VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

PROGRAMMING INTEGRATION PROJECT (CO3103)

Reservee: Restaurant Seat Reservation Web Application

Supervisor: Professor Trương Tuấn Anh

Students: Nguyễn Ngọc Quang - 2053360
Nguyễn Trấn Quốc - 2053384
Nguyễn Thanh Tùng - 2152337
Trương Vĩnh Phúc - 2152889
Nguyễn Mạnh Thanh - 2152962
Lê Đức Thuận - 2053467
Lâm Gia Trúc - 2052764

HO CHI MINH CITY, SEPTEMBER 2023

Contents

1 Requirement Elicitation

1.1 Domain Context

Restaurants have a long-standing tradition of serving delectable food and beverages to patrons. In the past, customers would physically visit a restaurant to savor their meals. However, the evolution of technology has significantly transformed this dining experience. With the advent of telecommunication technology, some clients began making reservations via phone calls to secure their spot at their preferred restaurant. The World Wide Web has taken this convenience a step further, enabling customers to easily reserve seats and even pre-order their favorite dishes online.

The objective of this project is to develop a comprehensive system that caters to various aspects of the reservation and ordering process. While this technological advancement greatly enhances the dining experience, it also introduces a new set of challenges. Specifically, the system must effectively handle the concurrent demands of multiple customers attempting to make reservations simultaneously. This complex issue will be thoroughly explored and addressed in greater detail in subsequent chapters.

The Restaurant Seat Reservation Web Application operates within a dynamic domain, shaped by a multitude of stakeholders and elements. This domain context comprises of several key components:

1. **User Environment**
    - **Customers:** The primary users of the application are customers seeking to reserve seats at various restaurants. They have diverse preferences and requirements when it comes to dining experiences.
    - **Restaurant Owners:** Restaurant owners represent another critical user group. They rely on the application to manage their restaurant profiles and handle reservations efficiently.
    - **Authentication Services:** The application incorporates user registration and authentication via their social media profile, allowing users to create accounts and log in using social media profiles, ensuring a secure and personalized experience.
2. **Reservation Process**
    2.1. **Finding restaurants**
        - **Restaurant Data:** The application aggregates restaurant data, including essential details such as photos, names, locations, cuisines, menus, ratings, available reservation slots, and seating capacities. This information is curated to aid customers in making informed choices.
        - **Search and Filtering:** Customers can search for restaurants based on keywords, such as name, location, and cuisine, and apply filters to narrow down their options based on preferences.

    2.2. **Reservation Creation**
        - After choosing a preferred restaurant, the customer goes on to authenticate themselves if they have not logged in yet. Then proceed to deposit the appropriate amount of money set by the restaurant owner (yet limited by our platform policy to be between ... and ...).

    2.3. **Reservation Management**
        - **Modification:**

  - Clients can change their preferred favorite position in the restaurant if the new seats are available. In addition, they could update the number of guests for a reservation on a condition that the total number of seats is available for reservation.
  - On the restaurant owner side, they could dynamically change the number of available seats during the day to meet their business goals.
- **Cancellation:** Users have the flexibility to cancel reservations, which enhances the convenience of the booking process. Besides, due to accidental or personal causes, clients cannot fulfill their reservation. The system allows the client to actively cancel the reservation only 4 hours before the time dictated on the reservation.
- **Notifications:** The application sends push notifications for reservation information after creating a reservation successfully, and a reminder notification 1 hour before the starting time.

3. **Analytics and Reporting**

   - The restaurant owner could track the total number of reservations each day and the status of each reservation. From these information, the restaurant owner may prepare better for the capacity at peak hour.

4. **Customer and Restaurant Profiles**

   - **Customer Profiles:** Each customer has a dedicated profile where they can manage their reservations, leave reviews and ratings, and access their reservation history. They can search for past reservations based on keywords and apply filters
   - **Restaurant Dashboard:** Restaurant owners have access to secure dashboards where they can manage their restaurant profiles, receive real-time notifications about reservations, and view their reservation history.

This domain context offers a comprehensive overview of the factors and stakeholders shaping the Restaurant Seat Reservation Web Application. It underscores the importance of understanding this context to effectively design, develop, and deploy the application to meet the diverse needs of both customers and restaurant owners while ensuring a secure and user-friendly experience.

1.2 Stakeholders

1. **Customers:** Customers are the main users of the web app. They use it to make reservations, view restaurant details, and provide feedback. Meeting their needs and ensuring a user-friendly experience is essential.

    - **Needs:**
        - **Ease of Reservation:** Customers need a straightforward and efficient reservation process. The app should allow them to quickly find available tables, choose their preferred time, and make reservations with minimal effort.
        - **Restaurant Information:** Customers require access to comprehensive restaurant details, including menus, pricing, location, hours of operation, and contact information. This information helps them make informed decisions.
        - **User-Friendly Interface:** The app should have an intuitive and user-friendly interface, making it easy for customers to navigate, search for restaurants, and complete reservations without confusion.
        - **Customization Options:** Some customers may have specific preferences, such as requesting a quiet corner or noting dietary restrictions. The app should allow customers to customize their reservations where applicable
        - **User Reviews and Ratings:** Access to user-generated reviews and ratings helps customers gauge the quality of restaurants and make informed decisions.
    - **Benefits from application**
        - **Convenience::** The web app offers customers the convenience of browsing and reserving restaurants from the comfort of their homes or on the go, eliminating the need for phone calls or physical visits.
        - **Feedback Influence:** Customers have the opportunity to provide feedback, which can influence restaurant quality and service improvements, ultimately benefiting them in the long run.

2. **Restaurant Owners/Managers:** Restaurant owners or managers are key stakeholders as they rely on the web app to manage reservations, track customer data, and optimize seating arrangements. They may also want to customize their restaurant's profile on the app.

    - **Needs:**
        - **Reservation Management:** Restaurant owners/managers need a robust system to efficiently manage reservations, including the ability to accept, modify, or cancel bookings as necessary.
        - **Customer Data Tracking:** Access to customer data is crucial for building relationships and understanding patron preferences. They need tools to capture and analyze customer information.
        - **Reservation Analytics:** Insights into reservation patterns, peak hours, and customer demographics help in making informed business decisions, such as staffing and marketing strategies.
    - **Benefits from application:**
        - **Efficient Operations:** The web app streamlines reservation management and seating arrangements, reducing manual work and enhancing the overall efficiency of restaurant operations.

- **Data-Driven Decisions:** Access to customer data and reservation analytics enables owners/managers to make data-driven decisions, such as menu changes or marketing campaigns, to boost revenue.
- **Increased Visibility:** A customized restaurant profile on the app provides visibility to a wider customer base, potentially attracting new patrons and increasing reservations.

By catering to the requirements of these stakeholders, the Restaurant Seat Reservation Web Application can yield several advantages. These encompass enhanced user experiences, streamlined management processes, financial oversight, and adherence to legal and compliance standards. These advantages collectively contribute to the establishment of a more effective and user-centric restaurant reservation service, elevating the overall dining experience.

1.3 Requirements

1.3.1 Functional Requirements

1. **Customer Actions:**
    - As a customer, you can create accounts or profiles.
    - As a customer, you can log in or authenticate using email, social media, or a phone number.
    - As a customer, you can search for restaurants based on location, cuisine, rating, or price range.
    - As a customer, you can specify the number of guests in your party
    - As a customer, you can filter and sort search results.
    - As a customer, you can select a restaurant and choose a preferred date and time for your reservation.
    - As a customer, you can confirm and book a table
    - As a customer, you can view and manage your reservations.
    - As a customer, you can leave reviews and ratings for restaurants you've visited.
    - As a customer, you logout and delete your account.
2. **Restaurant Owner Actions**
    - As a restaurant owner, you can access a dashboard to manage reservations.
    - As a restaurant owner, you can confirm or reject reservation requests.
    - As a restaurant owner, you can update restaurant information, such as operating hours, table availability, pictures, number of seats, and maps.
    - As a restaurant owner, you can respond to customer reviews.
    - As a restaurant owner, you logout and delete your account.
    - As a restaurant owner, you can view average ratings for your restaurant.

1.3.2 Non-Functional Requirements

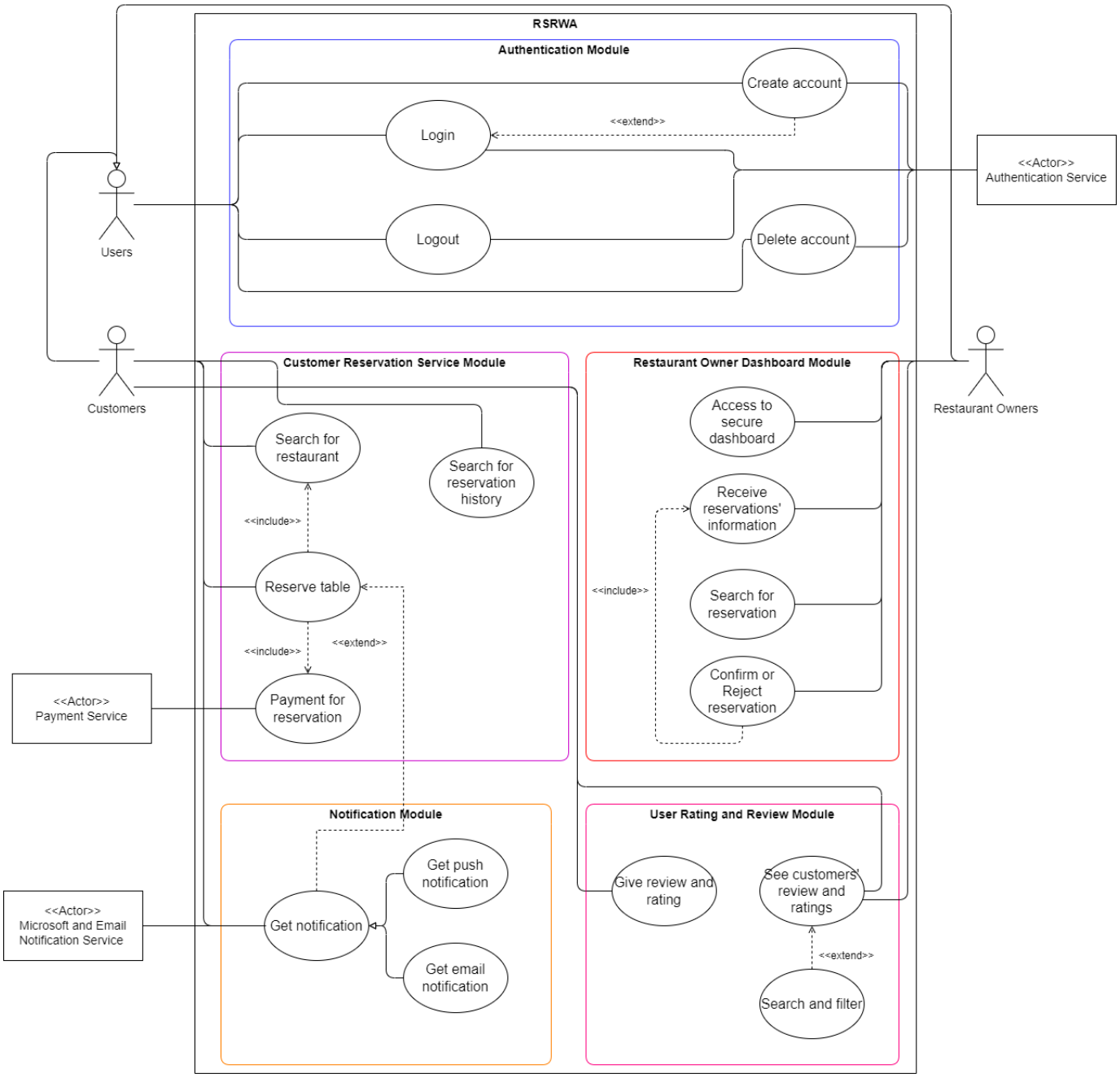
1. **Performance:**
    - The system should provide efficient and responsive performance even during peak usage times.
2. **Security:**
    - Ensure fast and responsive performance for the web app, even during peak booking times.
    - Optimize for quick loading of restaurant listings and reservation forms.
3. **Reliability and Availability:**
    - Ensure high availability of the web app, allowing users to make reservations at any time.
4. **Scalability:**
    - Design the web app to handle a growing number of users and restaurant listings.
    - Use scalable cloud hosting services if needed.
5. **Usability and Accessibility:**

- Ensure that the web user interface is intuitive and accessible to a wide range of users.
- Implement responsive design for mobile devices.

6. **Integration**
   - Implement secure and reliable integration with restaurant systems for reservation management.
   - Integrate with payment gateway services for secure transactions.
7. **Cross-Browser Compatibility:**
   - Ensure that the web app works correctly on major web browsers (e.g., Chrome, Firefox, Safari).
8. **Error Handling:**
   - Implement effective error handling mechanisms and log errors for debugging and troubleshooting.
   - Monitor application logs for issues.
9. **Feedback and Support:**
   - Provide a support channel for user inquiries and issues.
   - Collect user feedback to improve the web app.

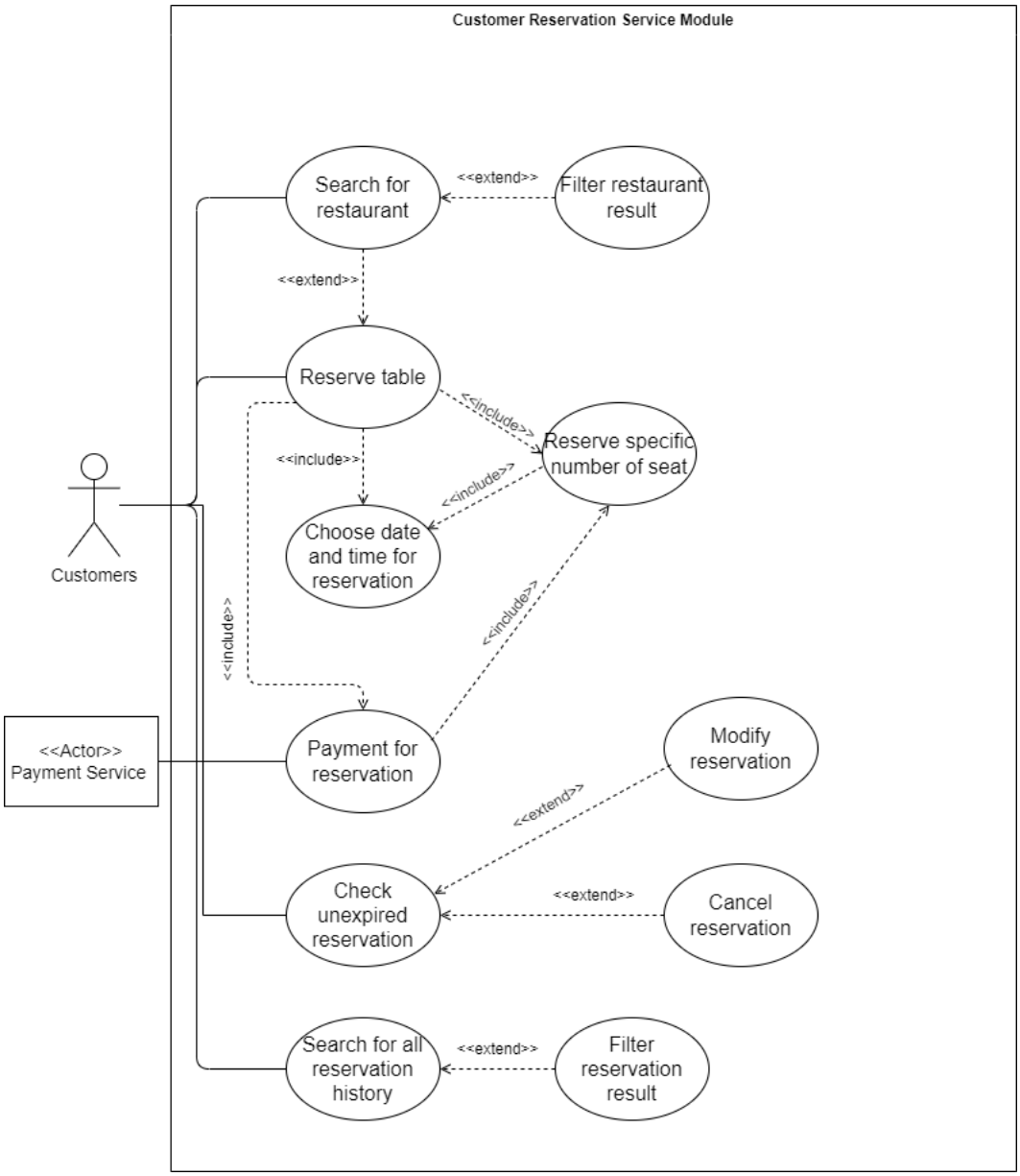
1.4 Use-case Diagram

1.4.1 System Use-case Diagram



1.4.2 Customer Reservation Service Module

1.4.2.a Use-case Diagram



1.4.2.b Use-case Diagram Description

Use-case Name	Search for Restaurant
Actor(s)	Customers
Description	Search for restaurants based on keywords (name, location, cuisine) and view matching results.
Pre-condition(s)	The customer has logged in successfully.
Post-condition(s)	A list of matching restaurants is displayed on the screen.
Trigger	Customers click the "Search" button and provide search criteria.
Normal flow	1. The system receives search criteria from the customer. 2. The system queries the database for matching restaurants. 3. The list of matching restaurants is displayed on the screen.
Alternative flow	No alternative flow.
Exception flow	None.

