A PROJECT REPORT

on

TRIP ASSITANT CHATBOT

Submitted by

Mr. Gautam Jagdish Chand Verma

in partial fulfillment for the award of the degree

of

BACHELOR OF SCIENCE

in

COMPUTER SCIENCE

under the guidance of

Dr. Rupesh Sudhakar Patil, M.Sc, Ph.D.
Head, Department of Computer Science

B. K. Birla Night Arts, Science and Commerce College, Kalyan
(Affiliated to University of Mumbai)

(Sem V)

(2023 - 2024)

KalyanCitizens'Education Society's

B. K. Birla Night Arts, Science and Commerce College, Kalyan

(Affiliated to University of Mumbai)

B. K. BIRLA COLLEGE ROAD, KALYAN (WEST) - 421301

Department of Computer Science

CERTIFICATE

This is to certify that **Mr. Gautam Jagdish Verma** of **T.Y.B.Sc. (Sem V)** class has satisfactorily completed the Project entitled **“Trip Assitant Chatbot”**, submitted towards partial fulfillment for the award of **Bachelor of Science** in **Computer Science** during the academic year **2023-2024.**

Date of Submission:

Project Guide

Head / Incharge,
Department of Computer Science

College Seal

Signature of Examiner

DECLARATION

I, **Gautam Jagdish Verma**, hereby declare that the project entitled **“Trip Assistant Chatbot”** Submitted towards partial fulfillment for the award of **Bachelor of Science** in **Computer Science** during the academic year **2023-2024** is my original work and the project has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

Place: Kalyan **Signature of the Student**

Date:

Acknowledgement

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

I am highly indebted to **Dr. Rupesh Sudhakar Patil** sir for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards all the respected teachers & Head of the Department **Dr. Rupesh Sudhakar Patil** sir of computer science department and principal **Dr. Harish Dubey** sir of **B. K. Birla Night Arts, Science and Commerce College, Kalyan** for their kind cooperation and encouragement which help me in completion of this project.

Mr. Gautam Jagdish Verma

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION 9
1.1 Background..9
1.2 Objectives 10
1.3 Purpose, Scope, and Applicability 10
1.3.1 Purpose 10
1.3.2 scope 10
1.3.3 Applicability..11
1.4 Achievements..11
1.5 Organization of Report 11
CHAPTER 2: SURVEY OF TECHNOLOGIES..13
CHAPTER 3: REQUIREMENTS AND ANALYSIS 19
3.1 Problem Definition 19
3.2 Requirements Specification 19
3.3 Planning and Scheduling 25
3.4 Software and Hardware Requirements..25
3.5 Preliminary Product Description 25
3.6 Conceptual Models 27
CHAPTER 4: SYSTEM DESIGN 29
4.1 Basic Modules 29
4.2 Data Design..29
4.2.1 Schema Design..30
4.2.2 Data Integrity and Constraints..31
4.3 Procedural Design 3..32
4.3.1 Logic Diagrams 3..33
4.3.2 Data Structures 34
4.3.3 Algorithms Design 34
4.4 user interface design 38
4.5 Security Issues 38

4.6 Test Cases Design 39

CHAPTER 5: IMPLEMENTATION AND TESTING 41

5.1 Implementation Approaches..........41

5.2 Coding Details and Code Efficiency 33

5.2.1 Code Efficiency..........33

5.3 Testing Approach..........33

5.3.1 unit Testing 33

5.3.2 Integrated Testing..........33

5.3.3 Beta Testing 34

5.4 Modifications and Improvements 34

5.5 Test Cases..........34

CHAPTER 6: RESULTS AND DISCUSSION 35

6.1 Test Reports..........35

6.2 user Documentation..........36

CHAPTER 7: CONCLUSIONS 36

7.1 Conclusion 37

7.1.1 Significance of the System 37

7.2 Limitations of the System 37

7.3 Future Scope of the Project..........37

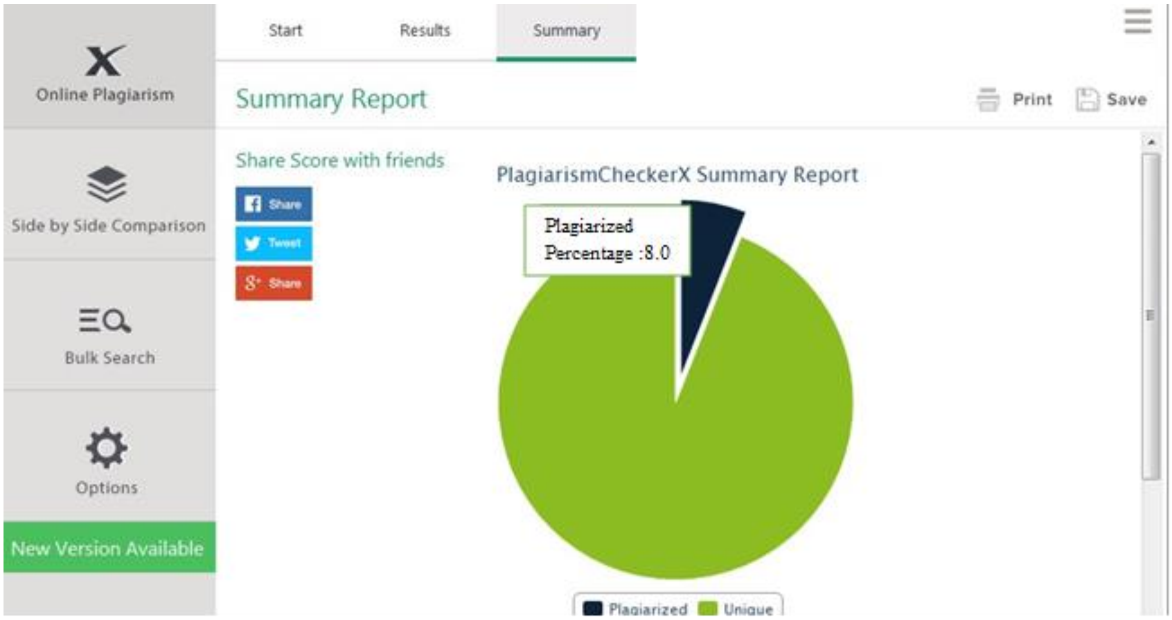
List of Tables

Sr No.	Table No.	List Of Tables	Page No.
1 .	6.1	Test Reports	35

List of Figures

Sr No.	Figure No.	List Of Figure	Page No.
1.	3.3	Gant Chart	25
2.	3.6	Flow Chart	27
3.	4.1	Use Case Diagram	31
4.	4.2	Activity Diagram	36
5.	4.3	Collaboration Diagram	37
6.	4.4	Class Diagram	41
7.	4.5	Object Diagram	43
8.	4.6	State Diagram	43

Plagiarism Report



Plagiarized Percentage :8.0

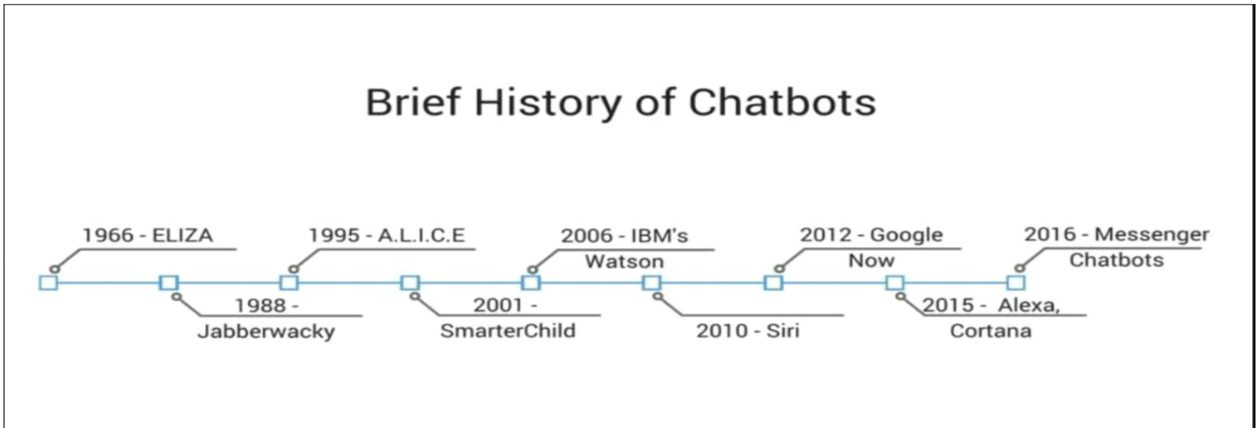
CHAPTER 1

INTRODUCTION

The trip assistant chatbot(Tripzo) is the android application which work like a virtual travel agent which help users to planned their trip in efficient manner And simplifying the process of booking and other queries.

- ## Background

With the increasing the number of mobile messaging user it expected() to surpass 2+ billion users, more and more business consider chat apps to be a new field of the opportunity. Largely due to the staggering growth of mobile messaging platforms, the new trend -Chatbots- is gearing up. Due to advancement of the technology and advances in AI(artificial intelligence) it mostly popular among us. A chatbot is a conversational agent where a computer program is designed to stimulate an intelligent conversation. Where it can take user input in many formats such as like text, voice, sentiments, etc for this purpose many open-source platforms are available. The more you interact with such a bot, the better it can learn your preferences and tailor the experience accordingly.



Going back in history of chatbot it traced back to 1960's. The first chatbot was created in the 1966 by joseph Weizenbaum. And the name of the first chatbot is "ELIZA" which is used by psychotherapist which recognize the certain keyword and pattern to answer accordingly, mimicking the conversation with the psychotherapist. In current scenarios the chatbots are capable of more than simply mimicking human speech and using of present scenarios. Some of them can be indistinguishable from a human. They have a personality, can perfectly understand human speech and can perform a wide variety of the tasks.

Our chatbot (Tripzo) offer an alternative by using existing platforms like browser and messenger's apps that almost everyone has on their phone and make it possible for users to receive notifications without downloading separate apps for each service during the trip and help the traveller to plan their trip easily.

• Objectives

The main purpose of the trip assistant chatbot (name of our chatbot) Is to help the customer to give him only information that he needs and help him to plan his trip and make the itenary for the trip in easy manner. It also useful to both customer and travel company for the company it's boost operational efficiency and bring the cost saving for it. And for the customer it added services to internal employees and external customer and also reduce the need for human interaction.

Following are objectives –

- Easy to use
- Chatbots save time of customer and travel company
- It reduces the processual workload
- Great channel to provide a better customer care
- Cost effective solution
- Attract the new customers
- Increase sales for the travel company

• Purpose, Scope and Applicability

• Purpose

Today in many industries the chatbots have become important for the many reasons. In the tourism sector the use of chatbots is changing the experience of the travellers.

The main purpose of the chatbot is it allows business to connect with the customers in a personal way without the expense of the human representatives. The chatbots offers smooth customer journey. To plan for trip or for getting information of the place by using the chatbot it enhances the experience of the customers.

• Scope

In today's world the scope of the chatbot is more because these chatbots helps the customer to plan his trip in very efficient manner. These chatbot is build or designed for the only specific information that user wants. The chatbot can automate the low-value tasks which are focusing on those that really add value to the both company and client

Our project aims at the automation in searching of hotel and planning for the trip.

- It helps to gather the user requirements regarding the trip
- Give Instant response
- Easy to operate
- Have good user interface
- Gives the needed and proper information about the destinations.

1.3.3 Applicability

The travellers are looking through so many websites to find the best trip and search about the place which they going to visit and to plan according to it. But the chatbot gave this all information on a single platforms so by using chatbots it increase the engagement of the customer. The travel chatbots can improve the booking experience of the customer by providing them with more relevant recommendations, while enhancing the business matrices of the organization.

- Gave all information on one platform
- User-friendly
- Overcome human efforts
- Easy to use

• Achievement

There will be no need to search on search engine(google) for searching for hotel of your budget and for planning of trip. You didn't get excess unnecessary information about place you only get the information which is useful for the planning of the trip . The chatbot works as assistant for the customers that it recommends the customers for hotels and flight and places to visit etc.

• Organization of report

A short overview for the chapters which are included in this report is as followed;

- Chapter 2: Survey of technologies

  In this chapter the major focus is on the brief overview about used technologies in the project.
- Chapter3: Requirement and Analysis

  These chapter focuses on requirements what user is required or supposed from system to do. During development phase re-validating the requirements with actual requirement comparison noted in these chapter.
- Chapter 4: System Design

What the system will contained, what are the different modules are discussed in the section.

- Chapter 5; Implementation and Testing

  This chapter is about Implementation of various modules. As the project is proceeding with the testing the various results are expected during the testing of the project.

- Chapter 6: Conclusion

  The significance of project, scope for the project in the near future is all discussed in this chapter.

Chapter 2

Survey of Technologies

An Android – Application is most commonly known referred as a apps or application. Android is type of android application designed to run on any device such as a computer, tablets, wearables, set-top box, smart TVs, notebooks etc. an android application is installed onto the device itself and are developed especially for the particular mobile operating system. These apps are available on the apps store such as Apple app store, Google Play Store, etc. The android apps work in the specific mobile operating systems such as an Apple iOS or android OS. If an app made for Android OS, then it will not work on Apple iOS or Windows OS. We have to build separate apps for each operating system.

How Android application work?

The android operating system is the largest installed base among the various platforms across the globe. Because of its user-friendly, having huge community support it provides a greater extent on customization and had large number of companies build Android-compatible smartphones. To develop an android application, it involves the several process that happen in the sequential manner. After writing source code files when developer click on Run button on android studio there are plenty of operations start at the background.

The Android apps can be written using Kotlin, Java and C++ language. The Android SDK tools compiled the code along with any data and resource files into an APK or an Android App Bundle. An android package, which is an archive file with an .apk suffix which contains the content of an android app that are required at runtime and it is the file that android – powered devices use to install the app.

Benefits

Android application have many different uses and with those uses, comes many potential benefits. some common benefits of Web-apps include:

- An android is an open-source project so we can customize the OS based on our requirements.
- Android supports different types of connectivity for GSM, CDM, Wi-Fi, Bluetooth etc. for conversation or data transfer.
- It contains multiple APIs to support location- tracking service such as GPS.
- It provides support for virtual reality 2D/3D Graphics.
- Android has an integrated open-source Web Kit layout-based web browser to support user interface like HTML5 and CSS3.

Available Technologies

Following are the technologies available for this project. These are separate technologies for the application

For development of app

FRONT END/GUI TOOLS	.Net technologies, XML (Extensible Markup Language), HTML, CSS
DBMS/BACK END	Oracle, MySQL, SQL Server
LANGUAGES	C++, Java, C#, R, Python, Kotlin
SCRIPTING LANGUAGES	JavaScript, JSON (API creation)
.NET Platform	C#.Net, Visual C#, ASP.Net

FRONT END/GUI TOOLS

- Net Technologies

The .NET is nothing but it is another name of facilities. So, these facilities are utilized by the beginners, Developers as well as by the expert also. To develop android application visual studio, provide Xamarine cross platform service.

Which use the integrated development environment (IDE). The .NET is the technology from Microsoft, on which all other Microsoft technologies will depending in future. It's major technology change introduced by Microsoft, to catch the market from SUN's java.

- XML

XML (Extensible Markup Language). The XML is a markup language much like a HTML which is used to describe data. It is derived from Standard Generalized Markup Language (SMGL). Basically, the XML tags are not predefined in XML. We need to implement and define the tags in it. The XML is used to implement UI- related data and its lightweight markup language that doesn't make layout heavy.

- HTML
- CSS

DBMS/BACK-END

- Oracle

Oracle is one of the largest vendors in the enterprise IT market and the shorthand name of its flagship product, a relational database management system (RDBMS) that's formally called Oracle Database. The database software sits at the center of many corporate IT environments, supporting a mix of transaction processing, business intelligence and analytics applications.

- MySQL

MySQL is an open-source relational database management system (RDBMS). Its name is combinations of ―My‖, the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL is a component of the LAMP web application software stack (and others), which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python.

- SQL Server

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).

Language

- C++

C++ is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation. C++ is a general-purpose programming language created by Bjarne Stroustrup as an extension of the C programming language, or "C with Classes". The language has expanded significantly over time, and modern C++ has object-oriented, generic, and functional features in addition to facilities for low-level memory manipulation. It is almost always implemented as a compiled language, and many vendors provide C++ compilers, including the Free Software Foundation, LLVM, Microsoft, Intel, and IBM, so it is available on many platforms.

- JAVA

Java is a general-purpose computer-programming language that is concurrent, class based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java bytecodes the platform-independent codes interpreted by the interpreter on the Java platform.

- Python

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently. It is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python is a MUST for students and working professionals to become a great Software Engineer especially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python.

Chosen Technologies

Following are the technologies which have been selected for the project.

FRONT END/GUI TOOLS	XML (extensible markup language)
DBMS/BACK END	Java
LANGUAGES	Java
API	Brainshop API

Justification for selection of technologies

XML (extensible markup language)

The XML used to implement the UI – related data. So, understand the core part of the UI interface with respect to XML is so important. It is a lightweight markup language which does not make layout heavy and the XML only contains tags, while implementing they need to be just invoked. The android provides three types of XML parsers which are DOM, SAX and XMLPullParser. Among all of them it recommends XMLPullParser because it is efficient and easy to use. So, we are going to use XMLPullParsing XML.

Java

Java is an easy to learn these language is easy to learn and has rich API, xml parsing, database connection, networking, utilities and it provide almost everything that a developer can expect. It is also open source and avalible free. So, the Java gives the best option for development of mobile applications that are based on Android consist of its own APIs and JAVA libraries. So, for the android applications you can use the JAVA to write code for android apps.

Brainshop API

Brainshop.ai is a cloud-based API (Application Programming Interface) that provides natural language processing (NLP) capabilities for creating chatbots, virtual assistants, and other conversational interfaces. The API allows developers to create custom chatbots that can understand and respond to natural language input from users.

The Brainshop.ai API offers a range of features, including:

Natural language understanding (NLU): The API uses advanced NLP algorithms to understand the intent and context of user input. It can recognize entities, extract key information, and provide appropriate responses based on the user's query.

Chatbot creation: Developers can use the API to create custom chatbots for their websites or applications. The API offers a range of tools for designing, testing, and deploying chatbots.

Integration with messaging platforms: The API can be integrated with popular messaging platforms such as Facebook Messenger, Slack, and WhatsApp, allowing users to interact with chatbots directly within those platforms.

Customization: The API allows developers to customize the chatbot's responses, create custom scripts, and tailor the chatbot's behavior to meet the specific needs of their application.

To use the Brainshop.ai API, developers need to sign up for an account and obtain an API key. Once they have access to the API, they can integrate it into their applications using the API's documentation and developer tools.

CHAPTER 3

REQUIREMENT AND ANALYSIS

3.1 Problem Definition:

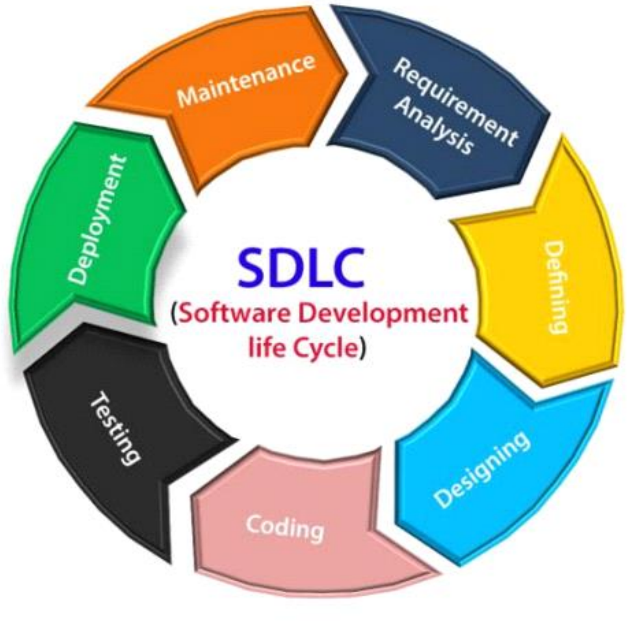
- Tourists have to spend a lot of time searching for an appropriate flight, hotel or restaurant.
- They face problems due to scattered data on websites and unable to find user-friendly interfaces.
- Being in a foreign land, they feel helpless in case of emergency or a panic situation.
- Their travel experience becomes miserable if they don't have anyone to guide them like a friend.
- There's no assistant to deal with tourists' problems such as medical emergency, currency exchange, etc

This project purposes to develop Android-application on travel assistant chatbot. As the internet is becoming world’s largest communication , it has provided an environment where everyone can be online at any given time. It will suggest the events (cultural or others) happening in the city and provide information about entry passes. The bot will be able to provide directions to go to a place by fetching data from API’s. (Recently, a tourist couple had medical problem while roaming around and they couldn’t find a hospital) it will be able to give the contact information of nearby police and hospital authorities.

3.2 Requirement Specification

A software requirements specification (SRS) is a description of a software system to be evolved. The software requirements specification consists of functional and non-functional requirements. It additionally includes use cases that describe consumer interactions that the software must provide to it. The software necessities specification report lists sufficient and essential requirements for the project development. For this the developer desires to have clear and thorough knowledge of the software program. this is executed through special and non-stop communication with the assignment crew and purchaser and analyzing each and every activity of project in detail.

Requirement specification or system analysis is part of Software Development life cycle in which we need to determine how the current information system functions and assess what users like to see in a new system. It has two subphases, viz. requirement determination and requirement structuring.



3.2.1 Functional requirements

Functional requirements are essential to your product because, as they state, they provide some sort of functionality.

Ask yourself the question "does this add to my tool's functionality?" Or "What function does this provide?" can help with this process. Within Medical devices especially, these functional requirements may have a subset of risks and requirements.

You may also have requirements that outline how your software will interact with other tools, which brings us to external interface requirements.

3.2.2 External Interface Requirements

External interface requirements are specific types of functional requirements. These are especially important when working with embedded systems. They outline how your product will interface with other components.

There are several types of interfaces you may have requirements for, including:

- User
- Hardware
- Software
- Communications
- System Features

System features are types of functional requirements. These are features that are required for a system to function.

3.2.3 Other Non-functional Requirements

Non-functional requirements can be just as important as functional ones.

These include:

- Performance
- Safety
- Security
- Quality

The importance of this type of requirement may vary depending on your industry. In the medical device industry, there are often regulations that require the tracking and accounting of safety.

3.3 Planning and scheduling

3.3.1 Planning

Project planning is a procedural step in project management, where required documentation is created to ensure successful project completion. Documentation includes all actions required to define, prepare, integrate and coordinate additional plans. The project plan clearly defines how the project is executed, monitored, controlled and closed. The project planning stage requires several inputs, including conceptual proposals, project schedules, resource requirements/limitations and success metrics. Project planning begins by setting the scope of a project and eventually working through each level of dependent action, tasks, checkpoints and deadlines. Project planning gives us an idea about what all activities are required to complete the project. It provides obstacles that arrive in the project such as non-availability of resources and so on. Project planning involves defining the roles and responsibilities of the project management team members. It ensures that the team members work according to the laid objectives.

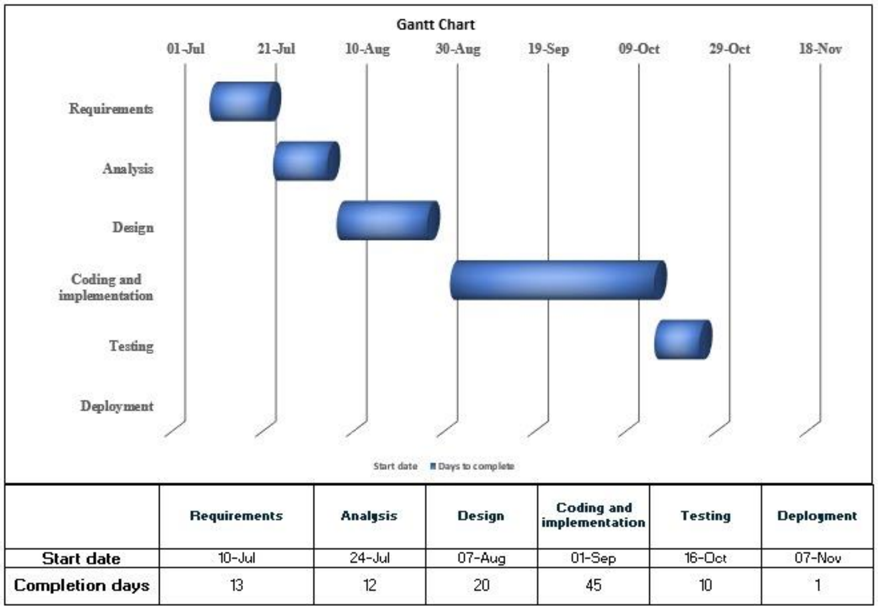
A project is made up of many tasks and each task is given a start and end date so it can be completed in time. The team members will have different schedules and their availability and vacation or leave dates need to be documented in order to successfully plan those tasks. Nowadays most of the teams use online project scheduling tools. In short scheduling is just one

of the features in a larger project management software solution and there are many places in software where scheduling takes place.
Our project is aimed to be completed within seven to eight months. Plan is made in such that half of the project that is up to designing phase it is to be completed in September. Remaining part will be started in November will continue up to February. Before designing, requirement and analysis activity is to be carried out. It includes gathering all the required information about the project either by interacting with the users who are going to use or by some research and studying different surveys. We have gathered all information about our project from different users who are going to use this. In designing, we have done designing of some basic modules which includes input output screen, and environment of chatbot for android application.

