction',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
```

```
    fields=[
      _descriptor.FieldDescriptor(
        name='device_request_json',
    full_name='google.assistant.embedded.v1alpha1.DeviceAction.device_request_json',
    index=0,
        number=2, type=9, cpp_type=9, label=1,
        has_default_value=False, default_value=_b("").decode('utf-8'),
        message_type=None, enum_type=None, containing_type=None,
        is_extension=False, extension_scope=None,
        options=None),
    ],
    extensions=[
    ],
    nested_types=[],
    enum_types=[
    ],
    options=None,
    is_extendable=False,
    syntax='proto3',
    extension_ranges=[],
    oneofs=[
    ],
    serialized_start=1916,
    serialized_end=1959,
```

)

_CONVERSECONFIG.fields_by_name['audio_in_config'].message_type = _AUDIOINCONFIG

_CONVERSECONFIG.fields_by_name['audio_out_config'].message_type = _AUDIOOUTCONFIG

_CONVERSECONFIG.fields_by_name['converse_state'].message_type = _CONVERSESTATE

_CONVERSECONFIG.fields_by_name['device_config'].message_type = _DEVICECONFIG

_AUDIOINCONFIG.fields_by_name['encoding'].enum_type = _AUDIOINCONFIG_ENCODING

_AUDIOINCONFIG_ENCODING.containing_type = _AUDIOINCONFIG

_AUDIOOUTCONFIG.fields_by_name['encoding'].enum_type = _AUDIOOUTCONFIG_ENCODING

_AUDIOOUTCONFIG_ENCODING.containing_type = _AUDIOOUTCONFIG

_CONVERSERESULT.fields_by_name['microphone_mode'].enum_type = _CONVERSERESULT_MICROPHONEMODE

_CONVERSERESULT_MICROPHONEMODE.containing_type = _CONVERSERESULT

_CONVERSEREQUEST.fields_by_name['config'].message_type = _CONVERSECONFIG

_CONVERSEREQUEST.oneofs_by_name['converse_request'].fields.append(

_CONVERSEREQUEST.fields_by_name['config'])

_CONVERSEREQUEST.fields_by_name['config'].containing_oneof = _CONVERSEREQUEST.oneofs_by_name['converse_request']

```
_CONVERSEREQUEST.oneofs_by_name['converse_request'].fields.append(
  _CONVERSEREQUEST.fields_by_name['audio_in'])
_CONVERSEREQUEST.fields_by_name['audio_in'].containing_oneof =
_CONVERSEREQUEST.oneofs_by_name['converse_request']
CONVERSERESPONSE.fields_by_name['error'].message_type =
google_dot_rpc_dot_status_pb2._STATUS
_CONVERSERESPONSE.fields_by_name['event_type'].enum_type =
_CONVERSERESPONSE_EVENTTYPE
_CONVERSERESPONSE.fields_by_name['audio_out'].message_type =
_AUDIOOUT
_CONVERSERESPONSE.fields_by_name['device_action'].message_type =
_DEVICEACTION
_CONVERSERESPONSE.fields_by_name['result'].message_type =
_CONVERSERESULT
_CONVERSERESPONSE_EVENTTYPE.containing_type =
_CONVERSERESPONSE
_CONVERSERESPONSE.oneofs_by_name['converse_response'].fields.append(
  _CONVERSERESPONSE.fields_by_name['error'])
_CONVERSERESPONSE.fields_by_name['error'].containing_oneof =
_CONVERSERESPONSE.oneofs_by_name['converse_response']
_CONVERSERESPONSE.oneofs_by_name['converse_response'].fields.append(
  _CONVERSERESPONSE.fields_by_name['event_type'])
_CONVERSERESPONSE.fields_by_name['event_type'].containing_oneof =
_CONVERSERESPONSE.oneofs_by_name['converse_response']
_CONVERSERESPONSE.oneofs_by_name['converse_response'].fields.append(
  _CONVERSERESPONSE.fields_by_name['audio_out'])
```

```
_CONVERSERESPONSE.fields_by_name['audio_out'].containing_oneof =
_CONVERSERESPONSE.oneofs_by_name['converse_response']

_CONVERSERESPONSE.oneofs_by_name['converse_response'].fields.append(

  _CONVERSERESPONSE.fields_by_name['device_action'])

_CONVERSERESPONSE.fields_by_name['device_action'].containing_oneof =
_CONVERSERESPONSE.oneofs_by_name['converse_response']

_CONVERSERESPONSE.oneofs_by_name['converse_response'].fields.append(

  _CONVERSERESPONSE.fields_by_name['result'])

_CONVERSERESPONSE.fields_by_name['result'].containing_oneof =
_CONVERSERESPONSE.oneofs_by_name['converse_response']

DESCRIPTOR.message_types_by_name['ConverseConfig'] =
_CONVERSECONFIG

DESCRIPTOR.message_types_by_name['AudioInConfig'] = _AUDIOINCONFIG

DESCRIPTOR.message_types_by_name['AudioOutConfig'] =
_AUDIOOUTCONFIG

DESCRIPTOR.message_types_by_name['ConverseState'] = _CONVERSESTATE

DESCRIPTOR.message_types_by_name['AudioOut'] = _AUDIOOUT

DESCRIPTOR.message_types_by_name['ConverseResult'] = _CONVERSERESULT

DESCRIPTOR.message_types_by_name['ConverseRequest'] =
_CONVERSEREQUEST

DESCRIPTOR.message_types_by_name['ConverseResponse'] =
_CONVERSERESPONSE

DESCRIPTOR.message_types_by_name['DeviceConfig'] = _DEVICECONFIG

DESCRIPTOR.message_types_by_name['DeviceAction'] = _DEVICEACTION

_sym_db.RegisterFileDescriptor(DESCRIPTOR)
```

```
ConverseConfig = _reflection.GeneratedProtocolMessageType('ConverseConfig',
(_message.Message,), dict(
  DESCRIPTOR = _CONVERSECONFIG,
  _module_ = 'google.assistant.embedded.v1alpha1.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha1.ConverseConfig)
  ))
_sym_db.RegisterMessage(ConverseConfig)


AudioInConfig = _reflection.GeneratedProtocolMessageType('AudioInConfig',
(_message.Message,), dict(
  DESCRIPTOR = _AUDIOINCONFIG,
  _module_ = 'google.assistant.embedded.v1alpha1.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha1.AudioInConfig)
  ))
_sym_db.RegisterMessage(AudioInConfig)


AudioOutConfig = _reflection.GeneratedProtocolMessageType('AudioOutConfig',
(_message.Message,), dict(
  DESCRIPTOR = _AUDIOOUTCONFIG,
  _module_ = 'google.assistant.embedded.v1alpha1.embedded_assistant_pb2'
```

```
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha1.AudioOutConfig)
  ))
_sym_db.RegisterMessage(AudioOutConfig)


ConverseState = _reflection.GeneratedProtocolMessageType('ConverseState', (_message.Message,), dict(
  DESCRIPTOR = _CONVERSESTATE,
  _module_ = 'google.assistant.embedded.v1alpha1.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha1.ConverseState)
  ))
_sym_db.RegisterMessage(ConverseState)


AudioOut = _reflection.GeneratedProtocolMessageType('AudioOut', (_message.Message,), dict(
  DESCRIPTOR = _AUDIOOUT,
  _module_ = 'google.assistant.embedded.v1alpha1.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha1.AudioOut)
  ))
_sym_db.RegisterMessage(AudioOut)
```

```
ConverseResult = _reflection.GeneratedProtocolMessageType('ConverseResult',
(_message.Message,), dict(
  DESCRIPTOR = _CONVERSERESULT,
  _module_ = 'google.assistant.embedded.v1alpha1.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha1.ConverseResult)
  ))
_sym_db.RegisterMessage(ConverseResult)


ConverseRequest = _reflection.GeneratedProtocolMessageType('ConverseRequest',
(_message.Message,), dict(
  DESCRIPTOR = _CONVERSEREQUEST,
  _module_ = 'google.assistant.embedded.v1alpha1.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha1.ConverseRequest)
  ))
_sym_db.RegisterMessage(ConverseRequest)


ConverseResponse =
_reflection.GeneratedProtocolMessageType('ConverseResponse',
(_message.Message,), dict(
  DESCRIPTOR = _CONVERSERESPONSE,
```

```
  _module_ = 'google.assistant.embedded.v1alpha1.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha1.ConverseResponse)
  ))
_sym_db.RegisterMessage(ConverseResponse)


DeviceConfig = _reflection.GeneratedProtocolMessageType('DeviceConfig', (_message.Message,), dict(
  DESCRIPTOR = _DEVICECONFIG,
  _module_ = 'google.assistant.embedded.v1alpha1.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha1.DeviceConfig)
  ))
_sym_db.RegisterMessage(DeviceConfig)


DeviceAction = _reflection.GeneratedProtocolMessageType('DeviceAction', (_message.Message,), dict(
  DESCRIPTOR = _DEVICEACTION,
  _module_ = 'google.assistant.embedded.v1alpha1.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha1.DeviceAction)
  ))
```

```
_sym_db.RegisterMessage(DeviceAction)


DESCRIPTOR.has_options = True
DESCRIPTOR._options = _descriptor._ParseOptions(descriptor_pb2.FileOptions(),
_b('\n&com.google.assistant.embedded.v1alpha1B\016AssistantProtoP\001ZJgoogle.
golang.org/genproto/googleapis/assistant/embedded/v1alpha1;embedded'))


_EMBEDDEDASSISTANT = _descriptor.ServiceDescriptor(
  name='EmbeddedAssistant',
  full_name='google.assistant.embedded.v1alpha1.EmbeddedAssistant',
  file=DESCRIPTOR,
  index=0,
  options=None,
  serialized_start=1962,
  serialized_end=2104,
  methods=[
  _descriptor.MethodDescriptor(
    name='Converse',
    full_name='google.assistant.embedded.v1alpha1.EmbeddedAssistant.Converse',
    index=0,
    containing_service=None,
    input_type=_CONVERSEREQUEST,
    output_type=_CONVERSERESPONSE,
```

```
    options=None,
  ),
])
_sym_db.RegisterServiceDescriptor(_EMBEDDEDASSISTANT)


DESCRIPTOR.services_by_name['EmbeddedAssistant'] =
_EMBEDDEDASSISTANT


# @@protoc_insertion_point(module_scope)


import grpc


from google.assistant.embedded.v1alpha1 import embedded_assistant_pb2 as
google_dot_assistant_dot_embedded_dot_v1alpha1_dot_embedded_assistant_pb2



class EmbeddedAssistantStub(object):
  """Service that implements Google Assistant API.
  """

  def _init_(self, channel):
    """Constructor.
    Args:
      channel: A grpc.Channel.
```