Use-case Name	Filter Restaurant Result
Actor(s)	Customers
Description	Filter the list of restaurants based on preferences (location, ratings, available reservation times).
Pre-condition(s)	The customer has performed a search for restaurants.
Post-condition(s)	The filtered list of restaurants is displayed on the screen.
Trigger	Customers apply filter criteria (e.g., location, ratings) to refine search results.
Normal flow	1. The customer selects filter criteria. 2. The system applies the selected filters to the list of restaurants. 3. The filtered list of restaurants is displayed on the screen.
Alternative flow	No alternative flow.
Exception flow	None.

Use-case Name	Reserve table
Actor(s)	Customers
Description	Reserve table at the chosen restaurant.
Pre-condition(s)	1. Customers have logged in their account successfully. 2. Customers needs to ensure connection throughout the process.
Post-condition(s)	Customers are able to start specify detail about the reservation.
Trigger	Students click the "Reserve table" option button located at top right of the restaurant's box which they have searched for.
Normal flow	1. Customers choose the "Reserve table" option. 2. System gets restaurant information including empty seats, working hours, branches. 3. System display a box for reservation.
Alternative flow	Alternate 1: Customers can click the "Cancel" button option to close the box.
Exception flow	None

Use-case Name	Choose date and time for reservation
Actor(s)	Customers
Description	Choose date and time for the reservation from the available hours of the chosen restaurant.
Pre-condition(s)	1. Customers have opened the reservation box. 2. System has successfully retrieve available date and time from the database 3. Customers needs to ensure connection throughout the process.
Post-condition(s)	Customers successfully choose their wanted date and time.
Trigger	Customers click the "Choose date and time" option button.
Normal flow	1. Customers choose the "Choose date and time" option button. 2. System display a box for choosing date and time. 3. Customers choose an available date through a calendar. 4. System gets restaurant's working hours of that date. 5. Customers choose an available time. 6. Customers click "Confirm" button to confirm their choice.
Alternative flow	*Alternate 1*: Before step 6, Customers can choose "Cancel" button to go back to reservation box.
Exception flow	Exception 1: At step 6, when customers clicks "Confirm" button, if the available date or time are changed by the restaurant owners during the process, customers will receive a warning demanding them to choose the date and time again.

Use-case Name	Reserve specific number of seats
Actor(s)	Customers
Description	Choose a number of seats from the available empty seats.
Pre-condition(s)	1. Customers have opened the reservation box. 2. System has successfully retrieve available seats from the database 3. Customers needs to ensure connection throughout the process. 4. Customers needs to choose date and time for reservation.
Post-condition(s)	Customers successfully choose their wanted number of seats.
Trigger	Customers click the "Choose number of seats" option button.
Normal flow	1. Customers choose the "Choose number of seats" option button. 2. System display a box for choosing the seats. 3. Customers enter number of seats. 4. Customers click "Confirm" button to confirm their choice. 5. System gets restaurant's available seats. 6. System checks if the restaurant has enough seat. 7. If there is enough seats, the process is done successfully.
Alternative flow	*Alternate 1*: at step 7 7a. If there is not enough seats, the system will give a warning, demanding customers to choose again, together with the number of seats remaining. *Continue step 3 in the normal flow* *Alternate 2*: at step 5 5a. Customers can choose "Cancel" button to go back to reservation box.
Exception flow	None

Use-case Name	Payment for reservation
Actor(s)	Customers and Payment service
Description	Customers pay for their reservation using Payment service.
Pre-condition(s)	1. The customers have chosen date and time together with the number of seats. 2. The payment service must be available.
Post-condition(s)	Customers have successfully pay for their reservation.
Trigger	Customers click "Pay" button on the reservation box
Normal flow	1. Customers choose "Pay" option button. 2. The system takes in the bill to database and display a payment box 3. Customers choose the payment method, including various kind of credit card, e-wallet, and direct payment. 4. The system receive the choice and make a connection to the chosen payment service for transaction. 5. Customers make the transaction. 6. The system receives the money and notify customer.
Alternative flow	*Alternate 1:* at step 4 4a. If the customer choose direct payment, the system will confirm the reservation and end the process. 4b. If the customers' bank accounts do not have enough money, the system will give a warning. Continue step 3 in normal flow *Alternate 2:* before step 5 Customers can choose "Cancel" option button to go back to the reservation box.
Exception flow	*Exception 1*: at step 4 4a. If system fails to connect to the payment service due to unexpected reason, it will give customers a warning. Continue step 3 in normal flow *Exception 2*: at step 6 6a. If the customers already finish the transaction and receive confirmation about the transaction from the payment service, but there is no notification of the system indicating it has receive the money, the customers can contact the customer service to solve the problem.

Use-case Name	Check unexpired reservation
Actor(s)	Customers
Description	Check unexpired reservations of a customer
Pre-condition(s)	The customer has logged in successfully.
Post-condition(s)	1. The list of unexpired reservations is displayed on the screen. 2. The options to "modify reservation" and "cancel reservation" is displayed on the screen.
Trigger	Customers click the "Check unexpired reservation" button.
Normal flow	1. The system retrieves information about the customer's unexpired reservations from the database. 2. The customer's unexpired reservations is displayed on the screen.
Alternative flow	No alternative flow.
Exception flow	At step 1 of the normal flow, the system fails to retrieves information about the customer's current reservations.

Use-case Name	Modify reservation
Actor(s)	Customers
Description	The customer modifies the current reservation.
Pre-condition(s)	Cancel the current reservation.
Pre-condition(s)	1. The customer has logged in successfully. 2. The customer has at least one current reservation.
Post-condition(s)	1. The window to start the modification process is displayed.
Trigger	Customers click the "Modify reservation" button.
Normal flow	1. The system retrieves information about the customer's current chosen reservation from the database. 2. The customer's chosen reservation's detail is displayed on the screen. 3. The options to choose the use-case "Choose date and time for reservation" and "Reserve specific number of seat" is displayed. 4. Extra pay amount or refunds will be display on the screen, which will be account for when the customer go to the restaurant.
Alternative flow	The customer chooses the cancel modify reservation option.
Exception flow	At step 1 of the normal flow, the system fails to retrieves information about the customer's current chosen reservation from the database.

Use-case Name	Cancel reservation
Actor(s)	Customers
Description	Cancel the current reservation.
Pre-condition(s)	1. The customer has logged in successfully. 2. The customer has at least one current reservation.
Post-condition(s)	The current reservation is successfully cancelled.
Trigger	Customers click the "Cancel reservation" button.
Normal flow	1. System display a confirmation box. 2. System display the refund amount which customer need to go to the restaurant to take it back. 3.The information about the cancelled reservation will no longer be displayed on the screen. 2. The system will remove the information about the cancelled reservation in the database. 3. The system will update the availability of the restaurant accordingly.
Alternative flow	The customer can click the "Cancel" button to cancel the cancel reservation option.
Exception flow	None

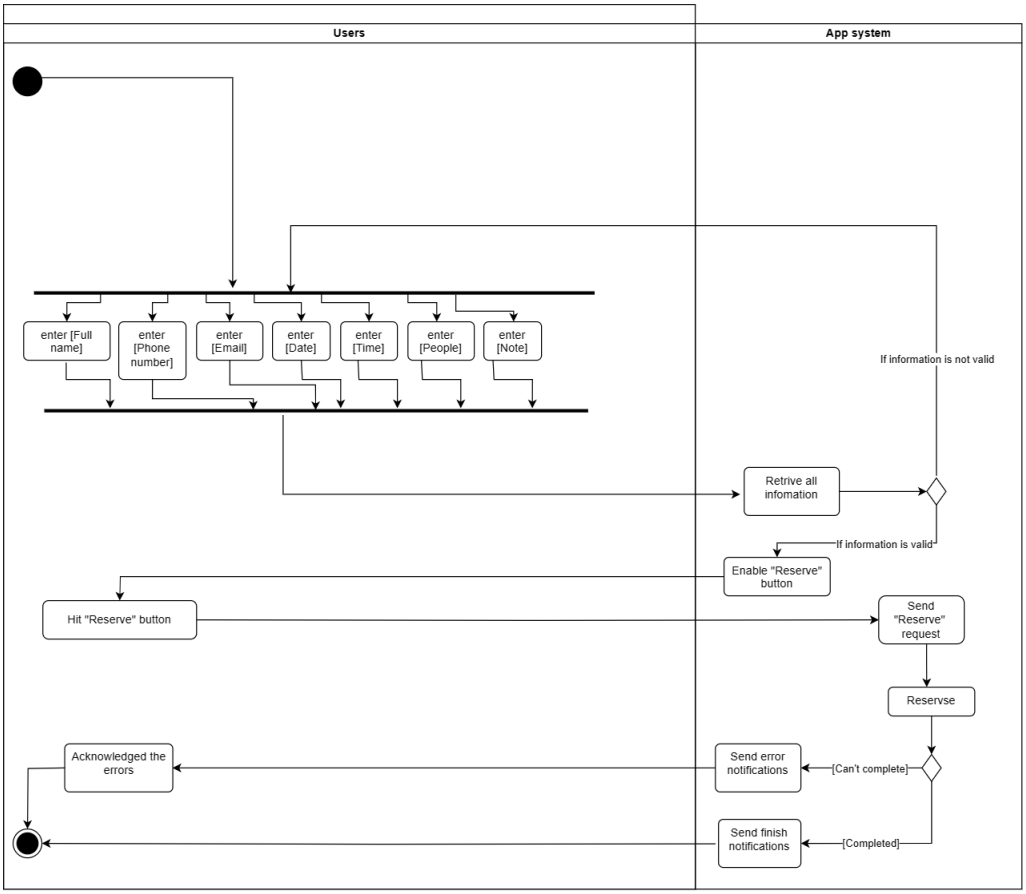
Use-case Name	Search for Reservation History
Actor(s)	Customers
Description	Search for reservation history based on keywords (restaurant names and locations) and view matching results.
Pre-condition(s)	The customer has logged in successfully.
Post-condition(s)	A list of matching reservation history entries is displayed on the screen.
Trigger	Customers click the "Search History" button and provide search criteria.
Normal flow	1. The system receives search criteria from the customer. 2. The system queries the database for matching reservation history entries. 3. The list of matching reservation history entries is displayed on the screen.
Alternative flow	No alternative flow.
Exception flow	None.

Use-case Name	Filter Reservation Result
Actor(s)	Customers
Description	Filter the list of reservation history entries based on preferences (date and time).
Pre-condition(s)	The customer has searched for reservation history.
Post-condition(s)	The filtered list of reservation history entries is displayed on the screen.
Trigger	Customers apply filter criteria (e.g., date and time) to refine history results.
Normal flow	1. The customer selects filter criteria (e.g., date and time). 2. The system applies the selected filters to the reservation history entries. 3. The filtered list of reservation history entries is displayed on the screen.
Alternative flow	No alternative flow.
Exception flow	None.

2 System Modelling

2.1 Activity Diagram

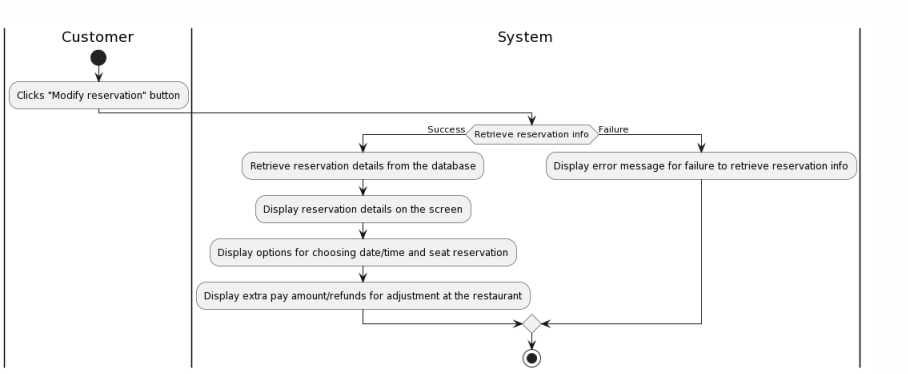
2.1.1 Reserve table



Description

The "Reserve Table" activity diagram depicts the sequential steps initiated by customers enter the information about: Full name, Phone number, Email, Reserve Date, Reserve Time, Number of People, and Notes about the Reservation. It then illustrates the process where the system retrieves details of all the information. Next, the system proceeds to verify the validity of the information. If the requested reservation is feasible, the diagram continues to the next step, which enable the “Reserve” button. If the reservation is completed, the system will send a finish notification. Otherwise, it will prompt an error.

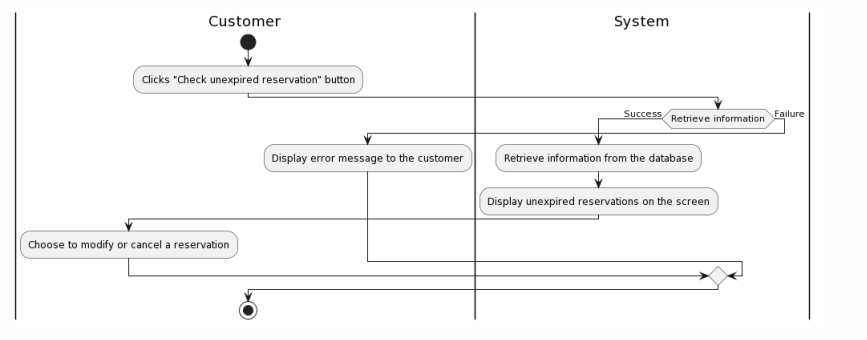
2.1.2 Modify reservation



Description

The "Modify Reservation" activity diagram depicts the sequential steps initiated by customers clicking the corresponding button within the system. It illustrates the process where the system retrieves details of the current reservation from the database, displaying these specifics on-screen for customer review. This prompts options for adjusting reservation parameters such as date, time, and seat selection. Additionally, the diagram highlights the display of financial implications, showcasing any necessary extra payments or refunds due to modifications, thus ensuring customers are informed about potential financial adjustments for their upcoming restaurant visit.

2.1.3 Check unexpired reservation



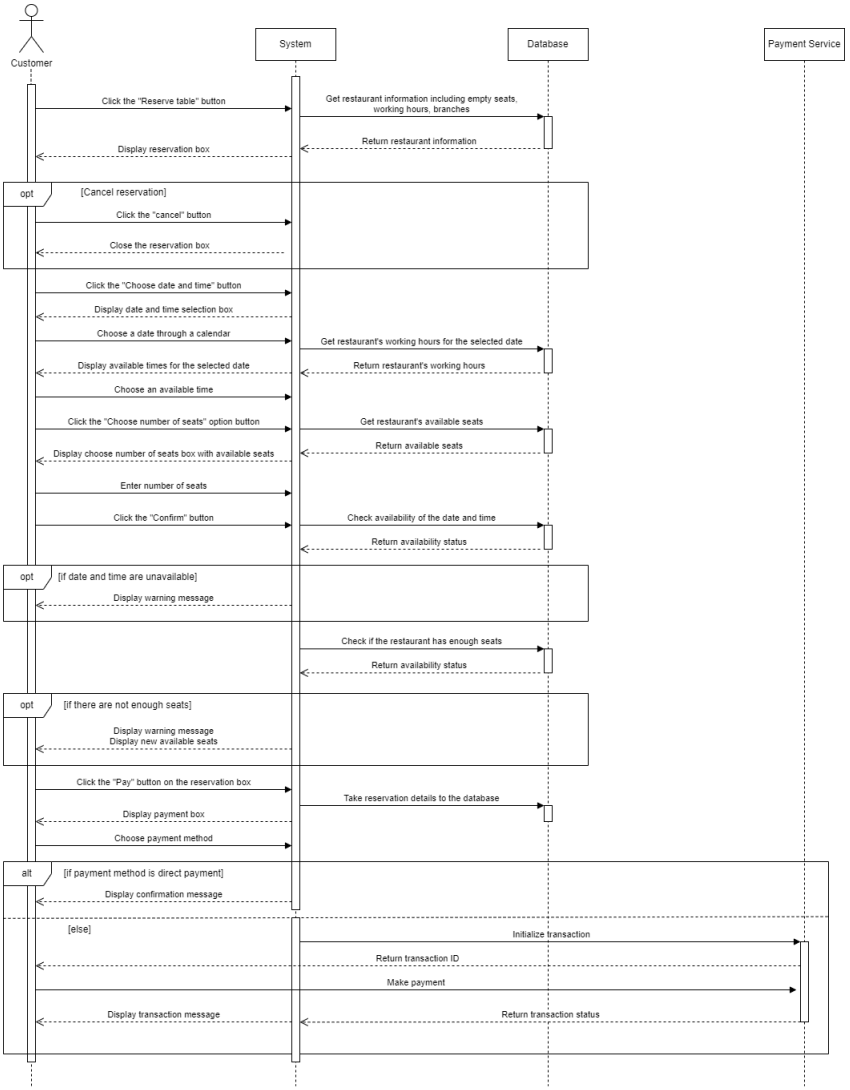
Description

The activity diagram captures the reservation process within the restaurant’s online platform. Customers start by initiating reservations through the "Check unexpired reservation" button, prompting the system to retrieve their existing reservations from the database. Upon successful retrieval, the system displays these reservations, granting customers the option to modify or cancel bookings. In case of a system failure during retrieval, an error message promptly notifies customers. This visual representation delineates each

step of the process, illustrating interactions between users and the system, ensuring a seamless reservation experience while addressing potential issues that may arise.