3.3.2 Scheduling

The project schedule is the tool that communicates what work needs to be performed, which resources of the organization will perform the work and the timeframes in which that work needs to be performed. The project schedule should reflect all of the work associated without a full and complete schedule, the project manager will be unable to communicate the complete effort, in terms of cost and resources, necessary to deliver the project. The project schedule can be viewed and updated by team members associated with the project, keeping everyone well informed on the overall project status. Gantt chart shows our project schedule.

3.3.3 Gantt Chart



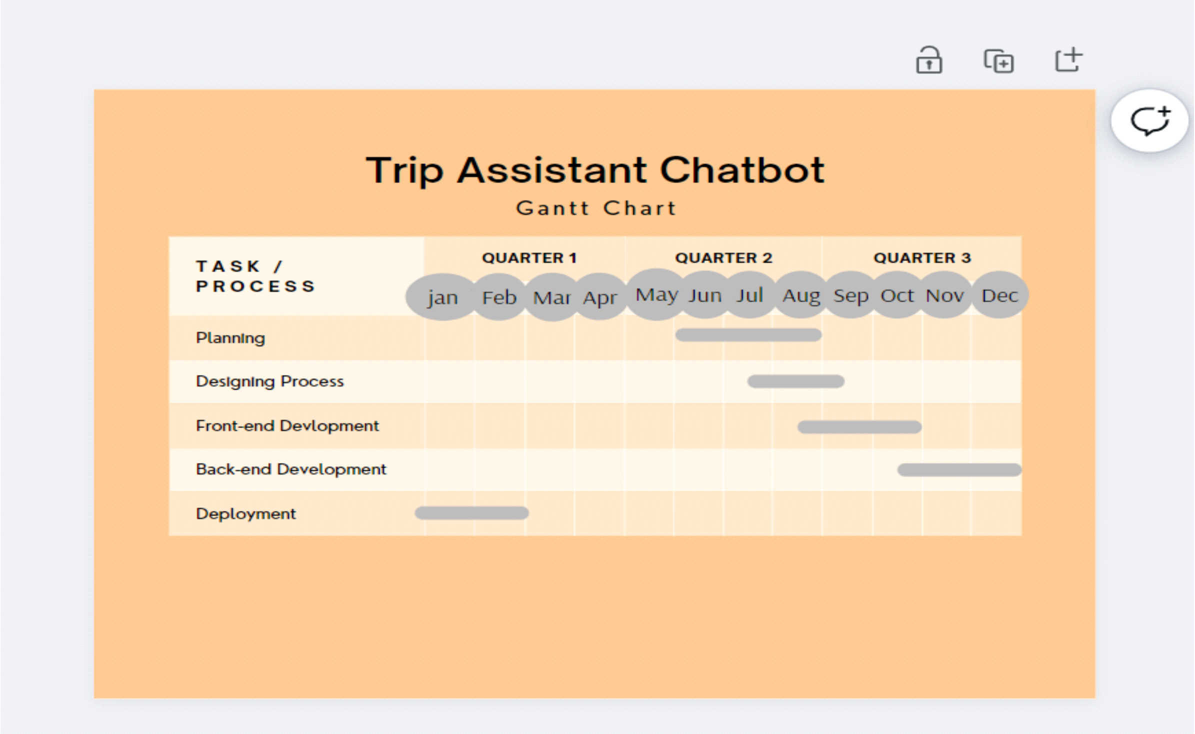
	Requirements	Analysis	Design	Coding and implementation	Testing	Deployment
Start date	10-Jul	24-Jul	07-Aug	01-Sep	16-Oct	07-Nov
Completion days	13	12	20	45	10	1

In the Gantt chart we show the time spent for each phase of the software development. Gantt charts are a project-planning tool that can be used to represent the timing of tasks required to complete a project. Because Gantt charts are simple to understand and easy to construct, they are used by most project managers for all but the most complex projects.

In a Gantt chart, each task takes up one row. Dates run along the top in increments of days, weeks or months, depending on the total length of the project. The expected time for each task is represented by a horizontal bar whose left end marks the expected beginning of the task and whose right end marks the expected completion date. Tasks may run sequentially, parallel or overlapping. As the project progresses, the chart is updated by filling in the bars to a length proportional to the fraction of work that has been accomplished on the task. This way, one can get a quick reading of project progress by drawing a vertical line through the chart at current date. Completed tasks lie to the left of the line and are completely filled in. Current tasks cross the line and are behind schedule if their filled-in section is to the left of the line and ahead of schedule if the filled-in section stops to the right of the line. Future tasks lie completely to the right of the line. In constructing a Gantt chart, keep the tasks to a manageable number (no more than 15 or 20) so that the chart fits on a single page. More complex projects may require subordinate charts which detail the timing of all the subtasks which make up one of the main tasks.

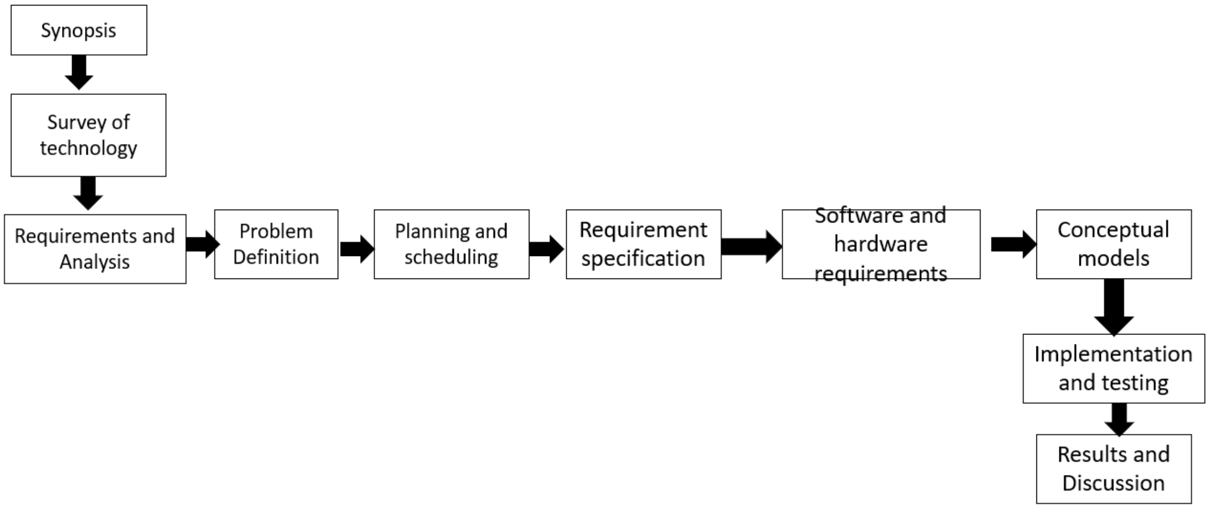
On a Gantt chart one can easily see:
- The start date of the project
- What the project tasks are
- When tasks start and finish
- How long each task will take
- How tasks group together, overlap and link with each other
- The finish date of the project.

For team projects, it often helps to have an additional column containing numbers or initials, which identify that on the team is responsible for the task. Often the project has important events, which you would like to appear on the project timeline, but which are not tasks. For example, you may wish to highlight when a prototype is complete or the date of a design review. We planned our project using according to the Gantt chart as shown as follows:



3.3.4 PROGRAM EVALUTION REVIEW TECHNIQUE (PERT)

The program (or project) evaluation and review technique (PERT) is a statistical tool used in project management, which was designed to analyse and represent the tasks involved in completing a given project. Figure shows PERT diagram for activities. PERT diagram shows activity no, target date, expected date and standard deviation of each activity. The activities involved are search for project, project approval, planning, requirement & analysis, documentation, designing, designing complete, database designing, documentation complete, coding & implementation and last testing.


Synopsis
Survey of technology
Requirements and Analysis
Problem Definition
Planning and scheduling
Requirement specification
Software and hardware requirements
Conceptual models
Implementation and testing
Results and Discussion

3.4 Software and hardware requirements

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as (computer) system requirements. Most software defines two sets of system requirements: minimum and recommended.

3.4.1 Software

- Programming Language: java, c++, python.
- Web Technology: kotlin, google.
- Operating System: Windows 10, Windows 11.
- Database: SQL Server.
- Software: Android Studio.

3.4.2 Hardware

- Processor – i5.
- Hard disk-
- Memory – 4GB RAM.
- Processor Speed – Dual Core 2.40 GHz.

3.5 Conceptual Models

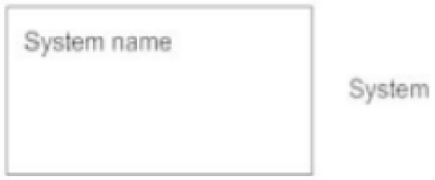
In the realm of software development, a conceptual model may be used to represent relationships of entities within a database. A conceptual model can easily represent abstract concepts of the relationships between objects in the system, such as Users and their relationships to accounts.

3.5.1 Use Case

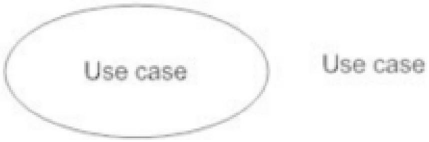
A use case diagram is a dynamic or behaviour diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. In this context, a "system" is something being developed or operated, such as a web site. The "actors" are people or entities operating under defined roles within the system. Use case diagrams are valuable for visualizing the functional requirements of a system that will translate into design choices and development priorities. They also help identify any internal or external factors that may influence the system and should be taken into consideration.

BASIC USE CASE DIAGRAM SYMBOLS

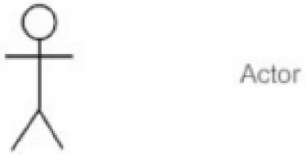
- **System**
  Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.



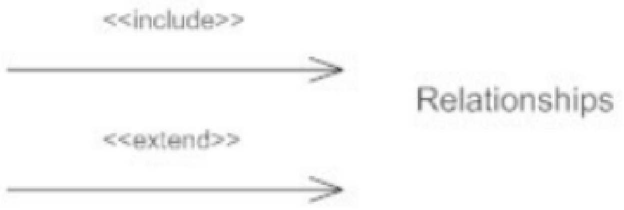
- **Use case**
  Draw use cases using ovals. Label the ovals with verbs that represent the system's Functions.



- **Actors**
  Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.



- **Relationships**
  Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labelled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.

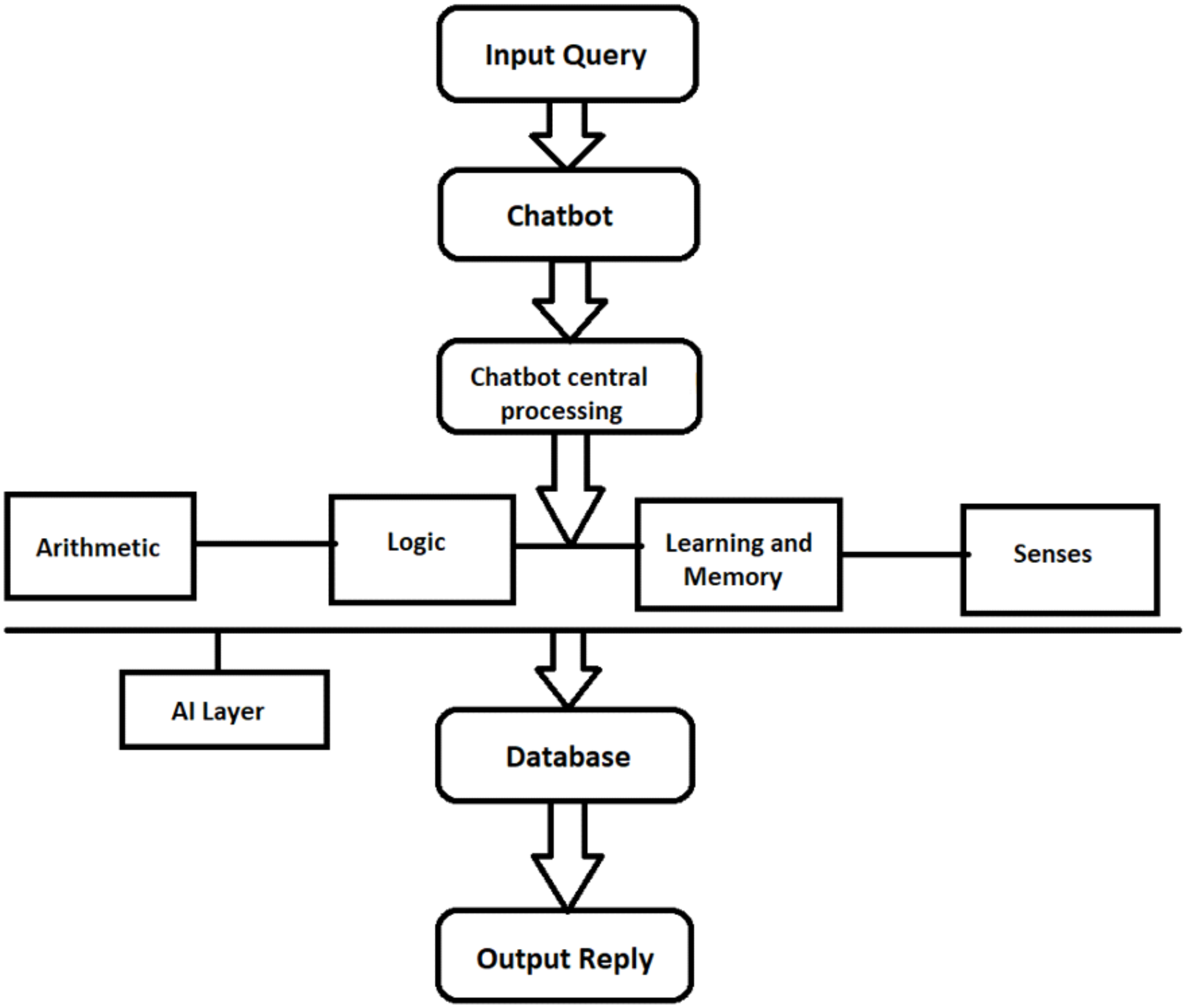


3.5.2 Data Flow Diagram

A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various sub processes the data moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationships. You can use these diagrams to map out an existing system and make it better or to plan out a new system for implementation. Visualizing each element makes it easy to identify inefficiencies and produce the best possible system. Data flow diagram levels Data flow diagrams are also categorized by level. Starting with the most basic, level 0, DFDs get increasingly complex as the level increases. As you build your own data flow diagram, you will need to decide which level your diagram will be. Level 0 DFDs, also known as context diagrams, are the most basic data flow diagrams. They provide a broad view that is easily digestible but offers little detail.

- Level 0: Data flow diagrams show a single process node and its connections to external entities.
- Level 1: DFDs are still a general overview, but they go into more detail than a context diagram. In a level 1 data flow diagram, the single process node from the context diagram is broken down into subprocesses. As these processes are added, the diagram will need additional data flows and data stores to link them together.
- Level 2: DFDs simply break processes down into more detailed subprocesses. In theory, DFDs could go beyond level 3, but they rarely do. Level 3 data flow diagrams are detailed enough that it doesn't usually make sense to break them down further.

Data Flow Diagram


Input Query
Chatbot
Chatbot central processing
Arithmetic
Logic
Learning and Memory
Senses
AI Layer
Database
Output Reply

CHAPTER 4

SYSTEM DESIGN

4.1 Basic Modules

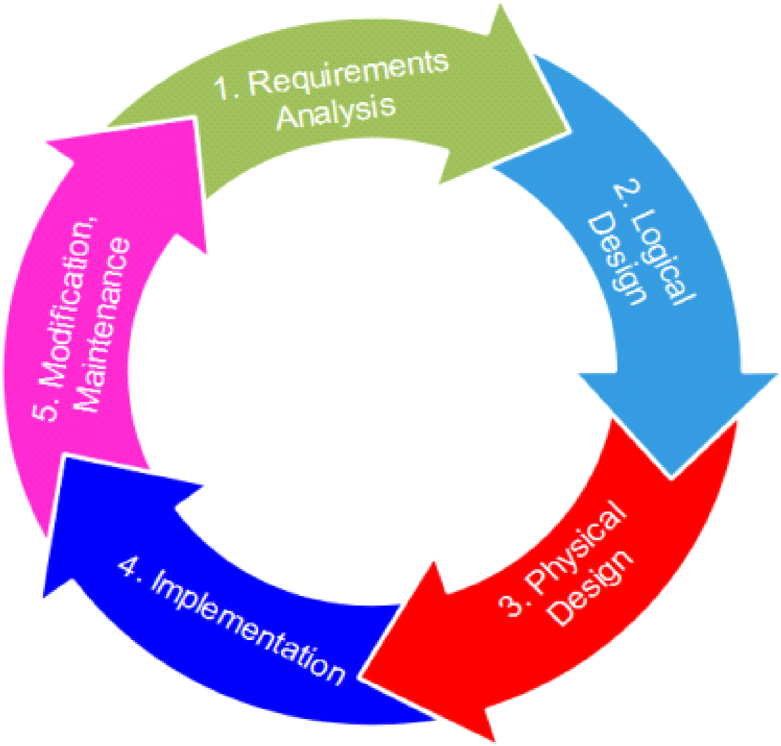
A module is a software component or part of a program that contains one or more routines. One or more independently developed modules make up a program. An enterprise-level software application may contain several different modules, and each module serves unique and separate business operations.

Modules make a programmer's job easy by allowing the programmer to focus on only one area of the functionality of the software application. Modules are typically incorporated into the program (software) through interfaces.

4.2 Data Design

What is database?

A database is a collection of information that is organized so that it can be easily accessed, managed and updated. Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added. Databases process workloads to create and update themselves, querying the data they contain and running applications against it.



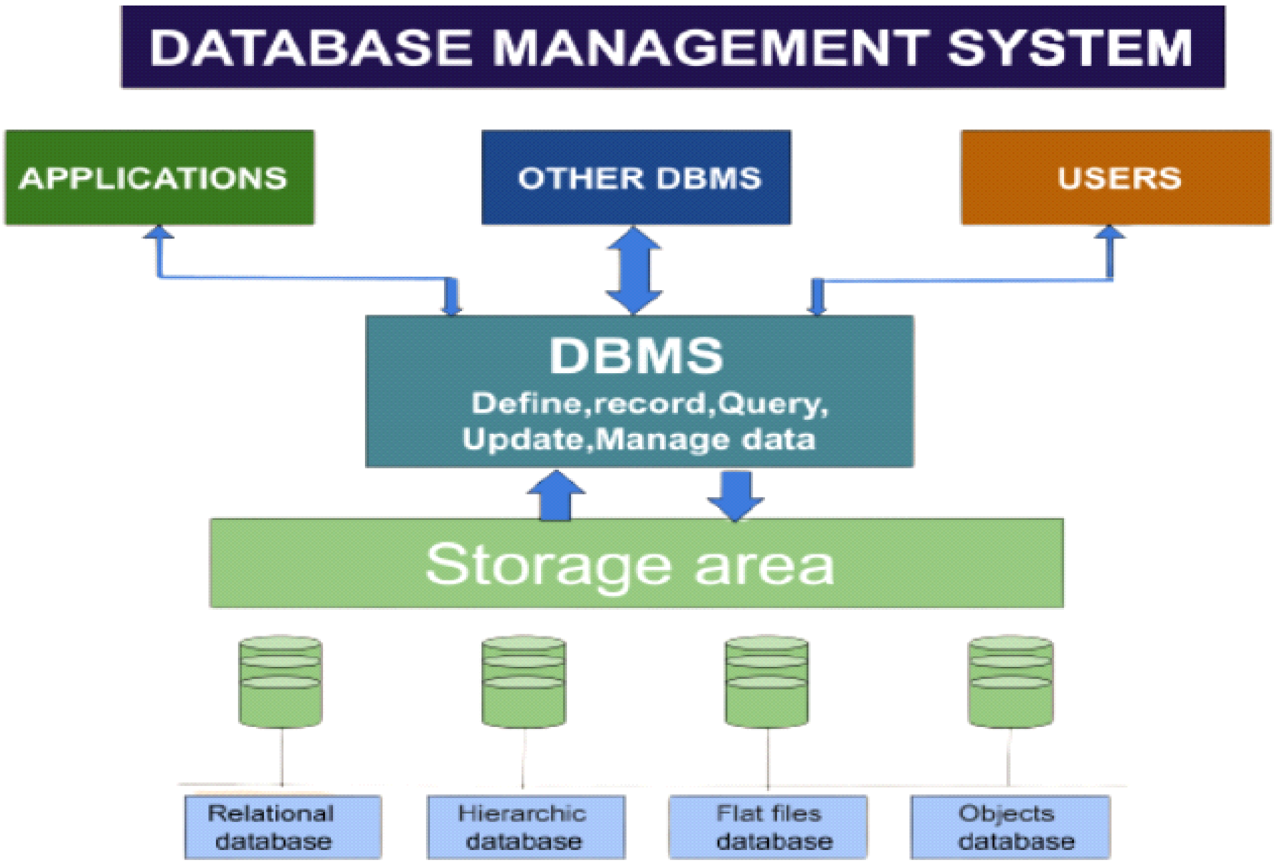
As stated above, a database system is a fundamental component of the larger enterprise information system. The database development life cycle (DDLC) is a process of designing, implementing and maintaining a database system to meet strategic or operational information needs of an organization or enterprise such as:

- Improved customer support and customer satisfaction.
- Better production management.
- Better inventory management.
- More accurate sales forecasting

The database development life cycle (DDLC) is inherently associated with the software development life cycle (SDLC) of the information system. DDLC goes hand-in-hand with the SDLC and database development activities starts right at the requirement phase.

4.2.1 Schema design

The database schema of a database system is its structure described in a formal language supported by the database management system (DBMS). The term "schema" refers to the organization of data as a blueprint of how the database is constructed.



4.2.2 Data integrity and constraints

DATA INTEGRITY

Data integrity is the maintenance of, and the assurance of the accuracy and consistency of, data over its entire lifecycle and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data. The term is broad in scope and may have widely different meanings depending on the specific context even under the same general umbrella of computing it is at times used as a proxy term for data quality while data validation is a pre-requisite for data integrity. Data integrity is the opposite of data corruption the overall intent of any data integrity technique is the same: ensure data is recorded exactly as intended (such as a database correctly rejecting mutually exclusive possibilities) and upon later retrieval, ensure the data is the same as it was when it was originally recorded. In short, data integrity aims to prevent unintentional changes to information. Data integrity is not to be confused with data, the discipline of protecting data from unauthorized parties.

Data integrity is enforced in both hierarchical and relational database models. The following three integrity constraints are used in a relational database structure to achieve data integrity.

- Entity Integrity
  This is concerned with the concept of primary keys. The rule states that every table must have its own primary key and that each has to be unique and not null.
- Referential Integrity
  This is the concept of foreign keys. The rule states that the foreign key value can be in two states. The first state is that the foreign key value would refer to a primary key value of another table, or it can be null. Being null could simply mean that there are no relationships, or that the relationship is unknown.
- Domain Integrity
  This states that all columns in a relational database are in a defined domain.

CONSTRAINT

Constraints are rules created at design-time that protect our data from becoming corrupt. It is essential for the long-time survival of our heart child of a database solution. Without constraints solution will definitely decay with time and heavy usage. A wise man once said: “Data must protect itself!” And this is what constraints do. It is rules that keep the data in your database as valid as possible. There are many ways of doing this. Following are some of the ways to do so. Foreign key constraints are probably the most used constraint, and ensure that references to other tables are only allowed if there actually exists a target row to reference. This also makes it impossible to break such a relationship by deleting the referenced row creating a dead link.

