

INDUSTRIAL TRAINING

*Submitted in partial
fulfillment of the
Requirements for the
award of the degree*



Of

Bachelor of Technology

in

Information Technology

By:

Saiyam Sachdeva(08813203120/IT-2/2020)

**Department of Information Technology & Engineering
Guru Tegh Bahadur Institute of Technology**

**Guru Gobind Singh
Indraprastha
University Dwarka,
New Delhi
Year 2020-2024**

Chat-Bot Application

By:

Saiyam Sachdeva(08813203120/IT-2/2020)

At

**UDEMY,
By Dr. Angela Yu**

(i)

DECLARATION

I hereby declare that all the work presented in this Industrial Training Report for the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in **Information Technology & Engineering**, Guru Tegh Bahadur Institute of Technology, affiliated to Guru Gobind Singh Indraprastha University Delhi is an authentic record of our own work carried out at Udeemy, Mobile Development by Dr. Angela Yu from 10th July, 2022 to 9th September, 2022.

Date:

Saiyam Sachdeva
(08813203120/IT-2/2020)

CERTIFICATE



Certificate no: UC-e41d13c0-e882-4658-8a97-44658266db65
Certificate url: ude.my/UC-e41d13c0-e882-4658-8a97-44658266db65
Reference Number: 0004

CERTIFICATE OF COMPLETION

The Complete Flutter Development Bootcamp with Dart

Instructors **Dr. Angela Yu**

Saiyam Sachdeva

Date Aug 31, 2023
Length 29 total hours

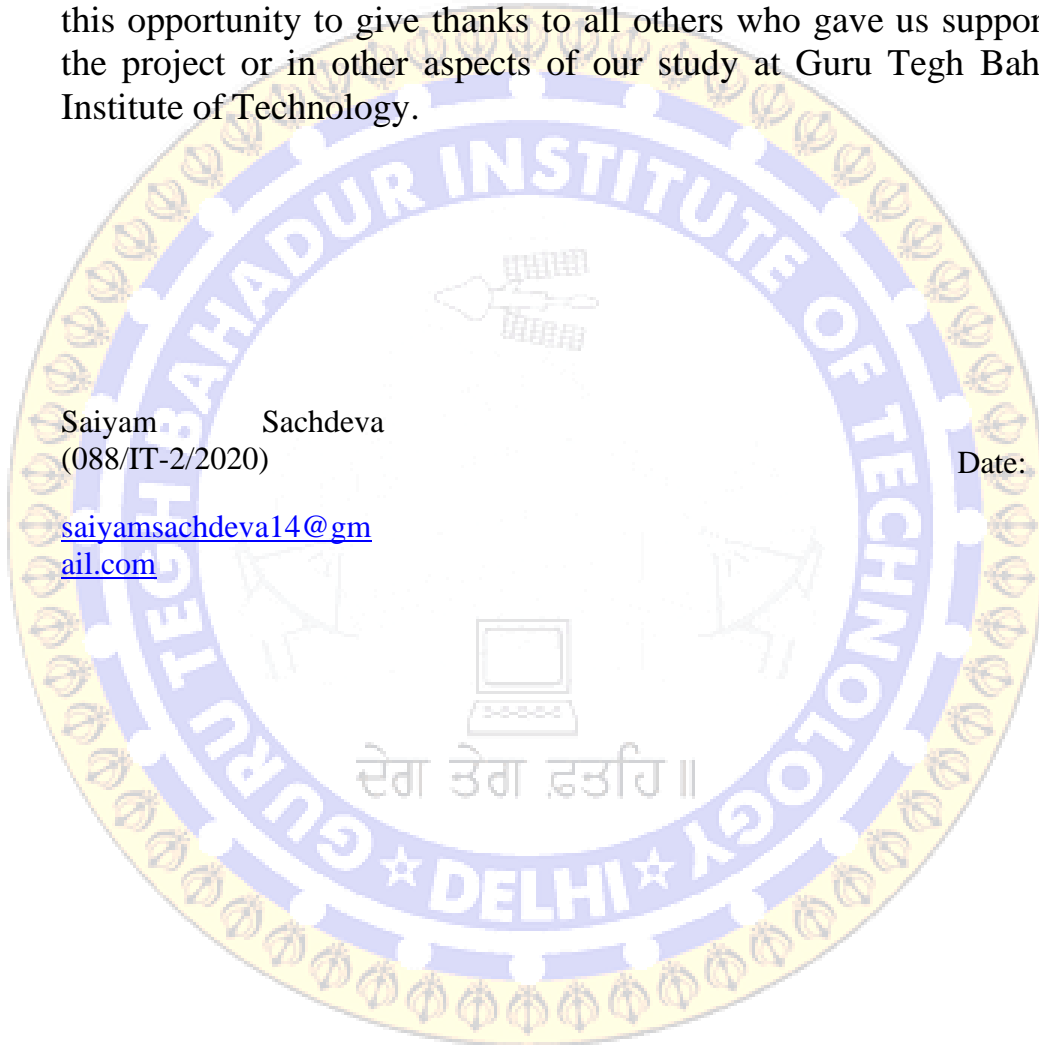
ACKNOWLEDGEMENT

I would like to express our great gratitude towards **Mrs. Rashmi** who has given us support and suggestions. Without their help we could not have presented this work upto the present standard. We also take this opportunity to give thanks to all others who gave us support for the project or in other aspects of our study at Guru Tegh Bahadur Institute of Technology.

Saiyam Sachdeva
(088/IT-2/2020)

saiyamsachdeva14@gmail.com

Date:



ABSTRACT

The Chat Bot application built using Dart and Flutter leverages the power of modern technology to provide users with an innovative and user-friendly conversational experience. Utilizing Dart's programming language and Flutter's rich widget library, the application offers a seamless and visually appealing interface for users to engage with the chat bot. It also serves as a virtual assistant, offering a dynamic platform for users to interact, seek information, and perform various tasks through natural language conversations. The incorporation of widgets and themes in Flutter ensures a visually pleasing and responsive design, enhancing the overall user experience.

The application's core functionality relies on its ability to understand and respond intelligently to user inputs, thereby assisting them in a wide array of tasks. By leveraging Dart's capabilities, the chat bot processes and interprets natural language efficiently, making conversations feel intuitive and engaging.

Furthermore, the integration of relevant packages enhances the application's feature set, allowing users to benefit from advanced functionalities seamlessly. The use of Dart in conjunction with Flutter ensures cross-platform compatibility, enabling the chat bot to reach a broader audience on different devices.

