Department of Computer Science & Engineering
The University of Texas at Arlington

Architectural Design Specification
CSE 4316: Senior Design I
Summer 2023

Team O
GourmetBook

An Hyeonjun
Ayalew Bereket
Shrestha Bhumika
Gyawali Reety
Rakshav Patel

Revision History

Revision	Date	Author(s)	Description
0.1	08.09.2023	HA	document creation
0.1.1	08.09.2023	HA	Diagrams inserted for 2, 3, 4, 5, and 6
0.2	08.09.2023	HA	Introduction & System Overview edited
0.3	08.09.2023	RP	Data/backend layer edited

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

Our product is a mobile based application built to enhance the dining experience for customers and to be an efficient way for restaurant to provide service. With a user-friendly interface and innovative features included, the application is truly an upgrade to the hospitality industry. This app aims to streamline the reservation process, offer convenient online pre-ordering, provide event information nearby, and ensure efficient restaurant management.

The growing demand and curiosity in experiencing well-treated and high-end quality cuisines are something that has recently caught people's attention through various sources of media. And there is no single application that aggregates all those fancy restaurants in one place. Moreover, Fine-dining restaurants tend to stay close to the traditional ways of reservations such as phone and website. In addition, there are people with credit cards who get priority to reserve. Since not everyone has access to this privilege, we are here building this application for the rest of the people who could enjoy fine dining without having specific credit cards. This app will be easy to use.

The following sections are detailed descriptions and explanations of each architectural layer, highlighting their subsystems.

2 System Overview

The application has three complementary layers, and each layer is again divided into different subsystems and functions. The layers are presentation, application, and data/backend.

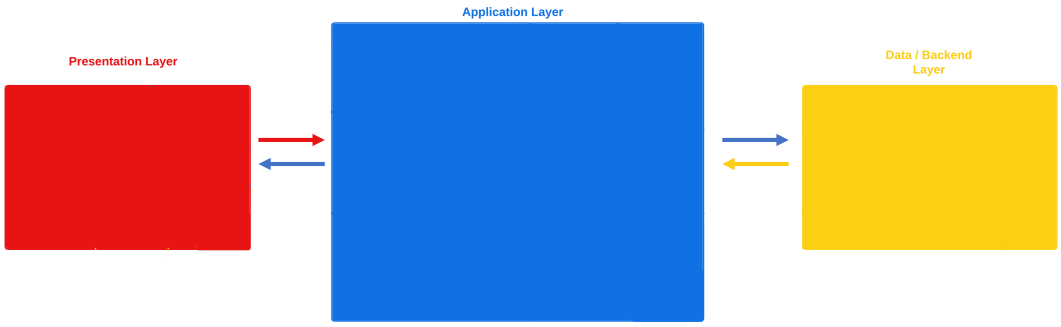

Figure 1: A simple architectural layer diagram

2.1 Presentation Layer Description

The presentation layer serves as the user-facing interface of the application. It encompasses the visual elements, user interactions, and overall user experience. This layer focuses on providing a visually appealing and intuitive interface for users to interact with the app's features. It includes components like screens, buttons, menus, and other user interface elements that allow users to navigate through the application and perform various actions. The presentation layer is responsible for capturing user input and displaying relevant information in a coherent and user-friendly manner.

2.2 Application Layer Description

The application layer acts as the logic and control center of the application. This layer handles the processing of user inputs, interactions, and business logic. It coordinates the different functionalities and services provided by the app, ensuring that the appropriate actions are taken in response to user requests. This layer is also responsible for managing the flow of data between the presentation layer and the data/backend layer. It encompasses various subsystems like the server that perform specific tasks, such as authentication, authorization, data processing, and communication with the backend.

2.3 Data/Backend Layer Description

The data/backend layer is responsible for managing the storage, retrieval, and manipulation of data used by the application. It communicates with remote servers and databases to retrieve and update information. This layer ensures data integrity, security, and efficient access. It includes subsystems related to data storage, retrieval, caching, and synchronization. Additionally, it handles communication protocols, APIs, and network interactions to seamlessly connect the mobile application with the remote server and databases.