TYPES OF CONSTRAINTS

Constraints can be divided into the following two types,

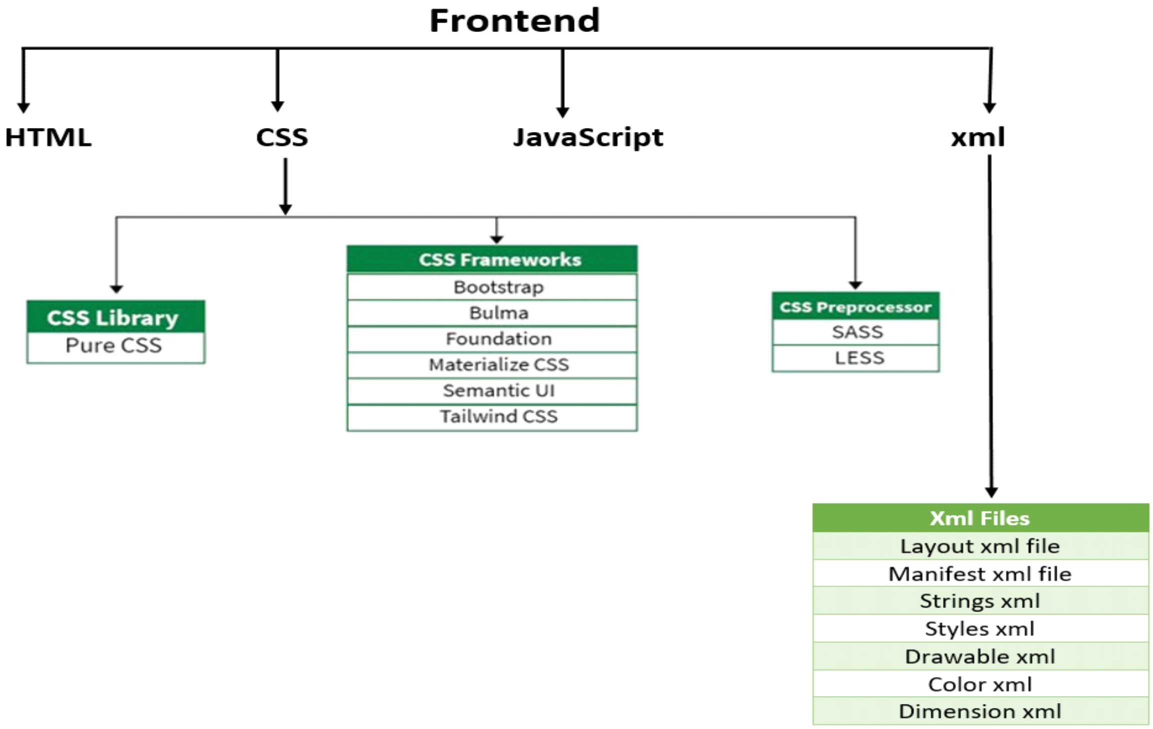
- Column level constraints: Limits only column data.
- Table level constraints: Limits whole table data.

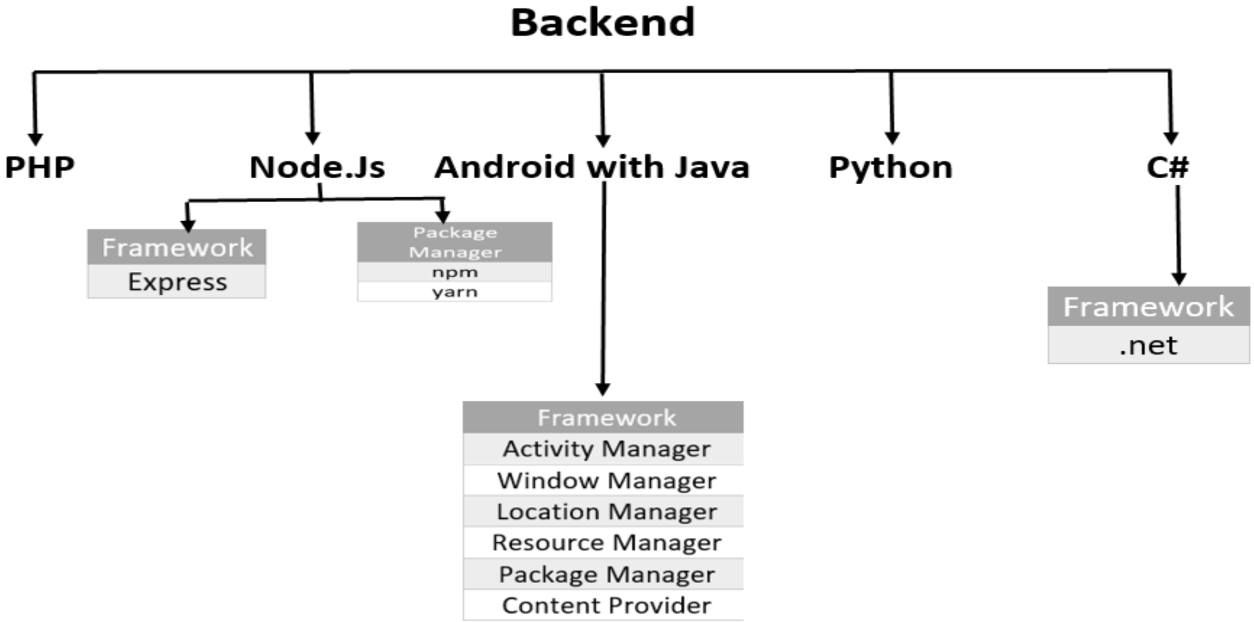
Constraints are used to make sure that the integrity of data is maintained in the database.

4.3 Procedural design

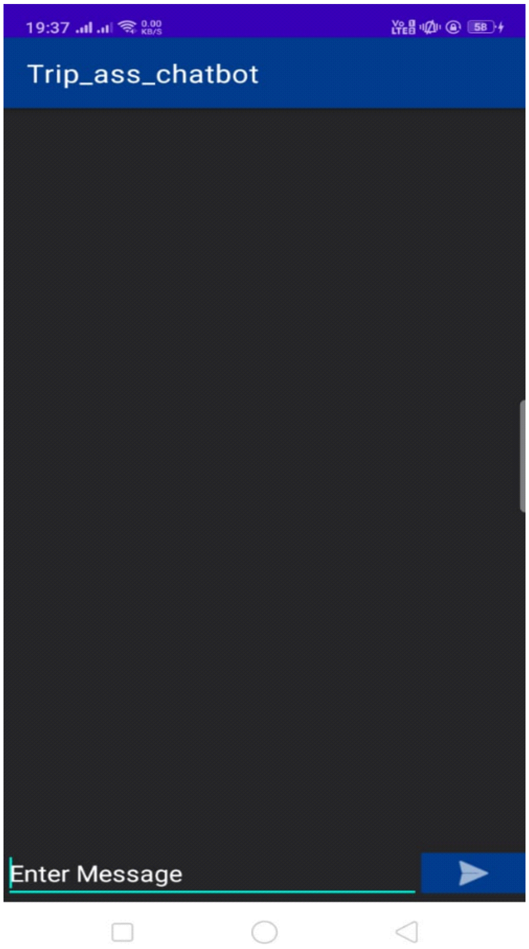
Procedural Programming which at times has been referred to as inline programming takes a more top-down approach to programming. Object-oriented Programming uses classes and objects, Procedural Programming takes on applications by solving problems from the top of the code down to the bottom. This happens when a program starts with a problem and then breaks that problem down into smaller sub-problems or sub-procedures. These sub-procedures are continually broken down in the process called functional decomposition until the sub procedure is simple enough to be solved.

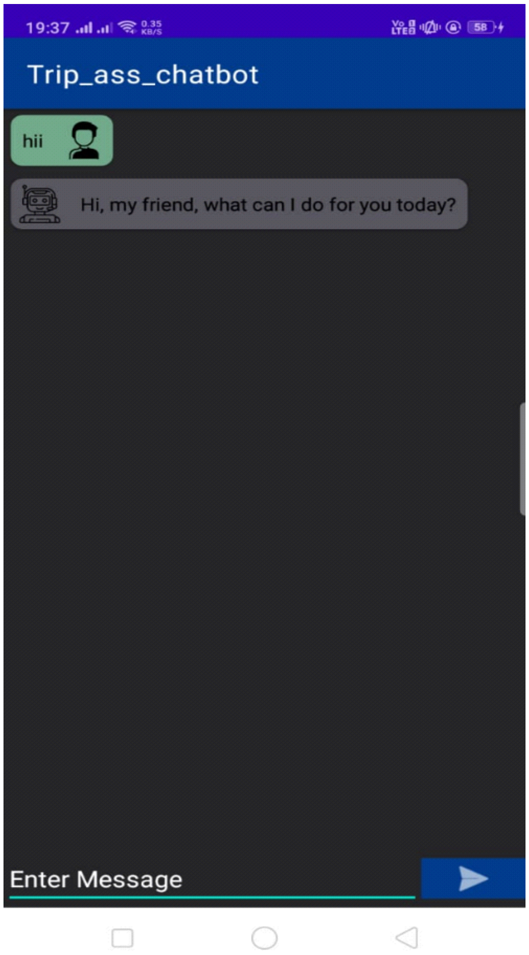
- Logic diagram





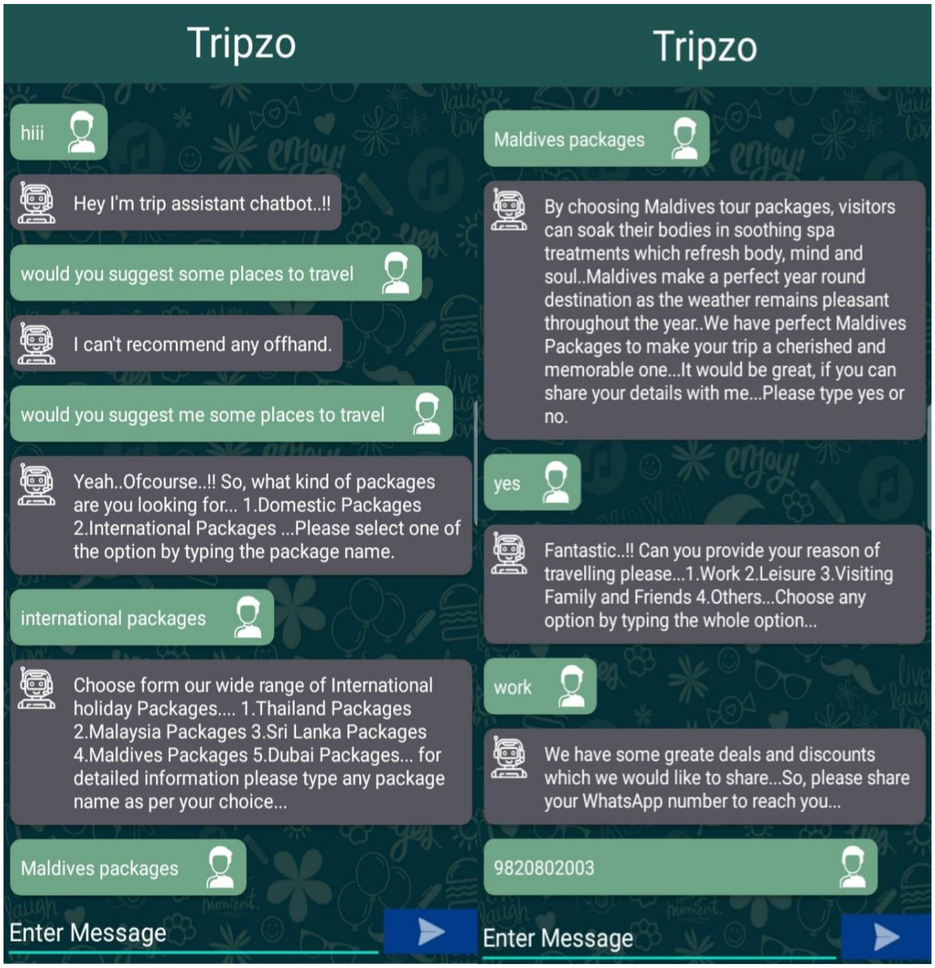
- User interface design


19:37
Trip_ass_chatbot
Enter Message



hii

Hi, my friend, what can I do for you today?



hiii

Hey I'm trip assistant chatbot..!!

would you suggest some places to travel

I can't recommend any offhand.

would you suggest me some places to travel

Yeah..Ofcourse..!! So, what kind of packages are you looking for... 1.Domestic Packages 2.International Packages ...Please select one of the option by typing the package name.

international packages

Choose form our wide range of International holiday Packages.... 1.Thailand Packages 2.Malaysia Packages 3.Sri Lanka Packages 4.Maldives Packages 5.Dubai Packages... for detailed information please type any package name as per your choice...

Maldives packages

Maldives packages

By choosing Maldives tour packages, visitors can soak their bodies in soothing spa treatments which refresh body, mind and soul..Maldives make a perfect year round destination as the weather remains pleasant throughout the year..We have perfect Maldives Packages to make your trip a cherished and memorable one...It would be great, if you can share your details with me...Please type yes or no.

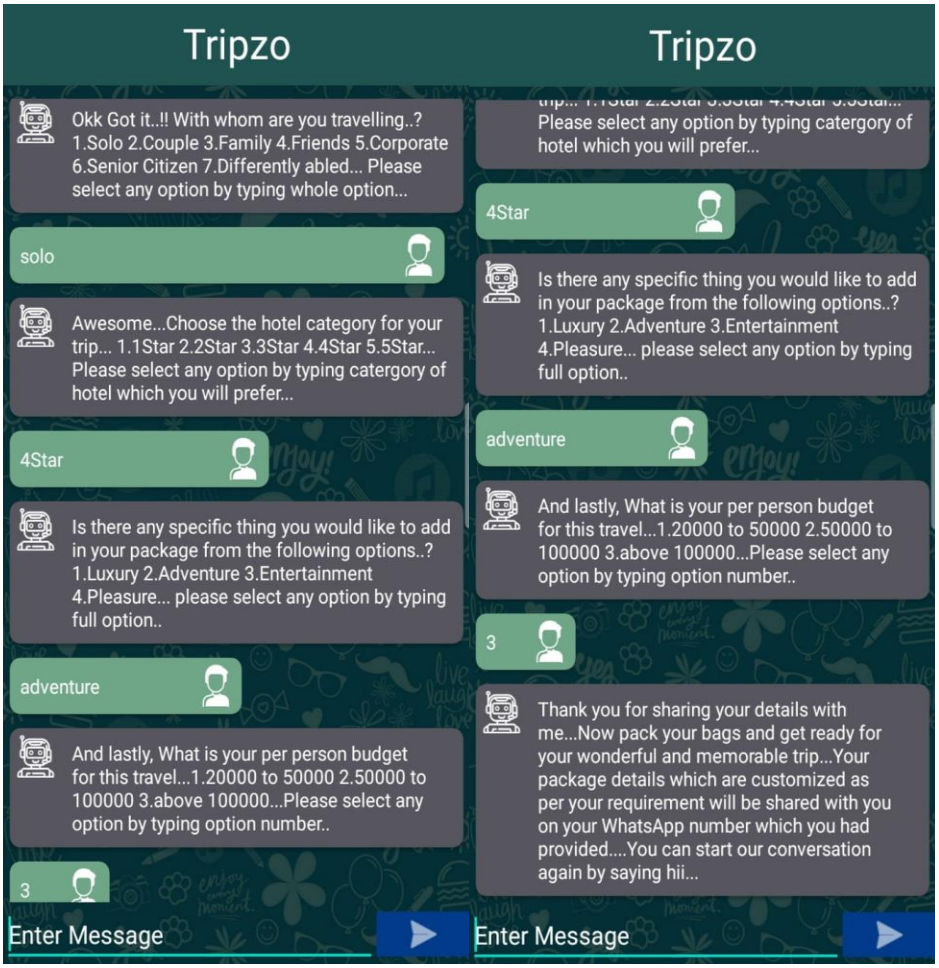
yes

Fantastic..!! Can you provide your reason of travelling please...1.Work 2.Leisure 3.Visiting Family and Friends 4.Others...Choose any option by typing the whole option...

work

We have some greate deals and discounts which we would like to share...So, please share your WhatsApp number to reach you...

9820802003



Okk Got it..!! With whom are you travelling..? 1.Solo 2.Couple 3.Family 4.Friends 5.Corporate 6.Senior Citizen 7.Differently abled... Please select any option by typing whole option...

solo

Awesome...Choose the hotel category for your trip... 1.1Star 2.2Star 3.3Star 4.4Star 5.5Star... Please select any option by typing catergory of hotel which you will prefer...

4Star

Is there any specific thing you would like to add in your package from the following options..? 1.Luxury 2.Adventure 3.Entertainment 4.Pleasure... please select any option by typing full option..

adventure

And lastly, What is your per person budget for this travel...1.20000 to 50000 2.50000 to 100000 3.above 100000...Please select any option by typing option number..

3

Please select any option by typing catergory of hotel which you will prefer...

4Star

Is there any specific thing you would like to add in your package from the following options..? 1.Luxury 2.Adventure 3.Entertainment 4.Pleasure... please select any option by typing full option..

adventure

And lastly, What is your per person budget for this travel...1.20000 to 50000 2.50000 to 100000 3.above 100000...Please select any option by typing option number..

3

Thank you for sharing your details with me...Now pack your bags and get ready for your wonderful and memorable trip...Your package details which are customized as per your requirement will be shared with you on your WhatsApp number which you had provided....You can start our conversation again by saying hii...

4.4 Security issues

Data is the most precious factor of today’s websites. Top business organizations spend billions of dollars every year to secure their computer networks and to keep their business data safe. We are dependent on computers today for controlling large money transfers between banks, insurance, markets, telecommunication, electrical power distribution, health and medical fields, nuclear power plants, space research and satellites. We cannot negotiate security in these critical areas. Security is freedom from, or resilience against, potential harm (or other unwanted coercive change) from external forces. Beneficiaries (technically referents) of security may be persons and social groups, objects and institutions, ecosystems, and any other entity or phenomenon vulnerable to unwanted change by its environment. The most important service in today’s world is security. As technology is increasing, ways to hack data are also increasing. For any website or application development, security is one of the crucial aspects. Security involves securing the data. In our system data is secured as only the person knowing username and password can enter the system. Passwords are stored in encrypted format. At time of registration, one security question is asked from users. If users forget their password at later stage, they can access their account by answering that security question. This way system ensures that no other can use someone else’s account. The security involved in this system is hierarchical security. Security is the degree of resistance to, or protection from attack. In order to elicit security requirements, ones should have the knowledge regarding security issues. The most common issues are CIA (Confidentiality, Integrity, and Availability)

- C: Confidentiality is prevention of unauthorized disclosure of information.
- I: Integrity is prevention of unauthorized modification of information.
- A: Availability is prevention of unauthorized withholding of information.

Security requirements ensure that the software is protected from unauthorized access to the system and its stored data. It considers different levels of authorization and authentication across different user’s roles.

- Sending abusive e-mails or posting offensive Web pages.
- Creation or transmission of any offensive or indecent images.
- Giving unauthorized access to other computing resources e.g., allowing an account to be used by someone not authorize to use it.
- Deliberately creating or spreading computer viruses or worms.
- Unauthorized running of applications that involve committing the website to sharing its user’s resources, e.g., network bandwidth, in an uncontrolled and unlimited way.

Cyber-attacks take advantage of software errors, such as not properly validating user input, inconsistencies in the design assumptions among system components, and unanticipated user and operator actions. Software errors can be introduced by disconnects and miscommunications during the planning, development, testing, and maintenance of the components. Although an application development team may be expert in the required business functionality, that team usually has limited or no applicable security expertise.

4.5 Test cases design

Software testing is an essential and important technique for assessing the quality of a particular software product/service. In software testing, test cases and scenarios play an inevitable and a pivotal role. A good strategic design and technique help to improve the quality of the software testing process. The process improves the quality of the product/service and ensures effectiveness. Software testing is the process of analysing a software item to know the differences between the existing and required conditions (bugs). Testing helps to evaluate the features of the software, to ensure it is free of bug. It is an activity that is carried out in co ordinance with the development cycle and before the deployment.

Divided mainly into three major categories, test case design techniques are crucial verifications steps that are created to design a software or application that is free from various kinds of defects and issues. The purpose of these techniques is to test the functionalities and features of the software with the assistance of some effective testcases.

The three categories of test case design techniques are:

- Specification Based or Black-Box Techniques.
- Structure Based or White-Box Techniques.
- Experience Based Techniques.

Black Box Testing:

In Black Box Testing, the tester tests an application without knowledge of the internal workings of the application being tested. Data are entered into the application and the outcome is compared with the expected results; what the program does with the input data or how the program arrives at the output data is not a concern for the tester performing black box testing. All that is tested is the behaviour of the functions being tested.

Therefore, black box testing is also known as functional testing which tests the functionality of a program. Note we can also have non-functional black box testing, such as performance testing which is a type of black box testing but instead of verifying the behaviour of the system, it tests how long it takes for a function to respond to user’s inputs and how long it takes to process the data and generate outputs.

White Box Testing:

White Box Testing (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system. This method is named so because the software program, in the eyes of the tester, is like a white/transparent box; inside which one clearly sees.

Experience Based Techniques:

These techniques are highly dependent on tester’s experience to understand the most important areas of the software. The outcomes of these techniques are based on the skills, knowledge, and expertise of the people involved. A test case software can help in writing better test cases and managing them. It also enables you to report bugs from any failed step. A tool provides robust reports generated through built-in filters also gives you actionable insights.

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Implementation Approaches

Define the chatbot's scope and functionality: Before starting the implementation process, define the scope of the chatbot and the specific tasks it should be able to perform. This will help guide the development process and ensure that the chatbot meets the intended requirements.

Use Brainshop AI to create the chatbot's conversational flow: Brainshop AI is a powerful tool that allows developers to create intelligent chatbots. Using Brainshop AI, create the conversational flow for the trip assistant chatbot, including defining intents, entities, and responses.

Build the Android Studio interface: Create the Android Studio interface for the chatbot, which will allow users to interact with the chatbot. This may include designing the chatbot's user interface, implementing text recognition, and integrating with other third-party services as needed.

Implement the chatbot's functionality: Using the conversational flow defined in Brainshop AI, implement the chatbot's functionality within Android Studio. This may include creating logic to handle user input, integrating with external APIs to retrieve travel information, and providing personalized recommendations based on user preferences.

Test and refine the chatbot: Once the chatbot has been implemented, test it thoroughly to ensure that it meets the intended requirements and performs as expected. Use feedback from users to refine the chatbot's functionality and improve the user experience.

Deploy the chatbot: Once the chatbot has been tested and refined, deploy it to a live environment where users can interact with it. Monitor the chatbot's performance and make adjustments as needed to ensure that it continues to meet user needs.

5.2 Coding Details and Code Efficiency

- **Android Studio**: This will be the primary development environment for building the Android app.
- **Python Integration**: You can use Python for various functionalities such as natural language processing (NLP), data processing, or accessing external APIs. This can be achieved using libraries like Chaquopy or SL4A (Scripting Layer for Android) to integrate Python code within the Android app.
- **Java and Android SDK**: Java will be used for the native Android app development, along with the Android Software Development Kit (SDK) for UI design, activities, intents, and more.
- **NLP and AI**: Implement natural language processing to understand user queries and intents. Libraries like NLTK, spaCy, or TensorFlow can be used for this purpose.
- **Trip Planning Algorithms and Data Processing**:
  - Use algorithms to help users plan trips, such as recommending destinations based on preferences, budgets, and timeframes.
  - Data processing for fetching and displaying information about hotels, flights, tourist spots, etc.
- **API Integrations**:
  - Integration with travel APIs (Google Maps, TripAdvisor, Expedia, etc.) to fetch real-time data like hotel availability, flight schedules, weather forecasts, etc.
- **UI/UX Design**:
  - Designing a user-friendly interface to interact with the chatbot, displaying trip suggestions, itinerary, maps, and other relevant information.

