**[Type the company name]**

**[Type the company address]**

**[Type the phone number]**

**[Type the fax number]**

**[Pick the date]**

**[Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.]**

[Title of Article]



Catalog

[Abstract 2](#_Toc18769)

[Chapter One: Introduction 3](#_Toc15666)

[1.1 Background of the Project 3](#_Toc3813)

[1.2 Problem Statement 3](#_Toc12405)

[1.3 Project Objectives 4](#_Toc27580)

[1.4 Project Scope and Limitations 5](#_Toc28209)

[Scope 5](#_Toc19811)

[Limitations 5](#_Toc12744)

[1.5 Significance of the Project 6](#_Toc7576)

[Chapter Two: Methodology 7](#_Toc29277)

[2.1 Overview of Selected Methodologies 7](#_Toc16870)

[2.2 Justification for Methodologies 7](#_Toc26229)

[2.3 Data Collection and Analysis 8](#_Toc2362)

[Data Collection 8](#_Toc20929)

[Data Analysis 8](#_Toc32535)

[2.4 Development Approach 8](#_Toc22025)

[Justification for Development Approach 9](#_Toc6467)

[Chapter Three: Feasibility Analysis 9](#_Toc2190)

[3.1 Operational Feasibility 9](#_Toc7220)

[3.2 Technical Feasibility 9](#_Toc22366)

[3.3 Economic Feasibility 10](#_Toc13184)

[3.4 Schedule Feasibility 11](#_Toc26222)

[Summary of Feasibility Analysis 12](#_Toc26388)

[Chapter Four: System Design 13](#_Toc2689)

[4.1 Proposed System Architecture 13](#_Toc6926)

[4.2 Features and Functional Requirements 14](#_Toc30742)

[4.3 Non-Functional Requirements 15](#_Toc30565)

[Summary of System Design 16](#_Toc12020)

[Chapter Five:Testing and Evaluation 16](#_Toc15810)

[5.1 Testing Methods 16](#_Toc12058)

[5.2 Test Cases and Results 17](#_Toc1849)

[5.3 Dynamic Modeling 17](#_Toc10774)

[5.3.1Sequence Diagram 17](#_Toc28871)

[1. Farmer Sends a Query: 18](#_Toc15366)

[2. Chatbot Processes the Query: 18](#_Toc13505)

[3. Backend Retrieves Data: 18](#_Toc20959)

[4. Response Generation: 18](#_Toc24713)

[5. Farmer Receives Response: 18](#_Toc20138)

[5.3.2Activity Diagrams 19](#_Toc4439)

[5.3.3State Diagrams 20](#_Toc20432)

[References 21](#_Toc14290)

# Abstract

The AI-enabled agricultural chatbot system is designed to provide users with accurate, timely, and personalized agricultural advice. The system leverages advanced technologies such as Dialogflow for natural language processing, Python for backend logic, Firebase for real-time data storage, and the Telegram API for user interaction. The chatbot supports bilingual communication in English and Amharic, ensuring accessibility for a wide range of users. Key functionalities include crop selection recommendations, pest control advice, soil preparation guidance, and weather forecasts tailored to specific regions. The system aims to bridge the information gap in Ethiopian agriculture, empowering farmers to make informed decisions and improve productivity. By combining AI with user-friendly design, the chatbot system contributes to sustainable agricultural development and food security in Ethiopia.

Keywords: AI, Chatbot, Agriculture, Dialogflow, Python, Telegram API, Firebase, Ethiopia

# Chapter One: Introduction

## 1.1 Background of the Project

Agriculture is the backbone of Ethiopia's economy, contributing significantly to the country's GDP and employing a large portion of the population. However, Ethiopian farmers face numerous challenges, including limited access to reliable agricultural information, language barriers, and delays in receiving timely advice during critical farming periods. These challenges are particularly pronounced in remote areas, where traditional methods of information dissemination are often ineffective.

To address these issues, this project proposes the development of an AI-enabled agricultural chatbot system. The chatbot will serve as a virtual assistant, providing farmers with real-time, location-specific advice in their preferred language. By leveraging cutting-edge technologies such as Dialog-flow, Python, Firebase, and the Telegram API, the chat-bot will bridge the information gap and empower farmers to make informed decisions, ultimately improving agricultural productivity and food security in Ethiopia.

## 1.2 Problem Statement

Ethiopian farmers often struggle with:

1.Limited Access to Information: Farmers in remote areas lack access to region-specific agricultural advice, such as crop selection, pest control, and soil management.