```
    """
    self.Converse = channel.stream_stream(
        '/google.assistant.embedded.v1alpha1.EmbeddedAssistant/Converse',
request_serializer=google_dot_assistant_dot_embedded_dot_v1alpha1_dot_embedded_assistant_pb2.ConverseRequest.SerializeToString,
response_deserializer=google_dot_assistant_dot_embedded_dot_v1alpha1_dot_embedded_assistant_pb2.ConverseResponse.FromString,
        )


class EmbeddedAssistantServicer(object):
  """Service that implements Google Assistant API.
  """

  def Converse(self, request_iterator, context):
    """Initiates or continues a conversation with the embedded assistant service.
    Each call performs one round-trip, sending an audio request to the service
    and receiving the audio response. Uses bidirectional streaming to receive
    results, such as the `END_OF_UTTERANCE` event, while sending audio.
    A conversation is one or more gRPC connections, each consisting of several
    streamed requests and responses.
    For example, the user says Add to my shopping list and the assistant
```

responds What do you want to add?. The sequence of streamed requests and responses in the first gRPC message could be:

* ConverseRequest.config
* ConverseRequest.audio_in
* ConverseRequest.audio_in
* ConverseRequest.audio_in
* ConverseRequest.audio_in
* ConverseResponse.event_type.END_OF_UTTERANCE
* ConverseResponse.result.spoken_request_text "add to my shopping list"
* ConverseResponse.result.microphone_mode.DIALOG_FOLLOW_ON
* ConverseResponse.audio_out
* ConverseResponse.audio_out
* ConverseResponse.audio_out

The user then says bagels and the assistant responds

OK, I've added bagels to your shopping list. This is sent as another gRPC connection call to the `Converse` method, again with streamed requests and responses, such as:

* ConverseRequest.config
* ConverseRequest.audio_in
* ConverseRequest.audio_in
* ConverseRequest.audio_in
* ConverseResponse.event_type.END_OF_UTTERANCE
* ConverseResponse.result.microphone_mode.CLOSE_MICROPHONE

```
    * ConverseResponse.audio_out
    * ConverseResponse.audio_out
    * ConverseResponse.audio_out
    * ConverseResponse.audio_out
    Although the precise order of responses is not guaranteed, sequential
    ConverseResponse.audio_out messages will always contain sequential portions
    of audio.
    """
    context.set_code(grpc.StatusCode.UNIMPLEMENTED)
    context.set_details('Method not implemented!')
    raise NotImplementedError('Method not implemented!')


def add_EmbeddedAssistantServicer_to_server(servicer, server):
  rpc_method_handlers = {
      'Converse': grpc.stream_stream_rpc_method_handler(
          servicer.Converse,
request_deserializer=google_dot_assistant_dot_embedded_dot_v1alpha1_dot_embedded_assistant_pb2.ConverseRequest.FromString,
response_serializer=google_dot_assistant_dot_embedded_dot_v1alpha1_dot_embedded_assistant_pb2.ConverseResponse.SerializeToString,
      ),
```

```
  }
  generic_handler = grpc.method_handlers_generic_handler(
      'google.assistant.embedded.v1alpha1.EmbeddedAssistant', rpc_method_handlers)
  server.add_generic_rpc_handlers((generic_handler,))
```

_int.py

```
"""google.assistant.embedded namespace package."""

try:
    import pkg_resources
    pkg_resources.declare_namespace(_name_)
except ImportError:
    import pkgutil
    _path_ = pkgutil.extend_path(_path, __name_)

import sys
_b=sys.version_info[0]<3 and (lambda x:x) or (lambda x:x.encode('latin1'))
from google.protobuf import descriptor as _descriptor
from google.protobuf import message as _message
from google.protobuf import reflection as _reflection
from google.protobuf import symbol_database as _symbol_database
from google.protobuf import descriptor_pb2
# @@protoc_insertion_point(imports)
```

```
_sym_db = _symbol_database.Default()



from google.api import annotations_pb2 as google_dot_api_dot_annotations__pb2

from google.type import latlng_pb2 as google_dot_type_dot_latlng__pb2



DESCRIPTOR = _descriptor.FileDescriptor(

  name='google/assistant/embedded/v1alpha2/embedded_assistant.proto',

  package='google.assistant.embedded.v1alpha2',

  syntax='proto3',


serialized_pb=_b('\n;google/assistant/embedded/v1alpha2/embedded_assistant.proto\x
12\"google.assistant.embedded.v1alpha2\x1a\x1cgoogle/api/annotations.proto\x1a\x1
8google/type/latlng.proto\"o\n\rAssistRequest\x12\x42\n\x06\x63onfig\x18\x01
\x01(\x0b\x32\x30.google.assistant.embedded.v1alpha2.AssistConfigH\x00\x12\x12\
n\x08\x61udio_in\x18\x02
\x01(\x0cH\x00\x42\x06\n\x04type\"\x91\x04\n\x0e\x41ssistResponse\x12P\n\nevent
_type\x18\x01
\x01(\x0e\x32<.google.assistant.embedded.v1alpha2.AssistResponse.EventType\x12?
\n\taudio_out\x18\x03
\x01(\x0b\x32,.google.assistant.embedded.v1alpha2.AudioOut\x12\x41\n\nscreen_out
\x18\x04 \x01(\x0b\x32-
.google.assistant.embedded.v1alpha2.ScreenOut\x12G\n\rdevice_action\x18\x06
\x01(\x0b\x32\x30.google.assistant.embedded.v1alpha2.DeviceAction\x12S\n\x0espe
ech_results\x18\x02
```

\x03(\x0b\x32;.google.assistant.embedded.v1alpha2.SpeechRecognitionResult\x12L\n\x10\x64ialog_state_out\x18\x05
\x01(\x0b\x32\x32.google.assistant.embedded.v1alpha2.DialogStateOut\"=\n\tEventType\x12\x1a\n\x16\x45VENT_TYPE_UNSPECIFIED\x10\x00\x12\x14\n\x10\x45ND_OF_UTTERANCE\x10\x01\"\xad\x03\n\x0c\x41ssistConfig\x12L\n\x0f\x61udio_in_config\x18\x01
\x01(\x0b\x32\x31.google.assistant.embedded.v1alpha2.AudioInConfigH\x00\x12\x14\n\ntext_query\x18\x06 \x01(\tH\x00\x12L\n\x10\x61udio_out_config\x18\x02
\x01(\x0b\x32\x32.google.assistant.embedded.v1alpha2.AudioOutConfig\x12N\n\x11screen_out_config\x18\x08
\x01(\x0b\x32\x33.google.assistant.embedded.v1alpha2.ScreenOutConfig\x12J\n\x0f\x64ialog_state_in\x18\x03
\x01(\x0b\x32\x31.google.assistant.embedded.v1alpha2.DialogStateIn\x12G\n\rdevice_config\x18\x04
\x01(\x0b\x32\x30.google.assistant.embedded.v1alpha2.DeviceConfigB\x06\n\x04type\"\xb6\x01\n\rAudioInConfig\x12L\n\x08\x65ncoding\x18\x01
\x01(\x0e\x32:.google.assistant.embedded.v1alpha2.AudioInConfig.Encoding\x12\x19\n\x11sample_rate_hertz\x18\x02
\x01(\x05\"<\n\x08\x45ncoding\x12\x18\n\x14\x45NCODING_UNSPECIFIED\x10\x00\x12\x0c\n\x08LINEAR16\x10\x01\x12\x08\n\x04\x46LAC\x10\x02\"\xe3\x01\n\x0e\x41udioOutConfig\x12M\n\x08\x65ncoding\x18\x01
\x01(\x0e\x32;.google.assistant.embedded.v1alpha2.AudioOutConfig.Encoding\x12\x19\n\x11sample_rate_hertz\x18\x02
\x01(\x05\x12\x19\n\x11volume_percentage\x18\x03
\x01(\x05\"L\n\x08\x45ncoding\x12\x18\n\x14\x45NCODING_UNSPECIFIED\x10\x00\x12\x0c\n\x08LINEAR16\x10\x01\x12\x07\n\x03MP3\x10\x02\x12\x0f\n\x0bOPUS_IN_OGG\x10\x03\"\xa7\x01\n\x0fScreenOutConfig\x12S\n\x0bscreen_mode\x18\x01
\x01(\x0e\x32>.google.assistant.embedded.v1alpha2.ScreenOutConfig.ScreenMode\"?\n\nScreenMode\x12\x1b\n\x17SCREEN_MODE_UNSPECIFIED\x10\x00\x12\x07\n\x03OFF\x10\x01\x12\x0b\n\x07PLAYING\x10\x03\"\xac\x01\n\rDialogStateIn\x12\x1a\n\x12\x63onversation_state\x18\x01

\x01(\x0c\x12\x15\n\rlanguage_code\x18\x02
\x01(\t\x12K\n\x0f\x64\x65vice_location\x18\x05
\x01(\x0b\x32\x32.google.assistant.embedded.v1alpha2.DeviceLocation\x12\x1b\n\x13is_new_conversation\x18\x07
\x01(\x08\":\n\x0c\x44\x65viceConfig\x12\x11\n\tdevice_id\x18\x01
\x01(\t\x12\x17\n\x0f\x64\x65vice_model_id\x18\x03
\x01(\t\"\x1e\n\x08\x41udioOut\x12\x12\n\naudio_data\x18\x01
\x01(\x0c\"\x8b\x01\n\tScreenOut\x12\x44\n\x06\x66ormat\x18\x01
\x01(\x0e\x32\x34.google.assistant.embedded.v1alpha2.ScreenOut.Format\x12\x0c\n\x04\x64\x61ta\x18\x02
\x01(\x0c\"\n\x06\x46ormat\x12\x16\n\x12\x46ORMAT_UNSPECIFIED\x10\x00\x12\x08\n\x04HTML\x10\x01\"+\n\x0c\x44\x65viceAction\x12\x1b\n\x13\x64\x65vice_request_json\x18\x01
\x01(\t\"@\n\x17SpeechRecognitionResult\x12\x12\n\ntranscript\x18\x01
\x01(\t\x12\x11\n\tstability\x18\x02
\x01(\x02\"\xa5\x02\n\x0e\x44ialogStateOut\x12!\n\x19supplemental_display_text\x18\x01 \x01(\t\x12\x1a\n\x12\x63onversation_state\x18\x02
\x01(\x0c\x12Z\n\x0fmicrophone_mode\x18\x03
\x01(\x0e\x32\x41.google.assistant.embedded.v1alpha2.DialogStateOut.MicrophoneMode\x12\x19\n\x11volume_percentage\x18\x04
\x01(\x05\"]\n\x0eMicrophoneMode\x12\x1f\n\x1bMICROPHONE_MODE_UNSPECIFIED\x10\x00\x12\x14\n\x10\x43LOSE_MICROPHONE\x10\x01\x12\x14\n\x10\x44IALOG_FOLLOW_ON\x10\x02\"D\n\x0e\x44\x65viceLocation\x12\n\x0b\x63oordinates\x18\x01
\x01(\x0b\x32\x13.google.type.LatLngH\x00\x42\x06\n\x04type2\x88\x01\n\x11\x45mbeddedAssistant\x12s\n\x06\x41ssist\x12\x31.google.assistant.embedded.v1alpha2.AssistRequest\x1a\x32.google.assistant.embedded.v1alpha2.AssistResponse(\x01\x30\x01\x42\x8f\x01\n&com.google.assistant.embedded.v1alpha2B\x0e\x41ssistantProtoP\x01ZJgoogle.golang.org/genproto/googleapis/assistant/embedded/v1alpha2;embedded\xa2\x02\x06\x41STSDKb\x06proto3')

,