Main\AndroidManifest.xml:-

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 package="com.example.trip_ass_chatbot">

 <uses-permission android:name="android.permission.INTERNET" />
 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

 <application
 android:allowBackup="true"
 android:dataExtractionRules="@xml/data_extraction_rules"
 android:fullBackupContent="@xml/backup_rules"
 android:icon="@mipmap/ic_launcher"
 android:label="@string/app_name"
 android:roundIcon="@mipmap/ic_launcher_round"
 android:supportsRtl="true"
 android:theme="@style/Theme.Trip_ass_chatbot"
 tools:targetApi="31">
 <activity
 android:name=".MainActivity"
 android:exported="true">
 <intent-filter>
 <action android:name="android.intent.action.MAIN" />

 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
 </activity>
 </application>

</manifest>
```

MessageRVAdapter.java

```
package com.example.trip_ass_chatbot;

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

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

import com.example.trip_ass_chatbot.MessageModal;
import com.example.trip_ass_chatbot.R;

import java.util.ArrayList;

public class MessageRVAdapter extends RecyclerView.Adapter {

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

 // constructor class.
 public MessageRVAdapter(ArrayList<MessageModal> messageModalArrayList, Context context) {
 this.messageModalArrayList = messageModalArrayList;
 this.context = context;
 }