2.2 Sequence Diagram

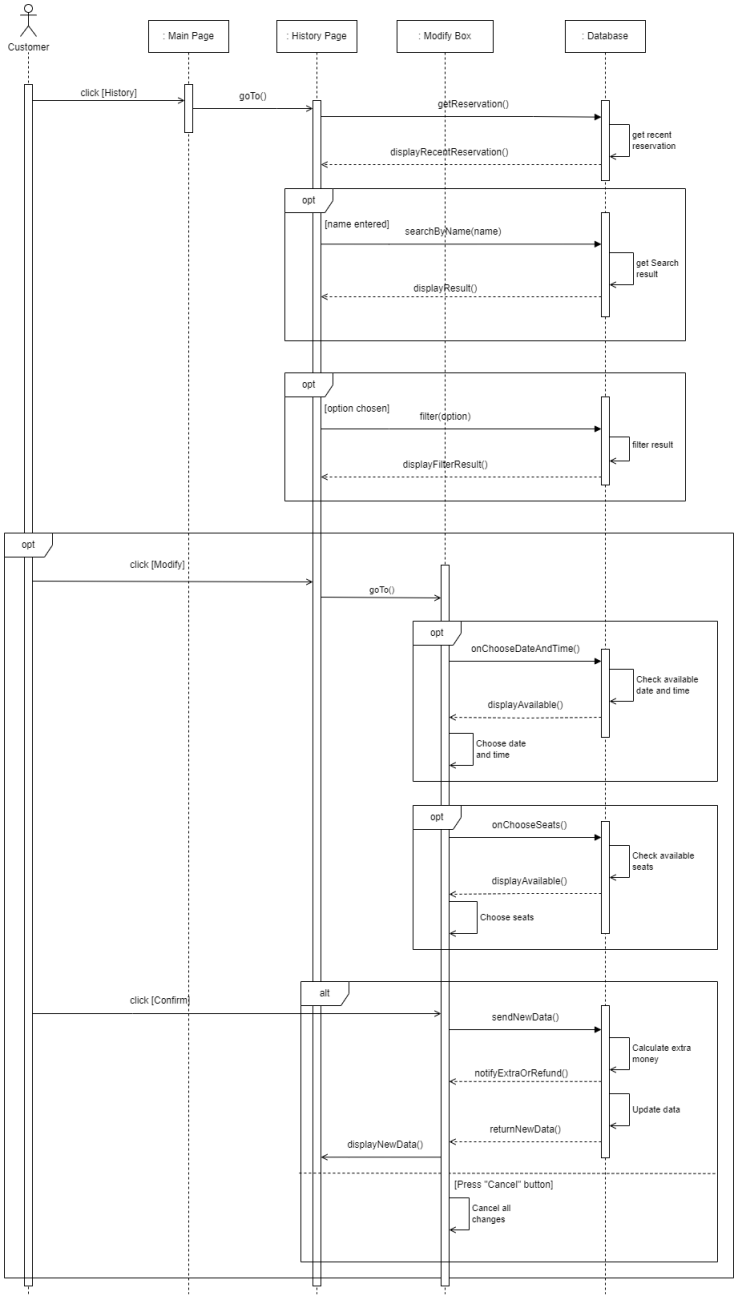
2.2.1 Reserve table



Description

The "Reserve Table" activity diagram depicts the sequential steps initiated by customers enter the information about: Full name, Phone number, Email, Reserve Date, Reserve Time, Number of People, and Notes about the Reservation. It then illustrates the process where the system retrieves details of all the information. Next, the system proceeds to verify the validity of the information. If the requested reservation is feasible, the diagram continues to the next step, which enable the “Reserve” button. If the reservation is completed, the system will send a finish notification. Otherwise, it will prompt an error.

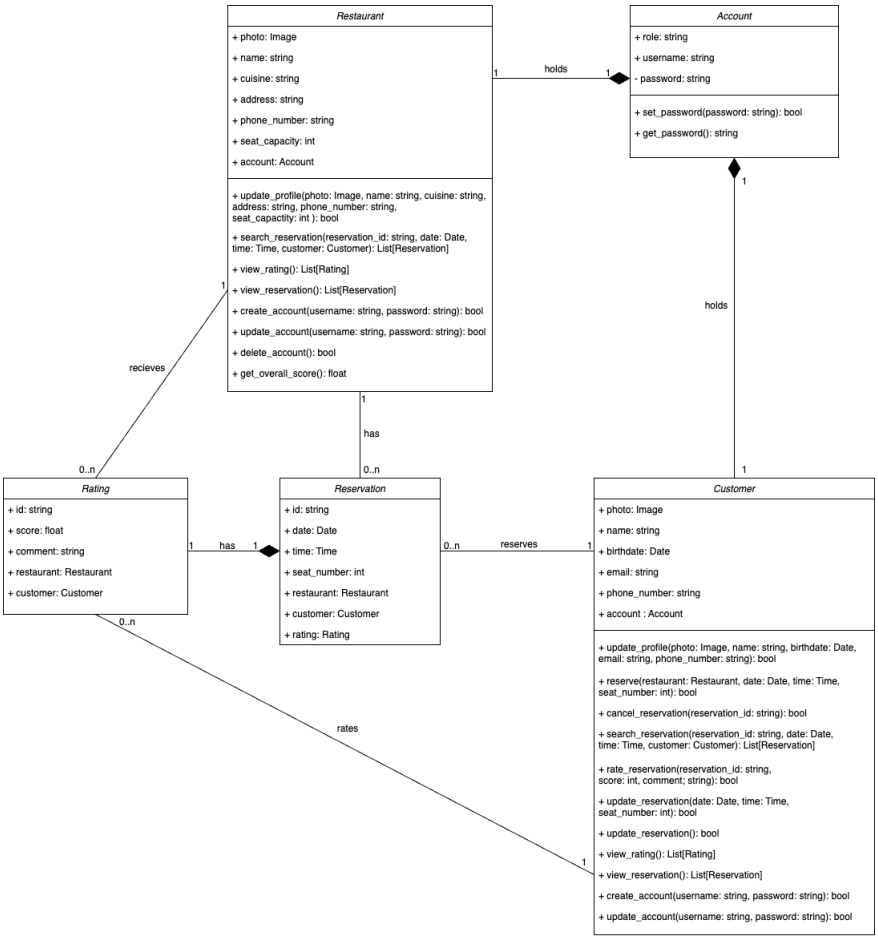
2.2.2 Manage reservation



Description

The "Manage Reservation" sequence diagram depicts the sequential steps initiated by customers clicking the "History" button within the system. It illustrates the process where the system retrieves details of the current reservation from the database, displaying these specifics on-screen for customer review. Customers will have an option to search specific reservation histories by name, and also filter the result. The system also prompts options for adjusting reservation parameters such as date, time, and seat selection. Additionally, the diagram highlights the display of financial implications, showcasing any necessary extra payments or refunds due to modifications, thus ensuring customers are informed about potential financial adjustments for their upcoming restaurant visit.

2.3 Class Diagram



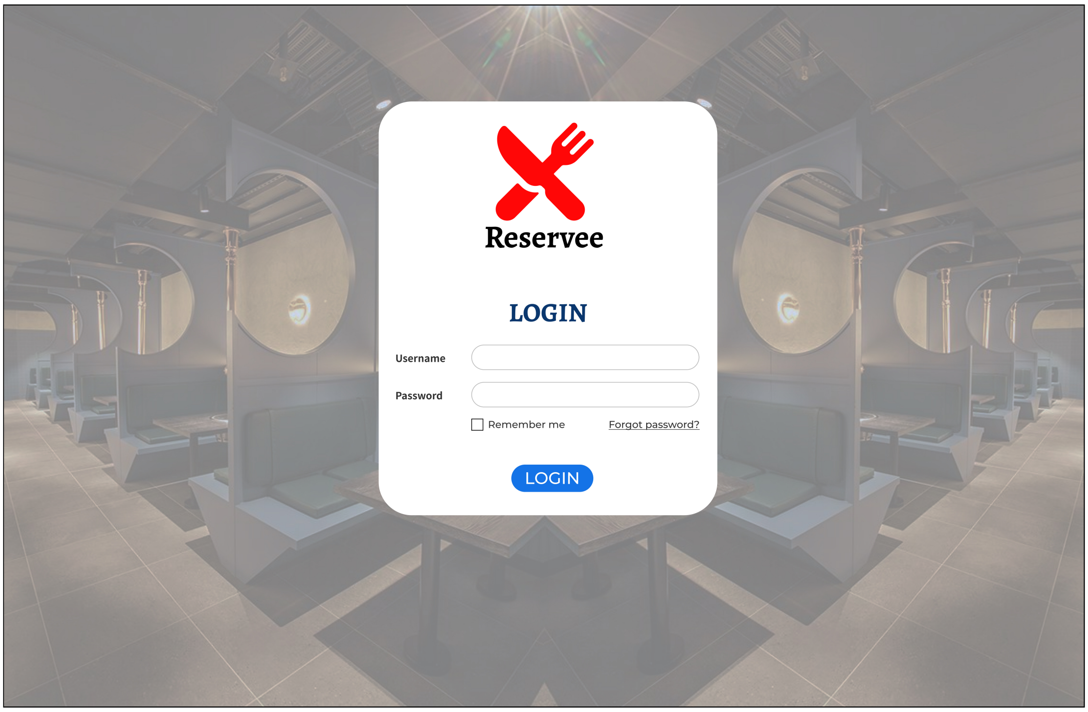
Description

The class diagram for the restaurant seat reservation system illustrates the system's core components and their interconnections, providing a visual representation of the system's primary functions. The system has five main classes: Restaurant, Customer, Account, Reservation, and Rating.

- Account includes role, username, and password. One account is shared with one user only.
- Restaurant consists of all public attributes, including photo, name, cuisine, address, phone number, seating capacity, and account. They can update their profiles, manage accounts, and view reservations and ratings. Each restaurant holds only one account and receives many ratings and reservations. When the account is deleted, the restaurant linked to that account no longer exists in the system.
- Student consists of all public attributes, including photo, name, birthday, email, phone number, and account. They can update their profiles, manage accounts or reservations, and view the reservations and ratings. Each student holds only one account, rates many ratings, and reserves many reservations. When the account is deleted, the student linked to that account no longer exists in the system.
- Reservation has the ID, date, time, number of seats, restaurant, customer, and rating. One customer can make one reservation for one restaurant only.
- Rating has the ID, score, comment, restaurant, and customer. A rating rates one restaurant by one customer for one reservation.

2.4 UI

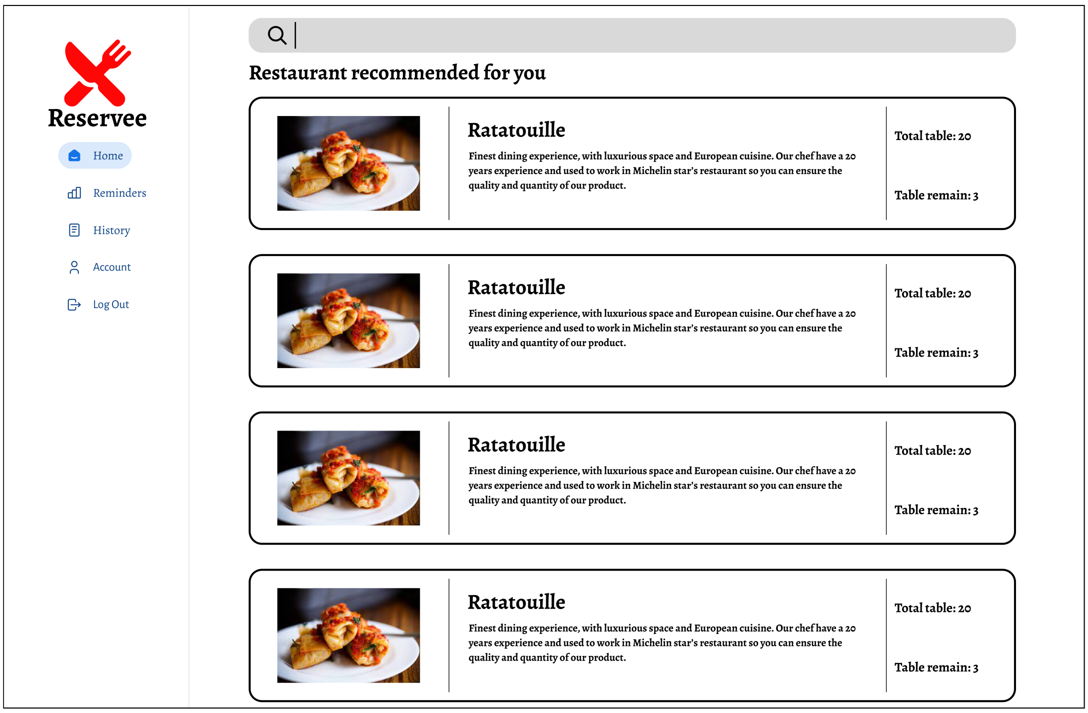
2.4.1 Login



Upon clicking the page, users are prompted to authenticate by logging into our system, entering their username and password for access. They can opt for the "Remember me" feature to retain their login information for future visits, and there's a "Forgot Password" button available in case users need to recover their password.

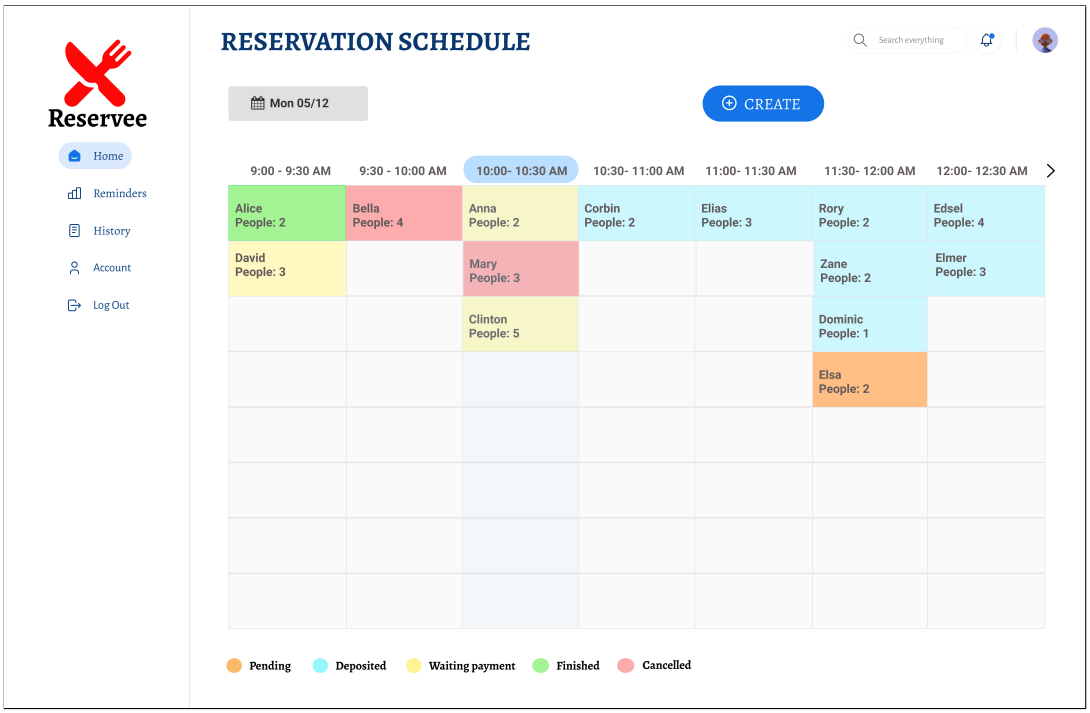
2.4.2 Main page

2.4.2.a Homepage

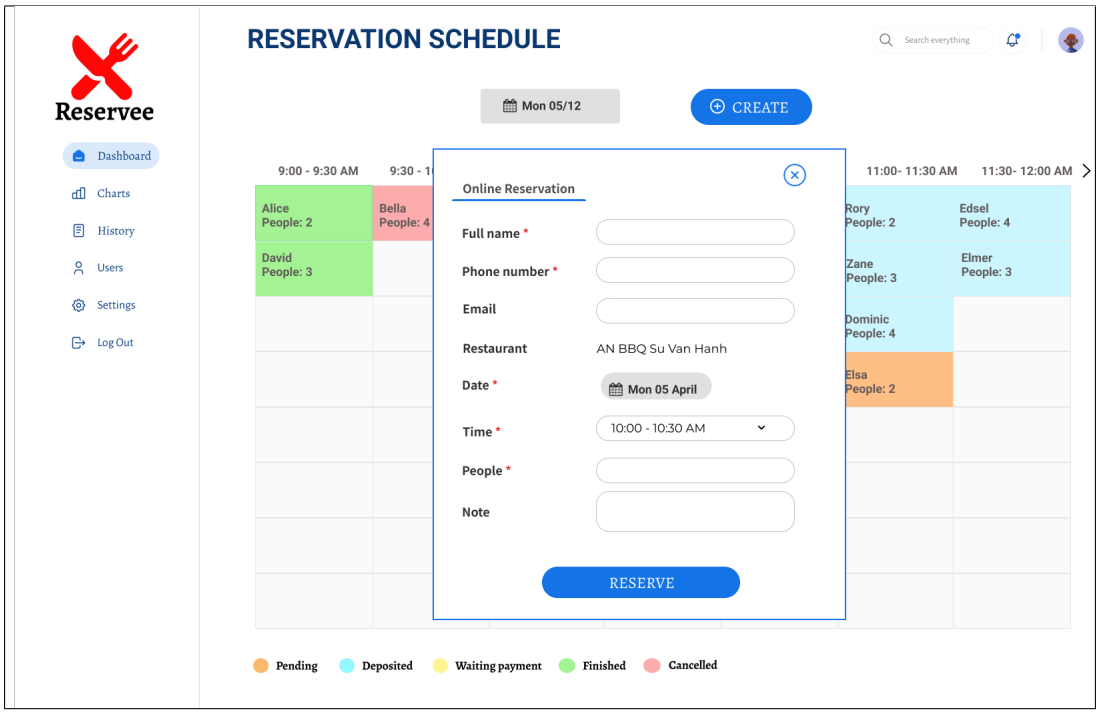


In the main homepage, there will be a navigation bar on the left and content section on the right. In the navigation bar, we have 5 options: Home, Reminder, History, Account and Log out.
In the homepage, a list of restaurant will be displayed and in each section, there will be a restaurant's name, description, total of table and table remaining so that user will know which to choose. A search bar is at the top and user can search for their preferred restaurant.

