COMP 3059 – Capstone Project I

Software Requirements Analysis and Design Assignment

This assignment is an overview to gather the software needs with requirements analysis and help to proceed with the design.

The requirements analysis helps to break down functional and non-functional requirements to a basic design view to provide a clear system development process framework. It involves various entities, including business, stakeholders and technology requirements.

The design is the activity following requirements specification and before programming. Software design usually involves problem solving and planning a software solution.

To work on this assignment you could use the references and a sample template given below. The sample template can be customised to suit the nature of your project.

Reference Readings/Example:
http://www.uacg.bg/filebank/acadstaff/userfiles/publ_bg_397_SDP_activities_and_steps.pdf

www.cse.msu.edu/~chengb/RE-491/Papers/SRSExample-webapp.doc

Source for this template:
www.tricity.wsu.edu/~mckinnon/cpts322/cpts322-srs-v1.doc

1.0 Introduction

The Introduction section provides an overview of the system using software requirements analysis and design for the scope of the system.

1.1 Purpose

It provides a clear understanding of the system's expected behaviour and capabilities, without detailing the technical implementation. This document will serve as reference for developers, stakeholders and other team members, ensuring that the system is developed according to the needs of both restaurant owners and customers. The primary goal is to streamline the queue management process, improving customer satisfaction and operational efficiency for restaurant. .

1.2 Scope

This explains what the proposed system will and will not do. Describe relevant benefits, objectives and goals. The description of scope should be consistent with the Project Plan.

The Queue Management Application (JUSTIN) will allow customers to view real-time wait times, join a queue, receive notifications when a table is ready, and leave reviews after dining. Restaurant managers will be able to monitor customer flow, update table availability, and receive real-time data from Raspberry Pi devices to optimize operations.

In scope :

- Real-time updates on table availability and wait times
- Integration with Raspberry Pi for automated data collection
- Customer interaction through a mobile application
- Restaurant management interface to monitor demand and customer flow
- Admin portal for system oversight and updates
- Customer review(Rating) system

Out of scope:

- Payment processing for orders
- Non-dining industries(although future updates could allow for expansion to other sectors)
- Reservations are not managed

2.0 System Overview

The System Overview section introduces the system context and design.

2.1 Project Perspective

The Project Perspective describes the context and origin of the system by defining whether the system is:

- a follow-on member of a system family
- a replacement for existing systems, or
- a new self-contained system.

The Queue management application(JUSTIN) is a new, self-contained system designed specifically to address the inefficiencies related to managing queues and wait times in restaurants. Unlike other systems that primarily focus on reservation management, this system focuses on real-itme data collection and customer flow management, providing an innovative solution for the dining industry. This is not a follow-on system, nor does it replace any existing systems, but instead introduces a new, and integrated solution for both customers and restaurants owners.

2.2 System Context

The System Context describes the resulting software within the business case, including strategic issues in which the system is involved or which it specifically addresses.

The Queue management system fits within the broader business strategy of enhancing customer experience and operational efficiency in restaurants. The system specifically addresses the strategic challenges of managing customer expectations during busy period by providing real-time information on seating availability. It helps restaurants balance customer flow and table turnover, while improving customer satisfaction by minimizing wait times and offering alternative restaurant suggestions if wait times are too long. The use of IOT device (Raspberry Pi) further enhances the system's ability to capture rea;-time data, making it a critical tool for modern restaurant management.

2.3 General Constraints

General Constraints identify any business or system constraints that will impact the manner in which the software is to be:

- specified
- designed
- implemented, or
- tested.

Business Constraints

- The system must be designed to serve only the restaurant industry initially, though future updates may expand it to other sectors such as retail.
- The system must comply with local regulations related to customer data privacy and security, especially since it handles personal data.

System Constraints

- Real-time performance : The system relies on Raspberry Pi device for real-time data, so it must process data quickly and efficiently to update wait times and table availability without delays.
- Scalability: The system must be scalable to handle large volumes of users, especially during peak restaurant hours in busy areas
- Device Limitations: The system's real-time functionality using Raspberry Pi may be limited by hardware performance, network range, and power availability

2.4 Assumptions and Dependencies

List any assumptions that have been made during the initiation of the project. In addition, list any dependencies that may impact its success or the desired result.

