

**VOICE/TEXTUAL BOT MOBILE APP**  
**A PROJECT REPORT**

*Submitted by,*

**Ms. Aameena Sardar-20221LCS0036**  
**Ms. Lakshmi M-20221LCS0033**  
**Ms. Nirmala r-20211CSE0658**  
**Ms. Soujanya S Badawadagi-20211CSE0636**  
**Ms. Bugga Iswarya-20211CSE0699**

*Under the guidance of,*

**Dr. Abdul Khadar A**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**  
**IN**  
**COMPUTER SCIENCE AND ENGINEERING**

**At**



**PRESIDENCY UNIVERSITY**  
**BENGALURU**  
**JANUARY 2025**

# **PRESIDENCY UNIVERSITY**

## **SCHOOL OF COMPUTER SCIENCE & ENGINEERING**

### **CERTIFICATE**

This is to certify that the Project report “**Voice/Textual Bot Mobile App**” being submitted by Ameena Sardar, Lakshmi M, Nirmala r, Soujanya S Badawadagi, Bugga Iswarya bearing roll number(s) 20221LCS0036, 20221LCS0033, 20211CSE0658, 20211CSE0636, 20211CSE0699 in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a Bonafide work carried out under my supervision.

**Dr.Abdul Khadar A**  
Associate Professor  
School of CSE  
Presidency University

**Dr. Asif Mohammed H.B**  
Associate Professor & HoD  
School of CSE  
Presidency University

**Dr. L. SHAKKEERA**  
Associate Dean  
School of CSE  
Presidency University

**Dr. MYDHILI NAIR**  
Associate Dean  
School of CSE  
Presidency University

**Dr. SAMEERUDDIN KHAN**  
Pro-Vc School of Engineering  
Dean -School of CSE&IS  
Presidency University

**PRESIDENCY UNIVERSITY**  
**SCHOOL OF COMPUTER SCIENCE & ENGINEERING**  
**DECLARATION**

We hereby declare that the work, which is being presented in the project report entitled **Voice/Textual Bot Mobile App** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Dr. Abdul Khadar A , Associate Professor, School of Computer Science & Engineering , Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Sl No	Student Name	Roll Number	Signature
1	Ameena Sardar	20221LCS0036	
2	Lakshmi M	20221LCS0033	
3	Nirmala r	20211CSE0658	
4	Bugga Iswarya	20211CSE0699	
5	Soujanya Badawadagi	20211CSE0636	

## **ABSTRACT**

The Food Chatbot Android Application offers a seamless and interactive way for users to browse, select, and order food using voice commands. The app begins with a simple registration and login process, ensuring user authentication and data security. Once logged in, users can explore a wide range of food options categorized by type, such as appetizers, main courses, desserts, and beverages, complete with descriptions and pricing. The app leverages voice input to make the user experience more intuitive. Users can add items to their cart simply by saying commands like “Add pizza to the cart.” They can also review the cart, which dynamically updates to display the selected items, their quantities, and the total amount. Removing items is just as easy, using either touch or voice commands like “Remove fries from the cart.”

Once the cart is finalized, users can place their orders with a single tap, and the app confirms the order details, providing a hassle-free experience. The application’s user-friendly interface ensures smooth navigation, and the integration of voice commands makes it accessible for users of all tech backgrounds. The Food Chatbot aims to simplify the food ordering process while offering a modern, hands-free approach to managing meals.

## ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Deans **Dr. Shakkeera L** and **Dr. Mydhili Nair**, School of Computer Science Engineering & Information Science, Presidency University, and Dr. **Asif Mohammed H.B**, Head of the Department, School of Computer Science Engineering & Information Science, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Dr. Abdul Khadar A** , **Associate Professor** and Reviewer **Dr. Shanmugarathinam** , **Professor SOCSE**, School of Computer Science Engineering & Information Science, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the PIP2001 Capstone Project Coordinators **Dr. Sampath A K**, **Dr. Abdul Khadar A** and **Mr. Md Zia Ur Rahman**, department Project Coordinators **Dr. Abdul Khadar A** and Git hub coordinator **Mr. Muthuraj**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

**Ameena Sardar**  
**Lakshmi M**  
**Nirmala r**  
**Soujanya S Badawadagi**  
**Bugga Iswarya**

## LIST OF FIGURES

Sl. No.	Figure Name	Caption	Page No.
1	Figure 4.1	BASIC NOTATIONS	16
	Figure 4.2	LEVEL 0 DFD	17
	Figure 4.3	USER LEVEL 1 DFD	18
	Figure 4.4	USER USE CASE DIAGRAM	19
	Figure 4.5	USER SEQUENCE DIAGRAM	21
	Figure 4.6	INITIAL ACTIVITY	22
	Figure 4.7	FINAL ACTIVITY	22
	Figure 4.8	ACTIVITY	22
	Figure 4.9	DECISIONS	22
	Figure 4.10	WORKFLOW	23
	Figure 4.11	USER ACTIVITY DIAGRAM	24
	Figure 8.1	GANTT CHART	35
	Figure B.1	HOME PAGE	48
	Figure B.2	LOGIN PAGE	49
	Figure B.3	REGISTRATION PAGE	50
	Figure B.4	MENU	51
	Figure C.1	SDG	52

## **TABLE OF CONTENTS**

<b>CHAP TER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	<b>ABSTRACT</b>	iv
	<b>ACKNOWLEDGMENT</b>	v
<b>1.</b>	<b>INTRODUCTION</b>	1
	1.1PROJECT DESCRIPTION	1
	1.2 MOTIVATION	1
	1.3 PROBLEM STATEMENT	1
	1.4 SCOPE	2
	1.5 OBJECTIVES	2
<b>2.</b>	<b>LITERATURE SURVEY</b>	3
	2.1 INTRODUCTION	3
	2.2 PROPOSED SYSTEM	6
	2.3 TOOLS AND TECHNOLOGIES USED	7
<b>3.</b>	<b>SOFTWARE REQUIREMENT SPECIFICATION</b>	11
	3.1 FUNCTIONAL REQUIREMENTS	11
	3.2 NON-FUNCTIONAL REQUIREMENTS	12
	3.3 HARDWARE AND SOFTWARE REQUIREMENTS	14
<b>4.</b>	<b>SYSTEM DESIGN</b>	15
	4.1 DATA FLOW DIAGRAM	15
	4.2 USE CASE DIAGRAM	18

	4.3 SEQUENCE DIAGRAM	20
	4.4 ACTIVITY DIAGRAMS	22
	4.5 METHODOLOGY	24
<b>5.</b>	<b>SYSTEM IMPLEMENTATION</b>	29
	5.1 INTRODUCTION	29
	5.2 PHASE - IN METHOD OF IMPLEMENTATION	29
<b>6.</b>	<b>OUTCOMES</b>	32
<b>7.</b>	<b>CONCLUSION AND FUTURE ENHANCEMENT</b>	33
<b>8.</b>	<b>TIMELINE FOR EXECUTION OF PROJECT</b>	35
	<b>REFERENCES</b>	36
<b>A</b>	<b>APPENDIX - PSUEDOCODE</b>	37
<b>B</b>	<b>APPENDIX - SCREENSHOTS</b>	48
<b>C</b>	<b>APPENDIX - ENCLOSURES</b>	52



# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 PROJECT DESCRIPTION**

The Food Chatbot Android Application revolutionizes the way users interact with food ordering systems by incorporating voice recognition technology for a hands-free experience. Designed with user convenience in mind, the app offers a simple and intuitive interface, allowing users to browse through a diverse menu, add items to their cart, and place orders using voice commands.

Registration and login ensure that only authorized users can access the app's features, maintaining privacy and security. Once logged in, users are greeted with a categorized menu that displays detailed information about available food items, such as descriptions and prices, making selection easy and informed. The standout feature of the app is its voice input functionality. Users can perform actions like adding or removing items from their cart using natural language commands, eliminating the need for manual interaction. The cart dynamically updates to show the selected items, their quantities, and the total cost, providing a clear and organized view of the order.

By merging traditional food ordering with modern technology, the Food Chatbot Android Application aims to enhance the overall user experience. Whether you're tech-savvy or new to mobile apps, this solution offers a fast, efficient, and user-friendly way to order meals on the go.

### **1.2 MOTIVATION**

The motivation behind the Food Chatbot Android Application stems from the need to simplify and modernize the food ordering process. With busy lifestyles and increasing reliance on technology, users seek faster, more efficient ways to complete daily tasks. Voice recognition technology offers a hands-free, accessible solution that caters to all, including individuals with limited mobility or tech experience. By integrating voice commands into food ordering, the app not only saves time but also enhances convenience and user engagement. This project is driven by the goal of creating a seamless, intuitive experience that blends practicality with the latest technological advancements.

### **1.3 PROBLEM STATEMENT**

In today's fast-paced world, traditional methods of ordering food through apps can be time-consuming and cumbersome, especially for users who are not tech-savvy or have physical limitations. Many food delivery applications rely heavily on manual navigation and input, which may not always provide an intuitive or user-friendly experience. Additionally, the lack of innovative features like voice input often limits accessibility and convenience for users seeking a quick, hands-free solution.

This project addresses the gap by creating a Food Chatbot Android Application that integrates voice recognition technology, enabling users to browse menus, add or remove items, and place orders using simple voice commands. By simplifying the interaction process and reducing dependency on manual inputs, the application aims to provide a modern, efficient, and inclusive solution for food ordering. The focus is on enhancing user satisfaction by combining convenience, accessibility, and technological innovation in a single platform.

### **1.4 SCOPE**

The scope of the Food Chatbot Android Application encompasses creating a seamless platform for food ordering with an emphasis on accessibility and convenience. The app allows users to browse food menus, add or remove items, and place orders using voice commands, making it suitable for a wide range of users, including those with physical or technological limitations. Key features include user registration, login, dynamic cart management, and real-time voice interaction for efficient order processing. This project aims to modernize the food ordering experience by combining intuitive design with advanced voice recognition technology, ensuring a user-friendly and inclusive solution for all.

### **1.5 OBJECTIVES**

- Simplify the food ordering process through an intuitive Android application.
- Provide a categorized menu with detailed food item descriptions and prices.
- Allow users to manage their cart dynamically, including adding and removing items via voice commands.
- Ensure secure access through user registration and login functionalities.
- Enhance accessibility and convenience for users of all technological backgrounds.
- Deliver a modern, efficient, and user-friendly food ordering experience.

## **CHAPTER 2**

### **SURVEY OF LITERATURE**

#### **2.1 OVERVIEW**

The survey procedure is necessary for every software development. To obtain the software requirements, the survey procedure is required. Studying the current system and the tools required for software development are also included in the survey. Having a thorough understanding of the tools is crucial. An excerpt of the data gathered during the literature survey is provided below. A literature review is a research process used to identify issues with the current system and suggest solutions for its problems.

##### **1. Voice-Enabled Food Ordering Systems**

This paper explores voice-enabled interfaces in food ordering applications, emphasizing their potential to improve user accessibility and convenience. The study discusses various speech-to-text models and evaluates their accuracy and usability. Results show that integrating voice technology significantly reduces task completion time and enhances user satisfaction, especially for non-tech-savvy individuals.

