A
Mini Project report on

VIRTUAL STUDENT ASSISTANT USING CHATBOT

Submitted To
Jawaharlal Nehru Technological University Hyderabad
In Partial Fulfilment of the Requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

ALAM K SATHYA CHOWDARY	197Z1A0508
BHOOMIKA MACHA	197Z1A0522
DONGOLU NAVYA	197Z1A0544

Under the Guidance of
Mr. G. VENU GOPAL
Assistant Professor

SCHOOL OF ENGINEERING
Department of Computer Science and Engineering

NALLA NARASIMHA REDDY
EDUCATION SOCIETY'S GROUP OF INSTITUTIONS
(Approved by AICTE, New Delhi, Affiliated to JNTU-Hyderabad)
Chowdariguda (VIll) Korremula 'x' Roads, Via Narapally, Ghatkesar (Mandal)
Medchal (Dist), Telangana-500088

2022-2023

Nalla Narasimha Reddy Group of Institutions
NNRG Integrated Campus
NALLA NARASIMHA REDDY
Education Society's Group of Institutions - Integrated Campus
Approved by AICTE, New Delhi, Affiliated to JNTU - Hyderabad
SCHOOL OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project report titled **"Virtual Student Assistant using Chatbot"** is being submitted by **Alam K Sathya Chowdary (197Z1A0508)**, **Bhoomika Macha (197Z1A0522)** and **Dongolu Navya (197Z1A0544)** in Partial fulfilment for the award of **Bachelor of Technology in Computer Science & Engineering** is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

Internal Guide
(Mr. G. Venu Gopal)

Head of the Department
(Dr. K. Rameshwaraiah)

Submitted for the University Examination held on…….

External Examiner

DECLARATION

We Alam K Sathya Chowdary, Bhoomika Macha and Dongolu Navya, the students of **Bachelor of Technology in Computer Science and Engineering, Nalla Narasimha Reddy Education Society's Group Of Institutions**, Hyderabad, Telangana, hereby declare that the work presented in this project work entitled **Virtual Student Assistant using Chatbot** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of engineering ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning.

Alam K Sathya Chowdary	197Z1A0508
Bhoomika Macha	197Z1A0522
Dongolu Navya	197Z1A0544

Date:

Signature:

ACKNOWLEDGEMENT

We express our sincere gratitude to our guide **Mr. G. Venu Gopal,** Assistant Professor, Department of Computer Science and Engineering, NNRESGI, who motivated throughout the period of the project and also for his valuable and intellectual suggestions apart from his adequate guidance, constant encouragement right throughout our work.

We would like to express our profound gratitude to our mini project In-charge **Mrs. CH. Ramya,** Assistant Professor, Department of Computer Science and Engineering, NNRESGI, for her support and guidance in completing our project and for giving us this opportunity to present the project work.

We profoundly express thanks to **Dr. K. Rameshwaraiah**, Professor & Head of Computer Science and Engineering, NNRESGI, for his cooperation and encouragement in completing the project successfully.

We wish to express our sincere thanks to **Dr. G. Janardhana Raju,** Dean School of Engineering, NNRESGI, for providing the facilities for completion of the project.

We wish to express our sincere thanks to **Dr**. **C. V. Krishna Reddy,** Director, NNRESGI, for providing the facilities for completion of the project.

Finally, we would like to thank overall mini-project coordinator, members of Project Review Committee (PRC), all the faculty members and supporting staff, Department of Computer Science and Engineering, NNRESGI, for extending their help in all circumstances.

By

Alam K Sathya Chowdary	197Z1A0508
Bhoomika Macha	197Z1A0522
Dongolu Navya	197Z1A0544

ABSTRACT

A chat-bot is an automated program that simulates human conversation through textual input, in lieu of providing direct contact with a live human like agent. It is designed to convincingly simulate the way a human would behave as a conversational partner. Students tend to spend hours in online research and then making the notes in their terms of understanding which gets tedious and monotonous.

Our Virtual Student Assistant is a Chabot intended to help students in gathering notes rapidly. It is an interactive and intelligent student assistant which has been proposed to overcome the challenges of traditional information/notes gathering process. The goal is to provide people a quick and easy way to have their requests addressed. As well as to offer them a medium to access notes instantly. This virtual student assistant enhances the user experience while delivering accurate and efficient responses for their queries with 24X7 availability. This system is trained using DialogFlow and is integrated with telegram application.

Keywords: Chatbot, Automated, DialogFlow, Telegram

CONTENTS

List of Figures

List of Tables

List of Abbreviations

S. No.	Abbreviation	Definition
1.	NLP	Natural Language Processing
2.	SQL	Structured Query Language

1. INTRODUCTION

1.1 MOTIVATION

A Chat-Bot is a computer program that simulates and processes human conversation, allowing humans to interact with digital devices as if they were communicating with a real person. They are now becoming essential in several industries. One such industry which can truly gain from this technology is the education sector.

This Virtual Student Assistant is a chatbot intended to help students in gathering notes rapidly. It enhances the user experience while delivering accurate and efficient responses for their queries. It allows students to receive personalized assistance and can tremendously save their time and efforts. It also improves user productivity and experience by managing tasks of the user and dispensing notes effortlessly.

1.2 PROJECT STATEMENT

Usually, user needs to manually manage multiple sets of applications/online sources to complete one task. That is Students tend to spend hours in online research and then making the notes in their terms of understanding. They check different sites and sources for the same; this gets extremely tedious and is extremely time consuming. There is need of a system that can manage these tasks effortlessly.

Virtual Student Assistant Chatbot is one such interactive and intelligent system intended to help students in gathering notes rapidly. It is an intelligently engineered entity capable of answering user questions accurately through proper context and intent understanding. All of this only happens with rigorous optimization and the by feeding the system with adequate amount of quality information for it to work efficiently.

1.3 OBJECTIVE OF PROJECT

The primary objective of this project is to create a chatbot that receives questions from users, tries to understand the question, and provides appropriate responses. The goal is to provide people a quick and easy way to have their requests addressed. As well as to offer them a medium to access notes rapidly.

Virtual assistants enable users to give natural language input commands in order to operate the chatbot (which is developed using DialogFlow and is integrated with telegram application).

There is an increased overall awareness and a higher level of comfort demonstrated specifically by millennial consumers. In this ever-evolving digital world where speed, efficiency, and convenience are constantly being optimized, it's clear that we are moving towards less screen interaction.

1.4 LIMITATIONS OF PROJECT

Chatbots can come close to mimicking a human like conversation experience. Melding with natural language processing and understanding technologies has made these experiences seem much more realistic. While the technology has become better, there are certain limitations of what the current virtual student assistant / NLU and NLP technology can do.

- **Limited Question Scope:** Creating a chatbot able to answer every single question about bot is not possible to implement with current technology and within the duration of the project, so the system will be able to answer questions about limited topics.
- **Language:** The system will only support questions in Standard English.
- **Need for Continuous Training Data:** There is a popular misconception that systems like these works by itself without human supervision, but the best chatbots require humans to review each programmatic rule and response, cleanse the data, and label the data. And train it further to extend its scope.

  As NLU and NLP technology continues to mature, these chatbot limitations can be improved over time. A way to get around these chatbot limitations is being able to provide a handoff to a human agent as well as having human resources reviewing the chatbot data to continue to refine the data and training phrases the chatbot is using when making the decisions on the conversation paths for the customer.

2. LITERATURE SURVEY

A literature survey or a literature review in a project report is that section which shows the various analyses and research made in the field of your interest and the results already published, taking into account the various parameters of the project and the extent of the project.

It is the most important part of your report as it gives you a direction in the area of your research. It helps you set a goal for your analysis - thus giving you your problem statement.

Literature survey is something when you look at a literature (publications) in a surface level, or an Ariel view. It incorporates the study of place people and productions are setting of research. It is phase where the analyst tries to know about what is all the literature related with one range of interest. Also, the relevant literature works are short-listed. Moreover, literature survey guides or helps the researcher to define/find out/identify a problem.

2.1 INTRODUCTION

A chatbot is a computer program designed to simulate conversation with human users, especially over the Internet. Chatbots are often used in customer service, marketing, and other areas where they can interact with users in real-time and provide information or assistance.

A telegram virtual student assistant chatbot is a type of chatbot that is specifically designed to provide assistance to students through the messaging app Telegram. These chatbots can be accessed by sending a message to the chatbot through the app, and students can communicate with them in natural language to request information or assistance

Virtual Student Assistant is a notes dispensing telegram chatbot that is specifically designed to provide students with access to notes and other study materials through the messaging app Telegram. These chatbots can be accessed through the app by sending a message to the chatbot and requesting specific notes or access to other study materials.

Notes dispensing virtual student assistant telegram chatbots can be a useful tool for students who are preparing for exams or working on a research project and prefer to use Telegram for communication and organization. They can help students save time by

providing quick and convenient access to the notes and materials they need, without the need to search through multiple sources or physically visit a library or resource centre.

Overall, telegram virtual student assistant chatbots are designed to make the lives of students easier and more efficient, allowing them to focus on their studies and other important tasks. They can be accessed anytime, anywhere, making them a convenient resource for students who need assistance outside of normal business hours.

2.2 EXISTING SYSTEM

Many times, students face trouble finding information and also face difficulty in searching for a particular piece of information from the internet. Due to which they have to manually visit the college in order to gather related information as per the needs. This process can be monotonous and time consuming. An effective solution is required to overcome these drawbacks.

Disadvantages:

There are several limitations of traditional information gathering for students that can make it difficult for them to access the resources and support they need. Some of these limitations include:

- **Limited access to information**: Students may not have access to all the information they need, either because it is not available to them or because they do not know where to find it.
- **Time constraints:** Students often have busy schedules and may not have enough time to search for and gather the information they need.
- **Limited resources:** Students may not have the resources they need, such as books or other materials, to access the information they need.
- **Lack of personalization:** Traditional information gathering methods may not be tailored to a student's specific needs and interests, making it harder for them to find relevant and useful information.
- **Limited support:** Students may not have access to support and guidance from advisors or other professionals who can help them navigate their academic journey.

Virtual student assistants can help to overcome some of these limitations by providing students with quick and easy access to information, personalized recommendations, and support from advisors and other professionals.

2.3 PROPOSED SYSTEM

To overcome the challenges of traditional information gathering process a Virtual Student Assistant Chatbot is been proposed. It is a Chabot intended to help students in gathering notes rapidly. This virtual assistant will address student queries and facilitates expert level assistance with 24x7 availability. The goal is to provide people a quick and easy way to have their requests addressed.

Advantages:

There are several advantages to using virtual student assistant chatbot:

- **24/7 availability:** Virtual student assistant chatbot can operate 24/7, allowing users to get assistance anytime they need it.
- **Increased efficiency:** Virtual student assistant Chatbot can handle multiple user inquiries at once, increasing the efficiency of user service.
- **Handling heavy workload:** Virtual student assistant chatbot can handle a large volume of user inquiries/requests, reducing the need for a large user service team and resulting in cost savings.
- **Convenience:** Virtual student assistant Chatbot allow users to get assistance quickly and easily through a familiar interface i.e. Telegram.
- **Improved user experience:** Chatbots can provide faster and more efficient user service, leading to improved user satisfaction. A virtual student assistant is a computer-based system that is designed to provide support and assistance to students in their academic endeavours by providing those notes rapidly. This can be done through Telegram chat interface, where students can type their questions and receive immediate responses in natural language.

Overall, virtual student assistants can be a valuable resource for students looking for additional support and guidance as they navigate their academic journey.

3. SYSTEM ANALYSIS

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components.

System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem-solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. Analysis specifies what the system should do.