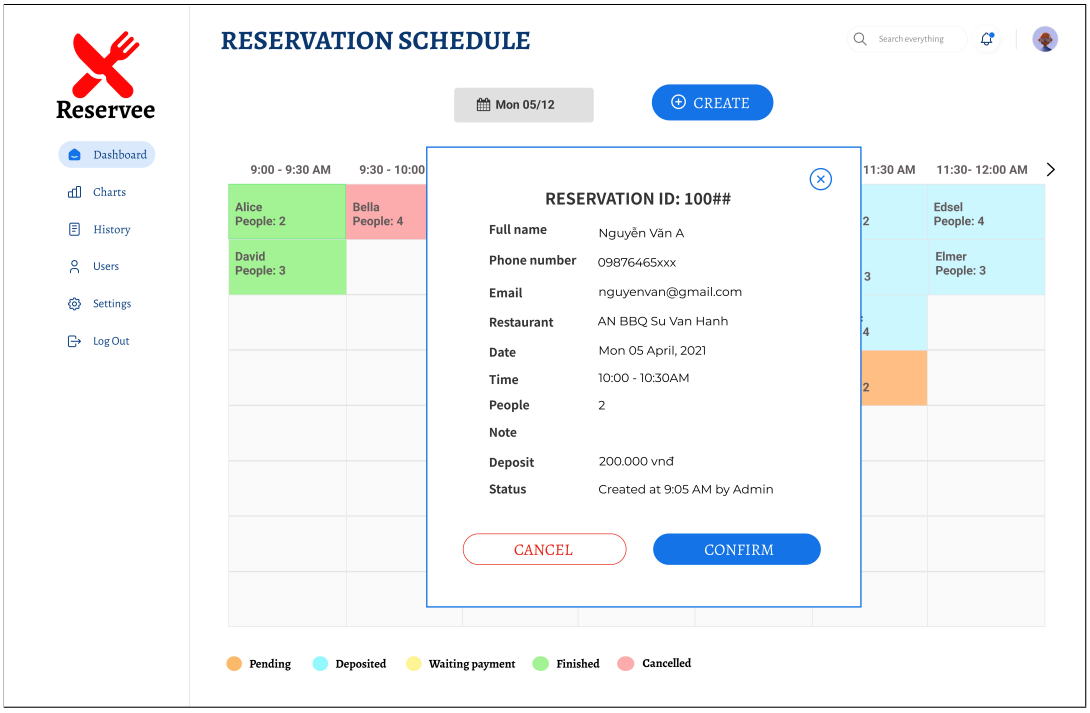
2.4.2.b Restaurant reservation



When click on the restaurant section, user will be redirected to the reservation schedule which displayed by a table, where they can see which table at which time are still occupied. When occupied, the section of the table will have color and the name of the customer and their table will be displayed to indicate whether the user can book the table or not. To create a reservation, user will click on "Create". A pop up form will appear.



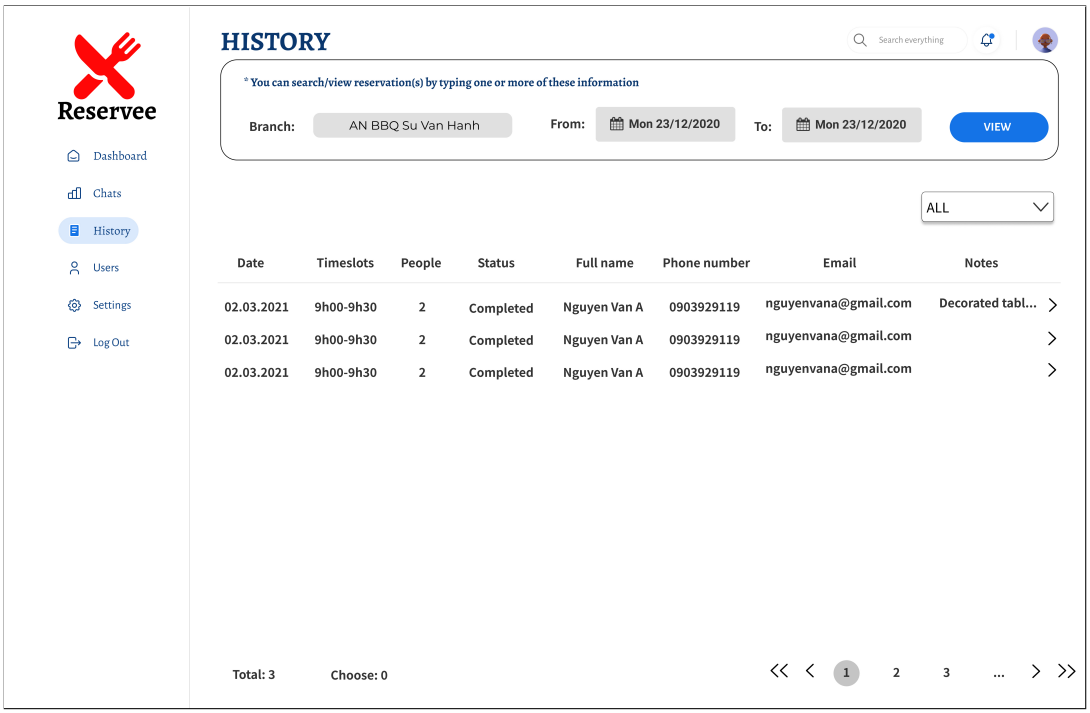
In here, user's info will automatically being filled, they only need to choose the appropriate time and note for the restaurant. Then click "Reserve" and it will be done.

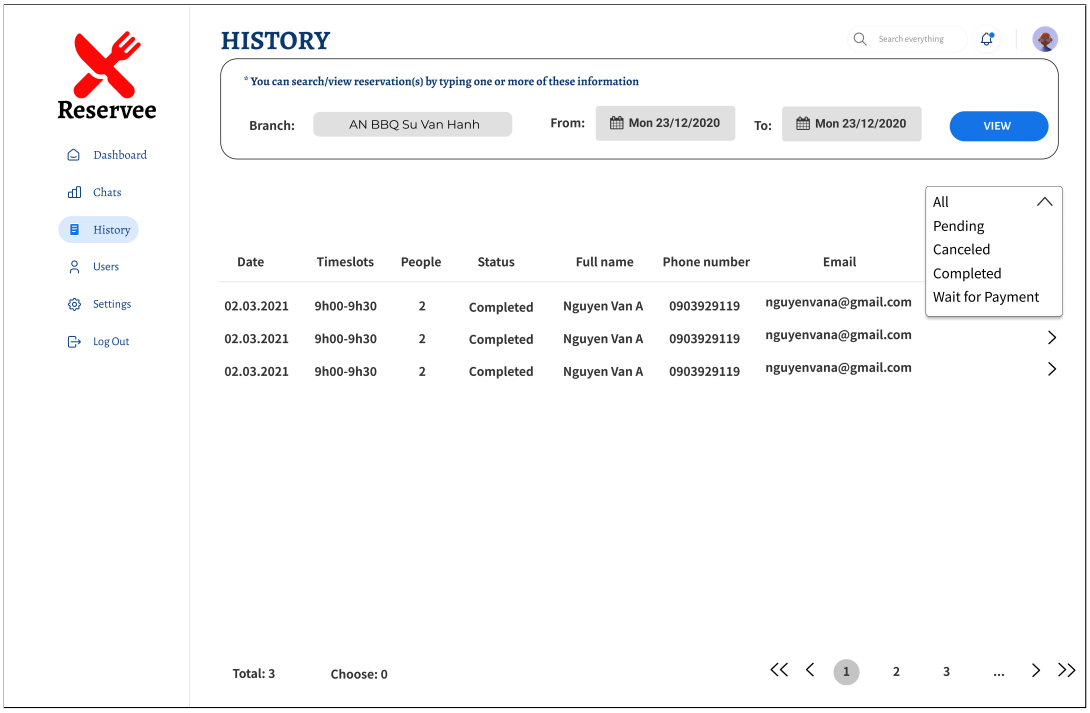


Here is where everything is check again and then user can choose whether they confirm or cancel the reservation.

2.4.3 History

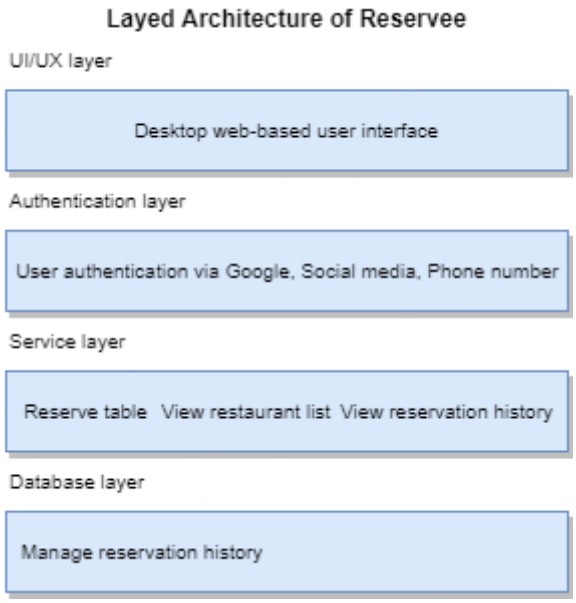
In history, user can check their past reservation as well as the status of the reservation. There will be a filter button so that user will know what status their reservation is.


Reservee
Dashboard
Chats
History
Users
Settings
Log Out
HISTORY
Search everything
* You can search/view reservation(s) by typing one or more of these information
Branch:
AN BBQ Su Van Hanh
From:
Mon 23/12/2020
To:
Mon 23/12/2020
VIEW
ALL
Date
Timeslots
People
Status
Full name
Phone number
Email
Notes
02.03.2021
9h00-9h30
2
Completed
Nguyen Van A
0903929119
nguyenvana@gmail.com
Decorated tabl...
Total: 3
Choose: 0
1
2
3
...


Reservee
Dashboard
Chats
History
Users
Settings
Log Out
HISTORY
Search everything
* You can search/view reservation(s) by typing one or more of these information
Branch:
AN BBQ Su Van Hanh
From:
Mon 23/12/2020
To:
Mon 23/12/2020
VIEW
All
Pending
Canceled
Completed
Wait for Payment
Date
Timeslots
People
Status
Full name
Phone number
Email
02.03.2021
9h00-9h30
2
Completed
Nguyen Van A
0903929119
nguyenvana@gmail.com
Total: 3
Choose: 0
1
2
3
...

3 Architecture

3.1 Layered Architecture diagram



Description
The layered diagram encompasses the UI/UX design, which is a web-based interface. The authentication layer allows users to log in using their Google account, social media credentials, or phone number. The service layer consists of three services: Reserve Table, View Restaurant List, and View Reservation History. The reservation history service interacts with the database.

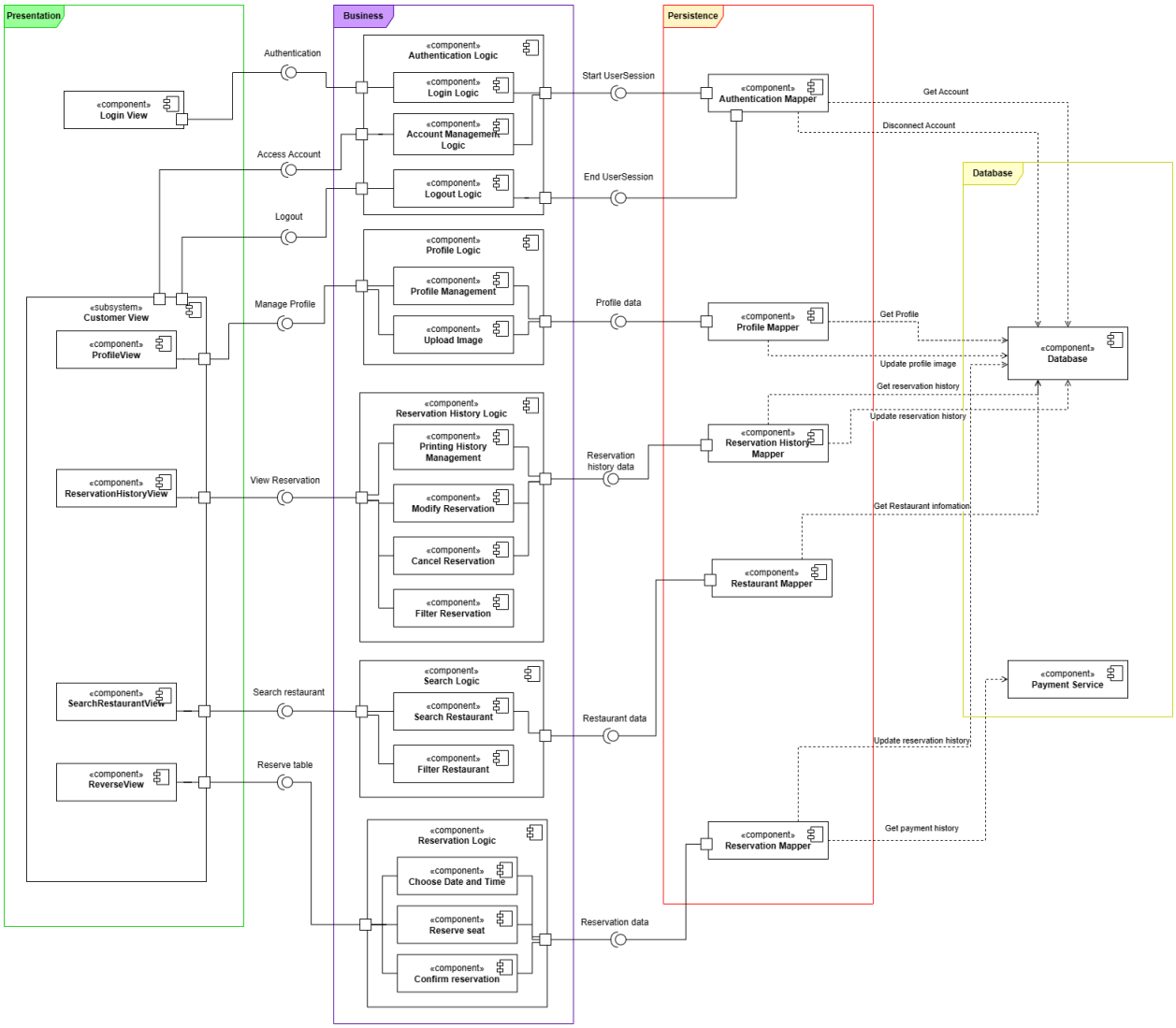
3.2 Presentation

Upon landing on the website, users are prompted to log in using their Google account, social media credentials, or phone number. If they have forgotten their password, a "Forgot password" option is available for retrieval. Alternatively, users can opt for the "Remember me" feature to retain their login information for faster access in the future. Once logged in, the main page presents three primary components for user interaction:

- **Navigation bar:** There will be 5 buttons to redirect the user to their chosen page: Home, Reminder, History, Account and Log out.
- **Home:** In homepage, there will be a list of restaurant for user to choose, there will also exist a search bar that user can search for their preferred restaurant. In the restaurant display section, user can see the restaurant's name, a small description, total of table that the restaurant have and number of table remaining.
- **Reservation:** Reservation page is where the main action will take place. In this page, a table will appear which indicate the restaurant reservation information, there will be a time display on top of the table, non-white cell mean the table already occupied. In the occupied cell, name and number of people will be displayed. To create a reservation, user need to click create, a pop up will appear, user will only need to enter date, time and note for their table, other information have already been provided to the system via login, so they are auto filled. After that user can either click "Cancel" to cancel or "Confirm" to reserve the table.

