**NLP Chatbot Development using Dialogflow**

**Software Requirements Specification**

Version 1.0



**Group Id: F24PROJECT38A67**

**Supervisor Name: Abdullah Qamar**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date (dd/mm/yyyy)** | **Version** | **Description** | **Author** |
| 30/11/2024 | 1.0 | This project involves creating an AI-driven chatbot specifically designed for restaurants using Google Dialogflow. The chatbot will simplify key operations, such as making table reservations, exploring menu options, placing orders, and handling customer inquiries. The primary aim is to improve customer satisfaction and streamline the overall restaurant workflow.  With this chatbot, customers will be able to conveniently reserve tables, view menu items, place orders, and receive prompt responses to their questions, all through a conversational platform. By leveraging Natural Language Processing (NLP), the chatbot ensures smooth communication, reduces customer waiting times, and alleviates the workload of restaurant staff, contributing to a more efficient operation. | BC230424569 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. [Scope (of the project)](#scope)
2. [Functional Requirements Non Functional requirements](#FRNFR)
3. [Use Case Diagram](#UCD)
4. [Usage Scenarios](#UCS)
5. [Adopted Methodology](#Adopted)
6. [Work Plan (Use MS Project to create Schedule/Work Plan)](#Gantt)

**SRS Document**

Scope of Project:

The purpose of this project is to develop an AI-powered chatbot specifically for restaurant businesses using Google Dialogflow. The chatbot will act as a virtual assistant, providing functionalities such as making table reservations, navigating the menu, placing food orders, and answering common queries. This system is designed to enhance customer interactions, improve efficiency, and offer greater convenience.

The chatbot will handle key customer requests such as answering frequently asked questions, processing reservation requests, navigating the menu, and assisting with placing orders. By offering quick, conversational responses, it ensures that customers receive timely assistance, thereby improving their overall experience.

However, the system will not address complex order customizations beyond predefined options. It will also exclude payment processing, detailed inventory management, and customer feedback analysis, as these are beyond the scope of this project.

Functional and non Functional Requirements:

**Functional Requirements:**

1. **Handling User Inquiries:**

* The chatbot should understand and respond to common user queries related to restaurant services.

1. **Reservation Management:**

* The chatbot should allow users to book a table at the restaurant through a conversational interface.
* Users should be able to specify the date, time, and number of people for the reservation.

1. **Navigating the Menu:**

* The chatbot should provide customers with menu details, including dish descriptions, prices, and availability upon request.

1. **Order Processing:**

* The chatbot should enable customers to place food orders directly via conversation.
* Customers should have the ability to confirm, modify, or cancel their orders as needed.

1. **Addressing Common Questions:**

* The chatbot must provide clear and accurate answers to common customer questions about the restaurant, such as its location, hours, and available services.

**Non Functional Requirements:**

1. **Performance**

* **Response Time:** The chatbot should reply to users' questions or requests within 2 seconds so that users don’t have to wait long.
* **Handling Multiple Users:** The chatbot should be able to handle many users at once, especially during busy times, without slowing down.

1. **Reliability**

* **Availability:** The chatbot should be available **most of the time**, meaning it should work most of the time with very little downtime.
* **Error Handling:** If something goes wrong, the chatbot should give helpful error messages and continue working without crashing.

1. **Scalability**

* **Ability to Grow:** The chatbot should be easily scalable to handle more users as the restaurant becomes more popular without causing any problems, like slow response times.

1. **Usability**

* **Easy to Use:** The chatbot should be simple and easy for customers to use, even if they aren’t familiar with technology.

1. **Security**

* **Data Protection:** The chatbot should keep customer information, such as names and order details, safe and secure.

1. **Maintainability**

* **Easy to Update:** The chatbot should be easy to update if new features are added or changes are needed, without causing problems for users.
* **Modular Design:** The chatbot should have a modular design, allowing easy updates or changes to specific components without affecting the entire system.

1. **Availability and Fault Tolerance**

* **Always Available:** The chatbot should be always available 24/7, especially during restaurant hours. If something goes wrong, there should be backup systems to keep it running.
* **Fault Recovery:** If there’s a problem, the chatbot should be able to fix itself or keep working without interrupting the user.

1. **Compatibility**

* **Works on Different Devices:** The chatbot should be compatible with multiple devices and platforms, ensuring users can access it on smartphones, tablets, and desktops.