3.1 INTRODUCTION

In this phase the requirements are gathered and analysed. User's requirements are gathered in this phase. This phase is the main focus of the users and their interaction with the system.

These are few questions raised:

- Who is going to use the system?
- How will they use the system?
- What data should be input into the system?
- What data should be output by the system?

These general questions are answered during a requirement gathering phase. After requirement gathering these requirements are analysed for their validity and the possibility of incorporating the requirements in the system to be development is also studied.Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

3.2 SOFTWARE REQUIREMENTS

It deals with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed. The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view. We should try to understand what sort of requirements may arise in

the requirement elicitation phase and what kinds of requirements are expected from the software system.

A software requirement can be of 3 types:

- Functional requirements
- Non-functional requirements
- Domain requirements

Functional Requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other.

They are also called non-behavioural requirements.

They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

NFR's are classified into following types:

- Interface constraints
- Performance constraints: response time, security, storage space, etc.
- Operating constraints
- Life cycle constraints: maintainability, portability, etc.
- Economic constraints

The process of specifying non-functional requirements requires the knowledge of the functionality of the system, as well as the knowledge of the context within which the system will operate.

Domain requirements: Domain requirements are the requirements which are characteristic of a particular category or domain of projects. The basic functions that a system of a specific domain must necessarily exhibit come under this category. For instance, in academic software that maintains records of a school or college, the functionality of being able to access the list of faculty and list of students of each grade is a domain requirement. These requirements are therefore identified from that domain model and are not user.

The software requirements that are required for this project are as follows:

- Operating System : Windows 7 and above
- Programming Language : Python
- Backend Framework : Django
- Chat Bot Training Interface : Dialog Flow API
- Database : PostgreSQL
- Tool : Visual Studio Code, Python Anywhere
- UML's : Star UML

3.3 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, a hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application.

The hardware requirements that are required for this project are as follows:

- Device : Any device with telegram functionality
- RAM : 1GB or above
- Processor : Intel Core i2 and above
- Hard Disk : Minimum 40GB

3.4 CONTENT DIAGRAM OF PROJECT

A content diagram is a visual representation of the structure and organization of the content within a website or application. It is typically used to plan and design the content strategy for a project, as well as to communicate the content hierarchy and relationships to stakeholders.

The Content diagram is an extension of UML notation. The purpose of the Content diagram is to generate or represent a project structure (diagrams) and relations between them.

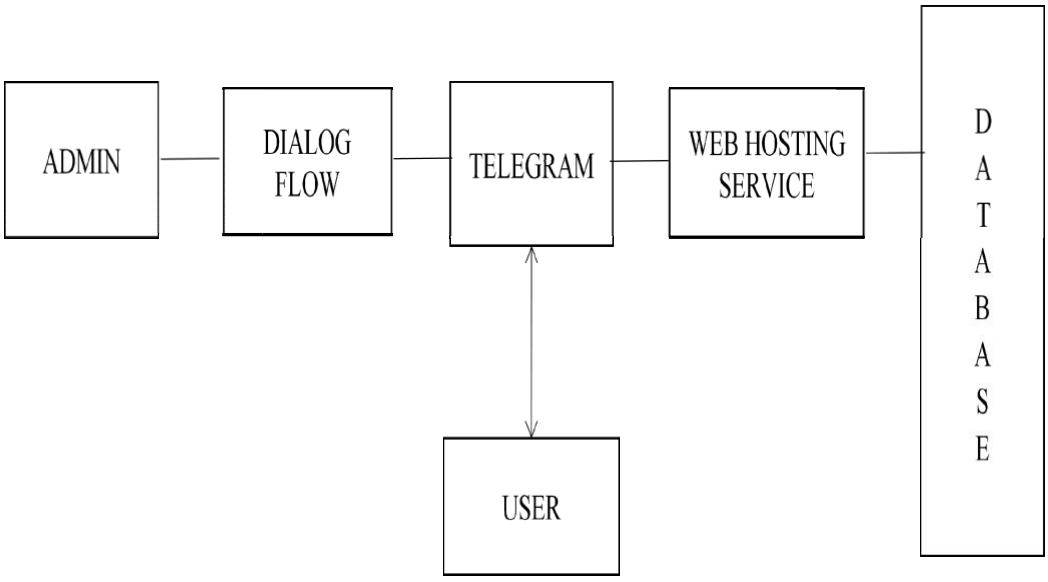


Fig: 3.1 Content diagram

First, admin designs conversation flow and personality of the chatbot via DialogFlow the bot is then integrated with telegram platform.

Finally, we deploy it onto a web hosting platform such as PythonAnywhere through this the data present in a database could be accessed.

The user can now receive suitable responses and notes / study materials just by sending a simple message to the virtual student assistant telegram chatbot.

4. SYSTEM DESIGN

The process of design involves “conceiving and planning out in mind and making a drawing, pattern or a sketch”. The system design transforms a logical representation of what a given system is required to do into the physical reality during development. Important design factors such as reliability, response time, throughput of the system, maintainability, expandability etc., should be taken into account. Design constraints like cost, hardware limitations, standard compliance etc should also be dealt with. The task of system design is to take the description and associate with it a specific set of facilities-men, machines (computing and other), accommodation, etc., to provide complete specifications of a workable system. This new system must provide for all of the essential data processing and it may also do some of those tasks identified during the work of analysis as optional extras. It must work within the imposed constraints and show improvement over the existing system. At the outset of design a choice must be made between the main approaches. Talks of ‘preliminary design’ concerned with identification analysis and selections of the major design options are available for development and implementation of a system. These options are most readily distinguished in terms of the physical facilities to be used for the processing who or what does the work.

4.1 INTRODUCTION

Software design is the process by which an agent creates a specification of a software artifact, intended to accomplish goals, using a set of primitive components and subject to constraints. Software design may refer to either “all the activity involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems” or “the activity following requirements specification and before programming, as in a stylized software engineering process.” Software design usually involves problem solving and planning a software solution. This includes both a low-level component design and a high-level, architecture design.

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization.

Once the software requirements have been analysed and specified the software design involves four technical activities – design, coding, implementation and testing that are required to build and verify the software.

The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer's requirements into finished software or a system.

- **Actor:** An actor is a model element that interacts with a system.

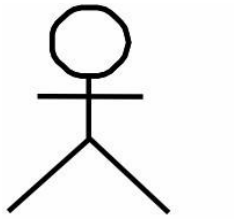


Fig: 4.1 Actor

- **Use Case:** A use case describes a function that a system performs to achieve the user's goal.

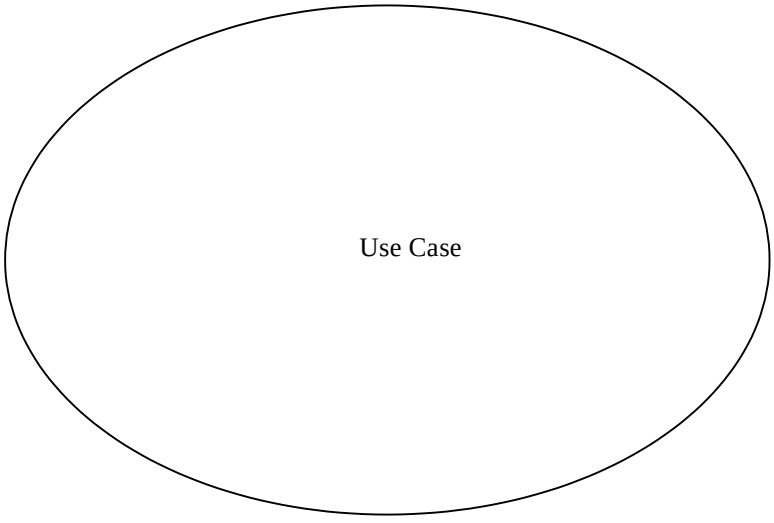


Fig: 4.2 Use Case

- **Use case system:** A system of a use case defines and represents boundaries of a business, software system, physical system or device, subsystem, component or even single class in relation to the requirements gathering and analysis.

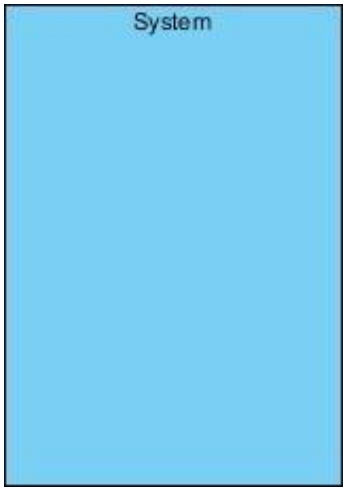


Fig: 4.3 System boundary

- **Associations:** A line between actors and use cases. It describes which actors are associated with which use cases.

Fig: 4.4 Association

- **Class:** A class represents an object or a set of objects that share a common structure and behaviour.

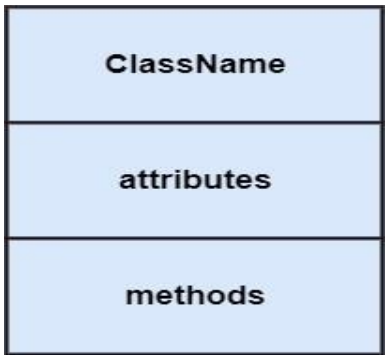


Fig: 4.5 Class

- **Generalization:** A generalization is a relationship between a parent class and a child class. In this, the child class is inherited from the parent class.

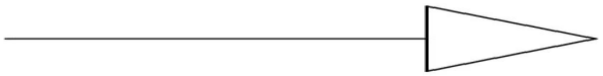


Fig: 4.6 Generalization

- **Composition:** It portrays the dependency between the parent and its child.

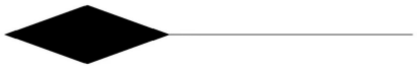


Fig: 4.7 Composition

- **Control flow:** The control flow determines the flow within an activity.

Fig: 4.8 Control Flow

- **Decision Box:** It makes sure that the control flow or object flow will follow only one path.

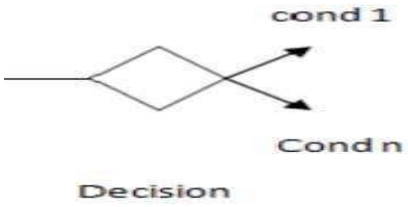


Fig: 4.9 Decision Box

- **Initial State:** It depicts the initial stage or beginning of the set of actions.

Fig: 4.10 Initial state

- **Action:** An action is a named element that is the fundamental unit of an executable functionality.

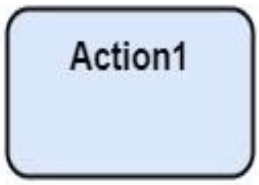


Fig: 4.11 Action Box

- **Join:** Join nodes are used to support concurrent activities converging into one.

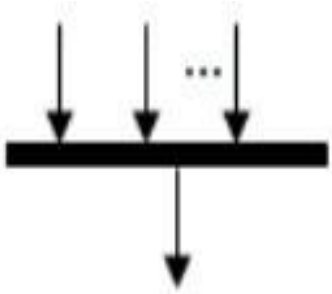


Fig: 4.12 Join

- **Call Message:** It depicts workflow or activity over time using messages passed from element to element.

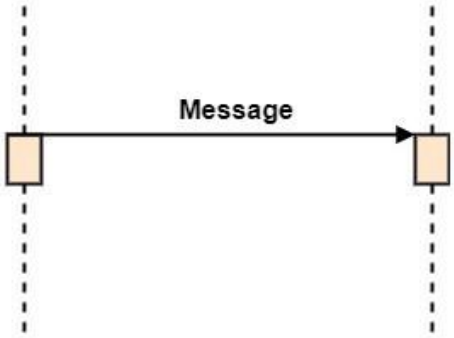


Fig: 4.13 Call Message

- **Activation:** It is represented by a thin rectangle on the life line. It describes that time period in which an operation is performed by an element.