INDEX

TITLE PAGE

1

DECLARATION

CERTIFICATE

ACKNOWLEDGEMENT

ABSTRACT

1. INTRODUCTION

1.1 Chat-Bot Application

1.1.1 The Benefits of Using a Chat-Bot

1.1.2 Efficient Task Automation

1.1.3 Chat-Bot as a solution to efficient task automation

1.1.4 Essential Functionalities

1.2 FLUTTER

1.2.1 Introduction of FLUTTER

1.3 DART

1.3.1 Introduction of DART

1.4 DialogFlow

1.4.1 Introduction of DialogFlow

2. Chat-Bot IMPLEMENTATION

2.1 Introduction

2.2 Purpose

2.2.1 Scope

2.2.2 Overview

2.2.3 Overall Description

2.3 Specific Requirements

2.3.1 Operating Environment

2.4 Hardware Interfaces

2.5 Software System Attributes

2.5.1 Reliability

3. METHODOLOGY

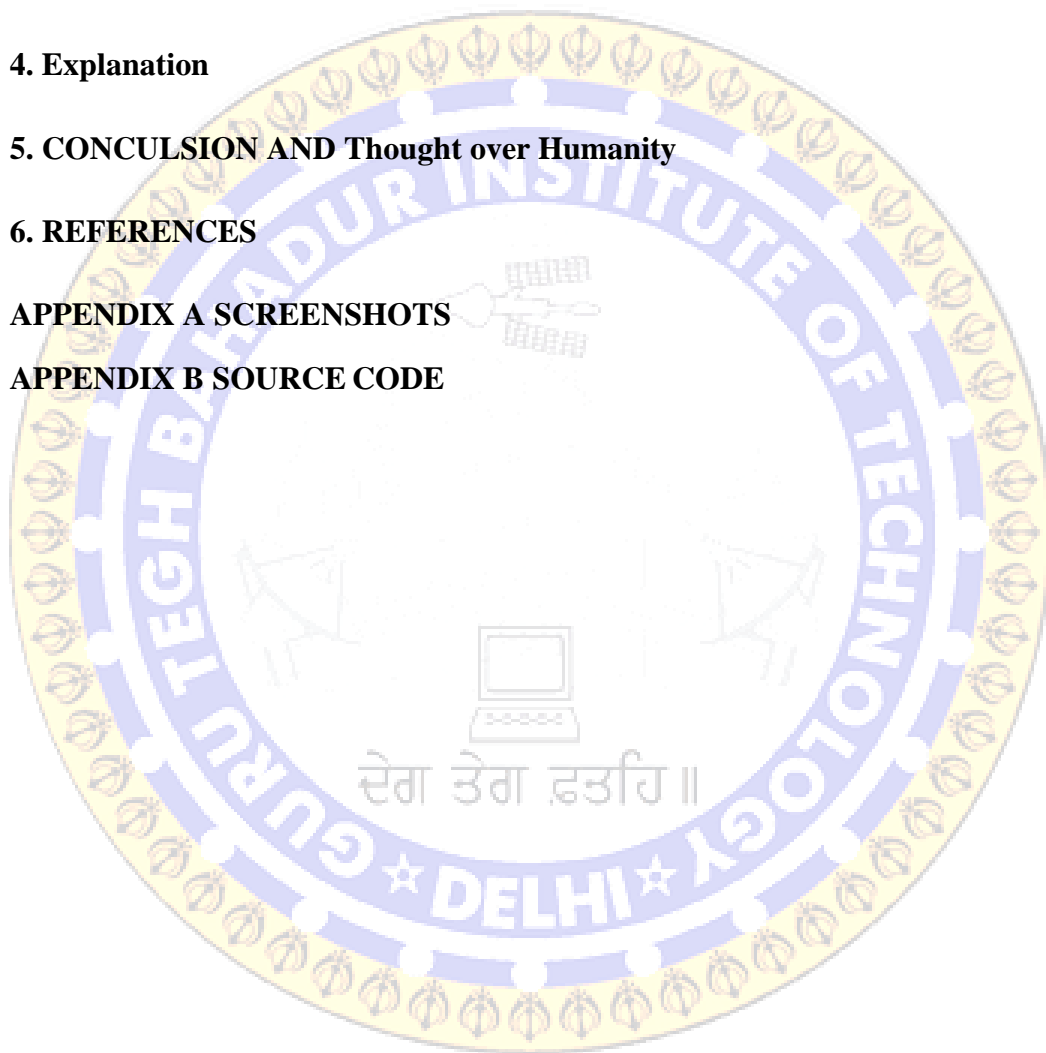
4. Explanation

5. CONCLUSION AND Thought over Humanity

6. REFERENCES

APPENDIX A SCREENSHOTS

APPENDIX B SOURCE CODE



INTRODUCTION

Introduction

The Chat Bot application built using Dart and Flutter leverages the power of modern technology to provide users with an innovative and user-friendly conversational experience. Utilizing Dart's programming language and Flutter's rich widget library, the application offers a seamless and visually appealing interface for users to engage with the chat bot.

This chat bot serves as a virtual assistant, offering a dynamic platform for users to interact, seek information, and perform various tasks through natural language conversations. The incorporation of widgets and themes in Flutter ensures a visually pleasing and responsive design, enhancing the overall user experience.

The application's core functionality relies on its ability to understand and respond intelligently to user inputs, thereby assisting them in a wide array of tasks. By leveraging Dart's capabilities, the chat bot processes and interprets natural language efficiently, making conversations feel intuitive and engaging.

Furthermore, the integration of relevant packages enhances the application's feature set, allowing users to benefit from advanced functionalities seamlessly. The use of Dart in conjunction with Flutter ensures cross-platform compatibility, enabling the chat bot to reach a broader audience on different devices.

