**NLP Chatbot Development using Dialogflow**

**Design Document**

**Version 1.0**



**Group Id: F24PROJECT38A67**

**Supervisor Name : Abdullah Qamar**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date (dd/mm/yyyy)** | **Version** | **Description** | **Author** |
| 26/02/2025 | 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. [Introduction of Design Document](#One)
2. [Entity Relationship Diagram (ERD)](#ERD)
3. [Sequence Diagrams](#Six)
4. [Architecture Design Diagram](#Seven)
5. [Class Diagram](#class)
6. [Database Design](#databasedesign)
7. [Interface Design](#interfacedesign)
8. [Test Cases](#testcases)

1. Introduction of Design Document

**Introduction of Design Document**

The **Design Document** is a critical phase in the software development process that defines the structure and architecture of the system. It acts as a blueprint that outlines how the system will be constructed, what components will be included, how these components will interact, and how they will be implemented. The purpose of this document is to ensure that all stakeholders—developers, project managers, and testers—have a clear understanding of the system before it is built, reducing errors and increasing the feasibility and efficiency of the project.

In this document, we have included the following key elements:

* **Entity Relationship Diagram (ERD)**: This diagram illustrates how data entities are related within the system, providing an understanding of how information flows through the system and is stored in the database.
* **Class Diagram**: A representation of the structure of the system's classes, their attributes, methods, and relationships. This diagram provides insight into the logical organization of the system and how various components interact with each other.
* **Sequence Diagrams**: These diagrams depict the interactions between the system components over time. They show how different entities exchange messages and how the system processes requests.
* **Architecture Design**: This visual representation of the system’s overall architecture outlines the components of the system and how they are connected to ensure efficient communication and data flow.
* **Database Design**: It provides a detailed plan for how data will be stored in the database, including tables, fields, relationships, and any constraints. This section ensures that the database is structured properly to handle the necessary data and operations.
* **Interface Design**: Describes the layout and user interface (UI) of the system, ensuring that the system is user-friendly, intuitive, and functional.
* **Test Cases**: A set of test cases that will validate the functionality of the system. These test cases ensure that the system behaves as expected and that the requirements are met.

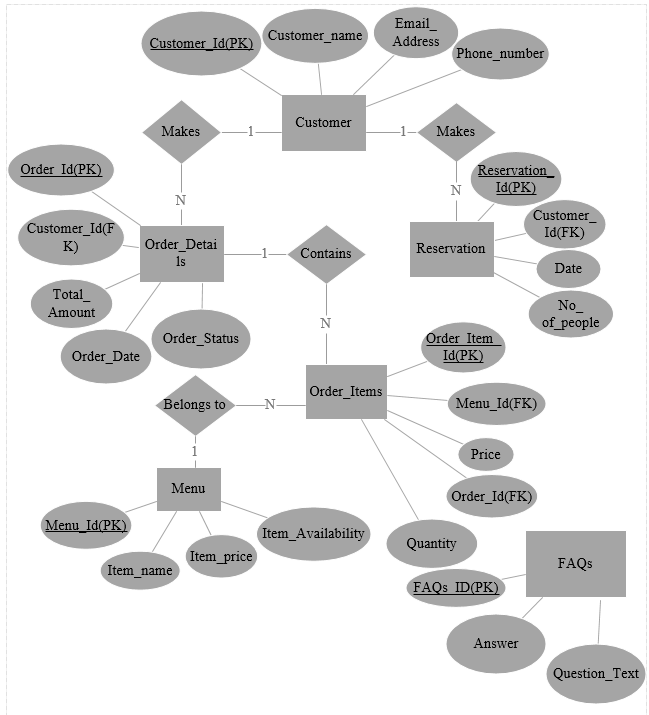
**Benefits of this Design Phase:**

* **Clarity and Direction**: Provides a clear understanding of the system’s architecture and design, ensuring all team members are aligned.
* **Error Prevention**: Helps prevent errors during the development phase by providing a detailed, well-documented design.
* **Improved Communication**: Ensures better communication between stakeholders by providing a common reference point for the system design.
* **Scalability and Flexibility**: Helps in designing a system that can be scaled or modified as needed without major rework.
* **Efficient Development**: A detailed design enables the development process to proceed smoothly and efficiently, reducing the risk of costly mistakes.

**Purpose of the Design Document:**

The purpose of the Design Document is to lay out the foundation for the development phase. By providing a detailed description of the system architecture, components, and their relationships, the document ensures that the development process is efficient and aligned with the project’s objectives, timelines, and budget. It serves as a reference point throughout the development and testing phases, helping to maintain consistency and quality in the final product.