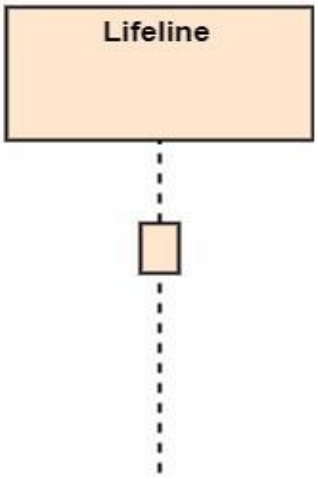


Fig: 4.14 Activation

- **Life Line:** Life Line represents the object that participates in an interaction.

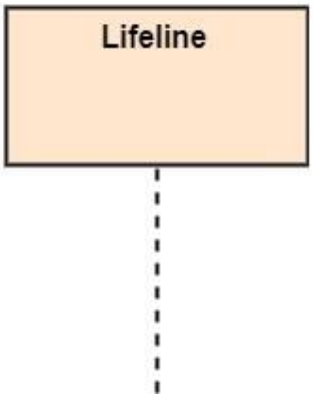


Fig: 4.15 Life Line

- **Final State:** It is the stage where all the control flows and object flows end.

Fig: 4.16 Final State

- **<<Include>> :** <<Include>> relationship represents behaviour that is factored out of the use case.

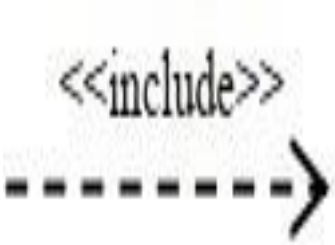


Fig: 4.17 Include

- **Objects:** The object is represented by specifying their name and class to differentiate one object from another, it is necessary to name them.

4.2 ER/UML DIAGRAMS

UML stands for Unified Modelling Language which is used in object-oriented software engineering. It is a standard language for specifying, visualizing, constructing, and documenting the artefacts of the software systems. UML is different from other common programming languages like C++, Java, and COBOL etc. It is pictorial language used to make software blueprints.

Although typically used in software engineering it is a rich language that can be used to model an application structure, behaviour and even business processes. There are 8 UML diagram types to help us model this behaviour.

There are two types of UML modelling:

- Structural Modelling
- Behavioural Modelling

Structural Modelling:

Structural model represents the framework for the system and this framework is the place where all other components exist. Hence, the class diagram, component diagram and deployment diagrams are part of structural modelling. They all represent the elements and the mechanism to assemble them.

The structural model never describes the dynamic behaviour of the system. Class diagram is the most widely used structural diagram.

Structural Modelling captures the static features of a system. They consist of the following:

i. Classes diagrams
ii. Objects diagrams
iii. Deployment diagrams
iv. Package diagrams
v. Composite structure diagram
vi. Component diagram

Behavioural Modelling:

Behavioural model describes the interaction in the system. It represents the interaction among the structural diagrams. Behavioural modelling shows the dynamic nature of the system. They consist of the following:

i. Activity diagrams
ii. Interaction diagrams
iii. Use case diagrams

All the above show the dynamic sequence of flow in a system.

4.1.1 USE CASE DIAGRAM:

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. The "actors" are people or entities operating under defined roles within the system.

As the most known diagram type of the behavioural UML diagrams, use-case diagrams give a graphic overview of the characters involved in a system, different functions needed by those characters and how these different functions are interacted.

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. They provide a good high-level analysis from outside the system. Use case diagrams specify how the system interacts with actors without worrying about the details of how that functionality is implemented.

A key concept of the use case modelling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating systems behaviour in the user's terms by specifying all externally visible systems behaviour.

Purpose of Use Case Diagram:

Use case diagrams are typically developed in the early stage of development and are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

When the initial task is complete, use case diagrams are modelled to present the outside view.

In brief, the purposes of use case diagrams can be said to be as follows −

- Specify the context of a system
- Capture the requirements of a system
- Validate a systems architecture
- Drive implementation and generate test cases
- Developed by analysts together with domain experts

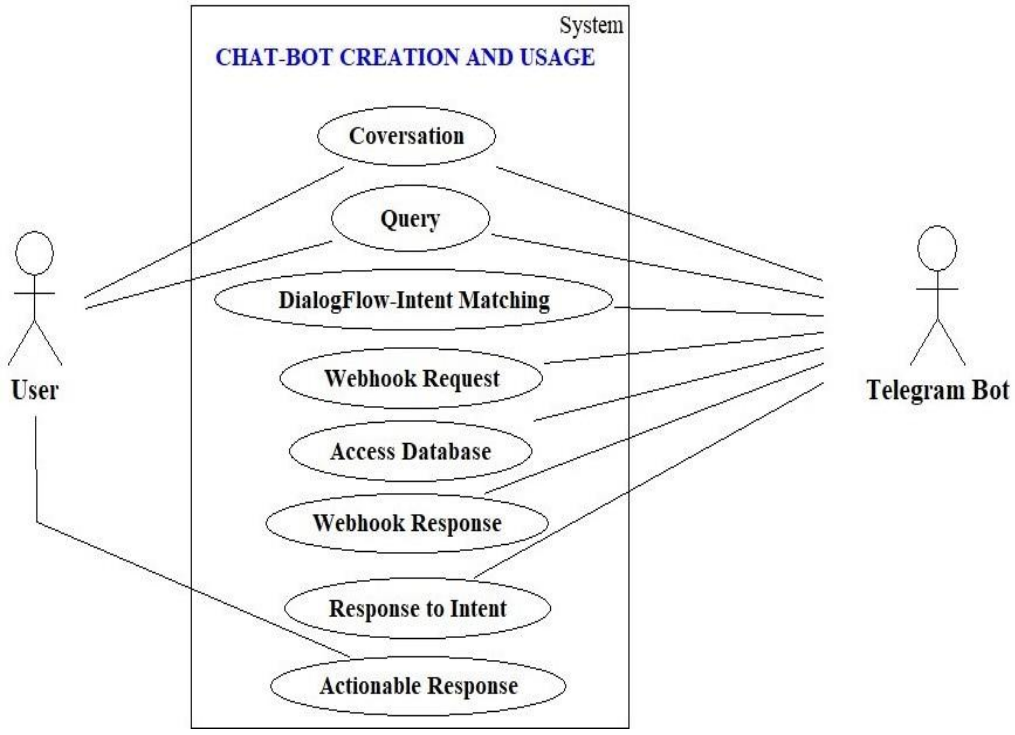


Fig: 4.2.1 Use case diagram

4.1.2 CLASS DIAGRAM

Class diagrams are the main building blocks of every object-oriented method. It is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

Purpose of Class Diagram:

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application; however class diagram is a bit different. It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as –

- This is the only UML which can appropriately depict various aspects of OOPs concept.
- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

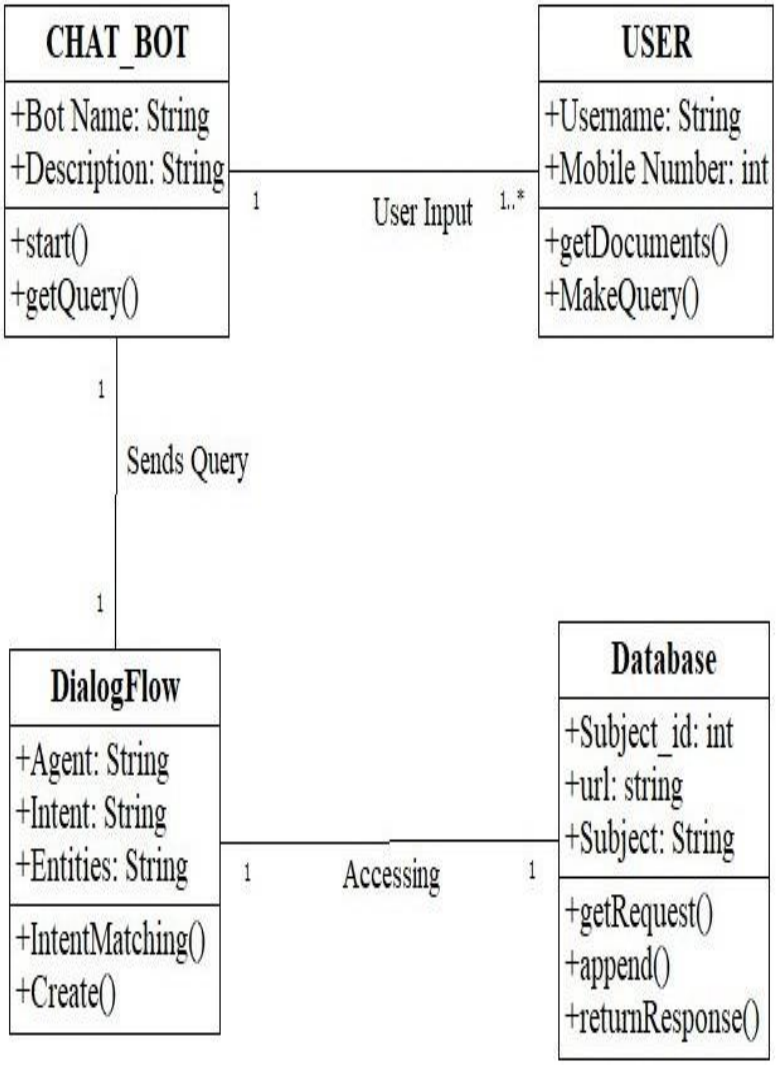


Fig: 4.2.2 Class diagram

4.1.3 SEQUENCE DIAGRAM

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

Sequence diagrams emphasize on time sequence of messages and are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.

Purpose of Sequence Diagram:

The purpose of sequence diagrams is to visualize the interactive behaviour of the system. Visualizing the interaction is a difficult task. Hence, the solution is to use different types of models to capture the different aspects of the interaction.

Sequence diagrams are used to capture the dynamic nature but from a different angle.

The purpose of sequence diagram is −

- To capture the dynamic behaviour of a system.
- To describe the message flow in the system.
- To describe the structural organization of the objects.
- To describe the interaction among objects.