- Third-party services : The system might rely on external APIs (map integration) that could impact performance if those services experience issues
-
- Hardware : The system's success depends on the Raspberry Pi devices functioning correctly and transmitting data as expected. Hardware malfunctions or issues with sensor accuracy could affect system reliability.
-
- Data privacy regulations : The system must adhere to all data privacy laws that affect how customer data is collected, stored, and processed, which could impact design decisions

- Restaurant participation: The system's value is dependent on restaurant adoption. If restaurants are slow to adopt the system or do not update table availability in real time, it may limit the usefulness for customers.

3.0 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

3.1 <Functional Requirement or Feature #1>

- Introduction
- Inputs
- Processing
- Outputs

Customer Detection

- Introduction
    - By using camera integrated with AI, our system will keep track how many customers in the restaurant
- Inputs
    - Camera devices connected with Raspberry pi
    - Customer inside the restaurant
- Processing
    - Trained model will count the customer
- Outputs
    - Display how many customers inside the restaurant
    - Calculate how much time must wait

Restaurant's location

- Introduction
  - After registered to application, restaurant's location and all necessary information will show in the map
- Inputs
  - Restaurant information such as address, name, image, phone etc
- Processing
  - By using Google Map API, display the location of registered restaurants
- Outputs
  - Registered restaurants will be displayed in the map

Waiting time estimations

- Introduction
  - Showing the estimation waiting time for the customer before going to the restaurant
- Inputs
  - Current's customer inside the restaurant
- Processing
  - Using algorithm and training AI to calculate the estimated time
- Outputs
  - Display the time in the map

...

3.2 Use Cases

- Case ID – 01 -> Customer side
- Case ID – 02 -> Restaurant Side
- Case ID – 03 -> Ai and Raspberry Pi Device

Case ID – 01

Description : one customer wants to join the queue for a table at a restaurant through the mobile application.

Primary Actor : Customer

Pre-Condtions

- The customer has the mobile application installed
- The customer is registered and logged into the application
- The customer knows the name of the restaurant / address

Trigger

- The customer wants to join a restaurant queue for dinning

Flows

1. The customer selects a restaurant from the list of available options on the applciation
2. The system displays the estimated wait time / Queues and available seating information
3. The customer selects the option to join the queue
4. The system adds the customer to the queue and provides a confirmation message with an estimated wait time and how many groups or people front of her/him
5. The customer receives a notification confirming their queue status

Post-conditions

- The customer is successfully added to the queue, and the restaurant's system is updated with the customer's position in the queue

Case ID – 02

Description : The restaurant manager updates the availability of tables in real-time based on the current occupancy

Primary Actor : Restaurant Manager

Pre-condition

- The restaurant manager is logged into the system's management interface
- The restaurant is connected to the queue management system

Trigger

- The restaurant manager wants to update table availability right away

Flows

1. The manager accesses the restaurant's dashboard
2. The system displays the current table availability and queue status
3. The manager updates the status of tables as they become available or occupied.
4. The system updates the information in real-time and notifies any waiting customers of their position in the queue.

Post-condition

- The system reflects accurate table availability, and waiting customers are notified of their updated queue status if relevant.

Case ID - 03

Description: The system monitors and displays the real-time count of customers within the restaurant.

Primary Actor: Queue Management System

Preconditions:

- Cameras integrated with AI and Raspberry Pi devices are installed in the restaurant.

Trigger:

- The system continuously monitors the restaurant for customer count updates.

Basic Flow:

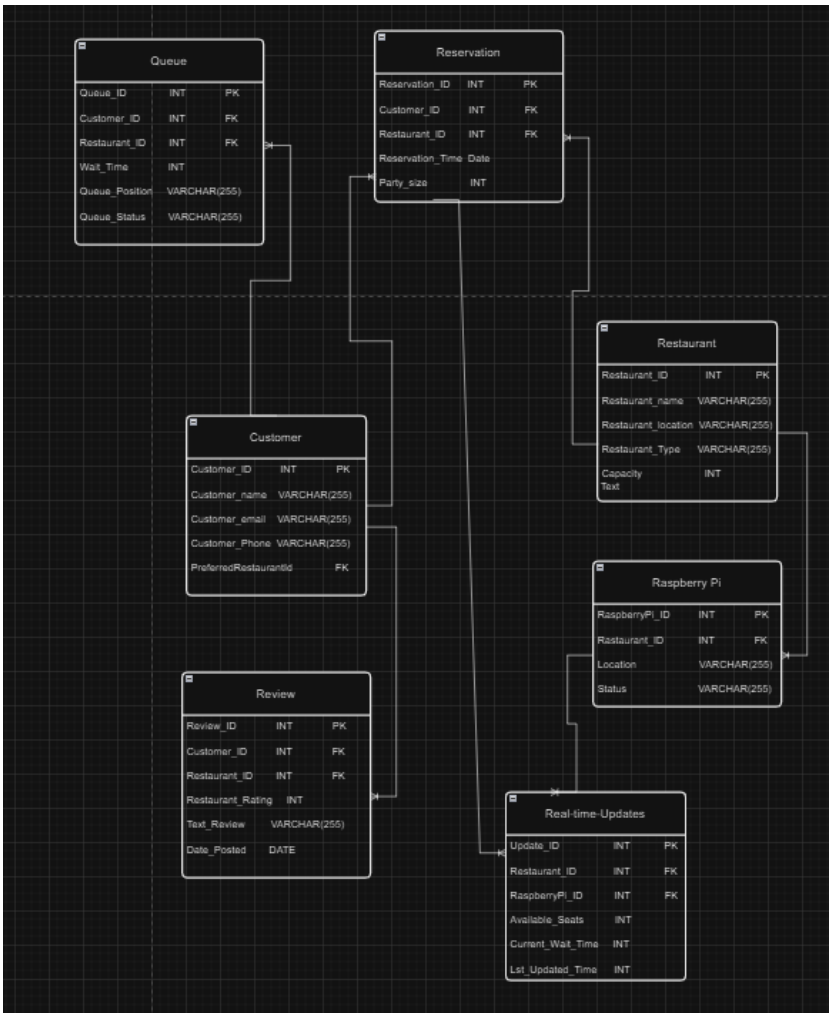
1. The camera detects customers entering and exiting the restaurant.
2. The AI model processes the footage and calculates the current number of customers.
3. The Raspberry Pi device updates the customer count in the system.
4. The system displays the real-time count of customers and updates estimated wait times accordingly.

Postconditions:

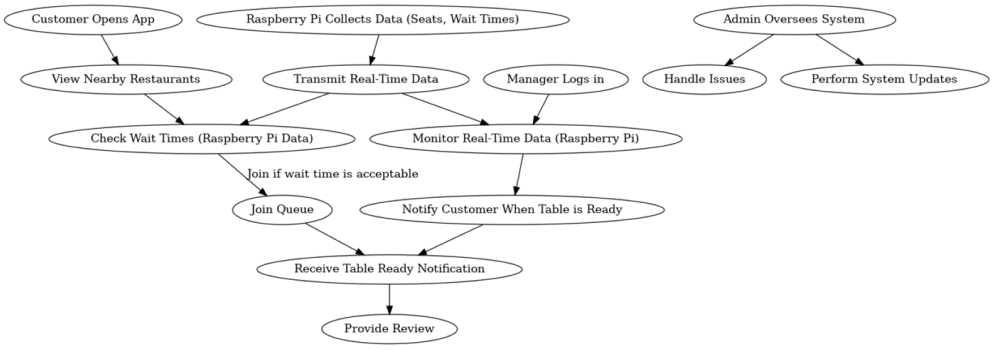
- The real-time customer count is accurately displayed to both customers and restaurant staff.