The application's underlying data management is facilitated by modern database technology, providing a robust and efficient system for storing and retrieving information. This ensures that user interactions are not only dynamic but also personalized, as the chat bot learns and adapts to individual preferences over time.

1.1.1 The Benefits of Using a Chat-Bot Application:

1. **24/7 Availability:** Chat bots can operate around the clock, providing users with instant assistance and information at any time of day. This is particularly beneficial for businesses with a global audience or users in different time zones.
2. **Instant Response:** Chat bots deliver immediate responses to user queries, reducing wait times and enhancing the overall user experience. This can lead to higher customer satisfaction and engagement.
3. **Scalability:** Chat bots can handle a large number of concurrent interactions, making them scalable for businesses experiencing growth in user interactions. They can efficiently manage high volumes of requests without compromising on response time.

1.1.2 EFFICIENT TASK Automation

Efficient task automation involves the use of technology to streamline, expedite, and simplify repetitive or time-consuming tasks. In the context of chat bot applications,

efficient task automation refers to the ability of these bots to perform specific functions without requiring constant human intervention.

1.1.3 Chat-Bot as a solution to Efficient Task Automation

A Chat-Bot serves as a powerful solution for efficient task automation across various domains. Leveraging natural language processing (NLP), machine learning, and integration capabilities, chat bots streamline processes, enhance user experiences, and contribute to overall operational efficiency. Here's how a Chat-Bot can be a valuable solution for efficient task automation:

1. Automated Customer Support:

1.1. Task Automation: Chat bots can handle routine customer inquiries, providing instant responses to frequently asked questions. This includes order tracking, product information, and support ticket creation.