3 Subsystem Definitions & Data Flow

The "GourmetBook" sequence diagram breaks down our system into easy-to-understand parts. It provides a clear view of how each part of the system interacts, highlighting the different sections within each layer. In the deeper layers of our "GourmetBook" system, it has meticulously mapped out the structure for clearer comprehension. Visually, each layer showcases distinct subsystems, which are essentially mini-programs, each handling a vital function of that layer. These are not standalone units; they are intertwined in function. Some subsystems push out crucial data, acting as sources, while others pull in this data, functioning as sinks. To give a clear picture of this data journey, a diagram is created so that it resembles the flow of data, grounding it in the core structure of the system. This approach ensures that anyone looking into our system has a clear and precise understanding of how data travels and how each subsystem plays its part.

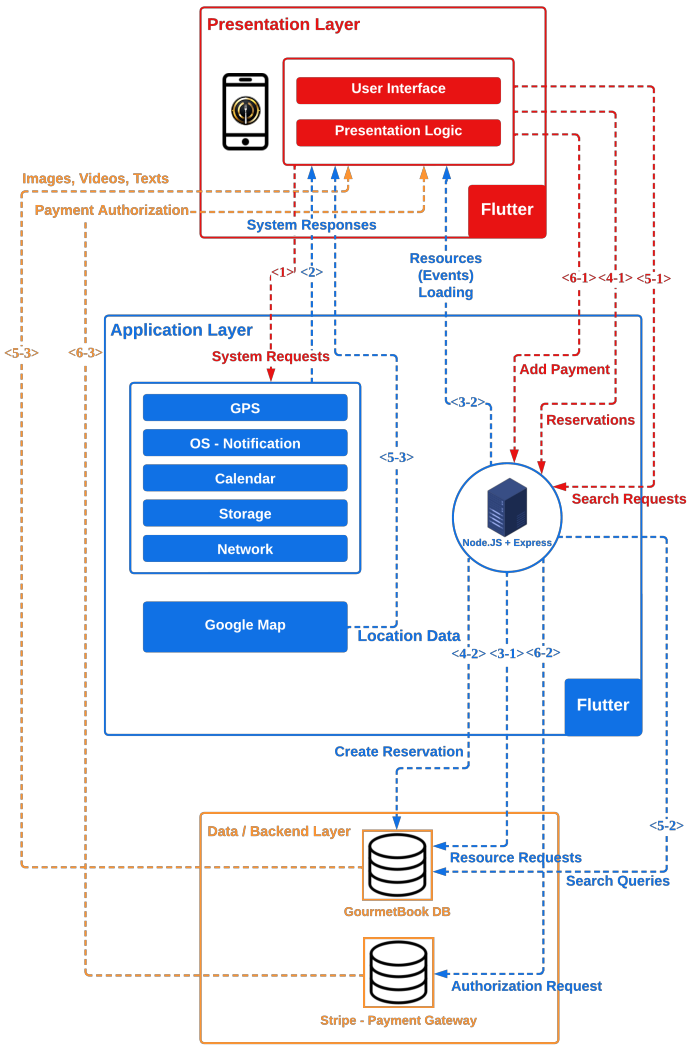

Figure 2: System Architecture Diagram with numbered directed data/request flow