##### **2. Speech Recognition in Mobile Applications**

The study focuses on the implementation of speech recognition technology in mobile apps. It highlights how voice inputs can replace manual navigation for a more intuitive experience. The research evaluates various algorithms like Google Speech API and CMU Sphinx, concluding that speech-based apps can improve interaction speed and accessibility.

##### **3. Enhancing User Experience with AI Chatbots**

This paper examines the use of AI-powered chatbots for customer service, including food ordering. It highlights how natural language processing (NLP) improves user interaction and provides personalized recommendations. Findings suggest that chatbots significantly boost engagement by understanding user preferences and simplifying the ordering process.

#### **4. Accessibility in Mobile Applications**

This study discusses the role of accessibility features in mobile apps, particularly voice-based interactions. It emphasizes the importance of inclusive design for individuals with disabilities. The research showcases voice interfaces as an effective solution for bridging the accessibility gap in daily-use applications.

#### **5. Cart Management in E-commerce Systems**

This paper focuses on cart functionalities in e-commerce platforms, discussing how dynamic updates and user-friendly design impact customer satisfaction. The research highlights the growing demand for interactive cart systems and suggests integrating voice commands for a smoother user experience.

#### **6. Adoption of Conversational Interfaces in Mobile Apps**

The study analyzes user adoption rates of conversational interfaces in mobile applications. It emphasizes the need for intuitive design and accurate voice recognition. Results indicate that conversational systems improve user retention and satisfaction by reducing cognitive load during app navigation.

#### **7. Human-Computer Interaction in Voice-Enabled Applications**

This research delves into the effectiveness of human-computer interaction models in voice-enabled applications. It evaluates usability metrics and suggests improvements in speech processing algorithms to make interactions more natural and user-friendly. The study highlights their potential in food ordering apps.

#### **8. Integrating Voice Assistants in Retail Systems**

The paper discusses the integration of voice assistants into retail applications, including food delivery. It explores the challenges of understanding diverse accents and handling complex commands. The study concludes that advanced NLP models improve command recognition and simplify task execution.

## **9. Speech-to-Text Accuracy in Mobile Platforms**

This paper evaluates the accuracy of speech-to-text systems across mobile platforms. It highlights challenges like background noise and accents but concludes that modern systems, such as Google's ASR, achieve over 90% accuracy, making them ideal for voice-enabled applications like food ordering.

## **10. User Behavior in Voice-Activated Applications**

The study investigates user behavior and acceptance of voice-activated applications. It reveals that users value speed, simplicity, and hands-free convenience. The research emphasizes designing voice-enabled systems that are responsive and adaptive to user preferences for maximum satisfaction.

## **RESEARCH GAPS IN EXISTING METHODS**

### **Limited Accuracy of Voice Recognition Systems**

Existing voice-enabled applications often struggle with understanding diverse accents, dialects, and speech patterns, leading to user frustration.

### **Accessibility Challenges**

Many food ordering apps lack robust accessibility features tailored for users with disabilities, limiting inclusivity.

### **Inefficient Error Handling**

Current systems fail to provide effective mechanisms for handling misinterpreted voice commands, causing delays in the ordering process.

### **Limited Integration of Dynamic Cart Features**

Few existing apps support seamless cart management using voice commands, such as updating quantities or removing items.

### **Background Noise Sensitivity**

Voice recognition systems often perform poorly in noisy environments, reducing reliability and user satisfaction.

### **High Dependency on Internet Connectivity**

Many solutions require constant internet connectivity for voice processing, making them less effective in low-network areas.

### **Suboptimal User Experience**

The interface design in some applications does not prioritize ease of use, making navigation cumbersome for non-tech-savvy users.

## **2.2 PROPOSED SYSTEM**

The proposed Food Chatbot Android Application aims to address the limitations of current food ordering systems by integrating voice recognition technology for a more intuitive, hands-free user experience. Unlike traditional apps that require manual navigation, this system allows users to browse the menu, add items to the cart, remove them, and place orders using simple voice commands. By leveraging speech-to-text technology, the system provides users with a seamless and accessible interface, particularly benefiting individuals with physical limitations or those unfamiliar with complex app interfaces.

The application ensures secure access with a registration and login system, so only authorized users can place orders and manage their preferences. The menu is categorized for easy navigation, displaying food items with detailed descriptions and prices. Voice commands can be used to interact with the cart, where users can dynamically update item quantities or remove unwanted items. The cart automatically updates to show the total cost in real-time.

Additionally, the system will incorporate advanced error-handling mechanisms to reduce the likelihood of misinterpreted voice commands, ensuring a smooth and reliable experience. The app will be designed with accessibility in mind, featuring clear voice prompts and simple navigation for users of all backgrounds and tech familiarity. By combining voice recognition, dynamic cart management, and a secure login system, the proposed Food Chatbot Android Application seeks to provide a modern, efficient, and user-friendly solution to online food ordering, enhancing both convenience and accessibility for all users.

## **ADVANTAGES**

### **Hands-Free Interaction**

The integration of voice recognition allows users to browse menus, add/remove items, and place orders without needing to manually interact with the app, enhancing convenience and accessibility.

### **Improved Accessibility**

The app is designed with accessibility in mind, making it ideal for users with physical disabilities or those who may find traditional interfaces difficult to navigate.

### **Enhanced User Experience**

Voice commands simplify the ordering process, making it faster and more intuitive, especially for non-tech-savvy users.

### **Real-Time Cart Management**

Users can update their cart in real-time, adding or removing items and seeing the total cost immediately, improving the shopping experience.

### **Secure User Authentication**

The registration and login system ensures that only authorized users can access personalized features, adding a layer of security to the app.

## **2.3 TOOLS AND TECHNOLOGIES USED**

### **OVERVIEW OF ANDROID STUDIO**

Android Studio is the official integrated development environment (IDE) for building Android applications. Developed by Google, it offers a comprehensive set of tools designed to help developers create, test, and deploy Android apps with ease. Android Studio is based on IntelliJ IDEA, a popular Java IDE, and is tailored specifically for Android development.

One of the key features of Android Studio is its powerful code editor, which supports various programming languages like Java, Kotlin, and XML. It offers features like code completion, syntax highlighting, and error checking to make writing code easier and more efficient. The IDE also includes a robust layout editor, which enables developers to design user interfaces visually by dragging and dropping elements, as well as a preview feature to see how the design looks on different screen sizes and resolutions.

Android Studio also comes with an emulator for testing apps on virtual devices, allowing developers to simulate different Android environments. Additionally, it integrates tools like Firebase and Google Cloud for adding backend functionalities, such as user authentication and cloud storage.

Another significant advantage of Android Studio is its ability to handle both the frontend and backend aspects of app development in one place. The IDE streamlines the build process with features like Gradle integration, making it easier to manage dependencies and automate tasks. Overall, Android Studio provides a robust and user-friendly environment for Android app development, with everything a developer needs to create high-quality, performant apps for the Android platform.

## **SQLite**

SQLite is a lightweight, serverless, self-contained relational database management system (RDBMS) that is widely used in mobile applications, desktop software, and embedded systems. It is an open-source database engine that is highly efficient and requires minimal setup and maintenance, making it a popular choice for developers working on small-to-medium-scale projects.

One of the main advantages of SQLite is that it operates without the need for a separate database server. All the database files are stored locally on the device or system, which allows the application to function independently of external databases or network connections. This makes SQLite ideal for mobile applications like Android and iOS, where data needs to be stored locally for offline use.

SQLite supports standard SQL syntax, which makes it easy for developers familiar with relational databases to work with it. It offers features like tables, views, indexes, and triggers, allowing for complex queries and efficient data manipulation. However, unlike full-fledged



database management systems, SQLite is designed to be lightweight and does not support advanced features like stored procedures or user management.

In Android development, SQLite is commonly used to store and manage structured data, such as user preferences, app settings, and local content. Android provides a built-in SQLite database engine and API, making it easy for developers to integrate SQLite into their applications. Overall, SQLite provides a simple, efficient, and reliable solution for local data storage, with a minimal overhead, making it a great choice for mobile and embedded application development.

## **JAVA**

Java is a versatile, high-level, object-oriented programming language that was first developed by Sun Microsystems in 1995 and is now owned by Oracle Corporation. It is widely used for building a variety of applications, from mobile apps and web applications to large-scale enterprise systems. Java is known for its portability, scalability, and robust security features.

One of Java's defining features is its "Write Once, Run Anywhere" philosophy, made possible by the Java Virtual Machine (JVM). The JVM allows Java code to be compiled into bytecode, which can then be executed on any device or operating system that has a JVM installed, making Java applications highly portable. This makes Java an excellent choice for cross-platform development, especially for web-based and mobile applications.

Java is an object-oriented language, meaning it organizes software design around objects, which are instances of classes. It supports key concepts like inheritance, polymorphism, encapsulation, and abstraction, which help in creating modular, reusable, and maintainable code. It also has a rich standard library (the Java API) that provides a wide range of built-in functionality, including utilities for networking, database interaction, file handling, and graphical user interface (GUI) development.

In mobile app development, Java is the primary language used for Android app development. Android Studio, the official IDE for Android development, supports Java and Kotlin as primary languages, though Java remains widely used due to its long-standing presence in the Android ecosystem. Overall, Java's platform independence, scalability, and extensive community support make it one of the most popular and enduring programming languages in the world.

## **XML**

XML (Extensible Markup Language) is a flexible, text-based markup language that is used to store and transport data. It is designed to be both human-readable and machine-readable, making it an ideal choice for data exchange across different systems and platforms. XML is widely used in web services, configuration files, data storage, and document formatting.

One of the key features of XML is its ability to define custom tags, which makes it highly adaptable to various use cases. Unlike HTML, which has predefined tags for displaying content in web browsers, XML allows users to create their own tags that describe the data in a meaningful way. For example, in an e-commerce system, XML tags might include `<product>`, `<price>`, and `<description>`, allowing developers to organize and store product information in a structured manner.

XML is hierarchical in nature, using a tree structure that organizes data into elements, attributes, and values. Each element is enclosed in opening and closing tags, and elements can be nested to represent complex relationships. This structure makes XML a great choice for representing hierarchical data such as document outlines, organizational structures, and product catalogs.

In Android development, XML is primarily used for defining user interfaces. Layouts for Android apps, such as buttons, text fields, and images, are typically described in XML files. Android Studio uses XML files to define the structure and appearance of the app's UI, separating the design from the underlying Java or Kotlin code. Overall, XML is a powerful, flexible language that plays a crucial role in data representation, storage, and communication across different platforms and applications.

## **CHAPTER 3**

### **SOFTWARE REQUIREMENTS SPECIFICATION**

A crucial piece of information that influences how the software development process is established is the Software Requirement Specification (SRS). SRS lists a framework's requirements and includes a representation of its key elements. At this point, the system's users—rather than its solutions—are the main focus. The requirement specification document's output outlines the software's purpose as well as the desired system's characteristics and limitations. Customers' and designers' understanding of the content of the product that will be produced is embodied in SRS. Since SRS makes a huge commitment to the overall development plan, it should be exact and fully represent the framework requirements.

SRS (Software Requirement Specification) is among the most important pieces of information. It provides comprehensive details on how to set up a software development process. It contains the representation of the key elements and documents the essential requirements of the framework.