2. Language Barriers: Many farmers are not proficient in English, which limits their ability to adopt modern farming techniques disseminated in foreign languages.

3. Timeliness of Information: Delays in receiving critical information during planting or harvesting seasons can lead to reduced yields and economic losses.

For example, a farmer in a remote village may not have access to the latest information on pest outbreaks, leading to crop damage and reduced yields. Similarly, language barriers may prevent farmers from understanding and implementing advanced agricultural practices.

The proposed chatbot system addresses these challenges by providing an automated, bilingual platform where farmers can receive instant, accurate, and personalized advice in their preferred language. By integrating with widely accessible platforms like Telegram, the chatbot ensures that farmers can easily interact with the system, regardless of their location or technical expertise.

## 1.3 Project Objectives

**1.3.1 General Objective**

The general objective of this project is to develop an AI-powered chatbot system that provides Ethiopian farmers with efficient, accurate, and timely agricultural information. The system will leverage advanced technologies to deliver personalized recommendations, bridging the information gap and empowering farmers to improve productivity and food security.

**1.3.2 Specific Objectives**

1. Bilingual Support: Deliver bilingual support in English and Amharic to cater to the linguistic diversity of Ethiopian farmers. This ensures that language barriers are minimized, allowing more farmers to benefit from the system.

2. Natural Language Processing: Utilize Dialog-flow for natural language processing and contextual understanding, enabling the chat-bot to interpret user queries accurately and provide relevant responses.

3. Seamless Integration: Enable seamless integration with Telegram for wide accessibility, ensuring that farmers can easily interact with the chat-bot using a familiar platform.

4. Real-Time Responses: Ensure real-time responses through robust back-end systems using Python, Firebase, and ngrok, allowing farmers to receive timely information when they need it most.

5. Personalized Recommendations: Tailor AI responses based on mapped agricultural data and scenarios, providing farmers with personalized recommendations that are relevant to their specific farming conditions.

6. Scalability and Reliability: Design the system to handle thousands of users simultaneously, ensuring 24/7 availability and reliability.

## 1.4 Project Scope and Limitations

**Scope**

The project focuses on developing a chat-bot system that provides agricultural advice to Ethiopian farmers. Key features include:

- Bilingual Support: The chat-bot will support communication in English and Amharic.

- Integration with Telegram: Farmers will interact with the chat-bot via Telegram, ensuring wide accessibility.

- Personalized Recommendations: The chat-bot will provide tailored advice based on regional data and farming scenarios.

- Feedback Mechanism: Farmers can provide feedback to improve the system's accuracy and relevance.

**Limitations**

1. Internet Connectivity: The system relies on internet access, which may be limited in remote areas.

2. Data Availability: The catboat's effectiveness depends on the availability of up-to-date agricultural data.

3. Language Support: While the chat-bot supports English and Amharic, expanding to other Ethiopian languages (e.g., Afan Oromo, Tigrinya) would require additional development.

## 

## 1.5 Significance of the Project

The AI-enabled agricultural chat-bot system has the potential to transform Ethiopian agriculture by:

1. Empowering Farmers: Providing farmers with timely and accurate information to improve productivity and decision-making.

2.Bridging the Information Gap: Ensuring that farmers in remote areas have access to the same quality of information as those in urban centers.

3. Enhancing Food Security: Supporting sustainable farming practices and increasing crop yields, contributing to national food security.

4. Promoting Technology Adoption: Introducing farmers to AI and digital tools, fostering a culture of innovation in agriculture.

By leveraging cutting-edge technology, the chat-bot system will play a crucial role in addressing real-world challenges in Ethiopian agriculture, ultimately leading to sustainable development and economic growth.

# Chapter Two: Methodology

## 2.1 Overview of Selected Methodologies

The development of the AI-enabled agricultural chatbot system follows the Agile methodology, specifically the Scrum framework. Agile was chosen for its iterative and incremental approach, which allows for continuous feedback and improvement throughout the development process. Key features of Agile include:

- Sprints: Development is divided into short, time-boxed iterations (sprints) lasting 2–4 weeks.

- User Stories: Requirements are captured as user stories, focusing on the needs of farmers.

- Daily Stand-ups: Regular team meetings to track progress and address challenges.

- Continuous Integration: Frequent integration of code to ensure a working product at all times.

