**NLP Restaurant Chatbot Development using Dialogflow**

**Project Report**

**Version 1.0**

**Group Id: F24PROJECT5D341 (BC210402852,BC210424799)**

**Supervisor Name : Abdullah Qamar**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date (dd/mm/yyyy)** | **Version** | **Description** | **Author** |
| 04/12/2024 | 1.0 | The **Restaurant Chatbot** is a cutting-edge conversational AI tool designed to enhance the dining experience by providing efficient and automated interactions with customers. It integrates seamlessly into restaurant operations, offering services such as real-time table reservations, personalized menu navigation, order placements, ultimately driving both customer satisfaction and business efficiency. | (BC210402852, BC210424799) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. **Scope (of the project)**
2. **Functional Requirements Non Functional requirements**
3. **Use Case Diagram**
4. **Usage Scenarios**
5. **Adopted Methodology**
6. **Work Plan (Use MS Project to create Schedule/Work Plan)**

**SRS Document**

Scope of Project:

The **Restaurant Chatbot** project aims to enhance the customer experience by automating key restaurant interactions through a conversational AI interface. Using Dialogflow’s natural language processing capabilities, the chatbot will allow customers to make table reservations, explore the menu, place orders, all in an efficient and intuitive manner. The primary purpose of this system is to streamline customer service, reduce wait times, and improve operational efficiency, while providing a seamless, personalized experience that integrates effortlessly with the restaurant's existing operations.

1. Scope Boundaries
   1. What the system will do:

* **Admin Dashboard:** Provide restaurant administrators with access to manage reservations, view orders, and track chatbot interactions.
* **Table Reservations:** Allow customers to reserve tables based on real-time availability.
* **Menu Navigation:** Provide detailed information on the menu, including dish descriptions, pricing, and recommendations.
* **Order Placement:** Enable customers to place orders for both dine-in and takeaway with real-time order confirmation.
* **FAQ Handling:** Answer common queries regarding restaurant hours, policies, and services.
  1. What the system will not do:
* Handle complex customer issues that require human intervention.
* Provide support for non-restaurant-related inquiries or personal requests outside the scope of the restaurant services.

Functional and non Functional Requirements:

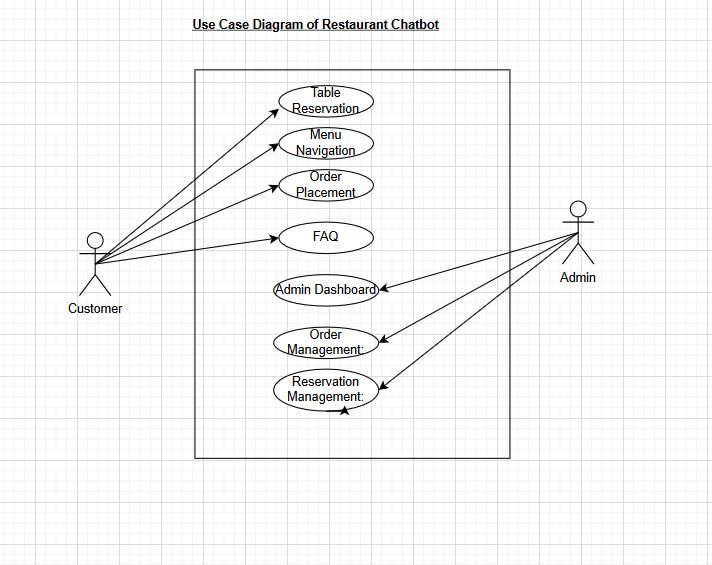
**Functional Requirements:**

1. **Table Reservation:**
   * The chatbot will allow customers to inquire about available tables for specific times and dates. It will confirm reservations or suggest alternatives when necessary.
2. **Menu Navigation:**
   * Customers will be able to ask the chatbot for detailed descriptions of dishes, including ingredients, prices, and available options.
3. **Order Placement:**
   * The chatbot will allow customers to place orders, confirming order details, quantities, and delivery options. It will handle both dine-in and takeaway orders.
4. **FAQ Handling:**
   * The chatbot will address frequently asked questions related to the restaurant's operations, such as hours of operation, contact information, and policies.
5. **Admin Dashboard:**
   * A dedicated admin panel will allow restaurant staff to view reservations, orders, and chatbot logs to manage operations more efficiently.