1.2. Issue Resolution: They can troubleshoot common problems, guide users through solutions, and escalate complex issues to human agents when necessary.

2. E-commerce Automation:

2.1. Order Processing: Chat bots can automate order placement, tracking, and status updates. Users can inquire about their orders, make changes, or initiate returns seamlessly.

2.2. Personalized Recommendations: Utilizing machine learning, chat bots can analyze user preferences and purchase history to offer personalized product recommendations.

3. Workflow Automation in Enterprises:

3.1. Approval Processes: Chat bots can automate approval workflows, such as expense approvals, leave requests, and document reviews. This reduces delays and ensures efficient decision-making.

3.2. Task Assignments: They can assist in task assignment, prioritization, and progress tracking within teams.

1.1.4 Essential Functionalities:

• Natural Language Processing (NLP):

- Intent Recognition: Ability to understand user intent behind messages.
- Entity Extraction: Identifying key information or entities in user input.

• User Input Handling:

- **Text Input Processing:** Managing and interpreting user messages in natural language.
- **Multimedia Input Support:** Handling images, voice messages, or other multimedia inputs.

• Response Generation:

- **Dynamic Responses:** Generating responses that are context-aware and dynamic.

- Personalization: Tailoring responses based on user preferences or historical interactions.

1.2 Flutter

1.2.1 Introduction of Flutter

Flutter is an open-source mobile SDK developer can use to build native-looking Android and iOS applications from the same code base. Flutter has been around since 2015 when Google introduced it and remained in the beta stage before its official launch in December 2018. Since then, the buzz around Flutter has been growing stronger. Flutter is now the top 11 software repos based on GitHub stars. Moreover, we have already seen thousands of Flutter apps being published on app stores. One of the most notable examples is the Xianyu app created by Alibaba team, used by over 50 million people.



Fig 1. Flutter

1.3 Dart

1.3.1 Introduction of Dart

Dart is an open- source general- purpose programming language developed by Google. It supports application development in both client and server-side. But it is widely used for the development of android apps, iOS apps, IoT(Internet of Things), and web applications using Flutter Framework.

Syntactically, Dart bears a strong resemblance to Java, C, and JavaScript. It is a dynamic object-oriented language with closure and

lexical scope. The Dart language was released in 2011 but came into popularity after 2015 with Dart 2.0.

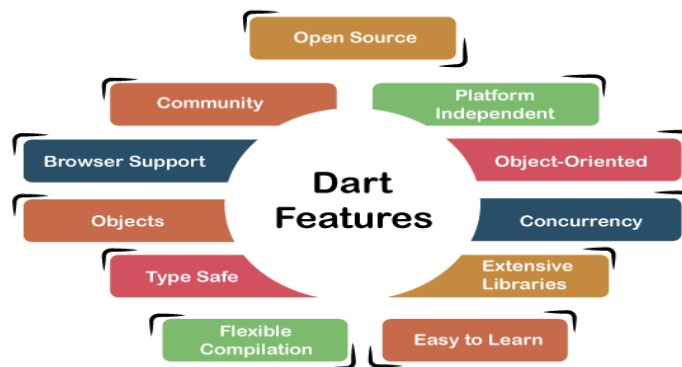


Fig 2. Dart

1.3 DialogFlow- Google API

1.3.1 Introduction of DialogFlow

Dialogflow is a natural language understanding (NLU) and conversation management platform developed by Google. It empowers developers to create chat bots, virtual agents, and interactive interfaces that can understand and respond to natural language inputs. Formerly known as API.AI, Dialogflow simplifies the process of building conversational applications by offering robust tools for language processing, intent recognition, and context management.



Fig 3. Dialogflow

IMPLEMENTATION

2.1 Introduction

This Project Chat-Bot is created using Dart language by using variety FLUTTER framework. This is used to take user input and adding it to a list._

Hive Database is used to store the data locally in the device so the user is not shown the default values again and again.



2.2 Purpose: The purpose of the Chat-Bot application designed for the Hospital Scheduling Process is to optimize and enhance the efficiency of the hospital's appointment scheduling system. By leveraging innovative chat-bot technology, the application aims to provide a seamless and user-friendly experience for both hospital staff and patients. The primary focus is on streamlining the scheduling process, reducing wait times, and maximizing the utilization of hospital resources. Through the implementation of this chat-bot solution, the overarching goal is to improve the overall capacity utilization of the hospital by 25%, ensuring that medical facilities are utilized to their full potential. Additionally, the project aims to significantly increase patient throughput by 75%, offering a more organized and expedited approach to appointment scheduling. Ultimately, the purpose of the chat-bot is to contribute to a more

efficient and patient-centric healthcare environment, where scheduling procedures are optimized, leading to improved overall healthcare services and patient satisfaction.