This methodology ensures that the chatbot system evolves based on user feedback and changing requirements, making it highly adaptable to the needs of Ethiopian farmers.

## 2.2 Justification for Methodologies

The Agile methodology was chosen for the following reasons:

1. Flexibility: Agile allows for changes in requirements, which is crucial for a project targeting diverse user needs (e.g., farmers in different regions with varying agricultural practices).

2. User-Centric Development: By focusing on user stories and continuous feedback, Agile ensures that the chatbot meets the actual needs of farmers.

3. Rapid Prototyping: Agile enables quick development and testing of prototypes, allowing for early validation of ideas.

4. Collaboration: Regular communication between team members and stakeholders ensures alignment and reduces the risk of misunderstandings.

For system design and architecture, a modular approach was adopted to ensure scalability and maintainability. This approach allows for easy integration of new features (e.g., additional languages or agricultural datasets) in the future.

## 2.3 Data Collection and Analysis

### Data Collection

The following tools and techniques were used for data collection:

1. Agricultural Data:

- Sources: Government agricultural databases, research papers, and expert consultations.

- Tools: Web scraping (using Python libraries like BeautifulSoup) and manual data entry.

2. User Feedback:

- Sources: Farmers interacting with the chatbot during testing phases.

- Tools: Firebase for storing feedback and Telegram for collecting user inputs.

### Data Analysis

- Natural Language Processing (NLP): Dialogflow was used to analyze user queries and extract intents (e.g., crop recommendation, pest control) and entities (e.g., region, soil type).

- Data Processing: Python scripts were used to process and analyze agricultural data, generating personalized recommendations based on user inputs.

- Visualization: Matplotlib and Seaborn libraries were used to visualize data trends and patterns (e.g., crop yield predictions).

## 2.4 Development Approach

The development approach for the chatbot system is iterative and incremental, with the following key steps:

1. Requirement Gathering:

- Conducted interviews with farmers and agricultural experts to identify key requirements.

- Defined user stories (e.g., "As a farmer, I want to know the best crops for my region so that I can maximize yield").

2. Design and Prototyping:

- Created wireframes and use case diagrams to visualize the system.

- Developed a basic prototype using Telegram, Python, and Dialogflow.

3. Implementation:

- Built the chatbot backend using Python and integrated it with Dialogflow and Firebase.

- Implemented features such as crop recommendations, pest control advice, and weather forecasts.

4. Testing and Feedback:

- Conducted unit testing, integration testing, and system testing.

- Collected feedback from farmers during pilot testing and iterated on the design.

5. Deployment:

- Deployed the chatbot on Telegram and Firebase for real-world use.

- Monitored system performance and addressed user feedback.

## Justification for Development Approach

- Iterative Development: Ensures that the system evolves based on user feedback and changing requirements.

- Modular Design: Allows for easy integration of new features and scalability.

- User Involvement: Farmers are involved throughout the development process, ensuring that the system meets their needs.

# Chapter Three: Feasibility Analysis

## 3.1 Operational Feasibility

The AI-enabled agricultural chat-bot system is designed to be operationally feasibly leveraging widely used technologies and platforms that are accessible to Ethiopian farmers. The system will integrate with Telegram, a popular messaging app, ensuring that farmers can easily interact with the chat-bot without requiring additional training or technical expertise. This approach minimizes the learning curve and maximizes user adoption.

Additionally, the chat-bot will provide bilingual support in English and Amharic, catering to the linguistic diversity of Ethiopian farmers. This ensures that language barriers are minimized, allowing a broader audience to benefit from the system. The system's user-friendly interface, combined with its ability to deliver real-time, personalized advice, makes it operationally viable for farmers across different regions and literacy levels.

## 3.2 Technical Feasibility

The technical feasibility of the chat-bot system is ensured by using state-of-the-art technologies that are readily available and well-documented. The system will leverage the following tools and frameworks:

1. Dialog-flow: For natural language processing (NLP) and contextual understanding, enabling the chat-bot to interpret user queries accurately and provide relevant responses.

2. Python: For back-end logic, API integration, and data processing, ensuring robust and saleable system performance.

3. Firebase: For real-time data storage and retrieval, providing a secure and scalable database solution for storing user data and agricultural information.

4. ngrok: For secure local server deployment during development and testing, facilitating seamless integration with Telegram and other APIs.

5. Telegram API: For user interaction, ensuring that farmers can access the chat-bot through a familiar and widely used platform.