#### **Non-Functional Requirements:**

1. **Performance:**
   * The chatbot must respond to all user queries within 2 seconds to ensure a smooth and efficient interaction.
2. **Reliability:**
   * The system should maintain 99.9% uptime to ensure that it is available to customers at all times, minimizing service disruptions.
3. **Usability:**
   * The interface must be user-friendly, supporting easy text and button-based interactions for seamless customer engagement.
4. **Scalability:**
   * The system must be scalable, capable of handling up to 500 simultaneous user interactions without performance degradation.
5. **Security:**
   * All customer data information must be encrypted using industry-standard security protocols to protect sensitive user data.
6. **Maintainability:**
   * The system should be easily maintainable, with the ability to quickly update menu items, and chatbot interactions as required.

Use Case Diagram(s):



**Fig.01 use case diagram   
Of   
NLP Restaurant Chatbot**

Usage Scenarios:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Use Case Title | Use Case ID | Actions | Description | Alternative Paths | Preconditions | Postconditions | Author | Exceptions |
| Table Reservation | UC-001 | User requests table reservati-on for a specific date and time | Allows the user to reserve a table for a specific date and time based on real-time availabili-ty. | If the requested time is unavaila-ble, the chatbot suggests alternati-ve time slots. | User provides accurate date/time | Reserva-tion details saved in the database | (BC210402852, BC210424799) | Invalid date/time |
| Menu Navigation | UC-002 | User asks for dish details | Chatbot retrieves informati-on from the database | Suggest similar dishes if the query is vague | Menu database is up-to-da-te | Display dish details | (BC210402852, BC210424799) | Database not updated |
| Order Placement | UC-003 | User places an order for takeaway | Chatbot confirms items, quantity, and estimated time | Modify order if items are unavaila-ble | Menu items are updated in real-time | Order stored in the system | (BC210402852, BC210424799) | Outdated menu data |
| FAQ | UC-004 | User queries restaurant hours | Chatbot provides relevant informati-on | Redirect to human support if query is complex | FAQ database is accurate | Display appropri-ate response | (BC210402852, BC210424799) | Query outside chatbot scope |
| Admin Dashboard | UC-005 | Admin accesses the dashboar-d, views reports, and manages orders and reservati-ons | Admin views a summary of current orders, reservati-ons, and customer interactio-ns. | Admin can filter by date, status, and more. | Admin logged in. | Admin can view and manage orders, reservati-ons, and reports. | (BC210402852, BC210424799) | Unauthor-ized access, Data not updated |
| Order Management | UC-006 | Admin views, modifies, or cancels orders | Admin manages customer orders, viewing details and modifying as needed. | Reject or complete orders. | Admin logged in. | Updated order status. | (BC210402852, BC210424799) | Unauthor-ized access |
| Reservation Management | UC-007 | Admin views, modifies, or cancels reservati-ons | Admin monitors table bookings and updates reservati-on details. | Modify reservati-on date/time. | Admin logged in. | Updated reservati-on details. | (BC210402852, BC210424799) | Overboo-ked scenario |

Adopted Methodology

Waterfall Model:

The **Waterfall Model** is a traditional and sequential software development methodology that progresses through distinct, linear phases. Each phase must be completed before the next one begins, with minimal overlap or iteration. It is one of the earliest models used for software development and is widely recognized for its structured approach.

### **Phases of the Waterfall Model:**

1. **Requirements Gathering and Analysis**:
   * In this phase, all project requirements are collected, analyzed, and documented.
   * Deliverables include a detailed Software Requirements Specification (SRS) document that outlines what the system should do.
2. **System Design**:
   * Based on the requirements, the overall system architecture and design are planned.
   * This includes hardware, software, network, and database specifications, as well as the design of user interfaces and workflows.
3. **Implementation (Coding)**:
   * The actual development of the software begins in this phase.
   * Developers write code based on the design specifications. The project is usually divided into smaller modules, and each is developed individually.
4. **Testing**:
   * Once the software is developed, it undergoes rigorous testing to identify and fix bugs or errors.
   * Functional, integration, system, and user acceptance testing ensure that the software meets the requirements and works as intended.
5. **Deployment**:
   * After successful testing, the software is delivered and deployed in the live environment.
   * This phase may also include installation, configuration, and initial training for users.
6. **Maintenance**:
   * After deployment, the software may require updates, patches, or modifications based on user feedback, operational issues, or evolving needs.