#### **3.1 FUNCTIONAL REQUIREMENTS**

##### **User Registration and Login**

The system must allow users to register and log in using their credentials (username, email, and password). Registered users can access personalized features, while unauthorized users will be prompted to sign up or log in.

##### **Voice Command Integration**

The application should support voice commands to allow users to interact with the system. Users should be able to browse the food menu, add items to their cart, and remove items using voice inputs.

##### **Menu Display**

The app must display a categorized menu with food items, descriptions, and prices. Users should be able to search for food items or navigate through categories using voice commands.

### **Cart Management**

The system should allow users to manage their cart by adding and removing food items through voice commands. It should display the current items in the cart and update the total price in real time.

### **Real-Time Price Calculation**

The application must automatically calculate the total price of the cart and display it to the user whenever items are added or removed.

### **User Feedback for Commands**

The system should provide real-time voice or visual feedback to the user for successful or unsuccessful voice commands, such as confirming that an item has been added or removed from the cart.

### **User Session Management**

The application should manage user sessions by keeping users logged in until they choose to log out or their session expires.

### **Voice Command Error Handling**

The system should be able to handle errors caused by unrecognized or misunderstood voice commands and prompt the user for clarification.

## **3.2 NON-FUNCTIONAL REQUIREMENTS**

### **Performance**

The application should respond to voice commands within 2-3 seconds to ensure a smooth and efficient user experience. The cart should update in real time, reflecting any changes made by the user.

### **Scalability**

The system should be able to handle a large number of users concurrently without performance degradation. It should be scalable to accommodate future growth, such as adding more food items, categories, or even expanding to multiple locations.

### **Availability**

The app should be available 24/7, with minimal downtime for maintenance. It must be able to handle occasional peak loads, such as during peak ordering hours or promotional events.

### **Usability**

The application should have an intuitive and user-friendly interface that is easy to navigate, even for users with limited technical knowledge. The voice interface should be simple and clear, ensuring ease of use for a wide range of users.

### **Compatibility**

The app must be compatible with a wide range of Android devices and screen sizes. It should work across different Android versions, starting from a specified minimum version (e.g., Android 5.0 and above).

### **Reliability**

The system should have high reliability, meaning it should be free from crashes or major bugs. It should be capable of handling voice input in various environmental conditions, such as background noise.

### **3.3 Requirements for hardware and software**

#### **MINIMUM HARDWARE NEEDS**

- Intel i5 2.53GHz processor
- 30GB hard drive
- 8 GB or more RAM;

#### **SOFTWARE REQUIREMENTS**

- System of operation: Windows 8 and later
- FrontEnd : XML
- BackEnd : Java
- IDE : Android Studio

## **CHAPTER 4**

### **DESIGN OF THE SYSTEM**

After the system design phase is over and the system design has been approved by the review, detailed design can begin. Developing the internal logic of each module discovered during system design is the aim of this phase. Module identification is the main focus of system design, while module logic design is the main focus of detailed design. To put it another way, the focus of system design is on the components that are required, but the problem in detailed design is how the components can be integrated into the program.

#### **4.1 Diagram of Data Flow**

The functions, or processes, that collect, process, store, and distribute data between a system and its environment as well as between system components are visually represented by DFD. It is an effective tool for communication between the system designer and the user because of its visual depiction. A broad overview can be started and expanded to a hierarchy of specific diagrams thanks to the DFD structure. DFD has often been used due to the following reasons:

- Logical information flow of the system
- Determination of physical system construction requirements
- Simplicity of notation
- Establishment of manual and automated systems requirements

**Basic Notation:**

Fig:4.1 Basic Notations

Any procedure that modifies data and generates an output is referred to as a notation process. It could carry out calculations, sort data logically, or control data flow according to business standards. The process is described with a simple label, like "Submit payment."

Data stores are files or repositories, such membership forms or database tables, that contain information for later use. Every data storage is given a straightforward label, like "Orders."

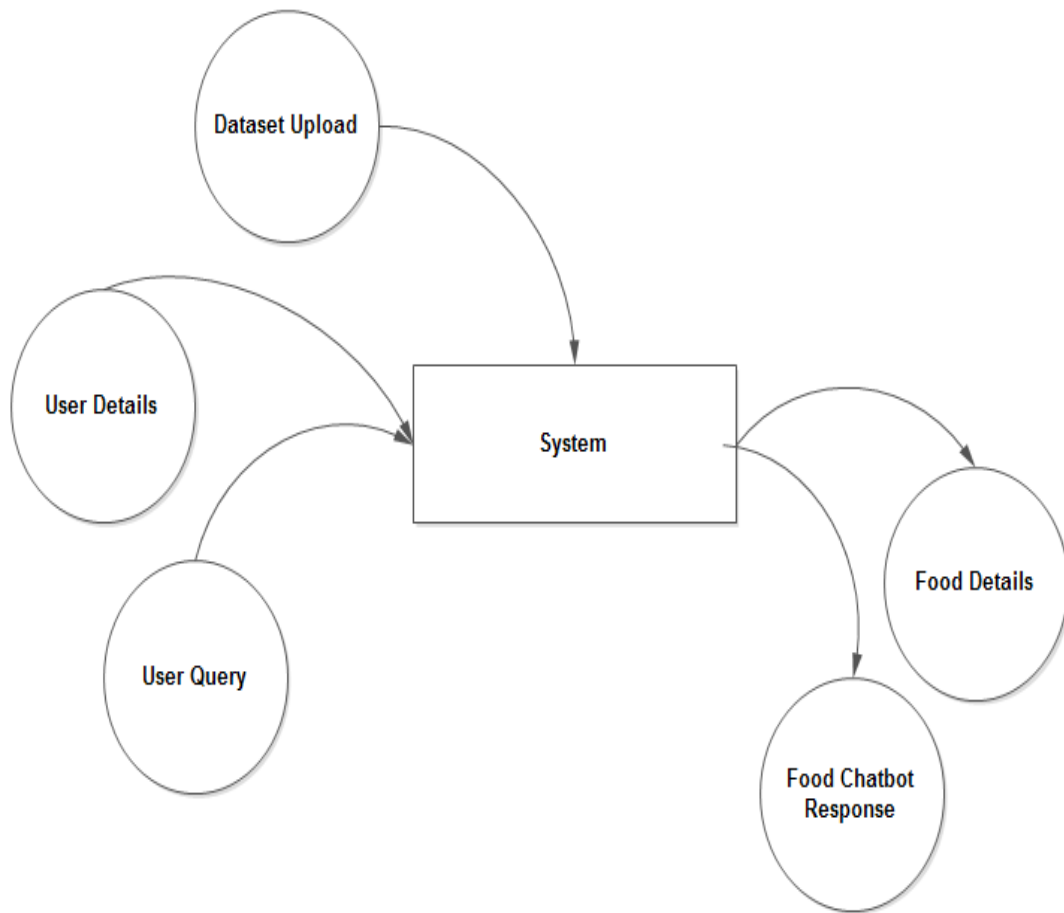
An external entity is a system that communicates with the system being diagrammed by sending or receiving data. Information that enters or exits the system comes from these sources and ends up there. They could be a business system, a computer system, or an external entity. Other names for them include actors, terminators, sources, and sinks. Usually, they are depicted on the diagram's edges.

The path that data takes between external entities, processes, and data repositories is known as data flow. It uses arrows to depict the interface between the other components and is usually labeled with a brief data name, such as "Billing details."

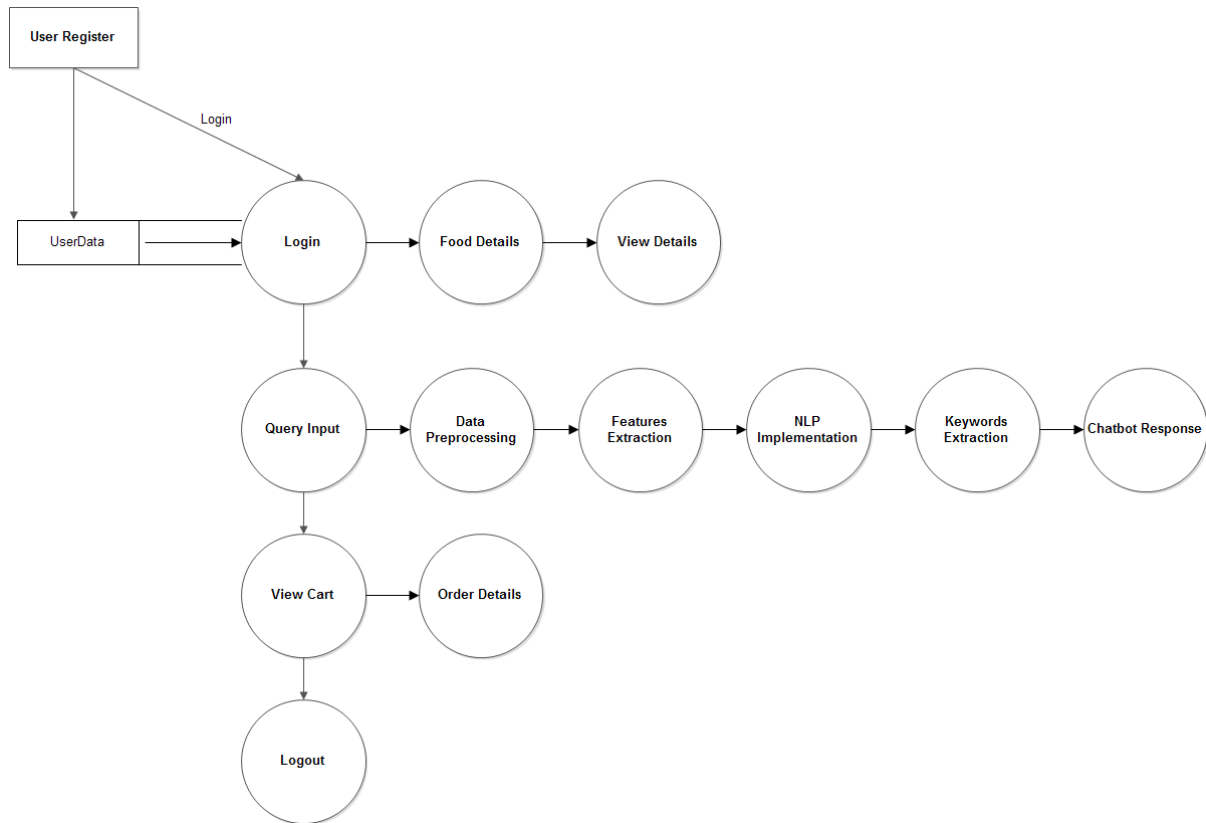
.