The use of these technologies ensures that the chat-bot system is technically feasible, saleable, and capable of handling thousands of users simultaneously. Furthermore, the system's modular architecture allows for future enhancements, such as adding support for additional languages or integrating voice-based interactions.

## 

## 3.3 Economic Feasibility

From an economic standpoint, the chat-bot system is designed to minimize costs while maximizing impact. The project utilizes free-tier cloud services (e.g., Firebase, Dialog-flow) and open-source tools (e.g., Python, ngrok), significantly reducing development and operational expenses. By leveraging these cost-effective solutions, the system remains economically viable for deployment in resource-constrained environments.

Additionally, the chat-bot system has the potential to generate long-term economic benefits by:

1. Improving Agricultural Productivity: Providing farmers with accurate and timely information can lead to increased crop yields and reduced losses, boosting their income.

2. Reducing Information Costs: Farmers will no longer need to rely on expensive or time-consuming methods to access agricultural advice.

3. Scaling with Minimal Costs: The system's cloud-based architecture ensures that it can scale to accommodate more users without significant additional costs.

Overall, the chat-bot system represents a cost-effective solution for addressing the information gap in Ethiopian agriculture.

## 

## 3.4 Schedule Feasibility

The development of the chat-bot system will follow a structured timeline to ensure that project milestones are met in a timely manner. The proposed schedule is as follows:

1. Requirements Gathering and Analysis (2 weeks):

- Identify user needs, functional requirements, and system specifications.

- Gather agricultural data and define use cases.

2. System Design and Architecture (3 weeks):

- Design the system's three-tier architecture (client, application, and data layers).

- Define the integration points for Dialogflow, Telegram API, Firebase, and Python.

3. Development and Testing (6 weeks):

- Develop the chatbot's core functionalities, including NLP, data processing, and API integration.

- Conduct unit testing and integration testing to ensure system reliability.

4. Deployment and User Training (2 weeks):

- Deploy the chatbot on Telegram and Firebase.

- Provide training materials and support for farmers to use the system effectively.

5. Evaluation and Feedback (2 weeks):

- Collect user feedback and evaluate system performance.

- Identify areas for improvement and implement necessary updates.

By adhering to this timeline, the project can progress smoothly and be completed within the allocated time-frame. The use of agile development methodologies will allow for iterative improvements and ensure that the system meets user needs effectively.

## Summary of Feasibility Analysis

|  |  |
| --- | --- |
| Feasibility Aspect | Key Points |
| Operational Feasibility | - Easy integration with Telegram for wide accessibility.  - Bilingual support in English and Amharic.  - User-friendly interface for farmers. |
| Technical Feasibility | - Uses Dialog-flow, Python, Firebase, ngrok, and Telegram API.  - Scalable and modular architecture.  - Capable of handling thousands of users simultaneously. |
| Economic Feasibility | - Utilizes free-tier cloud services and open-source tools.  - Cost-effective solution with long-term economic benefits. |
| Schedule Feasibility | - Structured timeline with clear milestones.  - Agile development for iterative improvements. |

# 

# Chapter Four: System Design

## 4.1 Proposed System Architecture

The AI-enabled agricultural chat-bot system will follow a three-tier architecture to ensure efficiency, scalability, and modularity. The architecture consists of the following layers:

**1. Client Layer (User Interface):**

- Farmers interact with the chat-bot through Telegram, a widely used messaging platform. Telegram provides a user-friendly interface and ensures accessibility for farmers, even in remote areas.

- The chat-bot supports bilingual communication (English and Amharic) to cater to the linguistic diversity of Ethiopian farmers.

**2. Application Layer (Logic and Processing):**

- This layer handles the core functionality of the chat-bot, including:

- Natural Language Processing (NLP): Powered by Dialog-flow, the chat-bot interprets user queries and generates contextually relevant responses.

- Back-end Logic: Implemented in Python, this component processes user inputs, retrieves data from the database, and generates personalized recommendations.

- API Integration: The chat-bot integrates with external APIs (e.g., weather APIs, agricultural databases) to provide real-time and location-specific advice.

**3. Data Layer (Storage and Retrieval):**

- The data layer is powered by Firebase, a cloud-based database that stores:

- Agricultural Data: Information on crops, soil types, pest control, and weather forecasts.

- User Data: Farmer profiles, interaction history, and feedback.

- Firebase ensures real-time data synchronization and provides a scalable solution for handling large volumes of data.