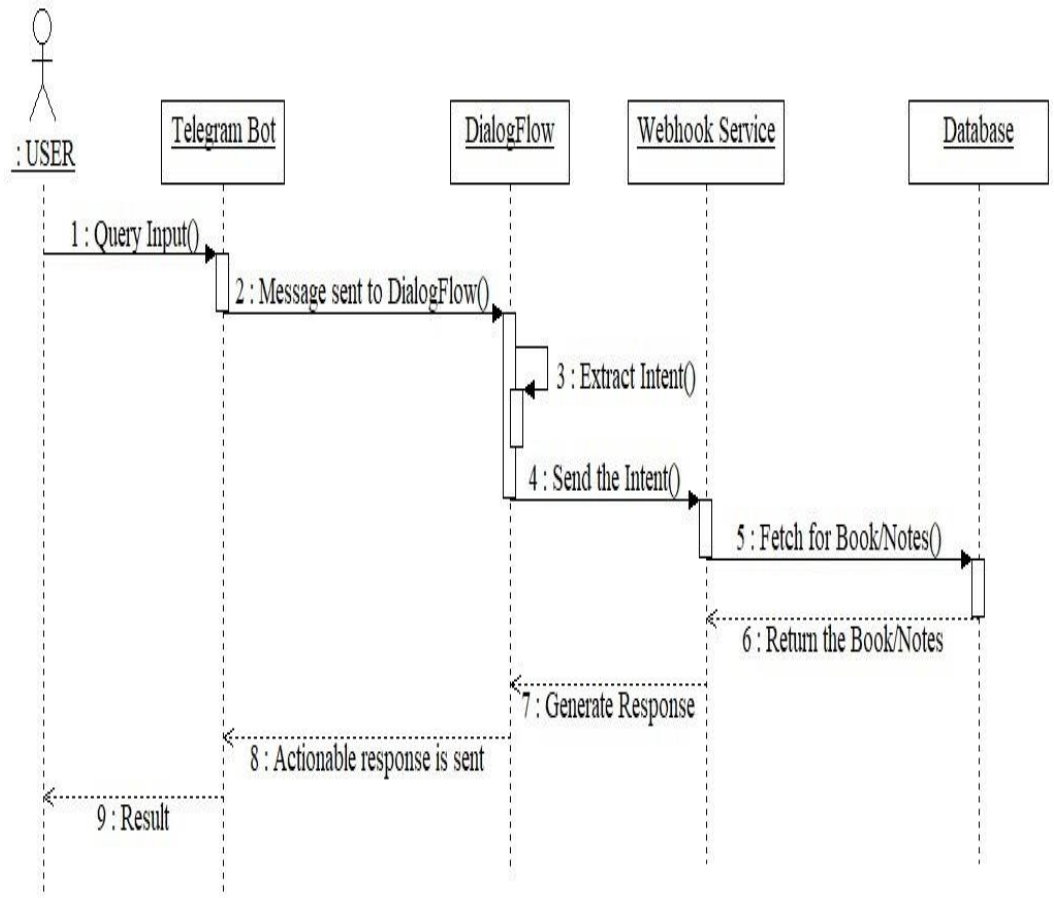


Fig: 4.2.3 Sequence diagram

4.2.4 COLLABORATION DIAGRAM

Collaboration diagram is also known as a communication diagram, is an illustration of the relationships and interaction among software objects in the Unified Modelling Language (UML). These diagrams can be used to portray the dynamic behaviour of a particular use case and define the role of each object. Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction. A model is then built using the relationships between those elements. Several vendors offer software for creating and editing collaboration diagrams.

Notations of a Collaboration Diagram

Following are the components of a component diagram that are enlisted below:

1. **Objects:** The representation of an object is done by an object symbol with its name and class underlined, separated by a colon.

In the collaboration diagram, objects are utilized in the following ways:

- The object is represented by specifying their name and class.
- It is not mandatory for every class to appear.
- A class may constitute more than one object.
- In the collaboration diagram, firstly, the object is created, and then its class is specified.
- To differentiate one object from another object, it is necessary to name them.

2. **Actors:** In the collaboration diagram, the actor plays the main role as it invokes the interaction. Each actor has its respective role and name. In this, one actor initiates the use case.

3. **Links:** The link is an instance of association, which associates the objects and actors. It portrays a relationship between the objects through which the messages are sent. It is represented by a solid line. The link helps an object to connect with or navigate to another object, such that the message flows are attached to links.

4. **Messages:** It is a communication between objects which carries information and includes a sequence number, so that the activity may take place. It is represented by a labelled arrow, which is placed near a link. The messages are sent from the sender to the receiver, and the direction must be navigable in that particular direction. The receiver must understand the message.

Steps for creating a Collaboration Diagram:

1. Identify behaviour whose realization and implementation is specified

2. Identify the structural elements (class roles, objects, subsystems) necessary to carry out the functionality of the collaboration.

- Decide on the context of interaction: system, subsystem, use case and operation

3. Model structural relationships between those elements to produce a diagram showing the context of the interaction

4. Consider the alternative scenarios that may be required

- Draw instance level collaboration diagrams, if required.
- Optionally draw a specification level collaboration diagram to summarize the alternative scenarios in the instance level sequence diagrams.

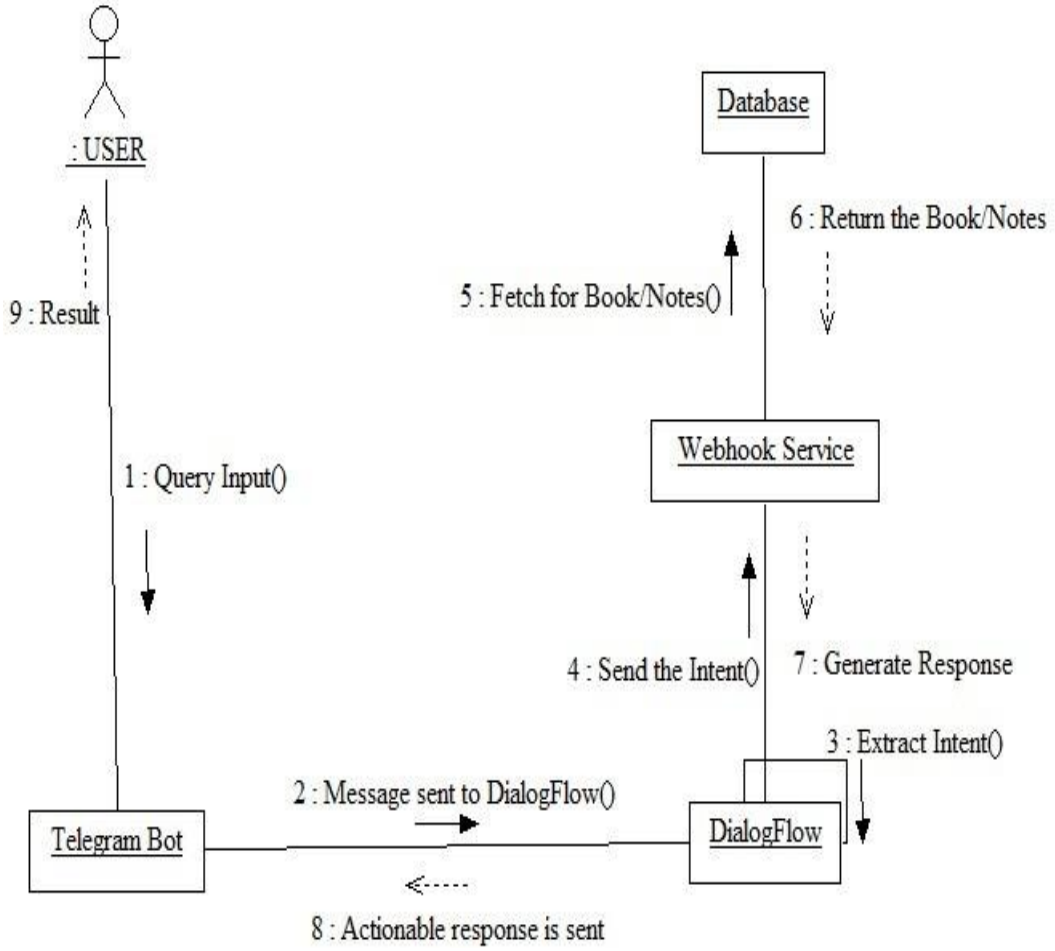
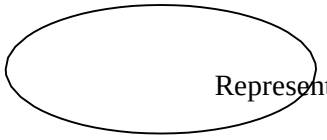
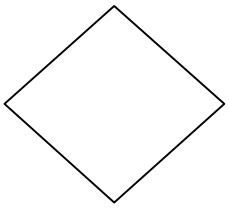


Fig: 4.2.4 Collaboration diagram

4.2.5 ENTITY-RELATIONSHIP DIAGRAM

- An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database.
- The main components of ER model are: entity set and relationship set.
- An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database..

The symbols used in E-R diagrams are:

SYMBOL	PURPOSE
	Represent Entity Sets
	Represent Attributes
	Represent Relationship Sets
	Line represents flow

1. **Entity:** An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
2. **Attribute:** The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.
3. **Relationship:** A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

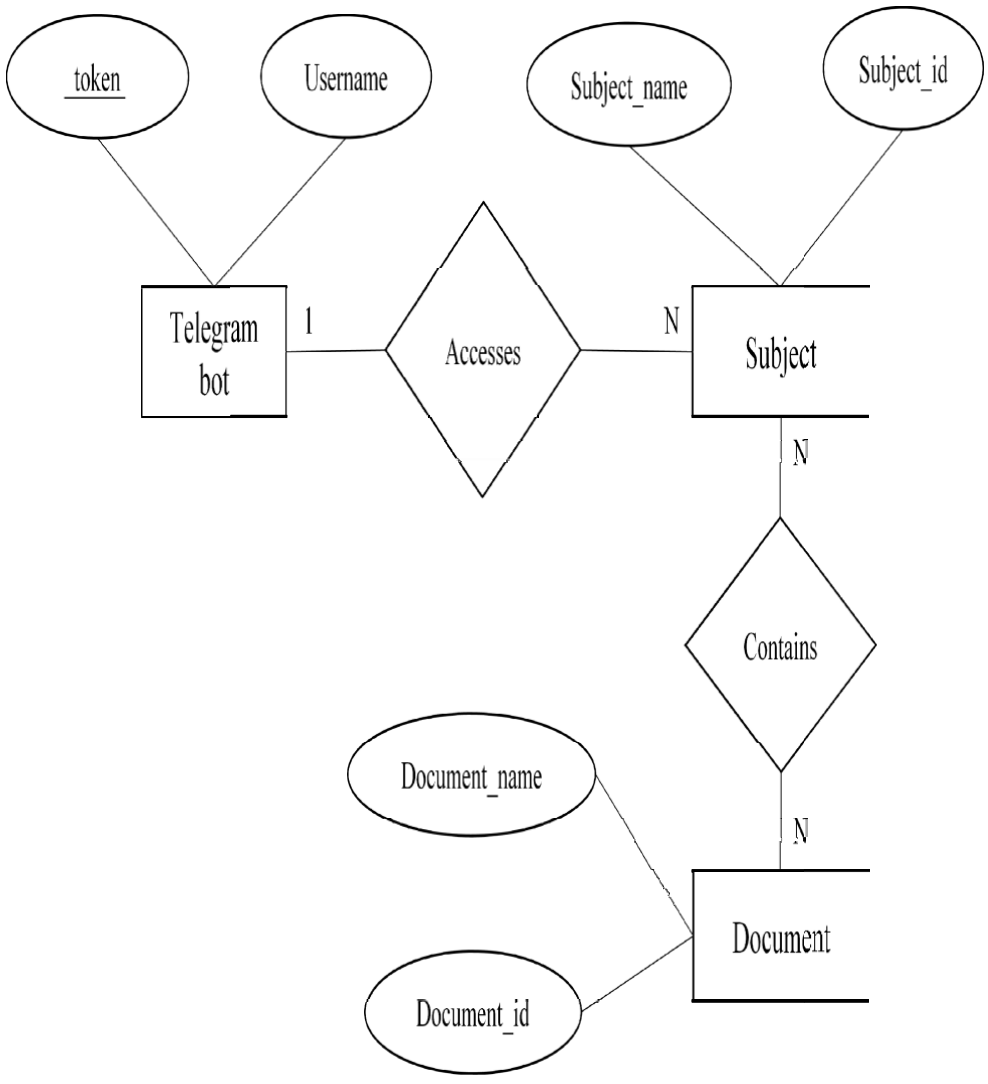


Fig: 4.2.5 ER diagram

5. IMPLEMENTATION AND RESULTS

5.1 INTRODUCTION

Functions are used for placing or storing the code which is to be repeated several times. For example, if we need same code, then we must have to write that code again and again. So in order to remove this we use functions.

Implementation is the stage where the theoretical design is turned into a working system. The most crucial stage in achieving a new successful system is giving confidence on the new system for the users that it will work efficiently and effectively.

The system can be implemented only after thorough testing is done and if it is found to work according to the specification.

It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change over an evaluation of change over methods apart from planning. Two major tasks for preparing the implementation are education and training of the users and testing of the system.