## Data Flow Diagram



**Fig:4.2 level 0 DFD**

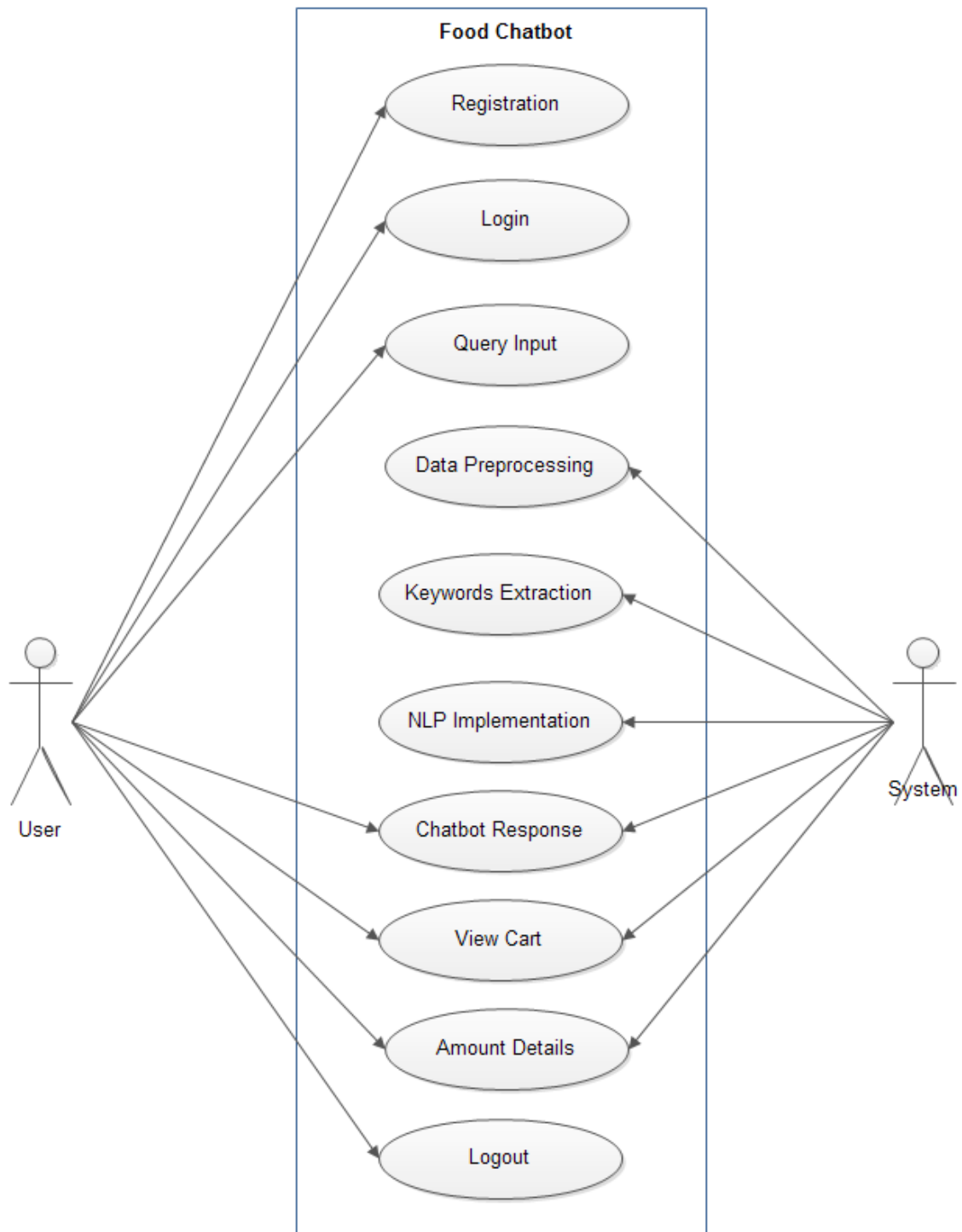


**Fig:4.3 User level 1 DFD**

#### 4.2 Diagram of the use case

A graph of actors, a collection of use cases surrounded by a system boundary, and the communication links between the actor and the use case make up a use case diagram. Each use case is a piece of functionality that a system offers its users, and the use case diagram illustrates how a system communicates with external actors. An actor is depicted as a stick figure with their name beneath it, while a use case is represented by an ellipse with the use case name.

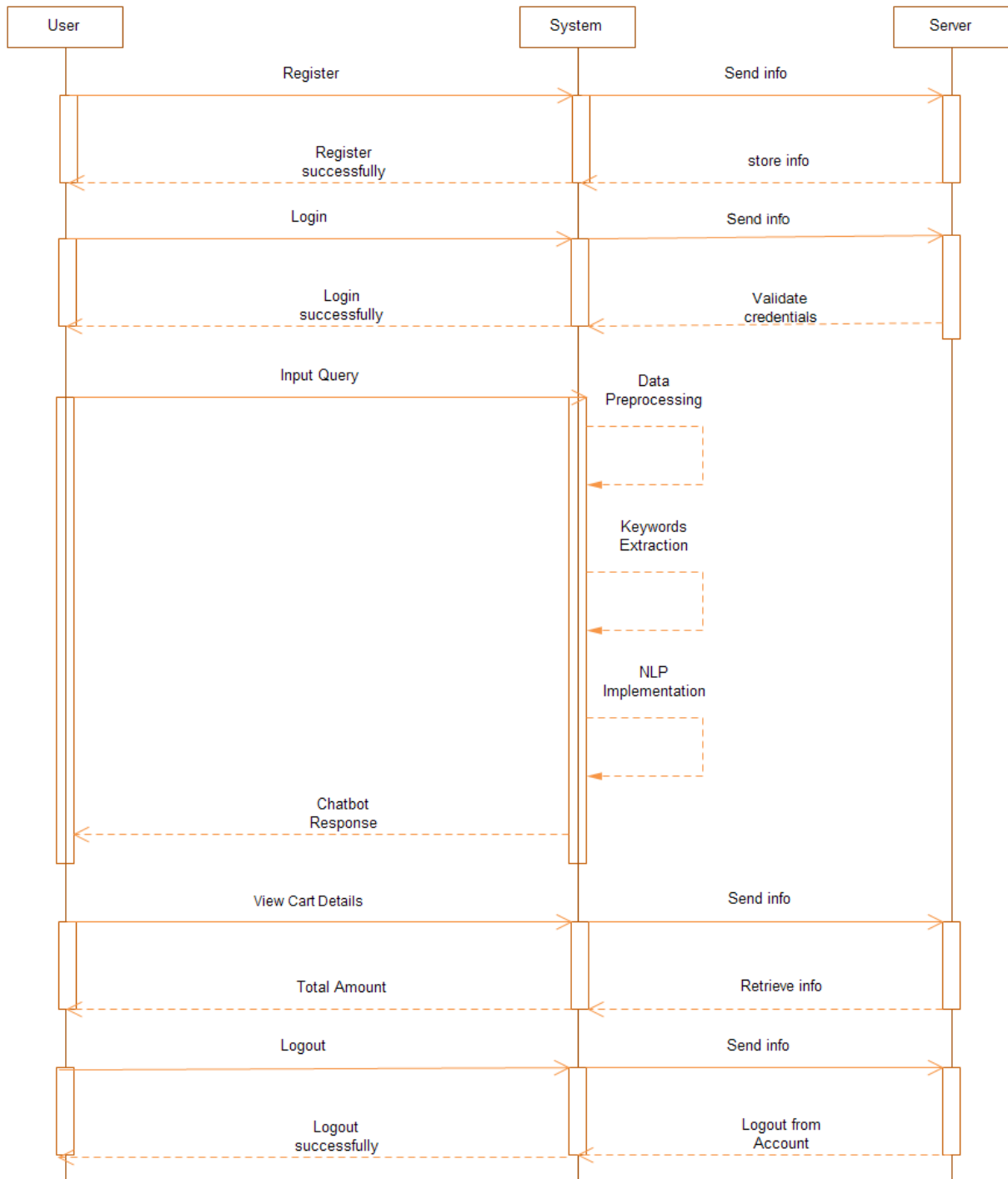
Use cases are used to separate and identify system functionality during a project's analysis phase. They divide the system into use cases and actors. The roles that system users play are portrayed by actors. These users may be other people, computers, devices, or even software programs.

**Diagram of the user use case****Fig:4.4 Use case diagram**

### **4.3 Diagram of Sequence**

A sequence diagram displays the interactions between objects in chronological order. It shows the classes and objects in the scenario as well as the messages that must be sent back and forth between the objects in order for the scenario to function. Event diagrams and event scenarios are other names for sequence diagrams.

The flow of events, messages, and actions between a system's objects or components can be represented or modeled using UML sequence diagrams. The header elements, which are shown horizontally at the top of the diagram, interact in a sequence that is represented by time in the vertical direction. Order Diagrams are mostly used to design, document, and validate the system's architecture, interfaces, and logic by outlining the steps that must be taken in order to finish a task or scenario. UML sequence diagrams offer a dynamic perspective on the behavior of the system, making them valuable design tools.

**Diagram of Sequence****Fig:4.5 Sequence Diagram**

#### 4.4 Activity Schematics

Activity diagrams show a system's operational and business processes. An activity diagram is a dynamic diagram that displays the event and activity that led to the object's current state. It is a straightforward and understandable example of what occurs in a workflow, what tasks can be completed concurrently, and whether there are other routes through the workflow.

##### Simple Notations



**Fig:4.6**

First Action

This displays the flow's initial activity or beginning point. A solid circle is used to represent it.



**Fig:4.7**

Last Task

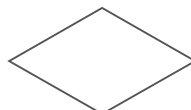
A bull's eye symbol, often known as a final activity, indicates the finish of the activity diagram.



**Fig:4.8**

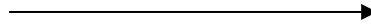
Action

Symbolized by a rectangle with edges that are rounded, almost oval.



**Fig:4.9**

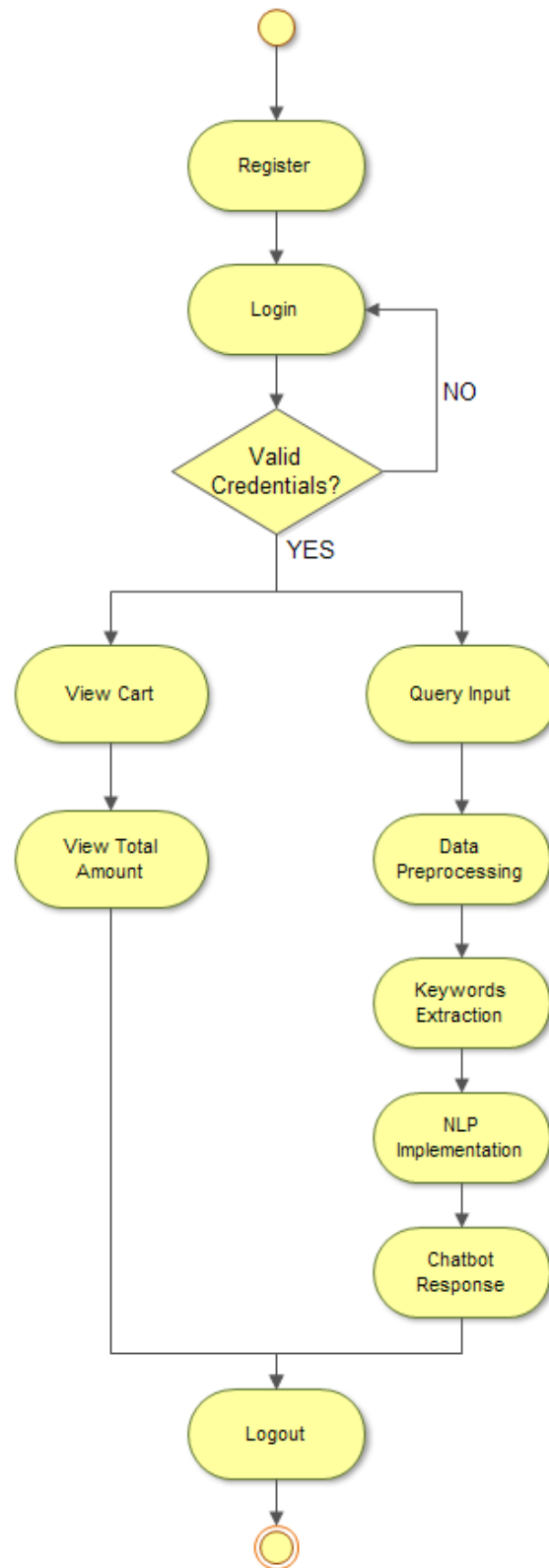
A rectangle with softened, nearly oval edges serves as the symbol for action.