 @NonNull
```

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

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

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

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

public static class UserViewHolder extends RecyclerView.ViewHolder {
 // creating a variable
 // for our text view.
 TextView userTV;

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

 public static class BotViewHolder extends RecyclerView.ViewHolder {

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

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

.

package android.accessibilityservice;

import static android.accessibilityservice.MagnificationConfig.MAGNIFICATION_MODE_FULLSCREEN;
import static android.view.WindowManager.LayoutParams.TYPE_ACCESSIBILITY_OVERLAY;
import android.accessibilityservice.GestureDescription.MotionEventGenerator;
import android.annotation.CallbackExecutor;
import android.annotation.CheckResult;
import android.annotation.ColorInt;
import android.annotation.IntDef;
import android.annotation.NonNull;
import android.annotation.Nullable;
import android.annotation.RequiresPermission;
import android.annotation.TestApi;
import android.app.Service;
import android.compat.annotation.UnsupportedAppUsage;
import android.content.Context;
import android.content.ContextWrapper;
import android.content.Intent;
import android.content.pm.ParceledListSlice;
import android.graphics.Bitmap;
import android.graphics.ColorSpace;
import android.graphics.ParcelableColorSpace;
import android.graphics.Region;
import android.hardware.HardwareBuffer;
import android.hardware.display.DisplayManager;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.HandlerExecutor;
import android.os.IBinder;
import android.os.Looper;
import android.os.RemoteCallback;
import android.os.RemoteException;
import android.os.SystemClock;
import android.util.ArrayMap;
import android.util.Log;
import android.util.Slog;
import android.util.SparseArray;
import android.view.Display;
import android.view.InputDevice;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.SurfaceControl;
import android.view.WindowManager;
import android.view.WindowManagerImpl;
import android.view.accessibility.AccessibilityCache;
import android.view.accessibility.AccessibilityEvent;
import android.view.accessibility.AccessibilityInteractionClient;
import android.view.accessibility.AccessibilityNodeInfo;
import android.view.accessibility.AccessibilityNodeInfo.AccessibilityAction;
import android.view.accessibility.AccessibilityWindowInfo;
import android.view.inputmethod.EditorInfo;

import com.android.internal.inputmethod.CancellationGroup;
import com.android.internal.inputmethod.IAccessibilityInputMethodSession;
import com.android.internal.inputmethod.IAccessibilityInputMethodSessionCallback;
import com.android.internal.inputmethod.IRemoteAccessibilityInputConnection;
import com.android.internal.inputmethod.RemoteAccessibilityInputConnection;
import com.android.internal.util.Preconditions;

import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.util.Collections;
import java.util.List;
import java.util.concurrent.Executor;
import java.util.function.Consumer;

**
 * Accessibility services should only be used to assist users with disabilities in using
 * Android devices and apps. They run in the background and receive callbacks by the system
 * when {@link AccessibilityEvent}s are fired. Such events denote some state transition
 * in the user interface, for example, the focus has changed, a button has been clicked,
 * etc. Such a service can optionally request the capability for querying the content
 * of the active window. Development of an accessibility service requires extending this
 * class and implementing its abstract methods.
 *
 * <div class="special reference">
 * <h3>Developer Guides</h3>
 * <p>For more information about creating AccessibilityServices, read the
 * <a href="{@docRoot}guide/topics/ui/accessibility/index.html">Accessibility</a>
 * developer guide.</p>
 * </div>
 *
 * <h3>Lifecycle</h3>
 * <p>
 * The lifecycle of an accessibility service is managed exclusively by the system and
 * follows the established service life cycle. Starting an accessibility service is triggered
 * exclusively by the user explicitly turning the service on in device settings. After the system
 * binds to a service, it calls {@link AccessibilityService#onServiceConnected()}. This method can
 * be overridden by clients that want to perform post binding setup.
 * </p>
 * <p>
 * An accessibility service stops either when the user turns it off in device settings or when
 * it calls {@link AccessibilityService#disableSelf()}.
 * </p>
 * <h3>Declaration</h3>
 * <p>
 * An accessibility is declared as any other service in an AndroidManifest.xml, but it
 * must do two things:
 * <ul>
 * <li>
 * Specify that it handles the "android.accessibilityservice.AccessibilityService"
 * {@link android.content.Intent}.
 * </li>
 * <li>
 * Request the {@link android.Manifest.permission#BIND_ACCESSIBILITY_SERVICE} permission to
 * ensure that only the system can bind to it.
 * </li>
 * </ul>
 * If either of these items is missing, the system will ignore the accessibility service.
 * Following is an example declaration:
 * </p>
 * <pre> &lt;service android:name=".MyAccessibilityService"
 * android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE"&gt;
 * &lt;intent-filter&gt;
 * &lt;action android:name="android.accessibilityservice.AccessibilityService" /&gt;
 * &lt;/intent-filter&gt;
 * . . .
 * &lt;/service&gt;</pre>
 * <h3>Configuration</h3>
 * <p>
 * An accessibility service can be configured to receive specific types of accessibility events,
 * listen only to specific packages, get events from each type only once in a given time frame,
 * retrieve window content, specify a settings activity, etc.
 * </p>
 * <p>
 * There are two approaches for configuring an accessibility service:
 * </p>
 * <ul>
 * <li>
 * Providing a {@link #SERVICE_META_DATA meta-data} entry in the manifest when declaring
 * the service. A service declaration with a meta-data tag is presented below:
 * <pre> &lt;service android:name=".MyAccessibilityService"&gt;
 * &lt;intent-filter&gt;
 * &lt;action android:name="android.accessibilityservice.AccessibilityService" /&gt;
 * &lt;/intent-filter&gt;
 * &lt;meta-data android:name="android.accessibilityservice" android:resource="@xml/accessibilityservice" /&gt;
 * &lt;/service&gt;</pre>
 * <p class="note">
 * <strong>Note:</strong> This approach enables setting all properties.
 * </p>
 * <p>
 * For more details refer to {@link #SERVICE_META_DATA} and
 * <code>&lt;{@link android.R.styleable#AccessibilityService accessibility-service}&gt;</code>.
 * </p>
 * </li>
 * <li>
 * Calling {@link AccessibilityService#setServiceInfo(AccessibilityServiceInfo)}. Note
 * that this method can be called any time to dynamically change the service configuration.
 * <p class="note">
 * <strong>Note:</strong> This approach enables setting only dynamically configurable properties:
 * {@link AccessibilityServiceInfo#eventTypes},
 * {@link AccessibilityServiceInfo#feedbackType},
 * {@link AccessibilityServiceInfo#flags},
 * {@link AccessibilityServiceInfo#notificationTimeout},
 * {@link AccessibilityServiceInfo#packageNames}
 * </p>
 * <p>
 * For more details refer to {@link AccessibilityServiceInfo}.
 * </p>
 * </li>
 * </ul>
 * <h3>Retrieving window content</h3>
 * <p>
 * A service can specify in its declaration that it can retrieve window
 * content which is represented as a tree of {@link AccessibilityWindowInfo} and
 * {@link AccessibilityNodeInfo} objects. Note that
 * declaring this capability requires that the service declares its configuration via
 * an XML resource referenced by {@link #SERVICE_META_DATA}.
 * </p>
 * <p>
 * Window content may be retrieved with
 * {@link AccessibilityEvent#getSource() AccessibilityEvent.getSource()},
 * {@link AccessibilityService#findFocus(int)},
 * {@link AccessibilityService#getWindows()}, or
 * {@link AccessibilityService#getRootInActiveWindow()}.
 * </p>
 * <p class="note">
 * <strong>Note</strong> An accessibility service may have requested to be notified for
 * a subset of the event types, and thus be unaware when the node hierarchy has changed. It is also
 * possible for a node to contain outdated information because the window content may change at any
 * time.
 * </p>
 * <h3>Drawing Accessibility Overlays</h3>
 * <p>Accessibility services can draw overlays on top of existing screen contents.
 * Accessibility overlays can be used to visually highlight items on the screen
 * e.g. indicate the current item with accessibility focus.
 * Overlays can also offer the user a way to interact with the service directly and quickly
 * customize the service's behavior.</p>
 * <p>Accessibility overlays can be attached to a particular window or to the display itself.
 * Attaching an overlay to a window allows the overly to move, grow and shrink as the window does.
 * The overlay will maintain the same relative position within the window bounds as the window
 * moves. The overlay will also maintain the same relative position within the window bounds if
 * the window is resized.
 * To attach an overlay to a window, use {@link #attachAccessibilityOverlayToWindow}.
 * Attaching an overlay to the display means that the overlay is independent of the active
 * windows on that display.
 * To attach an overlay to a display, use {@link #attachAccessibilityOverlayToDisplay}. </p>
 * <p> When positioning an overlay that is attached to a window, the service must use window
 * coordinates. In order to position an overlay on top of an existing UI element it is necessary
 * to know the bounds of that element in window coordinates. To find the bounds in window
 * coordinates of an element, find the corresponding {@link AccessibilityNodeInfo} as discussed
 * above and call {@link AccessibilityNodeInfo#getBoundsInWindow}. </p>
 * <h3>Notification strategy</h3>
 * <p>
 * All accessibility services are notified of all events they have requested, regardless of their
 * feedback type.
 * </p>
 * <p class="note">
 * <strong>Note:</strong> The event notification timeout is useful to avoid propagating
 * events to the client too frequently since this is accomplished via an expensive
 * interprocess call. One can think of the timeout as a criteria to determine when
 * event generation has settled down.</p>
 * <h3>Event types</h3>
 * <ul>
 * <li>{@link AccessibilityEvent#TYPE_VIEW_CLICKED}</li>
 * <li>{@link AccessibilityEvent#TYPE_VIEW_LONG_CLICKED}</li>
 * <li>{@link AccessibilityEvent#TYPE_VIEW_FOCUSED}</li>
 * <li>{@link AccessibilityEvent#TYPE_VIEW_SELECTED}</li>
 * <li>{@link AccessibilityEvent#TYPE_VIEW_TEXT_CHANGED}</li>
 * <li>{@link AccessibilityEvent#TYPE_WINDOW_STATE_CHANGED}</li>
 * <li>{@link AccessibilityEvent#TYPE_NOTIFICATION_STATE_CHANGED}</li>
 * <li>{@link AccessibilityEvent#TYPE_TOUCH_EXPLORATION_GESTURE_START}</li>
 * <li>{@link AccessibilityEvent#TYPE_TOUCH_EXPLORATION_GESTURE_END}</li>
 * <li>{@link AccessibilityEvent#TYPE_VIEW_HOVER_ENTER}</li>
 * <li>{@link AccessibilityEvent#TYPE_VIEW_HOVER_EXIT}</li>
 * <li>{@link AccessibilityEvent#TYPE_VIEW_SCROLLED}</li>
 * <li>{@link AccessibilityEvent#TYPE_VIEW_TEXT_SELECTION_CHANGED}</li>
 * <li>{@link AccessibilityEvent#TYPE_WINDOW_CONTENT_CHANGED}</li>
 * <li>{@link AccessibilityEvent#TYPE_ANNOUNCEMENT}</li>
 * <li>{@link AccessibilityEvent#TYPE_GESTURE_DETECTION_START}</li>
 * <li>{@link AccessibilityEvent#TYPE_GESTURE_DETECTION_END}</li>
 * <li>{@link AccessibilityEvent#TYPE_TOUCH_INTERACTION_START}</li>
 * <li>{@link AccessibilityEvent#TYPE_TOUCH_INTERACTION_END}</li>
 * <li>{@link AccessibilityEvent#TYPE_VIEW_ACCESSIBILITY_FOCUSED}</li>
 * <li>{@link AccessibilityEvent#TYPE_WINDOWS_CHANGED}</li>
 * <li>{@link AccessibilityEvent#TYPE_VIEW_ACCESSIBILITY_FOCUS_CLEARED}</li>
 * </ul>
 * <h3>Feedback types</h3>
 * <ul>
 * <li>{@link AccessibilityServiceInfo#FEEDBACK_AUDIBLE}</li>
 * <li>{@link AccessibilityServiceInfo#FEEDBACK_HAPTIC}</li>
 * <li>{@link AccessibilityServiceInfo#FEEDBACK_SPOKEN}</li>
 * <li>{@link AccessibilityServiceInfo#FEEDBACK_VISUAL}</li>
```

```
 * <li>{@link AccessibilityServiceInfo#FEEDBACK_GENERIC}</li>
 * <li>{@link AccessibilityServiceInfo#FEEDBACK_BRAILLE}</li>
 * </ul>
 * @see AccessibilityEvent
 * @see AccessibilityServiceInfo
 * @see android.view.accessibility.AccessibilityManager
 */
public abstract class AccessibilityService extends Service {

 /**
 * The user has performed a touch-exploration gesture on the touch screen without ever
 * triggering gesture detection. This gesture is only dispatched when {@link
 * AccessibilityServiceInfo#FLAG_SEND_MOTION_EVENTS} is set.
 *
 * @hide
 */
 public static final int GESTURE_TOUCH_EXPLORATION = -2;

 /**
 * The user has performed a passthrough gesture on the touch screen without ever triggering
 * gesture detection. This gesture is only dispatched when {@link
 * AccessibilityServiceInfo#FLAG_SEND_MOTION_EVENTS} is set.
 * @hide
 */
 public static final int GESTURE_PASSTHROUGH = -1;

 /**
 * The user has performed an unrecognized gesture on the touch screen. This gesture is only
 * dispatched when {@link AccessibilityServiceInfo#FLAG_SEND_MOTION_EVENTS} is set.
 */
 public static final int GESTURE_UNKNOWN = 0;

 /**
 * The user has performed a swipe up gesture on the touch screen.
 */
 public static final int GESTURE_SWIPE_UP = 1;

 /**
 * The user has performed a swipe down gesture on the touch screen.
 */
 public static final int GESTURE_SWIPE_DOWN = 2;

 /**
 * The user has performed a swipe left gesture on the touch screen.
 */
 public static final int GESTURE_SWIPE_LEFT = 3;

 /**
 * The user has performed a swipe right gesture on the touch screen.
 */
 public static final int GESTURE_SWIPE_RIGHT = 4;

 /**
```

```
* The user has performed a swipe left and right gesture on the touch screen.
*/
public static final int GESTURE_SWIPE_LEFT_AND_RIGHT = 5;

/**
* The user has performed a swipe right and left gesture on the touch screen.
*/
public static final int GESTURE_SWIPE_RIGHT_AND_LEFT = 6;

/**
* The user has performed a swipe up and down gesture on the touch screen.
*/
public static final int GESTURE_SWIPE_UP_AND_DOWN = 7;

/**
* The user has performed a swipe down and up gesture on the touch screen.
*/
public static final int GESTURE_SWIPE_DOWN_AND_UP = 8;

/**
* The user has performed a left and up gesture on the touch screen.
*/
public static final int GESTURE_SWIPE_LEFT_AND_UP = 9;

/**
* The user has performed a left and down gesture on the touch screen.
*/
public static final int GESTURE_SWIPE_LEFT_AND_DOWN = 10;

/**
* The user has performed a right and up gesture on the touch screen.
*/
public static final int GESTURE_SWIPE_RIGHT_AND_UP = 11;

/**
* The user has performed a right and down gesture on the touch screen.
*/
public static final int GESTURE_SWIPE_RIGHT_AND_DOWN = 12;

/**
* The user has performed an up and left gesture on the touch screen.
*/
public static final int GESTURE_SWIPE_UP_AND_LEFT = 13;

/**
* The user has performed an up and right gesture on the touch screen.
*/
public static final int GESTURE_SWIPE_UP_AND_RIGHT = 14;

/**
* The user has performed an down and left gesture on the touch screen.
*/
```

```
public static final int GESTURE_SWIPE_DOWN_AND_LEFT = 15;

/**
* The user has performed an down and right gesture on the touch screen.
*/
public static final int GESTURE_SWIPE_DOWN_AND_RIGHT = 16;

/**
* The user has performed a double tap gesture on the touch screen.
*/
public static final int GESTURE_DOUBLE_TAP = 17;

/**
* The user has performed a double tap and hold gesture on the touch screen.
*/
public static final int GESTURE_DOUBLE_TAP_AND_HOLD = 18;

/**
* The user has performed a two-finger single tap gesture on the touch screen.
*/
public static final int GESTURE_2_FINGER_SINGLE_TAP = 19;

/**
* The user has performed a two-finger double tap gesture on the touch screen.
*/
public static final int GESTURE_2_FINGER_DOUBLE_TAP = 20;

/**
* The user has performed a two-finger triple tap gesture on the touch screen.
*/
public static final int GESTURE_2_FINGER_TRIPLE_TAP = 21;

/**
* The user has performed a three-finger single tap gesture on the touch screen.
*/
public static final int GESTURE_3_FINGER_SINGLE_TAP = 22;

/**
* The user has performed a three-finger double tap gesture on the touch screen.
*/
public static final int GESTURE_3_FINGER_DOUBLE_TAP = 23;

/**
* The user has performed a three-finger triple tap gesture on the touch screen.
*/
public static final int GESTURE_3_FINGER_TRIPLE_TAP = 24;

/**
* The user has performed a two-finger swipe up gesture on the touch screen.
*/
public static final int GESTURE_2_FINGER_SWIPE_UP = 25;
```

```
/**
* The user has performed a two-finger swipe down gesture on the touch screen.
*/
public static final int GESTURE_2_FINGER_SWIPE_DOWN = 26;

/**
* The user has performed a two-finger swipe left gesture on the touch screen.
*/
public static final int GESTURE_2_FINGER_SWIPE_LEFT = 27;

/**
* The user has performed a two-finger swipe right gesture on the touch screen.
*/
public static final int GESTURE_2_FINGER_SWIPE_RIGHT = 28;

/**
* The user has performed a three-finger swipe up gesture on the touch screen.
*/
public static final int GESTURE_3_FINGER_SWIPE_UP = 29;

/**
* The user has performed a three-finger swipe down gesture on the touch screen.
*/
public static final int GESTURE_3_FINGER_SWIPE_DOWN = 30;

/**
* The user has performed a three-finger swipe left gesture on the touch screen.
*/
public static final int GESTURE_3_FINGER_SWIPE_LEFT = 31;

/**
* The user has performed a three-finger swipe right gesture on the touch screen.
*/
public static final int GESTURE_3_FINGER_SWIPE_RIGHT = 32;

/** The user has performed a four-finger swipe up gesture on the touch screen. */
public static final int GESTURE_4_FINGER_SWIPE_UP = 33;

/** The user has performed a four-finger swipe down gesture on the touch screen. */
public static final int GESTURE_4_FINGER_SWIPE_DOWN = 34;

/** The user has performed a four-finger swipe left gesture on the touch screen. */
public static final int GESTURE_4_FINGER_SWIPE_LEFT = 35;

/** The user has performed a four-finger swipe right gesture on the touch screen. */
public static final int GESTURE_4_FINGER_SWIPE_RIGHT = 36;

/** The user has performed a four-finger single tap gesture on the touch screen. */
public static final int GESTURE_4_FINGER_SINGLE_TAP = 37;

/** The user has performed a four-finger double tap gesture on the touch screen. */
public static final int GESTURE_4_FINGER_DOUBLE_TAP = 38;
```

```
/** The user has performed a four-finger triple tap gesture on the touch screen. */
public static final int GESTURE_4_FINGER_TRIPLE_TAP = 39;

/** The user has performed a two-finger double tap and hold gesture on the touch screen. */
public static final int GESTURE_2_FINGER_DOUBLE_TAP_AND_HOLD = 40;

/** The user has performed a three-finger double tap and hold gesture on the touch screen. */
public static final int GESTURE_3_FINGER_DOUBLE_TAP_AND_HOLD = 41;

/** The user has performed a two-finger triple-tap and hold gesture on the touch screen. */
public static final int GESTURE_2_FINGER_TRIPLE_TAP_AND_HOLD = 43;

/** The user has performed a three-finger single-tap and hold gesture on the touch screen. */
public static final int GESTURE_3_FINGER_SINGLE_TAP_AND_HOLD = 44;

/** The user has performed a three-finger triple-tap and hold gesture on the touch screen. */
public static final int GESTURE_3_FINGER_TRIPLE_TAP_AND_HOLD = 45;

/** The user has performed a two-finger double tap and hold gesture on the touch screen. */
public static final int GESTURE_4_FINGER_DOUBLE_TAP_AND_HOLD = 42;

/**
* The {@link Intent} that must be declared as handled by the service.
*/
public static final String SERVICE_INTERFACE =
"android.accessibilityservice.AccessibilityService";

/**
* Name under which an AccessibilityService component publishes information
* about itself. This meta-data must reference an XML resource containing an
* <code>&lt;{@link android.R.styleable#AccessibilityService accessibility-service}&gt;</code>
* tag. This is a sample XML file configuring an accessibility service:
* <pre> &lt;accessibility-service
* android:accessibilityEventTypes="typeViewClicked|typeViewFocused"
* android:packageNames="foo.bar, foo.baz"
* android:accessibilityFeedbackType="feedbackSpoken"
* android:notificationTimeout="100"
* android:accessibilityFlags="flagDefault"
* android:settingsActivity="foo.bar.TestBackActivity"
* android:canRetrieveWindowContent="true"
* android:canRequestTouchExplorationMode="true"
* . . .
* /&gt;</pre>
*/
public static final String SERVICE_META_DATA = "android.accessibilityservice";

/**
* Action to go back.
*/
public static final int GLOBAL_ACTION_BACK = 1;
```

```
/**
* Action to go home.
*/
public static final int GLOBAL_ACTION_HOME = 2;

/**
* Action to toggle showing the overview of recent apps. Will fail on platforms that don't
* show recent apps.
*/
public static final int GLOBAL_ACTION_RECENTS = 3;

/**
* Action to open the notifications.
*/
public static final int GLOBAL_ACTION_NOTIFICATIONS = 4;

/**
* Action to open the quick settings.
*/
public static final int GLOBAL_ACTION_QUICK_SETTINGS = 5;

/**
* Action to open the power long-press dialog.
*/
public static final int GLOBAL_ACTION_POWER_DIALOG = 6;

/**
* Action to toggle docking the current app's window.
* <p>
* <strong>Note:</strong> It is effective only if it appears in {@link #getSystemActions()}.
*/
public static final int GLOBAL_ACTION_TOGGLE_SPLIT_SCREEN = 7;

/**
* Action to lock the screen
*/
public static final int GLOBAL_ACTION_LOCK_SCREEN = 8;

/**
* Action to take a screenshot
*/
public static final int GLOBAL_ACTION_TAKE_SCREENSHOT = 9;

/**
* Action to send the KEYCODE_HEADSETHOOK KeyEvent, which is used to answer/hang up calls and
* play/stop media
*/
public static final int GLOBAL_ACTION_KEYCODE_HEADSETHOOK = 10;

/**
* Action to trigger the Accessibility Button
*/
```

```
public static final int GLOBAL_ACTION_ACCESSIBILITY_BUTTON = 11;

/**
* Action to bring up the Accessibility Button's chooser menu
*/
public static final int GLOBAL_ACTION_ACCESSIBILITY_BUTTON_CHOOSER = 12;

/**
* Action to trigger the Accessibility Shortcut. This shortcut has a hardware trigger and can
* be activated by holding down the two volume keys.
*/
public static final int GLOBAL_ACTION_ACCESSIBILITY_SHORTCUT = 13;

/**
* Action to show Launcher's all apps.
*/
public static final int GLOBAL_ACTION_ACCESSIBILITY_ALL_APPS = 14;

/**
* Action to dismiss the notification shade
*/
public static final int GLOBAL_ACTION_DISMISS_NOTIFICATION_SHADE = 15;

/**
* Action to trigger dpad up keyevent.
*/
public static final int GLOBAL_ACTION_DPAD_UP = 16;

/**
* Action to trigger dpad down keyevent.
*/
public static final int GLOBAL_ACTION_DPAD_DOWN = 17;

/**
* Action to trigger dpad left keyevent.
*/
public static final int GLOBAL_ACTION_DPAD_LEFT = 18;

/**
* Action to trigger dpad right keyevent.
*/
public static final int GLOBAL_ACTION_DPAD_RIGHT = 19;

/**
* Action to trigger dpad center keyevent.
*/
public static final int GLOBAL_ACTION_DPAD_CENTER = 20;

private static final String LOG_TAG = "AccessibilityService";

/**
* Interface used by IAccessibilityServiceClientWrapper to call the service from its main
* thread.
* @hide
*/
public interface Callbacks {
void onAccessibilityEvent(AccessibilityEvent event);
void onInterrupt();
void onServiceConnected();
void init(int connectionId, IBinder windowToken);
/** The detected gesture information for different displays */
boolean onGesture(AccessibilityGestureEvent gestureInfo);
boolean onKeyEvent(KeyEvent event);
/** Magnification changed callbacks for different displays */
void onMagnificationChanged(int displayId, @NonNull Region region,
MagnificationConfig config);
/** Callbacks for receiving motion events. */
void onMotionEvent(MotionEvent event);
/** Callback for tuch state changes. */
void onTouchStateChanged(int displayId, int state);
void onSoftKeyboardShowModeChanged(int showMode);
void onPerformGestureResult(int sequence, boolean completedSuccessfully);
void onFingerprintCapturingGesturesChanged(boolean active);
void onFingerprintGesture(int gesture);
/** Accessbility button clicked callbacks for different displays */
void onAccessibilityButtonClicked(int displayId);
void onAccessibilityButtonAvailabilityChanged(boolean available);
/** This is called when the system action list is changed. */
void onSystemActionsChanged();
/** This is called when an app requests ime sessions or when the service is enabled. */
void createImeSession(IAccessibilityInputMethodSessionCallback callback);
/** This is called when an app starts input or when the service is enabled. */
void startInput(@Nullable RemoteAccessibilityInputConnection inputConnection,
@NonNull EditorInfo editorInfo, boolean restarting);
}

/**
* Annotations for Soft Keyboard show modes so tools can catch invalid show modes.
* @hide
*/
@Retention(RetentionPolicy.SOURCE)
@IntDef(prefix = { "SHOW_MODE_" }, value = {
SHOW_MODE_AUTO,
SHOW_MODE_HIDDEN,
SHOW_MODE_IGNORE_HARD_KEYBOARD
})
public @interface SoftKeyboardShowMode {}

/**
* Allow the system to control when the soft keyboard is shown.
* @see SoftKeyboardController
*/
public static final int SHOW_MODE_AUTO = 0;

/**
* Never show the soft keyboard.
* @see SoftKeyboardController
*/
public static final int SHOW_MODE_HIDDEN = 1;

/**
* Allow the soft keyboard to be shown, even if a hard keyboard is connected
* @see SoftKeyboardController
*/
public static final int SHOW_MODE_IGNORE_HARD_KEYBOARD = 2;

/**
* Mask used to cover the show modes supported in public API
* @hide
*/
public static final int SHOW_MODE_MASK = 0x03;

/**
* Bit used to hold the old value of the hard IME setting to restore when a service is shut
* down.
* @hide
*/
public static final int SHOW_MODE_HARD_KEYBOARD_ORIGINAL_VALUE = 0x20000000;

/**
* Bit for show mode setting to indicate that the user has overridden the hard keyboard
* behavior.
* @hide
*/
public static final int SHOW_MODE_HARD_KEYBOARD_OVERRIDDEN = 0x40000000;

/**
* Annotations for error codes of taking screenshot.
* @hide
*/
@Retention(RetentionPolicy.SOURCE)
@IntDef(prefix = { "TAKE_SCREENSHOT_" }, value = {
ERROR_TAKE_SCREENSHOT_INTERNAL_ERROR,
ERROR_TAKE_SCREENSHOT_NO_ACCESSIBILITY_ACCESS,
ERROR_TAKE_SCREENSHOT_INTERVAL_TIME_SHORT,
ERROR_TAKE_SCREENSHOT_INVALID_DISPLAY,
ERROR_TAKE_SCREENSHOT_INVALID_WINDOW
})
public @interface ScreenshotErrorCode {}

/**
* The status of taking screenshot is success.
* @hide
*/
public static final int TAKE_SCREENSHOT_SUCCESS = 0;

/**
* The status of taking screenshot is failure and the reason is internal error.
*/
```

```
public static final int ERROR_TAKE_SCREENSHOT_INTERNAL_ERROR = 1;

/**
* The status of taking screenshot is failure and the reason is no accessibility access.
*/
public static final int ERROR_TAKE_SCREENSHOT_NO_ACCESSIBILITY_ACCESS = 2;

/**
* The status of taking screenshot is failure and the reason is that too little time has
* elapsed since the last screenshot.
*/
public static final int ERROR_TAKE_SCREENSHOT_INTERVAL_TIME_SHORT = 3;

/**
* The status of taking screenshot is failure and the reason is invalid display Id.
*/
public static final int ERROR_TAKE_SCREENSHOT_INVALID_DISPLAY = 4;

/**
* The status of taking screenshot is failure and the reason is invalid accessibility window Id.
*/
public static final int ERROR_TAKE_SCREENSHOT_INVALID_WINDOW = 5;

/**
* The status of taking screenshot is failure and the reason is the window contains secure
* content.
* @see WindowManager.LayoutParams#FLAG_SECURE
*/
public static final int ERROR_TAKE_SCREENSHOT_SECURE_WINDOW = 6;

/**
* The interval time of calling
* {@link AccessibilityService#takeScreenshot(int, Executor, Consumer)} API.
* @hide
*/
@TestApi
public static final int ACCESSIBILITY_TAKE_SCREENSHOT_REQUEST_INTERVAL_TIMES_MS = 333;

/** @hide */
public static final String KEY_ACCESSIBILITY_SCREENSHOT_STATUS =
"screenshot_status";

/** @hide */
public static final String KEY_ACCESSIBILITY_SCREENSHOT_HARDWAREBUFFER =
"screenshot_hardwareBuffer";

/** @hide */
public static final String KEY_ACCESSIBILITY_SCREENSHOT_COLORSPACE =
"screenshot_colorSpace";

/** @hide */
public static final String KEY_ACCESSIBILITY_SCREENSHOT_TIMESTAMP =
"screenshot_timestamp";

private int mConnectionId = AccessibilityInteractionClient.NO_ID;

@UnsupportedAppUsage
private AccessibilityServiceInfo mInfo;

@UnsupportedAppUsage(maxTargetSdk = Build.VERSION_CODES.P, trackingBug = 115609023)
private IBinder mWindowToken;

private WindowManager mWindowManager;

/** List of magnification controllers, mapping from displayId -> MagnificationController. */
private final SparseArray<MagnificationController> mMagnificationControllers =
new SparseArray<>(0);
/**
* List of touch interaction controllers, mapping from displayId -> TouchInteractionController.
*/
private final SparseArray<TouchInteractionController> mTouchInteractionControllers =
new SparseArray<>(0);

private SoftKeyboardController mSoftKeyboardController;
private InputMethod mInputMethod;
private boolean mInputMethodInitialized = false;
private final SparseArray<AccessibilityButtonController> mAccessibilityButtonControllers =
new SparseArray<>(0);

private int mGestureStatusCallbackSequence;

private SparseArray<GestureResultCallbackInfo> mGestureStatusCallbackInfos;

private final Object mLock = new Object();

private FingerprintGestureController mFingerprintGestureController;

private int mMotionEventSources;

/**
* Callback for {@link android.view.accessibility.AccessibilityEvent}s.
*
* @param event The new event. This event is owned by the caller and cannot be used after
* this method returns. Services wishing to use the event after this method returns should
* make a copy.
*/
public abstract void onAccessibilityEvent(AccessibilityEvent event);

/**
* Callback for interrupting the accessibility feedback.
*/
public abstract void onInterrupt();
```

```
/**
* Dispatches service connection to internal components first, then the
* client code.
*/
private void dispatchServiceConnected() {
synchronized (mLock) {
for (int i = 0; i < mMagnificationControllers.size(); i++) {
mMagnificationControllers.valueAt(i).onServiceConnectedLocked();
}
final AccessibilityServiceInfo info = getServiceInfo();
if (info != null) {
updateInputMethod(info);
mMotionEventSources = info.getMotionEventSources();
}
}
if (mSoftKeyboardController != null) {
mSoftKeyboardController.onServiceConnected();
}

// The client gets to handle service connection last, after we've set
// up any state upon which their code may rely.
onServiceConnected();
}

private void updateInputMethod(AccessibilityServiceInfo info) {
if (info != null) {
boolean requestIme = (info.flags
& AccessibilityServiceInfo.FLAG_INPUT_METHOD_EDITOR) != 0;
if (requestIme && !mInputMethodInitialized) {
mInputMethod = onCreateInputMethod();
mInputMethodInitialized = true;
} else if (!requestIme & mInputMethodInitialized) {
mInputMethod = null;
mInputMethodInitialized = false;
}
}
}

/**
* This method is a part of the {@link AccessibilityService} lifecycle and is
* called after the system has successfully bound to the service. If is
* convenient to use this method for setting the {@link AccessibilityServiceInfo}.
*
* @see AccessibilityServiceInfo
* @see #setServiceInfo(AccessibilityServiceInfo)
*/
protected void onServiceConnected() {

}

/**
* Called by {@link #onGesture(AccessibilityGestureEvent)} when the user performs a specific
* gesture on the default display.
```

```
*
* <strong>Note:</strong> To receive gestures an accessibility service must
* request that the device is in touch exploration mode by setting the
* {@link AccessibilityServiceInfo#FLAG_REQUEST_TOUCH_EXPLORATION_MODE}
* flag.
*
* @param gestureId The unique id of the performed gesture.
*
* @return Whether the gesture was handled.
* @deprecated Override {@link #onGesture(AccessibilityGestureEvent)} instead.
*
* @see #GESTURE_SWIPE_UP
* @see #GESTURE_SWIPE_UP_AND_LEFT
* @see #GESTURE_SWIPE_UP_AND_DOWN
* @see #GESTURE_SWIPE_UP_AND_RIGHT
* @see #GESTURE_SWIPE_DOWN
* @see #GESTURE_SWIPE_DOWN_AND_LEFT
* @see #GESTURE_SWIPE_DOWN_AND_UP
* @see #GESTURE_SWIPE_DOWN_AND_RIGHT
* @see #GESTURE_SWIPE_LEFT
* @see #GESTURE_SWIPE_LEFT_AND_UP
* @see #GESTURE_SWIPE_LEFT_AND_RIGHT
* @see #GESTURE_SWIPE_LEFT_AND_DOWN
* @see #GESTURE_SWIPE_RIGHT
* @see #GESTURE_SWIPE_RIGHT_AND_UP
* @see #GESTURE_SWIPE_RIGHT_AND_LEFT
* @see #GESTURE_SWIPE_RIGHT_AND_DOWN
*/
@Deprecated
protected boolean onGesture(int gestureId) {
return false;
}

/**
* Called by the system when the user performs a specific gesture on the
* specific touch screen.
*<p>
* <strong>Note:</strong> To receive gestures an accessibility service must
* request that the device is in touch exploration mode by setting the
* {@link AccessibilityServiceInfo#FLAG_REQUEST_TOUCH_EXPLORATION_MODE}
* flag.
*<p>
* <strong>Note:</strong> The default implementation calls {@link #onGesture(int)} when the
* touch screen is default display.
*
* @param gestureEvent The information of gesture.
*
* @return Whether the gesture was handled.
*
*/
public boolean onGesture(@NonNull AccessibilityGestureEvent gestureEvent) {
if (gestureEvent.getDisplayId() == Display.DEFAULT_DISPLAY) {
onGesture(gestureEvent.getGestureId());
}
return false;
}

/**
* Callback that allows an accessibility service to observe the key events
* before they are passed to the rest of the system. This means that the events
* are first delivered here before they are passed to the device policy, the
* input method, or applications.
* <p>
* <strong>Note:</strong> It is important that key events are handled in such
* a way that the event stream that would be passed to the rest of the system
* is well-formed. For example, handling the down event but not the up event
* and vice versa would generate an inconsistent event stream.
* </p>
* <p>
* <strong>Note:</strong> The key events delivered in this method are copies
* and modifying them will have no effect on the events that will be passed
* to the system. This method is intended to perform purely filtering
* functionality.
* <p>
*
* @param event The event to be processed. This event is owned by the caller and cannot be used
* after this method returns. Services wishing to use the event after this method returns should
* make a copy.
* @return If true then the event will be consumed and not delivered to
* applications, otherwise it will be delivered as usual.
*/
protected boolean onKeyEvent(KeyEvent event) {
return false;
}

/**
* Callback that allows an accessibility service to observe generic {@link MotionEvent}s.
* <p>
* Prefer {@link TouchInteractionController} to observe and control touchscreen events,
* including touch gestures. If this or any enabled service is using
* {@link AccessibilityServiceInfo#FLAG_REQUEST_TOUCH_EXPLORATION_MODE} then
* {@link #onMotionEvent} will not receive touchscreen events.
* </p>
* <p>
* <strong>Note:</strong> The service must first request to listen to events using
* {@link AccessibilityServiceInfo#setMotionEventSources}.
* {@link MotionEvent}s from sources in {@link AccessibilityServiceInfo#getMotionEventSources()}
* are not sent to the rest of the system. To stop listening to events from a given source, call
* {@link AccessibilityServiceInfo#setMotionEventSources} with that source removed.
* </p>
* @param event The event to be processed.
*/
public void onMotionEvent(@NonNull MotionEvent event) { }

/**
* Gets the windows on the screen of the default display. This method returns only the windows
* that a sighted user can interact with, as opposed to all windows.
* For example, if there is a modal dialog shown and the user cannot touch
* anything behind it, then only the modal window will be reported
* (assuming it is the top one). For convenience the returned windows
* are ordered in a descending layer order, which is the windows that
* are on top are reported first. Since the user can always
* interact with the window that has input focus by typing, the focused
* window is always returned (even if covered by a modal window).
* <p>
* <strong>Note:</strong> In order to access the windows your service has
* to declare the capability to retrieve window content by setting the
* {@link android.R.styleable#AccessibilityService_canRetrieveWindowContent}
* property in its meta-data. For details refer to {@link #SERVICE_META_DATA}.
* Also the service has to opt-in to retrieve the interactive windows by
* setting the {@link AccessibilityServiceInfo#FLAG_RETRIEVE_INTERACTIVE_WINDOWS}
* flag.
* </p>
*
* @return The windows if there are windows and the service is can retrieve
* them, otherwise an empty list.
*/
public List<AccessibilityWindowInfo> getWindows() {
return AccessibilityInteractionClient.getInstance(this).getWindows(mConnectionId);
}

/**
* Gets the windows on the screen of all displays. This method returns only the windows
* that a sighted user can interact with, as opposed to all windows.
* For example, if there is a modal dialog shown and the user cannot touch
* anything behind it, then only the modal window will be reported
* (assuming it is the top one). For convenience the returned windows
* are ordered in a descending layer order, which is the windows that
* are on top are reported first. Since the user can always
* interact with the window that has input focus by typing, the focused
* window is always returned (even if covered by a modal window).
* <p>
* <strong>Note:</strong> In order to access the windows your service has
* to declare the capability to retrieve window content by setting the
* {@link android.R.styleable#AccessibilityService_canRetrieveWindowContent}
* property in its meta-data. For details refer to {@link #SERVICE_META_DATA}.
* Also the service has to opt-in to retrieve the interactive windows by
* setting the {@link AccessibilityServiceInfo#FLAG_RETRIEVE_INTERACTIVE_WINDOWS}
* flag.
* </p>
*
* @return The windows of all displays if there are windows and the service is can retrieve
* them, otherwise an empty list. The key of SparseArray is display ID.
*/
@NonNull
public final SparseArray<List<AccessibilityWindowInfo>> getWindowsOnAllDisplays() {
return AccessibilityInteractionClient.getInstance(this).getWindowsOnAllDisplays(
mConnectionId);
}

/**
* Gets the root node in the currently active window if this service
* can retrieve window content. The active window is the one that the user
* is currently touching or the window with input focus, if the user is not
* touching any window. It could be from any logical display.
* <p>
* <strong>Note:</strong> In order to access the root node your service has
* to declare the capability to retrieve window content by setting the
* {@link android.R.styleable#AccessibilityService_canRetrieveWindowContent}
* property in its meta-data. For details refer to {@link #SERVICE_META_DATA}.
* </p>
*
* @return The root node if this service can retrieve window content.
* @see AccessibilityWindowInfo#isActive() for more explanation about the active window.
*/
public AccessibilityNodeInfo getRootInActiveWindow() {
return getRootInActiveWindow(AccessibilityNodeInfo.FLAG_PREFETCH_DESCENDANTS_HYBRID);
}

/**
* Gets the root node in the currently active window if this service
* can retrieve window content. The active window is the one that the user
* is currently touching or the window with input focus, if the user is not
