**Пример Технического Задания на Разработку Веб-приложения «АИС Ресторана»**

**1. Введение**

**1.1** **Цель документа**

Этот документ описывает требования к разработке автоматизированной информационной системы (АИС) ресторана, предназначенной для упрощения процесса управления заказами, бронирования столиков и доставки.

**1.2 Актуальность и назначение проекта**

В условиях растущей конкуренции в ресторанном бизнесе важно предоставлять клиентам удобный способ взаимодействия с рестораном через интернет. АИС ресторана поможет автоматизировать ключевые процессы, сократить время обработки заказов и повысить удовлетворенность клиентов.

**1.3 Основные пользователи системы**

Система предназначена для следующих категорий пользователей:

* **Пользователь** — просмотр меню, создание заказов, бронирование столиков, отслеживание доставки.
* **Администратор** — управление меню, бронированиями, заказами и пользователями.
* **Курьер** — просмотр назначенных доставок и обновление их статуса.

**2. Цели и задачи проекта**

2.1 **Основные цели**

* Автоматизация процессов заказа, бронирования и доставки.
* Создание удобного интерфейса для взаимодействия клиентов с рестораном.

2.2 **Конкретные задачи**

* Разработка модуля для просмотра меню с фильтрацией по категориям и особенностям блюд.
* Реализация функционала онлайн-заказов (включая оплату).
* Создание системы бронирования столиков с учетом доступности.
* Разработка интерфейса для отслеживания статуса доставки.
* Создание панели администратора для управления системой.

Пример: Пользователь может выбрать блюдо из меню, указать адрес доставки и оплатить заказ онлайн.

**3. Требования к системе**

3.1 **Функциональные требования**

* Меню: отображение категорий блюд, фильтры (вегетарианское, без глютена и т.д.).
* Заказы: создание заказа с выбором блюд и возможностью онлайн-оплаты.
* Бронирование: выбор даты, времени и количества гостей.
* Доставка: возможность отслеживания статуса заказа (готовится, в пути, доставлен).
* Администрирование: добавление и редактирование блюд, управление бронированиями и заказами, назначение доставок.

3.2 Нефункциональные требования

* Производительность: отклик системы не более 2 секунд при обработке заказов.
* Масштабируемость: поддержка 100+ одновременных пользователей.
* Безопасность: защита платежных данных через SSL.

4. Требования к пользовательскому интерфейсу

4.1 Основные экраны

* Главная страница: краткое описание ресторана и кнопки быстрого доступа (меню, заказы, бронирование).
* Меню: список блюд с фильтрацией и поиском.
* Корзина: просмотр выбранных блюд, итоговой суммы и кнопки оплаты.
* Личный кабинет пользователя: история заказов, статус бронирований.
* Панель администратора: управление меню, бронированиями, заказами и доставками.

4.2 Навигация  
Система должна включать удобное меню с разделами: "Меню", "Заказы", "Бронирование", "Доставка", "Личный кабинет".

4.3 Юзабилити

* Простая структура интерфейса для быстрого взаимодействия.
* Использование визуальных индикаторов (например, цветовые статусы для заказов: в пути, доставлен).

5. Требования к технической реализации

5.1 Языки и технологии

* Backend: Node.js, Express.
* Frontend: React.js или Angular.
* База данных: PostgreSQL.

5.2 Архитектура системы

* Микросервисная архитектура с REST API для взаимодействия между компонентами.

5.3 Интеграции

* Системы онлайн-оплаты (Stripe, PayPal или аналог).
* Карты для отслеживания доставки (например, Google Maps API).

6. Требования к безопасности

6.1 Аутентификация и авторизация

* Авторизация через email и пароль.
* Разграничение прав доступа для пользователей, администраторов и курьеров.

6.2 Шифрование данных

* HTTPS для всех передаваемых данных.
* Хэширование паролей пользователей.

7. Ограничения и допущения

7.1 Технические ограничения

* Поддержка только десктопной версии на первом этапе разработки.
* Бюджет ограничен 1 000 000 ₽.

7.2 Сроки выполнения

* Завершение разработки — 3 месяца.

8. Требования к тестированию и приемке

8.1 Типы тестирования

* Функциональное тестирование: проверка всех пользовательских сценариев.
* Тестирование безопасности: защита от SQL-инъекций и XSS.

8.2 Критерии приемки

* Отсутствие критических ошибок в функционале.
* Прохождение всех тестов на производительность и безопасность.
* 8.3 Критерии успешности
* Система считается успешно разработанной, если:
* 95% функциональных и нефункциональных требований выполнены.
* Пользователи подтверждают удобство и корректность работы ключевых функций (меню, заказы, бронирование, доставка).
* Все этапы тестирования успешно завершены без выявленных критических уязвимостей.
* 9. Требования к документации
* 9.1 Пользовательская документация
* Инструкции для клиентов по использованию функционала (например, как оформить заказ или забронировать столик).
* Часто задаваемые вопросы (FAQ) и решения типичных проблем.
* 9.2 Техническая документация
* Описание структуры базы данных.
* Документация по API для интеграции и дальнейшего развития системы.
* Руководство для администраторов и разработчиков: установка, настройка, администрирование.
* 10. План реализации
* 10.1 Этапы разработки
* Анализ требований и проектирование — 1 месяц.
* Сбор и уточнение требований, создание прототипов.
* Разработка прототипа — 1 месяц.
* Реализация базовой версии интерфейса и ключевых функций (меню и заказы).
* Полноценная разработка — 1,5 месяца.
* Разработка всех модулей, интеграция с платежными системами.
* Тестирование и доработка — 2 недели.
* Тестирование функциональности, исправление ошибок.
* Внедрение и обучение — 2 недели.
* Настройка системы и обучение администраторов.
* 10.2 Сроки выполнения этапов
* Дата начала: 1 декабря 2024 года.
* Дата завершения: 31 марта 2025 года.
* 10.3 Ответственные лица
* Менеджер проекта: Иванов И.И.
* Ведущий разработчик: Смирнов А.А.
* Тестировщик: Петров П.П.
* 11. Приложения
* Приложение A: Прототипы интерфейса (эскизы основных экранов).
* Приложение B: ER-диаграмма базы данных.
* Приложение C: Спецификации API для взаимодействия модулей.