**Fig:4.10**

**Workflow**

An arrow is used to represent workflow. The activity diagram illustrates the workflow's direction.

**User Activity Diagram****Fig:4.11 Activity Diagram**



## **4.5 METHODOLOGY:**

### **User Registration and Authentication**

Objective: Securely register and authenticate users.

#### **Method:**

Users will register with their credentials (username, email, and password).

Registration details will be stored securely using SQLite, ensuring password encryption.

After registration, users can log in to the system using the same credentials to access personalized features, such as order history and saved preferences.

### **2. Voice Command Recognition**

Objective: Allow users to interact with the system using voice commands.

#### **Method:**

Implement a voice recognition system that captures user input in natural language.

Use Android's built-in speech recognition APIs (like Google Speech-to-Text) to convert voice input into text.

The system will handle commands like "Show me the menu," "Add pizza to the cart," or "Remove salad from my cart."

### **3. NLP for Keyword Extraction**

Objective: Enhance the accuracy of voice commands by extracting meaningful keywords for better interaction.

**Method:**

Speech-to-Text: After the voice input is captured and converted into text, NLP algorithms will process the text to extract relevant keywords.

Keyword Extraction: Using NLP techniques such as Named Entity Recognition (NER) and Part-of-Speech (POS) tagging, the system will identify important keywords related to food items, actions (like add/remove), and quantities (e.g., large, small).

Example: If the user says, "Add two pizzas and one soda to the cart," the system will extract keywords like "two," "pizzas," and "one soda."

Intent Recognition: Using predefined intents, the system will understand the user's request. Intents like "Add to cart," "Remove from cart," "Show menu," and "Checkout" will be recognized based on the extracted keywords.

Named Entity Recognition (NER): NER will identify food items, quantities, and specific actions (add/remove) from the text.

#### **4. Menu Display and Interaction**

Objective: Display food items in a categorized menu and allow users to select items through voice commands.

**Method:**

The app will present a categorized menu with food items, prices, and descriptions.

Categories could include appetizers, main dishes, drinks, etc.

Users can say commands like "Show me drinks" or "What is in the appetizers category?" The system will return the corresponding category from the menu.

Voice Command Example: "Show me pizza options," and the system will respond with the available pizzas.

## 5. Cart Management

Objective: Allow users to manage their cart with voice commands.

### Method:

The system will update the cart in real-time as users add or remove items.

Adding Items: When a user says "Add two pizzas to the cart," the NLP module will extract the quantity and item, then add the respective food item to the cart.

Removing Items: The system will also allow users to remove items from the cart with commands like "Remove one soda."

Cart Updates: The total price will be dynamically updated after each addition or removal of an item.

Voice Command Example: "Add two burgers and one coke to my cart," and the system will add these items to the cart, updating the total cost.

## 6. Real-Time Price Calculation

Objective: Automatically calculate the total cost of the items in the cart.

### Method:

As items are added or removed, the system will calculate the total cost in real-time.

The application will access a local or remote database (SQLite) to fetch item prices and perform the necessary calculations.

Users can ask "What is my total?" and the system will respond with the updated total cost of their cart.

## **7. Order Summary and Confirmation**

Objective: Display an order summary and confirm the order before placing it

### **Method:**

Once the user has finalized the cart, they will receive an order summary with the list of items and the total cost.

The system will prompt the user to confirm the order with voice commands like "Confirm order" or "Cancel."

If the user says "Confirm," the order will be placed, and the system will acknowledge with a confirmation message.

## **8. Error Handling and Feedback**

Objective: Ensure the system provides feedback in case of errors or misunderstandings.

### **Method:**

If the system misinterprets the voice command, it will ask the user for clarification, using prompts like "Sorry, I didn't understand that. Could you repeat?"

It will also validate the cart before confirming the order, asking the user if they missed any items.

## **9. User Session Management**

Objective: Manage user sessions securely.

### **Method:**

The system will maintain user sessions so that they stay logged in until they log out or the session expires.

User data, such as previous orders or preferences, will be stored in the app's local database and displayed upon the next login.

## **CHAPTER 5**

### **SYSTEM IMPLEMENTATION**

#### **5.1 OVERVIEW**

The process of turning a new or updated system design into an operational one is called implementation. With the least amount of expense, danger, and personal annoyance possible, the goal is to implement the tested new or updated system. Making ensuring that the organization's operations won't be disrupted is a crucial part of the implementation process. Using carefully thought-out tests to test any new programs is the greatest way to acquire control while implementing any new system. Text files must be made on the old system, transferred to the new system, and used for the first test of every application before production files are used to test actual data.

#### **5.2 PHASE - IN METHOD OF IMPLEMENTATION**

##### **Backend:**

Utilize SQLite for real-time database management and authentication. Design a robust database schema to handle user profiles (students, coordinators, admins), complaint records, and status updates. Implement REST APIs if needed for custom server-side logic. Ensure secure data storage and access control with Firebase Authentication and Firestore Security Rules.

##### **Frontend:**

Develop the app using Android Studio with Java or Kotlin. Create XML layouts for user-friendly interfaces, including forms for complaint submission, dashboards for coordinators and admins, and status tracking screens. Use Android's RecyclerView for dynamic list views.

##### **Testing:**

Conduct unit testing using JUnit for individual components and Espresso for UI testing. Perform integration tests to ensure seamless interaction between different modules.

## **Deployment:**

Prepare the app for deployment by generating a signed APK or AAB. Follow Google Play Store guidelines for submission, including privacy policies and app descriptions. Regularly update the app based on user feedback and emerging security practices.

## **ALGORITHM USED**

### **NLP**

Natural Language Processing (NLP) is a subfield of artificial intelligence (AI) and linguistics that focuses on the interaction between computers and human language. It involves the development of algorithms that allow computers to understand, interpret, and generate human language in a way that is both meaningful and useful. NLP is crucial for tasks such as machine translation, sentiment analysis, chatbot development, text summarization, and much more.

NLP combines computational linguistics (the study of language structure), computer science, and statistical methods to process and analyze large amounts of natural language data.

### **Key Components of NLP**

#### **Tokenization:**

Tokenization is the process of splitting a text into smaller units called tokens, which could be words, sentences, or subwords.

For example, in the sentence "I love programming," tokenization would break it into the tokens: ["I", "love", "programming"].

#### **Part-of-Speech (POS) Tagging:**

POS tagging is the process of assigning each word in a sentence its correct part of speech, such as a noun, verb, adjective, etc.

For example, in the sentence "The cat sleeps," the POS tagging would assign: "The" (determiner), "cat" (noun), and "sleeps" (verb).

### Named Entity Recognition (NER):

NER is a technique used to identify and classify named entities (such as people, organizations, locations, dates, etc.) in text.

For example, in the sentence "Apple was founded by Steve Jobs in Cupertino," NER would identify "Apple" as an organization, "Steve Jobs" as a person, and "Cupertino" as a location.

### Lemmatization and Stemming:

Stemming involves reducing words to their root form (e.g., "running" becomes "run").

Lemmatization is a more advanced form of word reduction, where the word is reduced to its base or dictionary form, considering its meaning and context (e.g., "better" becomes "good").

Both techniques help standardize words for easier analysis.

### Speech Recognition:

Speech recognition is a component of NLP that converts spoken language into written text. It involves tasks like detecting speech patterns, phonemes, and transcribing audio into words.

For example, turning spoken commands like "Add pizza to my cart" into text that the system can process.

## **CHAPTER-6**

### **OUTCOMES**

#### **OUTCOMES OF THE PROJECT**

- **Voice-Based Food Ordering:** Users can seamlessly place food orders using voice input, enhancing convenience and accessibility.
- **Voice Command Cart Management:** Users can add or remove items from the cart through voice commands, making the shopping experience more interactive.
- **Total Amount Calculation:** The cart automatically updates and displays the total amount as users add or remove items.
- **User Authentication:** Only registered and logged-in users can access the food ordering features, ensuring security and personalized experience.
- **Improved User Experience:** Voice input offers a hands-free and efficient method of interacting with the application, improving usability.
- **Real-Time Updates:** As items are added or removed from the cart, the total price is dynamically updated, providing immediate feedback to users.
- **Optimized for Android Devices:** The application is developed to run smoothly on Android devices, ensuring broad accessibility.
- **Integration of NLP for Commands:** The system uses Natural Language Processing (NLP) for understanding and processing user commands, making it more accurate and effective.
- **Increased Engagement:** Voice-enabled features encourage more interaction and usage, boosting overall engagement with the app.



## **CHAPTER-7**

### **CONCLUSION AND FUTURE ENHANCEMENT**

#### **CONCLUSION:**

In conclusion, the proposed food ordering Android application effectively integrates voice commands and personalized features to enhance user experience. By enabling users to interact with the app through voice input, it simplifies the food ordering process, making it more accessible and convenient. The ability to add and remove items from the cart, along with dynamic price updates, provides real-time feedback, improving overall usability. Additionally, the registration and login system ensures secure access to these features, creating a personalized experience for each user.

The integration of Natural Language Processing (NLP) adds an advanced layer of understanding to the voice commands, ensuring that the system accurately interprets and responds to user requests. The combination of these technologies leads to a more efficient and enjoyable way of ordering food, setting the app apart from traditional text-based interfaces. As technology continues to evolve, this project lays the foundation for further innovations in voice-based applications. It showcases the potential of combining machine learning, voice recognition, and mobile app development to provide seamless user experiences in everyday tasks. Overall, this system represents a significant step toward making digital interactions more intuitive, accessible, and engaging for users across various demographics.