1. Entity Relationship Diagram (ERD) (To be developed using Microsoft Visio or any other drawing software of your choice)



1. Sequence Diagrams (To be developed using Rational Rose or any other drawing software of your choice)



1. Architecture Design Diagram



1. Class Diagram

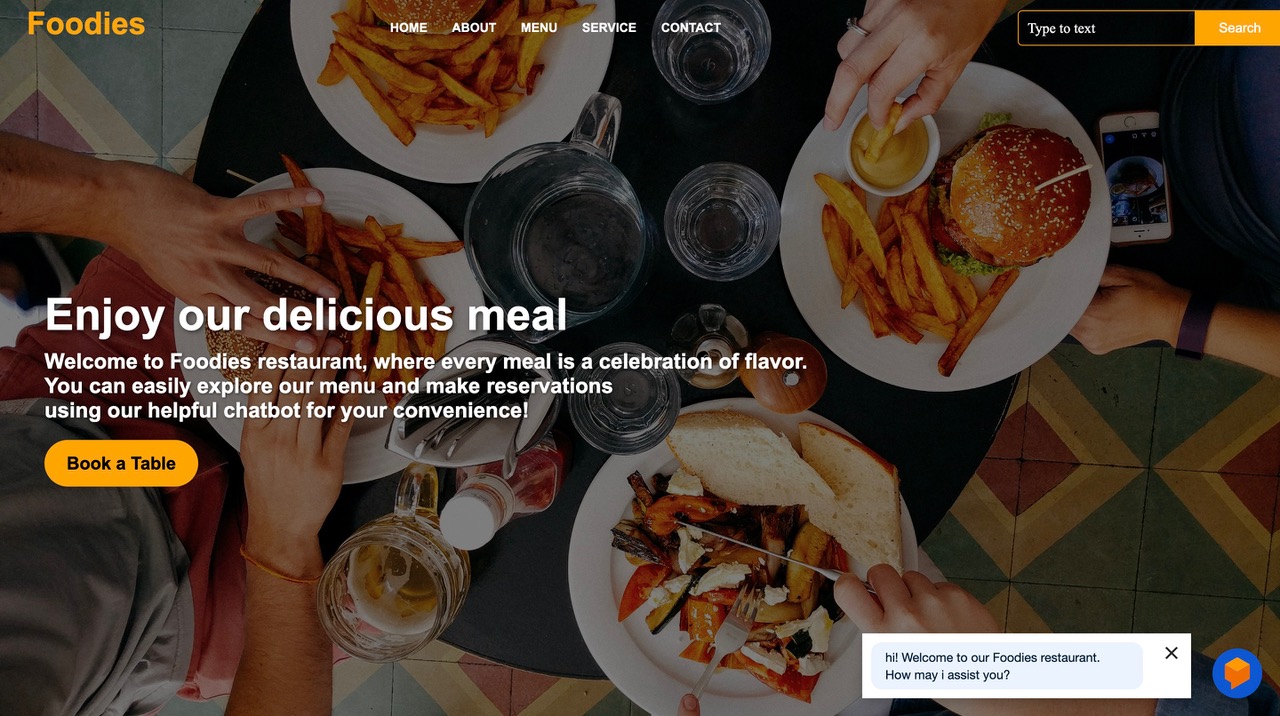


1. Database Design

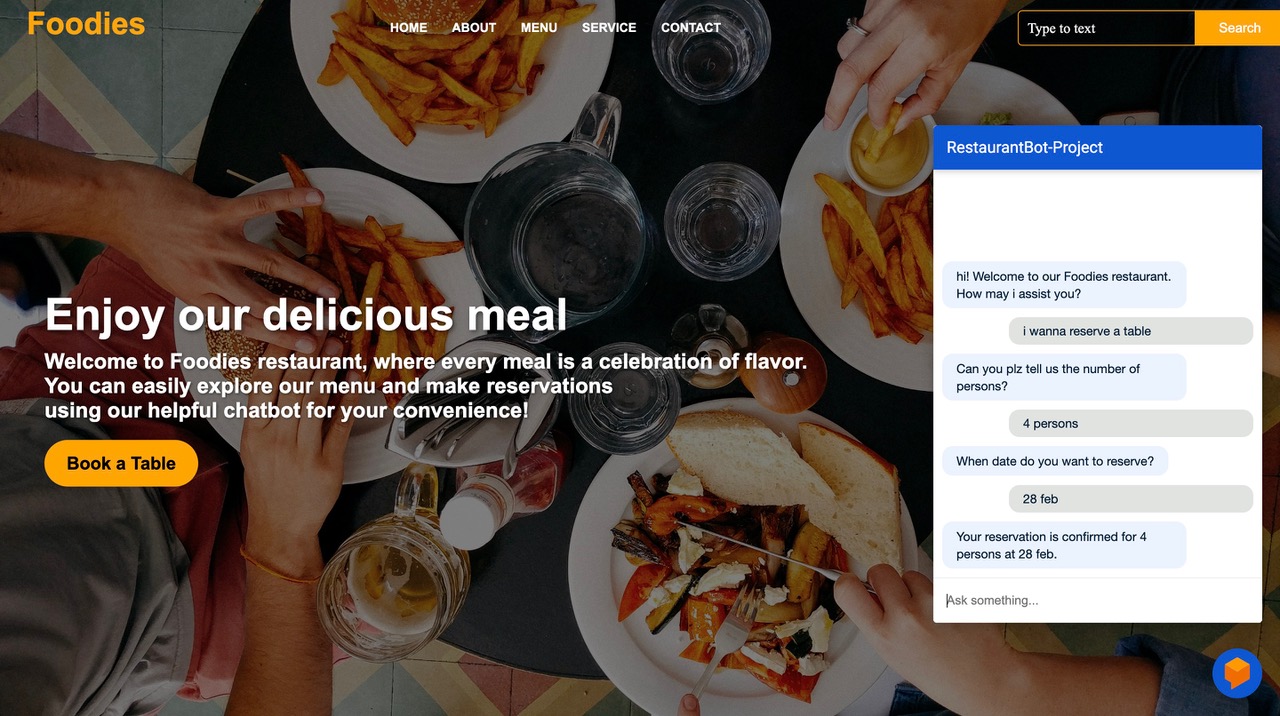


1. Interface Design

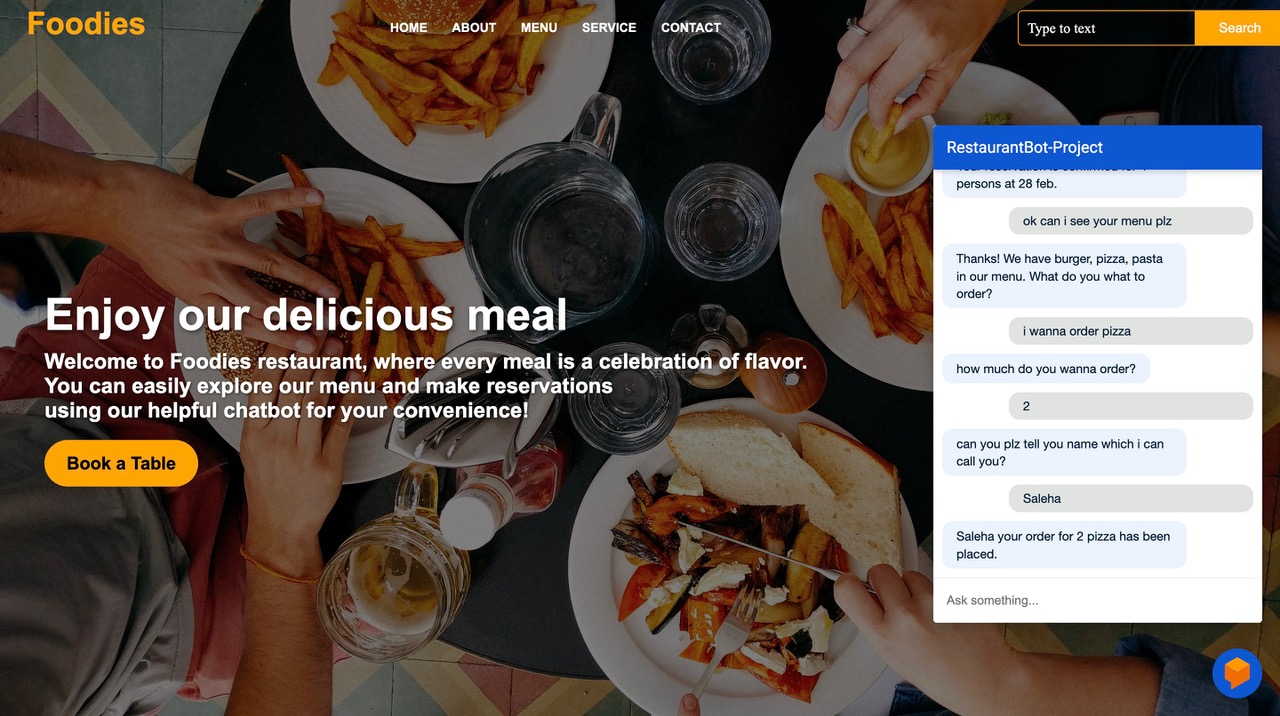
**First Screenshot:**



**Second Screenshot:**



**Third Screenshot:**



1. **Test Cases**

**Test Case #01: Verifying Chatbot's Response to User Inquiries**

|  |  |
| --- | --- |
| **Preconditions:** | * The user has initiated a conversation with the chatbot. * The system is successfully connected to the backend. |
| **Actions:** | * The user submits a query such as: "What are the restaurant hours?" * The chatbot processes the query. |
| **Expected Results:** | * The chatbot should provide an accurate and relevant response, for example: "Our restaurant operates from 10 AM to 10 PM every day." * The system should generate the response without any errors. |
| **Tested by:** | BC230424569 |
| **Result:** | Pass |

**Test Case #02: Verifying Reservation Functionality**

|  |  |
| --- | --- |
| **Preconditions:** | * The user is interacting with the chatbot interface. * The user has access to the reservation system. |
| **Actions:** | * The user requests to make a reservation, for example: "I want to reserve a table for four at 7 PM tomorrow." * The chatbot processes the reservation and provides confirmation. |
| **Expected Results:** | * The chatbot should confirm the reservation, e.g., "Your table for 4 at 7 PM tomorrow is successfully reserved." * The system should save the reservation information. |
