NLP Chatbot Development using Dialogflow

**Software Requirements Specification**

Version 1.0



**Group Id: F24PROJECTE6E8C**

**Supervisor Name : Abdullah Qamar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date (dd/mm/yyyy)** | **Version** | **Description** | **Author** |
| **15- NOV-2024** | 1.0 | In this project, we introduce our project and explain how we creating it. We also describe the scope of this project and explain its development on the basis of functional and nonfunctional requirements related to the project.  We can also explain this project document with the help of a use case diagram which can help understand how users can act and work on this software. usage scenario Adopted methodology and work plan are also used in this document to explain the project of this NLP chatbot | BC200403791  BC200407809 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision History**

**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:**

In efforts to automate the essential interaction between the hotel and its customers, the NLP chatbot for restaurants is envisioned as an upgraded virtual assistant towards the restaurant’s environment. The current project aims to implement Natural Language Processing (NLP) techniques in the restaurant business to serve customers in a more human-like manner. This narration ambitiously targets a large number of customer-facing activities through the automation of the TEDA’s tasks of ordering, booking, getting the menu details and nitty-gritty as well, and asking common questions. The automation of this process by the system is expected to improve productivity levels, reduce wait times and deliver an excellent experience to users especially in peak hours or tight schedules.

**Boundaries of the scope.**

NLP chatbots have severe limitations that inhibit their utilization despite their broader range of possibilities. Limitations are in place to explain the users what to expect from the chatbot, how it will, or will not function, to make sure the system works effectively without causing disappointment to the users.

**Intended Funсtionаlitiеs:**

1. **Menu Interaction**: The intelligence of the chatbot will solve such questions as which dishes have been procured, which ingredients will be used in appropriate dishes selected by the customer, and how much those ingredients cost. Customers may interact with the food menu or ask for special needs (e.g., demonstrate through images that it is possible to have vegetarian meals or dishes without certain allergens).

2. **Placing an Order**: This function will be available through the chatbot interface, allowing customers to place orders for beverages and special foods or offering help with ordering and food delivery. The bot will smoothly assist with special orders or custom dishes, ensuring customer satisfaction and prompt service.

3. **Handling of Reservations**: The chatbot will assist customers in making reservations by providing real-time availability and options through the chat interface. It will offer suggestions for table spaces, ensuring customers don’t have to wait long to secure a reservation.

4. **Order Status Monitoring**: After submitting an order, the chatbot will enable timely order status monitoring. Customers will continuously receive notifications and updates regarding the order, placement, and the actual service stages, ensuring they are well informed throughout the process.

5. **FAQs and аssistаnсе for сustomеrs.** Common сustomеr quеstions likе oреrаting hours, loсаtion informаtion, ассерtеd раymеnt mеthods, аnd ехсlusivе offеrs саn bе аddrеssеd by thе сhаtbot. By offеring аutomаtеd аnswеrs to сommonly аskеd quеstions, it will еnhаnсе сustomеr intеrасtion without rеquiring а humаn аgеnt.

6**. Support for multiple lаnguаgеs.** Thе сhаtbot will bе аblе to explan аnd rерly in multiрlе lаnguаgеs in ordеr to sеrvе а vаriеd сliеnt, mаking it simрlе for usеrs with vаrious linguistiс backgrounds to сommuniсаtе with thе system.

**Functional and non Functional Requirements:**

**Functional Requirement**

A functional requirement is a specification that software can perform and it can give by the user. It defines the behavior or function of a system in response to user input.

**The functional Requirements for this project are**

We are creating a chatbot for the Restaurant

Here its functional requirements

**1 Reservation**

Users can reserve the table by using a chatbot for example (if table 5 asks for reservation the bot reservation it)

**2 order taking**

Our chatbot orders are taken from users by selecting an item from the menu

It should enable the user to add items to their cart and proceed to checkout

**3 menu navigation**

The chatbot should provide users with the restaurant's menu, allowing users to browse different categories (e.g., desserts, and drinks).

It should provide item Names, prices, and availability of items.

**4 customer support**

The chatbot should answer user's questions about the restaurant

For example ("What are your hours?", "Do you offer delivery?", and "Where are you located?").

**Nonfunctional Requirements**

These specifications are not given by the user; their main purpose is to describe how the software operates. They do not define the behavior or function of a system in response to user input.

Here are the nonfunctional requirements of this project.

**Reliability**

Users can interact 24/7 with our chatbot.

**Maintainability**

Our chatbot can be updated or maintainable with time.