* touching any window. It could be from any logical display.
*
* @param prefetchingStrategy the prefetching strategy.
* @return The root node if this service can retrieve window content.
*
* @see #getRootInActiveWindow()
* @see AccessibilityNodeInfo#getParent(int) for a description of prefetching.
*/
@Nullable
public AccessibilityNodeInfo getRootInActiveWindow(
@AccessibilityNodeInfo.PrefetchingStrategy int prefetchingStrategy) {
return AccessibilityInteractionClient.getInstance(this).getRootInActiveWindow(
mConnectionId, prefetchingStrategy);
}

/**
* Disables the service. After calling this method, the service will be disabled and settings
* will show that it is turned off.
*/
public final void disableSelf() {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(this).getConnection(mConnectionId);
if (connection != null) {
try {
connection.disableSelf();
} catch (RemoteException re) {
throw new RuntimeException(re);
}
}
}

@NonNull
@Override
```

```
public Context createDisplayContext(Display display) {
return new AccessibilityContext(super.createDisplayContext(display), mConnectionId);
}

@NonNull
@Override
public Context createWindowContext(int type, @Nullable Bundle options) {
final Context context = super.createWindowContext(type, options);
if (type != TYPE_ACCESSIBILITY_OVERLAY) {
return context;
}
return new AccessibilityContext(context, mConnectionId);
}

@NonNull
@Override
public Context createWindowContext(@NonNull Display display, int type,
@Nullable Bundle options) {
final Context context = super.createWindowContext(display, type, options);
if (type != TYPE_ACCESSIBILITY_OVERLAY) {
return context;
}
return new AccessibilityContext(context, mConnectionId);
}

/**
* Returns the magnification controller, which may be used to query and
* modify the state of display magnification.
* <p>
* <strong>Note:</strong> In order to control magnification, your service
* must declare the capability by setting the
* {@link android.R.styleable#AccessibilityService_canControlMagnification}
* property in its meta-data. For more information, see
* {@link #SERVICE_META_DATA}.
*
* @return the magnification controller
*/
@NonNull
public final MagnificationController getMagnificationController() {
return getMagnificationController(Display.DEFAULT_DISPLAY);
}

/**
* Returns the magnification controller of specified logical display, which may be used to
* query and modify the state of display magnification.
* <p>
* <strong>Note:</strong> In order to control magnification, your service
* must declare the capability by setting the
* {@link android.R.styleable#AccessibilityService_canControlMagnification}
* property in its meta-data. For more information, see
* {@link #SERVICE_META_DATA}.
*
* @param displayId The logic display id, use {@link Display#DEFAULT_DISPLAY} for
* default display.
* @return the magnification controller
*
* @hide
*/
@NonNull
public final MagnificationController getMagnificationController(int displayId) {
synchronized (mLock) {
MagnificationController controller = mMagnificationControllers.get(displayId);
if (controller == null) {
controller = new MagnificationController(this, mLock, displayId);
mMagnificationControllers.put(displayId, controller);
}
return controller;
}
}

/**
* Get the controller for fingerprint gestures. This feature requires {@link
* AccessibilityServiceInfo#CAPABILITY_CAN_REQUEST_FINGERPRINT_GESTURES}.
*
*<strong>Note: </strong> The service must be connected before this method is called.
*
* @return The controller for fingerprint gestures, or {@code null} if gestures are unavailable.
*/
@RequiresPermission(android.Manifest.permission.USE_FINGERPRINT)
public final @NonNull FingerprintGestureController getFingerprintGestureController() {
if (mFingerprintGestureController == null) {
mFingerprintGestureController = new FingerprintGestureController(
AccessibilityInteractionClient.getInstance(this).getConnection(mConnectionId));
}
return mFingerprintGestureController;
}

/**
* Dispatch a gesture to the touch screen. Any gestures currently in progress, whether from
* the user, this service, or another service, will be cancelled.
* <p>
* The gesture will be dispatched as if it were performed directly on the screen by a user, so
* the events may be affected by features such as magnification and explore by touch.
* </p>
* <p>
* <strong>Note:</strong> In order to dispatch gestures, your service
* must declare the capability by setting the
* {@link android.R.styleable#AccessibilityService_canPerformGestures}
* property in its meta-data. For more information, see
* {@link #SERVICE_META_DATA}.
* </p>
*
* @param gesture The gesture to dispatch
* @param callback The object to call back when the status of the gesture is known. If
* {@code null}, no status is reported.
* @param handler The handler on which to call back the {@code callback} object. If
* {@code null}, the object is called back on the service's main thread.
*
```

```
* @return {@code true} if the gesture is dispatched, {@code false} if not.
*/
public final boolean dispatchGesture(@NonNull GestureDescription gesture,
@Nullable GestureResultCallback callback,
@Nullable Handler handler) {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(this).getConnection(mConnectionId);
if (connection == null) {
return false;
}
int sampleTimeMs = calculateGestureSampleTimeMs(gesture.getDisplayId());
List<GestureDescription.GestureStep> steps =
MotionEventGenerator.getGestureStepsFromGestureDescription(gesture, sampleTimeMs);
try {
synchronized (mLock) {
mGestureStatusCallbackSequence++;
if (callback != null) {
if (mGestureStatusCallbackInfos == null) {
mGestureStatusCallbackInfos = new SparseArray<>();
}
GestureResultCallbackInfo callbackInfo = new GestureResultCallbackInfo(gesture,
callback, handler);
mGestureStatusCallbackInfos.put(mGestureStatusCallbackSequence, callbackInfo);
}
connection.dispatchGesture(mGestureStatusCallbackSequence,
new ParceledListSlice<>(steps), gesture.getDisplayId());
}
} catch (RemoteException re) {
throw new RuntimeException(re);
}
return true;
}

/**
* Returns the sample time in millis of gesture steps for the current display.
*
* <p>For gestures to be smooth they should line up with the refresh rate of the display.
* On versions of Android before R, the sample time was fixed to 100ms.
*/
private int calculateGestureSampleTimeMs(int displayId) {
if (getApplicationInfo().targetSdkVersion <= Build.VERSION_CODES.Q) {
return 100;
}
Display display = getSystemService(DisplayManager.class).getDisplay(
displayId);
if (display == null) {
return 100;
}
int msPerSecond = 1000;
int sampleTimeMs = (int) (msPerSecond / display.getRefreshRate());
if (sampleTimeMs < 1) {
// Should be impossible, but do not return 0.
return 100;
}
```

```
return sampleTimeMs;
}

void onPerformGestureResult(int sequence, final boolean completedSuccessfully) {
if (mGestureStatusCallbackInfos == null) {
return;
}
GestureResultCallbackInfo callbackInfo;
synchronized (mLock) {
callbackInfo = mGestureStatusCallbackInfos.get(sequence);
mGestureStatusCallbackInfos.remove(sequence);
}
final GestureResultCallbackInfo finalCallbackInfo = callbackInfo;
if ((callbackInfo != null) && (callbackInfo.gestureDescription != null)
&& (callbackInfo.callback != null)) {
if (callbackInfo.handler != null) {
callbackInfo.handler.post(new Runnable() {
@Override
public void run() {
if (completedSuccessfully) {
finalCallbackInfo.callback
.onCompleted(finalCallbackInfo.gestureDescription);
} else {
finalCallbackInfo.callback
.onCancelled(finalCallbackInfo.gestureDescription);
}
}
});
return;
}
if (completedSuccessfully) {
callbackInfo.callback.onCompleted(callbackInfo.gestureDescription);
} else {
callbackInfo.callback.onCancelled(callbackInfo.gestureDescription);
}
}
}

private void onMagnificationChanged(int displayId, @NonNull Region region,
MagnificationConfig config) {
MagnificationController controller;
synchronized (mLock) {
controller = mMagnificationControllers.get(displayId);
}
if (controller != null) {
controller.dispatchMagnificationChanged(region, config);
}
}

/**
* Callback for fingerprint gesture handling
* @param active If gesture detection is active
*/
private void onFingerprintCapturingGesturesChanged(boolean active) {
```

```
getFingerprintGestureController().onGestureDetectionActiveChanged(active);
}

/**
* Callback for fingerprint gesture handling
* @param gesture The identifier for the gesture performed
*/
private void onFingerprintGesture(int gesture) {
getFingerprintGestureController().onGesture(gesture);
}

int getConnectionId() {
return mConnectionId;
}

/**
* Used to control and query the state of display magnification.
*/
public static final class MagnificationController {
private final AccessibilityService mService;
private final int mDisplayId;

/**
* Map of listeners to their handlers. Lazily created when adding the
* first magnification listener.
*/
private ArrayMap<OnMagnificationChangedListener, Handler> mListeners;
private final Object mLock;

MagnificationController(@NonNull AccessibilityService service, @NonNull Object lock,
int displayId) {
mService = service;
mLock = lock;
mDisplayId = displayId;
}

/**
* Called when the service is connected.
*/
void onServiceConnectedLocked() {
if (mListeners != null && !mListeners.isEmpty()) {
setMagnificationCallbackEnabled(true);
}
}

/**
* Adds the specified change listener to the list of magnification
* change listeners. The callback will occur on the service's main
* thread.
*
* @param listener the listener to add, must be non-{@code null}
*/
public void addListener(@NonNull OnMagnificationChangedListener listener) {
```

```
addListener(listener, null);
}

/**
* Adds the specified change listener to the list of magnification
* change listeners. The callback will occur on the specified
* {@link Handler}'s thread, or on the service's main thread if the
* handler is {@code null}.
*
* @param listener the listener to add, must be non-null
* @param handler the handler on which the callback should execute, or
* {@code null} to execute on the service's main thread
*/
public void addListener(@NonNull OnMagnificationChangedListener listener,
@Nullable Handler handler) {
synchronized (mLock) {
if (mListeners == null) {
mListeners = new ArrayMap<>();
}

final boolean shouldEnableCallback = mListeners.isEmpty();
mListeners.put(listener, handler);

if (shouldEnableCallback) {
// This may fail if the service is not connected yet, but if we
// still have listeners when it connects then we can try again.
setMagnificationCallbackEnabled(true);
}
}
}

/**
* Removes the specified change listener from the list of magnification change listeners.
*
* @param listener the listener to remove, must be non-null
* @return {@code true} if the listener was removed, {@code false} otherwise
*/
public boolean removeListener(@NonNull OnMagnificationChangedListener listener) {
if (mListeners == null) {
return false;
}

synchronized (mLock) {
final int keyIndex = mListeners.indexOfKey(listener);
final boolean hasKey = keyIndex >= 0;
if (hasKey) {
mListeners.removeAt(keyIndex);
}

if (hasKey && mListeners.isEmpty()) {
// We just removed the last listener, so we don't need
// callbacks from the service anymore.
setMagnificationCallbackEnabled(false);
}

return hasKey;
}
}

private void setMagnificationCallbackEnabled(boolean enabled) {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
connection.setMagnificationCallbackEnabled(mDisplayId, enabled);
} catch (RemoteException re) {
throw new RuntimeException(re);
}
}
}

/**
* Dispatches magnification changes to any registered listeners. This
* should be called on the service's main thread.
*/
void dispatchMagnificationChanged(final @NonNull Region region,
final MagnificationConfig config) {
final ArrayMap<OnMagnificationChangedListener, Handler> entries;
synchronized (mLock) {
if (mListeners == null || mListeners.isEmpty()) {
Slog.d(LOG_TAG, "Received magnification changed "
+ "callback with no listeners registered!");
setMagnificationCallbackEnabled(false);
return;
}

// Listeners may remove themselves. Perform a shallow copy to avoid concurrent
// modification.
entries = new ArrayMap<>(mListeners);
}

for (int i = 0, count = entries.size(); i < count; i++) {
final OnMagnificationChangedListener listener = entries.keyAt(i);
final Handler handler = entries.valueAt(i);
if (handler != null) {
handler.post(() -> {
listener.onMagnificationChanged(MagnificationController.this,
region, config);
});
} else {
// We're already on the main thread, just run the listener.
listener.onMagnificationChanged(this, region, config);
}
}
}
```

```
/**
* Gets the {@link MagnificationConfig} of the controlling magnifier on the display.
* <p>
* <strong>Note:</strong> If the service is not yet connected (e.g.
* {@link AccessibilityService#onServiceConnected()} has not yet been
* called) or the service has been disconnected, this method will
* return null.
* </p>
*
* @return the magnification config that the service controls
*/
public @Nullable MagnificationConfig getMagnificationConfig() {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
return connection.getMagnificationConfig(mDisplayId);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to obtain magnification config", re);
re.rethrowFromSystemServer();
}
}
return null;
}

/**
* Returns the current magnification scale.
* <p>
* <strong>Note:</strong> If the service is not yet connected (e.g.
* {@link AccessibilityService#onServiceConnected()} has not yet been
* called) or the service has been disconnected, this method will
* return a default value of {@code 1.0f}.
* </p>
* <p>
* <strong>Note:</strong> This legacy API gets the scale of full-screen
* magnification. To get the scale of the current controlling magnifier,
* use {@link #getMagnificationConfig} instead.
* </p>
*
* @return the current magnification scale
* @deprecated Use {@link #getMagnificationConfig()} instead
*/
@Deprecated
public float getScale() {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
return connection.getMagnificationScale(mDisplayId);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to obtain scale", re);
re.rethrowFromSystemServer();
}
}
return 1.0f;
}

/**
* Returns the unscaled screen-relative X coordinate of the focal
* center of the magnified region. This is the point around which
* zooming occurs and is guaranteed to lie within the magnified
* region.
* <p>
* <strong>Note:</strong> If the service is not yet connected (e.g.
* {@link AccessibilityService#onServiceConnected()} has not yet been
* called) or the service has been disconnected, this method will
* return a default value of {@code 0.0f}.
* </p>
* <p>
* <strong>Note:</strong> This legacy API gets the center position of full-screen
* magnification. To get the magnification center of the current controlling magnifier,
* use {@link #getMagnificationConfig} instead.
* </p>
*
* @return the unscaled screen-relative X coordinate of the center of
* the magnified region
* @deprecated Use {@link #getMagnificationConfig()} instead
*/
@Deprecated
public float getCenterX() {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
return connection.getMagnificationCenterX(mDisplayId);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to obtain center X", re);
re.rethrowFromSystemServer();
}
}
return 0.0f;
}

/**
* Returns the unscaled screen-relative Y coordinate of the focal
* center of the magnified region. This is the point around which
* zooming occurs and is guaranteed to lie within the magnified
* region.
* <p>
* <strong>Note:</strong> If the service is not yet connected (e.g.
* {@link AccessibilityService#onServiceConnected()} has not yet been
* called) or the service has been disconnected, this method will
* return a default value of {@code 0.0f}.
* </p>
```

```java
* <p>
* <strong>Note:</strong> This legacy API gets the center position of full-screen
* magnification. To get the magnification center of the current controlling magnifier,
* use {@link #getMagnificationConfig} instead.
* </p>
*
* @return the unscaled screen-relative Y coordinate of the center of
* the magnified region
* @deprecated Use {@link #getMagnificationConfig()} instead
*/
@Deprecated
public float getCenterY() {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
return connection.getMagnificationCenterY(mDisplayId);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to obtain center Y", re);
re.rethrowFromSystemServer();
}
}
return 0.0f;
}

/**
* Returns the region of the screen currently active for magnification. Changes to
* magnification scale and center only affect this portion of the screen. The rest of the
* screen, for example input methods, cannot be magnified. This region is relative to the
* unscaled screen and is independent of the scale and center point.
* <p>
* The returned region will be empty if magnification is not active. Magnification is active
* if magnification gestures are enabled or if a service is running that can control
* magnification.
* <p>
* <strong>Note:</strong> If the service is not yet connected (e.g.
* {@link AccessibilityService#onServiceConnected()} has not yet been
* called) or the service has been disconnected, this method will
* return an empty region.
* </p>
* <p>
* <strong>Note:</strong> This legacy API gets the magnification region of full-screen
* magnification. To get the magnification region of the current controlling magnifier,
* use {@link #getCurrentMagnificationRegion()} instead.
* </p>
*
* @return the region of the screen currently active for magnification, or an empty region
* if magnification is not active.
* @deprecated Use {@link #getCurrentMagnificationRegion()} instead
*/
@Deprecated
@NonNull
public Region getMagnificationRegion() {
```

```
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
return connection.getMagnificationRegion(mDisplayId);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to obtain magnified region", re);
re.rethrowFromSystemServer();
}
}
return Region.obtain();
}

/**
* Returns the region of the screen currently active for magnification if the
* controlling magnification is {@link MagnificationConfig#MAGNIFICATION_MODE_FULLSCREEN}.
* Returns the region of screen projected on the magnification window if the
* controlling magnification is {@link MagnificationConfig#MAGNIFICATION_MODE_WINDOW}.
*
* <p>
* If the controlling mode is {@link MagnificationConfig#MAGNIFICATION_MODE_FULLSCREEN},
* the returned region will be empty if the magnification is
* not active. And the magnification is active if magnification gestures are enabled
* or if a service is running that can control magnification.
* </p><p>
* If the controlling mode is {@link MagnificationConfig#MAGNIFICATION_MODE_WINDOW},
* the returned region will be empty if the magnification is not activated.
* </p><p>
* <strong>Note:</strong> If the service is not yet connected (e.g.
* {@link AccessibilityService#onServiceConnected()} has not yet been
* called) or the service has been disconnected, this method will
* return an empty region.
* </p>
*
* @return the magnification region of the currently controlling magnification
*/
@NonNull
public Region getCurrentMagnificationRegion() {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
return connection.getCurrentMagnificationRegion(mDisplayId);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to obtain the current magnified region", re);
re.rethrowFromSystemServer();
}
}
return Region.obtain();
}

/**
```

```
* Resets magnification scale and center to their default (e.g. no
* magnification) values.
* <p>
* <strong>Note:</strong> If the service is not yet connected (e.g.
* {@link AccessibilityService#onServiceConnected()} has not yet been
* called) or the service has been disconnected, this method will have
* no effect and return {@code false}.
* <p>
* <strong>Note:</strong> This legacy API reset full-screen magnification.
* To reset the current controlling magnifier, use
* {@link #resetCurrentMagnification(boolean)} ()} instead.
* </p>
*
* @param animate {@code true} to animate from the current scale and
* center or {@code false} to reset the scale and center
* immediately
* @return {@code true} on success, {@code false} on failure
*/
public boolean reset(boolean animate) {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
return connection.resetMagnification(mDisplayId, animate);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to reset", re);
re.rethrowFromSystemServer();
}
}
return false;
}

/**
* Resets magnification scale and center of the controlling magnification
* to their default (e.g. no magnification) values.
* <p>
* <strong>Note:</strong> If the service is not yet connected (e.g.
* {@link AccessibilityService#onServiceConnected()} has not yet been
* called) or the service has been disconnected, this method will have
* no effect and return {@code false}.
* </p>
*
* @param animate {@code true} to animate from the current scale and
* center or {@code false} to reset the scale and center
* immediately
* @return {@code true} on success, {@code false} on failure
*/
public boolean resetCurrentMagnification(boolean animate) {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
```

```
return connection.resetCurrentMagnification(mDisplayId, animate);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to reset", re);
re.rethrowFromSystemServer();
}
}
return false;
}

/**
* Sets the {@link MagnificationConfig}. The service controls the magnification by
* setting the config.
* <p>
* <strong>Note:</strong> If the service is not yet connected (e.g.
* {@link AccessibilityService#onServiceConnected()} has not yet been
* called) or the service has been disconnected, this method will have
* no effect and return {@code false}.
* </p>
*
* @param config the magnification config
* @param animate {@code true} to animate from the current spec or
* {@code false} to set the spec immediately
* @return {@code true} on success, {@code false} on failure
*/
public boolean setMagnificationConfig(@NonNull MagnificationConfig config,
boolean animate) {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
return connection.setMagnificationConfig(mDisplayId, config, animate);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to set magnification config", re);
re.rethrowFromSystemServer();
}
}
return false;
}

/**
* Sets the magnification scale.
* <p>
* <strong>Note:</strong> If the service is not yet connected (e.g.
* {@link AccessibilityService#onServiceConnected()} has not yet been
* called) or the service has been disconnected, this method will have
* no effect and return {@code false}.
* <p>
* <strong>Note:</strong> This legacy API sets the scale of full-screen
* magnification. To set the scale of the specified magnifier,
* use {@link #setMagnificationConfig} instead.
* </p>
*
* @param scale the magnification scale to set, must be >= 1 and <= 8
* @param animate {@code true} to animate from the current scale or
* {@code false} to set the scale immediately
* @return {@code true} on success, {@code false} on failure
* @deprecated Use {@link #setMagnificationConfig(MagnificationConfig, boolean)} instead
*/
@Deprecated
public boolean setScale(float scale, boolean animate) {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
final MagnificationConfig config = new MagnificationConfig.Builder()
.setMode(MAGNIFICATION_MODE_FULLSCREEN)
.setScale(scale).build();
return connection.setMagnificationConfig(mDisplayId, config, animate);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to set scale", re);
re.rethrowFromSystemServer();
}
}
return false;
}

/**
* Sets the center of the magnified viewport.
* <p>
* <strong>Note:</strong> If the service is not yet connected (e.g.
* {@link AccessibilityService#onServiceConnected()} has not yet been
* called) or the service has been disconnected, this method will have
* no effect and return {@code false}.
* </p>
* <p>
* <strong>Note:</strong> This legacy API sets the center of full-screen
* magnification. To set the center of the specified magnifier,
* use {@link #setMagnificationConfig} instead.
* </p>
*
* @param centerX the unscaled screen-relative X coordinate on which to
* center the viewport
* @param centerY the unscaled screen-relative Y coordinate on which to
* center the viewport
* @param animate {@code true} to animate from the current viewport
* center or {@code false} to set the center immediately
* @return {@code true} on success, {@code false} on failure
* @deprecated Use {@link #setMagnificationConfig(MagnificationConfig, boolean)} instead
*/
@Deprecated
public boolean setCenter(float centerX, float centerY, boolean animate) {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
```

```
final MagnificationConfig config = new MagnificationConfig.Builder()
.setMode(MAGNIFICATION_MODE_FULLSCREEN)
.setCenterX(centerX).setCenterY(centerY).build();
return connection.setMagnificationConfig(mDisplayId, config, animate);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to set center", re);
re.rethrowFromSystemServer();
}
}
return false;
}

/**
* Listener for changes in the state of magnification.
*/
public interface OnMagnificationChangedListener {
/**
* Called when the magnified region, scale, or center changes.
* <p>
* <strong>Note:</strong> This legacy callback notifies only full-screen
* magnification change.
* </p>
*
* @param controller the magnification controller
* @param region the magnification region
* @param scale the new scale
* @param centerX the new X coordinate, in unscaled coordinates, around which
* magnification is focused
* @param centerY the new Y coordinate, in unscaled coordinates, around which
* magnification is focused
* @deprecated Override
* {@link #onMagnificationChanged(MagnificationController, Region, MagnificationConfig)}
* instead
*/
@Deprecated
void onMagnificationChanged(@NonNull MagnificationController controller,
@NonNull Region region, float scale, float centerX, float centerY);

/**
* Called when the magnified region, mode, scale, or center changes of
* all magnification modes.
* <p>
* <strong>Note:</strong> This method can be overridden to listen to the
* magnification changes of all magnification modes then the legacy callback
* would not receive the notifications.
* Skipping calling super when overriding this method results in
* {@link #onMagnificationChanged(MagnificationController, Region, float, float, float)}
* not getting called.
* </p>
*
* @param controller the magnification controller
* @param region the magnification region
* If the config mode is
* {@link MagnificationConfig#MAGNIFICATION_MODE_FULLSCREEN},
* it is the region of the screen currently active for magnification.
* that is the same region as {@link #getMagnificationRegion()}.
* If the config mode is
* {@link MagnificationConfig#MAGNIFICATION_MODE_WINDOW},
* it is the region of screen projected on the magnification window.
* @param config The magnification config. That has the controlling magnification
* mode, the new scale and the new screen-relative center position
*/
default void onMagnificationChanged(@NonNull MagnificationController controller,
@NonNull Region region, @NonNull MagnificationConfig config) {
if (config.getMode() == MAGNIFICATION_MODE_FULLSCREEN) {
onMagnificationChanged(controller, region,
config.getScale(), config.getCenterX(), config.getCenterY());
}
}
}
}

/**
* Returns the soft keyboard controller, which may be used to query and modify the soft keyboard
* show mode.
*
* @return the soft keyboard controller
*/
@NonNull
public final SoftKeyboardController getSoftKeyboardController() {
synchronized (mLock) {
if (mSoftKeyboardController == null) {
mSoftKeyboardController = new SoftKeyboardController(this, mLock);
}
return mSoftKeyboardController;
}
}

/**
* The default implementation returns our default {@link InputMethod}. Subclasses can override
* it to provide their own customized version. Accessibility services need to set the
* {@link AccessibilityServiceInfo#FLAG_INPUT_METHOD_EDITOR} flag to use input method APIs.
*
* @return the InputMethod.
*/
@NonNull
public InputMethod onCreateInputMethod() {
return new InputMethod(this);
}

/**
* Returns the InputMethod instance after the system calls {@link #onCreateInputMethod()},
* which may be used to input text or get editable text selection change notifications. It will
* return null if the accessibility service doesn't set the
* {@link AccessibilityServiceInfo#FLAG_INPUT_METHOD_EDITOR} flag or the system doesn't call
* {@link #onCreateInputMethod()}.
*
* @return the InputMethod instance
```

```
*/
@Nullable
public final InputMethod getInputMethod() {
return mInputMethod;
}

private void onSoftKeyboardShowModeChanged(int showMode) {
if (mSoftKeyboardController != null) {
mSoftKeyboardController.dispatchSoftKeyboardShowModeChanged(showMode);
}
}

/**
* Used to control, query, and listen for changes to the soft keyboard show mode.
* <p>
* Accessibility services may request to override the decisions normally made about whether or
* not the soft keyboard is shown.
* <p>
* If multiple services make conflicting requests, the last request is honored. A service may
* register a listener to find out if the mode has changed under it.
* <p>
* If the user takes action to override the behavior behavior requested by an accessibility
* service, the user's request takes precendence, the show mode will be reset to
* {@link AccessibilityService#SHOW_MODE_AUTO}, and services will no longer be able to control
* that aspect of the soft keyboard's behavior.
* <p>
* Note: Because soft keyboards are independent apps, the framework does not have total control
* over their behavior. They may choose to show themselves, or not, without regard to requests
* made here. So the framework will make a best effort to deliver the behavior requested, but
* cannot guarantee success.
*
* @see AccessibilityService#SHOW_MODE_AUTO
* @see AccessibilityService#SHOW_MODE_HIDDEN
* @see AccessibilityService#SHOW_MODE_IGNORE_HARD_KEYBOARD
*/
public static final class SoftKeyboardController {
private final AccessibilityService mService;

/**
* Map of listeners to their handlers. Lazily created when adding the first
* soft keyboard change listener.
*/
private ArrayMap<OnShowModeChangedListener, Handler> mListeners;
private final Object mLock;

/** @hide */
@Retention(RetentionPolicy.SOURCE)
@IntDef({
ENABLE_IME_SUCCESS,
ENABLE_IME_FAIL_BY_ADMIN,
ENABLE_IME_FAIL_UNKNOWN
})
public @interface EnableImeResult {}
/**
```

```
* Return value for {@link #setInputMethodEnabled(String, boolean)}. The action succeeded.
*/
public static final int ENABLE_IME_SUCCESS = 0;
/**
* Return value for {@link #setInputMethodEnabled(String, boolean)}. The action failed
* because the InputMethod is not permitted by device policy manager.
*/
public static final int ENABLE_IME_FAIL_BY_ADMIN = 1;
/**
* Return value for {@link #setInputMethodEnabled(String, boolean)}. The action failed
* and the reason is unknown.
*/
public static final int ENABLE_IME_FAIL_UNKNOWN = 2;

SoftKeyboardController(@NonNull AccessibilityService service, @NonNull Object lock) {
mService = service;
mLock = lock;
}

/**
* Called when the service is connected.
*/
void onServiceConnected() {
synchronized(mLock) {
if (mListeners != null && !mListeners.isEmpty()) {
setSoftKeyboardCallbackEnabled(true);
}
}
}

/**
* Adds the specified change listener to the list of show mode change listeners. The
* callback will occur on the service's main thread. Listener is not called on registration.
*/
public void addOnShowModeChangedListener(@NonNull OnShowModeChangedListener listener) {
addOnShowModeChangedListener(listener, null);
}

/**
* Adds the specified change listener to the list of soft keyboard show mode change
* listeners. The callback will occur on the specified {@link Handler}'s thread, or on the
* services's main thread if the handler is {@code null}.
*
* @param listener the listener to add, must be non-null
* @param handler the handler on which to callback should execute, or {@code null} to
* execute on the service's main thread
*/
public void addOnShowModeChangedListener(@NonNull OnShowModeChangedListener listener,
@Nullable Handler handler) {
synchronized (mLock) {
if (mListeners == null) {
mListeners = new ArrayMap<>();
}
final boolean shouldEnableCallback = mListeners.isEmpty();
mListeners.put(listener, handler);

if (shouldEnableCallback) {
// This may fail if the service is not connected yet, but if we still have
// listeners when it connects, we can try again.
setSoftKeyboardCallbackEnabled(true);
}
}
}

/**
* Removes the specified change listener from the list of keyboard show mode change
* listeners.
*
* @param listener the listener to remove, must be non-null
* @return {@code true} if the listener was removed, {@code false} otherwise
*/
public boolean removeOnShowModeChangedListener(
@NonNull OnShowModeChangedListener listener) {
if (mListeners == null) {
return false;
}

synchronized (mLock) {
final int keyIndex = mListeners.indexOfKey(listener);
final boolean hasKey = keyIndex >= 0;
if (hasKey) {
mListeners.removeAt(keyIndex);
}

if (hasKey && mListeners.isEmpty()) {
// We just removed the last listener, so we don't need callbacks from the
// service anymore.
setSoftKeyboardCallbackEnabled(false);
}

return hasKey;
}
}

private void setSoftKeyboardCallbackEnabled(boolean enabled) {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
connection.setSoftKeyboardCallbackEnabled(enabled);
} catch (RemoteException re) {
throw new RuntimeException(re);
}
}
```

```
}

/**
* Dispatches the soft keyboard show mode change to any registered listeners. This should
* be called on the service's main thread.
*/
void dispatchSoftKeyboardShowModeChanged(final int showMode) {
final ArrayMap<OnShowModeChangedListener, Handler> entries;
synchronized (mLock) {
if (mListeners == null || mListeners.isEmpty()) {
Slog.w(LOG_TAG, "Received soft keyboard show mode changed callback"
+ " with no listeners registered!");
setSoftKeyboardCallbackEnabled(false);
return;
}

// Listeners may remove themselves. Perform a shallow copy to avoid concurrent
// modification.
entries = new ArrayMap<>(mListeners);
}

for (int i = 0, count = entries.size(); i < count; i++) {
final OnShowModeChangedListener listener = entries.keyAt(i);
final Handler handler = entries.valueAt(i);
if (handler != null) {
handler.post(new Runnable() {
@Override
public void run() {
listener.onShowModeChanged(SoftKeyboardController.this, showMode);
}
});
} else {
// We're already on the main thread, just run the listener.
listener.onShowModeChanged(this, showMode);
}
}
}

/**
* Returns the show mode of the soft keyboard.
*
* @return the current soft keyboard show mode
*
* @see AccessibilityService#SHOW_MODE_AUTO
* @see AccessibilityService#SHOW_MODE_HIDDEN
* @see AccessibilityService#SHOW_MODE_IGNORE_HARD_KEYBOARD
*/
@SoftKeyboardShowMode
public int getShowMode() {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
```

```
return connection.getSoftKeyboardShowMode();
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to set soft keyboard behavior", re);
re.rethrowFromSystemServer();
}
}
return SHOW_MODE_AUTO;
}

/**
* Sets the soft keyboard show mode.
* <p>
* <strong>Note:</strong> If the service is not yet connected (e.g.
* {@link AccessibilityService#onServiceConnected()} has not yet been called) or the
* service has been disconnected, this method will have no effect and return {@code false}.
*
* @param showMode the new show mode for the soft keyboard
* @return {@code true} on success
*
* @see AccessibilityService#SHOW_MODE_AUTO
* @see AccessibilityService#SHOW_MODE_HIDDEN
* @see AccessibilityService#SHOW_MODE_IGNORE_HARD_KEYBOARD
*/
public boolean setShowMode(@SoftKeyboardShowMode int showMode) {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
return connection.setSoftKeyboardShowMode(showMode);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to set soft keyboard behavior", re);
re.rethrowFromSystemServer();
}
}
return false;
}

/**
* Listener for changes in the soft keyboard show mode.
*/
public interface OnShowModeChangedListener {
/**
* Called when the soft keyboard behavior changes. The default show mode is
* {@code SHOW_MODE_AUTO}, where the soft keyboard is shown when a text input field is
* focused. An AccessibilityService can also request the show mode
* {@code SHOW_MODE_HIDDEN}, where the soft keyboard is never shown.
*
* @param controller the soft keyboard controller
* @param showMode the current soft keyboard show mode
*/
void onShowModeChanged(@NonNull SoftKeyboardController controller,
@SoftKeyboardShowMode int showMode);
}
```

```
/**
* Switches the current IME for the user for whom the service is enabled. The change will
* persist until the current IME is explicitly changed again, and may persist beyond the
* life cycle of the requesting service.
*
* @param imeId The ID of the input method to make current. This IME must be installed and
* enabled.
* @return {@code true} if the current input method was successfully switched to the input
* method by {@code imeId},
* {@code false} if the input method specified is not installed, not enabled, or
* otherwise not available to become the current IME
*
* @see android.view.inputmethod.InputMethodInfo#getId()
*/
public boolean switchToInputMethod(@NonNull String imeId) {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
return connection.switchToInputMethod(imeId);
} catch (RemoteException re) {
throw new RuntimeException(re);
}
}
return false;
}

/**
* Enable or disable the specified IME for the user for whom the service is activated. The
* IME needs to be in the same package as the service and needs to be allowed by device
* policy, if there is one. The change will persist until the specified IME is next
* explicitly enabled or disabled by whatever means, such as user choice, and may persist
* beyond the life cycle of the requesting service.
*
* @param imeId The ID of the input method to enable or disable. This IME must be installed.
* @param enabled {@code true} if the input method associated with {@code imeId} should be
* enabled.
* @return status code for the result of enabling/disabling the input method associated
* with {@code imeId}.
* @throws SecurityException if the input method is not in the same package as the service.
*
* @see android.view.inputmethod.InputMethodInfo#getId()
*/
@CheckResult
@EnableImeResult
public int setInputMethodEnabled(@NonNull String imeId, boolean enabled)
throws SecurityException {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(mService).getConnection(
mService.mConnectionId);
if (connection != null) {
try {
```

```java
return connection.setInputMethodEnabled(imeId, enabled);
} catch (RemoteException re) {
throw new RuntimeException(re);
}
}
return ENABLE_IME_FAIL_UNKNOWN;
}
}

/**
* Returns the controller for the accessibility button within the system's navigation area.
* This instance may be used to query the accessibility button's state and register listeners
* for interactions with and state changes for the accessibility button when
* {@link AccessibilityServiceInfo#FLAG_REQUEST_ACCESSIBILITY_BUTTON} is set.
* <p>
* <strong>Note:</strong> Not all devices are capable of displaying the accessibility button
* within a navigation area, and as such, use of this class should be considered only as an
* optional feature or shortcut on supported device implementations.
* </p>
*
* @return the accessibility button controller for this {@link AccessibilityService}
*/
@NonNull
public final AccessibilityButtonController getAccessibilityButtonController() {
return getAccessibilityButtonController(Display.DEFAULT_DISPLAY);
}

/**
* Returns the controller of specified logical display for the accessibility button within the