- **History:** On the history page, users can view their past and current reservations, including their status. Users can also filter their reservations by status and edit their existing reservations.

3.3 Component diagram



Description

The diagram depicts the interactions between different components in the Reserving service module (Users' side). The system revolves around 4 layers that are described in layered architecture part.

Presentation layer:

- **Login View:** Render the Account authentication interface from component Account Logic (internal component Authentication)
- **Customer View:**
  Profile View: Render profile information obtained from Profile logic component and can be used for uploading profile image.
  Reservation History View: Render printing log information obtained from Reservation History Logic component and can modify, cancel, and filter reservation process on waiting.
  Search Restaurant View: Render payment log information obtained from Search Logic component as well as a button to search and filter restaurant.

Reverse View: Render printing window for printing process through Reservation Logic component.

Business layer:
Consists of 5 major components representing 5 modules of the Business layer we have discussed earlier in layered architecture.

- **Authentication Logic:** Login, logout, retrieve account session.
- **Profile Logic:** View profile and upload image.
- **Reservation History Logic:** Modify, cancel, and filter reservation.
- **Search Logic:** Search and filter restaurant.
- **Reservation Logic:** Functions for customers to choose date and time, reserve seat, and confirm reservation.

Persistence layer:
There are 5 data mapper:

- **Authentication Mapper:** Mapping authentication service from Authentication Logic to database.
- **Profile Mapper:** Mapping profile information from Profile Logic to database.
- **Reservation History Mapper:** Mapping reservation history log information from Reservation History Logic to database.
- **Restaurant Mapper:** Mapping search information from Search Logic to database.
- **Reservation Mapper:** Mapping payment data from Reservation Logic to Payment Service and updating reservation history to database.

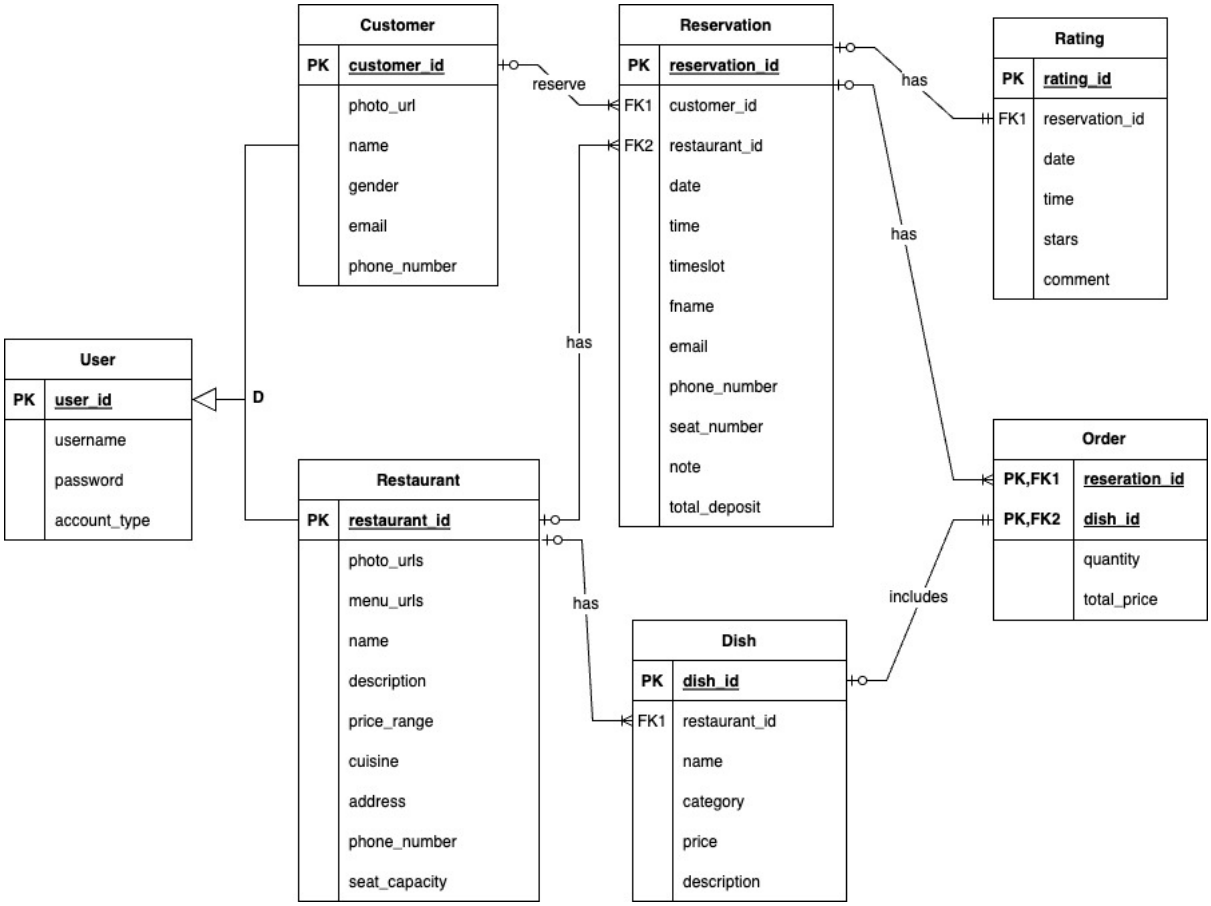
Database layer:
Consists of a database module providing data to each data mapper in the Persistence layer. There is also the Payment service to retrieve data about payment logs.

3.4 Entity Relationship Diagram

The following figure illustrates the Entity Relationship Diagram (ERD) that conceptually captures the data requirements of the restaurant reservation application. Overall, it has 7 entities including: *User*, *Customer*, *Restaurant*, *Reservation*, *Rating*, *Dish*, *Order*.

Specifically, we model *Restaurant* to have at least one dish to offer for customer. In addition, one client can reserve many places in a restaurant and leave corresponding rating for each reservation they made. In order to streamline the payment process, entity *Order* contains information about the reservation and dishes used at the meal and characterizes a total price as well.

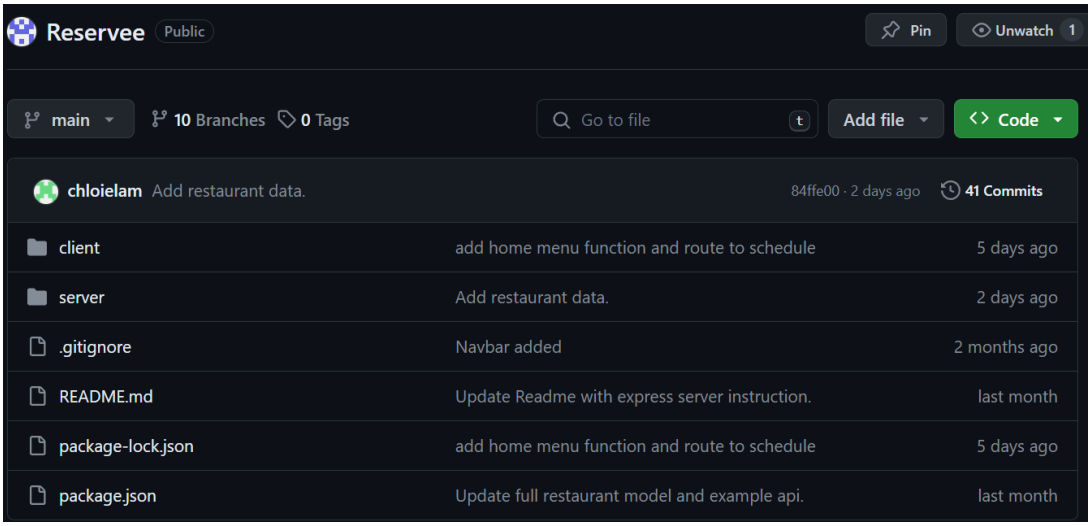
To account for scalability when the number of stakeholders increases in the future, we dedicate a base entity called *User* to keep the authentication of them while using a foreign key *user_id* to keep track of their specific information.



4 Implementation 1

4.1 Github repository

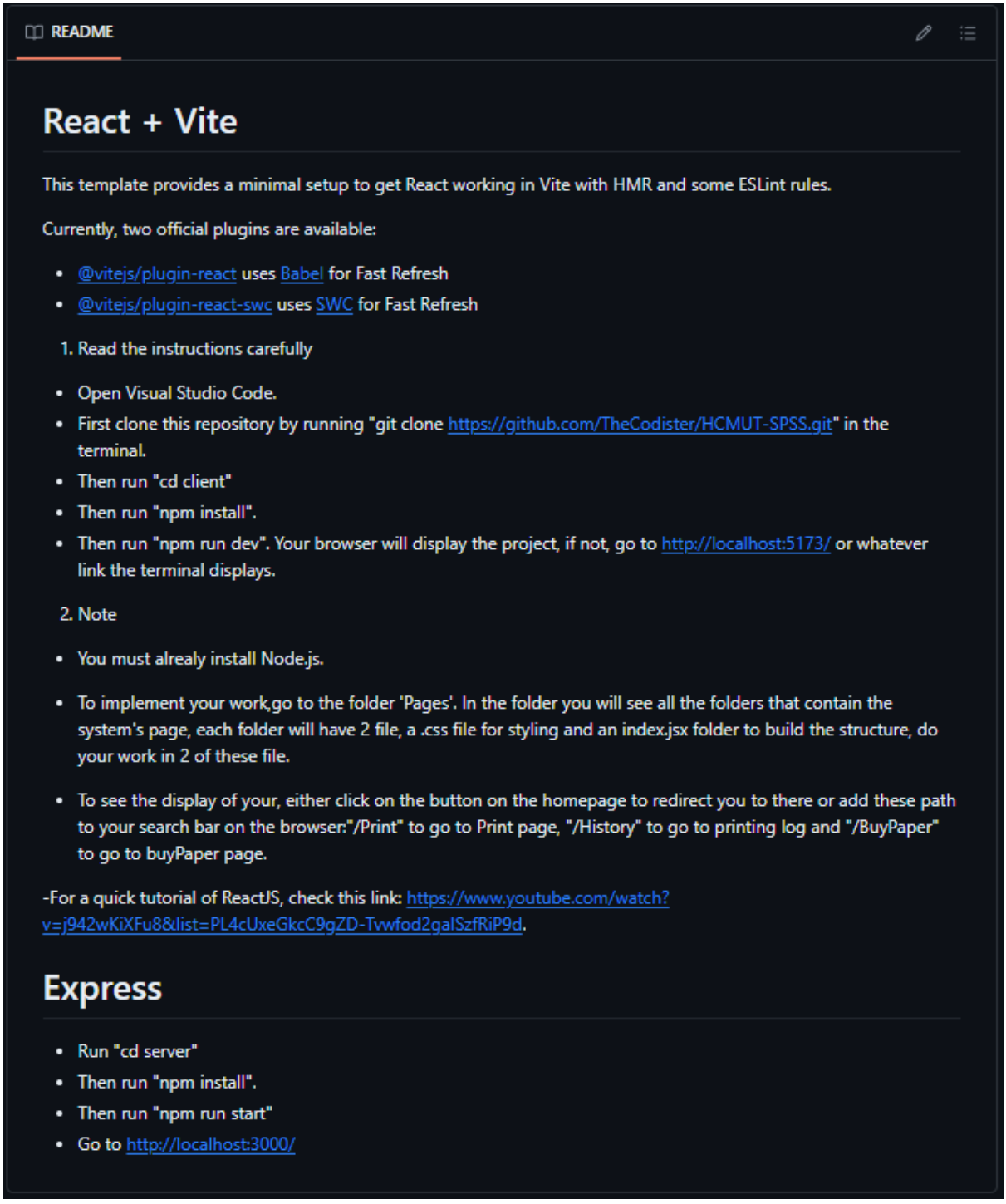
To facilitate the development of Reservee, our team has crafted a GitHub repository encompassing completed design and architectural files, along with future code implementations. Here is the link to our Reservee app's GitHub repository: https://github.com/TheCodister/Reservee.git



In our repository, we divided into 2 folders, client for Frontend and server for Backend.

4.2 Adding material

4.2.1 Instruction/Documentation



README

React + Vite

This template provides a minimal setup to get React working in Vite with HMR and some ESLint rules.

Currently, two official plugins are available:

- @vitejs/plugin-react uses Babel for Fast Refresh
- @vitejs/plugin-react-swc uses SWC for Fast Refresh

1. Read the instructions carefully

- Open Visual Studio Code.
- First clone this repository by running "git clone https://github.com/TheCodister/HCMUT-SPSS.git" in the terminal.
- Then run "cd client"
- Then run "npm install".
- Then run "npm run dev". Your browser will display the project, if not, go to http://localhost:5173/ or whatever link the terminal displays.

2. Note

- You must alreay install Node.js.
- To implement your work,go to the folder 'Pages'. In the folder you will see all the folders that contain the system's page, each folder will have 2 file, a .css file for styling and an index.jsx folder to build the structure, do your work in 2 of these file.
- To see the display of your, either click on the button on the homepage to redirect you to there or add these path to your search bar on the browser:"/Print" to go to Print page, "/History" to go to printing log and "/BuyPaper" to go to buyPaper page.

-For a quick tutorial of ReactJS, check this link: https://www.youtube.com/watch?v=j942wKiXFu8&list=PL4cUxeGkcC9gZD-Tvwfod2gaISzfRiP9d.

Express

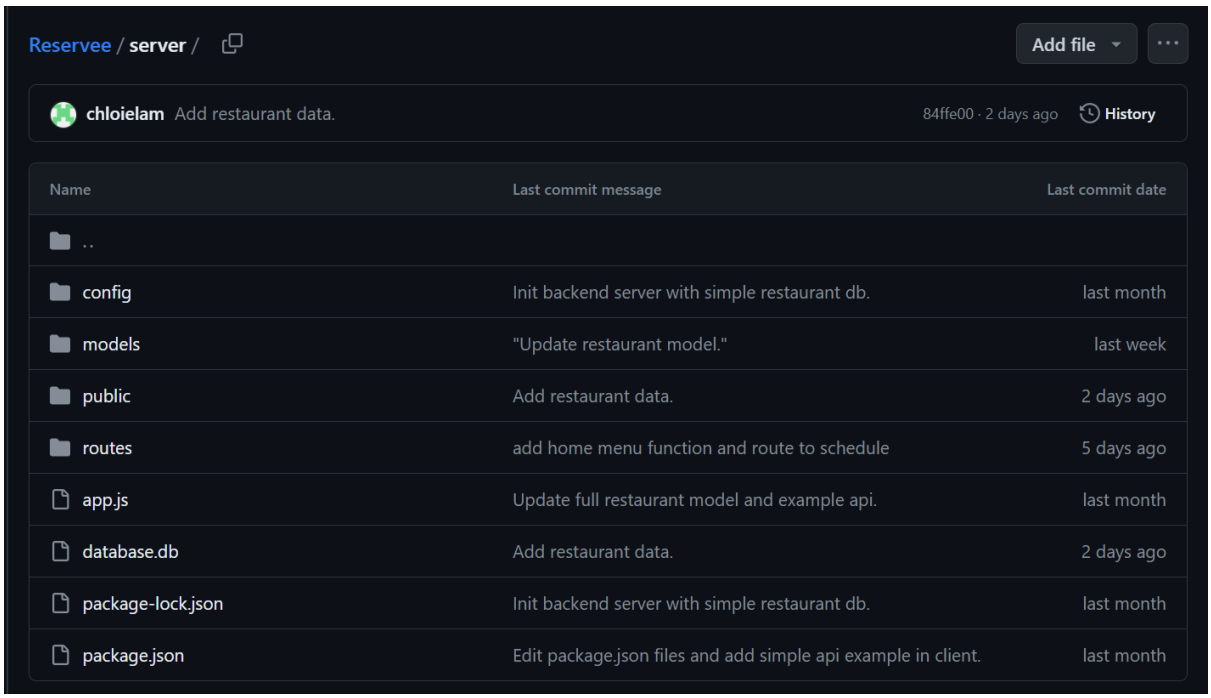
- Run "cd server"
- Then run "npm install".
- Then run "npm run start"
- Go to http://localhost:3000/

We've documented comprehensive instructions for running both the frontend, built with ReactJS, and the backend, using ExpressJS, in our project.