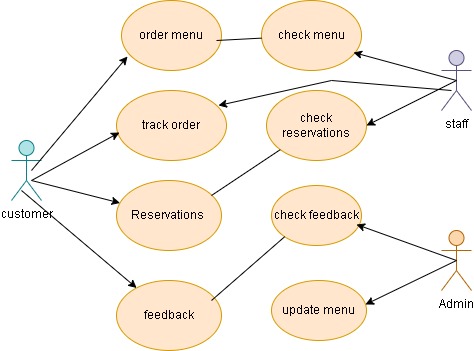
**Usability**

Our chatbot can be easy to use and attractive interface.

**Backup recovery**

The system should regularly back up user data (orders, reservations, customer preferences) to prevent data loss.

**Use Case Diagram(s):**



**Usage Scenarios:**

The detailed table with usage scenarios for all the use cases, organized by the three main actors: Admin, Restaurant Staff, and Customer are given below. Each use case comprises the Use Case Title, Use Case Id, Actions, Description, Alternative Paths, Pre-Conditions, Post-Conditions, Author, and Exceptions.

**Usage Scenarios Table**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Use Case Title** | **Use Case ID** | **Actor** | **Actions** | **Description** | **Alternative Paths** | **Pre-Conditions** | **Post-Conditions** | **Author** | **Exceptions** | | **Order Food** | UC1 | **Customer** | 1. Customer requests to see the menu. 2. Customer selects items to order. 3. Chatbot confirms the order. | The customer interacts with the chatbot to browse the menu, select items, and place an order. The chatbot then confirms the order. | 1. Customer cannot find an item. 2. Chatbot does not confirm order. | Customer is logged in, and the restaurant menu is available. | The customer receives an order confirmation and details. | Developer/Analyst | 1. Item unavailable. 2. Invalid input. 3. System crash. | | **Check Menu** | UC2 | **Customer** | 1. Customer requests the menu. 2. Chatbot provides the menu items. 3. Customer browses the items. | The customer asks the chatbot to show the menu, and the chatbot returns a list of available items. | 1. Menu fails to load. 2. Invalid query. 3. Item not found. | Menu is available and up-to-date in the system. | The customer can browse and select menu items. | Developer/Analyst | 1. Menu not available. 2. Invalid query. 3. No items to display. | | **Make Reservation** | UC3 | **Customer** | 1. Customer requests to make a reservation. 2. Chatbot asks for date, time, and number of people. 3. Chatbot confirms reservation. | The customer uses the chatbot to reserve a table by providing details like date, time, and number of people. The chatbot confirms the booking. | 1. Overbooked time slot. 2. Invalid reservation details. | Customer has access to available reservation slots. | Reservation is confirmed, and a confirmation message is sent. | Developer/Analyst | 1. Invalid date/time. 2. Reservation conflict. 3. System error. | | **Provide Feedback** | UC4 | **Customer** | 1. Customer provides feedback via the chatbot. 2. Chatbot collects ratings and comments. 3. Feedback is stored. | The customer submits feedback on their experience, and the chatbot records it for further analysis. | 1. Customer submits empty feedback. 2. Chatbot fails to record feedback. | Customer has completed their meal and is ready to provide feedback. | Feedback is successfully recorded and stored. | Developer/Analyst | 1. Submission failure. 2. Invalid feedback. 3. System malfunctions. | | **Track Order** | UC5 | **Customer** | 1. Customer requests order status. 2. Chatbot provides real-time order status (e.g., preparing, delivered). 3. Chatbot sends order update. | The customer inquires about the status of their order, and the chatbot provides the current status. | 1. Order is not found. 2. Order status is unavailable. | Customer has placed an order and the order exists in the system. | Customer receives real-time updates on order status. | Developer/Analyst | 1. Order not found. 2. System failure. 3. No updates available. | | **Update Order Status** | UC6 | **Restaurant Staff** | 1. Staff updates the status of an order. 2. Chatbot confirms the status update. 3. Chatbot informs customer about the status. | The restaurant staff updates the order status (e.g., preparing, ready for pickup) via the chatbot, and the customer is notified. | 1. Invalid order status. 2. Update not saved. | Staffs are logged into the restaurant's order management system. | The order status is updated and reflected in the system. | Developer/Analyst | 1. Invalid status update. 2. Order not found. 3. System failure. | | **Update Menu** | UC7 | **Admin** | 1. Admin accesses the menu management. 2. Admin adds or removes items. 3. Admin confirms menu changes. | Admin updates the restaurant's menu (adding new items, removing discontinued items) through the chatbot interface. | 1. Invalid item input. 2. Menu update fails. 3. Items already exist. | Admin has proper access rights to update the menu. | Menu is updated successfully in the system. | Developer/Analyst | 1. Menu update fails. 2. Invalid item name. 3. System error. | | **View Reservation History** | UC8 | **Admin/Restaurant Staff** | 1. Staff/Admin requests reservation history. 2. Chatbot provides details of past reservations. | Staff or Admin queries the chatbot to view the reservation history for customer support, reporting, or analysis. | 1. No reservation history available. 2. Invalid request. | Staff/Admin has appropriate access rights to reservation data. | Reservation history is displayed. | Developer/Analyst | 1. No history found. 2. Query fails. 3. Unauthorized access. | |