```
dependencies=[google_dot_api_dot_annotations_pb2.DESCRIPTOR,google_dot_typ
e_dot_latlng_pb2.DESCRIPTOR,])



_ASSISTRESPONSE_EVENTTYPE = _descriptor.EnumDescriptor(
  name='EventType',
  full_name='google.assistant.embedded.v1alpha2.AssistResponse.EventType',
  filename=None,
  file=DESCRIPTOR,
  values=[
    _descriptor.EnumValueDescriptor(
      name='EVENT_TYPE_UNSPECIFIED', index=0, number=0,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='END_OF_UTTERANCE', index=1, number=1,
      options=None,
      type=None),
  ],
  containing_type=None,
  options=None,
  serialized_start=737,
```

```
  serialized_end=798,
)
_sym_db.RegisterEnumDescriptor(_ASSISTRESPONSE_EVENTTYPE)


_AUDIOINCONFIG_ENCODING = _descriptor.EnumDescriptor(
  name='Encoding',
  full_name='google.assistant.embedded.v1alpha2.AudioInConfig.Encoding',
  filename=None,
  file=DESCRIPTOR,
  values=[
    _descriptor.EnumValueDescriptor(
      name='ENCODING_UNSPECIFIED', index=0, number=0,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='LINEAR16', index=1, number=1,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='FLAC', index=2, number=2,
      options=None,
      type=None),
  ],
```

```
  containing_type=None,
  options=None,
  serialized_start=1355,
  serialized_end=1415,
)
_sym_db.RegisterEnumDescriptor(_AUDIOINCONFIG_ENCODING)


_AUDIOOUTCONFIG_ENCODING = _descriptor.EnumDescriptor(
  name='Encoding',
  full_name='google.assistant.embedded.v1alpha2.AudioOutConfig.Encoding',
  filename=None,
  file=DESCRIPTOR,
  values=[
    _descriptor.EnumValueDescriptor(
      name='ENCODING_UNSPECIFIED', index=0, number=0,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='LINEAR16', index=1, number=1,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='MP3', index=2, number=2,
```

```
    options=None,
    type=None),
  _descriptor.EnumValueDescriptor(
    name='OPUS_IN_OGG', index=3, number=3,
    options=None,
    type=None),
 ],
 containing_type=None,
 options=None,
 serialized_start=1569,
 serialized_end=1645,
)
_sym_db.RegisterEnumDescriptor(_AUDIOOUTCONFIG_ENCODING)


_SCREENOUTCONFIG_SCREENMODE = _descriptor.EnumDescriptor(
 name='ScreenMode',
 full_name='google.assistant.embedded.v1alpha2.ScreenOutConfig.ScreenMode',
 filename=None,
 file=DESCRIPTOR,
 values=[
  _descriptor.EnumValueDescriptor(
    name='SCREEN_MODE_UNSPECIFIED', index=0, number=0,
    options=None,
```

```
      type=None),
    _descriptor.EnumValueDescriptor(
      name='OFF', index=1, number=1,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='PLAYING', index=2, number=3,
      options=None,
      type=None),
  ],
  containing_type=None,
  options=None,
  serialized_start=1752,
  serialized_end=1815,
)
_sym_db.RegisterEnumDescriptor(_SCREENOUTCONFIG_SCREENMODE)


_SCREENOUT_FORMAT = _descriptor.EnumDescriptor(
  name='Format',
  full_name='google.assistant.embedded.v1alpha2.ScreenOut.Format',
  filename=None,
  file=DESCRIPTOR,
  values=[
```

```
    _descriptor.EnumValueDescriptor(
      name='FORMAT_UNSPECIFIED', index=0, number=0,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='HTML', index=1, number=1,
      options=None,
      type=None),
  ],
  containing_type=None,
  options=None,
  serialized_start=2182,
  serialized_end=2224,
)
_sym_db.RegisterEnumDescriptor(_SCREENOUT_FORMAT)


_DIALOGSTATEOUT_MICROPHONEMODE = _descriptor.EnumDescriptor(
  name='MicrophoneMode',
  full_name='google.assistant.embedded.v1alpha2.DialogStateOut.MicrophoneMode',
  filename=None,
  file=DESCRIPTOR,
  values=[
    _descriptor.EnumValueDescriptor(
```

```
      name='MICROPHONE_MODE_UNSPECIFIED', index=0, number=0,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='CLOSE_MICROPHONE', index=1, number=1,
      options=None,
      type=None),
    _descriptor.EnumValueDescriptor(
      name='DIALOG_FOLLOW_ON', index=2, number=2,
      options=None,
      type=None),
  ],
  containing_type=None,
  options=None,
  serialized_start=2538,
  serialized_end=2631,
)
_sym_db.RegisterEnumDescriptor(_DIALOGSTATEOUT_MICROPHONEMODE)



_ASSISTREQUEST = _descriptor.Descriptor(
  name='AssistRequest',
  full_name='google.assistant.embedded.v1alpha2.AssistRequest',
```

```
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='config',
full_name='google.assistant.embedded.v1alpha2.AssistRequest.config', index=0,
      number=1, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='audio_in',
full_name='google.assistant.embedded.v1alpha2.AssistRequest.audio_in', index=1,
      number=2, type=12, cpp_type=9, label=1,
      has_default_value=False, default_value=_b(""),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
  ],
  extensions=[
  ],
  nested_types=[],
```

```
  enum_types=[
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
    _descriptor.OneofDescriptor(
      name='type', full_name='google.assistant.embedded.v1alpha2.AssistRequest.type',
      index=0, containing_type=None, fields=[]),
  ],
  serialized_start=155,
  serialized_end=266,
)



_ASSISTRESPONSE = _descriptor.Descriptor(
  name='AssistResponse',
  full_name='google.assistant.embedded.v1alpha2.AssistResponse',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
```

```
    _descriptor.FieldDescriptor(
      name='event_type',
full_name='google.assistant.embedded.v1alpha2.AssistResponse.event_type',
index=0,
      number=1, type=14, cpp_type=8, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='audio_out',
full_name='google.assistant.embedded.v1alpha2.AssistResponse.audio_out', index=1,
      number=3, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='screen_out',
full_name='google.assistant.embedded.v1alpha2.AssistResponse.screen_out',
index=2,
      number=4, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
```

```
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='device_action',
full_name='google.assistant.embedded.v1alpha2.AssistResponse.device_action',
index=3,
      number=6, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='speech_results',
full_name='google.assistant.embedded.v1alpha2.AssistResponse.speech_results',
index=4,
      number=2, type=11, cpp_type=10, label=3,
      has_default_value=False, default_value=[],
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='dialog_state_out',
full_name='google.assistant.embedded.v1alpha2.AssistResponse.dialog_state_out',
index=5,
      number=5, type=11, cpp_type=10, label=1,
```

```
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
    _ASSISTRESPONSE_EVENTTYPE,
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=269,
  serialized_end=798,
)



_ASSISTCONFIG = _descriptor.Descriptor(
```

```
  name='AssistConfig',
  full_name='google.assistant.embedded.v1alpha2.AssistConfig',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='audio_in_config',
full_name='google.assistant.embedded.v1alpha2.AssistConfig.audio_in_config',
index=0,
      number=1, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='text_query',
full_name='google.assistant.embedded.v1alpha2.AssistConfig.text_query', index=1,
      number=6, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
```

```
      name='audio_out_config',
full_name='google.assistant.embedded.v1alpha2.AssistConfig.audio_out_config',
index=2,
      number=2, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='screen_out_config',
full_name='google.assistant.embedded.v1alpha2.AssistConfig.screen_out_config',
index=3,
      number=8, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='dialog_state_in',
full_name='google.assistant.embedded.v1alpha2.AssistConfig.dialog_state_in',
index=4,
      number=3, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
```

```
    options=None, file=DESCRIPTOR),
  _descriptor.FieldDescriptor(
    name='device_config',
full_name='google.assistant.embedded.v1alpha2.AssistConfig.device_config',
index=5,
    number=4, type=11, cpp_type=10, label=1,
    has_default_value=False, default_value=None,
    message_type=None, enum_type=None, containing_type=None,
    is_extension=False, extension_scope=None,
    options=None, file=DESCRIPTOR),
 ],
 extensions=[
 ],
 nested_types=[],
 enum_types=[
 ],
 options=None,
 is_extendable=False,
 syntax='proto3',
 extension_ranges=[],
 oneofs=[
  _descriptor.OneofDescriptor(
    name='type', full_name='google.assistant.embedded.v1alpha2.AssistConfig.type',
    index=0, containing_type=None, fields=[]),
```

```
  ],
  serialized_start=801,
  serialized_end=1230,
)



_AUDIOINCONFIG = _descriptor.Descriptor(
  name='AudioInConfig',
  full_name='google.assistant.embedded.v1alpha2.AudioInConfig',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='encoding',
full_name='google.assistant.embedded.v1alpha2.AudioInConfig.encoding', index=0,
      number=1, type=14, cpp_type=8, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
```

```
      name='sample_rate_hertz',
full_name='google.assistant.embedded.v1alpha2.AudioInConfig.sample_rate_hertz',
index=1,
      number=2, type=5, cpp_type=1, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
    _AUDIOINCONFIG_ENCODING,
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=1233,
  serialized_end=1415,
)
```

```
_AUDIOOUTCONFIG = _descriptor.Descriptor(
  name='AudioOutConfig',
  full_name='google.assistant.embedded.v1alpha2.AudioOutConfig',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='encoding',
full_name='google.assistant.embedded.v1alpha2.AudioOutConfig.encoding', index=0,
      number=1, type=14, cpp_type=8, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='sample_rate_hertz',
full_name='google.assistant.embedded.v1alpha2.AudioOutConfig.sample_rate_hertz',
index=1,
      number=2, type=5, cpp_type=1, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
```

```
    is_extension=False, extension_scope=None,
    options=None, file=DESCRIPTOR),
  _descriptor.FieldDescriptor(
    name='volume_percentage',
full_name='google.assistant.embedded.v1alpha2.AudioOutConfig.volume_percentage
', index=2,
    number=3, type=5, cpp_type=1, label=1,
    has_default_value=False, default_value=0,
    message_type=None, enum_type=None, containing_type=None,
    is_extension=False, extension_scope=None,
    options=None, file=DESCRIPTOR),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
   _AUDIOOUTCONFIG_ENCODING,
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
```