4.2.2 Frontend

In the client folder, we store all source code for the frontend, since we using ReactJS + Vite, Github don't have to store dependencies(node_modules files), members of our group can reinstall the dependencies in local after they pull their code.

4.2.3 Backend



In the server folder, we store the source code for the back-end, we also don't need to store the node_modules file here. The access to the server services is provided through route handling by *Ex-*

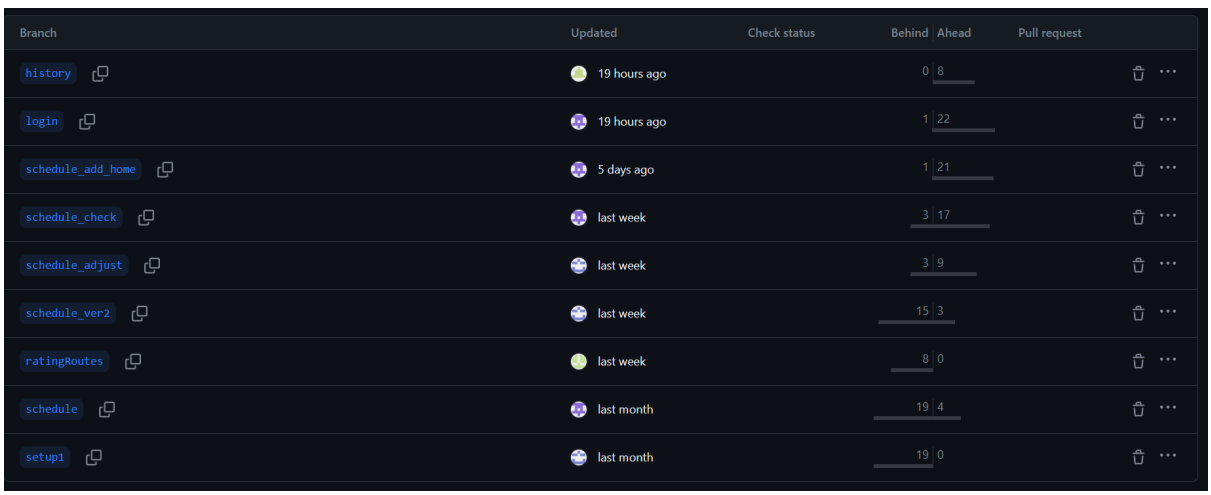
press.js, a popular web application framework for *Node.js*. This framework simplifies the process of defining routes and handling HTTP requests and responses. Each route is associated with a specific path and HTTP method, such as GET or POST, and is responsible for executing specific server-side operations. These operations range from fetching data from a database, processing user inputs, to sending back dynamic content to the client. The modular nature of *Express.js* allows for a clean and maintainable code structure, enabling easy expansion and management of the server's capabilities. This approach ensures that the server can efficiently handle a variety of tasks, from user authentication to data management, while maintaining a robust and scalable architecture. For the database storage, we use SQLite3 as the database management system for easy access and query execution. Our data is stored in *database.db* file.

More specifically, there are total 4 routes that our server provides:

- *Restaurant route:* The key functionalities provided by these routes include *Fetching All Restaurants*, *Adding a New Restaurant*, *Getting Reservation Information by Restaurant ID*, *Searching for Restaurants*.
- *Customer route:* several RESTful endpoints are defined to handle customer data operations in this route, including *Creating a New Customer*, *Retrieving All Customers*, *Retrieving a Single Customer by ID*, *Retrieving Customers by Name*, *Updating a Customer's Information*.
- *Reservation route:* Utilizing Express.js and a database model, it offers a comprehensive set of functionalities for reservations: *Fetching All Reservations*, *Adding a New Reservation*, *Fetching Reservation by Reservation ID*, *Fetching Reservations by Customer ID*, *Fetching Reservations by Restaurant ID*, *Deleting a Reservation*, *Updating Reservation Information*, *Updating Reservation Date, Time, and Seats*
- *Rating route:* The key functionalities of this route include: *Fetching User-Specific Ratings by User ID*, *Fetching Restaurant-Specific Ratings by Restaurant ID*, *Creating a New Rating*, *Updating a Rating*.

The comprehensive list of RESTFUL API endpoints enables frontend team to make the website fully functional at a basic level, i.e., supporting Create - Read - Update - Delete (CRUD) operations. Given the current project structure at the backend side, it is well scalable in case of more demanding user requirements in the future.

4.2.4 Branches



We possess a total of nine branches (excluding the main branch), each dedicated to testing specific aspects of the project individually.

5 Implementation 2

5.1 Implementation overview

1. **Methodology and Project Scope:**
   The project adopts a Waterfall methodology, chosen due to the well-defined nature of the Minimum Viable Product (MVP). The MVP encompasses a small-scale web application, with all necessary requirements predetermined. This approach facilitates a structured and sequential development process, aligning with the project's clear objectives and timeline.

2. **Technological Stack:**
   The web application is developed using ExpressJS for the backend and ReactJS for the frontend. The MVP currently utilizes hard-coded example databases stored in the data folder, as a simplified representation of future backend functionalities. The project strictly follows the Model-View-Controller (MVC) architectural pattern, ensuring modularity and maintainability.

3. **Project Structure:**
   The project features a well-organized and categorized folder structure, contributing to the maintainability and scalability of the codebase. Each module and component is logically placed within its designated folder, promoting code clarity and ease of navigation.

4. **User Interface Design and Implementation:**
   The User Interface (UI) components have been meticulously designed and integrated into each corresponding view. Following best practices, the UI design is consistent, user-friendly, and aligns with the overall aesthetic vision of the application.

5. **Backend Development and MVP Simplification:**
   Our backend development primarily relies on ExpressJS, and we've integrated SQLite as our chosen database system. This strategic decision to implement SQLite contributes significantly to our project by ensuring a swift and seamless experience for our users, owing to its capacity for providing efficient and rapid data processing.

6. **MVC Pattern Adherence:**
   The application strictly adheres to the Model-View-Controller (MVC) architectural pattern, separating concerns and promoting code maintainability. This modular structure facilitates future enhancements and modifications to different components without affecting the entire system.

7. **Sprint 2 Objectives - Implementation and Wrap-up:**
   Sprint 2 focuses on the continued implementation of additional features and the refinement of existing functionalities. The wrap-up phase aims to conduct thorough testing, address any identified issues, and prepare the project for the next stage of development.

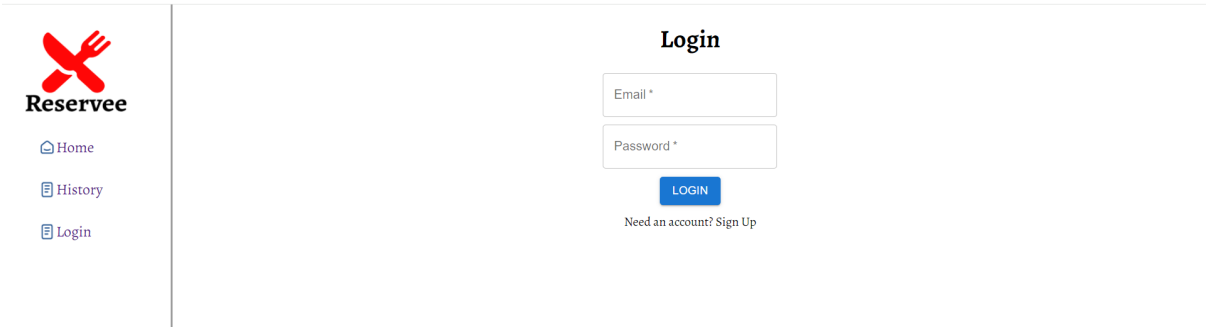
8. **Next Steps:**
   Following the completion of Sprint 2, the project will undergo further testing and refinement before moving into subsequent phases. Future iterations will include the integration of new feature for the web like order food, cancel reservation and many more.

5.2 Reservee Web

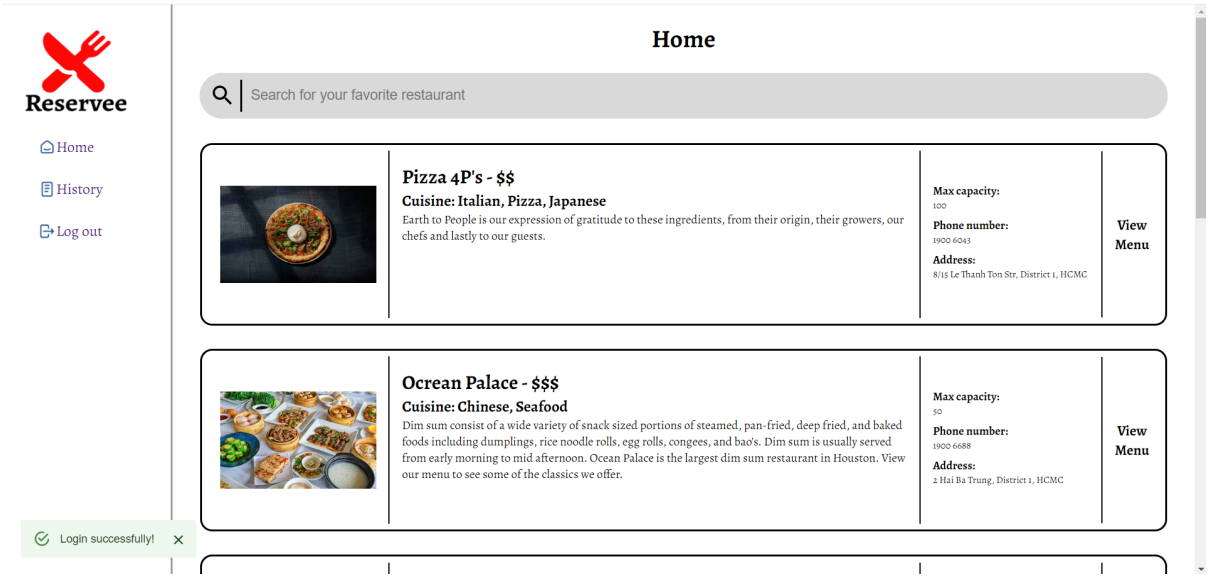
5.2.1 Login

Users can securely access their accounts by providing their email and password, ensuring a personalized and protected experience on the platform. Access to certain functionalities, such as table reservations

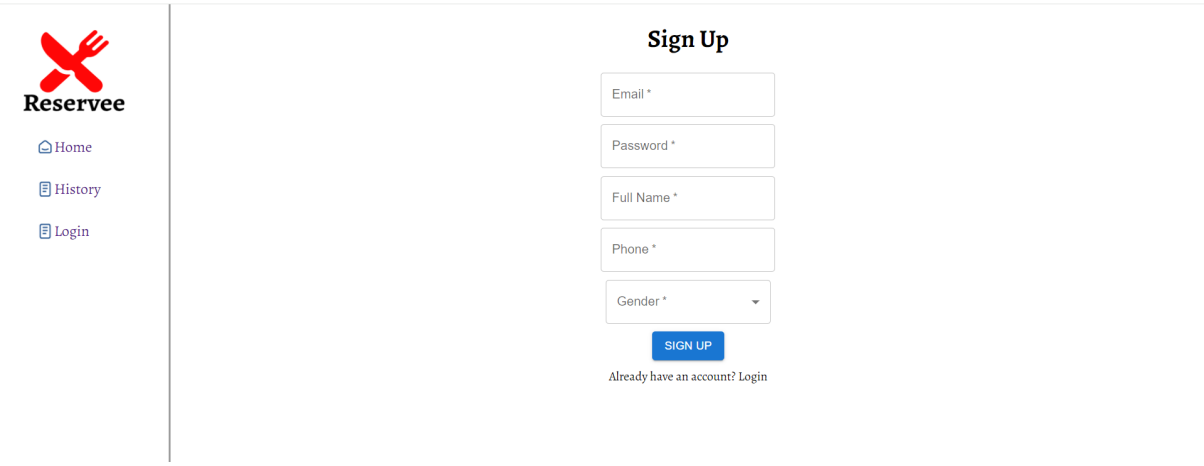
and viewing transaction history, is contingent upon successful login, thereby safeguarding user data and ensuring a tailored experience.



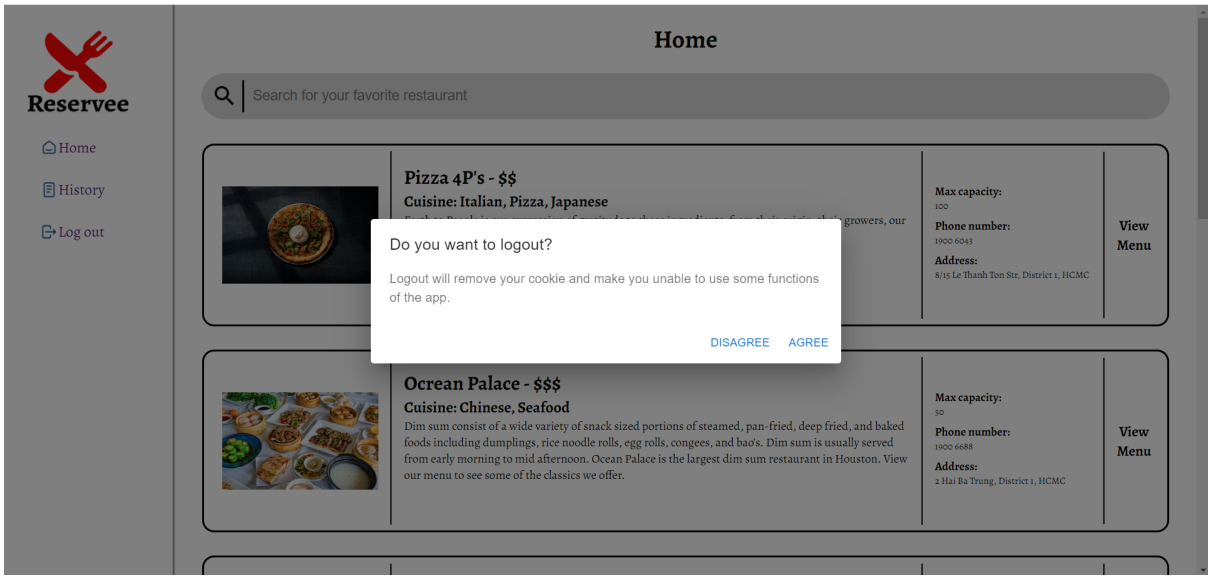
After the user has logged in successfully, they will be redirected to the home page along with a message showing login successfully.



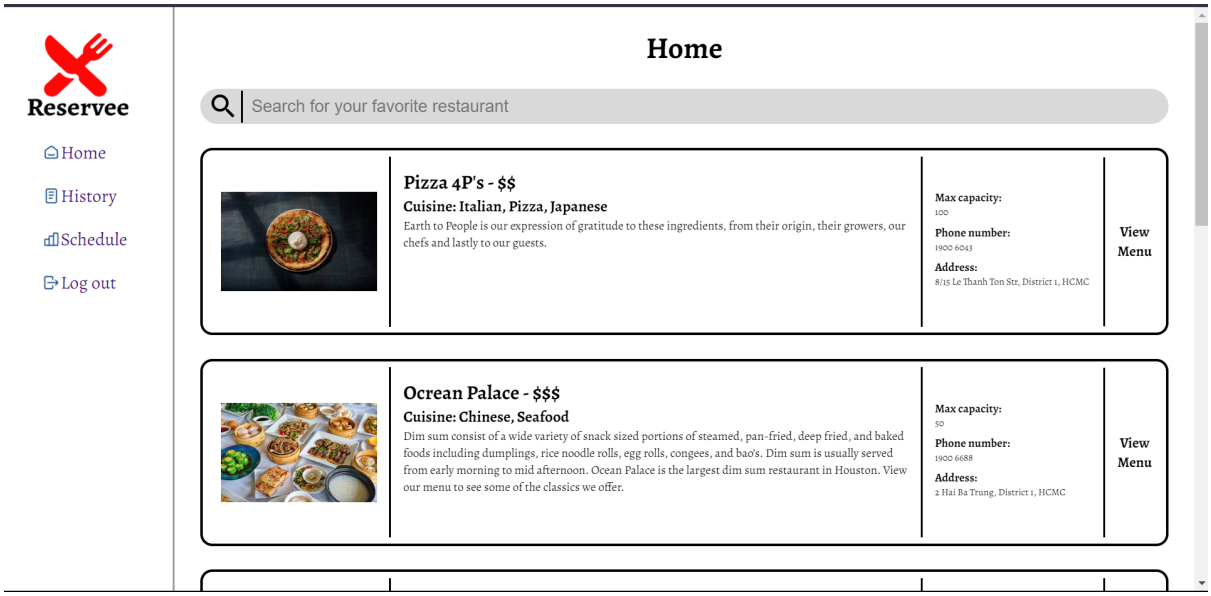
For those who do not possess an account, an option to "Sign up" is provided below the login form. The signup process involves the submission of essential information, including Email, Password, Full Name, Phone number, and Gender. Notably, the system ensures that each Email is unique, preserving the integrity of user accounts.