3.3 Data Modelling and Analysis

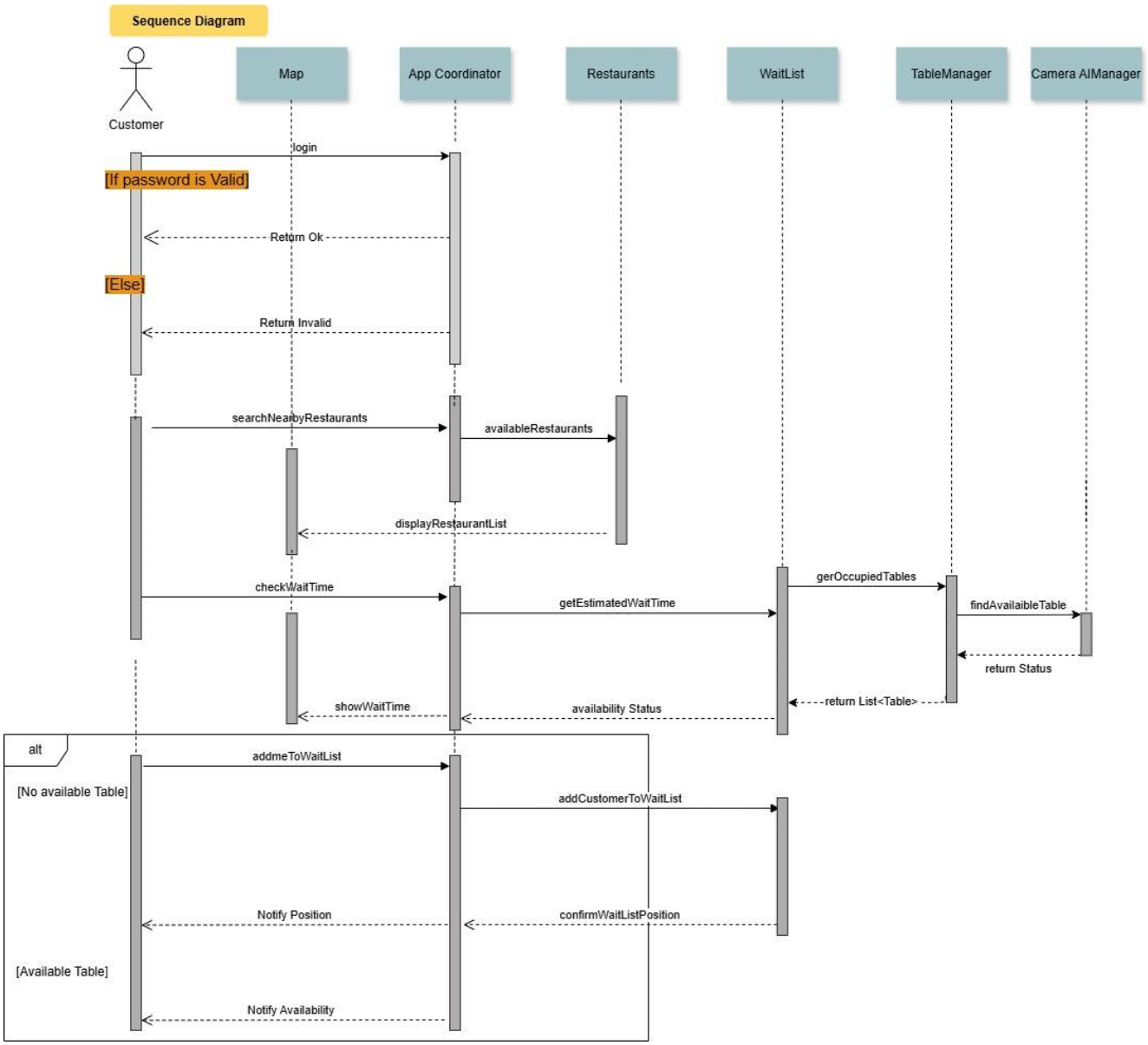
- Normalized Data Model Diagram



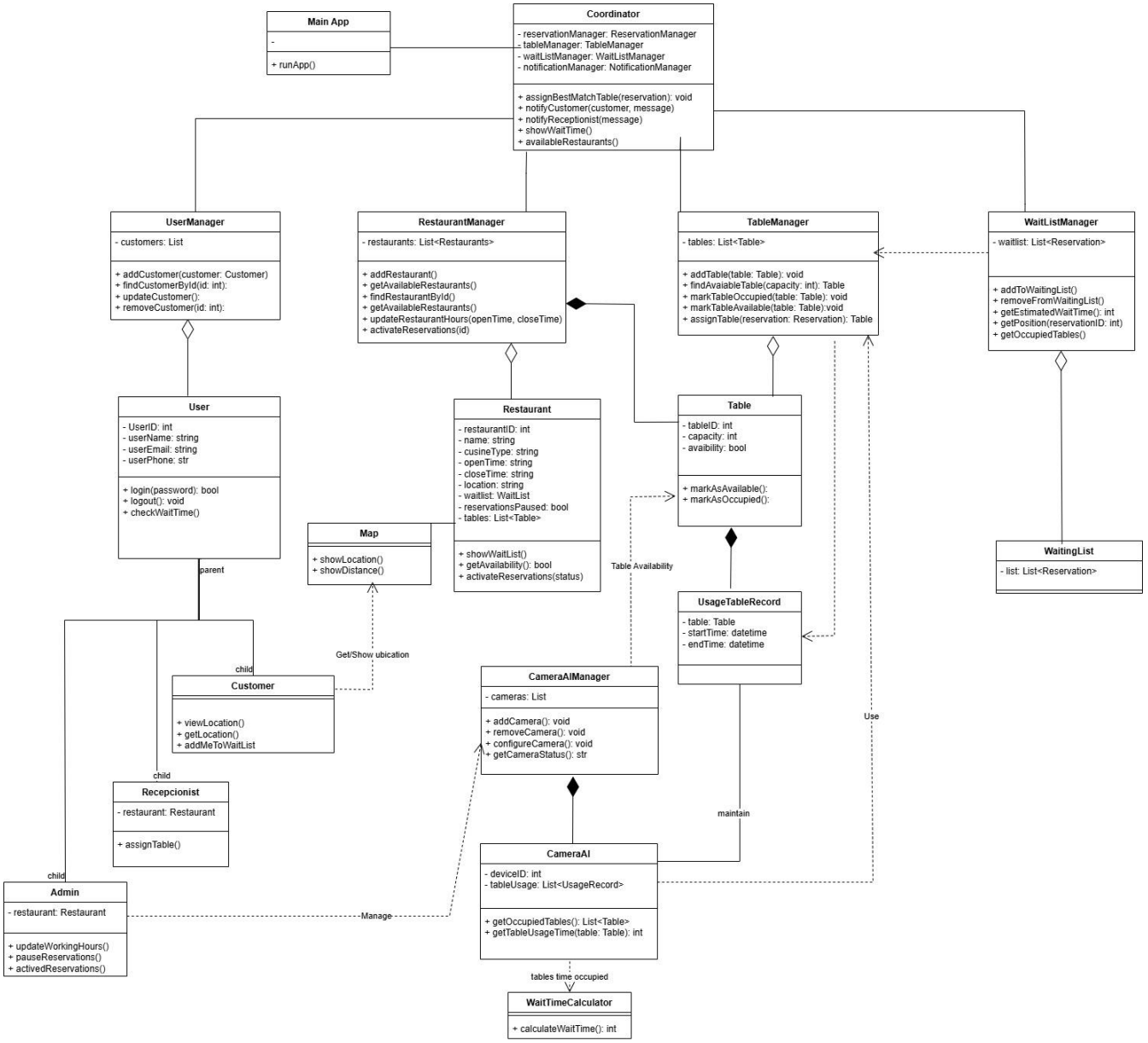
- Activity Diagrams



- Sequence Diagrams

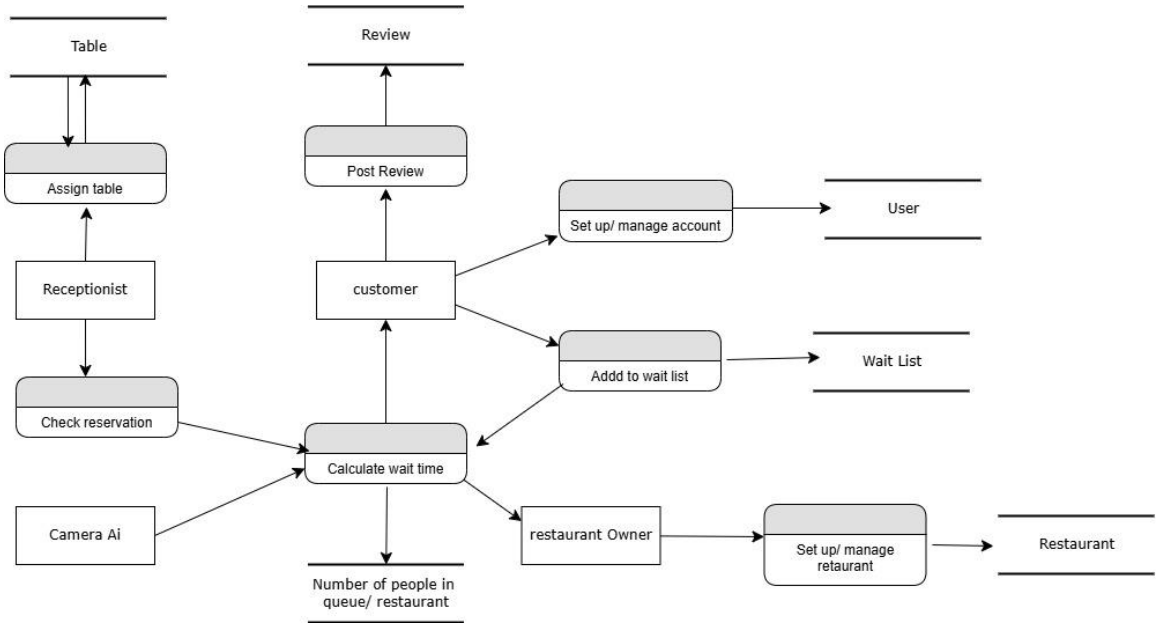


- UML Class Diagram



3.4 Process Modelling

- Data Flow Diagram



4.0 Non-Functional Requirements

The non-functional requirements for a system are typically constraints on the functional requirements – that is, not what the system does, but how it does it (e.g. how quickly, how efficiently, how easily from the user's perspective, etc.).

Non-functional requirements may exist for any of the following attributes – Performance, Reliability, Availability, Security, Maintainability, Portability.

Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, etc).

Performance

- **Response Time**: The application should respond to user actions (e.g., button clicks, page loads) within 2 seconds, even under peak load conditions.
- **Throughput**: The system must handle 1000 concurrent users without degradation in performance.
- **Latency**: For real-time features, such as map-based directions, the application should provide updates with a maximum latency of 1 second.
- **Resource Utilization**: CPU usage should not exceed 80% and memory usage should not exceed 75% under average load.

Reliability

- **Error Rate**: The system should have an error rate of less than 0.05% of transactions.
- **Fault Tolerance**: If a server fails, the system should automatically switch to a backup server without interrupting user experience.
- **Data Consistency**: After any transaction, data should be immediately consistent across all systems.