## **FUTURE ENHANCEMENT:**

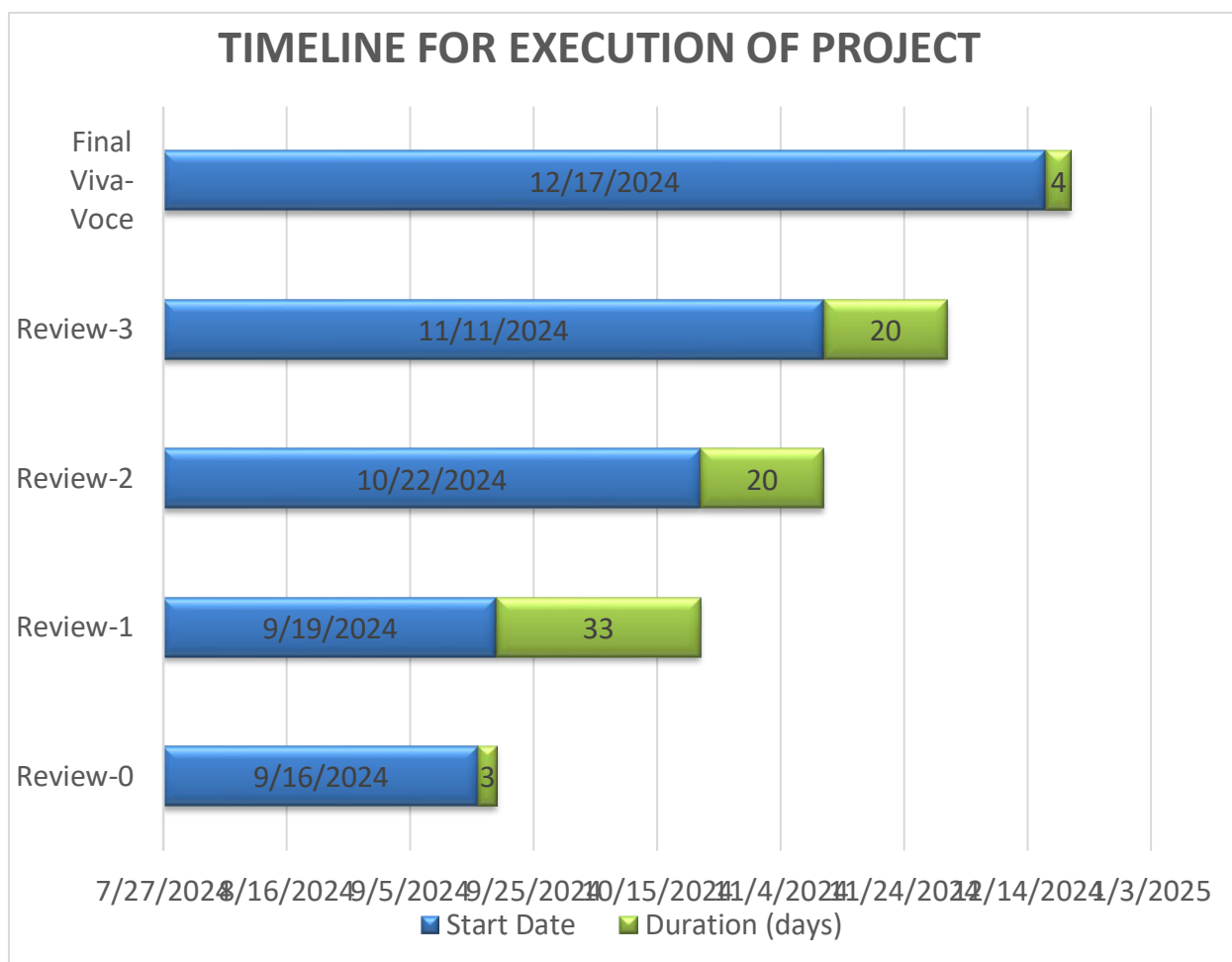
The future scope of this food ordering application is vast and can be extended in several directions to enhance user experience and functionality. One potential development is the integration of advanced machine learning algorithms to offer even more personalized food recommendations based on user preferences, dietary restrictions, or past orders. This could lead to a more tailored dining experience, ensuring users always receive suggestions they are likely to enjoy.

Additionally, expanding the voice recognition feature to support multiple languages and accents can make the application accessible to a wider audience globally. This would also involve improving the Natural Language Processing (NLP) capabilities to handle more complex commands and conversational interactions.

Another area for future improvement is the incorporation of features like real-time tracking of food delivery, notifications for special offers or promotions, and integration with various payment gateways for a seamless checkout experience.

## CHAPTER-8

### TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)



**Fig: 8.1 GANTT CHART**

## REFERENCES

- [1] T. Hansen and T. U. Thomsen, "The influence of consumers' interest in healthy eating definitions of healthy eating and personal values on perceived dietary quality", *Food Policy*, vol. 80, pp. 55-67, Oct. 2018.
- [2] K. Ditlevsen, P. Sandoe and J. Lassen, "Healthy food is nutritious but organic food is healthy because it is pure: The negotiation of healthy food choices by Danish consumers of organic food", *Food Qual Prefer*, vol. 71, pp. 46-53, Jan. 2019.
- [3] P. E. Arroyo, J. Linan and J. Vera Martinez, "Who really values healthy food?", *British Food Journal*, vol. 123, no. 2, pp. 720-738, Jan. 2021.
- [4] S. Safitri, T. Mantoro, M. A. C. Bhakti and W. Wandy, "Cooking and Food Information Chatbot System using GPT-3," 2023 IEEE 9th International Conference on Computing, Engineering and Design (ICCED), Kuala Lumpur, Malaysia, 2023, pp. 1-6, doi: 10.1109/ICCED60214.2023.10425553.
- [5] Zahid, Hashaam. (2023). Food Delivery System With AI Chatbot. 10.13140/RG.2.2.13717.50408.
- [6] The Effect of Website Interface Design on User Experience in Online Food Ordering Systems by Yuhui Sun, Yuhong Wang, and Jing Wang (2019).
- [7] Chatbots in Education and Research: A Critical Examination of Ethical Implications and Solutions by Ana-Paula Correia, Paula de Jesus Pereira, and Ana Alice Baptista (2020)
- [8] Machine learning algorithms for teaching AI chat bots :- Evgeny Tebenkov , Igor Prokhorov [National Research Nuclear University MEPhI ,2020].
- [9] Towards a deep learning based contextual chat bot for preventing depression in young children with autistic spectrum disorder :- Sid Ahmed Hadri , Abdelkrim Bouramoul [University of Abdelhamid Mehri Constantine , 2023]
- [10] Restaurant Chatbot using IBM Watson :- Prof. Sachin Kolekar ,Vedant Vaidya, Sanskruti Sandbhor [Zeal College of Engineering & Research , Pune ,Bharat 2021]
- [11] Artificial Intelligence Chatbot using Python :- Dr. C. K. Gomathy, Redrouthu Venkata Narayana , Dr. V Geetha [Dept. of Computer Science & Engineering SCSVMV , Kanchipuram 2022]

## APPENDIX-A

### PSUEDOCODE

#### HOME PAGE:

```
public class HomeActivity extends AppCompatActivity {  
    private DrawerLayout drawerLayout;  
    private NavigationView navigationView;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_home);  
  
        // Initialize views  
        drawerLayout = findViewById(R.id.drawer_layout);  
        navigationView = findViewById(R.id.nav_view);  
  
        // Configure the ActionBar  
        getSupportActionBar().setTitle("CHATBOT");  
        getSupportActionBar().setDisplayHomeAsUpEnabled(true); // This shows the menu  
icon  
        getSupportActionBar().setHomeAsUpIndicator(R.drawable.ic_menu); // Set custom  
menu icon  
  
        // Set up navigation item click listener  
        navigationView.setNavigationItemSelectedListener(item -> {  
            int itemId = item.getItemId();  
            if (itemId == R.id.nav_home) {  
                Intent intent = new Intent(HomeActivity.this, HomeActivity.class);  
                finish(); // Close the current instance  
                startActivity(intent);  
            } else if (itemId == R.id.nav_login) {  
                Intent intent = new Intent(HomeActivity.this, LoginActivity.class);
```

```
        startActivity(intent);
    } else if (itemId == R.id.nav_register) {
        Intent intent = new Intent(HomeActivity.this, RegisterActivity.class);
        startActivity(intent);
    }

    drawerLayout.closeDrawer(GravityCompat.END);
    return true;
});
}
```

@Override

```
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == android.R.id.home) {
        if (!drawerLayout.isDrawerOpen(GravityCompat.END)) {
            drawerLayout.openDrawer(GravityCompat.END);
        } else {
            drawerLayout.closeDrawer(GravityCompat.END);
        }
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

@Override

```
public void onBackPressed() {
    if (drawerLayout.isDrawerOpen(GravityCompat.END)) {
        drawerLayout.closeDrawer(GravityCompat.END);
    } else {
        super.onBackPressed();
    }
}
}
```

**REGISTRATION PAGE:**

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

    SharedPreferences preferences2 = getSharedPreferences("user_session",
MODE_PRIVATE);
    String email2 = preferences2.getString("email", null);

    if (email2 != null) {
        // If email exists, user is already logged in, so redirect to MainActivity
        startActivity(new Intent(RegisterActivity.this, MainActivity.class));
        finish(); // Finish the login activity so the user cannot navigate back
    }

    // Initialize views
    drawerLayout = findViewById(R.id.drawer_layout);
    navigationView = findViewById(R.id.nav_view);
    edtName = findViewById(R.id.edtName);
    edtEmail = findViewById(R.id.edtEmail);
    edtPassword = findViewById(R.id.edtPassword);
    edtPhone = findViewById(R.id.edtPhone);
    edtAddress = findViewById(R.id.edtAddress);
    btnSubmit = findViewById(R.id.btnSubmit);

    // Initialize database helper
    dbHelper = new DatabaseHelper(this);

    // Configure the ActionBar
    getSupportActionBar().setTitle("Register");
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    getSupportActionBar().setHomeAsUpIndicator(R.drawable.ic_menu);
```

```
// Set up navigation item click listener
navigationView.setNavigationItemSelectedListener(item -> {
    int itemId = item.getItemId();
    if (itemId == R.id.nav_home) {
        Intent intent = new Intent(RegisterActivity.this, HomeActivity.class);
        startActivity(intent);
    } else if (itemId == R.id.nav_login) {
        Intent intent = new Intent(RegisterActivity.this, LoginActivity.class);
        startActivity(intent);
    } else if (itemId == R.id.nav_register) {
        Intent intent = new Intent(RegisterActivity.this, RegisterActivity.class);
        finish(); // Close the current instance
        startActivity(intent);
    }
    drawerLayout.closeDrawer(GravityCompat.END);
    return true;
});

// Set submit button action
btnSubmit.setOnClickListener(v -> {
    String name = edtName.getText().toString();
    String email = edtEmail.getText().toString();
    String password = edtPassword.getText().toString();
    String phone = edtPhone.getText().toString();
    String address = edtAddress.getText().toString();

    // Check if any field is empty
    if (name.isEmpty() || email.isEmpty() || password.isEmpty() || phone.isEmpty() ||
address.isEmpty()) {
        Toast.makeText(RegisterActivity.this, "All fields are required",
Toast.LENGTH_SHORT).show();
    }
}
```

---



```
// Validate email
else if (!isValidEmail(email)) {
    Toast.makeText(RegisterActivity.this, "Invalid email address",
Toast.LENGTH_SHORT).show();
}
// Validate phone
else if (!isValidPhone(phone)) {
    Toast.makeText(RegisterActivity.this, "Invalid phone number",
Toast.LENGTH_SHORT).show();
}
else {
    // Save to database
    saveToDatabase(name, email, password, phone, address);
}
});

}

private boolean isValidEmail(String email) {
    return Patterns.EMAIL_ADDRESS.matcher(email).matches();
}

// Phone validation method using regex
private boolean isValidPhone(String phone) {
    // Example: Simple regex for 10-digit phone numbers
    return phone.matches("^\\d{10}$");
}

private void saveToDatabase(String name, String email, String password, String phone,
String address) {
    SQLiteDatabase db = dbHelper.getReadableDatabase();

    // Check if user already exists
    String[] columns = {"id"};
    String selection = "email = ?";
```