| **Tested by:** | BC230424569 |
| **Result:** | Pass |

**Test Case #03: Verifying Display of Reservation Details**

|  |  |
| --- | --- |
| **Preconditions:** | * The user has already completed a reservation. * The system has stored the reservation details correctly. |
| **Actions:** | * The user inquires about the details of their reservation, such as: "What time is my reservation?" * The chatbot retrieves the relevant reservation information and displays it. |
| **Expected Results:** | * The chatbot should display the reservation details, e.g., "Your reservation for 4 people is confirmed at 7 PM tomorrow." |
| **Tested by:** | BC230424569 |
| **Result:** | Pass |

**Test Case #04: Verifying Menu Navigation Feature**

|  |  |
| --- | --- |
| **Preconditions:** | * The user has accessed the chatbot interface. * The system is equipped with an up-to-date menu. |
| **Actions:** | * The user asks to see the menu, for instance: "Can you show me the menu?" * The chatbot displays the list of available items along with their prices. |
| **Expected Results:** | * The system correctly displays the menu with item names and prices as expected. |
| **Tested by:** | BC230424569 |
| **Result:** | Pass |

**Test Case #05: Verifying Order Placement Feature**

|  |  |
| --- | --- |
| **Preconditions:** | * The user has accessed the menu. * The user is logged into the chatbot interface. |
| **Actions:** | * The user selects an item from the menu (e.g., "I’d like to order a pizza"). * The chatbot processes and confirms the order. |
| **Expected Results:** | * The chatbot confirms the order, e.g., "Your pizza has been ordered successfully." * The system updates the order details accordingly. |
| **Tested by:** | BC230424569 |
| **Result:** | Pass |