- **Recovery Time Objective (RTO)**: In case of a failure, the system should recover within 7 minutes.

Availability

- **Uptime Requirement**: The application should have 99% uptime annually, which allows for no more than 8.96 hours of downtime per year.
- **Maintenance Window**: Scheduled maintenance should be conducted only during off-peak hours (Monday and Tuesday late night), with prior notification to users, and should not exceed 2 hours per week.
- **Redundancy**: All critical components (e.g., database, application server) should have redundant instances to minimize the risk of downtime.

Security

- **Authentication**: Users must log in using two-factor authentication for sensitive actions.
- **Data Encryption**: All sensitive data should be encrypted in transit (using HTTPS) and at rest (AES-256 encryption).
- **Access Control**: The application should restrict access to sensitive data and features based on user roles.
- **Data Privacy**: User and Restaurant data from Camera must comply with GDPR and CCPA regulations, with options for data anonymization and deletion.

Maintainability

- **Modular Design**: The codebase should follow a modular design to make it easier to update specific features without affecting the entire application.
- **Documentation**: Code and architecture should be well-documented, with at least 80% coverage on key classes and functions.
- **Error Logging**: The system should log errors with sufficient detail for debugging, including error codes and timestamps.
- **Mean Time to Repair (MTTR)**: The system should have an MTTR of under 1 hour, meaning issues should typically be resolved within 1 hour.

Portability

- **Platform Compatibility**: The application should be compatible with Android and iOS platforms.