---

```
String[] selectionArgs = {email};

Cursor cursor = db.query("chatbot_2024_user", columns, selection, selectionArgs, null,
null, null);

if (cursor != null && cursor.getCount() > 0) {
    // User already exists
    Toast toast = Toast.makeText(this, "User already exists", Toast.LENGTH_SHORT);
    toast.setGravity(Gravity.TOP, 0, 150); // Position at the top (y-offset 150px)
    toast.show();
    cursor.close();
    db.close();
    return; // Exit the method without saving
}
cursor.close();

// Proceed to save to database if no existing user
db = dbHelper.getWritableDatabase();
ContentValues values = new ContentValues();
values.put("name", name);
values.put("email", email);
values.put("password", password);
values.put("phone", phone);
values.put("address", address);

long rowId = db.insert("chatbot_2024_user", null, values);
if (rowId != -1) {
    // Show success toast at the top of the screen
    Toast toast = Toast.makeText(this, "Registration Successful!",
Toast.LENGTH_SHORT);
    toast.setGravity(Gravity.TOP, 0, 150); // Position at the top (y-offset 150px)
    toast.show();
} else {
```

```
// Show error toast at the top of the screen
Toast toast = Toast.makeText(this, "Error in registration", Toast.LENGTH_SHORT);
toast.setGravity(Gravity.TOP, 0, 150); // Position at the top (y-offset 150px)
toast.show();
}
db.close();
}
```

@Override

```
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == android.R.id.home) {
        if (!drawerLayout.isDrawerOpen(GravityCompat.END)) {
            drawerLayout.openDrawer(GravityCompat.END);
        } else {
            drawerLayout.closeDrawer(GravityCompat.END);
        }
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

@Override

```
public void onBackPressed() {
    if (drawerLayout.isDrawerOpen(GravityCompat.END)) {
        drawerLayout.closeDrawer(GravityCompat.END);
    } else {
        super.onBackPressed();
    }
}
}
```

**LOGIN PAGE:**

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_login);  
  
    SharedPreferences preferences2 = getSharedPreferences("user_session",  
MODE_PRIVATE);  
    String email2 = preferences2.getString("email", null);  
  
    if (email2 != null) {  
        // If email exists, user is already logged in, so redirect to MainActivity  
        startActivity(new Intent(LoginActivity.this, MainActivity.class));  
        finish(); // Finish the login activity so the user cannot navigate back  
    }  
    // Initialize views  
    drawerLayout = findViewById(R.id.drawer_layout);  
    navigationView = findViewById(R.id.nav_view);  
    edtEmail = findViewById(R.id.edtEmail);  
    edtPassword = findViewById(R.id.edtPassword);  
    btnSubmit = findViewById(R.id.btnSubmit);  
  
    // Initialize database helper  
    dbHelper = new DatabaseHelper(this);  
  
    // Configure the ActionBar  
    getSupportActionBar().setTitle("Login");  
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
    getSupportActionBar().setHomeAsUpIndicator(R.drawable.ic_menu);  
  
    // Set up navigation item click listener  
    navigationView.setNavigationItemSelectedListener(item -> {  
        int itemId = item.getItemId();
```

```
        if (itemId == R.id.nav_home) {
            Intent intent = new Intent(LoginActivity.this, HomeActivity.class);
            startActivity(intent);
        } else if (itemId == R.id.nav_login) {
            Intent intent = new Intent(LoginActivity.this, LoginActivity.class);
            finish(); // Close the current instance
            startActivity(intent);
        } else if (itemId == R.id.nav_register) {
            Intent intent = new Intent(LoginActivity.this, RegisterActivity.class);
            startActivity(intent);
        }

        drawerLayout.closeDrawer(GravityCompat.END);
        return true;
    });

    // Set submit button action
    btnSubmit.setOnClickListener(v -> {
        String email = edtEmail.getText().toString();
        String password = edtPassword.getText().toString();

        // Check if any field is empty
        if (email.isEmpty() || password.isEmpty()) {
            Toast.makeText(LoginActivity.this, "All fields are required",
                Toast.LENGTH_SHORT).show();
        }
        // Validate email format
        else if (!IsValidEmail(email)) {
            Toast.makeText(LoginActivity.this, "Invalid email address",
                Toast.LENGTH_SHORT).show();
        }
        else {
            // Check credentials
```

```
        if (checkCredentials(email, password)) {
            // Save session using SharedPreferences
            SharedPreferences preferences = getSharedPreferences("user_session",
MODE_PRIVATE);

            SharedPreferences.Editor editor = preferences.edit();
            editor.putString("email", email);
            editor.apply();

            // Navigate to MainActivity
            Toast.makeText(LoginActivity.this, "Login Successful",
Toast.LENGTH_SHORT).show();

            startActivity(new Intent(LoginActivity.this, MainActivity.class));
            finish(); // Close the login activity
        } else {
            Toast.makeText(LoginActivity.this, "Invalid email or password",
Toast.LENGTH_SHORT).show();
        }
    }
});
}

private boolean isValidEmail(String email) {
    return Patterns.EMAIL_ADDRESS.matcher(email).matches();
}

// Check email and password in database
private boolean checkCredentials(String email, String password) {
    SQLiteDatabase db = dbHelper.getReadableDatabase();
    String[] columns = {"id"};
    String selection = "email = ? AND password = ?";
    String[] selectionArgs = {email, password};

    Cursor cursor = db.query("chatbot_2024_user", columns, selection, selectionArgs, null,
null, null);
```

---

```
if (cursor != null && cursor.getCount() > 0) {  
    cursor.close();  
    return true;  
} else {  
    cursor.close();  
    return false;  
}  
}
```

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    if (item.getItemId() == android.R.id.home) {  
        if (!drawerLayout.isDrawerOpen(GravityCompat.END)) {  
            drawerLayout.openDrawer(GravityCompat.END);  
        } else {  
            drawerLayout.closeDrawer(GravityCompat.END);  
        }  
        return true;  
    }  
    return super.onOptionsItemSelected(item);  
}
```