### **Key Features of the Waterfall Model:**

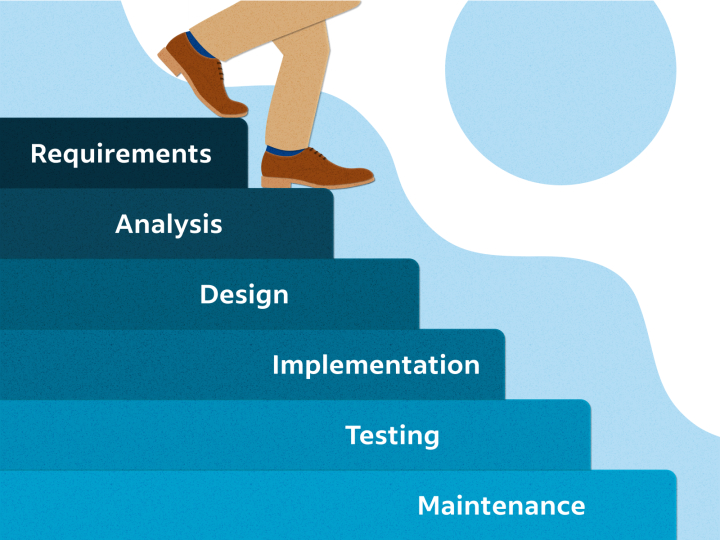
* **Sequential Flow**: Each phase has defined inputs and outputs, and transitions occur only when the current phase is complete.
* **Emphasis on Documentation**: Every phase produces detailed documentation, which acts as a reference for the next phase.
* **Fixed Deliverables**: Milestones are clearly defined, with no overlapping of phases.

### **Advantages of the Waterfall Model:**

1. **Simple and Easy to Understand**:
2. **Structured Approach**:
3. **Clear Documentation**:
4. **Good for Smaller Projects**:
5. **Early Identification of Goals**:
6. **Ease of Management**:

### **Disadvantages of the Waterfall Model:**

1. **Lack of Flexibility**:
2. **Late Testing**:
3. **Risky for Long-term Projects**:
4. **Dependency on Early Stages**:
5. **No Customer Feedback in Between**:
6. **Not Suitable for Complex or Dynamic Projects**:



**Fig.02 Waterfall Model**

Agile Model:

The **Agile Model** is a modern software development methodology that focuses on iterative development, collaboration, and flexibility. Unlike traditional models such as the Waterfall, Agile embraces change and delivers software incrementally, allowing teams to adapt quickly to evolving requirements and customer feedback.

### **Key Principles of the Agile Model:**

1. **Iterative and Incremental Development**:
   * Software is developed in small, manageable pieces called "iterations" or "sprints," each lasting 1-4 weeks.
   * At the end of each iteration, a working piece of software is delivered.
2. **Customer Collaboration**:
   * Continuous interaction with customers or stakeholders ensures the software meets their expectations.
   * Feedback is incorporated into the next iteration.
3. **Flexibility to Changes**:
   * Agile welcomes changes in requirements, even late in the development process, to deliver the best product.
4. **Cross-Functional Teams**:
   * Teams consist of developers, testers, designers, and product owners working closely together to ensure fast delivery and quality output.
5. **Focus on Working Software**:
   * The primary measure of progress is delivering functional, usable software, rather than excessive documentation.

### **Phases of the Agile Model:**

1. **Concept / Initial Planning**:
   * High-level requirements are gathered to define the overall scope and goals of the project.
   * Teams create a **Product Backlog** (a prioritized list of features or tasks).
2. **Iteration Planning**:
   * For each sprint, the team selects items from the product backlog to work on.
   * The **Sprint Backlog** is created with detailed tasks for the iteration.
3. **Iteration Execution**:
   * Teams work on the selected features, which includes coding, testing, and integrating components.
   * Daily **Scrum Meetings** help monitor progress and address challenges.
4. **Demo / Review**:
   * At the end of each sprint, the team presents the working software to stakeholders for feedback.
5. **Retrospective**:
   * Teams reflect on what went well and what could be improved for the next iteration.
6. **Release**:
   * After several iterations, a stable version of the software is delivered to customers.