2.2.1 Scope: The scope of the Hospital Scheduling Chat-Bot project is extensive, encompassing various facets of the hospital's appointment scheduling process. It involves the creation of an intelligent and user-friendly chat-bot application tailored to meet the specific needs of both hospital staff and patients. The project aims to automate and streamline the entire scheduling workflow, from appointment creation to confirmation, rescheduling, and cancellation. Additionally, the chat-bot will be designed to handle inquiries regarding available time slots, physician availability, and other relevant scheduling details. The scope extends to the integration of the chat-bot with the existing hospital information system, ensuring seamless communication and data exchange. The project's focus on increasing hospital capacity utilization by 25% and enhancing patient throughput by 75% underscores its ambition to bring about a transformative impact on the hospital's operational efficiency.

2.2.2 Overview: The Chat Bot application built using Dart and Flutter leverages the power of modern technology to provide users with an innovative and user-friendly conversational experience. Utilizing Dart's programming language and Flutter's rich widget library, the application offers a seamless and visually appealing interface for users to engage with the chat bot.

2.2.3 Overall Description: The chat-bot application developed for this project is meant to be a minimal, free and open-source application that can help improve the productivity of a user without taking anything. It aims to solve at least some of the problems discussed earlier in this report.

2.3 Specific Requirements:

A basic understanding of Flutter and Dart.

A code editor. eg: Visual Studio Code or Android Studio

2.3.1 Operating Environment:

A physical device to view the application. (Android Device used).

2.4 Hardware Interfaces

The user can communicate with application using the keyboard which is used as input and a graphic interface displayed on the user's screen.