* system's navigation area. This instance may be used to query the accessibility button's
* state and register listeners for interactions with and state changes for the accessibility
* button when {@link AccessibilityServiceInfo#FLAG_REQUEST_ACCESSIBILITY_BUTTON} is set.
* <p>
* <strong>Note:</strong> Not all devices are capable of displaying the accessibility button
* within a navigation area, and as such, use of this class should be considered only as an
* optional feature or shortcut on supported device implementations.
* </p>
*
* @param displayId The logic display id, use {@link Display#DEFAULT_DISPLAY} for default
* display.
* @return the accessibility button controller for this {@link AccessibilityService}
*/
@NonNull
public final AccessibilityButtonController getAccessibilityButtonController(int displayId) {
synchronized (mLock) {
AccessibilityButtonController controller = mAccessibilityButtonControllers.get(
displayId);
if (controller == null) {
controller = new AccessibilityButtonController(
AccessibilityInteractionClient.getInstance(this).getConnection(mConnectionId));
mAccessibilityButtonControllers.put(displayId, controller);
}
return controller;
}
```

```java
}

private void onAccessibilityButtonClicked(int displayId) {
getAccessibilityButtonController(displayId).dispatchAccessibilityButtonClicked();
}

private void onAccessibilityButtonAvailabilityChanged(boolean available) {
getAccessibilityButtonController().dispatchAccessibilityButtonAvailabilityChanged(
available);
}

/** Sets the cache status.
*
* <p>If {@code enabled}, enable the cache and prefetching. Otherwise, disable the cache
* and prefetching.
* Note: By default the cache is enabled.
* @param enabled whether to enable or disable the cache.
* @return {@code true} if the cache and connection are not null, so the cache status is set.
*/
public boolean setCacheEnabled(boolean enabled) {
AccessibilityCache cache =
AccessibilityInteractionClient.getCache(mConnectionId);
if (cache == null) {
return false;
}
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getConnection(mConnectionId);
if (connection == null) {
return false;
}
try {
connection.setCacheEnabled(enabled);
cache.setEnabled(enabled);
return true;
} catch (RemoteException re) {
Log.w(LOG_TAG, "Error while setting status of cache", re);
re.rethrowFromSystemServer();
}
return false;
}

/** Invalidates {@code node} and its subtree in the cache.
* @param node the node to invalidate.
* @return {@code true} if the subtree rooted at {@code node} was invalidated.
*/
public boolean clearCachedSubtree(@NonNull AccessibilityNodeInfo node) {
AccessibilityCache cache =
AccessibilityInteractionClient.getCache(mConnectionId);
if (cache == null) {
return false;
}
return cache.clearSubTree(node);
}
```

```java
/** Clears the cache.
* @return {@code true} if the cache was cleared
*/
public boolean clearCache() {
AccessibilityCache cache =
AccessibilityInteractionClient.getCache(mConnectionId);
if (cache == null) {
return false;
}
cache.clear();
return true;
}

/** Checks if {@code node} is in the cache.
* @param node the node to check.
* @return {@code true} if {@code node} is in the cache.
*/
public boolean isNodeInCache(@NonNull AccessibilityNodeInfo node) {
AccessibilityCache cache =
AccessibilityInteractionClient.getCache(mConnectionId);
if (cache == null) {
return false;
}
return cache.isNodeInCache(node);
}

/** Returns {@code true} if the cache is enabled. */
public boolean isCacheEnabled() {
AccessibilityCache cache =
AccessibilityInteractionClient.getCache(mConnectionId);
if (cache == null) {
return false;
}
return cache.isEnabled();
}

/** This is called when the system action list is changed. */
public void onSystemActionsChanged() {
}

/**
* Returns a list of system actions available in the system right now.
* <p>
* System actions that correspond to the global action constants will have matching action IDs.
* For example, an with id {@link #GLOBAL_ACTION_BACK} will perform the back action.
* </p>
* <p>
* These actions should be called by {@link #performGlobalAction}.
* </p>
*
* @return A list of available system actions.
*/
```

```
public final @NonNull List<AccessibilityAction> getSystemActions() {
IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(this).getConnection(mConnectionId);
if (connection != null) {
try {
return connection.getSystemActions();
} catch (RemoteException re) {
Log.w(LOG_TAG, "Error while calling getSystemActions", re);
re.rethrowFromSystemServer();
}
}
return Collections.emptyList();
}

/**
* Performs a global action. Such an action can be performed
* at any moment regardless of the current application or user
* location in that application. For example going back, going
* home, opening recents, etc.
*
* <p>
* Note: The global action ids themselves give no information about the current availability
* of their corresponding actions. To determine if a global action is available, use
* {@link #getSystemActions()}
*
* @param action The action to perform.
* @return Whether the action was successfully performed.
*
* Perform actions using ids like the id constants referenced below:
* @see #GLOBAL_ACTION_BACK
* @see #GLOBAL_ACTION_HOME
* @see #GLOBAL_ACTION_NOTIFICATIONS
* @see #GLOBAL_ACTION_RECENTS
* @see #GLOBAL_ACTION_DPAD_UP
* @see #GLOBAL_ACTION_DPAD_DOWN
* @see #GLOBAL_ACTION_DPAD_LEFT
* @see #GLOBAL_ACTION_DPAD_RIGHT
* @see #GLOBAL_ACTION_DPAD_CENTER
*/
public final boolean performGlobalAction(int action) {
IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(this).getConnection(mConnectionId);
if (connection != null) {
try {
return connection.performGlobalAction(action);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Error while calling performGlobalAction", re);
re.rethrowFromSystemServer();
}
}
return false;
}

/**
```

```
* Find the view that has the specified focus type. The search is performed
* across all windows.
* <p>
* <strong>Note:</strong> In order to access the windows your service has
* to declare the capability to retrieve window content by setting the
* {@link android.R.styleable#AccessibilityService_canRetrieveWindowContent}
* property in its meta-data. For details refer to {@link #SERVICE_META_DATA}.
* Also the service has to opt-in to retrieve the interactive windows by
* setting the {@link AccessibilityServiceInfo#FLAG_RETRIEVE_INTERACTIVE_WINDOWS}
* flag. Otherwise, the search will be performed only in the active window.
* </p>
* <p>
* <strong>Note:</strong> If the view with {@link AccessibilityNodeInfo#FOCUS_INPUT}
* is on an embedded view hierarchy which is embedded in a {@link android.view.SurfaceView} via
* {@link android.view.SurfaceView#setChildSurfacePackage}, there is a limitation that this API
* won't be able to find the node for the view. It's because views don't know about
* the embedded hierarchies. Instead, you could traverse all the nodes to find the
* focus.
* </p>
*
* @param focus The focus to find. One of {@link AccessibilityNodeInfo#FOCUS_INPUT} or
* {@link AccessibilityNodeInfo#FOCUS_ACCESSIBILITY}.
* @return The node info of the focused view or null.
*
* @see AccessibilityNodeInfo#FOCUS_INPUT
* @see AccessibilityNodeInfo#FOCUS_ACCESSIBILITY
*/
public AccessibilityNodeInfo findFocus(int focus) {
return AccessibilityInteractionClient.getInstance(this).findFocus(mConnectionId,
AccessibilityWindowInfo.ANY_WINDOW_ID, AccessibilityNodeInfo.ROOT_NODE_ID, focus);
}

/**
* Gets the an {@link AccessibilityServiceInfo} describing this
* {@link AccessibilityService}. This method is useful if one wants
* to change some of the dynamically configurable properties at
* runtime.
*
* @return The accessibility service info.
*
* @see AccessibilityServiceInfo
*/
public final AccessibilityServiceInfo getServiceInfo() {
IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(this).getConnection(mConnectionId);
if (connection != null) {
try {
return connection.getServiceInfo();
} catch (RemoteException re) {
Log.w(LOG_TAG, "Error while getting AccessibilityServiceInfo", re);
re.rethrowFromSystemServer();
}
}
return null;
```

```
}

/**
* Sets the {@link AccessibilityServiceInfo} that describes this service.
* <p>
* Note: You can call this method any time but the info will be picked up after
* the system has bound to this service and when this method is called thereafter.
*
* @param info The info.
*/
public final void setServiceInfo(AccessibilityServiceInfo info) {
mInfo = info;
updateInputMethod(info);
mMotionEventSources = info.getMotionEventSources();
sendServiceInfo();
}

/**
* Sets the {@link AccessibilityServiceInfo} for this service if the latter is
* properly set and there is an {@link IAccessibilityServiceConnection} to the
* AccessibilityManagerService.
*/
private void sendServiceInfo() {
IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(this).getConnection(mConnectionId);
if (mInfo != null && connection != null) {
if (!mInfo.isWithinParcelableSize()) {
throw new IllegalStateException(
"Cannot update service info: size is larger than safe parcelable limits.");
}
try {
connection.setServiceInfo(mInfo);
mInfo = null;
AccessibilityInteractionClient.getInstance(this).clearCache(mConnectionId);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Error while setting AccessibilityServiceInfo", re);
re.rethrowFromSystemServer();
}
}
}

@Override
public Object getSystemService(@ServiceName @NonNull String name) {
if (getBaseContext() == null) {
throw new IllegalStateException(
"System services not available to Activities before onCreate()");
}

// Guarantee that we always return the same window manager instance.
if (WINDOW_SERVICE.equals(name)) {
if (mWindowManager == null) {
mWindowManager = (WindowManager) getBaseContext().getSystemService(name);
final WindowManagerImpl wm = (WindowManagerImpl) mWindowManager;
// Set e default token obtained from the connection to ensure client could use
// accessibility overlay.
wm.setDefaultToken(mWindowToken);
}
return mWindowManager;
}
return super.getSystemService(name);
}

/**
* Takes a screenshot of the specified display and returns it via an
* {@link AccessibilityService.ScreenshotResult}. You can use {@link Bitmap#wrapHardwareBuffer}
* to construct the bitmap from the ScreenshotResult's payload.
* <p>
* <strong>Note:</strong> In order to take screenshot your service has
* to declare the capability to take screenshot by setting the
* {@link android.R.styleable#AccessibilityService_canTakeScreenshot}
* property in its meta-data. For details refer to {@link #SERVICE_META_DATA}.
* </p>
*
* @param displayId The logic display id, must be {@link Display#DEFAULT_DISPLAY} for
* default display.
* @param executor Executor on which to run the callback.
* @param callback The callback invoked when taking screenshot has succeeded or failed.
* See {@link TakeScreenshotCallback} for details.
* @see #takeScreenshotOfWindow
*/
public void takeScreenshot(int displayId, @NonNull @CallbackExecutor Executor executor,
@NonNull TakeScreenshotCallback callback) {
Preconditions.checkNotNull(executor, "executor cannot be null");
Preconditions.checkNotNull(callback, "callback cannot be null");
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(this).getConnection(mConnectionId);
if (connection == null) {
sendScreenshotFailure(ERROR_TAKE_SCREENSHOT_INTERNAL_ERROR, executor, callback);
return;
}
try {
connection.takeScreenshot(displayId, new RemoteCallback((result) -> {
final int status = result.getInt(KEY_ACCESSIBILITY_SCREENSHOT_STATUS);
if (status != TAKE_SCREENSHOT_SUCCESS) {
sendScreenshotFailure(status, executor, callback);
return;
}
final HardwareBuffer hardwareBuffer =
result.getParcelable(KEY_ACCESSIBILITY_SCREENSHOT_HARDWAREBUFFER,
android.hardware.HardwareBuffer.class);
final ParcelableColorSpace colorSpace =
result.getParcelable(KEY_ACCESSIBILITY_SCREENSHOT_COLORSPACE,
android.graphics.ParcelableColorSpace.class);
final ScreenshotResult screenshot = new ScreenshotResult(hardwareBuffer,
colorSpace.getColorSpace(),
result.getLong(KEY_ACCESSIBILITY_SCREENSHOT_TIMESTAMP));
sendScreenshotSuccess(screenshot, executor, callback);
}));
```

```
} catch (RemoteException re) {
throw new RuntimeException(re);
}
}

/**
* Takes a screenshot of the specified window and returns it via an
* {@link AccessibilityService.ScreenshotResult}. You can use {@link Bitmap#wrapHardwareBuffer}
* to construct the bitmap from the ScreenshotResult's payload.
* <p>
* <strong>Note:</strong> In order to take screenshots your service has
* to declare the capability to take screenshot by setting the
* {@link android.R.styleable#AccessibilityService_canTakeScreenshot}
* property in its meta-data. For details refer to {@link #SERVICE_META_DATA}.
* </p>
* <p>
* Both this method and {@link #takeScreenshot} can be used for machine learning-based visual
* screen understanding. Use <code>takeScreenshotOfWindow</code> if your target window might be
* visually underneath an accessibility overlay (from your or another accessibility service) in
* order to capture the window contents without the screenshot being covered by the overlay
* contents drawn on the screen.
* </p>
*
* @param accessibilityWindowId The window id, from {@link AccessibilityWindowInfo#getId()}.
* @param executor Executor on which to run the callback.
* @param callback The callback invoked when taking screenshot has succeeded or failed.
* See {@link TakeScreenshotCallback} for details.
* @see #takeScreenshot
*/
public void takeScreenshotOfWindow(int accessibilityWindowId,
@NonNull @CallbackExecutor Executor executor,
@NonNull TakeScreenshotCallback callback) {
AccessibilityInteractionClient.getInstance(this).takeScreenshotOfWindow(
mConnectionId, accessibilityWindowId, executor, callback);
}

/**
* Sets the strokeWidth and color of the accessibility focus rectangle.
* <p>
* <strong>Note:</strong> This setting persists until this or another active
* AccessibilityService changes it or the device reboots.
* </p>
*
* @param strokeWidth The stroke width of the rectangle in pixels.
* Setting this value to zero results in no focus rectangle being drawn.
* @param color The color of the rectangle.
*/
public void setAccessibilityFocusAppearance(int strokeWidth, @ColorInt int color) {
IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(this).getConnection(mConnectionId);
if (connection != null) {
try {
connection.setFocusAppearance(strokeWidth, color);
} catch (RemoteException re) {
```

```
Log.w(LOG_TAG, "Error while setting the strokeWidth and color of the "
+ "accessibility focus rectangle", re);
re.rethrowFromSystemServer();
}
}
}

/**
* Implement to return the implementation of the internal accessibility
* service interface.
*/
@Override
public final IBinder onBind(Intent intent) {
return new IAccessibilityServiceClientWrapper(this, getMainExecutor(), new Callbacks() {
@Override
public void onServiceConnected() {
AccessibilityService.this.dispatchServiceConnected();
}

@Override
public void onInterrupt() {
AccessibilityService.this.onInterrupt();
}

@Override
public void onAccessibilityEvent(AccessibilityEvent event) {
AccessibilityService.this.onAccessibilityEvent(event);
}

@Override
public void init(int connectionId, IBinder windowToken) {
mConnectionId = connectionId;
mWindowToken = windowToken;

// The client may have already obtained the window manager, so
// update the default token on whatever manager we gave them.
if (mWindowManager != null) {
final WindowManagerImpl wm = (WindowManagerImpl) mWindowManager;
wm.setDefaultToken(mWindowToken);
}
}

@Override
public boolean onGesture(AccessibilityGestureEvent gestureEvent) {
return AccessibilityService.this.onGesture(gestureEvent);
}

@Override
public boolean onKeyEvent(KeyEvent event) {
return AccessibilityService.this.onKeyEvent(event);
}

@Override
```

```
public void onMagnificationChanged(int displayId, @NonNull Region region,
MagnificationConfig config) {
AccessibilityService.this.onMagnificationChanged(displayId, region, config);
}

@Override
public void onMotionEvent(MotionEvent event) {
AccessibilityService.this.sendMotionEventToCallback(event);
}

@Override
public void onTouchStateChanged(int displayId, int state) {
AccessibilityService.this.onTouchStateChanged(displayId, state);
}

@Override
public void onSoftKeyboardShowModeChanged(int showMode) {
AccessibilityService.this.onSoftKeyboardShowModeChanged(showMode);
}

@Override
public void onPerformGestureResult(int sequence, boolean completedSuccessfully) {
AccessibilityService.this.onPerformGestureResult(sequence, completedSuccessfully);
}

@Override
public void onFingerprintCapturingGesturesChanged(boolean active) {
AccessibilityService.this.onFingerprintCapturingGesturesChanged(active);
}

@Override
public void onFingerprintGesture(int gesture) {
AccessibilityService.this.onFingerprintGesture(gesture);
}

@Override
public void onAccessibilityButtonClicked(int displayId) {
AccessibilityService.this.onAccessibilityButtonClicked(displayId);
}

@Override
public void onAccessibilityButtonAvailabilityChanged(boolean available) {
AccessibilityService.this.onAccessibilityButtonAvailabilityChanged(available);
}

@Override
public void onSystemActionsChanged() {
AccessibilityService.this.onSystemActionsChanged();
}

@Override
public void createImeSession(IAccessibilityInputMethodSessionCallback callback) {
```

```
if (mInputMethod != null) {
mInputMethod.createImeSession(callback);
}
}

@Override
public void startInput(@Nullable RemoteAccessibilityInputConnection connection,
@NonNull EditorInfo editorInfo, boolean restarting) {
if (mInputMethod != null) {
if (restarting) {
mInputMethod.restartInput(connection, editorInfo);
} else {
mInputMethod.startInput(connection, editorInfo);
}
}
}
});
}

/**
* Implements the internal {@link IAccessibilityServiceClient} interface to convert
* incoming calls to it back to calls on an {@link AccessibilityService}.
*
* @hide
*/
public static class IAccessibilityServiceClientWrapper extends
IAccessibilityServiceClient.Stub {

private final Callbacks mCallback;
private final Context mContext;
private final Executor mExecutor;

private int mConnectionId = AccessibilityInteractionClient.NO_ID;

/**
* This is not {@code null} only between {@link #bindInput()} and {@link #unbindInput()} so
* that {@link RemoteAccessibilityInputConnection} can query if {@link #unbindInput()} has
* already been called or not, mainly to avoid unnecessary blocking operations.
*
* <p>This field must be set and cleared only from the binder thread(s), where the system
* guarantees that {@link #bindInput()},
* {@link #startInput(IRemoteAccessibilityInputConnection, EditorInfo, boolean)},
* and {@link #unbindInput()} are called with the same order as the original calls
* in {@link com.android.server.inputmethod.InputMethodManagerService}.
* See {@link IBinder#FLAG_ONEWAY} for detailed semantics.</p>
*/
@Nullable
CancellationGroup mCancellationGroup = null;

public IAccessibilityServiceClientWrapper(Context context, Executor executor,
Callbacks callback) {
mCallback = callback;
mContext = context;
```

```
mExecutor = executor;
}

public IAccessibilityServiceClientWrapper(Context context, Looper looper,
Callbacks callback) {
this(context, new HandlerExecutor(new Handler(looper)), callback);
}

public void init(IAccessibilityServiceConnection connection, int connectionId,
IBinder windowToken) {
mExecutor.execute(() -> {
mConnectionId = connectionId;
if (connection != null) {
AccessibilityInteractionClient.getInstance(mContext).addConnection(
mConnectionId, connection, /*initializeCache=*/true);
if (mContext != null) {
try {
connection.setAttributionTag(mContext.getAttributionTag());
} catch (RemoteException re) {
Log.w(LOG_TAG, "Error while setting attributionTag", re);
re.rethrowFromSystemServer();
}
}
mCallback.init(mConnectionId, windowToken);
mCallback.onServiceConnected();
} else {
AccessibilityInteractionClient.getInstance(mContext)
.clearCache(mConnectionId);
AccessibilityInteractionClient.getInstance(mContext).removeConnection(
mConnectionId);
mConnectionId = AccessibilityInteractionClient.NO_ID;
mCallback.init(AccessibilityInteractionClient.NO_ID, null);
}
return;
});
}

public void onInterrupt() {
mExecutor.execute(() -> {
if (mConnectionId != AccessibilityInteractionClient.NO_ID) {
mCallback.onInterrupt();
}
});
}

public void onAccessibilityEvent(AccessibilityEvent event, boolean serviceWantsEvent) {
mExecutor.execute(() -> {
if (event != null) {
// Send the event to AccessibilityCache via AccessibilityInteractionClient
AccessibilityInteractionClient.getInstance(mContext).onAccessibilityEvent(
event, mConnectionId);
if (serviceWantsEvent
&& (mConnectionId != AccessibilityInteractionClient.NO_ID)) {
// Send the event to AccessibilityService
mCallback.onAccessibilityEvent(event);
}
}
return;
});
}

@Override
public void onGesture(AccessibilityGestureEvent gestureInfo) {
mExecutor.execute(() -> {
if (mConnectionId != AccessibilityInteractionClient.NO_ID) {
mCallback.onGesture(gestureInfo);
}
return;
});
}

public void clearAccessibilityCache() {
mExecutor.execute(() -> {
AccessibilityInteractionClient.getInstance(mContext).clearCache(mConnectionId);
return;
});
}

@Override
public void onKeyEvent(KeyEvent event, int sequence) {
mExecutor.execute(() -> {
try {
IAccessibilityServiceConnection connection = AccessibilityInteractionClient
.getInstance(mContext).getConnection(mConnectionId);
if (connection != null) {
final boolean result = mCallback.onKeyEvent(event);
try {
connection.setOnKeyEventResult(result, sequence);
} catch (RemoteException re) {
/* ignore */
}
}
} finally {
// Make sure the event is recycled.
try {
event.recycle();
} catch (IllegalStateException ise) {
/* ignore - best effort */
}
}
return;
});
}

/** Magnification changed callbacks for different displays */
public void onMagnificationChanged(int displayId, @NonNull Region region,
MagnificationConfig config) {
mExecutor.execute(() -> {
```

```
if (mConnectionId != AccessibilityInteractionClient.NO_ID) {
mCallback.onMagnificationChanged(displayId, region, config);
}
return;
});
}

public void onSoftKeyboardShowModeChanged(int showMode) {
mExecutor.execute(() -> {
if (mConnectionId != AccessibilityInteractionClient.NO_ID) {
mCallback.onSoftKeyboardShowModeChanged(showMode);
}
return;
});
}

public void onPerformGestureResult(int sequence, boolean successfully) {
mExecutor.execute(() -> {
if (mConnectionId != AccessibilityInteractionClient.NO_ID) {
mCallback.onPerformGestureResult(sequence, successfully);
}
return;
});
}

public void onFingerprintCapturingGesturesChanged(boolean active) {
mExecutor.execute(() -> {
if (mConnectionId != AccessibilityInteractionClient.NO_ID) {
mCallback.onFingerprintCapturingGesturesChanged(active);
}
return;
});
}

public void onFingerprintGesture(int gesture) {
mExecutor.execute(() -> {
if (mConnectionId != AccessibilityInteractionClient.NO_ID) {
mCallback.onFingerprintGesture(gesture);
}
return;
});
}

/** Accessibility button clicked callbacks for different displays */
public void onAccessibilityButtonClicked(int displayId) {
mExecutor.execute(() -> {
if (mConnectionId != AccessibilityInteractionClient.NO_ID) {
mCallback.onAccessibilityButtonClicked(displayId);
}
return;
});
}
```

```
public void onAccessibilityButtonAvailabilityChanged(boolean available) {
mExecutor.execute(() -> {
if (mConnectionId != AccessibilityInteractionClient.NO_ID) {
mCallback.onAccessibilityButtonAvailabilityChanged(available);
}
return;
});
}

/** This is called when the system action list is changed. */
public void onSystemActionsChanged() {
mExecutor.execute(() -> {
if (mConnectionId != AccessibilityInteractionClient.NO_ID) {
mCallback.onSystemActionsChanged();
}
return;
});
}

/** This is called when an app requests ime sessions or when the service is enabled. */
public void createImeSession(IAccessibilityInputMethodSessionCallback callback) {
mExecutor.execute(() -> {
if (mConnectionId != AccessibilityInteractionClient.NO_ID) {
mCallback.createImeSession(callback);
}
});
}

/**
* This is called when InputMethodManagerService requests to set the session enabled or
* disabled
*/
public void setImeSessionEnabled(IAccessibilityInputMethodSession session,
boolean enabled) {
try {
AccessibilityInputMethodSession ls =
((AccessibilityInputMethodSessionWrapper) session).getSession();
if (ls == null) {
Log.w(LOG_TAG, "Session is already finished: " + session);
return;
}
mExecutor.execute(() -> {
if (mConnectionId != AccessibilityInteractionClient.NO_ID) {
ls.setEnabled(enabled);
}
return;
});
} catch (ClassCastException e) {
Log.w(LOG_TAG, "Incoming session not of correct type: " + session, e);
}
}

/** This is called when an app binds input or when the service is enabled. */
public void bindInput() {
```

```
if (mCancellationGroup != null) {
Log.e(LOG_TAG, "bindInput must be paired with unbindInput.");
}
mCancellationGroup = new CancellationGroup();
}

/** This is called when an app unbinds input or when the service is disabled. */
public void unbindInput() {
if (mCancellationGroup != null) {
// Signal the flag then forget it.
mCancellationGroup.cancelAll();
mCancellationGroup = null;
} else {
Log.e(LOG_TAG, "unbindInput must be paired with bindInput.");
}
}

/** This is called when an app starts input or when the service is enabled. */
public void startInput(IRemoteAccessibilityInputConnection connection,
EditorInfo editorInfo, boolean restarting) {
if (mCancellationGroup == null) {
Log.e(LOG_TAG, "startInput must be called after bindInput.");
mCancellationGroup = new CancellationGroup();
}
mExecutor.execute(() -> {
if (mConnectionId != AccessibilityInteractionClient.NO_ID) {
final RemoteAccessibilityInputConnection ic = connection == null ? null
: new RemoteAccessibilityInputConnection(
connection, mCancellationGroup);
editorInfo.makeCompatible(mContext.getApplicationInfo().targetSdkVersion);
mCallback.startInput(ic, editorInfo, restarting);
}
});
}

@Override
public void onMotionEvent(MotionEvent event) {
mExecutor.execute(() -> {
mCallback.onMotionEvent(event);
});
}

@Override
public void onTouchStateChanged(int displayId, int state) {
mExecutor.execute(() -> {
mCallback.onTouchStateChanged(displayId, state);
});
}
}

/**
* Class used to report status of dispatched gestures
*/
```

```
public static abstract class GestureResultCallback {
/** Called when the gesture has completed successfully
*
* @param gestureDescription The description of the gesture that completed.
*/
public void onCompleted(GestureDescription gestureDescription) {
}

/** Called when the gesture was cancelled
*
* @param gestureDescription The description of the gesture that was cancelled.
*/
public void onCancelled(GestureDescription gestureDescription) {
}
}

/* Object to keep track of gesture result callbacks */
private static class GestureResultCallbackInfo {
GestureDescription gestureDescription;
GestureResultCallback callback;
Handler handler;

GestureResultCallbackInfo(GestureDescription gestureDescription,
GestureResultCallback callback, Handler handler) {
this.gestureDescription = gestureDescription;
this.callback = callback;
this.handler = handler;
}
}

private void sendScreenshotSuccess(ScreenshotResult screenshot, Executor executor,
TakeScreenshotCallback callback) {
executor.execute(() -> callback.onSuccess(screenshot));
}

private void sendScreenshotFailure(@ScreenshotErrorCode int errorCode, Executor executor,
TakeScreenshotCallback callback) {
executor.execute(() -> callback.onFailure(errorCode));
}

/**
* Interface used to report status of taking screenshot.
*/
public interface TakeScreenshotCallback {
/** Called when taking screenshot has completed successfully.
*
* @param screenshot The content of screenshot.
*/
void onSuccess(@NonNull ScreenshotResult screenshot);

/** Called when taking screenshot has failed. {@code errorCode} will identify the
* reason of failure.
*
```

```
* @param errorCode The error code of this operation.
*/
void onFailure(@ScreenshotErrorCode int errorCode);
}

/**
* Can be used to construct a bitmap of the screenshot or any other operations for
* {@link AccessibilityService#takeScreenshot} API.
*/
public static final class ScreenshotResult {
private final @NonNull HardwareBuffer mHardwareBuffer;
private final @NonNull ColorSpace mColorSpace;
private final long mTimestamp;

/** @hide */
public ScreenshotResult(@NonNull HardwareBuffer hardwareBuffer,
@NonNull ColorSpace colorSpace, long timestamp) {
Preconditions.checkNotNull(hardwareBuffer, "hardwareBuffer cannot be null");
Preconditions.checkNotNull(colorSpace, "colorSpace cannot be null");
mHardwareBuffer = hardwareBuffer;
mColorSpace = colorSpace;
mTimestamp = timestamp;
}

/**
* Gets the {@link ColorSpace} identifying a specific organization of colors of the
* screenshot.
*
* @return the color space
*/
@NonNull
public ColorSpace getColorSpace() {
return mColorSpace;
}

/**
* Gets the {@link HardwareBuffer} representing a memory buffer of the screenshot.
* <p>
* <strong>Note:</strong> The application should call {@link HardwareBuffer#close()} when
* the buffer is no longer needed to free the underlying resources.
* </p>
*
* @return the hardware buffer
*/
@NonNull
public HardwareBuffer getHardwareBuffer() {
return mHardwareBuffer;
}

/**
* Gets the timestamp of taking the screenshot.
*
* @return milliseconds of non-sleep uptime before screenshot since boot and it's from
* {@link SystemClock#uptimeMillis()}
*/
public long getTimestamp() {
return mTimestamp;
};
}

/**
* When {@link AccessibilityServiceInfo#FLAG_REQUEST_TOUCH_EXPLORATION_MODE} is enabled, this
* function requests that touch interactions starting in the specified region of the screen
* bypass the gesture detector. There can only be one gesture detection passthrough region per
* display. Requesting a new gesture detection passthrough region clears the existing one. To
* disable this passthrough and return to the original behavior, pass in an empty region. When
* {@link AccessibilityServiceInfo#FLAG_REQUEST_TOUCH_EXPLORATION_MODE} is disabled this
* function has no effect.
*
* @param displayId The display on which to set this region.
* @param region the region of the screen.
*/
public void setGestureDetectionPassthroughRegion(int displayId, @NonNull Region region) {
Preconditions.checkNotNull(region, "region cannot be null");
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(this).getConnection(mConnectionId);
if (connection != null) {
try {
connection.setGestureDetectionPassthroughRegion(displayId, region);
} catch (RemoteException re) {
throw new RuntimeException(re);
}
}
}

/**
* When {@link AccessibilityServiceInfo#FLAG_REQUEST_TOUCH_EXPLORATION_MODE} is enabled, this
* function requests that touch interactions starting in the specified region of the screen
* bypass the touch explorer and go straight to the view hierarchy. There can only be one touch
* exploration passthrough region per display. Requesting a new touch explorationpassthrough
* region clears the existing one. To disable this passthrough and return to the original
* behavior, pass in an empty region. When {@link
* AccessibilityServiceInfo#FLAG_REQUEST_TOUCH_EXPLORATION_MODE} is disabled this function has
* no effect.
*
* @param displayId The display on which to set this region.
* @param region the region of the screen .
*/
public void setTouchExplorationPassthroughRegion(int displayId, @NonNull Region region) {
Preconditions.checkNotNull(region, "region cannot be null");
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(this).getConnection(mConnectionId);
if (connection != null) {
try {
connection.setTouchExplorationPassthroughRegion(displayId, region);
} catch (RemoteException re) {
throw new RuntimeException(re);
}
}
}

/**
* Sets the system settings values that control the scaling factor for animations. The scale
* controls the animation playback speed for animations that respect these settings. Animations
* that do not respect the settings values will not be affected by this function. A lower scale
* value results in a faster speed. A value of <code>0</code> disables animations entirely. When
* animations are disabled services receive window change events more quickly which can reduce
* the potential by confusion by reducing the time during which windows are in transition.
*
* @see AccessibilityEvent#TYPE_WINDOWS_CHANGED
* @see AccessibilityEvent#TYPE_WINDOW_STATE_CHANGED
* @see android.provider.Settings.Global#WINDOW_ANIMATION_SCALE
* @see android.provider.Settings.Global#TRANSITION_ANIMATION_SCALE
* @see android.provider.Settings.Global#ANIMATOR_DURATION_SCALE
* @param scale The scaling factor for all animations.
*/
public void setAnimationScale(float scale) {
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getInstance(this).getConnection(mConnectionId);
if (connection != null) {
try {
connection.setAnimationScale(scale);
} catch (RemoteException re) {
throw new RuntimeException(re);
}
}
}

private static class AccessibilityContext extends ContextWrapper {
private final int mConnectionId;

private AccessibilityContext(Context base, int connectionId) {
super(base);
mConnectionId = connectionId;
setDefaultTokenInternal(this, getDisplayId());
}

@NonNull
@Override
public Context createDisplayContext(Display display) {
return new AccessibilityContext(super.createDisplayContext(display), mConnectionId);
}

@NonNull
@Override
public Context createWindowContext(int type, @Nullable Bundle options) {
final Context context = super.createWindowContext(type, options);
if (type != TYPE_ACCESSIBILITY_OVERLAY) {
return context;
}
```

```
return new AccessibilityContext(context, mConnectionId);
}

@NonNull
@Override
public Context createWindowContext(@NonNull Display display, int type,
@Nullable Bundle options) {
final Context context = super.createWindowContext(display, type, options);
if (type != TYPE_ACCESSIBILITY_OVERLAY) {
return context;
}
return new AccessibilityContext(context, mConnectionId);
}

private void setDefaultTokenInternal(Context context, int displayId) {
final WindowManagerImpl wm = (WindowManagerImpl) context.getSystemService(
WINDOW_SERVICE);
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getConnection(mConnectionId);
IBinder token = null;
if (connection != null) {
try {
token = connection.getOverlayWindowToken(displayId);
} catch (RemoteException re) {
Log.w(LOG_TAG, "Failed to get window token", re);
re.rethrowFromSystemServer();
}
wm.setDefaultToken(token);
}
}
}

/**
* Returns the touch interaction controller for the specified logical display, which may be used
* to detect gestures and otherwise control touch interactions. If
* {@link AccessibilityServiceInfo#FLAG_REQUEST_TOUCH_EXPLORATION_MODE} is disabled the
* controller's methods will have no effect.
*
* @param displayId The logical display id, use {@link Display#DEFAULT_DISPLAY} for default
* display.
* @return the TouchExploration controller
*/
@NonNull
public final TouchInteractionController getTouchInteractionController(int displayId) {
synchronized (mLock) {
TouchInteractionController controller = mTouchInteractionControllers.get(displayId);
if (controller == null) {
controller = new TouchInteractionController(this, mLock, displayId);
mTouchInteractionControllers.put(displayId, controller);
}
return controller;
}
}
```

```
void sendMotionEventToCallback(MotionEvent event) {
boolean sendingTouchEventToTouchInteractionController = false;
if (event.isFromSource(InputDevice.SOURCE_TOUCHSCREEN)) {
TouchInteractionController controller;
synchronized (mLock) {
int displayId = event.getDisplayId();
controller = mTouchInteractionControllers.get(displayId);
}
if (controller != null) {
sendingTouchEventToTouchInteractionController = true;
controller.onMotionEvent(event);
}
}
final int eventSourceWithoutClass = event.getSource() & ~InputDevice.SOURCE_CLASS_MASK;
if ((mMotionEventSources & eventSourceWithoutClass) != 0
&& !sendingTouchEventToTouchInteractionController) {
onMotionEvent(event);
}
}

void onTouchStateChanged(int displayId, int state) {
TouchInteractionController controller;
synchronized (mLock) {
controller = mTouchInteractionControllers.get(displayId);
}
if (controller != null) {
controller.onStateChanged(state);
}
}

/**
* <p>Attaches a {@link android.view.SurfaceControl} containing an accessibility
* overlay to the
* specified display. This type of overlay should be used for content that does
* not need to
* track the location and size of Views in the currently active app e.g. service
* configuration
* or general service UI.</p>
* <p>Generally speaking, an accessibility overlay will be a {@link android.view.View}.
* To embed the View into a {@link android.view.SurfaceControl}, create a
* {@link android.view.SurfaceControlViewHost} and attach the View using
* {@link android.view.SurfaceControlViewHost#setView}. Then obtain the SurfaceControl by
* calling <code> viewHost.getSurfacePackage().getSurfaceControl()</code>.</p>
* <p>To remove this overlay and free the associated
* resources, use
* <code> new SurfaceControl.Transaction().reparent(sc, null).apply();</code>.</p>
* <p>If the specified overlay has already been attached to the specified display
* this method does nothing.
* If the specified overlay has already been attached to a previous display this
* function will transfer the overlay to the new display.
* Services can attach multiple overlays. Use
* <code> new SurfaceControl.Transaction().setLayer(sc, layer).apply();</code>.
* to coordinate the order of the overlays on screen.</p>
*
```

```
* @param displayId the display to which the SurfaceControl should be attached.
* @param sc the SurfaceControl containing the overlay content
*/
public void attachAccessibilityOverlayToDisplay(int displayId, @NonNull SurfaceControl sc) {
Preconditions.checkNotNull(sc, "SurfaceControl cannot be null");
final IAccessibilityServiceConnection connection =
AccessibilityInteractionClient.getConnection(mConnectionId);
if (connection == null) {
return;
}
try {
connection.attachAccessibilityOverlayToDisplay(displayId, sc);
} catch (RemoteException re) {
re.rethrowFromSystemServer();
}
}

/**
* <p>Attaches an accessibility overlay {@link android.view.SurfaceControl} to the
* specified
* window. This method should be used when you want the overlay to move and
* resize as the parent window moves and resizes.</p>
* <p>Generally speaking, an accessibility overlay will be a {@link android.view.View}.
* To embed the View into a {@link android.view.SurfaceControl}, create a
* {@link android.view.SurfaceControlViewHost} and attach the View using
* {@link android.view.SurfaceControlViewHost#setView}. Then obtain the SurfaceControl by
* calling <code> viewHost.getSurfacePackage().getSurfaceControl()</code>.</p>
* <p>To remove this overlay and free the associated resources, use
* <code> new SurfaceControl.Transaction().reparent(sc, null).apply();</code>.</p>
* <p>If the specified overlay has already been attached to the specified window
* this method does nothing.
* If the specified overlay has already been attached to a previous window this
* function will transfer the overlay to the new window.
* Services can attach multiple overlays. Use
* <code> new SurfaceControl.Transaction().setLayer(sc, layer).apply();</code>.
* to coordinate the order of the overlays on screen.</p>
*
* @param accessibilityWindowId The window id, from
* {@link AccessibilityWindowInfo#getId()}.
* @param s
*/
public void attachAccessibilityOverlayToWindow(
int accessibilityWindowId, @NonNull SurfaceControl sc) {
Preconditions.checkNotNull(sc, "SurfaceControl cannot be null")
AccessibilityInteractionClient.getInstance(this)
.attachAccessibilityOverlayToWindow(mConnectionId, accessibilityWindowId, sc);
}
}
```