### **Advantages of the Agile Model:**

1. **Flexibility and Adaptability**:
2. **Faster Delivery**:
3. **Customer Satisfaction**:
4. **Risk Reduction**:
5. **Improved Team Collaboration**:

### **Disadvantages of the Agile Model:**

1. **Less Predictable**:
2. **Requires Experienced Teams**:
3. **Limited Documentation**:
4. **Potential Scope Creep**:
5. **Not Suitable for All Projects**:



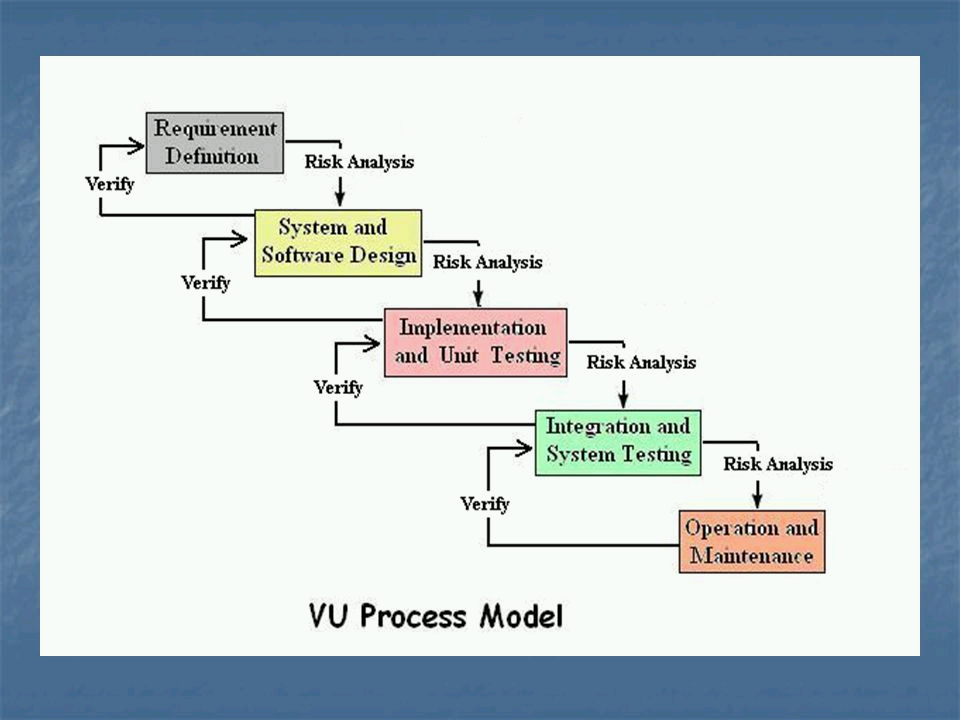
**Fig.03 Agile Model**

VU Process Model:

The **VU Process Model** typically refers to the specific guidelines and approaches outlined by **Virtual University (VU)** for software development projects. While not an official "industry standard," the VU model adapts traditional software engineering principles and methodologies to fit academic project requirements. It combines elements of **Waterfall** and **Agile** models for structured learning and incremental delivery. Here's an overview:

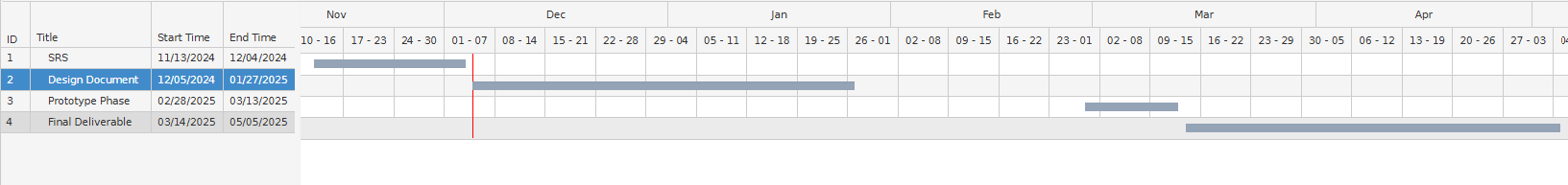
### **VU Process Model Phases:**

1. **Requirement Gathering**:
   * Define the project’s scope.
   * Identify functional and non-functional requirements.
   * Deliverable: **Software Requirements Specification (SRS)** document.
2. **System Analysis**:
   * Perform feasibility analysis (technical, operational, economic).
   * Create use cases and identify system interactions.
   * Deliverable: Use Case Diagram, Context Diagram, and Analysis Report.
3. **System Design**:
   * Create system architecture, data flow diagrams (DFDs), and database schema.
   * Deliverable: **System Design Document (SDD)**.
4. **Development**:
   * Code the system based on the design.
   * Modular development with supervisor reviews at intervals.
   * Deliverable: Working modules of the system.
5. **Testing**:
   * Create test cases and perform unit testing, integration testing, and system testing.
   * Deliverable: **Test Report** and bug log.
6. **Deployment**:
   * Deploy the project in a real or simulated environment.
   * Deliverable: Final working system.
7. **Maintenance** (Optional for Academic Projects):
   * Address potential post-deployment issues.
   * Deliverable: Maintenance Report.

****

**Fig.01 VU Process Model**

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



**Table of Contents**

1. **Introduction of Design Document**
2. **Entity Relationship Diagram (ERD) (To be developed using Microsoft Visio or**

**any other drawing software of your choice)Use Case Diagram**

1. **Sequence Diagrams (To be developed using Rational Rose or any other drawing**

**software of your choice)**

1. **Architecture Design Diagram**
2. **Class Diagram**
3. **Database Design**
4. **Interface Design**
5. **Test Cases**

**Design Document**

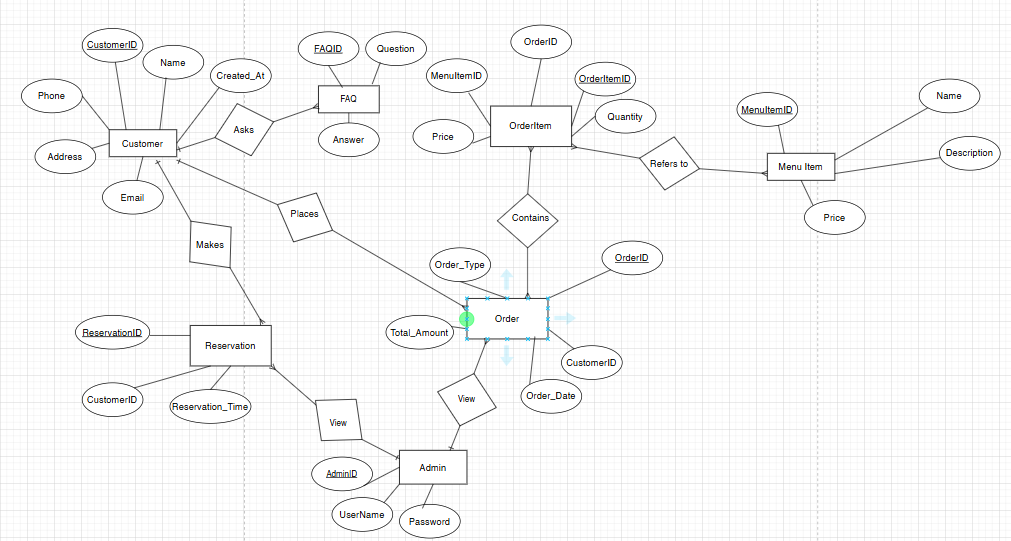
1.Introduction of Design Document:

The Design Document provides a detailed overview of the system’s architecture, including **ERD, Sequence Diagram, Class Diagram, Database, Interface Design, and Test Cases**. It outlines system requirements, subsystem architecture, processing logic, and external interfaces. The document ensures structured software development by capturing key design decisions that impact performance and maintainability. Additionally, it supports software integration and unit testing by defining module interactions and interfaces.