5.2 METHOD OF IMPLEMENTATION

The more complex the system being implemented, the more involved will be the systems analysis and design efforts required for implementation. The implementation phase comprises of several activities. The required hardware and software acquisition is carried out. The system may require some software to be developed. For this, programs are written and tested. The user then change over to his new fully tested system and the old system is discontinued.

5.2.1 MODULES DESCRIPTION

This project is composed of five main modules:

- Module 1: Chatbot creation and Administration
- Module 2: Chatbot usage and Running
- Module 3: Back-End Development
- Module 4: Training and Integration
- Module 5: Deployment

Chatbot creation and Administration: First, you will need to create a bot by talking to the BotFather on Telegram. To do this, you will need to send a message to the BotFather (using the @BotFather username) with the command "/newbot" and follow the prompts to

choose a name and username for your bot. The BotFather will then provide you with an API token that you will use to authenticate your bot.

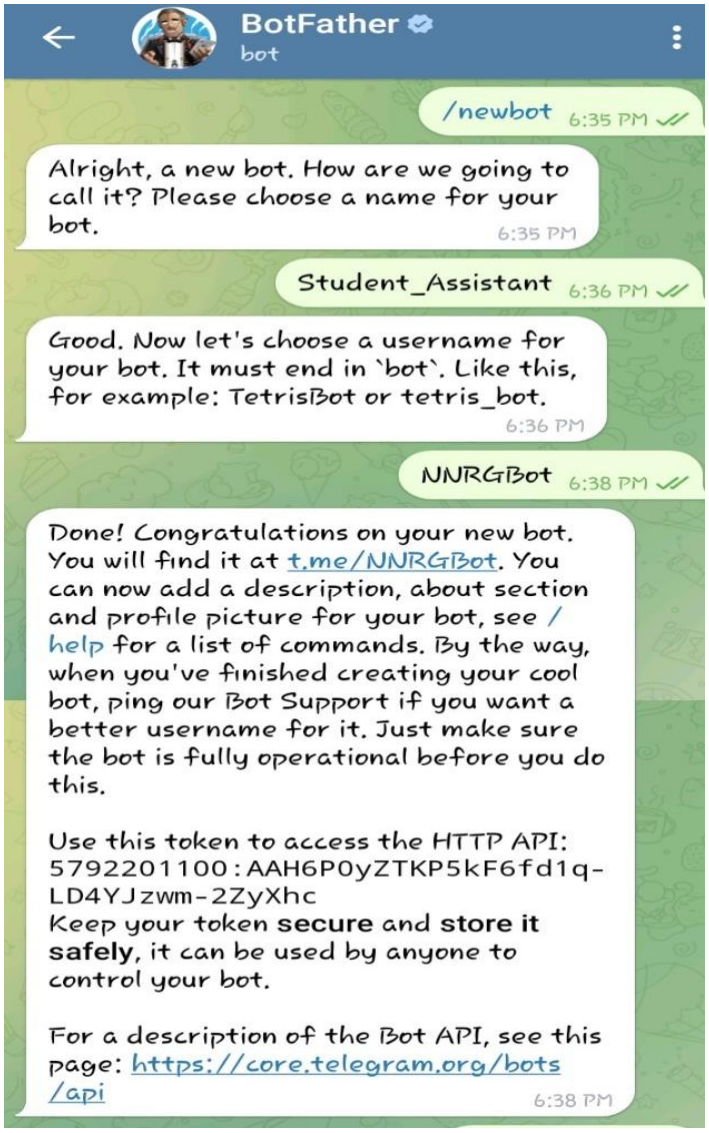


Fig: 5.2.1.1 Creating a chatbot on telegram application

Chatbot usage and Running: In order to use a virtual student assistant Chatbot, we need to look for it in telegram application i.e. search it by its username. You will be able to find the chat interface.

In order to run the Chatbot and to start a conversation with it /Start command is entered.

Back-End Development: This module contains the hardware and software requirements that have been used to implement the proposed system. The section of hardware is very

important in the existence and proper working of any software. In the selection of hardware, the size and the capacity requirements are also important.

Training and Integration: Dialog flow is a natural language understanding platform that makes it easy to design and integrate a conversational user interface into your telegram app. Using Dialog flow; you can provide new and engaging ways for users to interact with your Chatbot.

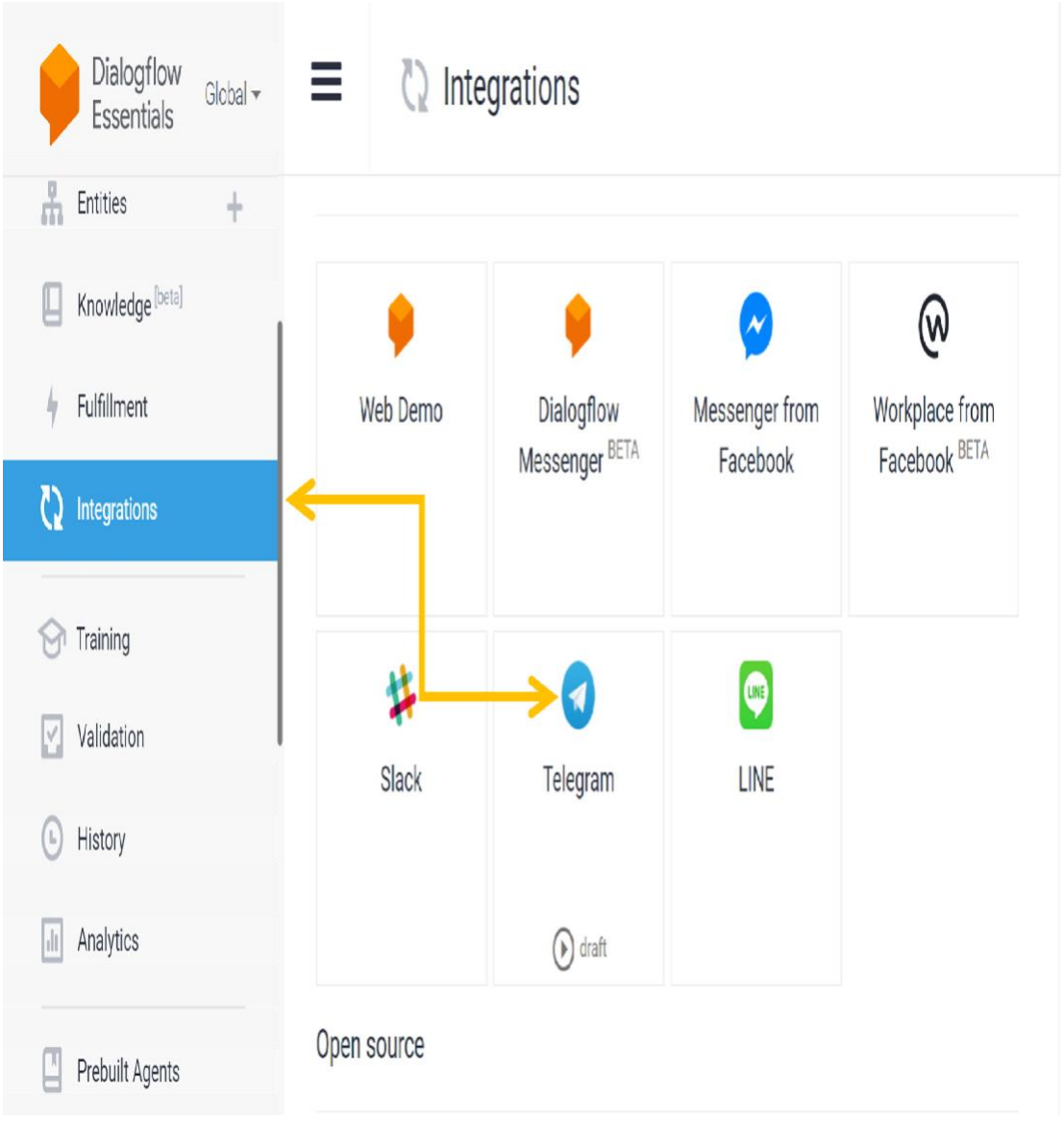


Fig 5.2.1.2 Integration

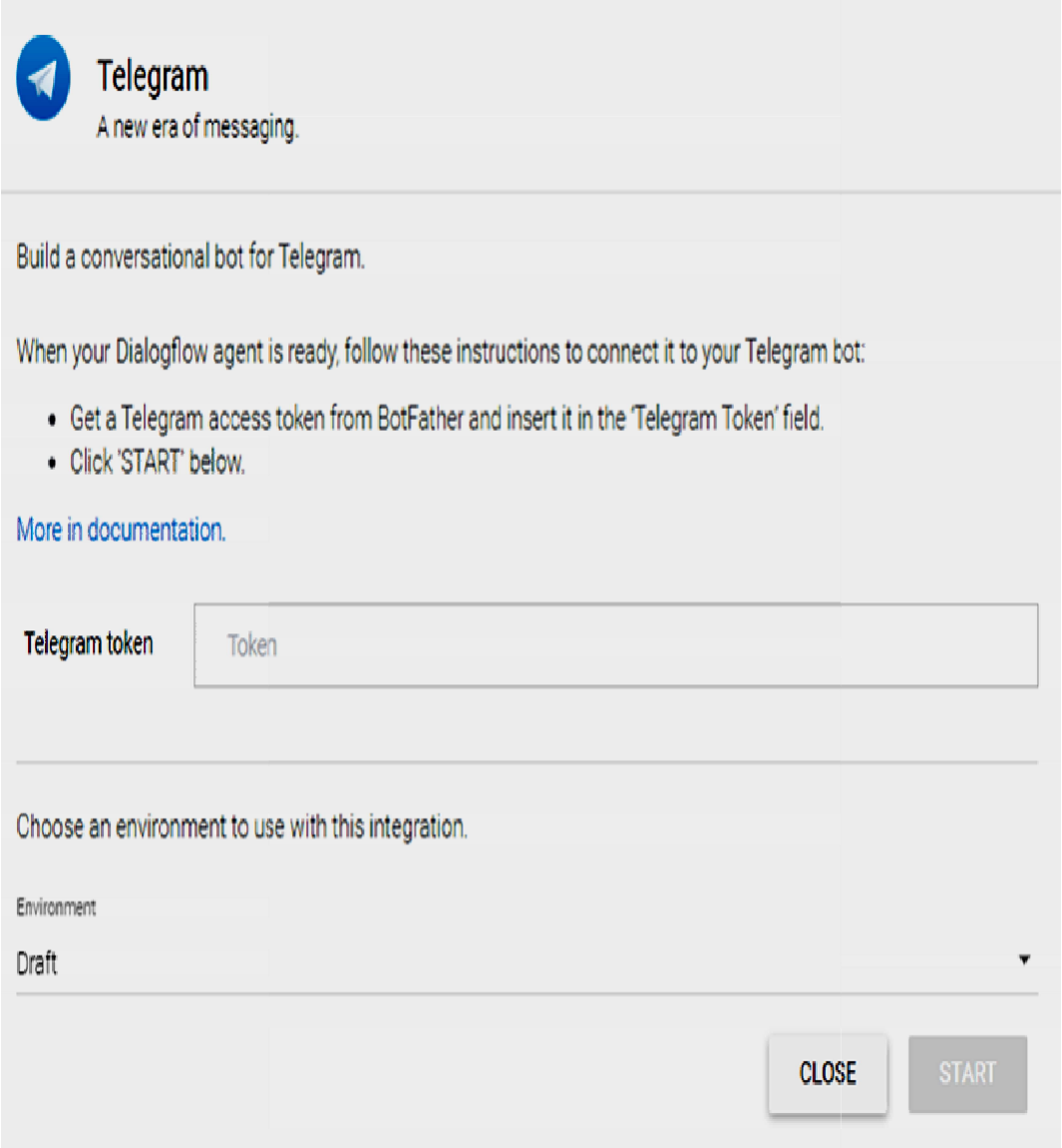


Fig 5.2.1.3 Integrating the chatbot using token

Deployment: This module describes how to deploy virtual student assistant Chatbot on pythonAnywhere webhosting site. Pythonanywhere is an online integrated development environment. It facilitates easy deployment of a Chatbot.

