Enterprise Reservation Platform

Pawaris Lertchaiprasert 60090027

Traphume Methasurarak 60090043

Bachelor of Engineering in Software Engineering,
Department of Computer Engineering,
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Academic Year 2020
ISBN

Thesis - Academic Year 2020

Bachelor of Engineering in Software Engineering

Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

Title : Enterprise Reservation Platform

Authors :

1. Pawaris Lertchaiprasert 60090027
2. Traphume Methasurarak 60090043

Approved for Submission

…….

(Dr. Pipat Sookavatana)
Advisor

Date …...../……./…….

Abstract

The aim of this software engineering project is to create a platform where people can list their businesses (bar or restaurant) for an online reservation. Such that businesses can easily define their seating placement, create a time slot for them to be booked online by their customers through the mobile application, and validate the generated electronic tickets of the reservations provided to the customers.

The system consists of four main points of interactions: the customer's mobile application, the business web application, the platform admin web application, and employee on-site mobile application. The customer's mobile application will allow seats to be selected and reserved, generated electronic tickets to be viewed and discovery of businesses through a search functionality. The business web application will give the businesses the ability to create descriptions of their business, display pictures, menu items available, seating placements, and available booking time slots. The platform admin web application is for managing the businesses themselves, this refers to approving businesses who have registered on the platform and approving requests to update the business information.The on-site is to be placed physically at the business to validate customers who have reserved seats through the mobile application.

The mobile clients are developed using Dart and the Flutter framework. The business and admin web application is developed with Next.js framework which is built on top of React library. The backend service uses Go as the main programming language. We take advantage of metaprogramming to generate boilerplate server interfaces, request and response objects, and client stubs for mobile applications.

Table of contents

1. Objectives and Problem Description

1.1) Problem description 1
1.2) Objectives 1
1.3) Scope of Work 2
1.4) Thesis structure 2

2. Related works

2.1) SF Cinema 4
2.2) QueQ 6
2.3) Floorplanner 6

3. Background knowledge

3.1) MongoDB 8
3.1.1) Data Modeling 8
3.1.2) Concurrency Control 9
3.2) HTML Canvas 9
3.3) React 10
3.4) Flutter 11
3.5) gRPC 11

4. Requirement analysis / Design / Architecture

4.1) Requirement analysis 13
4.1.1) Functional requirement 13
4.1.2) Non-functional requirement 15
4.2) Usecase diagram 16
4.21) Fully dressed use case 17
4.3) Architecture 29
4.3.1) Backend 29
4.3.2) Customer Mobile Application 33
4.3.3) Business Web Application 37
4.3.4) Platform Admin Web Application 37
4.3.5) Employee Mobile Application 38
4.3.6) Backend Observable Pattern 38
4.4) Database Schema 38

5. Development

5.1) Development tools 40
5.1.1) General 40
5.1.2) Backend Server 40
5.1.3) Mobile Development 41
5.1.4) Web Development 41

5.2) Development method 42
5.2.1) Requirement Analysis 42
5.2.2) System Design 43
5.2.3) Backend Server Development 43
5.2.4) Mobile Development 43
5.2.5) Web Development 44

6. Results
6.1) Customer Mobile Application 45
6.2) Business Web Application 61
6.3) Platform Admin Web Application 70
6.4) Employee Mobile Application 72

7. Conclusion
7.1) Summary 77
7.2) Future Work 77

8. Bibliography 81

List of Figures

Figure 2.1 SF Cinema application movie listing and seat selection 5
Figure 2.2 Floorplanner's editor 7
Figure 3.1 gRPC 12
Figure 4.1 Use case diagram 16
Figure 4.2 Backend packages 30
Figure 4.3 Backend entity package 31
Figure 4.4 Backend grpc_user package 32
Figure 4.5 Customer Mobile packages 33
Figure 4.6 Customer Mobile entity package 34
Figure 4.7 Customer Mobile user_repository package 35
Figure 4.8 Favorite Bloc classes 36
Figure 4.9 Web package 37
Figure 4.11 Database schema 39
Figure 6.1 Customer Mobile Login Page 44
Figure 6.2 Customer Mobile Sign Up Page 45
Figure 6.3 Customer Mobile Navigation Drawer Page 46
Figure 6.4 Customer Mobile Home Page 47
Figure 6.5 Customer Mobile Business Detail Page 48
Figure 6.6 Customer Mobile Business Location Page 49
Figure 6.7 Customer Mobile View Schedule Page 50
Figure 6.8 Customer Mobile Reservation Page 51
Figure 6.9 Customer Mobile Seat Update 52
Figure 6.10 Customer Mobile Order Page 53
Figure 6.11 Customer Mobile Order Detail Page 54
Figure 6.12 Customer Mobile Search Reservation Page 55
Figure 6.13 Customer Mobile Search Reservation Result 56
Figure 6.14 Customer Mobile Search Business Page 57
Figure 6.15 Customer Mobile Search Business Result 58
Figure 6.16 Customer Mobile Favorites Page 59
Figure 6.17 Business Web Login Page 60
Figure 6.18 Business Web Sign Up Page 61
Figure 6.19 Business Web Home Page 62
Figure 6.20 Business Web Request Change Modal 63
Figure 6.21 Business Web Schedule Page 64
Figure 6.22 Business Web Schedule Reservation Placement 65
Figure 6.23 Business Web Placement Page 66
Figure 6.24 Business Web Menu Page 67
Figure 6.25 Business Web Employee Page 68
Figure 6.26 Homepage of Admin Web Application 69
Figure 6.27 Detail page of Admin Web Application 69
Figure 6.28 Admin Web Registration Detail Page 70
Figure 6.29 Admin Web Change Request List Page 70
Figure 6.30 Admin Web Change Request Detail Page 71