2. Entity Relationship Diagram (ERD) (To be developed using Microsoft Visio or

any other drawing software of your choice)

An **Entity-Relationship Diagram (ERD)** visually represents the structure of a database by illustrating **entities (tables), attributes (fields), and relationships (connections between tables)**. ERDs help in designing and understanding the database schema before actual implementation.

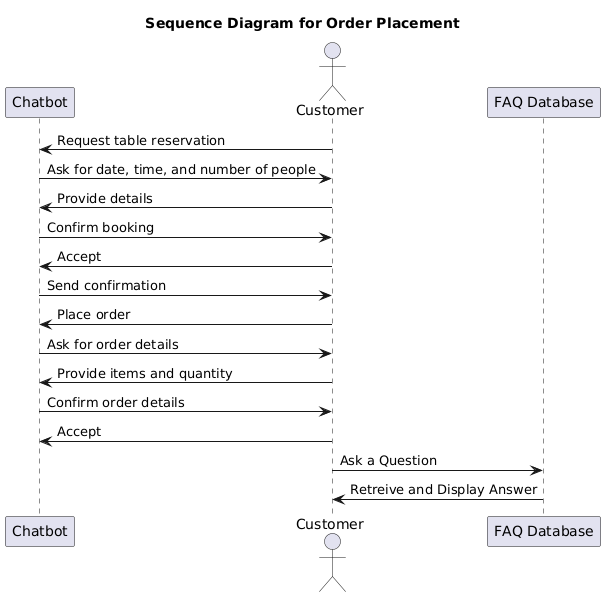


**Fig.01 ERD diagram   
Of   
NLP Restaurant Chatbot**

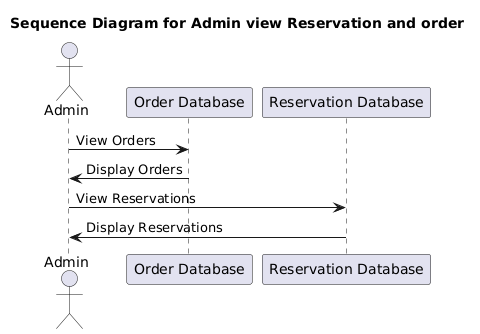
3. Sequence Diagrams (To be developed using Rational Rose or any other drawing

software of your choice)

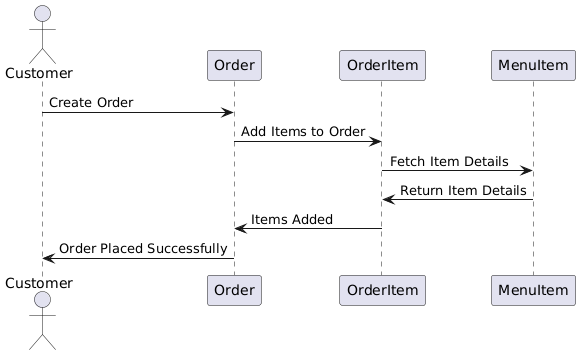
A Sequence Diagram provides clear communication by visually mapping interactions between system components, ensuring better understanding among stakeholders. It enhances system design by identifying inefficiencies, redundancies, and areas for improvement. Additionally, it aids in debugging by allowing developers to trace the sequence of events and pinpoint errors effectively. By serving as a shared reference, it fosters better collaboration between developers, designers, and business teams, ensuring smooth implementation of system workflows.