```
  serialized_start=1418,
  serialized_end=1645,
)



_SCREENOUTCONFIG = _descriptor.Descriptor(
  name='ScreenOutConfig',
  full_name='google.assistant.embedded.v1alpha2.ScreenOutConfig',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='screen_mode',
full_name='google.assistant.embedded.v1alpha2.ScreenOutConfig.screen_mode',
index=0,
      number=1, type=14, cpp_type=8, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
  ],
  extensions=[
  ],
```

```
  nested_types=[],
  enum_types=[
    _SCREENOUTCONFIG_SCREENMODE,
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=1648,
  serialized_end=1815,
)



_DIALOGSTATEIN = _descriptor.Descriptor(
  name='DialogStateIn',
  full_name='google.assistant.embedded.v1alpha2.DialogStateIn',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
```

```
      name='conversation_state',
full_name='google.assistant.embedded.v1alpha2.DialogStateIn.conversation_state',
index=0,
      number=1, type=12, cpp_type=9, label=1,
      has_default_value=False, default_value=_b(""),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='language_code',
full_name='google.assistant.embedded.v1alpha2.DialogStateIn.language_code',
index=1,
      number=2, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='device_location',
full_name='google.assistant.embedded.v1alpha2.DialogStateIn.device_location',
index=2,
      number=5, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
```

```
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='is_new_conversation',
full_name='google.assistant.embedded.v1alpha2.DialogStateIn.is_new_conversation',
index=3,
      number=7, type=8, cpp_type=7, label=1,
      has_default_value=False, default_value=False,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=1818,
  serialized_end=1990,
```

)

```
_DEVICECONFIG = _descriptor.Descriptor(
  name='DeviceConfig',
  full_name='google.assistant.embedded.v1alpha2.DeviceConfig',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='device_id',
full_name='google.assistant.embedded.v1alpha2.DeviceConfig.device_id', index=0,
      number=1, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='device_model_id',
full_name='google.assistant.embedded.v1alpha2.DeviceConfig.device_model_id',
index=1,
      number=3, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
```

```
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=1992,
  serialized_end=2050,
)



_AUDIOOUT = _descriptor.Descriptor(
  name='AudioOut',
  full_name='google.assistant.embedded.v1alpha2.AudioOut',
```

```
    filename=None,
    file=DESCRIPTOR,
    containing_type=None,
    fields=[
      _descriptor.FieldDescriptor(
        name='audio_data',
    full_name='google.assistant.embedded.v1alpha2.AudioOut.audio_data', index=0,
        number=1, type=12, cpp_type=9, label=1,
        has_default_value=False, default_value=_b(""),
        message_type=None, enum_type=None, containing_type=None,
        is_extension=False, extension_scope=None,
        options=None, file=DESCRIPTOR),
    ],
    extensions=[
    ],
    nested_types=[],
    enum_types=[
    ],
    options=None,
    is_extendable=False,
    syntax='proto3',
    extension_ranges=[],
    oneofs=[
```

```
  ],
  serialized_start=2052,
  serialized_end=2082,
)



_SCREENOUT = _descriptor.Descriptor(
  name='ScreenOut',
  full_name='google.assistant.embedded.v1alpha2.ScreenOut',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='format',
full_name='google.assistant.embedded.v1alpha2.ScreenOut.format', index=0,
      number=1, type=14, cpp_type=8, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='data', full_name='google.assistant.embedded.v1alpha2.ScreenOut.data',
index=1,
```

```
      number=2, type=12, cpp_type=9, label=1,
      has_default_value=False, default_value=_b(""),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
    _SCREENOUT_FORMAT,
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=2085,
  serialized_end=2224,
)
```

```
_DEVICEACTION = _descriptor.Descriptor(
  name='DeviceAction',
  full_name='google.assistant.embedded.v1alpha2.DeviceAction',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='device_request_json',
full_name='google.assistant.embedded.v1alpha2.DeviceAction.device_request_json',
index=0,
      number=1, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
  ],
  options=None,
  is_extendable=False,
```

```
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=2226,
  serialized_end=2269,
)



_SPEECHRECOGNITIONRESULT = _descriptor.Descriptor(
  name='SpeechRecognitionResult',
  full_name='google.assistant.embedded.v1alpha2.SpeechRecognitionResult',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='transcript',
full_name='google.assistant.embedded.v1alpha2.SpeechRecognitionResult.transcript',
index=0,
      number=1, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
```

```
    options=None, file=DESCRIPTOR),
  _descriptor.FieldDescriptor(
    name='stability',
full_name='google.assistant.embedded.v1alpha2.SpeechRecognitionResult.stability',
index=1,
    number=2, type=2, cpp_type=6, label=1,
    has_default_value=False, default_value=float(0),
    message_type=None, enum_type=None, containing_type=None,
    is_extension=False, extension_scope=None,
    options=None, file=DESCRIPTOR),
 ],
 extensions=[
 ],
 nested_types=[],
 enum_types=[
 ],
 options=None,
 is_extendable=False,
 syntax='proto3',
 extension_ranges=[],
 oneofs=[
 ],
 serialized_start=2271,
 serialized_end=2335,
```

)

```
_DIALOGSTATEOUT = _descriptor.Descriptor(
  name='DialogStateOut',
  full_name='google.assistant.embedded.v1alpha2.DialogStateOut',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
    _descriptor.FieldDescriptor(
      name='supplemental_display_text',
full_name='google.assistant.embedded.v1alpha2.DialogStateOut.supplemental_displa
y_text', index=0,
      number=1, type=9, cpp_type=9, label=1,
      has_default_value=False, default_value=_b("").decode('utf-8'),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='conversation_state',
full_name='google.assistant.embedded.v1alpha2.DialogStateOut.conversation_state',
index=1,
      number=2, type=12, cpp_type=9, label=1,
```

```
      has_default_value=False, default_value=_b(""),
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='microphone_mode',
full_name='google.assistant.embedded.v1alpha2.DialogStateOut.microphone_mode',
index=2,
      number=3, type=14, cpp_type=8, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
    _descriptor.FieldDescriptor(
      name='volume_percentage',
full_name='google.assistant.embedded.v1alpha2.DialogStateOut.volume_percentage',
index=3,
      number=4, type=5, cpp_type=1, label=1,
      has_default_value=False, default_value=0,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
  ],
  extensions=[
```

```
  ],
  nested_types=[],
  enum_types=[
    _DIALOGSTATEOUT_MICROPHONEMODE,
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
  ],
  serialized_start=2338,
  serialized_end=2631,
)



_DEVICELOCATION = _descriptor.Descriptor(
  name='DeviceLocation',
  full_name='google.assistant.embedded.v1alpha2.DeviceLocation',
  filename=None,
  file=DESCRIPTOR,
  containing_type=None,
  fields=[
```

```
    _descriptor.FieldDescriptor(
      name='coordinates',
full_name='google.assistant.embedded.v1alpha2.DeviceLocation.coordinates',
index=0,
      number=1, type=11, cpp_type=10, label=1,
      has_default_value=False, default_value=None,
      message_type=None, enum_type=None, containing_type=None,
      is_extension=False, extension_scope=None,
      options=None, file=DESCRIPTOR),
  ],
  extensions=[
  ],
  nested_types=[],
  enum_types=[
  ],
  options=None,
  is_extendable=False,
  syntax='proto3',
  extension_ranges=[],
  oneofs=[
    _descriptor.OneofDescriptor(
      name='type',
full_name='google.assistant.embedded.v1alpha2.DeviceLocation.type',
      index=0, containing_type=None, fields=[]),
```

```
  ],
  serialized_start=2633,
  serialized_end=2701,
)


_ASSISTREQUEST.fields_by_name['config'].message_type = _ASSISTCONFIG
_ASSISTREQUEST.oneofs_by_name['type'].fields.append(
  _ASSISTREQUEST.fields_by_name['config'])
_ASSISTREQUEST.fields_by_name['config'].containing_oneof =
_ASSISTREQUEST.oneofs_by_name['type']
_ASSISTREQUEST.oneofs_by_name['type'].fields.append(
  _ASSISTREQUEST.fields_by_name['audio_in'])
_ASSISTREQUEST.fields_by_name['audio_in'].containing_oneof =
_ASSISTREQUEST.oneofs_by_name['type']
_ASSISTRESPONSE.fields_by_name['event_type'].enum_type =
_ASSISTRESPONSE_EVENTTYPE
_ASSISTRESPONSE.fields_by_name['audio_out'].message_type = _AUDIOOUT
_ASSISTRESPONSE.fields_by_name['screen_out'].message_type = _SCREENOUT
_ASSISTRESPONSE.fields_by_name['device_action'].message_type =
_DEVICEACTION
_ASSISTRESPONSE.fields_by_name['speech_results'].message_type =
_SPEECHRECOGNITIONRESULT
_ASSISTRESPONSE.fields_by_name['dialog_state_out'].message_type =
_DIALOGSTATEOUT
_ASSISTRESPONSE_EVENTTYPE.containing_type = _ASSISTRESPONSE
```

```
_ASSISTCONFIG.fields_by_name['audio_in_config'].message_type =
_AUDIOINCONFIG

_ASSISTCONFIG.fields_by_name['audio_out_config'].message_type =
_AUDIOOUTCONFIG

_ASSISTCONFIG.fields_by_name['screen_out_config'].message_type =
_SCREENOUTCONFIG

_ASSISTCONFIG.fields_by_name['dialog_state_in'].message_type =
_DIALOGSTATEIN

_ASSISTCONFIG.fields_by_name['device_config'].message_type =
_DEVICECONFIG

_ASSISTCONFIG.oneofs_by_name['type'].fields.append(

  _ASSISTCONFIG.fields_by_name['audio_in_config'])

_ASSISTCONFIG.fields_by_name['audio_in_config'].containing_oneof =
_ASSISTCONFIG.oneofs_by_name['type']

_ASSISTCONFIG.oneofs_by_name['type'].fields.append(

  _ASSISTCONFIG.fields_by_name['text_query'])

_ASSISTCONFIG.fields_by_name['text_query'].containing_oneof =
_ASSISTCONFIG.oneofs_by_name['type']

_AUDIOINCONFIG.fields_by_name['encoding'].enum_type =
_AUDIOINCONFIG_ENCODING

_AUDIOINCONFIG_ENCODING.containing_type = _AUDIOINCONFIG

_AUDIOOUTCONFIG.fields_by_name['encoding'].enum_type =
_AUDIOOUTCONFIG_ENCODING

_AUDIOOUTCONFIG_ENCODING.containing_type = _AUDIOOUTCONFIG

_SCREENOUTCONFIG.fields_by_name['screen_mode'].enum_type =
_SCREENOUTCONFIG_SCREENMODE
```