- **Browser Compatibility**: If a web component is included, the application should support the latest versions of Chrome, Firefox, Safari, and Edge.
- **Operating System Independence**: The backend should be deployable on both Linux and Windows servers without modification.
- **Migration Support**: The application should support easy migration to cloud providers such as AWS, Azure, or Google Cloud Platform without significant code changes.

5.0 Logical Database Requirements

Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc?

We will use the Database and we consider using the NoSQL Database such as MongoDB. For the detail's requirements, it will be like the list below:

1. Data Formats
    a. MongoDB will use the Binary JSON as a main format for all the data store as documentation type
2. Storage Capabilities
    a. Support scalability for map and customer, restaurant data. For the media we will store it in Cloud service such as Amazon or Azure
3. Data retention
    a. Retain Restaurant data until it closed, soft delete User data for 90 days. Data from Camera will not be the object to be stored.
4. Data Integrity
    a. Enforce foreign keys, validation rules (e.g., email format, positive values), and ACID-compliant transactions.

6.0 Other Requirements

Additional requirements, if any.

Multi-platform compatibility: Android, iOS, Web Browser
Authentication and Authorization:
Data Encryption: All sensitive data should be encrypted in transit (using HTTPS) and at rest (AES-256 encryption).

7.0 Approval

The signatures below indicate their approval of the contents of this document.

Project Role	Name	Signature	Date
Project Coordinator/Full Stack Developer	Chau Minh Truong	CM	2024/11/5