**System Flow:**

1. A farmer sends a query (e.g., "What crops grow best in my region?") via Telegram.

2. The query is processed by Dialog-flow for NLP and intent recognition.

3. The back-end logic (Python) retrieves relevant data from Firebase and generates a response.

4. The chat-bot sends the response back to the farmer via Telegram.

## 4.2 Features and Functional Requirements

The chat-bot system will offer the following features and functionalities:

1. Bilingual Support:

- The chat-bot will support communication in English and Amharic, ensuring accessibility for a wide range of users.

2. Agricultural Advice:

- The chat-bot will provide personalized recommendations on:

- Crop Selection: Based on soil type, climate, and region.

- Pest Control: Tailored advice for managing pests and diseases.

- Soil Preparation: Guidance on soil testing and fertilization.

- Weather Forecasts: Real-time weather updates for planning farming activities.

3. Feedback Mechanism:

- Farmers can provide feedback on the chatbot's responses, enabling continuous improvement of the system.

4. AI Mapping:

- The chatbot will use \*\*mapped agricultural data\*\* to provide location-specific advice, ensuring that recommendations are relevant to each farmer's unique conditions.

5. User Authentication:

- Farmers can create profiles to save their preferences and interaction history, enabling personalized experiences.

6. Scalability:

- The system will be designed to handle thousands of users simultaneously, ensuring reliable performance during peak usage periods.

## 4.3 Non-Functional Requirements

In addition to the functional requirements, the chat-bot system will adhere to the following non-functional requirements:

1. Scalability:

- The system will be designed to scale horizontally, accommodating an increasing number of users without compromising performance.

2. Reliability:

- The chat-bot will guarantee 24/7 availability, ensuring that farmers can access information whenever they need it.

3. Security:

- All communications between the farmer and the chat-bot will be encrypted to protect sensitive information.

- User data stored in Firebase will be secured using authentication and access control mechanisms.

4. Usability:

- The chat-bot will feature a user-friendly interface with support for Amharic keyboard input, making it easy for farmers to interact with the system.

5. Performance:

- The system will provide real-time responses, with a target response time of less than 2 seconds for most queries.

6. Maintainability:

- The system will be designed with modularity in mind, allowing for easy updates and maintenance.

- Comprehensive documentation will be provided to facilitate future enhancements.

## Summary of System Design

|  |  |
| --- | --- |
| Aspect | Details |
| Architecture | Three-tier architecture: Client (Telegram), Application (Python, Dialogflow), Data (Firebase). |
| Key Features | Bilingual support, agricultural advice, feedback mechanism, AI mapping, user authentication. |
| Non-Functional Requirements | Scalability, reliability, security, usability, performance, maintainability. |

# Chapter Five:Testing and Evaluation

## 5.1 Testing Methods

The chat-bot system was tested using the following methods:

- Unit Testing: Individual components (e.g., NLP module, response generator) were tested for functionality.

- Integration Testing: The interaction between Dialog-flow, Python back-end, and Firebase was tested to ensure seamless data flow.

- System Testing: The entire system was tested to verify that it meets the specified requirements.

## 5.2 Test Cases and Results

- Test Case 1: Query submission and response generation.

Input: "What crops grow best in my region?"

Expected Output: A list of recommended crops based on soil type and climate.

Result: Passed.

- Test Case 2: Bilingual support.

Input: "የግብርና ድጋፍ እንዴት ማግኘት እችላለሁ?"

Expected Output: Response in Amharic.

Result: Passed.

## 5.3 Dynamic Modeling

Dynamic modeling is used to represent the behavior of the system over time. It focuses on how the system responds to external events and internal processes. For the chatbot system, dynamic modeling is essential to visualize the flow of interactions, processes, and states of the system.

5.3.1Sequence Diagrams

Purpose:

Sequence diagrams illustrate the flow of interactions between different components of the system. They show how the user, chatbot, and backend systems communicate to achieve a specific task.

Key Components:

- Actors: Represented by vertical lifelines (e.g., Farmer, Chatbot, Dialogflow, Firebase).

- Messages: Arrows between lifelines showing the flow of information or actions.

- Activation Bars: Rectangles on lifelines indicating when a component is active.

Example: Query Submission and Response Generation

1. Farmer Sends a Query:

- The farmer sends a query (e.g., "What crops grow best in my region?") via Telegram.

- The Telegram API forwards the query to the chatbot backend.