```
_SCREENOUTCONFIG_SCREENMODE.containing_type =
_SCREENOUTCONFIG

_DIALOGSTATEIN.fields_by_name['device_location'].message_type =
_DEVICELOCATION

_SCREENOUT.fields_by_name['format'].enum_type = _SCREENOUT_FORMAT

_SCREENOUT_FORMAT.containing_type = _SCREENOUT

_DIALOGSTATEOUT.fields_by_name['microphone_mode'].enum_type =
_DIALOGSTATEOUT_MICROPHONEMODE

_DIALOGSTATEOUT_MICROPHONEMODE.containing_type =
_DIALOGSTATEOUT

DEVICELOCATION.fields_by_name['coordinates'].message_type =
google_dot_type_dot_latlng_pb2._LATLNG

_DEVICELOCATION.oneofs_by_name['type'].fields.append(

  _DEVICELOCATION.fields_by_name['coordinates'])

_DEVICELOCATION.fields_by_name['coordinates'].containing_oneof =
_DEVICELOCATION.oneofs_by_name['type']

DESCRIPTOR.message_types_by_name['AssistRequest'] = _ASSISTREQUEST

DESCRIPTOR.message_types_by_name['AssistResponse'] = _ASSISTRESPONSE

DESCRIPTOR.message_types_by_name['AssistConfig'] = _ASSISTCONFIG

DESCRIPTOR.message_types_by_name['AudioInConfig'] = _AUDIOINCONFIG

DESCRIPTOR.message_types_by_name['AudioOutConfig'] =
_AUDIOOUTCONFIG

DESCRIPTOR.message_types_by_name['ScreenOutConfig'] =
_SCREENOUTCONFIG

DESCRIPTOR.message_types_by_name['DialogStateIn'] = _DIALOGSTATEIN
```

```
DESCRIPTOR.message_types_by_name['DeviceConfig'] = _DEVICECONFIG

DESCRIPTOR.message_types_by_name['AudioOut'] = _AUDIOOUT

DESCRIPTOR.message_types_by_name['ScreenOut'] = _SCREENOUT

DESCRIPTOR.message_types_by_name['DeviceAction'] = _DEVICEACTION

DESCRIPTOR.message_types_by_name['SpeechRecognitionResult'] =
_SPEECHRECOGNITIONRESULT

DESCRIPTOR.message_types_by_name['DialogStateOut'] = _DIALOGSTATEOUT

DESCRIPTOR.message_types_by_name['DeviceLocation'] = _DEVICELOCATION

_sym_db.RegisterFileDescriptor(DESCRIPTOR)


AssistRequest = _reflection.GeneratedProtocolMessageType('AssistRequest',
(_message.Message,), dict(

  DESCRIPTOR = _ASSISTREQUEST,

  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'

  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.AssistR
equest)

  ))

_sym_db.RegisterMessage(AssistRequest)


AssistResponse = _reflection.GeneratedProtocolMessageType('AssistResponse',
(_message.Message,), dict(

  DESCRIPTOR = _ASSISTRESPONSE,

  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'
```

```
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.AssistResponse)
  ))
_sym_db.RegisterMessage(AssistResponse)


AssistConfig = _reflection.GeneratedProtocolMessageType('AssistConfig', (_message.Message,), dict(
  DESCRIPTOR = _ASSISTCONFIG,
  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.AssistConfig)
  ))
_sym_db.RegisterMessage(AssistConfig)


AudioInConfig = _reflection.GeneratedProtocolMessageType('AudioInConfig', (_message.Message,), dict(
  DESCRIPTOR = _AUDIOINCONFIG,
  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.AudioInConfig)
  ))
_sym_db.RegisterMessage(AudioInConfig)
```

```
AudioOutConfig = _reflection.GeneratedProtocolMessageType('AudioOutConfig',
(_message.Message,), dict(
  DESCRIPTOR = _AUDIOOUTCONFIG,
  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.AudioO
utConfig)
  ))
_sym_db.RegisterMessage(AudioOutConfig)


ScreenOutConfig = _reflection.GeneratedProtocolMessageType('ScreenOutConfig',
(_message.Message,), dict(
  DESCRIPTOR = _SCREENOUTCONFIG,
  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.Screen
OutConfig)
  ))
_sym_db.RegisterMessage(ScreenOutConfig)


DialogStateIn = _reflection.GeneratedProtocolMessageType('DialogStateIn',
(_message.Message,), dict(
  DESCRIPTOR = _DIALOGSTATEIN,
  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'
```

```
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.DialogStateIn)

  ))

_sym_db.RegisterMessage(DialogStateIn)


DeviceConfig = _reflection.GeneratedProtocolMessageType('DeviceConfig',
(_message.Message,), dict(

  DESCRIPTOR = _DEVICECONFIG,

  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'

  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.DeviceConfig)

  ))

_sym_db.RegisterMessage(DeviceConfig)


AudioOut = _reflection.GeneratedProtocolMessageType('AudioOut',
(_message.Message,), dict(

  DESCRIPTOR = _AUDIOOUT,

  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'

  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.AudioOut)

  ))

_sym_db.RegisterMessage(AudioOut)
```

```
ScreenOut = _reflection.GeneratedProtocolMessageType('ScreenOut',
(_message.Message,), dict(
  DESCRIPTOR = _SCREENOUT,
  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.Screen
Out)
  ))
_sym_db.RegisterMessage(ScreenOut)


DeviceAction = _reflection.GeneratedProtocolMessageType('DeviceAction',
(_message.Message,), dict(
  DESCRIPTOR = _DEVICEACTION,
  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.Device
Action)
  ))
_sym_db.RegisterMessage(DeviceAction)


SpeechRecognitionResult =
_reflection.GeneratedProtocolMessageType('SpeechRecognitionResult',
(_message.Message,), dict(
  DESCRIPTOR = _SPEECHRECOGNITIONRESULT,
```

```
  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.Speech
RecognitionResult)
  ))
_sym_db.RegisterMessage(SpeechRecognitionResult)


DialogStateOut = _reflection.GeneratedProtocolMessageType('DialogStateOut',
(_message.Message,), dict(
  DESCRIPTOR = _DIALOGSTATEOUT,
  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.DialogS
tateOut)
  ))
_sym_db.RegisterMessage(DialogStateOut)


DeviceLocation = _reflection.GeneratedProtocolMessageType('DeviceLocation',
(_message.Message,), dict(
  DESCRIPTOR = _DEVICELOCATION,
  _module_ = 'google.assistant.embedded.v1alpha2.embedded_assistant_pb2'
  #
@@protoc_insertion_point(class_scope:google.assistant.embedded.v1alpha2.Device
Location)
  ))
```

```
_sym_db.RegisterMessage(DeviceLocation)



DESCRIPTOR.has_options = True

DESCRIPTOR._options = _descriptor._ParseOptions(descriptor_pb2.FileOptions(),
_b('\n&com.google.assistant.embedded.v1alpha2B\016AssistantProtoP\001ZJgoogle.
golang.org/genproto/googleapis/assistant/embedded/v1alpha2;embedded\242\002\006
ASTSDK'))



_EMBEDDEDASSISTANT = _descriptor.ServiceDescriptor(

 name='EmbeddedAssistant',

 full_name='google.assistant.embedded.v1alpha2.EmbeddedAssistant',

 file=DESCRIPTOR,

 index=0,

 options=None,

 serialized_start=2704,

 serialized_end=2840,

 methods=[

 _descriptor.MethodDescriptor(

  name='Assist',

  full_name='google.assistant.embedded.v1alpha2.EmbeddedAssistant.Assist',

  index=0,

  containing_service=None,

  input_type=_ASSISTREQUEST,
```

```
    output_type=_ASSISTRESPONSE,
    options=None,
  ),
])
_sym_db.RegisterServiceDescriptor(_EMBEDDEDASSISTANT)


DESCRIPTOR.services_by_name['EmbeddedAssistant'] =
_EMBEDDEDASSISTANT


language: python
python:
  - "3.6"
before_install:
  - sudo apt-get -qq update
  - sudo apt-get install -y portaudio19-dev libffi-dev libssl-dev
install:
  - pip install nox
script:
  - nox -f google-assistant-grpc/noxfile.py -s lint
  - nox -f google-assistant-sdk/noxfile.py -s lint unittest
```

**********nox.file**********

```
import os


import nox



@nox.session(python=["3"])
def lint(session):
    session.install('pip', 'setuptools')
    session.install('docutils', 'flake8', 'readme_renderer')
    session.run('flake8', 'nox.py', 'setup.py')
    session.run('python', 'setup.py', 'check',
                '--restructuredtext', '--strict')



@nox.session
def protoc(session):
    session.install('pip', 'setuptools')
    session.install('grpcio-tools')
    if os.path.exists('proto'):
        session.run('python', '-m', 'grpc_tools.protoc',
                    '--proto_path=proto',
```

```
                '--proto_path=googleapis',
                '--python_out=.',
                '--grpc_python_out=.',
                'proto/google/assistant/embedded/v1alpha2/'
                'embedded_assistant.proto',)
    else:
        session.run('python', '-m', 'grpc_tools.protoc',
                '--proto_path=googleapis',
                '--python_out=.',
                '--grpc_python_out=.',
                'googleapis/google/assistant/embedded/v1alpha2/'
                'embedded_assistant.proto')


@nox.session
def release(session):
    session.install('pip', 'setuptools', 'wheel')
    session.run('python', 'setup.py', 'sdist', 'bdist_wheel')
    session.run('python', 'setup.py', 'sdist', 'bdist_wheel')
```

**********Setup.py**********

```
from setuptools import setup, find_packages
import io


install_requires = [
    'googleapis-common-protos>=1.5.2',
    'grpcio>=1.3.5',
]
with io.open('README.rst', 'r') as fh:
    long_description = fh.read()
setup(
    name='google-assistant-grpc',
    version='0.3.0',
    author='Google Assistant SDK team',
    author_email='proppy+assistant-sdk@google.com',
    description='Google Assistant API gRPC bindings',
    long_description=long_description,
    url='https://github.com/googlesamples/assistant-sdk-python',
    packages=find_packages(),
    namespace_packages=[
        'google',
```