To conclude a session, users can log out by selecting the "Log out" button, ensuring a secure and straightforward logout process.

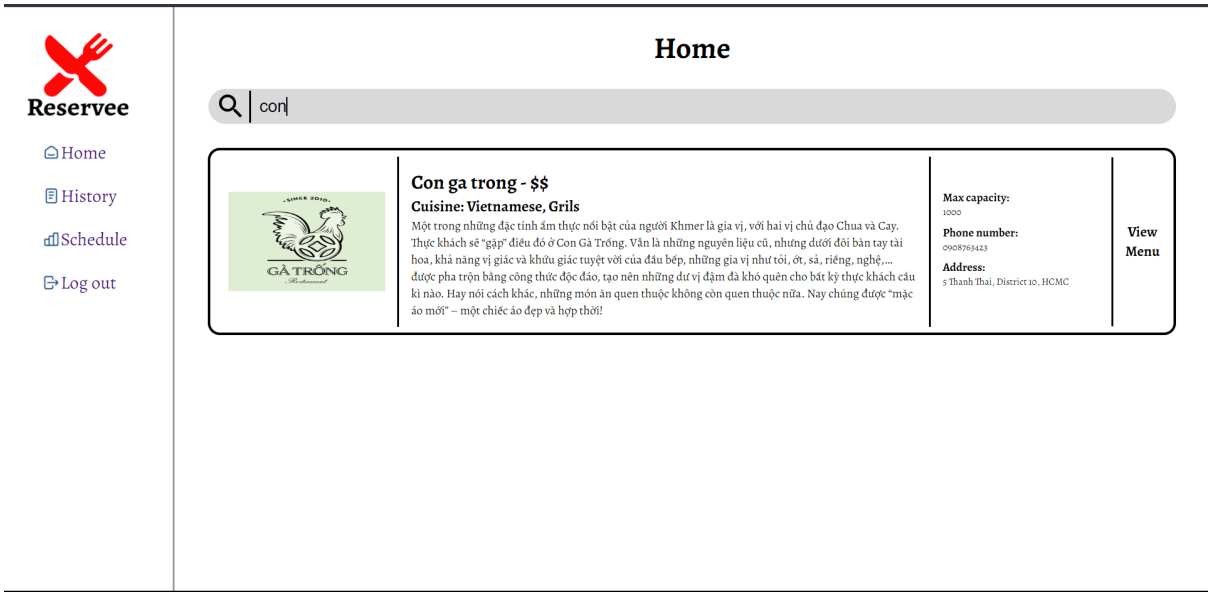


5.2.2 Mainpage



Our page consists of two primary sections: the navigation bar and the content area. The navigation bar features four buttons-Home, History, Schedule, and Log Out-providing easy access to various sections. The main content section showcases recommended restaurants, allowing users to search for their preferred dining establishment using the search bar. Within each restaurant's profile, users can access essential details such as the maximum capacity, phone number, and address. Additionally, clicking on the "View Menu" button enables users to explore the restaurant's menu offerings.

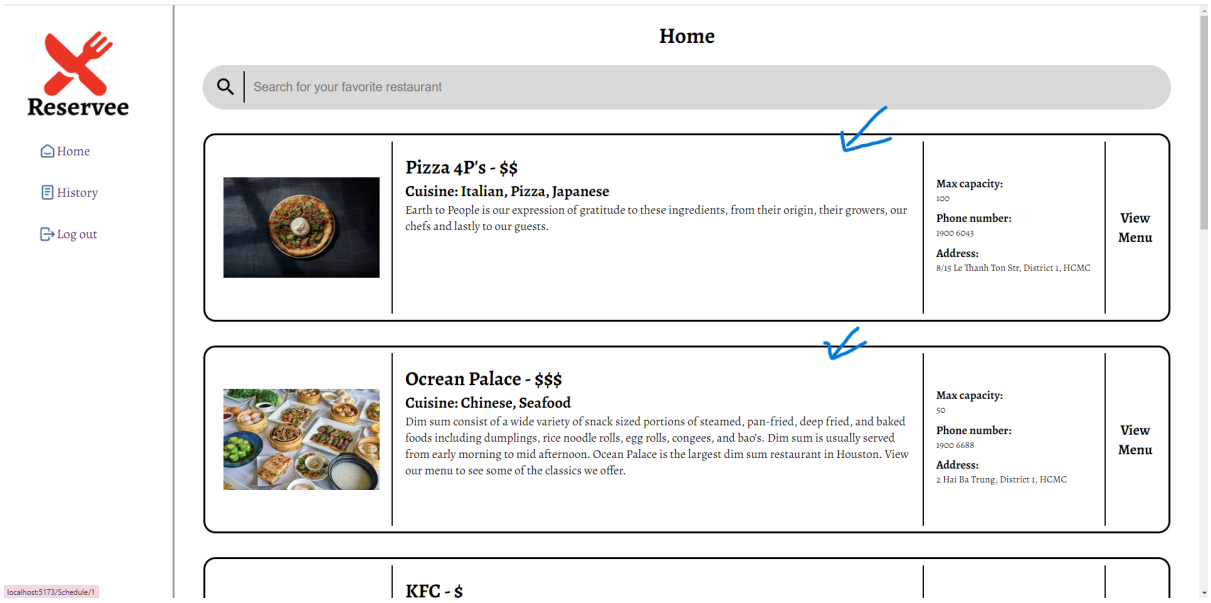
User can search for their preferred restaurant by typing their restaurant's name into the search bar.



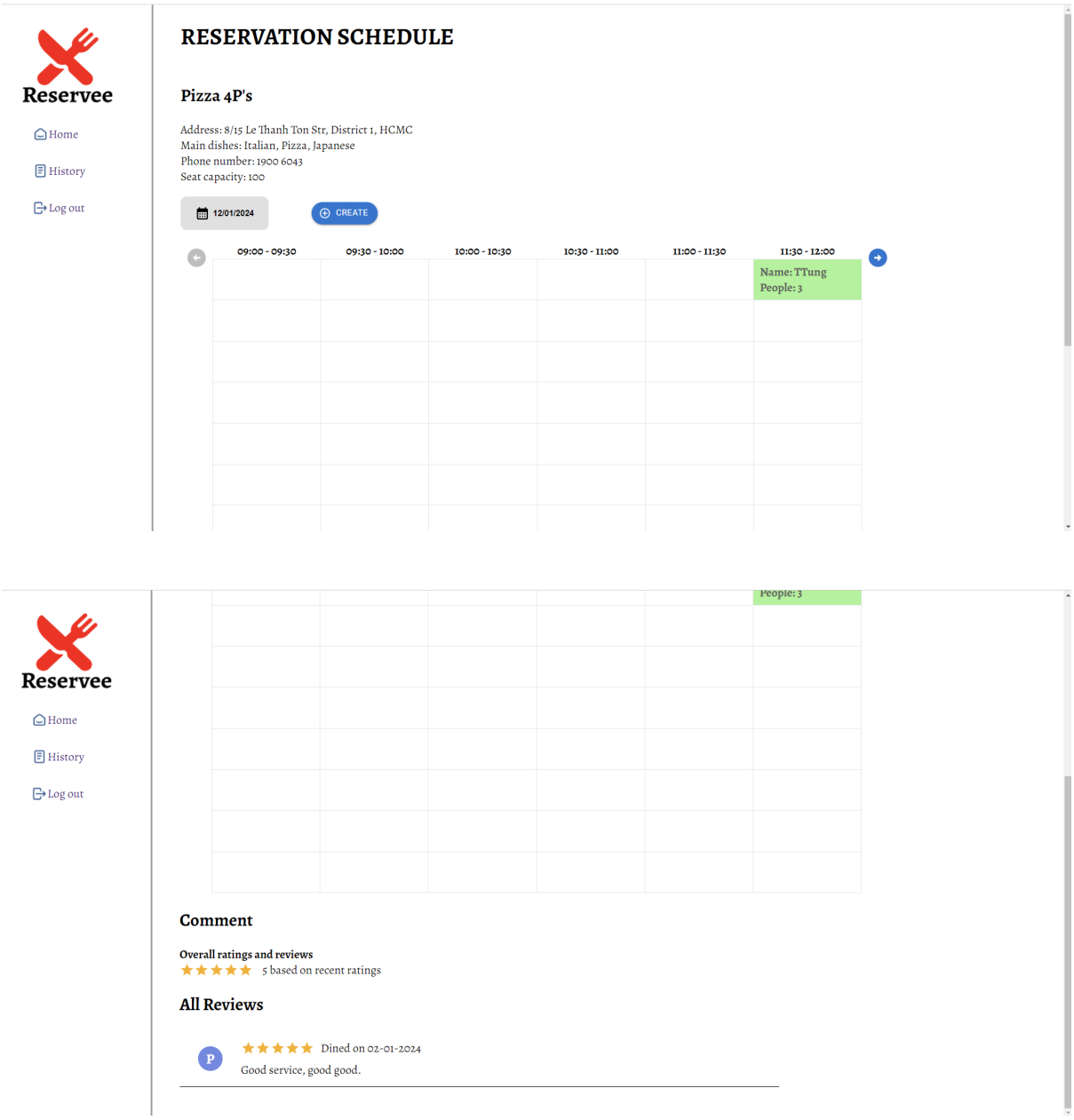
If users click on the Restaurant's profile, it will navigate the them to the table's schedule of that restaurant.

5.2.3 Schedule

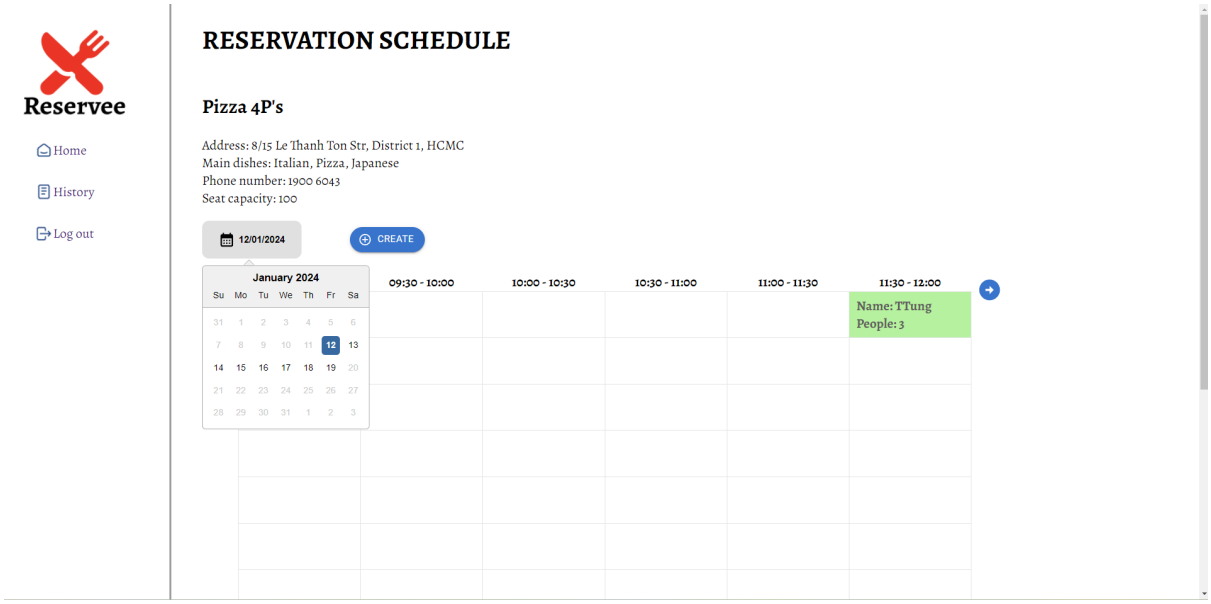
Users can click on one of the restaurant to access the schedule page to reserve tables for that restaurant.



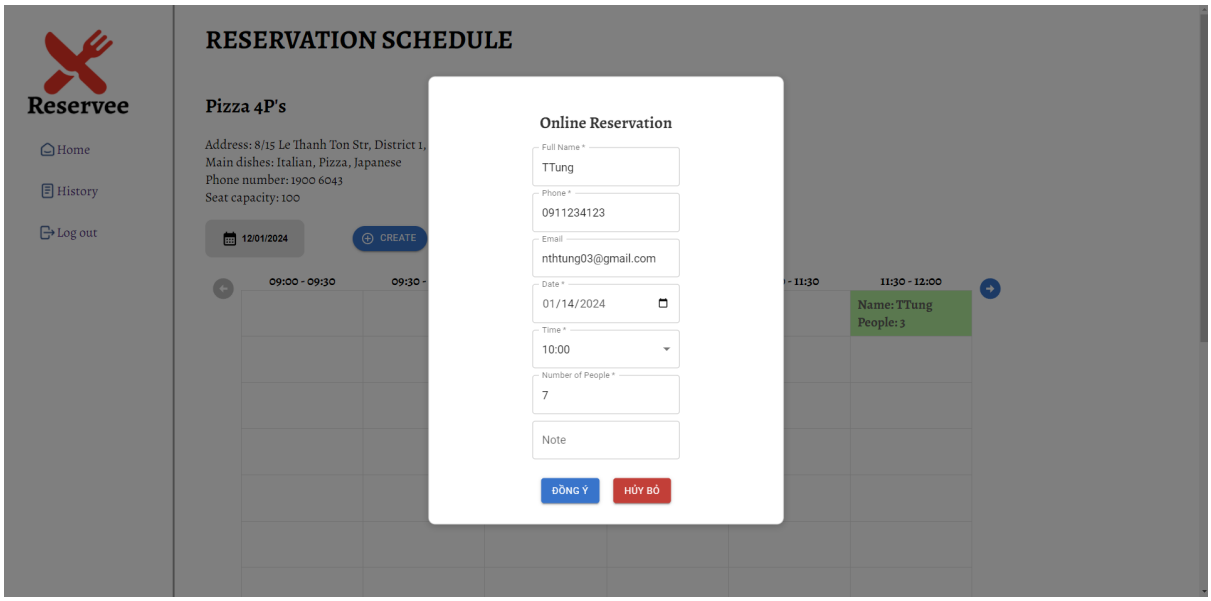
Below is the UI of Schedule Page. It includes restaurant's information, a reserve table to show the already reserve of users and other customers. The Calendar button (gray one) is used to change the displayed date of the reserve table. The "Create" button is used to open a form modal to make new reservation. There is also a section to display review and rating of the restaurant.



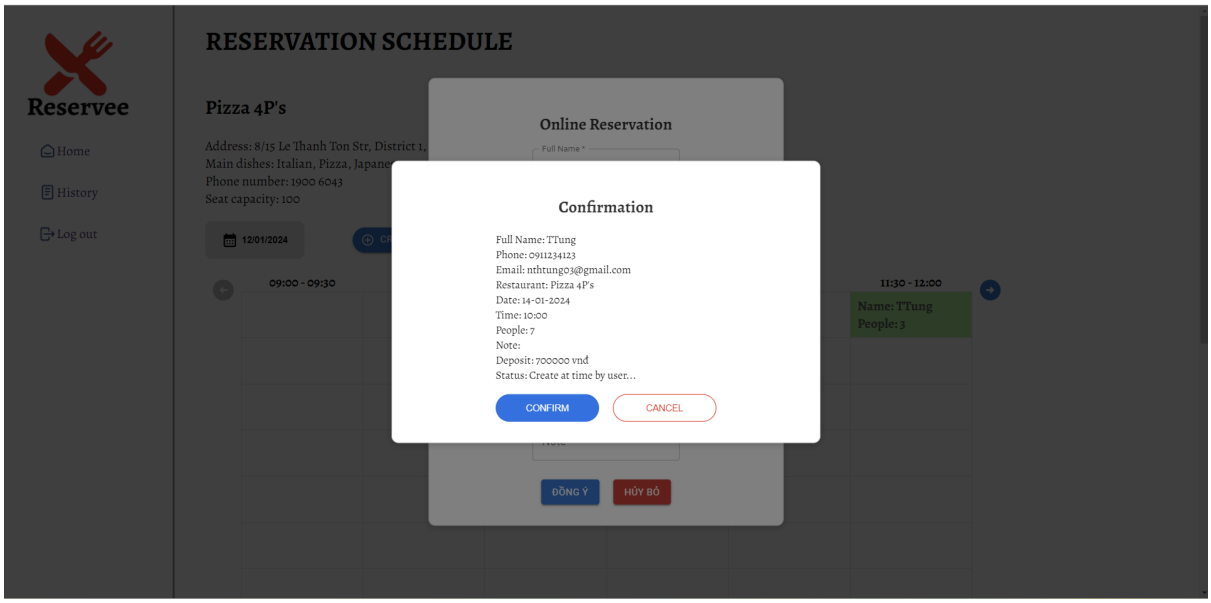
On clicking the Calendar button, a calendar will be displayed to choose the date for display. Users can only choose days from current date to 7 days onward.