2. Chatbot Processes the Query:

- The chatbot backend sends the query to Dialogflow for natural language processing (NLP).

- Dialogflow identifies the intent (e.g., "crop recommendation") and extracts relevant entities (e.g., region, soil type).

3. Backend Retrieves Data:

- The chatbot backend queries Firebase for agricultural data (e.g., crop recommendations based on soil type and climate).

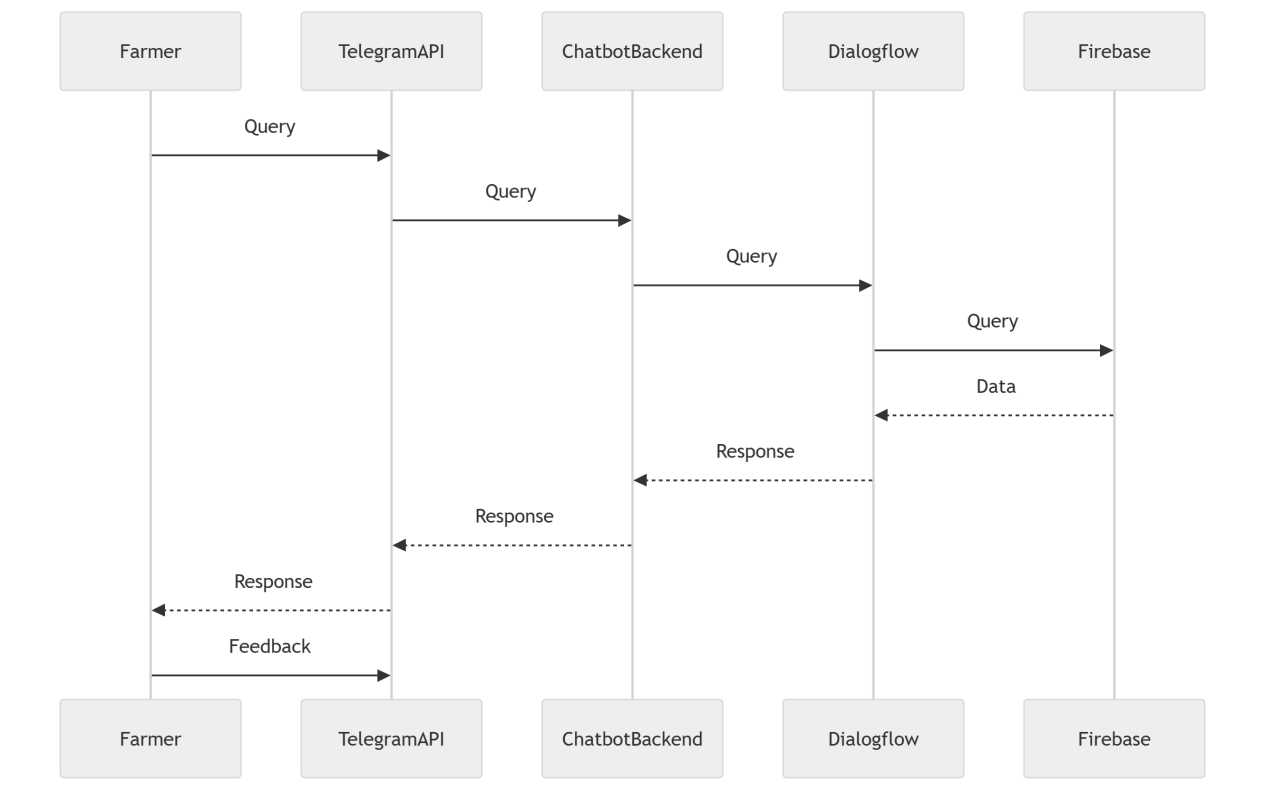
4. Response Generation:

- The backend generates a response (e.g., "For your region, maize and teff are recommended crops.").

- The response is sent back to the farmer via Telegram.

1. Farmer Receives Response:

- The farmer receives the response and can provide feedback (e.g., "Was this helpful?").



### 5.3.2Activity Diagrams

Purpose:

Activity diagrams show the step-by-step processes involved in handling a query, retrieving data, and generating a response. They are useful for visualizing workflows and decision points.

Key Components:

- Start and End Nodes: Represent the beginning and end of the process.

- Actions: Rectangles with rounded corners representing specific tasks (e.g., "Receive Query").