```
        'google.assistant',
        'google.assistant.embedded',
    ],
    install_requires=install_requires,
    license='Apache 2.0',
    keywords='google assistant api grpc',
    classifiers=(
        'Programming Language :: Python :: 2',
        'Programming Language :: Python :: 2.7',
        'Programming Language :: Python :: 3',
        'Programming Language :: Python :: 3.4',
        'Programming Language :: Python :: 3.5',
        'Development Status :: 4 - Beta',
        'Intended Audience :: Developers',
        'License :: OSI Approved :: Apache Software License',
        'Operating System :: POSIX',
        'Operating System :: Microsoft :: Windows',
        'Operating System :: MacOS :: MacOS X',
        'Operating System :: OS Independent',
        'Topic :: Internet :: WWW/HTTP',
    ),
)
```

Comments & Description of Coding Segments

When utilizing the steady model set up what usefulness is needed in every cycle to prompt the effective improvement of the undertaking. If any new necessities are found during the improvement cycle, they will either be added to a current emphasis assuming little or another emphasis will be added to the undertaking plan.

ITERATION	FUNCTIONALITY IMPLEMENTATION
Iteration 1	• Introduce and arrange VirtualBox VM and Transient • Introduce Laravel Estate • Interface with the information base • Add Login usefulness utilizing Auth library • Carry out Google Two-Element Confirmation
Iteration 2	• Incorporate Starling Programming interface • Have chatbot settle on decisions to Starling Programming interface to return client information when combined with the proper client goal and activity • Incorporate Fixer.io Programming interface to return changes when matched with suitable client expectation and activity • Carry out Email exchanges

Iteration 3	• Carry out the Dialogflow Programming interface • Train Chatbot utilizing purposes – expected inquiries made by clients • Train chatbot to discuss with the client • Make a webhook to get approaching messages from the client and concentrate elements • Carry out Twilio Programming interface for arrangement affirmation • Carry out discourse acknowledgment and amalgamation
Iteration 4	• Execute Chabot UI • Coordinate application to google colleague and Google Home

Table 2 Project Coding Comments & Description

Standardization of Coding

Standardization of coding ensures that software is:

- **Safe:** It can be used without causing harm.
- **Secure:** It can't be hacked.
- **Reliable:** It functions as it is designed for, every time.
- **Testable:** It can be tested at every code level.
- **Maintainable:** It can be maintained, even as your codebase grows.
- **Portable:** It works the same in a different environment.

There are four key benefits of using coding standards that we've used:

1. Compliance with industry standards (e.g., ISO).
2. Consistent code quality — no matter if different people write the code.
3. Software security from the start.
4. Reduced development costs and accelerated time to market.

Code Efficiency

We've followed these points to achieve better code efficiency in our application:

- Removed the unnecessary code
- Used optimal memory
- Ensured better runtime of the app
- Reusability of code whenever possible
- Used proper statements with error handling
- Optimized the code for better performance
- Used better programming concepts to implement the algorithms
- Used best practices for the data access & data management

Error Handling

We have used the debugging method to handle all the errors. With the help of debugging, we were able to find all the errors, and with the help of stack overflow and many open websites, we were able to handle all the errors smartly.

Parameters Calling/Passing

- **Formal Parameter:** A variable and its type as they are shown in the prototype of the function or method.
- **Actual Parameter:** The variable or expression in the same context as a formal parameter that appears in the function or method call in the calling environment.
- **Modes:**
  - **IN:** Passes info from user to code
  - **OUT:** code writes values in the program.
  - **IN/OUT:** user tells program value of a variable, which may be updated by the user.

Important methods of Parameter Passing

1. **Pass By Value:** This method used *in-mode* semantics. Changes made to formal parameters do not get transmitted back to the caller. Any modifications to the formal parameter variable inside the called function or method affect only the separate storage location and will not be reflected in the actual parameter in the calling environment. This method is also called ***call by value***.
2. **Pass by reference(aliasing) :** This technique uses *in/out-mode* semantics. Changes made to formal parameters do get transmitted back to the caller through parameter passing. Any changes to the formal parameter are reflected in the actual parameter in the calling environment as the formal parameter

receives a reference (or pointer) to the actual data. This method is also called **call by reference**. This method is efficient in both time and space

Validation Checks

Validation is an automatic computer checker to ensure that the data entered is sensible and reasonable. It does not check the accuracy of data.

Different validation types can be used to check the data that is being entered.

Validation type	Working	Example
Check digit	The last one or two digits in a code are used to check the other digits are correct	Bar code readers in supermarkets use check digits
Format check	Checks the data is in the right format	A National Insurance number is in the form LL 99 99 99 L where L is any letter and 9 is any number
Length check	Checks the data isn't too short or too long	A password which needs to be six letters long
Lookup table	Looks up acceptable values in a table	There are only seven possible days of the week
Presence check	Checks that data has been entered into a field	In most databases, a key field cannot be left blank
Range check	Checks that a value falls within the specified range	The number of hours worked must be less than 50 and more than 0
Spell check	Looks up words in a dictionary	When word processing

Table 3 Validation Check

Chapter 5: Testing

Testing Techniques and Testing Strategies Used

You can also use the built-in test functions to uncover bugs and prevent problems beforehand. To test your agent, you can also create a test casing using the simulator, then you can run and execute the test cases as per need. Executing tests also verifies that the agent responses do not differ from end-user inputs defined in the test case.

The instructions below show you how to use the console, but you can also find the same functionality in the API.

Testing Plan Used

Create a test case

To create a conversation:

1. Open the Dialogflow CX Console.
2. Choose your GCP project.
3. Select your agent.
4. Click **Test Agent** to open the simulator.
5. Chat with the agent to create a conversation that covers the functionality you want to test. For each turn, it checks the correct values for the intent, the agent response, and the session parameters.

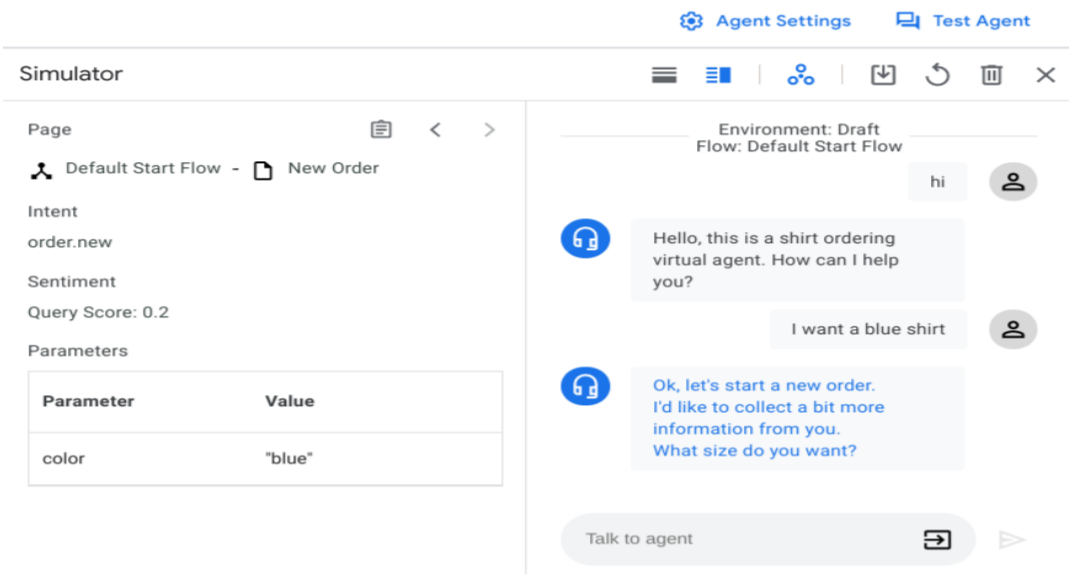

Figure 10 Test Cases 1

To save a conversation as a test case:

1. Click the save system_update_ alt button.
2. Enter a test case display name. Every test case must have a unique name.
3. Optionally provide a tag name. Tags help you organize your test cases. All tags must begin with a "#".
4. Optionally provide a note that describes the purpose of the test case.
5. Optionally select parameters you want to track in the test case. A list of suggested parameters is provided. You can also input other parameters. If you choose tracking parameters, the parameter implementation is checked when running the test case. See more details on the parameter implementation in the Run test cases section.
6. Click **Save** to save the test case.

Run test cases

To view all the test cases for an agent, click on test cases from the manage tab. The table shows the test name, tags, latest time and environment, and the result.

To run test cases:

1. Select the test cases you want to run, and click Run, or you can click on run all tests.
2. Select the environment you want to run the test cases against.
3. The tests start running and you can view the status in the task queue. The test result will be shown when it is completed.

To view the latest detailed result, go to the test case. the golden test case and the latest cases are shown simultaneously.

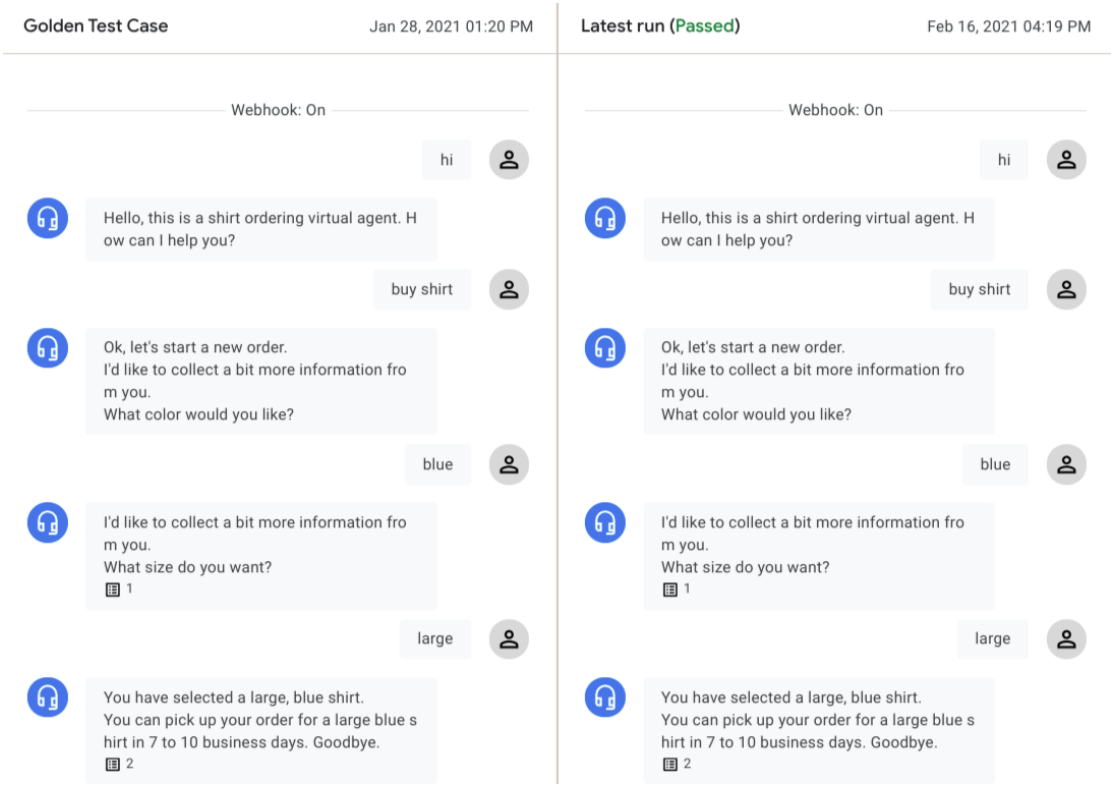

Figure 11 Test Cases 2

Click on an agent's conversational to see the details for that turn. The test engine checks the following types of data to evaluate the test result:

- **Agent dialogue**:

  For each conversational cycle, the agent dialogue is compared from the latest run. If they are not the same, a warning will be displayed. These differences do not stop a test from passing, because agent dialogue sometimes varies for the same agent state.

- **Matched intent**:

  The matched intent should be the same for each turn for a test to qualify.

- **Current page**:

  The active page should be the same for each turn for a test to qualify.

- **Session parameters**:

  If you include tracking parameters when you created the test cases, the test engine will check all the corresponding session parameters and do not qualify the test if there are missing or more than required parameters or the parameter value does not match.

In some scenarios, a test case may have an expected malfunction due to an update to an agent. You can select Save as golden to overwrite the golden test case if the dialogue in the most recent run reflects the expected changes.

Edit test cases

To edit a test case, select the test case from the Test cases table, then click the edit icon next to the name of the test case. The Update Test Cases dialog appears.

