Project Charter:

## Executive Summary:

Smart Restaurant System is an unused approach to make better customer experience in restaurant which will assist the restaurant in maintaining its authenticity and transactional activities online for better ease of both customer and restaurant staff. This smart restaurant system will help in modernizing the traditional approach of handling customers orders and provide customer a better choice through its rating feature.

Let’s assume we are customer in a new restaurant with single waiter to handle our request. We are hungry and want to order food fast so that we can fill ourselves up. Unfortunately, the waiter is stuck with a single customer due to miscommunication. But the restaurant has a smart restaurant system. Then, we take the QR provided in the respective table and scan it. The QR will navigate us to the website with the restaurant’s menu option where we can see each items name, image preview, estimated time, price, rating and variations. Suddenly with our hungry instinct our eyes gets caught by the item ‘Whatever Sandwich’, but we have no idea what is in it nor how is it. Looking at its rating it is 5/5 so we press the button with ‘?’ which shows a brief summary about the food. So you decide to order it. We add its quantity and press the order button. Now a payment option pop ups for the payment. We pay the food and the estimated time starts to show for all the food item with its status. The food gets delivered and we get to rate the food based on your experience so that other will have awareness if the food is good.

## Vision:

* To deliver a robust, scalable smart restaurant system for all scale restaurant
* To successfully introduce new customer service processes to the marketplace in restaurant.

## Objectives:

### Technological Objective

1. To modernize traditional restaurant system
2. Help customer have better restaurant experience
3. To assist restaurant owners on improving their food service through reviews

## Scope:

### Functional Scope

#### Order System:

1. **QR code Integration :** Customers can scan a QR code at their table to access the menu and place orders through a web interface.
2. **Menu Display:** Display the restaurant's menu with descriptions, prices, and customer reviews for each item.
3. **Order Customization:** Allows customer to customize their order (for example: specifying cooking preference)
4. **Order Submission:** Submit orders directly to the kitchen for preparation.

#### Billing and Payment:

1. **Bill Generation:** Automatically generate the bill for each table based on the items ordered.
2. **Receipt Generation:** Provide digital or printed receipts for customers.

#### Customer Reviews and Feedback:

1. **Review Submission:** Allow customers to submit reviews and ratings for menu items after their meal.
2. **Review Display:** Display customer reviews on the menu interface to help other customers make informed choices.

#### User Management:

1. **Role-Based Access Control:** Differentiate user roles (e.g., admin, chef, waiter, customer) and provide appropriate access levels.
2. **User Authentication:** Implement login and authentication for restaurant staff and customers (if needed for loyalty programs or reviews).

### Non-functional Scope:

#### Performance Requirements:

1. **Response Time:** Ensure that the system responds to user actions within a specific time frame (e.g., within 2 seconds for most operations).
2. **Scalability:** Design the system to handle a certain number of concurrent users and orders without degradation in performance.

#### Security Requirements:

1. **Data Encryption:** Ensure that sensitive data (e.g., payment information, user credentials) is encrypted in transit and at rest.
2. **User Authentication:** Implement secure authentication mechanisms to protect access to the system.

#### Usability Requirements:

* **Intuitive Interface:** Design a user-friendly interface that is easy to navigate for both customers and restaurant staff.
* **Accessibility:** Ensure the system is accessible to users with disabilities, following guidelines like WCAG (Web Content Accessibility Guidelines).

#### Reliability and Availability:

* **Uptime:** Target a specific uptime percentage (e.g., 99.9%) to ensure the system is available during restaurant operating hours.
* **Error Handling:** Implement robust error handling to manage unexpected issues gracefully without crashing the system.

#### Maintenance and Support:

* **Documentation:** Provide comprehensive documentation for system administrators, developers, and end-users.
* **Update Mechanism:** Implement a mechanism for deploying updates and patches without disrupting ongoing operations.

### Future Enhancements:

#### Billing and Payment:

1. **Multiple Payment Options:** Support various payment methods, including mobile payments, and cash.
2. **Split Billing:** Allow customers to split the bill among multiple payers.

#### Kitchen Management System:

1. **Order Queue Management:** Organize and prioritize orders in the kitchen based on the time of submission and table number.
2. **Order Status Tracking:** Track the preparation status of each order (e.g., pending, in progress, ready).
3. **Notification System:** Notify waitstaff when orders are ready to be served.

#### Mobile Applications:

The initial phase of the project is focused solely on a web-based system, developing a mobile app will be considered later on.

#### Advanced AI Features:

This feature includes providing customer recommendations based on their food selection.

Reporting and Analytics:

1. **Sales Reports:** Generate detailed sales reports by item, category, table, and time period.
2. **Revenue Tracking:** Track revenue, including breakdowns by payment method and discounts applied.
3. **Customer Insights:** Analyze customer behavior, including popular items, average spending, and peak dining times.