- Decisions: Diamonds representing decision points (e.g., "Is the query valid?").

- Flows: Arrows showing the sequence of actions.#### Example: Query Handling Process

1. Start: The process begins when the farmer sends a query.

2. Receive Query: The chatbot backend receives the query from Telegram.

3. Validate Query: The system checks if the query is valid (e.g., contains relevant keywords).

- If valid, proceed to the next step.

- If invalid, send an error message to the farmer.

4. Process Query: The query is sent to Dialogflow for NLP.

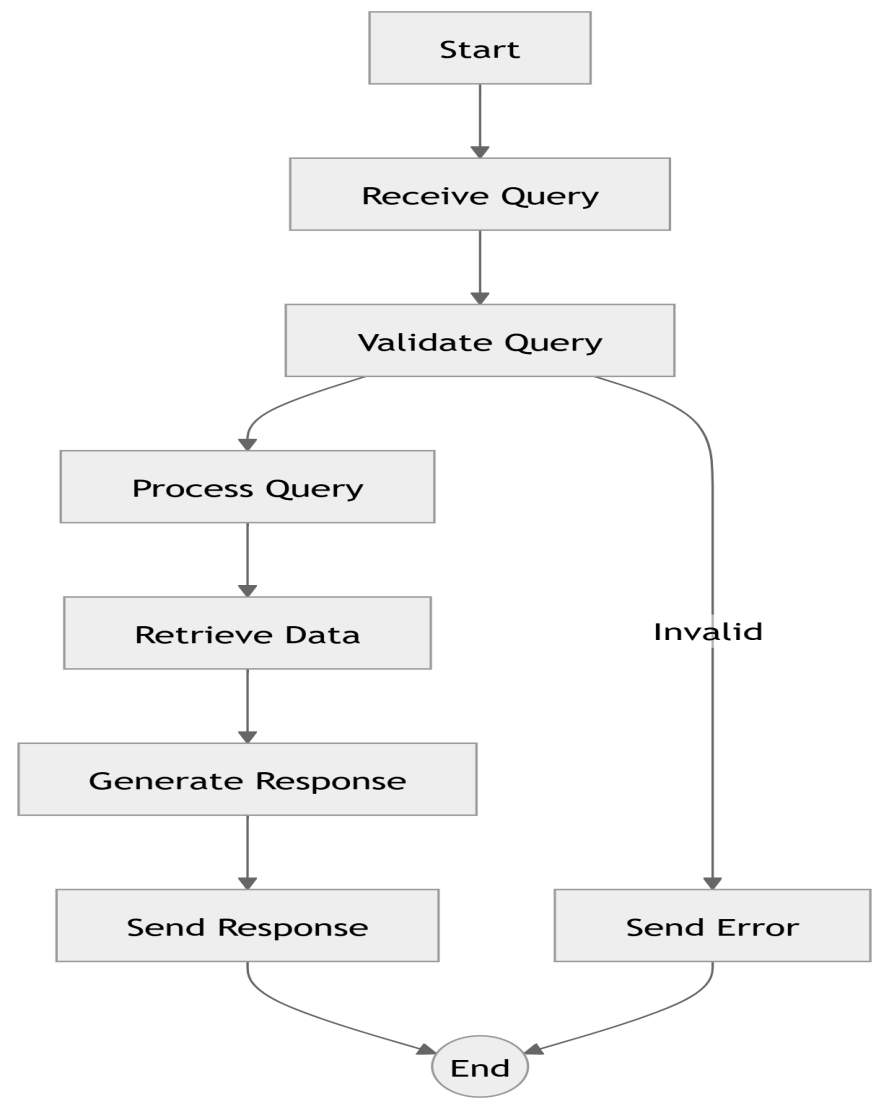
5. Retrieve Data: The backend queries Firebase for relevant data.

6. Generate Response: The backend generates a response based on the retrieved data.

7. Send Response: The response is sent back to the farmer via Telegram.

8. End: The process ends when the farmer receives the response.

Diagram Example:



### 5.3.3State Diagrams

Purpose:

State diagrams represent the different states of the chatbot and how it transitions between them based on events or conditions. They are useful for understanding the lifecycle of the chatbot.

Key Components:

- States: Represented by rectangles with rounded corners (e.g., "Idle", "Processing", "Responding").

- Transitions: Arrows showing how the system moves from one state to another.

- Events: Labels on transitions indicating what triggers the state change (e.g., "Query Received").