Use Case Diagram(s):



Usage Scenarios:

|  |  |
| --- | --- |
| **Use Case Title** | Answer User Inquiries |
| **Use Case Id** | UC-001 |
| **Description** | The chatbot understands and responds to user queries related to restaurant services. |
| **Action** | 1. Customer types a query.  2. Chatbot processes the query using NLP.  3. Chatbot provides a relevant response or information. |
| **Alternate Path** | If the query is unclear, chatbot requests for clarification. |
| **Pre-Condition** | The chatbot must be active and connected to the system, and capable of understanding queries. |
| **Post-Condition** | The Customer inquiry is resolved, or they are prompted to ask another question. |
| **Author** | Bc230424569 |
| **Exception** | If chatbot fails to understand the query and will ask the user to rephrase the question for better understanding. |

|  |  |
| --- | --- |
| **Use Case Title** | Make Reservation |
| **Use Case Id** | UC-002 |
| **Description** | The chatbot allows users to book a table at the restaurant by providing details like date, time, number of people. |
| **Action** | 1. Customer enters details( date, time, number of people).  2. Chatbot checks availability and confirms the reservation. |
| **Alternate Path** | If the requested time is unavailable, chatbot suggests alternative slots. |
| **Pre-Condition** | The Customer must have access to internet so that chatbot must have access to the reservation system . |
| **Post-Condition** | The customer receives a confirmation of the reservation with details. |
| **Author** | BC230424569 |
| **Exception** | If the user enters invalid data, the chatbot will prompt for corrections. |

|  |  |
| --- | --- |
| **Use Case Title** | Navigate Menu |
| **Use Case Id** | UC-003 |
| **Description** | The chatbot helps customers browse the restaurant’s menu by showing categories and item details like description and prices. |
| **Action** | 1. Customer asks to see the menu.  2. Chatbot displays categories of the menu.  3. Customer selects a category.  4. Chatbot shows items in the selected category along with descriptions and prices.  5. Customer asks for details of a specific item, if needed. |
| **Alternate Path** | If the customer directly searches for an item, the chatbot skips categories and shows the relevant item. |
| **Pre-Condition** | The chatbot must have access to the restaurant’s menu database. |
| **Post-Condition** | The customer successfully navigates the menu and views the required information. |
| **Author** | BC230424569 |
| **Exception** | If the customer enters an invalid category, the chatbot prompts them to choose from the available categories. |

|  |  |
| --- | --- |
| **Use Case Title** | Process Order |
| **Use Case Id** | UC-004 |
| **Description** | The chatbot processes the customer’s food order by selecting dishes and confirming the details. |
| **Action** | 1. Customer selects items from menu.  2. Chatbot confirms the selected items and processes the order. |
| **Alternate Path** | If items are unavailable, chatbot suggests alternatives. |
| **Pre-Condition** | Customer must have access to the menu, and the chatbot must be connected to the order processing system. |
| **Post-Condition** | The order is successfully placed and confirmed. |
| **Author** | BC230424569 |
| **Exception** | 1. The order cannot be placed due to system failure or unavailable menu items.  2. The Customer cancels the order before confirmation. |

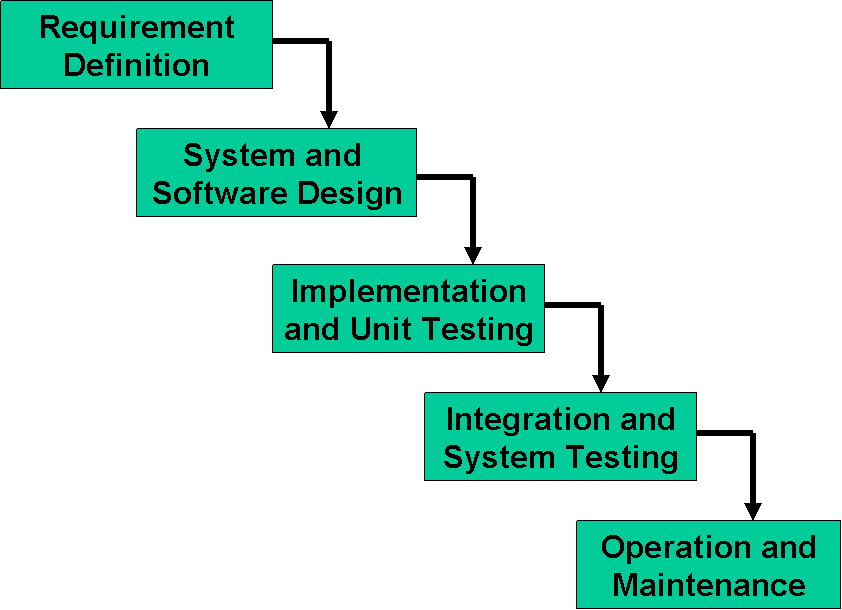
|  |  |
| --- | --- |
| **Use Case Title** | Answer FAQs |
| **Use Case Id** | UC-005 |
| **Description** | The chatbot answers frequently asked questions about restaurant details like hours, location, and services. |
| **Action** | 1. Customer asks question about restaurant details.  2. Chatbot retrieves and provides the required information. |
| **Alternate Path** | If the question is not recognized, chatbot asks for more details. |
| **Pre-Condition** | The chatbot must be able to recognize and respond to frequently asked questions. |
| **Post-Condition** | The customer receives accurate answers to their questions. |
| **Author** | BC230424569 |
| **Exception** | Chatbot fails to retrieve or provide the requested information. |

Adopted Methodology

For my project, I have chosen a combination of the **Waterfall model and the Incremental model.** This hybrid methodology ensures a structured and sequential progression while allowing flexibility in development phases. Every phase is critically reviewed, and successful completion is required to advance to the next phase.