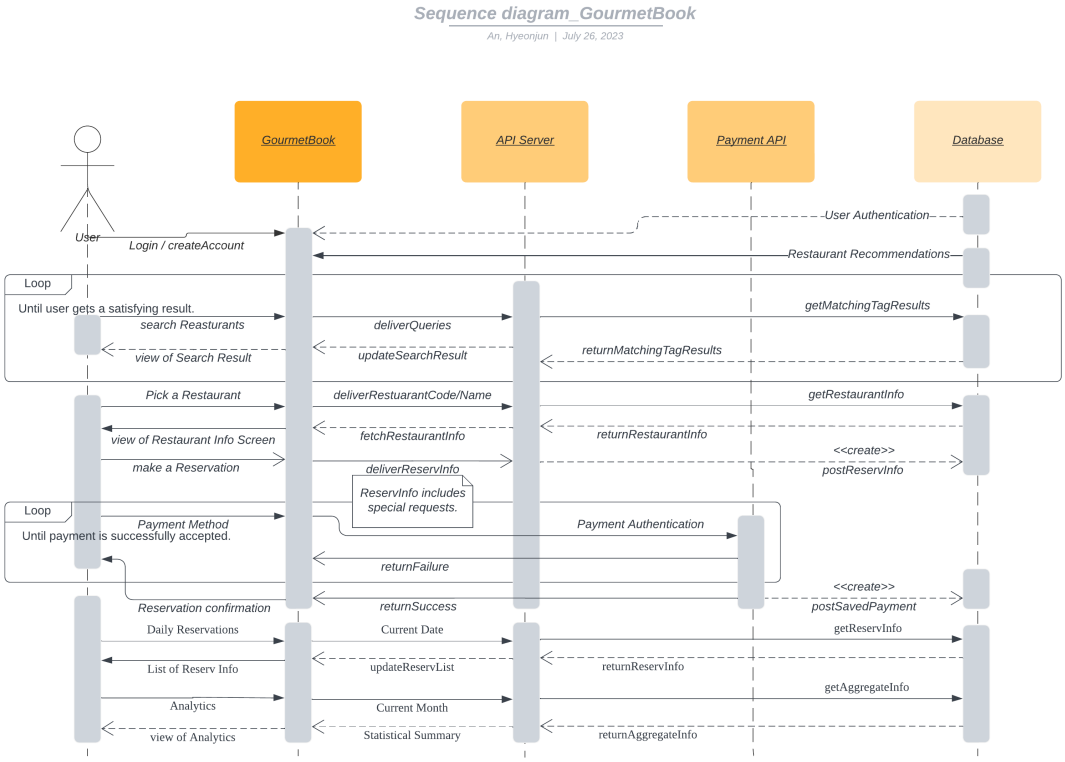

Figure 3: Sequence diagram of the application

4 Presentation Layer Subsystems

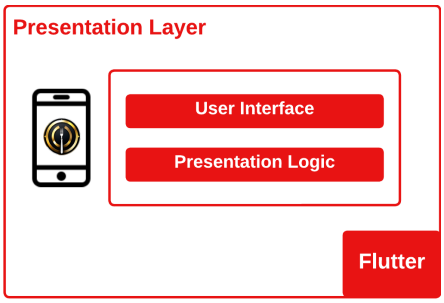

Figure 4: Presentation layer diagram

4.1 Subsystem 1- User Interface

One of the two subsystems which are displayed in the diagram below is UI (User Interface subsystem). This subsystem is responsible for the visual representation of the application that the user interacts with. It includes all the elements and components that users see and interact with on their devices. This subsystem includes features/buttons such as a list of Restaurants, Menus, Events, Search and Filters, User Accounts, History tab, etc.

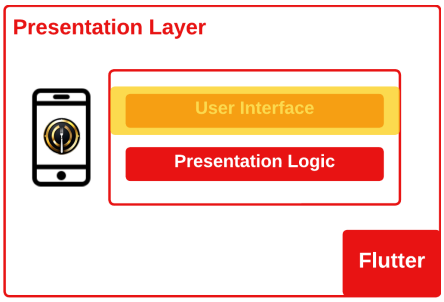

Figure 5: User Interface diagram

4.1.1 Assumptions

- Users will interact with the application through a graphical interface.
- User devices such as Smartphones, computers, or any smart devices are supportable and are capable of rendering the UI components.
- The UI layer assumes that the Application and Database/ Backend layers are functional and can provide the necessary data and services.