Example: Chatbot States

1. Idle: The chatbot is waiting for a query.

- Transition: When a query is received, the chatbot moves to the "Processing" state.

2. Processing: The chatbot is analyzing the query and retrieving data.

- Transition: Once the response is generated, the chatbot moves to the "Responding" state.

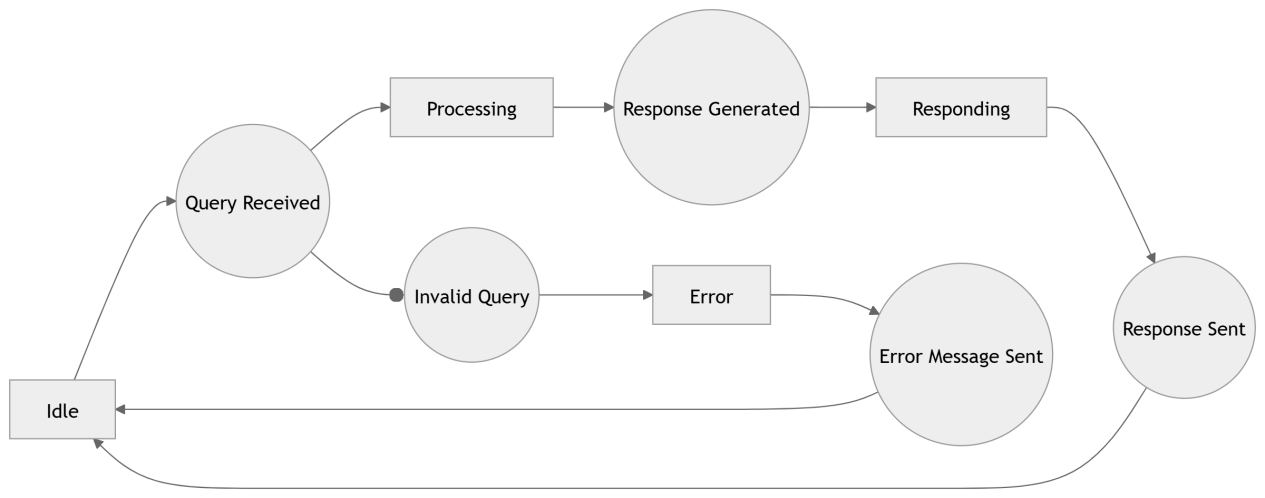
3. Responding: The chatbot sends the response to the farmer.

- Transition: After sending the response, the chatbot returns to the "Idle" state.

4. Error: If an error occurs (e.g., invalid query), the chatbot moves to the "Error" state.

- Transition: After sending an error message, the chatbot returns to the "Idle" state.

Diagram :



**Use-Case Diagram Description**

Actors:

Farmer: The primary user of the chatbot system.

Admin: Responsible for managing the system (e.g., updating agricultural data, monitoring system performance).

**Use Cases:**

Submit Query:

The farmer submits a query (e.g., "What crops grow best in my region?").

The chatbot processes the query and provides a response.

Receive Crop Recommendation:

The farmer receives personalized crop recommendations based on soil type, climate, and region.

Receive Pest Control Advice:

The farmer receives advice on how to manage pests and diseases affecting their crops.

Receive Weather Forecast:

The farmer receives real-time weather updates for their region.

Provide Feedback:

The farmer provides feedback on the chatbot’s responses (e.g., "Was this helpful?").

Update Agricultural Data:

The admin updates the agricultural database with new information (e.g., crop data, pest control methods).

Monitor System Performance:

The admin monitors the chatbot’s performance and user interactions.

# Use case diagram

# References

1. Google Cloud, "Dialogflow Documentation," [Online]. Available: https://cloud.google.com/dialogflow/docs. [Accessed: Dec. 15, 2024].
2. Firebase, "Firebase Realtime Database," [Online]. Available: https://firebase.google.com/docs/database. [Accessed: Dec. 15, 2024].
3. Telegram, "Telegram Bot API," [Online]. Available: https://core.telegram.org/bots/api. [Accessed: Dec. 15, 2024].
4. K. Pohl, Requirements Engineering: Fundamentals, Principles, and Techniques. Berlin, Germany: Springer, 2010.

5. A. Mavin, P. C. Sawyer, and K. L. Jones, "Obstacles to Effective Requirements Engineering," IEEE Software, vol. 29, no. 3, pp. 72–80, May/June 2012.