Steps that could be followed to implement a virtual student assistant chatbot are:

First, set up a Django project and create a PostgreSQL database to store the chatbot's data. Use Dialogflow to design and build the chatbot's conversation flow. This includes creating intents (representing the user's desired actions or information) and training the chatbot with example phrases for each intent. Integrate the Dialogflow chatbot

with the Django project using the Dialogflow API and the Django Rest Framework. This will allow the chatbot to communicate with the Django backend and access the database.

Write Python code to handle the chatbot's responses and any actions that the chatbot needs to perform, such as retrieving information from the database or performing calculations. Test the chatbot locally to ensure that it is functioning correctly.

Deploy the Django project to a hosting platform such as PythonAnywhere. This will make the chatbot accessible to users through a web interface or chatbot platform such as Telegram. Continuously monitor and update the chatbot as needed to improve its performance and address any issues that arise.

5.2.2 TECHNOLOGIES USED

PYTHON:

- Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.
- The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third party python modules, programs and tools, and additional documentation.
- The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

FEATURES:

- Easy to code: Python is a high-level programming language
- Free and Open Source
- Object-Oriented Language
- GUI Programming Support
- High-Level Language
- Extensible feature
- Python is Portable language

- Python is Integrated language

Why is Python so popular?

Python is popular for a number of reasons. Here's a deeper look at what makes it so versatile and easy to use for coders.

- It has a simple syntax that mimics natural language, so it's easier to read and understand. This makes it quicker to build projects, and faster to improve on them.
- It's versatile. Python can be used for many different tasks, from web development to machine learning.
- It's beginner friendly, making it popular for entry-level coders.
- Its open source, which means it's free to use and distribute, even for commercial purposes.
- Python's archive of modules and libraries—bundles of code that third-party users have created to expand Python's capabilities—is vast and growing.
- Python has a large and active community that contributes to Python's pool of modules and libraries, and acts as a helpful resource for other programmers. The vast support community means that if coders run into a stumbling block, finding a solution is relatively easy; somebody is bound to have run into the same problem before.

BOT -Telegram Bot Server:

The Bot API is an HTTP-based interface created for developers keen on building bots for Telegram. Bots are third-party applications that run inside Telegram. Users can interact with bots by sending those messages, commands and inline requests. You control your bots using HTTPS requests to Telegram's Bot API.

HOW DO BOTS WORK?

At the core, Telegram Bots are special accounts that do not require an additional phone number to set up. Users can interact with bots in two ways:

- Send messages and commands to bots by opening a chat with them or by adding them to groups.
- Send requests directly from the input field by typing the bot's @username and a query. This allows sending content from inline bots directly into any chat, group or channel.

Messages, commands and requests sent by users are passed to the software running on your servers. Our intermediary server handles all encryption and communication with the Telegram API for you. You communicate with this server via a simple HTTPS-interface that offers a simplified version of the Telegram API. We call that interface our BOTS API.

HOW DO I CREATE A BOT?

There's a bot for that just talk to **BotFather** (described below) and follow a few simple steps. Once you've created a bot and received your authentication token, head down to the Bot API manual to see what you can teach your bot to do.

HOW ARE BOTS DIFFERENT FROM HUMANS?

- Bots have no online status and no last seen timestamps, the interface shows the label 'bot' instead.
- Bots have limited cloud storage — older messages may be removed by the server shortly after they have been processed.
- Bots can't initiate conversations with users. A user must either add them to a group or send them a message first. People can use t.me/ links or username search to find your bot.
- Bot usernames always end in 'bot' (e.g. @TriviaBot, @GitHub_bot).
- When added to a group, bots do not receive all messages by default (see Privacy mode). Bots never eat, sleep or complain (unless expressly programmed otherwise).

THEORY AND HISTORY:

In June 2015, Telegram launched a platform for third-party developers to create bots. Bots are Telegram accounts operated by programs. They can respond to messages or mentions, can be invited into groups, and can be integrated with other programs. Bots can also accept online payments made with credit cards or Apple Pay. The Dutch website *Tweakers* reported that an invited bot can potentially read all group messages when the bot controller changes the access settings silently at a later point in time. Telegram pointed out that it considered implementing a feature that would announce such a status change within the relevant group. There are also inline bots, which can be used from any chat screen. type the bot's username and a query in

- Bots can also handle transactions provided by Payment wall, Yandex. Money, Stripe, Rave pay, Razor pay, Qi Wi and Google Pay for different countries. Bots

also power Telegram's gaming platform, which utilizes HTML5, so games are loaded on-demand as needed, like ordinary webpage's. Games work on iPhones 4 and newer and on Android 4.4 devices and newer.

- In February 2018, Telegram launched their social login feature to its users, named Telegram Login. It features a website widget that could be emb dded into websites, allowing users to sign into a third-party website with their Telegram account. The gateway sends users' Telegram name, username, and profile picture to the website owner, while users' phone number remains hidden. The gateway is integrated with a bot, which is linked with the developer's specific website domain.

5.2.2.1 ARCHITECTURE:

Chat-bot architecture is the heart of chat-bot development. Based on the usability and context of business operations the architecture involved in building a chat-bot changes dramatically. So, based on client requirements we need to alter different elements; but the basic communication flow remains the same.

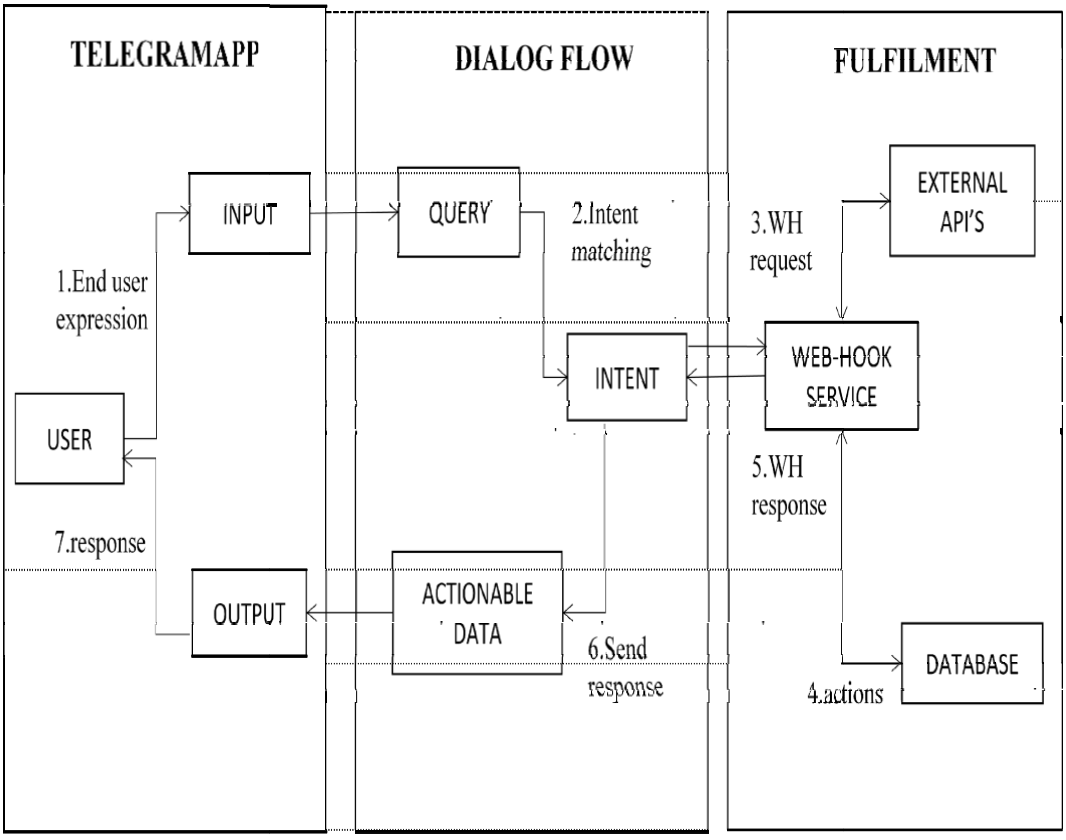


Fig: 5.2.2.1 Architecture

Architecture of a virtual student assistant telegram chatbot developed using dialogflow, django, python language, PostgreSQL and deployed on pythonanywhere.

A virtual student assistant chatbot developed using Dialogflow, Django, Python, and PostgreSQL can be designed as follows:

- **DialogFlow:** DialogFlow is a natural language processing (NLP) platform that can be used to build chatbots that can understand and respond to user queries in a conversational manner. Dialogflow can be integrated with a variety of messaging platforms, including Telegram, to provide a chatbot interface for users.
- **Django:** Django is a web framework written in Python that can be used to build web applications. In this case, Django can be used to build the backend for the chatbot, which will handle the communication between Dialogflow and the database (PostgreSQL).
- **Python:** Python is the programming language used to build the chatbot. It can be used to write the code for the chatbot's logic and functionality, as well as to integrate the chatbot with Dialogflow and Django.
- **PostgreSQL:** PostgreSQL is a relational database management system (RDBMS) that can be used to store and retrieve data for the chatbot. The Django backend can use PostgreSQL to store information about users, courses, assignments, etc.
- **PythonAnywhere:** PythonAnywhere is a hosting service that can be used to deploy and host the chatbot. It provides a web server and a Python environment, allowing the chatbot to run continuously and serve requests from users.

Overall, the virtual student assistant chatbot would work by receiving user input through the Telegram interface, sending the input to Dialogflow for processing, and then using the Django backend to retrieve data from the PostgreSQL database or perform other tasks as needed. The chatbot would then send a response back to the user through the Telegram interface

5.2.2.2 BACKEND TECHNOLOGIES:

- Backend is the server-side of the website. It stores and arranges data, and also makes sure everything on the client-side of the website works fine. It is the part of the website that you cannot see and interact with.
- It is the portion of software that does not come in direct contact with the users. The parts and characteristics developed by backend designers are indirectly accessed by

users through a front-end application. Activities, like writing APIs, creating libraries, and working with system components without user interfaces or even systems of scientific programming, are also included in the backend.

DJANGO FRAMEWORK:

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

HISTORY OF DJANGO

Django was created in 2003 when web developers at the Lawrence Journal-World newspaper started using Python for their web development. After creating a number of websites, they started to factor out and reuse lots of common code and design patterns. That common code led to a generic web development framework that was open-sourced as the "Django" project in 2005. Since the original developers were surrounded by those newspaper writers, well written documentation is a key part of Django. This means that there are excellent references to check out on the official Django documentation pages.

FEATURES OF DJANGO:

Some features that make Django an ideal framework for web application development are as follows:

- **Super-fast:** Django development is extremely fast. Our ideas can take the shape of a product very quickly.
- **Fully loaded:** Django has dozens of projects that can be integrated to carry out common tasks such as user authentication, authorization, and content administration.
- **Versatile:** Django can be used for almost any kind of project, from CMSs to ecommerce apps to on-demand delivery platforms.
- **Secure:** Django also has support to prevent common security issues, including cross site request forgery, cross-site scripting, SQL injection, and clickjacking.
- **Scalable:** Django websites can scale fast to meet high traffic demands.

WHY DO YOU USE DJANGO?