**Waterfall model:**

The Waterfall Model, the first process model introduced, is a linear-sequential life cycle model that is simple to understand and implement. In this model, each phase must be completed before proceeding to the next, ensuring no overlap between phases.



**Incremental Model:**

The Incremental Model is a software development process where requirements are divided into multiple standalone modules, each undergoing the phases of requirements, design, implementation, and testing.With each iteration, new functionality is added to the previous release, and the process continues until the complete system is developed.

**Implementation, integration**

**Deliver to client**

**Design**

**Specification**

**Implementation, integration**

**Deliver to client**

**Design**

**Specification**

**Implementation, integration**

**Deliver to client**

**Design**

**Specification**

**Build 1**

**Implementation, integration**

**Deliver to client**

**Design**

**Specification**

**Build 2**

**Build 3**

**Build n**

**Specification team**

**Design team**

**Implementation, integration team**

### Phase-Wise Explanation of the Methodology

**Phase 1: SRS (Software Requirements Specification)**

* **Waterfall Approach:** In this initial phase, the scope, functional and non-functional requirements, use case diagrams, and usage scenarios are documented in detail. The SRS document must be reviewed and approved by the supervisor to proceed further.
* **Reason:** A clear and structured SRS provides a solid foundation and minimizes ambiguities in the later stages.

**Phase 2: Design Document**

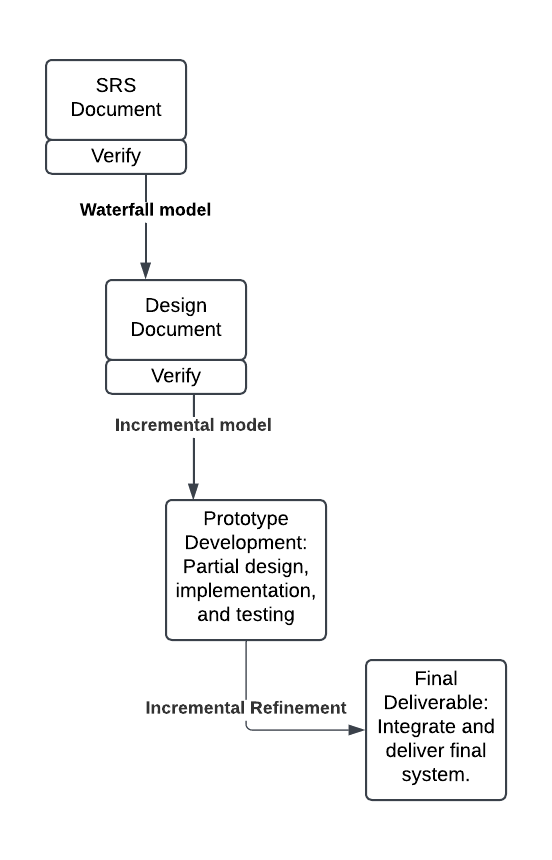
* **Waterfall Approach:** This phase involves creating detailed designs, including ER diagrams, class diagrams, architecture design, and interface diagrams. The complete design document is reviewed and approved by the supervisor before moving forward.
* **Reason:** Finalizing the design ensures that the prototype development has a well-defined structure to follow.

**Phase 3:** **Prototype Development**

* **Incremental Approach:** In this phase, the system is developed incrementally as a working prototype. Each increment represents a functional part of the system. The prototype must be reviewed and approved as a complete phase before proceeding to the final deliverable.
* **Reason:** Using the Incremental model here allows testing and refining functionalities in smaller parts, ensuring a better-aligned and functional final deliverable.

**Phase 4: Final Deliverable**

* **Incremental Approach:** The final system is developed and refined in increments based on feedback from the prototype phase. This phase incorporates any necessary adjustments or enhancements identified earlier. Passing this phase is crucial to ensure the system meets all requirements and expectations.
* **Reason:** Incremental refinement ensures that the system is robust, user-friendly, and meets the project's goals effectively.

**Waterfall+Incremental model**

The combination of **Waterfall + Incremental model** is an ideal methodology for my project. It ensures a structured, sequential progression in the initial phases (SRS and Design) while allowing iterative development and improvement in the later phases (Prototype and Final Deliverable). Mandatory phase approvals guarantee quality at every step, making this hybrid approach both practical and efficient.

Work Plan (Use MS Project to create Schedule/Work Plan)