4.1.2 Responsibilities

i. UI Navigation
Providing mechanisms for users to navigate back and forth through various sections.
ii. Rendering Interface
Presenting content such as text, images, buttons, forms, and menus in a user-friendly manner.
iii. User Input Handling
Validating and processing user inputs to ensure they are appropriate and conform to expected formats.

iv. Data Presentation
Displaying data retrieved from the Application layer in a comprehensible format for users.
v. Error Handling
Handling errors that may occur during user interactions or data retrieval by displaying user-friendly error messages.
vi. Connection with Backend Layer
Requesting the Backend layer to retrieve, update, or submit data and receiving responses from the backend and displaying it on UI properly.
vii. User Integrity
Ensuring that sensitive user data is transferred safely, in coordination with the Backend layer.

4.1.3 Subsystem Interfaces

Table 2: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	UI Navigation	Click on "Search" button	Navigates to write the keyword to search
#02	Rendering Interface	Click on "restaurants" button	List of restaurants will be shown.
#03	User Input Handling	Click on "Reserve table" button	Date and time entered is processed for the reservation .
#04	Data Presentation	Click on "Menu" button	List of items in the menu will be shown.
#05	Error Handling	Submit reservation field with no date or time	Error message will be shown indicating its missing.
#06	Connection with Backend Layer	Send request for event details	All necessary event details will be displayed.
#07	User Integrity	Enter Login credentials	login successful.

4.2 Subsystem 2- Presentation Logic

Another Subsystem is the Presentation Logic subsystem which is responsible to controls the behavior and flow of the UI. It handles and integrates user inputs, processes them, and coordinates with the backend to retrieve and display the proper data. It includes Data Validation, updating UI, Interaction with Application Layer, user inputs, and navigation.

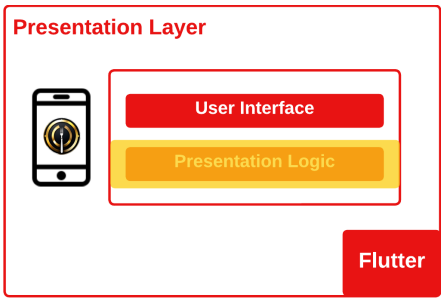

Figure 6: Presentation Logic diagram

4.2.1 Assumptions

- The Application layer is functional and can handle business logic and data processing.
- The Database/Backend layer can store and retrieve data as needed by the Presentation Logic.
- The subsystem layer, the UI layer will provide the user inputs for processing.

4.2.2 Responsibilities

i. UI control/updates
It controls the behavior of the UI components and navigates UI to respond to different inputs accordingly.
ii. Flow Control
It determines which screens or views should be displayed based on user actions.
iii.Data Validation
It converts and validates data into a suitable format to display on UI.
iv. Interaction with Application Layer
Basically communicates with the Application layer and initiates requests to perform specific actions or retrieve data
v. Performance Optimization It controls when to cache data or how to efficiently retrieve information from the Backend layer.

4.2.3 Subsystem Interfaces

Table 3: Subsystem interfaces

ID	Description	Inputs	Outputs
#08	UI control/updates	Click on "Home" button	Navigates to main home page
#09	Flow Control	User proceeds to payment page after confirming orders	Transition to Payment page.
#10	Data Validation	Click on "Order details" button	Order summary including its description and prices will be displayed.
#11	Interaction with Application Layer	Click on "Confirm Reservation" button	Confirmation notification sent.
#12	Performance Optimization	Cache frequently retrieved restaurants data	access cached restaurants data.

5 Application Layer Subsystems

The application layer serves as the logic and control center of our app. The application layer processes the information it receives from the presentation layer and sends it to the data/back-end layer. Below are the several subsystems that work together to handle user inputs, interactions, business logic, and data flow.