2.5 Software System Attributes

2.5.1 Reliability:

- The user inputs should be valid.
- Normal flow of the program

METHODOLOGY

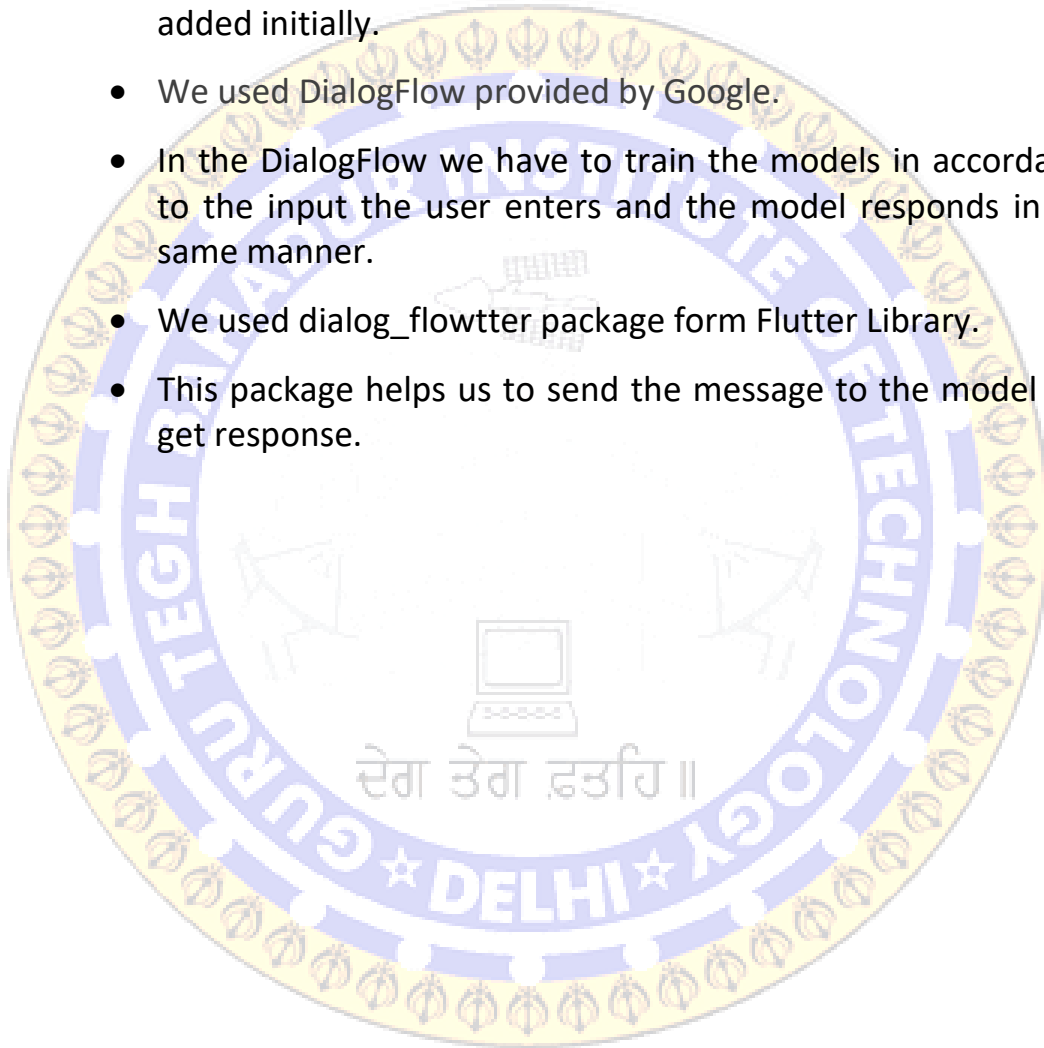
3.1 Methodology

The methodology for developing a chat-bot application entails a systematic and well-structured approach to its design, implementation, testing, and optimization. Initially, the objectives and use cases are meticulously defined, outlining the specific scenarios where the chat-bot will deliver value. Understanding the target audience follows, with a careful analysis of user characteristics and preferences. The choice of a suitable technology stack, including programming languages and frameworks, is then made, with consideration given to integrating natural language processing (NLP) and machine learning tools for advanced functionality. Conversation flows are designed, mapping out expected user journeys and defining key intents and entities for the chat-bot to recognize. A basic prototype is developed to visualize the user interface and interaction flow before moving on to the actual development phase. The chat-bot is implemented, incorporating NLP capabilities for user input comprehension and features such as context management for coherent conversations. Integration points with external systems are established to facilitate seamless data exchange, while user authentication, authorization, and security measures are implemented to protect user data and ensure compliance with regulations. Thorough testing, including unit testing and user acceptance testing, is conducted, with identified issues addressed before deployment. After deployment, monitoring tools are employed to track the chat-bot's performance, and user feedback is actively collected for iterative improvements. Documentation is created to outline functionalities, APIs, and integration points, while user training resources and support channels are provided to ensure a seamless user experience. This comprehensive methodology ensures the systematic development and continuous enhancement of a chat-bot application tailored to meet user needs and deliver optimal performance.

Explanation

5.1 Explanation :

- This Project Chat-Ly is created using Dart by using an Flutter framework to take user input and adding it to a arraylist.
- Dart is used to develop the user interface of the application
- We created a dynamic list that stores the tasks and has 2 tasks added initially.
- We used DialogFlow provided by Google.
- In the DialogFlow we have to train the models in accordance to the input the user enters and the model responds in the same manner.
- We used dialog_flowttr package form Flutter Library.
- This package helps us to send the message to the model and get response.



Conclusion And **Thought over** **Humanity**

6.1 Explanation :

In the realm of chat-bot development, it's crucial to acknowledge that effective task management surpasses basic organization and scheduling. Recent research has highlighted the challenges of managing tasks within email platforms, a common source of digital overload for users. Recognizing the integral role of email in daily tasks, a successful chat-bot application must tightly integrate with email functionalities. This ensures that the chat-bot becomes an indispensable tool, seamlessly managing tasks and facilitating efficient communication within the email ecosystem. This approach reflects a commitment to addressing contemporary challenges in productivity, aligning the chat-bot's capabilities with the evolving needs of users navigating the complexities of modern task management.

6.2 How Does the App help in Betterment of Humanity:

1. Healthcare Accessibility:

- **Telemedicine:** Chat apps support telehealth initiatives, allowing individuals to consult with healthcare professionals remotely. This is particularly beneficial for those in remote areas or facing mobility challenges, improving access to medical advice and reducing healthcare disparities.

2. Education and Learning:

- **Remote Learning:** Chat applications support virtual classrooms and online learning platforms. They provide a space for students and educators to interact, collaborate on projects, and share resources, promoting accessible and flexible education.

3. Crisis Response and Support:

- **Emergency Communication:** During crises such as natural disasters or public health emergencies, chat apps become vital for disseminating information, coordinating relief efforts, and offering support to affected individuals and communities.

4. Business and Employment Opportunities:

- **Remote Work:** Chat applications play a crucial role in remote work, enabling seamless communication and collaboration among team members regardless of geographical locations.

