# Restaurant Management System

## Technical Summary Document

Date: August 31, 2025

## Overview

This technical document summarizes the development work performed on the Restaurant Management System application, including feature implementations, bug fixes, and architectural improvements. The system is built using ASP.NET Core MVC with Entity Framework Core for data access and Microsoft SQL Server for data storage.

## System Architecture

The application follows an MVC (Model-View-Controller) architecture pattern with the following components:

* **Models**: Data representations for business entities
* **Views**: UI templates implemented with Razor views
* **Controllers**: Handle user requests and coordinate interactions
* **Middleware**: Custom components for cross-cutting concerns
* **Data Layer**: Entity Framework Core for database interactions

## Core Modules Implemented

### 1. Table Service Module

* Implemented dashboard for restaurant floor management
* Added ability to seat guests from reservations or waitlist
* Created server assignment functionality for tables
* Developed table status tracking (Available, Occupied, Dirty)
* Fixed issue with server dropdown in SeatGuest view by correcting SQL query to concatenate FirstName and LastName fields

### 2. Category Management

* Implemented CRUD operations for menu categories
* Fixed bug in CategoryForm.cshtml that prevented editing of categories
* Modified CategoryForm view to properly handle edit/view states
* Resolved issue with computed column "CategoryName" by refactoring model to use "Name" as primary property

### 3. Kitchen Management System

* Developed kitchen ticket tracking system
* Created kitchen stations configuration
* Implemented order routing to appropriate kitchen stations
* Added ticket status tracking and updates
* Developed kitchen dashboard for monitoring active tickets

### 4. Online Order Integration

* Built framework for integrating with third-party delivery services
* Created order source configuration management
* Implemented menu item mapping for external services
* Developed order processing pipeline
* Added API webhook handlers for order status updates

### 5. Database Improvements

* Created DatabaseColumnFixMiddleware for automatic schema reconciliation
* Implemented proper Entity Framework Core model configurations
* Added SQL migration scripts for database schema changes
* Fixed issues with computed columns in database

## Technical Implementations

### Entity Framework Core Integration

* Configured DbContext with fluent API for entity mapping
* Set up proper relationships between entities
* Implemented data access patterns for efficient querying

### Middleware Components

* Created DatabaseColumnFixMiddleware that:
  + Automatically detects database schema mismatches
  + Fixes computed column issues (e.g., CategoryName)
  + Ensures database schema compatibility with models

### Model Refactoring

Refactored Category model to resolve computed column issues:

public class Category  
{  
    public int Id { get; set; }  
    public required string Name { get; set; }  
    public bool IsActive { get; set; }  
      
    // Property wrapper that maps to the old computed column  
    public string CategoryName  
    {  
        get => Name;  
        set => Name = value;  
    }  
}

### User Interface Enhancements

* Improved form handling for edit/view modes
* Fixed disabled fields in CategoryForm
* Enhanced table service interface for better usability
* Created dashboards for different functional areas

### SQL Query Optimization

Fixed TableServiceController.GetAvailableServers() method:

private List<SelectListItem> GetAvailableServers()  
{  
    var servers = new List<SelectListItem>();  
      
    using (SqlConnection connection = new SqlConnection(\_connectionString))  
    {  
        connection.Open();  
        using (SqlCommand command = new SqlCommand(@"  
            SELECT Id, FirstName + ' ' + ISNULL(LastName, '') AS FullName  
            FROM Users  
            WHERE Role = 2 -- Server role  
            AND IsActive = 1  
            ORDER BY FirstName, LastName", connection))  
        {  
            using (SqlDataReader reader = command.ExecuteReader())  
            {  
                while (reader.Read())  
                {  
                    servers.Add(new SelectListItem  
                    {  
                        Value = reader.GetInt32(0).ToString(),  
                        Text = reader.GetString(1)  
                    });  
                }  
            }  
        }  
    }  
      
    return servers;  
}

## Bug Fixes

### 1. Category Edit Functionality

* **Problem:** Category edit form had disabled input fields in edit mode
* **Solution:** Modified JavaScript in CategoryForm.cshtml to only disable fields in view mode
* **Impact:** Users can now properly edit category information

### 2. Server Dropdown Empty Issue

* **Problem:** Server dropdown in SeatGuest page was empty
* **Solution:** Fixed SQL query in GetAvailableServers() to use FirstName + LastName instead of non-existent FullName column
* **Impact:** Restaurant hosts can now assign servers to tables

### 3. Computed Column Database Conflicts

* **Problem:** EF Core was trying to modify computed columns in the database
* **Solution:**
  + Created DatabaseColumnFixMiddleware to detect and fix schema issues
  + Refactored models to use appropriate property mappings
  + Updated DbContext configurations to ignore computed columns
* **Impact:** System now handles database schema properly without errors

## Architecture Improvements

### 1. Middleware for Database Schema Management

* Created custom middleware that runs at application startup
* Automatically detects and fixes schema mismatches
* Prevents runtime errors due to database column conflicts

### 2. Model Separation

* Separated view models from data models
* Created dedicated ViewModels folder structure
* Improved code organization and maintainability

### 3. SQL Scripts for Database Management

* Added SQL setup scripts for new modules
* Created migration scripts for schema changes
* Improved deployment process documentation

## Development Workflow

1. Source code managed with Git
2. Local development and testing performed with:

* dotnet run --project RestaurantManagementSystem

1. Commits made to local repository with descriptive messages
2. Changes pushed to remote repository at github.com/apporel27-collab/eMEDRestaurant

## Future Recommendations

### 1. Code Modernization

* Replace deprecated System.Data.SqlClient with Microsoft.Data.SqlClient
* Address numerous warning messages in the codebase
* Implement proper nullable reference type handling

### 2. Architecture Improvements

* Consider implementing repository pattern for better separation of concerns
* Add unit tests for critical business logic
* Implement dependency injection more consistently

### 3. Database Optimizations

* Consider using migrations instead of custom middleware
* Optimize database queries for better performance
* Implement database indexing strategy

## Conclusion

The Restaurant Management System has been significantly enhanced with new modules for Kitchen Management and Online Orders, along with fixes to existing functionality. The system architecture has been improved with better database handling, and critical bugs have been resolved.

The codebase is now better structured and more maintainable, though there are still opportunities for further improvement as outlined in the recommendations section.