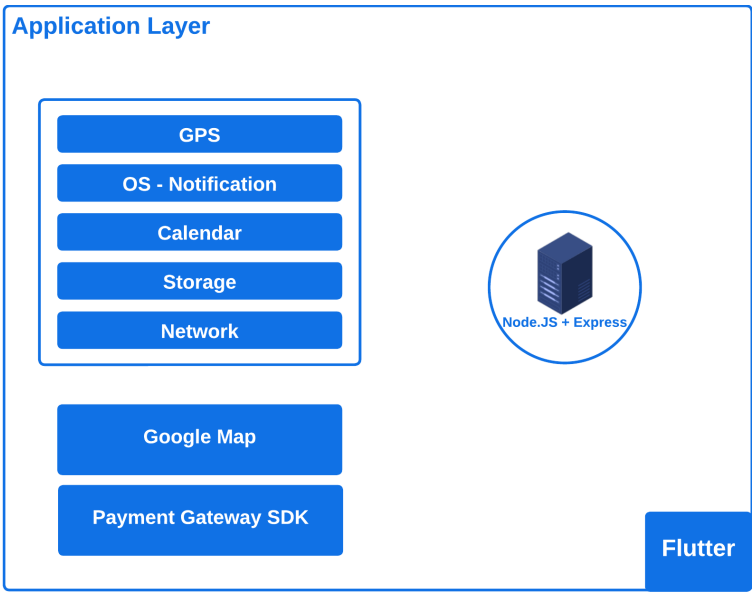

Figure 7: Application layer diagram

5.1 Subsystem 1: System Services (GPS, OS-Notifications, Calendar, Storage, Network)

It handles events from the Presentation Layer and processes them internally to update the system services of phones that have applications. The phone's system services' data is necessary for application logic to run according to a specific user's needs. GPS updates user location, OS notifications alert user, calendar is needed to mark important reservation dates, storage is needed to hold application data, and network provides internet access. This subsystem sends updated system service data to the Presentation Layer to display changes.

5.1.1 Assumptions

- Incoming data is in an expected format.
- Application logic can handle data complexity.
- Device has all system service capabilities.

5.1.2 Responsibilities

- Retrieves user location, storage capacity, uses available network connection.
- It stores data in a limited environment.
- It handles all possible events and user inputs/requests.
- It handles missing/invalid data.

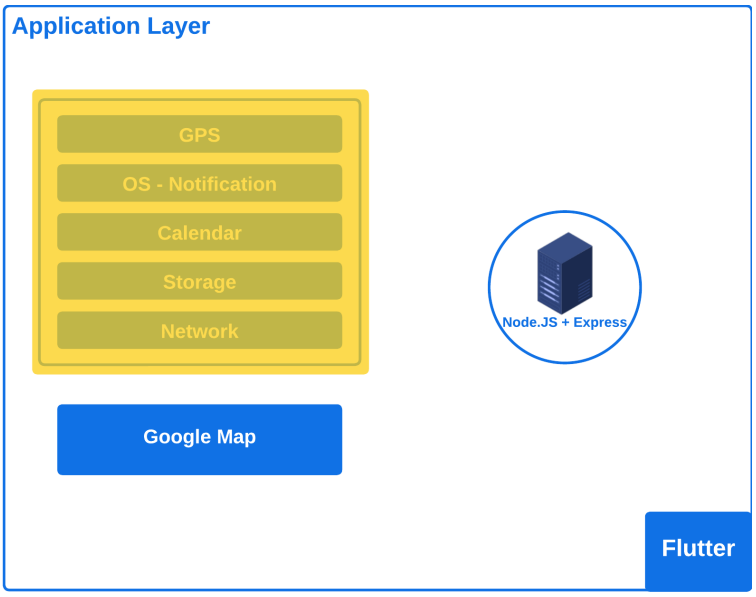

Figure 8: System Services subsystem diagram

- It handles various data types.

5.1.3 Subsystem Interfaces

Table 4: Subsystem interfaces

ID	Description	Inputs	Outputs
#13	Data validation	invalid data	error message
#14	Data transformation	raw data	transformed data

5.2 Subsystem 2: Google Maps

Google Maps is a GPS navigation application developed by Google for mobile devices. It provides various API services through Google Maps SDK for both Android and iOS. Once the user requests the location information of a restaurant they are interested in, Google Maps API will transmit the GPS coordinates of the restaurant to the presentation layer.

5.2.1 Assumptions

- System Services data has been processed already.
- Presentation Layer displays map according to Google Maps data.
- Compatible and reliable external services.

5.2.2 Responsibilities

- User Interface navigational decisions are made based on events received in this subsystem.
- Many app functionalities are implemented in this subsystem.