This contributes to a more flexible and inclusive work environment.

5. **Social Activism and Awareness:**

- **Information Sharing:** Chat apps are powerful tools for spreading awareness about social issues, mobilizing communities, and organizing activism efforts. They facilitate the rapid dissemination of information, fostering collective action for positive change.



References

7. REFERENCES:

1. Thube, S., Estate, W. I., & Naka, C. (2017). "Real-time Databases for Chat-Bot Applications." *International Research Journal of Engineering and Technology (IRJET)*, 4(06), 2078-2082.
2. Bean, M., Tilkov, S., & Vinoski, S. (2010). "Building High-Performance Chat-Bot Applications with Node.js." *IEEE Internet Computing*, 14(6), 80-83.
3. Syed, B. A. (2014). "Chat-Bot Development with Node.js." In *Beginning Node.js* (pp. 181-182). New York City: Apress.
4. Morales-Morell, A. (2001). "Usability Aspects of a Location-Aware Chat-Bot Application." *University of Puerto Rico, Mayaguez (Puerto Rico)*.
5. Gaspar, F., Taniça, L., Tomás, P., Ilic, A., & Sousa, L. (2015). "A Framework for Application-Guided Task Management in Chat-Bot Systems." *ACM Transactions on Architecture and Code Optimization (TACO)*, 12(4), 1-25.
6. Lin, C. Y., Hung, M. T., & Huang, W. H. (2012, September). "Location-Based Personal Task Management for Chat-Bot Applications in Indoor and Outdoor Environments." In *2012 15th International Conference on Network-Based Information Systems* (pp. 582-587). IEEE.

Appendix - A

Screenshots

Chat Bot

Hello

Good Morning Sir. Welcome to Safdurjung Hospital. List of Doctors that will be on duty today are: 1. Dr. Ram Kumar 2. Dr. Vinay Mathur 3. Dr. Rahul S. 4. Dr. Veena K. Please Enter the name of the doctor to know the timings, building and floor number.

Dr. Ram Kumar

Dr. Ram Kumar has his assigned duty at Block-A, 3rd Floor. 1:00 p.m. to 4:00 p.m. (Monday- Thursday).

Dr. Rahul S.

Dr. Rahul S. has his assigned duty at Block-I, 3rd Floor. 10:00 a.m. to 2:00 p.m. (Monday- Thursday).

Chat Bot

Dr. Rahul S. has his assigned duty at Block-I, 3rd Floor. 10:00 a.m. to 2:00 p.m. (Monday- Thursday).

Dr. Veena K.

Dr. Veena K. has his assigned duty at Block-U, 2nd Floor. 1:00 p.m. to 4:00 p.m. (Monday- Thursday).

Dr. Vinay Mathur

Dr. Rahul S. has his assigned duty at Block-I, 3rd Floor. 10:00 a.m. to 2:00 p.m. (Monday- Thursday).

Thank You!

Chat Bot

Thank You!

What was that?

Dr. Saiyam

Dr. Rahul S. has his assigned duty at Block-I, 3rd Floor. 10:00 a.m. to 2:00 p.m. (Monday- Thursday).

Dr. harsh

Dr. Rahul S. has his assigned duty at Block-I, 3rd Floor. 10:00 a.m. to 2:00 p.m. (Monday- Thursday).

Appendix - B

Source Code

Code:

1. Main.dart

```
import 'package:dialog_flowtter/dialog_flowtter.dart';  
import 'package:flutter/material.dart';  
import 'package:new_chat_bot_app/messages.dart';
```

```
void main() => runApp(MyApp());
```

```
class MyApp extends StatelessWidget {  
  const MyApp({super.key});
```

```
  @override
```

```
  Widget build(BuildContext context) {
```

```
    return MaterialApp(  
      title: 'Chat Bot by Saiyam',  
      theme: ThemeData(  
        brightness: Brightness.dark,  
      ),  
      home: Home(),  
    );
```

```
  }
```

```
}
```

```
class Home extends StatefulWidget {
```

```
  const Home({super.key});
```

```
  @override
```

```
  State<Home> createState() => _HomeState();
```

```
}
```

```

class _HomeState extends State<Home> {
  late DialogFlowtter dialogFlowtter;
  final TextEditingController _controller = TextEditingController();
  List<Map<String, dynamic>> messages = [];

  @override
  void initState() {
    DialogFlowtter.fromFile().then((instance) => dialogFlowtter = instance);
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Chat Bot'),
      ),
      body: Container(
        child: Column(
          children: [
            Expanded(child: MessagesScreen(messages: messages)),
            Container(
              padding: EdgeInsets.symmetric(
                horizontal: 14,
                vertical: 8,
              ),
              color: Colors.deepPurple,
              child: Row(
                children: [
                  Expanded(
                    child: TextField(