- Django is designed in such a way that encourages developers to develop websites fast, clean and with practical design. Django's practical approach to getting things done is where it stands out from the crowd.
- If you're planning to build a highly customizable app, such as social media website, Django is one of the best frameworks to consider. Django strength lies in its interaction between users or its ability to share different types of media. One of the great advantages of django is its ability to utilize large community-based support which gives you highly customizable third-party ready to use plug-in in your applications.

Below are the top ten reasons to choose Django for web development –

- **Python**: Python is arguably one of the easiest programming languages to learn because of its simple language constructs, flow structure and easy syntax. It is versatile and runs websites; desktop applications and mobile applications embedded in many devices and is used in other applications as a popular scripting language.
- **Built-in admin:** Django has an in-built administration interface which lets you handle your models, user/ group permissions and to manage users. With model interface in place, there is no need for a separate database administration program for all but advanced database functions.
- **Doesn't get in your way:** Creating a Django application adds no boilerplate and no unnecessary functions. There are no mandatory imports, third-party libraries and no XML configuration files.
- **Scalable:** Django is based on MVC design pattern. It means that all the entities like db (database), back-end and front-end code are individual entity. Django allows us to separate code from the static media including pictures, files, CSS and JavaScript that make up your site. Django supports a full list of third-party libraries for web servers, caching, performance management, clustering and balancing. One of the advantages Django provides is the support for major email and messaging applications and services like ReST and OAuth.
- **Battle tested:** Django was first open-sourced in 2005. After 12 years of growth, Django now not only runs news publishing websites but also runs all or part of

major global enterprise like Pinterest, Instagram, Disqus, Bitbucket, EventBrite and Zapier. This makes it a robust and reliable web framework to work with.

- **Huge package support:** Because of its large community support and huge developers network, there is a high possibility that whatever you intend to do might have been done before. Large international community of developers contribute to the community by releasing their projects as open-source packages.

  One such repository of these projects is Django Package site. Currently, Django packages list over 3400 plus reusable Django apps, sites and tools to use in our Django projects.
- **Actively developed:** One of the biggest risks associated with open source project is its sustainability. We cannot be sure if it lasts long. 23 There is no such risk with Django as it is 12 years old. Its consistent releases, newer/better versions and active community is growing every-day with a large core team of voluntary contributors who maintains and improve the code base every-day.
- **Stable releases:** Open-source software projects like Django are, in many cases, actively developed and more secure than competing proprietary software as many developers are developing and testing it every day. However, the drawback of an open-source software project is the absence of a stable codebase to commercially viable development.
- **First Class Documentation:** From the very first release, Django developers made sure that there must be proper comprehensive documents available and the tutorials are easily understood.

BOT TRAINING: DIALOG FLOW:

- Dialog flow is a natural language understanding platform that makes it easy to design and integrate a conversational user interface into your mobile app, web application, device, bot, interactive voice response system, and so on. Using Dialog flow, you can provide new and engaging ways for users to interact with your product.
- Dialog flow can analyse multiple types of input from your customers, including text or audio inputs (like from a phone or voice recording). It can also respond to your customers in a couple of ways, either through text or with synthetic speech.

Key Features:

The process of creating the bot inside Telegram is quick since you can do it by sending messages to a designated "student assistant" account

- Telegram has only a single access token, so the integration steps are fewer
- Telegram supports a few rich responses such as buttons
- Hyperlinks in the text response of Dialog Flow are automatically converted to clickable links inside the Telegram chat window
- Telegram supports multiple responses per message

DATABASE: POSTGRESQL

PostgreSQL is a powerful, open-source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.

Key Features:

Key Features of PostgreSQL are:

PostgreSQL runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows. It supports text, images, sounds, and 25 video, and includes programming interfaces for C / C++, Java, Perl, Python, Ruby, Tcl and Open Database Connectivity (ODBC).

PostgreSQL supports a large part of the SQL standard and offers many modern features including the following –

- Complex SQL queries
- SQL Sub-selects
- Foreign keys
- Trigger
- Views
- Transactions
- Multiversion concurrency control (MVCC)
- Streaming Replication (as of 9.0)
- Hot Standby (as of 9.0)

5.3 EXPLANATION OF KEY FUNCTIONS

A notes dispensing virtual student assistant chatbot using Dialogflow can perform a variety of functions to assist students with their studies and provide access to notes and

other resources. Some key functions of a notes dispensing virtual student assistant chatbot using Dialogflow might include:

- **Providing notes / resources:** The chatbot can be programmed to provide students with access to notes, study guides, and other resources that can help them with their studies. This might include class notes, summaries, and other materials that are relevant to specific courses or subjects
- **Organizing and categorizing notes:** The chatbot can be programmed to organize and categorize notes in a way that is easy for students to access and use. This might involve creating a searchable database of notes or organizing them by subject or course.
- **Updating and adding new notes:** The chatbot can be programmed to regularly update and add new notes to its database, ensuring that students have access to the most current and relevant information.

SAMPLE CODE:

Create tables in database:

```
from django.db import models
# Create your models here.
class Subject(models.Model):

DIFFICULTY_CHOICES = [("easy", "easy"), ("medium", "medium"), ("hard", "hard")]
   name = models.CharField (max_length=200, unique=True)
   difficulty = models.CharField (max_length=50, choices=DIFFICULTY_CHOICES)
   def __str__(self):
      return self.name
   class Meta:
db_table = "subject"

class Document(models.Model):
url = models.URLField()
   subject = models.ForeignKey(
      Subject, on_delete=models.CASCADE, related_name="documents")
```

```
    def __str_(self):
        return self.url

 class Meta:
db_table = "documents”
```

Creating Django URL’S:

```
from bot import views

from django.urls import path

urlpatterns = [

    path("webhook", views.WebHook.as_view()),

]

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

    path("admin/", admin.site.urls),

    path("api/v1/", include("bot.urls")),

]
```

Creating Views:

```
from rest_framework.views import APIView

from rest_framework.response import Response

from bot.models import Subject

import json


class WebHook(APIView):

    def post(self, request):
```

```
        print(json.dumps(request.data, indent=2))
        if request.data["queryResult"]["intent"]["displayName"] == "getListSubjects":
textResponse = []

            for subject in Subject.objects.all():
                    textResponse.append({"text": {"text": [subject.name]}})
            return Response({"fulfillmentMessages": textResponse})
subjectCodes = request.data["queryResult"]["parameters"]["subject"]
responseText = []
        for code in subjectCodes:
            try:
                subject = Subject.objects.get(name=code)
                for document in subject.documents.all():
responseText.append ({"text": {"text": [document.url]}})

            except Exception as e:
responseText.append ({"text": {"text": [str(e)]}})
        return Response({"fulfillmentMessages": responseText}
```