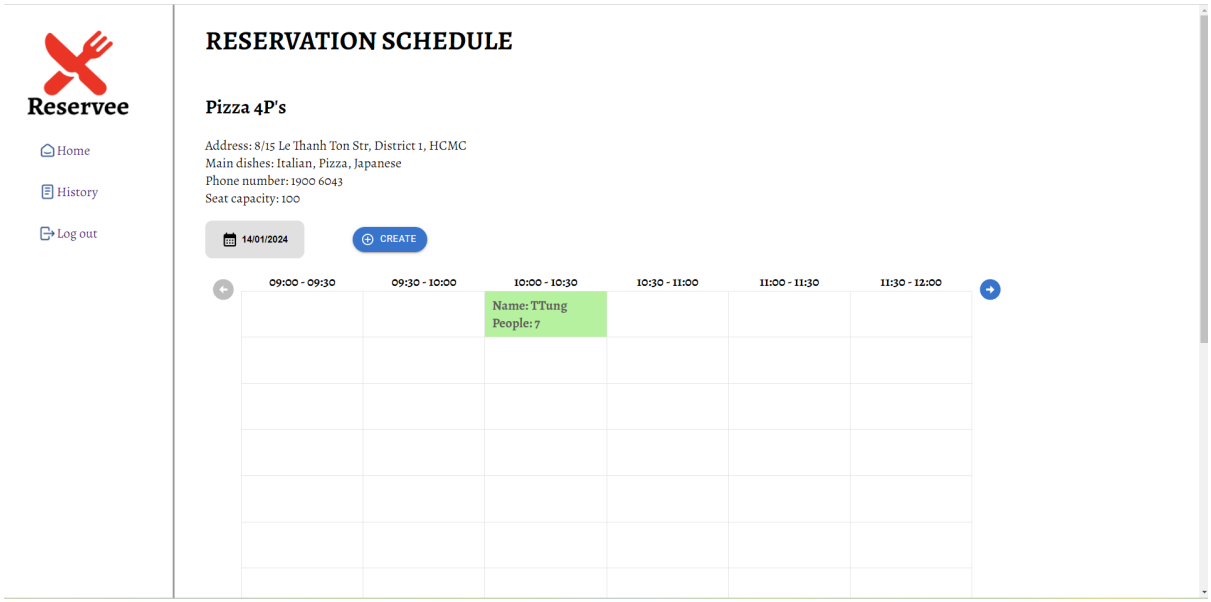
On clicking the "Create" button, if the user is not login, navigate user to login page. If the user is login, a form modal for reserving table is displayed. Values of "Full name", "Phone", "Email" will be customer's information on default but can be changed if user wants. User can reserve table from current date to 7 days onward, choose available time, and number of seat if it does not exceed current time's capacity. User can also note for the restaurant, but it is optional.



On clicking "Confirm" button, the confirm modal is displayed.

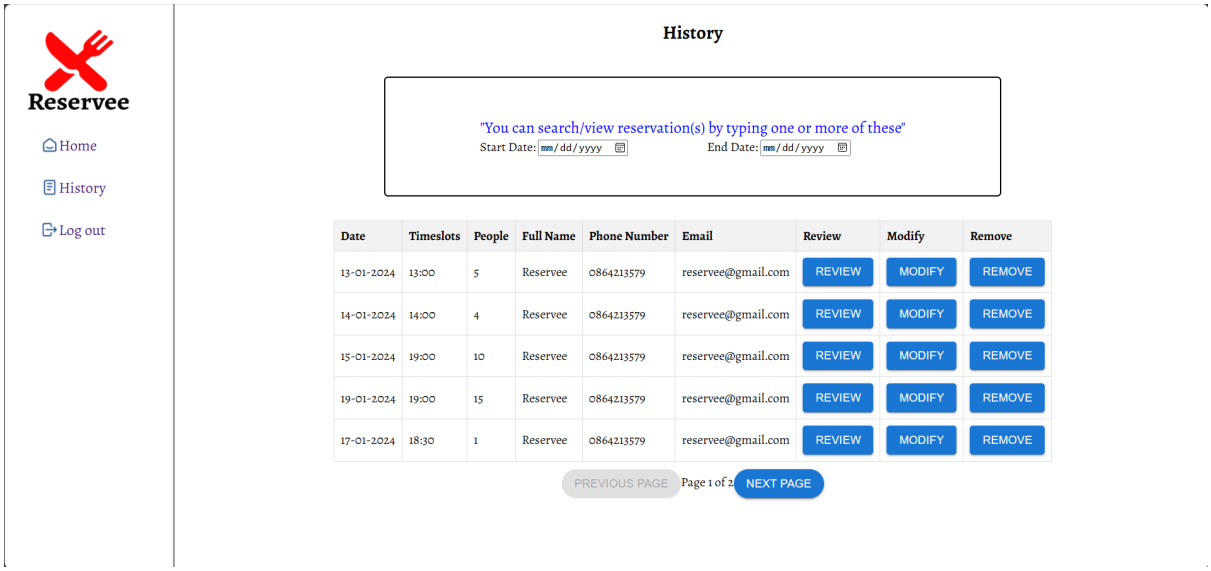


On clicking "Confirm" button on Confirm modal, the data is stored at database and is displayed on reserve table.

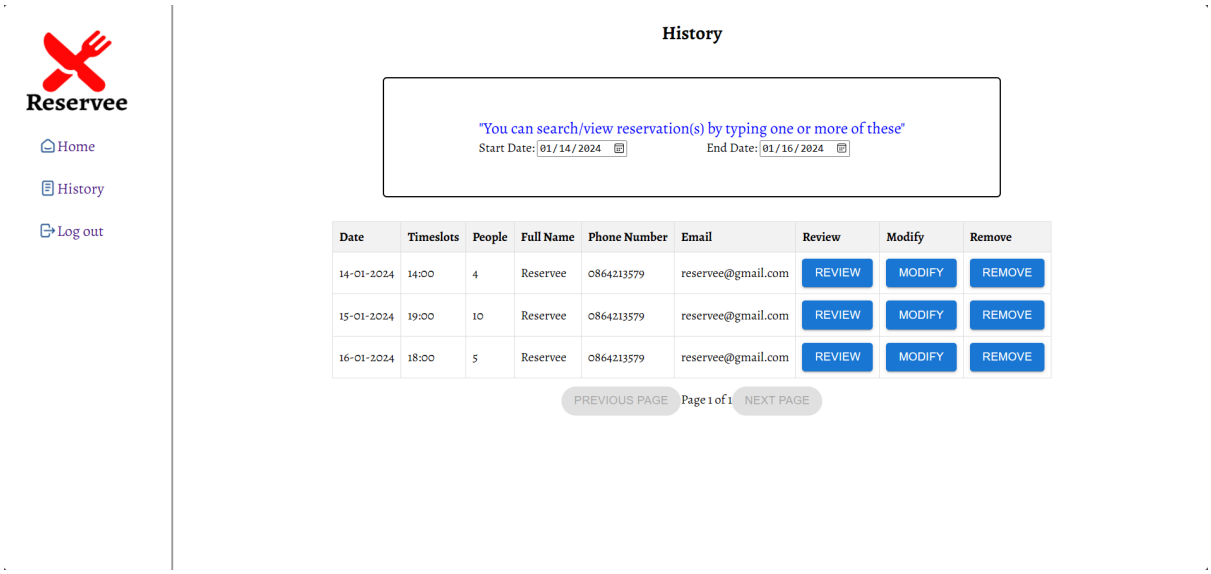


5.2.4 History

Below is the UI of History Page. It shows the user's record of reservations. With each reservation, user can add a review, modify the reservation, or delete one.

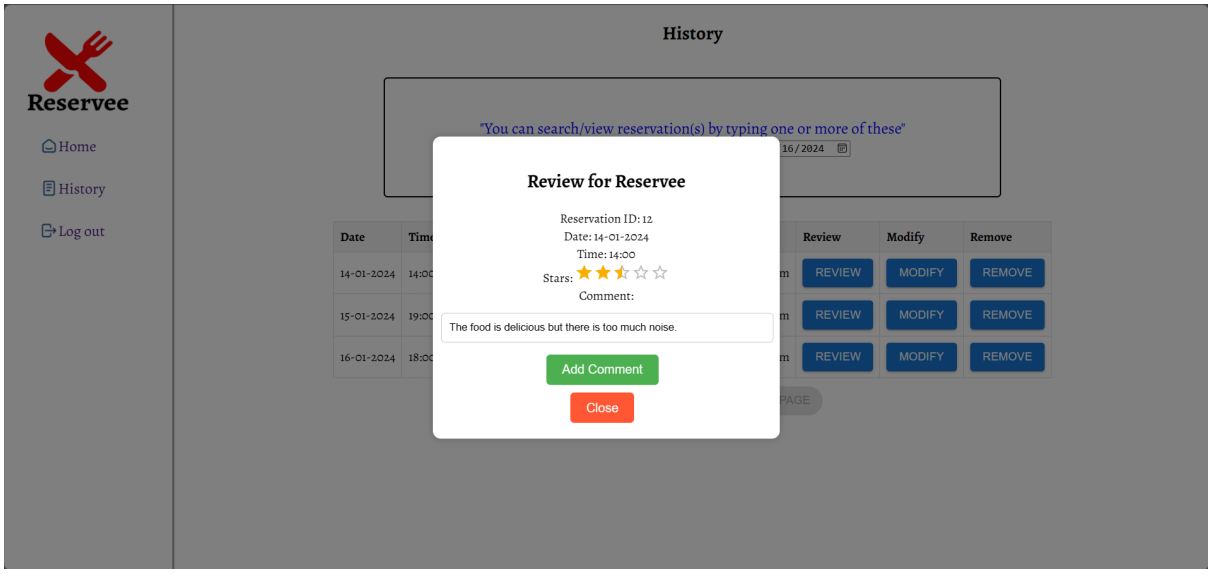


Additionally, users have the convenience of easily filtering reservations according to either the start or end date, providing a more streamlined and tailored booking experience.

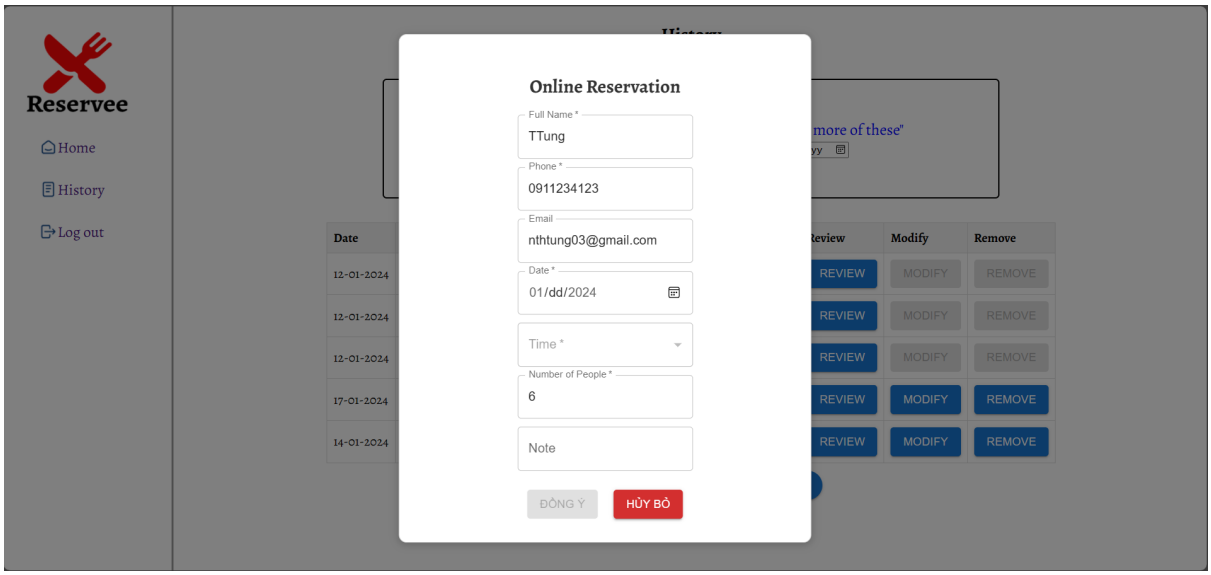


Each reservation entry provides key details, and users can click one of the three buttons to interact with the reservation:

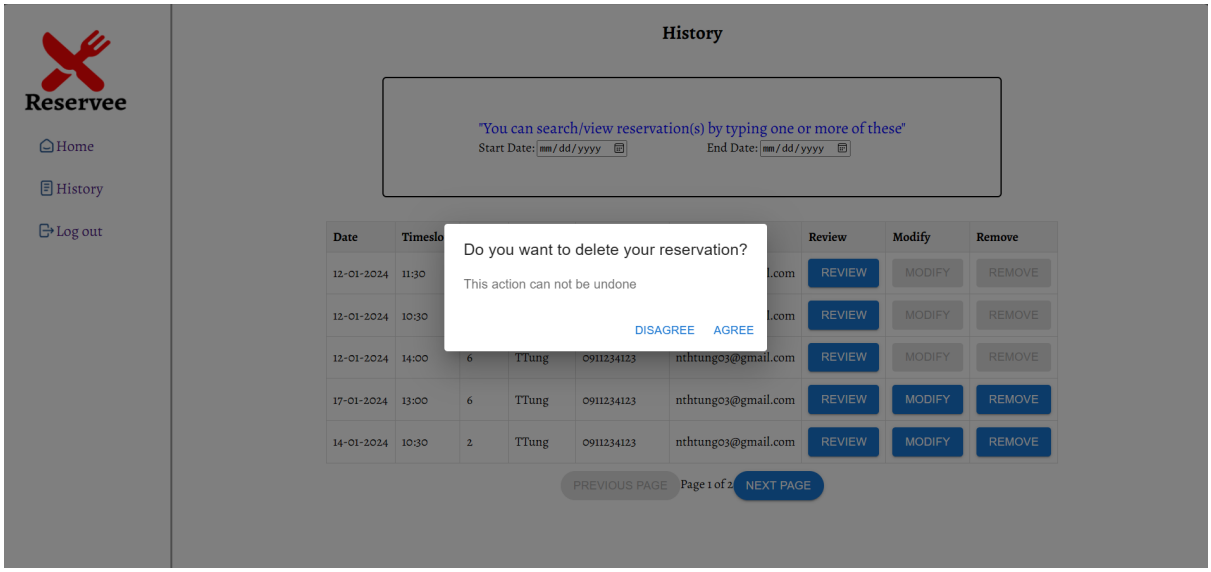
- *Add review:* The "Add Review" option allows users to share their feedback and experiences by rating using stars and comments. Also, this review will be displayed in the corresponding restaurant.



- *Modify reservation:* The "Modify Reservation" option empowers users to make adjustments to their booking details, ensuring flexibility and personalized control over their reservations.



- *Delete reservation:* The "Delete Reservation" option enables users to remove a reservation, offering a straightforward way to manage their bookings.



6 Conclusion

In the course of this Intergration Project, we embarked on a comprehensive journey encompassing various phases of the software development life cycle. Beginning with requirement elicitation, where we employed use case diagrams to capture and understand user interactions, we progressed through system modeling using sequence and activity diagrams, architecture design with component diagrams, and a rigorous usability test to ensure a user-friendly experience.

As we transitioned to the implementation phase using ReactJS and ExpressJS, collaboration became key, and GitHub served as the linchpin for version control and collaborative development.

1. **Project Evaluation:**
    - In an overall assessment, our team successfully completed all the required tasks with due consideration. However, certain challenges highlighted areas for improvement, particularly in communication and collaborative decision-making. The diagram creation process suffered from a lack of collective discussion, leading to errors that could have been mitigated through shared insights. Additionally, ambiguity arose in the architecture design phase, exposing a gap in our understanding of how detailed or complex the models should be.
2. **Challenges to Solutions:**
    - Communication Gaps: Lack of comprehensive discussion, particularly during diagram creation, led to errors.
    - Ambiguity in Architecture Design: Uncertainty regarding the depth and complexity required for the architecture design.
    - Limited Research and References: Insufficient references in the report, reflecting a gap in the depth of research.
3. **Lessons Learned:**
    - Communication is Key: Effective communication is crucial for collaborative success. Regular team discussions, especially during critical phases like diagram creation, enhance the quality of work and minimize errors.

- Clarity in Decision-Making: Establish clear criteria for decision-making, especially in ambiguous areas like architecture design. Define standards for depth and complexity, ensuring alignment with project goals.
- Research Strategies: Develop effective research strategies to enhance the depth of the assignment. Encourage team members to explore diverse sources, both internal and external, to enrich the content and provide a well-rounded perspective.
- Documentation and References: Emphasize the importance of comprehensive documentation. Every decision, design choice, and solution should be well-documented, and references should be cited to ensure credibility and provide additional context.

Moving forward, we acknowledge the iterative nature of software development and the importance of continuous improvement. By embracing the lessons learned from this assignment, we are better equipped to navigate future projects with confidence and deliver high-quality software solutions.