5.2.2 Code Efficiency

Testing Objectives:

These are several rules that can save as testing objectives they are:

Testing is a process of executing program with the intent of finding an error.

A good test case is one that has high probability of finding an undiscovered error.

Source Code Testing:

This examines the logic of the system. if we are getting the output that is required by the user, then we can say that the logic is perfect.

Specification Testing:

We can set with, what program should do and how it should perform under various condition. This testing is a comparative study of evolution of system performance and system requirements.

Module level Testing:

In this error will be found at each individual module, it encourages the programmer to find and rectify the errors without affecting the other module.

5.3 Testing Approach

5.3.1 UNIT TESTING

Unit testing focuses on verifying the effort on the smallest unit of software-module. The local data structure is examined to ensure that the data stored temporarily maintains its integrity during all steps in the algorithm execution. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.

5.3.2 INTEGRATED TESTING

Data can be tested across an interface. One module can have an inadvertent, adverse effect on the other. Integration testing is a systematic technique for constructing a program structure while conducting tests to uncover errors associated with interring.

5.3.3 VALIDATION TESTING

It begins after the integration testing is successfully assembled. Validation succeeds when the software function in a manner that can be reasonably accepted by the client. In this the majority of the validation is done during the data entry operation where there is maximum possibility of entering wrong data. Other validation will be performed in all process where correct details and data should be entered to get the required results.

5.3.4 RECOVERY TESTING

Recovery testing is system that forces the software to fail in variety of ways and verifies that the recovery is properly performed. If recovery is automatic, re-initialization, and data recovery are each evaluated for correctness.

5.3.5 SECURITY TESTING

Security testing attempts to verify that protection mechanism built into system will in fact protect it from improper penetration. the tester may attempt to acquire password through external clerical means, may attack the system with custom software design to break down any defenses to others, and may purposely cause errors.

5.3.6 PERFORMANCE TESTING

Performance testing is used to test runtime performance of software within the context of an integrated system. Performance test are often coupled with stress testing and require both software and instrumentation.

5.3.7 BLACKBOX TESTING

Black-box testing focuses on functional requirement of software. It enables to derive it’s of input conditions that will fully exercise all functional requirements for program. It find error for following category:

- Incorrect or missing function
- Interface errors
- Errors in data structure or external database access.

5.3.8 OUTPUT TESTING

After performing the validation testing, the next step is output testing of the proposed system since no system would be termed as useful until it does produce the required output in the specified format. It mainly considers in two ways the screen format and printer format.

5.4 Test Cases

The test case is defined as a group of conditions under which a tester determined whether a software application is working as per requirement or not. Test cases designing includes

Chapter 6

RESULTS AND DISCUSSION

6.1 Test Reports

Reason of travelling intent: This intent is triggered when a user decides the destination for travelling and agrees to provide more details. With the help of this trigger the chatbot gets the reason of travelling of the user and as per that the further process will be carried out.

Hotel quality intent: This intent is triggered when a user provides all the details with whom they are travelling and all. This trigger ensures that the user wants to stay in which quality of hotel either 1Star, 2Star, 3Star, 4Star or 5Star.

Budget inquiry intent: This intent is triggered when a user plans a trip for the destination as per their choice. This trigger helps to gather the user’s budget for their trip so that the tourist company can customize the travelling package for the user.

Tourist attraction intent: This intent is triggered when a user asks the chatbot for recommendations on tourist attractions in a particular location. The chatbot would then provide information on popular sights, activities, and events.

Chapter 7

CONCLUSIONS

7.1 Conclusion

In conclusion, the trip assistant Chatbot developed using brainshop.ai and android studio is a powerful tool that can help users plan their trips and make travel arrangements. With its ability to understand natural language, the chatbot provides a seamless user experience and can provide helpful recommendations for hotels, restaurants, and attractions based on user preferences.

By leveraging Brainshop.ai's advanced AI technology, the chatbot can learn from user interactions and become smarter over time. Its integration with Android Studio makes it easy to deploy the chatbot on mobile devices, enabling users to access its features on-the-go.

Overall, the Trip Assistant Chatbot is an innovative solution that simplifies the trip planning process and makes travel more enjoyable. With its ability to provide personalized recommendations and 24/7 availability, it has the potential to become an essential tool for travellers around the world.

7.2 Limitations of the System

Limited Understanding: While the chatbot is capable of understanding natural language to some extent, it may not be able to accurately interpret all user inputs. This can lead to misunderstandings and incorrect responses.

Lack of Emotional Intelligence: As an AI-powered system, the chatbot lacks emotional intelligence and cannot respond to human emotions such as empathy or sympathy. This can lead to a suboptimal user experience, especially when dealing with sensitive topics.

Limited Customization: While the chatbot can be trained to learn from user interactions, its customization options are limited. This means that it may not be able to cater to the specific needs and preferences of individual users.

Inability to Handle Complex Queries: The chatbot may struggle to handle complex queries or multi-step processes, which can lead to frustration for users trying to complete a task.

7.3 Future Scope of the Project

Integration with Other Platforms: The chatbot could be integrated with other platforms such as social media, travel booking websites, and messaging apps. This would allow users to access its features across multiple channels and make it more accessible.

Enhanced Personalization: The chatbot could be improved to provide more personalized recommendations based on user preferences, travel history, and behavior patterns. This would enhance the user experience and make the chatbot more valuable to users.

Multilingual Support: The chatbot could be developed to support multiple languages, allowing users from different regions to access its features. This would increase its reach and make it more useful for a wider audience.

Improved Understanding: The chatbot could be trained to understand and interpret user inputs more accurately. This would make it more reliable and reduce the chances of misunderstandings or incorrect responses.

Overall, the Trip Assistant Chatbot system has a lot of potential for future development and enhancement, and with the right improvements, it could become an essential tool for travellers around the world.

REFRENCES

- Bansal, Himanshu, and Rizwan Khan. "A review paper on human computer interaction." International Journals of Advanced Research in Computer Science and Software Engineering 8 (2018): 53-56.
- Chatbot /Definition of chatbot in English by Lexico Dictionaries, 2019.
- Abu Shawar, B., & Atwell, E. (2007). Chatbots: Are They Really Useful? LDV-Forum Band, 22, 29-49.
- Ukpabi, D., Aslam, B., & Karjaluoto, H. (2019). Chatbot Adoption in Tourism Services: A Conceptual Exploration. in S. Ivanov, & C. Webster (Eds.), Robots, Artificial Intelligence, and Service Automation in Travel, Tourism and Hospitality (pp. 105-121). Emerald Publishing Limited. https://doi.org/10.1108/978-1-78756-687-320191006
- Agostinho, D. (2016). Chat Bots: how could they impact the travel industry? Amadeus IT Group, Retrieved from: http://www.amadeus.com/blog/22/04/chat-bots-impact-travelindustry/, Date of Access: 3 July, 2017
- Reshmi, S., & Balakrishnan, K. (2016). Implementation of an inquisitive chatbot for database supported knowledge bases. Sādhanā, 41(10), 1173- 1178.
- Dale, R. (2016). The return of the chatbots. Natural Language Engineering, 22(5), 811-817. doi: https://doi.org/10.1017/S1351324916000243.