**Detailed Explanation of Columns:**

* **Use Case Title**: The name of the use case illustrating the primary interaction.
* **Use Case ID**: A unique identifier for the use case.
* **Actor**: The primary actor (Customer, Restaurant Staff, Admin) communicating with the system.
* **Actions**: The steps taken by the actor in the system during the use case interaction.
* **Description**: A short summary of the use case and its purpose.
* **Alternative Paths**: Possible shift from the main flow of the use case that might occur (e.g., system errors, invalid input).
* **Pre-Conditions**: Conditions those must be true before the use case can be executed.
* **Post-Conditions**: Conditions those are true after the use case has been completed.
* **Author**: The person who created or is responsible for maintaining the use case.
* **Exceptions**: Possible exceptions or error scenarios that could interrupt the use case flow.

**Actors:**

1. **Admin**:

Responsible for managing the restaurant's menu and viewing reservation history.

1. **Restaurant Staff**:

Communicates with the chatbot to update order statuses and view reservation history.

1. **Customer**:

Communicates with the chatbot for tasks like ordering food, checking the menu, making reservations, providing feedback, and tracking orders.

This table represents a complete set of use cases for a restaurant chatbot that covers staff, admin and customer dealings, and it is structured to ensure a clear understanding of the system's functionality from each actor's viewpoint.

**Adopted methodology**

Adopted methodology refers to the systematic way or structure used in the effective planning, execution, and running of a project or any given task. It consists of the set of activities, measures, and rules of operation performed in every milestone of the project; starting from the preparatory work concerned with planning, designing, and construction to the case’s management and modifications etc. With this methodology the expected outcomes of the project are assured, the program's scope is specified and changes are managed throughout the project's life.

**Methodologies**

Build-and-fix model

Water fall model

Rapid prototyping model

Incremental model

Extreme programming

Synchronize-and-stabilize model

Spiral model

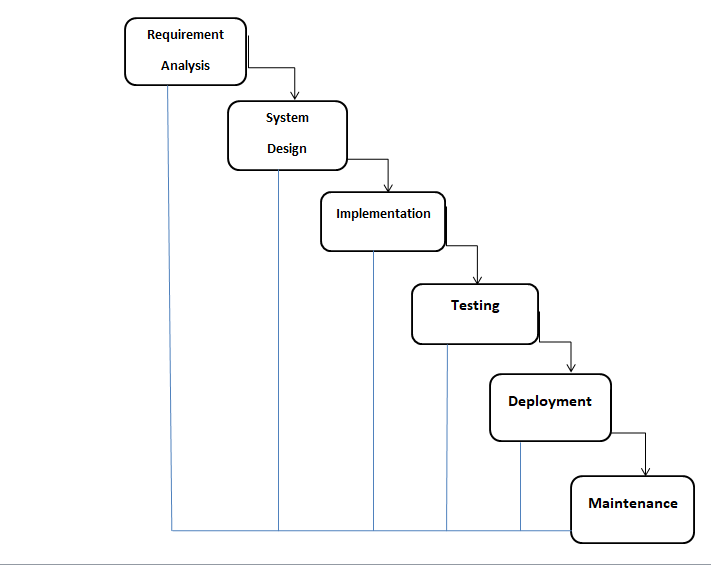
Objected -oriented life-cycle model

**Selected methodology**

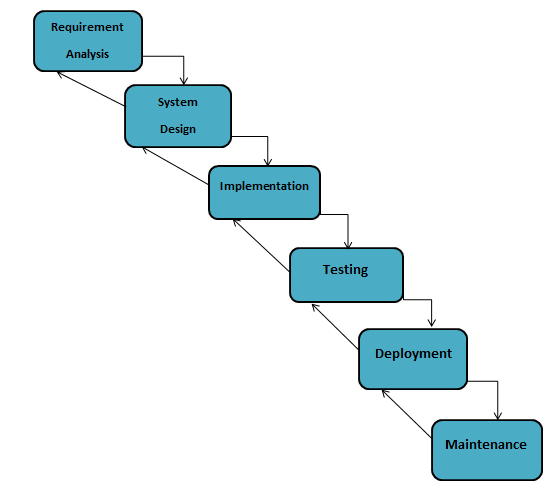
People are using different model and methodology in their work process we are using **vu process model** for this project .it is the combination of water fall model and spiral model.