**Test Case #06: Verifying Modify or Cancel Order Functionality**

|  |  |
| --- | --- |
| **Preconditions:** | * The user has already placed an order. * The system supports modifications or cancellations of orders. |
| **Actions:** | * The user requests to modify or cancel the order, e.g., "Can I change my pizza order?" * The chatbot confirms the change or cancellation. |
| **Expected Results:** | * The chatbot should allow modifications or cancellations and update the system accordingly. * The chatbot confirms the change, e.g., "Your pizza order has been updated." |
| **Tested by:** | BC230424569 |
| **Result:** | Pass |

**Test Case #07: Verifying FAQs Responses**

|  |  |
| --- | --- |
| **Preconditions:** | * The user has opened the chatbot interface. * The system contains a set of predefined FAQs. |
| **Actions:** | * The user asks a frequently asked question, e.g., "What time do you close?" * The chatbot provides the relevant FAQ answer. |
| **Expected Results:** | * The chatbot should provide the correct response to the FAQ, and the reply should be accurate and timely. |
| **Tested by:** | BC230424569 |
| **Result:** | Pass |

**Test Case #08: Verifying Handling of Invalid Input**

|  |  |
| --- | --- |
| **Preconditions:** | * The user is interacting with the chatbot. * The user enters an unclear or invalid query. |
| **Actions:** | * The user inputs an ambiguous or incorrect query, e.g., "bljahsldksld" or "Can I order food?" without providing enough context. * The chatbot asks the user for clarification or rephrasing. |
| **Expected Results:** | * The chatbot should respond with a clarification request, e.g., "Sorry, I did not understand. Could you please rephrase?" |
| **Tested by:** | BC230424569 |
| **Result:** | Pass |