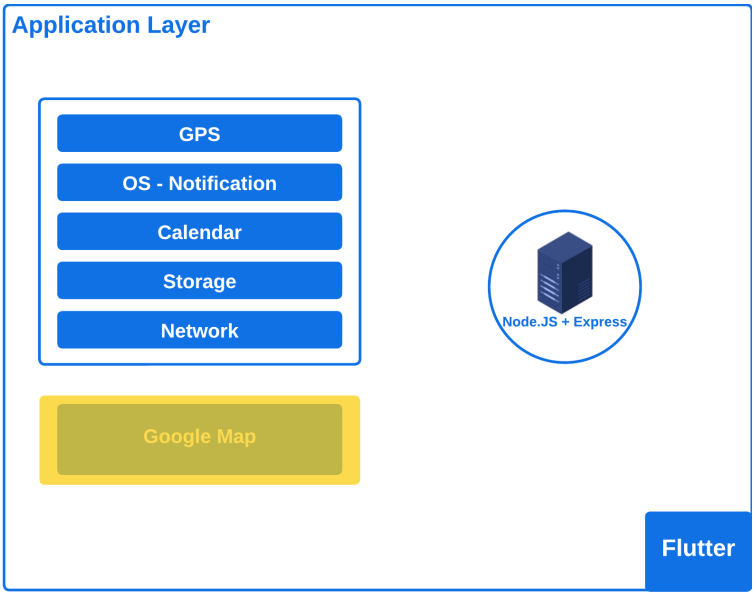

Figure 9: Google Maps Subsystem diagram

- Response time should be reasonable for the app to run smoothly.
- It handles errors in any of the above.

5.2.3 Subsystem Interfaces

Table 5: Subsystem interfaces

ID	Description	Inputs	Outputs
#15	Initial server connection setup	authentication credentials	connection status
#16	User event handling	processed event	triggered action

5.3 Subsystem 3: Node JS + Express

This subsystem 3 handles communication with backend services, APIs, and databases to fetch and store data. Node.JS and Express will be used as a serverless system setup to connect the backend layer and its subsystem to this layer. This subsystem is in charge of sending or retrieving information from the GourmetBook DB, as well as sending payment information to our payment gateway for authorization.

5.3.1 Assumptions

- Back-end services are available and reliable.
- Connection is already established to back-end services.
- The amount of data transmitted will not overwhelm back-end services.

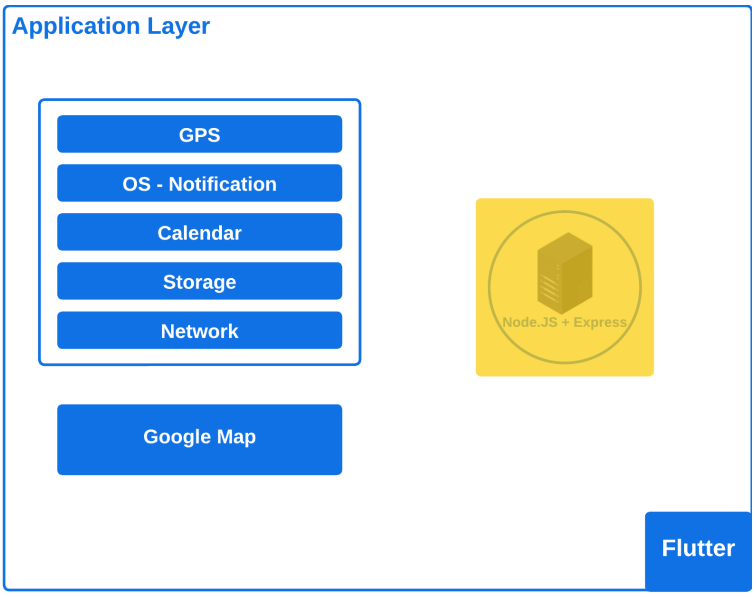

Figure 10: Node JS + Express diagram

5.3.2 Responsibilities

- It constructs valid HTTP requests with the appropriate HTTP methods (GET, POST, PUT, DELETE) based on the frontend's needs (using Express).
- HTTPS encryption is used to ensure sensitive data is transmitted securely.
- It includes the necessary credentials or tokens in requests to authenticate the front-end with the backend.
- The backend will have rate limits regarding how many requests they can handle at a time, and manage requests accordingly.
- User authentication/authorization
- It retrieves the status of transactions
- Error handling, retry, or timeout if back-end request fails