To edit the metadata and settings for the test cases, you can click on the Settings tab.

1. You can edit the Test case name, Tags, and Note fields, or add new tracking parameters.
2. Click Save.

To edit the user input for the test cases, you can click on the User Input tab.

1. Add, remove, or edit the user inputs in JSON format.
2. Click Confirm. An automatic test run begins, and the updated conversation displays after the test run are complete.
3. If you want to overwrite the golden test case click on save, or you can click on Save as to create a new test case with the changes.

View test coverage

To view a test coverage or report for all test cases, you can click on Coverage.

The Coverage page includes the following tabs:

- Transitions coverage is determined for all state handlers with a transition target implemented by the test case. The source page and transition target page are listed in the table.
- The coverage of intent is determined for all intents that fit the test case.
- The coverage of all route groups matched by the test case is determined.

Import and export test cases

To export test cases:

1. Select test cases and click **Export** or click **Export all test cases**.
2. Click **Download to a local file**, or provide a Cloud Storage bucket URI and click **Export to Google Cloud Storage**.

To import test cases:

1. Click **Import**.
2. Choose a local file or provide a Cloud Storage bucket URI.

Test Reports for Unit Test Cases & System Test Cases

S. No.	Topic	Objective	Status	
			YES	NO
1.	Main Menu	Contains the main headings like Introduction, About, Contact, etc.	✔	
2.	Background theme Image and Website Name (College 360°)	College 360° signifies that what the website is all about.	✔	
3.	Portfolio	This section contains the blogs and the basic necessary information of colleges	✔	
4.	Contact	This section contains the name, email, messages	✔	

	(Get in touch with us)	(Comment) that will help us to get the information and any required update in the post or website required.		
5.	Chatbot	This is the section where the user can ask for the information or any query that the user has and want to enquire can ask the BOT. If the inquiry asked by the user matches the database, then the Bot will confirm the query and information will be provided to the user. If the query asked by the user does not match the database, the BOT will ask for the information of the user that will help the admin to get the information for the user.	✔	

Table 4 Test Cases

Debugging And Code Improvement

1. **Chatfuel**

   It gives highlights like adding content cards and sharing updates to your devotees naturally, gathering data inside Courier visits with structures, and allowing clients to demand information and associate with your bot employing buttons.

2. **Botsify**

   Botsify utilizes a drag and drops layout to make bots. Elements like simple incorporations through modules, Savvy computer-based intelligence, AI, and examination mix are likewise accessible.

3. **Stream XO**

   This is the just chatbot stage to give more than 100 combinations. It brags a simple to-utilize visual manager. It likewise gives numerous pre-constructed layouts to a speedy beginning.

4. **Signal Boop**

   gives a start to finish engineer experience that permits clients to zero in on building incredible bots. Signal Boop is more equipped towards giving the best and simplest method of making slack bots.

5. **Bottr**

   gives you a choice to install your bot on your site. You can likewise add information from a Medium, WordPress, or Wikipedia site for better information inclusion.

6. **Movement man-made intelligence**

   to outwardly construct, train, and convey chatbots on FB Courier, Slack, Kiss, or your site. It allows you to graph your discussion stream like a flowchart to get a visual outline of the results of a bot inquiry.

7. **Chatty people**

   has predefined chatbots prepared to get going. The site has formats for a web-based business, client service, and food business. If you select a web-based

business chatbot you just need to add Items, back and forth discussion, and some broad settings.

8. **QnA Producer**

   Microsoft has made a QnA bot wherein you need to share the URL of your FAQ page and the bot will be made in almost no time utilizing the data on the FAQ page and organized information.

9. **Recast.AI | Shared Bot Stage**

   empowers you to prepare, fabricate and run your bot. Making and dealing with your discussion rationale with Bot Manufacturer and a visual stream interface assists you with setting up reactions rapidly.

10. **Botkit**

    incorporates an assortment of helpful devices, as Botkit Studio, standard application starter packs, a central library, and modules to broaden your bot abilities.

11. **Chatterton**

    The stage assists you with building the bot stream and setting up the man-made intelligence by entering a couple of instances of the normal discussion between the client and bot.

12. **Octane.ai**

    They have pre-assembled highlights that make it simple for you to add content, messages, conversations, finish-up structures, grandstand products, and more to your bot.

13. **Converse.io**

    It is coordinated into Various Stages, incorporates total Client, Solicitation, and Discussion Following, and has its own NLP parsing motor.

14. **Gupshup**

    Dissimilar to plain-instant messages, GupShup's creative keen messages contain organized information and insight, accordingly empowering progressed informing work processes and robotization.

System Security Measures (Implementation of the Security for the Project Developed)

Encryption:

Information while traveling can likewise be abused. There exist various conventions that give encryption, while resolving these issues of abuse and altering.

As per article 32 (a) of the Overall Information Assurance Guideline (GDPR), "it is explicitly necessitated that organizations go to lengths to de-distinguish and scramble individual information. Along these lines, chatbots approach just to scrambled channels and impart through those".

For example, Facebook Courier presented the new component called "Secret Discussions" that empowered start to finish encryption dependent on Signal Convention.

Confirmation and Approval:

Confirmation is performed when the client needs to check their character. This is frequently utilized for bank chatbots.

Created validation tokens confirm information that is mentioned through a chatbot. On finishing the confirmation of the client's character, the Application creates a protected validation token, alongside the solicitation.

One more advance of safety efforts is a verification break. The token created is utilized for just a specific measure of time, after which the application needs to handle another one.

A two-way check is one more interaction where the client is approached to approve their email address or to get a code employing SMS. This is a significant cycle which is important to confirm that the client of that record is the genuine client that is utilizing the chatbot.

Affirmation and Endorsement:

Affirmation is performed when the customer needs to take a look at their person. This is as often as possible used for bank chatbots.

Made approval tokens affirm data that is referenced through a chatbot. On completing the affirmation of the customer's personality, the Application makes a secured approval token, close by the sales.

Another development of wellbeing endeavors is a confirmation break. The token made is used for simply a particular proportion of time, after which the application needs to deal with another.

A two-way check is another connection where the customer is drawn nearer to support their email address or to get a code through SMS. This is a huge cycle which is essential to affirm that the customer of that record is the certified customer that is using the chatbot.

Individual Output:

When working with individual information, it is important to go to security safety measures and lengths.

Apple was the principal organization that additional finger verification to their iPhones. This innovation is currently being utilized broadly to confirm a singular personality. This is performed while starting an exchange or when you need to get to your ledger utilizing a chatbot that an individual output is required.

Information Stockpiling:

Chatbots are viable because they recover and store data from clients.

For example, if you have a chatbot that performs online installments, this can imply that your customers are giving their monetary data to a chatbot.

The best arrangement in the present circumstance is to store such data in a protected state for a necessary measure of time and to dispose of this information later on.

Some different worries are the accompanying:

- Biometric verification: Iris sweeps and finger impression filters are well known and hearty.
- Client ID: Client IDs include handling secure login certifications.
- Verification Breaks: A 'ticking clock' for right confirmation input. This forestalls offering programmers a chance to speculate more passwords.
- Different techniques could incorporate 2FA, conduct examination, and credit to the steadily advancing computer-based intelligence patterns.

Handling Human Causes:

The unrivaled other component or because that can't be changed is the human variable. With business applications in explicit, that chatbot security and end-client

strategy must be settled. This will guarantee the chatbots from being powerless against dangers.

Database/Data Security