#### Analytics:

1. Provide performance of the chefs and their improvements.
2. Menu enhancement based on popular food items

#### Multilingual Support:

Adding support for multiple languages to cater to a diverse customer base.

#### Remote Ordering:

Allows customers to place orders remotely before arriving at the restaurant or for delivery or takeout via mobile application.

## Stake Holders

None

* **Purpose:** A high-level document that outlines the project’s objectives, scope, stakeholders, and overall goals. This is often the first document created.
* **Contents:**
  + **Project Title and Overview:** A brief description of what the project is about.
  + **Objectives:** Clear, measurable goals.
  + **Scope:** What is included and what is not included in the project.
  + **Stakeholders:** Key individuals or groups involved in or affected by the project.
  + **Deliverables:** Major outputs of the project.
  + **Timeline:** High-level timeline or phases.
  + **Budget:** Estimated costs.

**2. Requirements Document:**

* **Purpose:** This document details what the system should do, both functionally and non-functionally. It serves as the foundation for all future development.
* **Contents:**
  + **Functional Requirements:** Specific behaviors or functions of the system (e.g., "The system shall allow users to order food from their table using a QR code.").
  + **Non-Functional Requirements:** Performance, security, usability, and scalability requirements (e.g., "The system shall handle 100 concurrent users with a response time of less than 2 seconds.").
  + **User Stories or Use Cases:** Describe scenarios from the user’s perspective (e.g., "As a customer, I want to view the menu on my mobile device, so I can choose what to order.").
  + **Acceptance Criteria:** Conditions that must be met for a requirement to be accepted (e.g., "The order confirmation page should display a summary of the items ordered, total price, and estimated preparation time.").

**3. System Architecture Document:**

* **Purpose:** This document provides an overview of the system’s architecture, including how different components will interact.
* **Contents:**
  + **System Overview:** A high-level description of the system’s architecture.
  + **Component Diagrams:** Visual representation of how the system components interact (e.g., user interface, database, server-side logic).
  + **Technology Stack:** The specific technologies, programming languages, frameworks, and tools that will be used.
  + **Data Flow Diagrams:** Shows how data will move through the system.
  + **Integration Points:** How the system will integrate with other systems (e.g., payment gateways, third-party services).

**4. Wireframes and Prototypes:**

* **Purpose:** Visual representations of the user interface, showing how users will interact with the system.
* **Contents:**
  + **Wireframes:** Basic layouts of pages/screens without detailed design elements.
  + **Mockups:** High-fidelity designs showing colors, fonts, and other visual elements.
  + **Interactive Prototypes:** Clickable prototypes that simulate user interactions.

**5. Project Plan:**

* **Purpose:** This document outlines the detailed plan for executing the project, including tasks, resources, timelines, and risk management.
* **Contents:**
  + **Work Breakdown Structure (WBS):** Breaks the project into smaller tasks or phases.
  + **Timeline/Gantt Chart:** Visual timeline showing when tasks will be completed.
  + **Resource Allocation:** Who will work on which tasks.
  + **Risk Management Plan:** Potential risks and mitigation strategies.
  + **Milestones and Deliverables:** Key points where progress is reviewed, and deliverables are due.

**6. Testing Plan:**

* **Purpose:** Describes how the system will be tested to ensure it meets the requirements.
* **Contents:**
  + **Test Cases:** Specific scenarios to be tested, with expected outcomes.
  + **Testing Tools:** Tools that will be used for testing (e.g., automated testing frameworks).
  + **Test Environment:** Description of the environment in which testing will occur.
  + **Acceptance Testing:** Criteria for the final acceptance of the system.

**7. Deployment Plan:**

* **Purpose:** Describes how the system will be deployed, including any pre-launch and post-launch activities.
* **Contents:**
  + **Deployment Strategy:** How the system will be deployed (e.g., all at once, phased rollout).
  + **Environment Setup:** Details of production, staging, and development environments.
  + **Backup and Rollback Plan:** Procedures if something goes wrong during deployment.
  + **Training:** How users and staff will be trained to use the new system.

**8. Maintenance and Support Plan:**

* **Purpose:** Describes how the system will be maintained after deployment, including support structures.
* **Contents:**
  + **Maintenance Schedule:** Regular updates, bug fixes, and improvements.
  + **Support:** How users can get help (e.g., helpdesk, support tickets).
  + **Documentation:** Ongoing documentation updates.

**9. Final Documentation and Handoff:**

* **Purpose:** A comprehensive document handed over to the client or end-users, containing everything needed to understand, use, and maintain the system.
* **Contents:**
  + **User Manuals:** Instructions for end-users on how to use the system.
  + **Technical Documentation:** Detailed technical specs for developers.
  + **Change Logs:** Record of all changes made during the project.