5.3.3 Subsystem Interfaces

Table 6: Subsystem interfaces

ID	Description	Inputs	Outputs
#17	Data transformation	raw back-end data in JSON	front-end friendly output
#18	Front-end requests	HTTP requests	back-end response (JSON)
#19	Data security for back-end	encrypted data	decrypted data
#20	Error handling	failed request	error message

6 Data/Backend Layer Subsystems

The data/backend layer serves as a secure data storage and end host system of the data communications between the application and necessary resources. Instead of directly communicating with and accessing the databases, the application requests the middle agent, which is the node.js server, to send request(s) on behalf of the authorized users, and to respond back to the application with the data accesses in the databases. In this manner, it can achieve the highly developed security of the application and data communication, and in case of failing to acquire the data needed, the middle agent can respond with a warning message instead of crashing the entire process of the application.

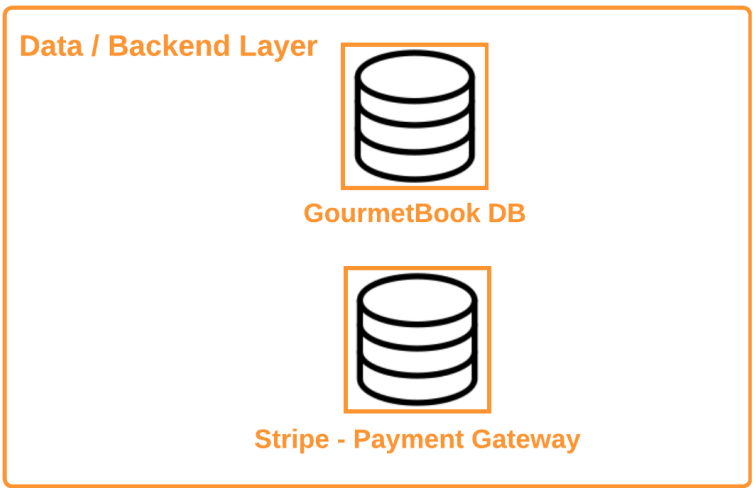

Figure 11: Data/backend layer diagram

6.1 Subsystem 1: GourmetBook DB

When the application is opened by the user the first thing that they will see is the login screen. the application will store data for the username and password. The data of all the restaurants listed will be stored. Also to give our customers an enhanced view the application has given them options to view the menu. All the menus from the restaurants will be stored on this database as well. All the past and future reservations will be stored on the database.

6.1.1 Assumptions

Users will have to create their own unique username and password. Creating an account will also require a unique email address that hasn't been used before on our database. When the user has finally created their login they will have to type in their zip or address so from our database the application will pull out and display all the possible restaurants near them. Once they click on one of the restaurants they will be able to view the menu for that particular restaurant. If they have selected to go to a particular restaurant they can create a reservation for it and this reservation will be stored on our database with a unique id. So, every time a user logs into their account they are displayed with their previous reservation or their upcoming reservation.

6.1.2 Responsibilities

Login - username should be unique so if there are two people with the same name they have to either have a unique way to display their name by using caps or small letters or they will have to use numbers or special characters to have a unique username.
Password should be unique the application will make sure that all our customers make their password