```
@Override  
public void onBackPressed() {  
    if (drawerLayout.isDrawerOpen(GravityCompat.END)) {  
        drawerLayout.closeDrawer(GravityCompat.END);  
    } else {  
        super.onBackPressed();  
    }  
}
```

## APPENDIX-B

### SCREENSHOTS

HOME:

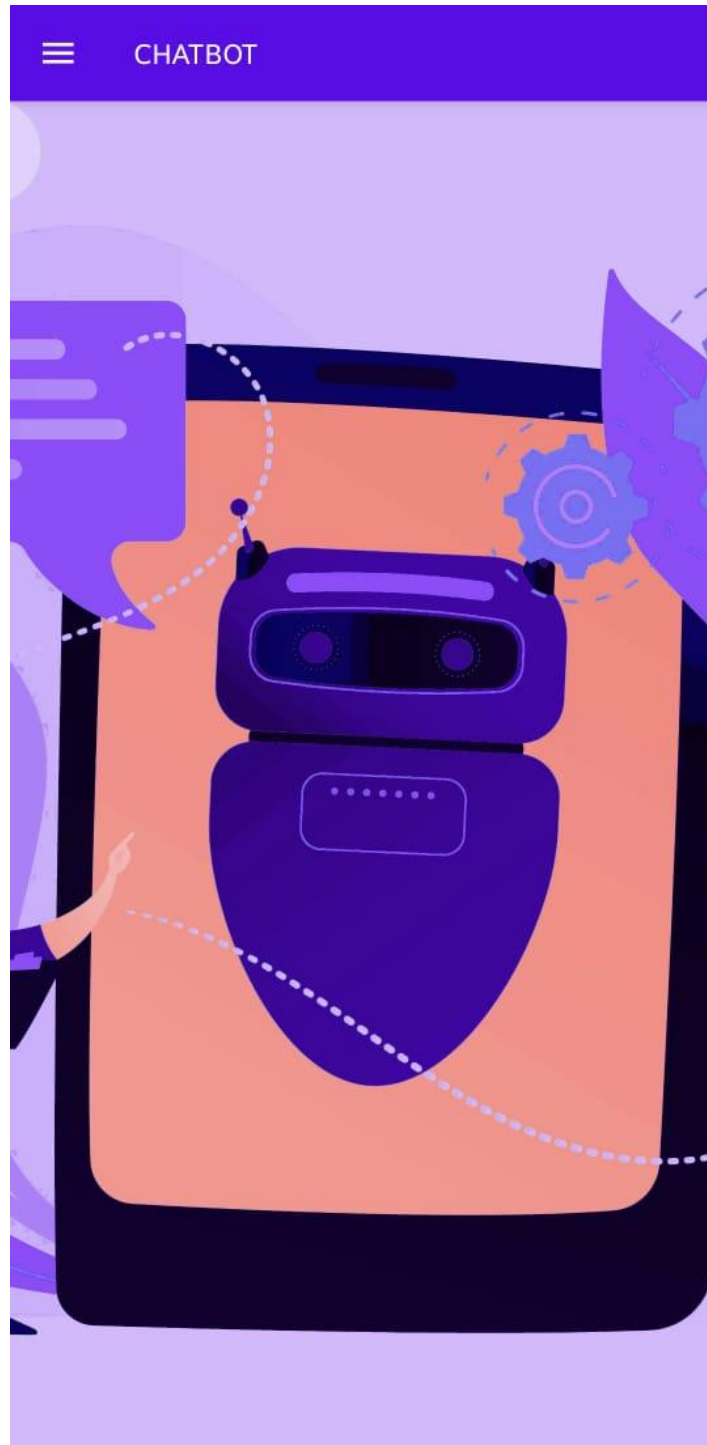
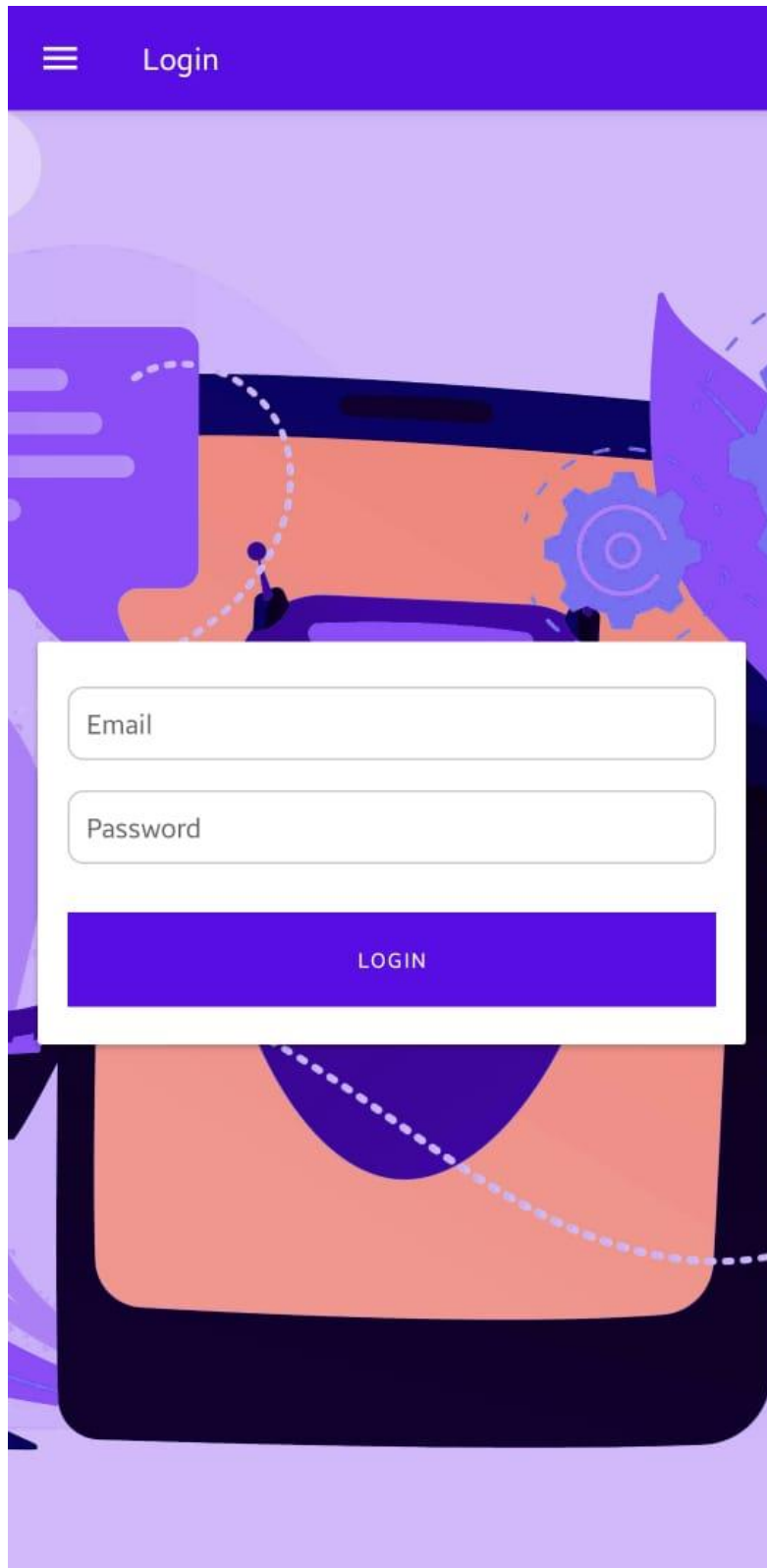


Fig:B.1

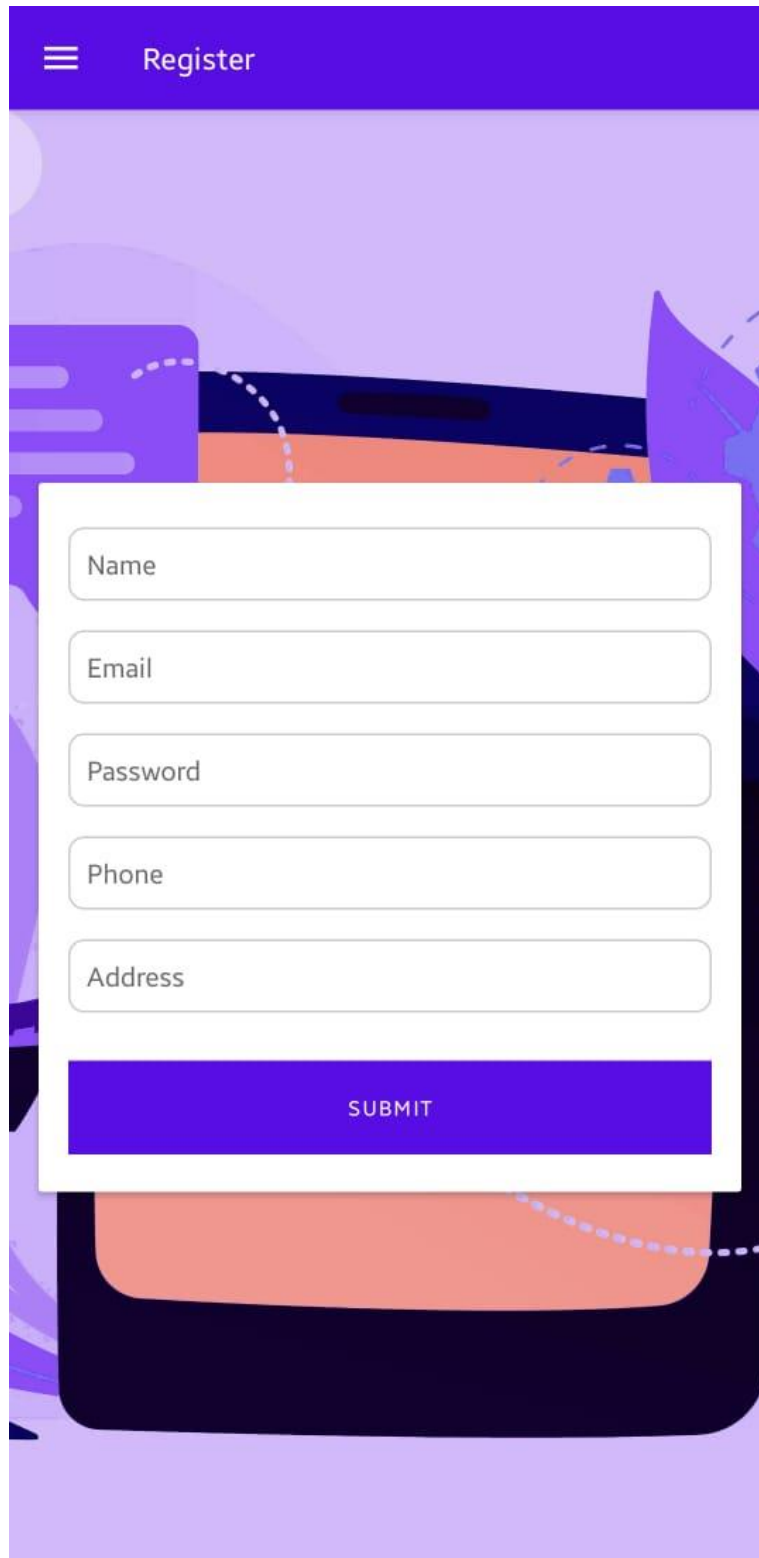


**LOGIN:**



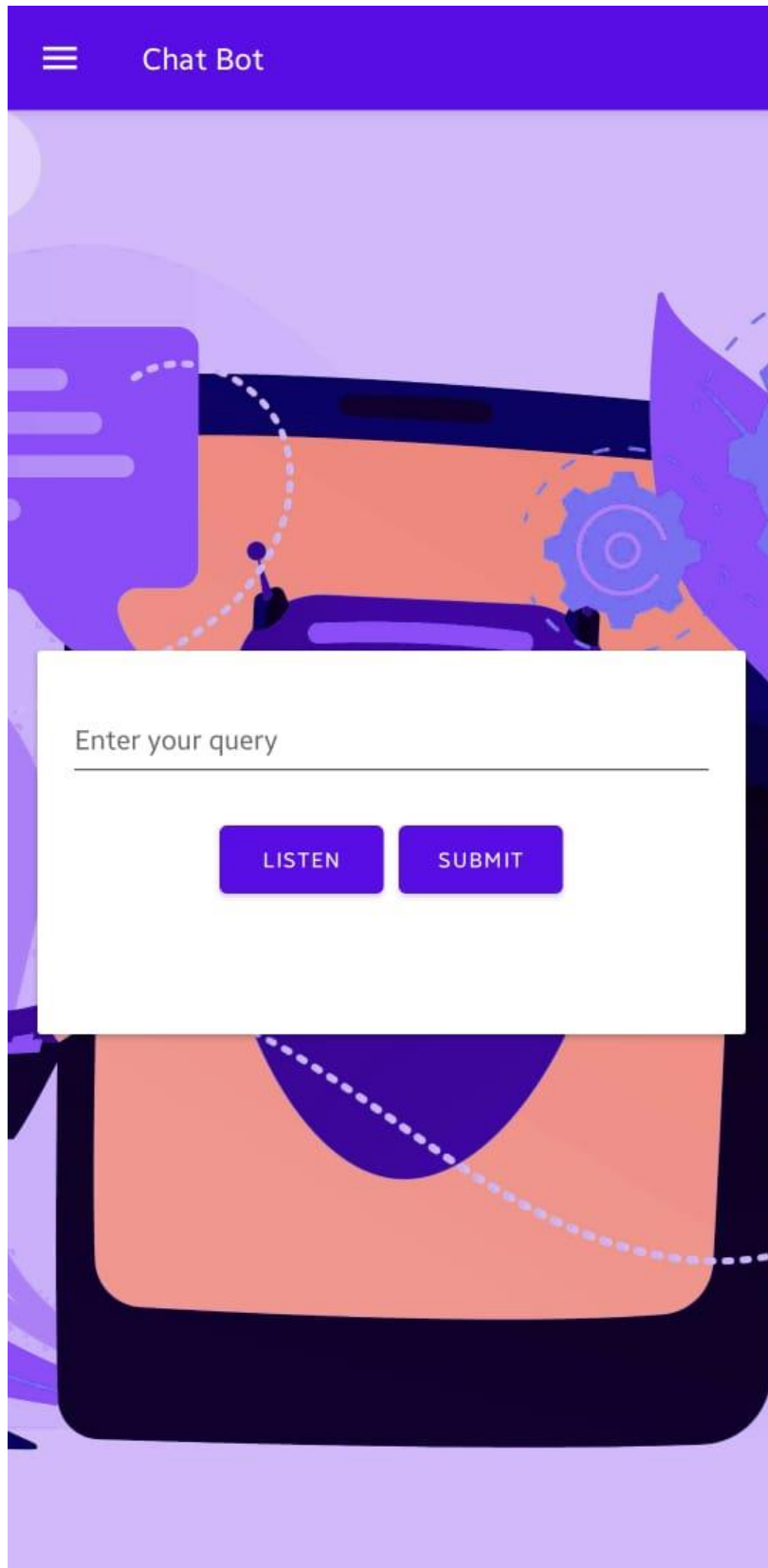
**Fig:B.2**

## REGISTRATION:



The image shows a mobile application interface with a registration form. The background is a purple gradient with abstract shapes. At the top, there is a purple header bar with a white hamburger menu icon on the left and the word "Register" in white text. Overlaid on this is a white rectangular form with rounded corners. The form contains five input fields, each with a light gray border and a light gray placeholder text: "Name", "Email", "Password", "Phone", and "Address". Below these fields is a solid purple button with the word "SUBMIT" in white, uppercase letters. The form is centered on the screen.

**Fig:B.3**



**Fig:B.4**

## **APPENDIX-C**

### **ENCLOSURES**

#### **1. Details of mapping the project with the Sustainable Development Goals (SDGs).**