```

```

        controller: _controller,
        style: TextStyle(color: Colors.white),
      ),
    ),
    IconButton(onPressed: () {
      sendMessage(_controller.text);
      _controller.clear();
    }, icon: Icon(Icons.send)),
  ],
),
),
],
),
),
);
}

```

```

sendMessage(String text) async {
  if (text.isEmpty) {
    print('Message is Empty');
  } else {
    setState(() {
      addMessage(Message(text: DialogText(text: [text])), true);
    });
  }
}

```

```

DetectIntentResponse response = await dialogFlowtter.detectIntent(
  queryInput: QueryInput(text: TextInput(text: text)));

```

```

if(response.message == null) return;
setState(() {
  addMessage(response.message!);
}

```



```

    });
  }
}

```

```

addMessage(Message message, [bool isUserMessage = false]) {
  messages.add({'message': message, 'isUserMessage': isUserMessage});
}
}

```

2. message.dart

```
import 'package:flutter/material.dart';
```

```

class MessagesScreen extends StatefulWidget {
  final List messages;
  const MessagesScreen({super.key, required this.messages});

  @override
  State<MessagesScreen> createState() => _MessagesScreenState();
}

```

```

class _MessagesScreenState extends State<MessagesScreen> {
  @override
  Widget build(BuildContext context) {
    var w = MediaQuery.of(context).size.width;
    return ListView.separated(
      itemBuilder: (context, index) {
        return Container(
          margin: EdgeInsets.all(10),
          child: Row(
            mainAxisAlignment: widget.messages[index]['isUserMessage']
              ? MainAxisAlignment.end
              : MainAxisAlignment.start,

```

```

children: [
  Container(
    padding: EdgeInsets.symmetric(vertical: 14, horizontal: 14),
    decoration: BoxDecoration(
      borderRadius: BorderRadius.only(
        bottomLeft: Radius.circular(
          20,
        ),
        topRight: Radius.circular(20),
        bottomRight: Radius.circular(
          widget.messages[index]['isUserMessage'] ? 0 : 20),
        topLeft: Radius.circular(
          widget.messages[index]['isUserMessage'] ? 20 : 0),
      ),
      color: widget.messages[index]['isUserMessage']
        ? Colors.grey.shade800
        : Colors.grey.shade900.withOpacity(0.8)),
    constraints: BoxConstraints(maxWidth: w * 2 / 3),
    child:
      Text(widget.messages[index]['message'].text.text[0])),
  ],
),
);
},
separatorBuilder: (_, i) => Padding(padding: EdgeInsets.only(top:
10)),
itemCount: widget.messages.length);
}
}

```

3. pubspec.yaml

name: new_chat_bot_app

description: A new Flutter project.

The following line prevents the package from being accidentally published to

pub.dev using `flutter pub publish`. This is preferred for private packages.

publish_to: 'none' # Remove this line if you wish to publish to pub.dev

version: 1.0.0+1

environment:

sdk: '>=2.19.0-444.1.beta <3.0.0'

versions available, run `flutter pub outdated`.

dependencies:

cupertino_icons: ^1.0.2

dialog_flowttr: ^0.3.3

flutter:

sdk: flutter

dev_dependencies:

flutter_lints: ^2.0.0

flutter_test:

sdk: flutter

For information on the generic Dart part of this file, see the

The following line ensures that the Material Icons font is

included with your application, so that you can use the icons in

the material Icons class.

uses-material-design: true

```

# To add assets to your application, add an assets section, like this:
assets:
  - assets/

# An image asset can refer to one or more resolution-specific "variants",
see
# https://flutter.dev/assets-and-images/#resolution-aware
# For details regarding adding assets from package dependencies, see
# https://flutter.dev/assets-and-images/#from-packages
# To add custom fonts to your application, add a fonts section here,
# in this "flutter" section. Each entry in this list should have a
# "family" key with the font family name, and a "fonts" key with a
# list giving the asset and other descriptors for the font. For
# example:
# fonts:
#   - family: Schyler
#     fonts:
#       - asset: fonts/Schyler-Regular.ttf
#       - asset: fonts/Schyler-Italic.ttf
#         style: italic
#   - family: Trajan Pro
#     fonts:
#       - asset: fonts/TrajanPro.ttf
#       - asset: fonts/TrajanPro_Bold.ttf
#         weight: 700
#

```