**Fig.01 sequence diagram   
Of   
NLP Restaurant Chatbot**

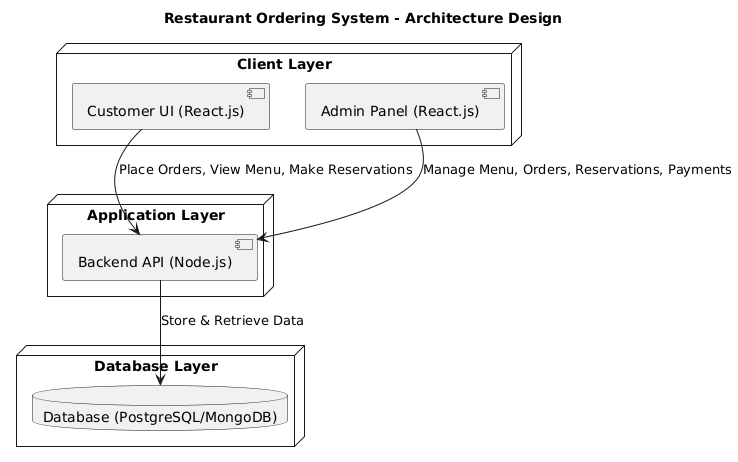
****

**Fig.02 sequence diagram   
Of   
NLP Restaurant Chatbot**

****

**Fig.03 sequence diagram   
Of   
NLP Restaurant Chatbot**

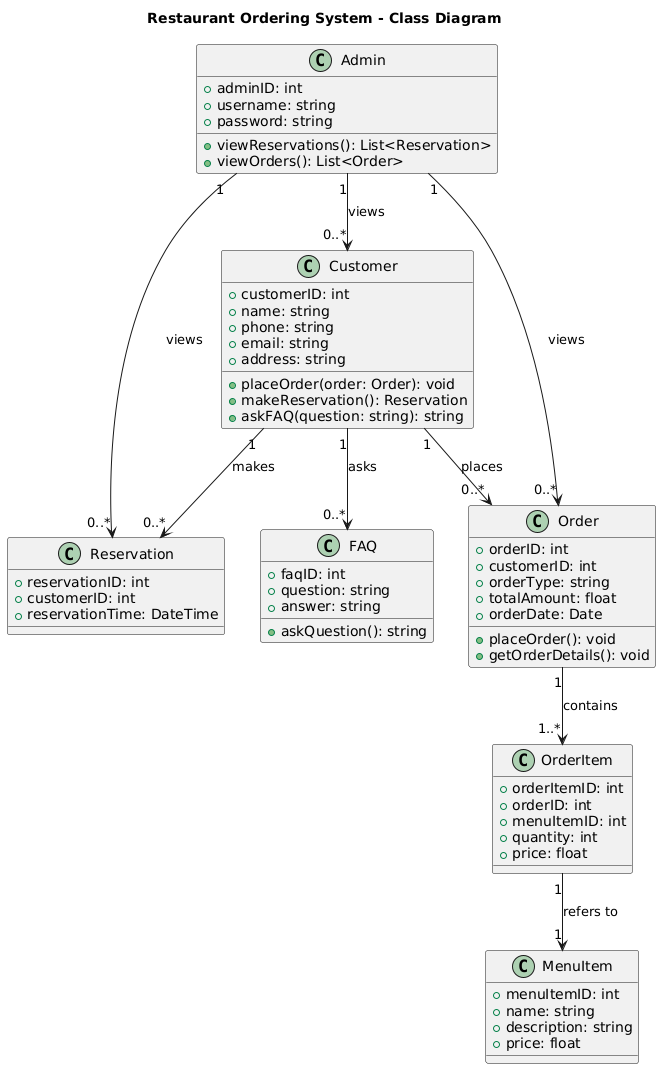
4. Architecture Design Diagram:



**Fig.01 Three-Tier Architecture diagram   
Of   
NLP Restaurant Chatbot**

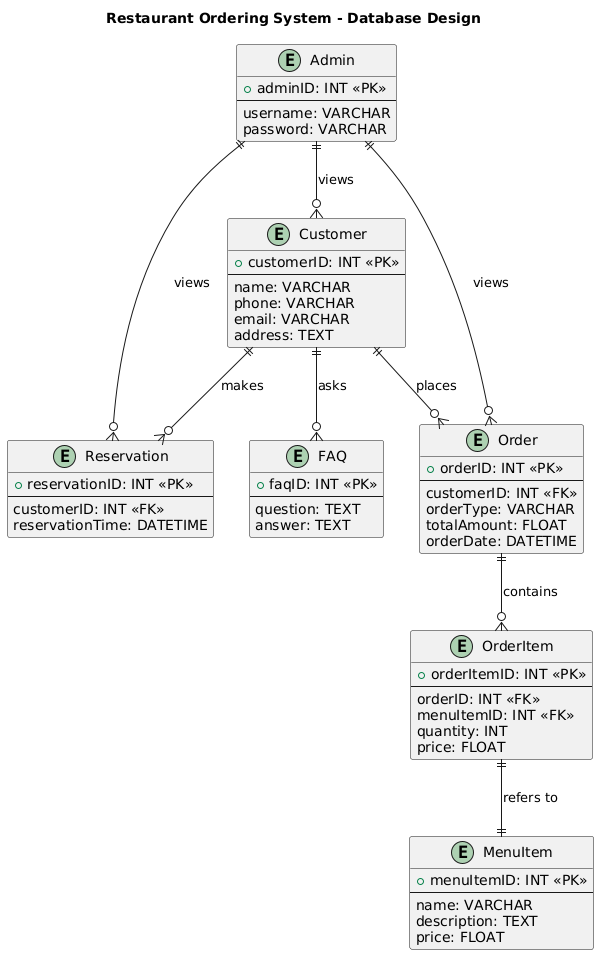
Class Diagram:

A **Class Diagram** defines the structure of a system by representing its classes, attributes, methods, and relationships. It serves as a **blueprint for system design**, ensuring clarity in interactions and dependencies. Developers use it to **guide implementation, debugging, and future modifications**. Additionally, it improves **team communication** by providing a shared visual representation of the system

**Fig.01 Class Diagram   
Of   
NLP Restaurant Chatbot**

Database Design:

A **Database Design** ensures efficient data organization, storage, and retrieval for a system. It helps in maintaining **data integrity, consistency, and scalability** while optimizing performance. A well-structured design **prevents redundancy and ensures security** through proper relationships and constraints. It serves as the **foundation for application development**, enabling smooth data operations.

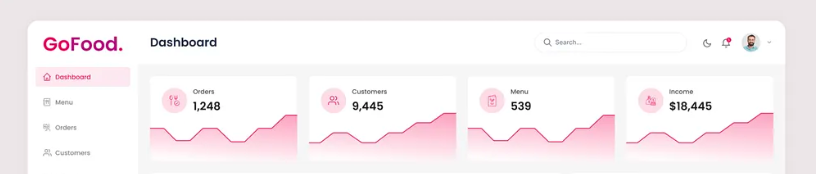


**Fig.01 Database Design Of  
 NLP Restaurant Chatbot**

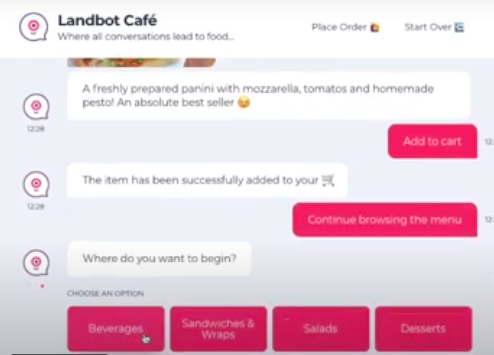
Interface Design:



**Fig.01 Interface Design Of  
 NLP Restaurant Chatbot**



**Fig.02 Interface Design Of  
 NLP Restaurant Chatbot**

****

**Fig.03 Interface Design Of  
 NLP Restaurant Chatbot**

Test Cases:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Precondition** | **Action** | **Expected Results** | **Tested By** |
| TC001 | View Menu Items | Menu items are available | Navigate to the menu page | Menu items are displayed | (BC210402852, BC210424799) |
| TC002 | Place an Order | Customer selects menu items | Select menu items, enter quantity, and confirm order | Order is placed successfully with order ID | (BC210402852, BC210424799) |
| TC003 | Make a Reservation | Available time slots exist | Enter reservation details and confirm | Reservation is successfully created | (BC210402852, BC210424799) |
| TC004 | Admin Login | Admin is registered | Enter valid credentials and login | Admin is redirected to dashboard | (BC210402852, BC210424799) |
| TC005 | Admin View All Customers | Admin is logged in | Navigate to "View Customers" section | List of all customers is displayed | (BC210402852, BC210424799) |
| TC006 | Admin View All Orders | Admin is logged in | Navigate to "View Orders" section | List of all orders is displayed | (BC210402852, BC210424799) |
| TC007 | Admin View All Reservations | Admin is logged in | Navigate to "View Reservations" section | List of all reservations is displayed | (BC210402852, BC210424799) |
| TC008 | Logout | Admin is logged in | Click "Logout" button | Admin is logged out and redirected to login page | (BC210402852, BC210424799) |