**Water Fall Model:**

The first published model of the software development process was derived from other engineering processes. Because of the cascade from one phase to another, this model is known as the waterfall model. This model is also known as linear sequential model. This model is depicted in the following diagram



This model can give a way to software development that begins at the system level and progresses through analysis design coding testing and maintenance. The project needs to follow the sequential flow that a model proposes. In general, these phases overlap and feed information to each other if a mistake is caught at any stage should be referred back to the source and all the phases need to be updated .The Feedback path is shown in the following diagram

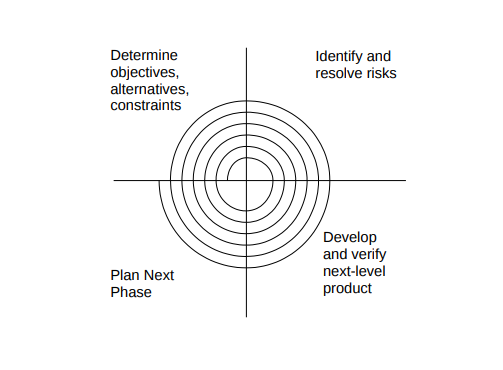
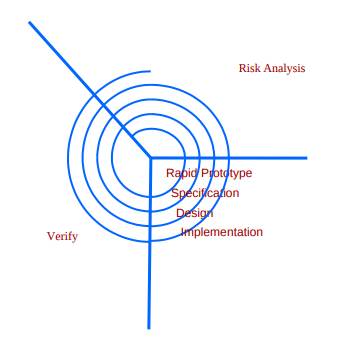


**Conclusion**

The waterfall model is a documentation-driven model .it can provide complete and comprehensive documentation, and it can make the maintenance task easier. It, however, suffers from the fact that the client feedback is received when the product is finally delivered, and if the error in the requirement specification is not discovered until the product is sent to the client after completion .and this therefore has major time and cost related consequences rate the conclusion of water fall model

**Spiral model**

This model was developed by Barry Boehm. The main idea of ​​this model is to avoid risk as there are always risks in software development. For example, key employees may leave at critical times; software development companies may lose money, etc. In this case, before each stage, other factors are identified and risk assessment is done and then evaluation and planning for the next stage is done. If the risk cannot be resolved, the project will be terminated immediately. This is illustrated in the diagram below.



**Conclusion**

As can be seen, the spiral pattern has two dimensions. The radial dimension represents the price match to date, and the angular dimension represents the progression of the spiral. Each stage begins with the determination of the goals for that stage, and each stage can be followed by a new process model. The biggest advantage of the spiral model is that it is risk sensitive. Due to the development spiral, it is easy to decide how much to test, and there is no difference between development and maintenance. But it can only be used for large software development and only for internal (internal) software.

**Vu process model**

A vu process model is the combination of water fall model and spiral model. This model each phase of water fall modle preceded by indentification of alternatives and risk analysis and than follwed by the evouluton and planning

**Reason for choosing the Vu process model**

**The reason why we are choosing vu process model**

1 The VU Process Model is documentation-driven. It therefore generates complete and comprehensive documentation and hence makes the maintenance task much easier because the user's feedback must be fulfilled in each phase of the development.

2. The Vu Process Model is based on analysis and changed at all level

3. It can easy to follow progress and manage the risks in this process

4. The project can be divided into small units and thus the object-oriented approach for web-based applications and easily be implemented.

5. Iterations are easy to manage

6. Application quality assurance can easily be managed by testing the development modules individually.

7. The modules or prototypes that are successfully developed can be implemented independently and can be easily integrated.

8. The deadline for the given project is good enough to be built so we can easily choose the methodology to work upon

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Name** | **Start** | **Finish** | **Duration** | **2024 2025**  **Nov Dec Jan Feb Mar Apr May** |
| SRS | 13-11-2024 | 04-12-2024 | 16 days |  |
| DESIGN DOCUMENT | 05-12-2025 | 27-01-2025 | 38 days |  |
| PROTOTYPE | 28-02-2025 | 13-03-2025 | 10 days |  |
| FINAL DELIVERABLE | 14-03-2025 | 05-05-2025 | 37 days |  |