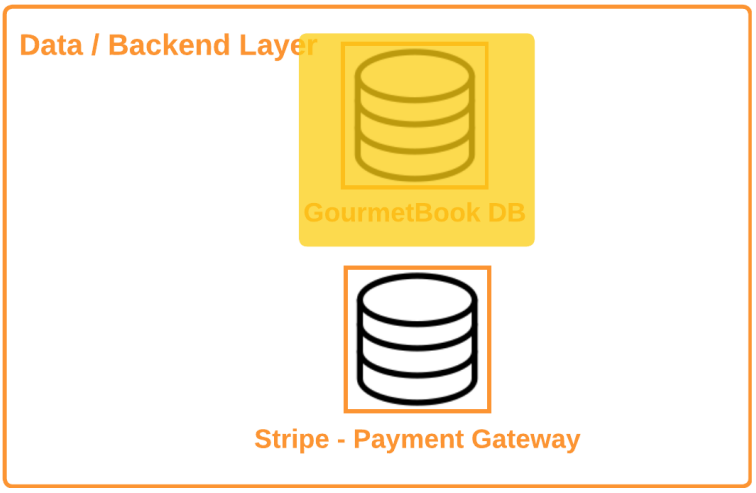

Figure 12: GourmetBook DB

using 8 letters should be one capital rest small number and one special character.

Restaurants - The data stored on our database will have a list of all the fine dining restaurants each restaurant will be saved with a unique id. This unique id will have one picture of the restaurant their name, address, and phone number. Also, each restaurant will have its own menu.

Reservation - Once the user has decided on their fine dining restaurant they can create a reservation this reservation will have their first name, last name, phone number, time of dining, date of dining, and their payment details.

6.1.3 Subsystem Interfaces

Table 7: Subsystem interfaces

ID	Description	Inputs	Outputs
#21xx	JSON base NoSQL Database Management System	input 1 input 2	output 1
#22	Description of the interface/bus	N/A	output 1

6.2 Subsystem 2: Stripe - Payment Gateway

A payment gateway is a technology used by merchants to accept debit or credit card purchases from customers. The term includes not only the physical card-reading devices found in brick-and-mortar retail stores but also the payment processing portals found in online stores. the application will be using stripe payment API to store card detail.

6.2.1 Assumptions

Once the user has created a reservation they will need to provide their card details in case they cancel their reservation on time they will need to pay some fees to restaurants as well as are company. Stripe uses a Payment Intent object to represent an intent to collect payment from a customer, tracking charge

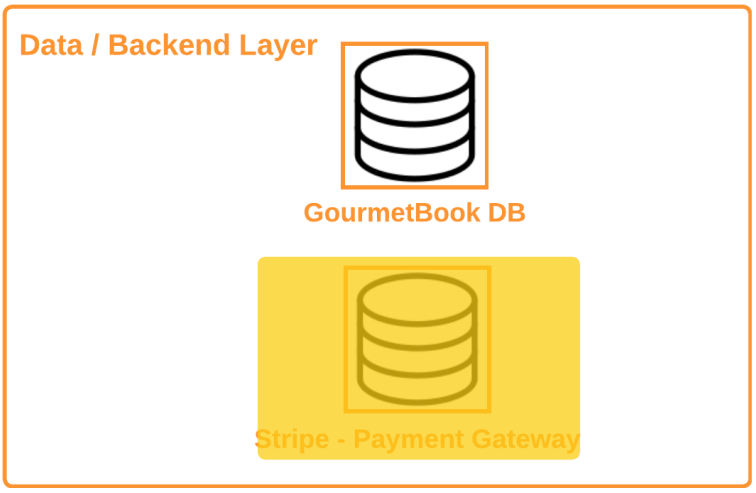

Figure 13: Stripe - Payment Gateway diagram

attempts and payment state changes throughout the process. Always decide how much to charge on the server side, a trusted environment, as opposed to the client. This prevents malicious customers from being able to choose their own prices. Also if the payment has successfully been authorized the user will get a confirmation email.

6.2.2 Responsibilities

Stripe API will help us store card details. the application will only charge customers when they have canceled or ordered food online. This can be done by storing the card detail for later payments. If the payment fails the application will also let the user know that the payment has failed and you will need to update your card details.

6.2.3 Subsystem Interfaces

Table 8: Subsystem interfaces

ID	Description	Inputs	Outputs
#23	Stripe API	Card number Exp date cvv	token authorization
#24	Conformation letter	Card approved	email conformation

References