5.4 OUTPUT SCREENS:

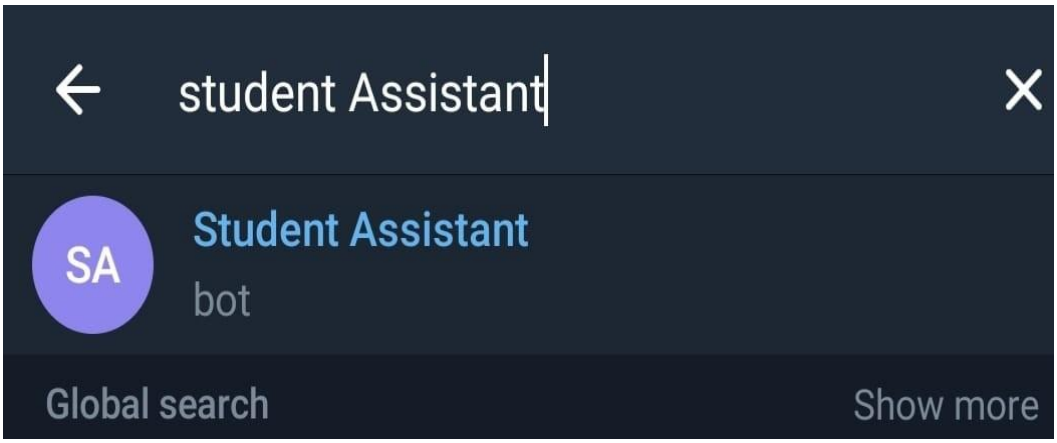


Fig: 5.4.1 Searching Student Assistant bot

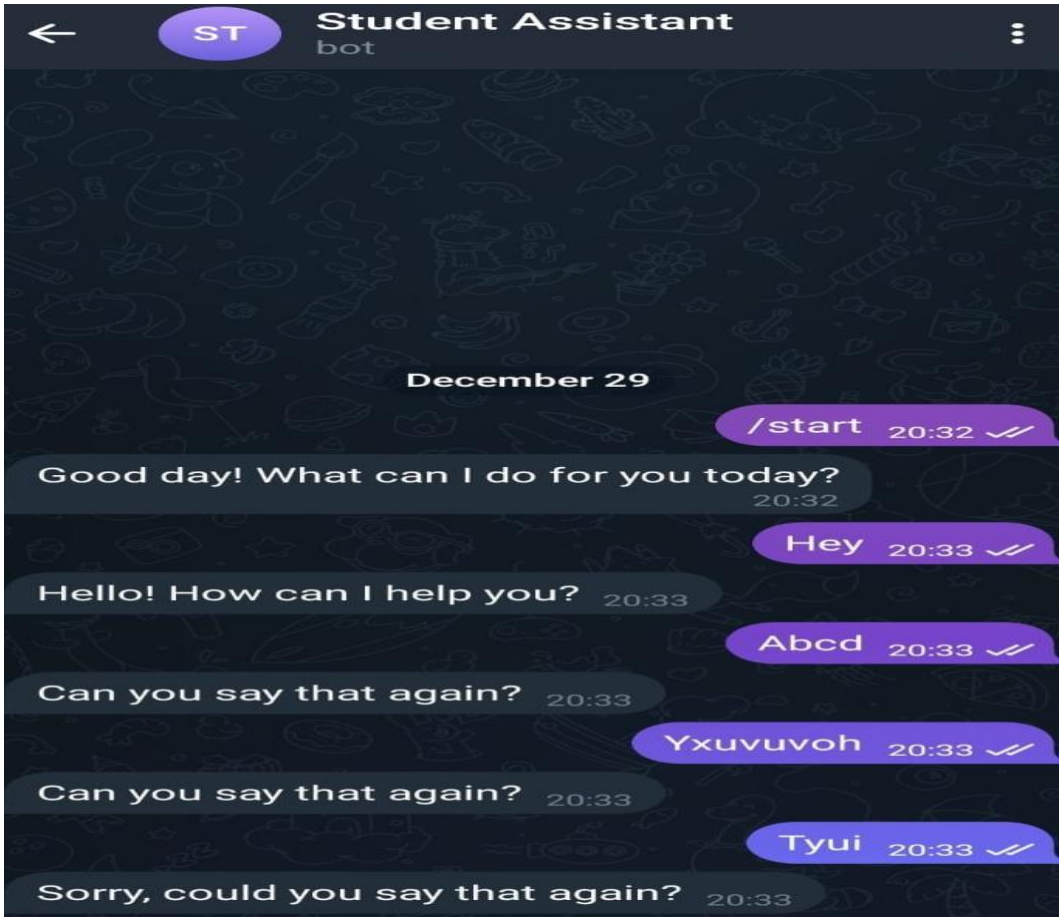


Fig: 5.4.2 Testing the Student Assistant bot

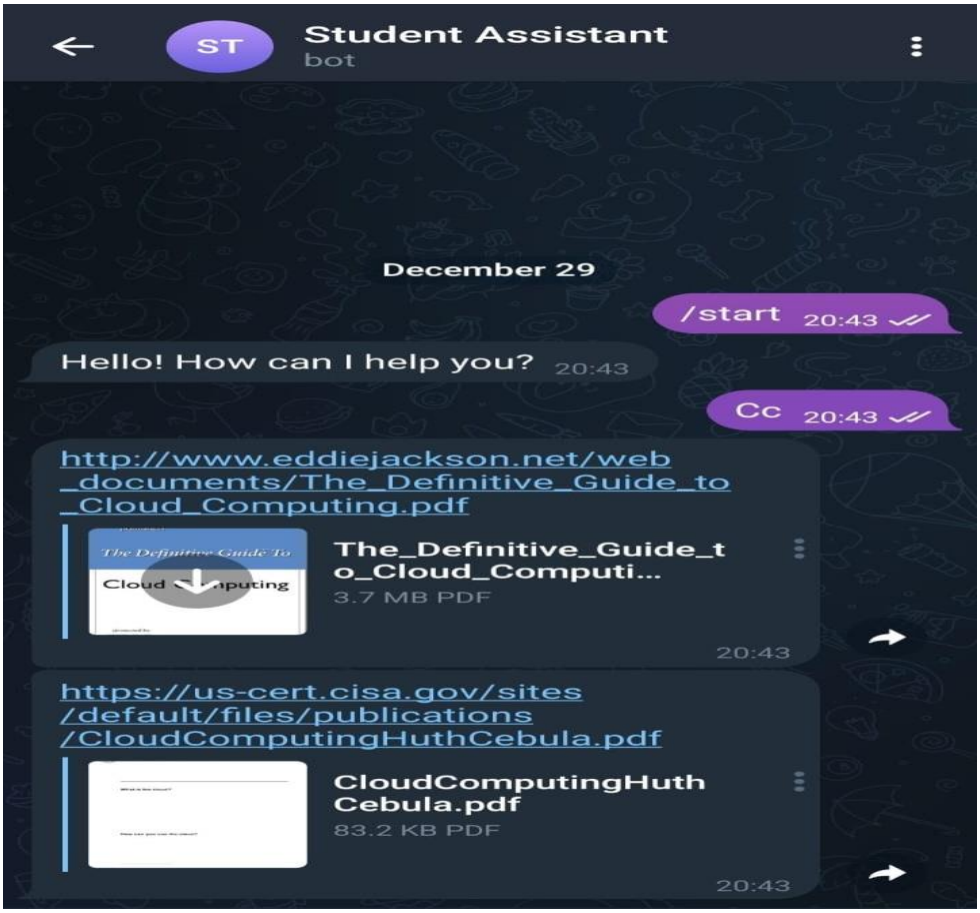


Fig: 5.4.3 Accessing the CC notes

Fig: 5.4.4 Accessing the ML notes

Fig: 5.4.5 Accessing the DM notes

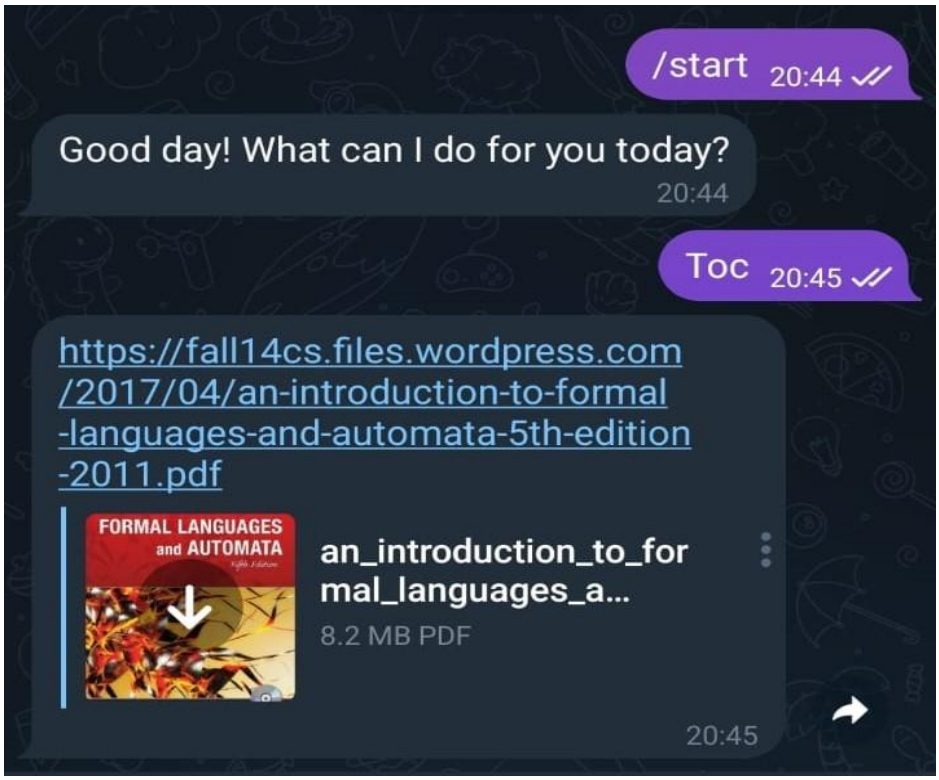


Fig: 5.4.6 Accessing the TOC notes

6. TESTING

Software Testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is defect free; it involves execution of a software component or system component to evaluate one or more properties of interest. Software testing also helps to identify errors, gaps or missing requirements in contrast to the actual requirements. It can be either done manually or using automated tools. Some prefer saying Software testing as a White Box and Black Box Testing.

In simple terms, Software Testing means Verification of Application under Test (AUT). Software testing is a critical element of software quality and assurance and represents ultimate review of specifications, design and coding. Testing is an exposure of the system to trial input to see whether it produces correct output.

The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy and usability. It mainly aims at measuring specification, functionality and performance of a software program or application.

Software Testing can be done in two ways:

1. **Verification:** It refers to the set of tasks that ensure that software correctly implements a specific function.
2. **Validation:** It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

Verification: "Are we building the product, right?"

Validation: "Are we building the right product?"

Importance of Software Testing

The importance of software testing is imperative. Software Testing is important because of the following reasons:

Software Testing points out the defects and errors that were made during the development phases. It looks for any mistake made by the programmer during the implementation phase of the software.

It ensures that the customer finds the organization reliable and their satisfaction in the application is maintained. Sometimes contracts include monetary penalties with respect to the timeline and quality of the product and software testing prevent monetary losses.

It also ensures the Quality of the product. Quality product delivered to the customers helps in gaining their confidence. It makes sure that the software application requires lower maintenance cost and results in more accurate, consistent and reliable results.

Users are not inclined to use software that has bugs. They may not adopt software if they are not happy with the stability of the application. Testing is important for the product to stay in business.

It's important to ensure that the application should not result in any failures because it can be very expensive in the future or in the later stages of the development.

SOFTWARE TESTING TECHNIQUES:

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, designing and coding.

Testing Objective:

- Testing is process of executing a program with the intent of finding an error.
- A good test case design is one that has a probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an as yet undiscovered error.

These above objectives imply a dramatic change in view port.

Testing cannot show the absence of defects, it can only show that software errors are present.

There are three types of testing strategies:

1) Unit test

2) Integration test
3) Performance test

Unit Testing: Unit testing focuses verification efforts on the smallest unit of software design module. The unit test is always white box oriented. The tests that occur as part of unit testing are testing the module interface, examining the local data structures, testing the boundary conditions, execution all the independent paths and testing error-handling paths.

Integration Testing: Integration testing is a systematic technique or construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. Scope of testing summarizes the specific functional, performance, and internal design characteristics that are to be tested. It employs top-down testing and bottom-up testing methods for this case.

Performance Testing: Timing for both read and update transactions should be gathered to determine whether system functions are being performed in an acceptable timeframe.

Test cases:

Table: 6.1 Test cases

TEST CASE ID	TEST CASE SCENARIO	INPUTS	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
1	A little knock or sound on its display	Open the chatbot	A little knock/sound on opening the chatbot	There will be a knock or sound on its display	PASS
2	Taking user inputs as trained	User input or query	The chatbot must take user inputs appropriately	The chatbot takes user inputs appropriately	PASS

3	Verify whether it's showing greeting	User enters /start, hi , hey	Greeting message must be dispensed	Chatbot dispenses greeting message	PASS
4	Verify whether the bot understands user input	User query	User input must be understood properly	User input is understood	PASS
5	Verify whether the bot understands mistakes in user inputs	Inappropriate user input	Mistakes must be identified and must ask for user's confirmation	Identifies mistakes and seeks user's confirmation	PASS
6	Ensuring that the chatbot performs as expected	User query	Chatbot must perform as expected	Chatbot performs as expected	PASS
7	To make sure that the system meets user requirements	User query	Chatbot must meet user requirements by dispensing accurate responses	Chatbot meets user requirements	PASS
8	Ensuring faster delivery of responses	User input or query	Chatbot must quickly respond to user queries	Chatbot delivers quick responses in real time	PASS
9	Ensuring 24/7 availability	upon accessing the chatbot	Chatbot must facilitate expert level assistance with 24/7 availability	Chatbot facilitates 24/7 assistance	PASS

7. CONCLUSION & FUTURE ENHANCEMENT

7.1 PROJECT CONCLUSION

In today's scenario, the lifestyle of people has changed. People now prefer online round-the-clock service for assistance.

A virtual student assistant chatbot can be a useful tool for students to get quick answers to their questions or help with tasks such as notes gathering. Virtual student assistant chatbot is designed to handle a variety of tasks, such as answering questions, offering course materials or providing study notes and resources.

To implement a virtual student assistant chatbot, a combination of tools and technologies such as Dialogflow, PostgreSQL, Django, and Python are used. These tools can be used to design and build the chatbot's conversation flow, integrate the chatbot with a backend system, and handle the chatbot's responses and actions.

Once the chatbot has been developed and tested, it can be deployed to a hosting platform such as PythonAnywhere and made available to users through a web interface or chatbot platform like Telegram.

In conclusion, virtual student assistant chatbots can be a valuable resource for students looking for support and guidance in their academic endeavours. These chatbots can provide students with quick and easy access to information i .e notes.

7.2 FUTURE ENHANCEMENT

In further development stages of project, there is a scope for inserting data for all the departments, training the bot with varied data, testing it on live website, and based on that feedback inserting more training data to the bot. Some of the new features which can be added to the bot are speech recognition feature through which students can ask their queries verbally and get the answers from the bot, integration with multiple web channels and various social media platforms like Skype, Facebook and Twitter.

8. REFERENCES

8.1 Journals

1) Ana Paula Chaves and Marco Aurélio Gerosa How Should My Chatbot Interact? A Survey on Social Characteristics in HumanChatbot Interaction Design Int. J. Hum. Comput. Interact, 37(8), pp. 729-758, 2021
2) Sherwin Fernandes, Rutvij Gawas, Preston Alvares, "Survey on Various Conversational Systems," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE).
3) Neelkumar Patel, Devangi Parikh, Prof. Darshan Patel, Prof. Ronak Patel, "AI and Web-Based Human-Like Interactive University Chatbot (UNIBOT)," Proceedings of the Third International Conference on Electronics Communication and Aerospace Technology
4) Naing Naing Khin, Khin Mar Soe, "University Chatbot"
5) NitirajsinghSandu, Ergun Gide, "Adoption of Chatbots to Enhance Student Learning Experience in Higher Education in India".

8.2 Books

1) Django for APIs: Build web APIs with Python & Django
2) Two Scoops of Django 1.11: Best Practices for the Django Web Framework
3) Django for Beginners: Build Websites with Python and Django

8.3 Sites

1) IEEE Xplorer
2) www.google.com
3) https://dialogflow.cloud.google.com/
4) https://www.djangoproject.com/
5) http://realpython.com/