As the principal focal point of connection is between the client and the chatbot, which is brought out through regular language, the discussion configuration is an urgent viewpoint to think about when planning the chatbot. To accomplish this; the fitting discourse will be planned through the Dialogflow console utilizing follow-up and fall-back aims. A specific goal will be set, that once summoned by the client, the chatbot will end the discussion successfully paying little heed to the conversational setting.

User: “please book me an appointment”

The exchange is the assortment of words and expressions used to react to client input. The exchange will be planned with human-like expressions to react to clients to sound more regular. Another plan perspective will incorporate fallback aims. If a client's expression can't be coordinated to an expectation, the chatbot will react with a message expressing it didn't coordinate with the contribution to plan and invite the client to rehash the inquiry or give more subtleties. The discourse is likewise planned to utilize follow-up expectations, these give a more regular conversational stream while connecting with the chatbot. Follow-up purposes extricate additional data from clients as and when needed, for example; a client might say;

User: “Please Book me an appointment”

The chatbot knows it needs additional data from the client, for example, date, time, and telephone number and will thusly utilize follow-up expectations to separate the necessary data from the client. Dialogflow suggests planning a straight discourse for cooperation where information present in the discussion is removed to assist the clients with accomplishing their objective, this is applied to the plan particularly is applied to the chatbot as it is an entrusted-based association chatbot that comprises banking space information.

Creation of User Profiles & Access Rights

Not available as there is no need to create a user account

Reports (Sample Layouts Should be Placed-*if Possible*)

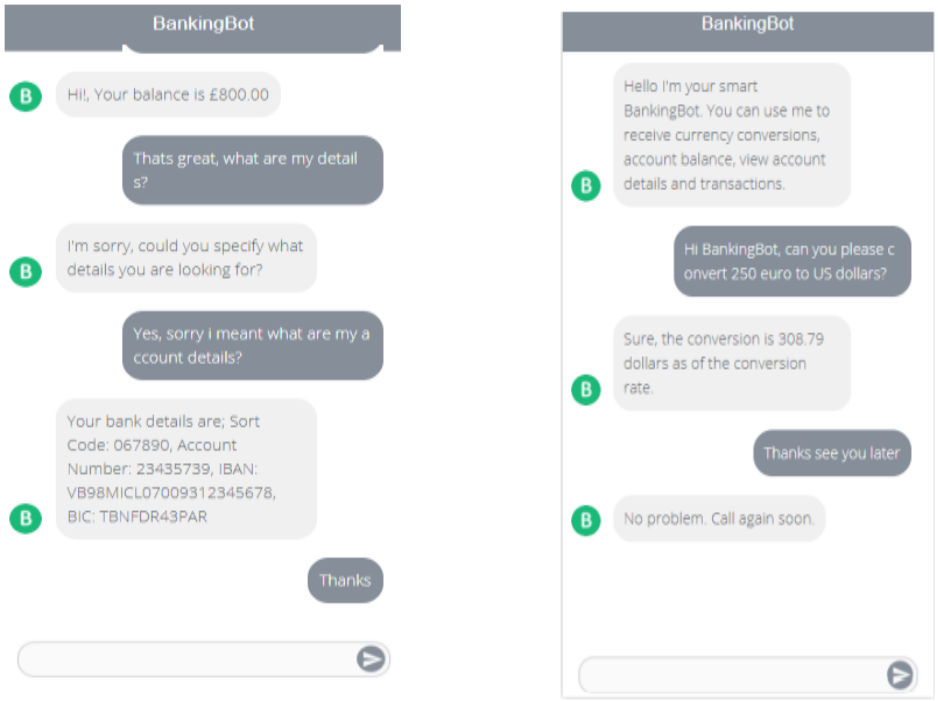

Figure 12 Sample ChatBot Layout

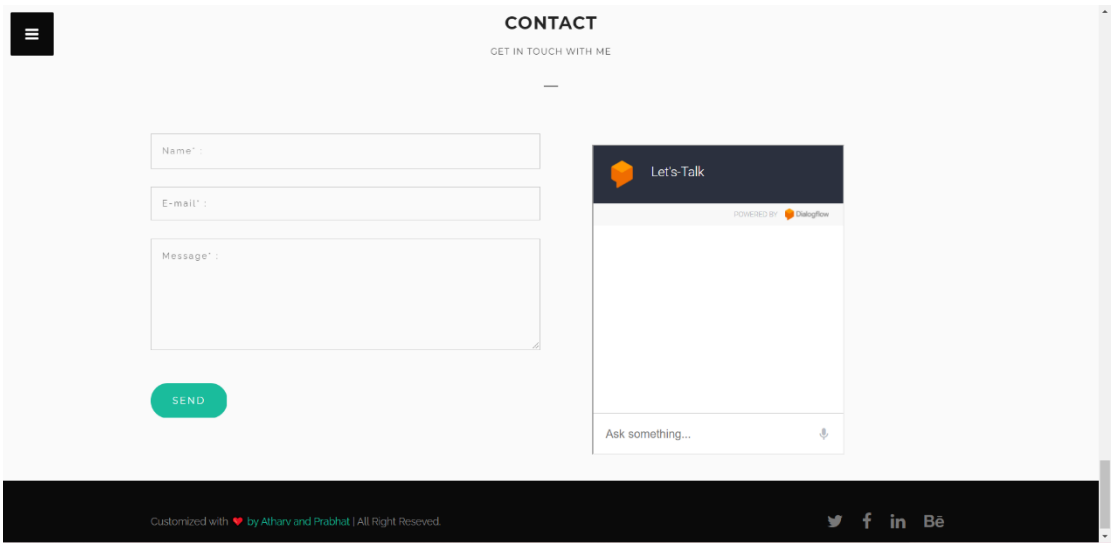

Figure 13 Chatbot implementation on Website

Chapter 6: Documentation

Cost Estimation of Project along with Cost Estimation Model

Based on our requirements and project, we selected the **Empirical Estimation Technique**.

App Feature	Est. Time in Hrs.	(Framework) in Hrs.	Backend in Hrs.	Development Cost (INR) in Hrs.	Total Hrs.	Total Cost (INR)
Strategy	8	9	NIL	70	9	630
Analysis and planning	6	5	NIL	90	5	450
UI/UX design	15	5	7	92	12	1104
Information Architecture and Workflow	24	10	12	175	22	3850
Software development (coding, designing)	12	5	8	93	13	1209
Testing (website and chat bot combined working)	11	3	6	82	9	738

Deployment and finalizing	15	7	9	85	16	1360
Total	91 Hrs.	44 Hrs.	42 Hrs.	--------	87 Hrs.	₹ 9340

Pre-Requisite for Project Installation

To run this project a user must have an active internet connection and the web address of the website in which the chatbot is included.

The updated web browser is a must requirement as many older versions do not support the use of chatbots.

Comparative Analysis

When we compare our project (Chatbot) with other projects currently available in the market we were able to observe that -

We have designed our chatbot so that the user does not have to scroll down each time he wants to access the chatbot / has any queries, our chatbot (let's talk) will follow the user's mouse cursor and adjust its position according to that at the bottom of the screen, meanwhile other chatbots available in the market are always at the end of the website.

Negative Results

We have tried to run our chatbot as smoothly as possible but there might be some bugs (minor) which are not been discovered yet due to our lack of experience, but we are always open to suggestions. Furthermore, the chatbot may deny providing information of some specific institutions that may be listed in our database

Chapter 7: Conclusion & Future Scope

Conclusion

From my perspective, chatbots or smart assistants with artificial intelligence are dramatically changing businesses. There is a wide range of chatbot-building platforms that are available for various enterprises, such as e-commerce, retail, banking, leisure, travel, healthcare, and so on.

Chatbots can reach out to a large audience on messaging apps and be more effective than humans. They may develop into a capable information-gathering tool shortly.

A chatbot is a program or service that easily connects with you to help solve your queries. The services that a chatbot can deliver are quite diverse, from providing important life-saving health messages to checking the weather forecast to purchasing a new pair of shoes. While interacting with a chatbot, you should feel as if you are talking with a real person.

Future Scope & Further Enhancement of the project

One of the combination s streams that were at first considered during improvement was to coordinate the chatbot with the Facebook Messenger administration, notwithstanding, during advancement, it was exposed through late media occasions, for example, the Cambridge Analytica.

a. Chatbots allow you to communicate with your users without hiring people and paying them.

b. communication with chatbots is practically brought to the level of human conversation.

c. chatbots get smarter with time.

d. The more conversations they take, the smarter they get

e. Chatbot applications streamline interactions between people and services, enhancing customer experience.

f. They offer companies new opportunities to improve the customer's engagement process

g. Operational efficiency by reducing the typical cost of customer service

h. A chatbot is a programmed chat interface a website visitor can interact with.

i. They are programmed to closely mimic human behavior and interact with the website visitor conversationally.

j. Chances are, you've used a live chat tool on a website.

k. Chatbots can be programmed to respond the same way each time, to respond differently to messages containing certain keywords, and even to use machine learning to adapt their responses to fit the situation.

From noticing the clients cooperate with the chatbot during client testing, there were various ideas to incorporate the chatbot across other famous stages like Amazon Echo or Dab, this would expand the accessibility of the chatbot and the incorporation of the DialogFlow API permit the chatbot to be handily traded to Amazon Alexa.

Bibliography (APA Format)

Google. (n.d.-a). *Google Cloud Platform Terms Of Service*. Google Cloud. Retrieved September 1, 2021, from https://cloud.google.com/terms

Google. (n.d.-b). *Release Notes | Dialogflow Documentation |*. Google Cloud. Retrieved September 1, 2021, from https://cloud.google.com/dialogflow/docs/release-notes

Rajesh, R. [Raji Rajesh]. (2021, April 7). *Creating Entities and Updating Avatar | Chatbot 2 | Google Dialogflow* [Video]. YouTube. https://www.youtube.com/watch?v=yupkCZTp4Ms&feature=youtu.be

Load, E. (2020, May 5). *Creating ChatBot || Dialogflow || Just in 30 Minutes*. YouTube.https://www.youtube.com/watch?v=0SDfi3JvacE&feature=youtu.be

Learners, W. (2021, July 22). *How to Add A Chatbot to Your Website*. YouTube. https://www.youtube.com/watch?v=y0CRhaWNpto&feature=youtu.be

Bergant, Jana. “ChatBots: Messenger ChatBot - DialogFlow and Nodejs | Udemy.” *Udemy*, Udemy, Oct. 2021, https://www.udemy.com/course/chatbots/.

“College Reviews – Careers360.Com.” *Careers360: Explore Exams, Colleges, Courses and Latest News about Education*, Careers360.com, Sept. 2021, https://www.careers360.com/colleges/reviews.

“Step-by-Step Guide On Building a Chatbot Using DialogFlow.” *Maruti Techlabs*, Maruti Techlabs, 30 Aug. 2019, https://marutitech.com/build-a-chatbot-using-dialogflow/.

UNIVERSITY INSTITUTE OF COMPUTING

DIVISION- MCA/BCA/BSc(CS)

Appendices (if any-Daily report)

Not Applicable

UNIVERSITY INSTITUTE OF COMPUTING

DIVISION- MCA/BCA/BSc(CS)

Glossary

Abbreviation/Phrases	Meaning
nodejs	Node.js® is a JavaScript runtime built on Chrome
Html	HyperText Markup Language
Css	Cascade style Sheets
JavaScript	JavaScript is the programming language of the Web.
Dialogflow	Dialogflow is a natural understanding platform that makes it easy to design and integrate

UNIVERSITY INSTITUTE OF COMPUTING

DIVISION- MCA/BCA/BSc(CS)

Plagiarism & Grammar Report

Plagiarism report

This Section is not yet completed.

As per the instructions provided by our supervisor (Deepak Vats).

Grammar report By Grammarly

UNIVERSITY INSTITUTE OF COMPUTING

DIVISION- MCA/BCA/BSc(CS)

grammarly Report: Final report_19bca1132 No Code

Final report_19bca1132 No Code

by Atharv_singh

General metrics

69,983	10,978	1074	43 min 54 sec	1 hr 24 min
characters	words	sentences	reading time	speaking time

Score

82

This text scores better than 82% of all texts checked by Grammarly

Writing Issues

49	✓	49
Issues left	Critical	Advanced

Unique Words

Measures vocabulary diversity by calculating the percentage of words used only once in your document

20% unique words

Rare Words

Measures depth of vocabulary by identifying words that are not among the 5,000 most common English words.

46% rare words

grammarly Report: Final report_19bca1132 No Code

Word Length 5

Measures average word length characters per word

Sentence Length 10.2

Measures average sentence length words per sentence

UNIVERSITY INSTITUTE OF COMPUTING

DIVISION- MCA/BCA/BSc(CS)

Outcome-based Conversion of Project into Patent/Copyright /GitHub Repository/Research paper.

Based on the outcome, our supervisor advised us to convert our project to a GitHub Repository. So, we've created a Repository on GitHub which includes all the files of the project and is accessible to everyone on the internet.

The link for our GitHub repository is:

https://github.com/Atharv753/college-enquiry-chatbot

Link for Executable Files or Setup of the Complete Project

https://drive.google.com/drive/folders/1sd48cBMyNeZd379ny8yA96MSGnB4ZL8U?usp=sharing

UNIVERSITY INSTITUTE OF COMPUTING

DIVISION- MCA/BCA/BSc(CS)