Figure 6.31 Employee Mobile Login Page 72
Figure 6.32 Employee Mobile Reservation List 73
Figure 6.33 Employee Mobile Reservation Detail 74
Figure 6.34 Employee Mobile Scanned Ticket 75

Table of Tables

Table 4.1 Fully dressed “Submit request for business information modification” use case 17
Table 4.2 Fully dressed “Create placements” use case 17
Table 4.3 Fully dressed “Create reservation time slot” use case 18
Table 4.4 Fully dressed “Cancel reservation time slot” use case 18
Table 4.5 Fully dressed “View reserved seat” use case 19
Table 4.6 Fully dressed “Create menu” use case 19
Table 4.7 Fully dressed “Create employee account” use case 20
Table 4.8 Fully dressed “View Business Information” use case 20
Table 4.9 Fully dressed “Cancel reservation” use case 21
Figure 4.10 Fully dressed “Select and reserve specific seats” use case 22
Table 4.11 Fully dressed “Search for Business” use case 22
Table 4.12 Fully dressed “View available time slot for seat reservation” use case 23
Table 4.13 Fully dressed “View list of reservation made” use case 23
Table 4.14 Fully dressed “View QR ticket” use case 24
Table 4.15 Fully dressed “View list of time slot for their business” use case 24
Table 4.16 Fully dressed “Verify Ticket” use case 25
Table 4.17 Fully dressed “Scan Qr ticket” use case 25
Table 4.18 Fully dressed “View the list of awaiting business registration” use case 26
Table 4.19 Fully dressed “Approve await business registration requests” use case 26
Table 4.20 Fully dressed “Reject await business registration requests” use case 27
Table 4.21 Fully dressed View the list of pending business request“” use case 27
Table 4.22 Fully dressed “Approve request” use case 28
Table 4.23 Fully dressed “Reject request” use case 28

Chapter 1
Objectives and Problem Description

1.1 Motivation

When customers reserve seats at an establishment (high end restaurants, bars etc.), they sometimes want the ability to select their seats, so that they do not feel as though they are given an unfavourable seating placement. Customers want the ability to view available seats in real time, and make their own selection without hassle. It is usually easier for customers to interact with a mobile application than wait for human support as they can be slow to respond. In addition, most businesses don't want to create their own online reservation and schedule system. Businesses want to take advantage of a platform that provides online reservation and scheduling with minimal effort. The platform becomes another place for their business to be discovered by potential customers.

Existing online reservation applications are usually categorized into two types, those with seat selection or simple queuing system. For the former, fine-grained control over seat selection can usually be found at applications made by giant cinema enterprises such as Major Cineplex Group or SF Group. The latter are systems that simply put users into a waiting queue and can usually be found in restaurants (they solve the problem of queuing but not reservation). Furthermore, both of these existing systems are either aimed at one specific business (privately owned) or are too simple for certain businesses.

Our system solves the needs of smaller businesses who find it unprofitable to take on the challenge of developing their own online reservation application from scratch but want their customers to easily make reservations. The mobile application eliminates the need for human interaction during the reservation process while providing the necessary visual information to the customers, thus resulting in a more efficient and better user experience.

1.2 Objectives

The objective of the software engineering project is to create a platform that

allows businesses to take advantage of the reservation system, and allow customers to easily discover and make reservations. The objectives are listed as follows:

1. Develop a mobile application for customers to search, view details and reserve specific seats at a business.

2. Develop a web application for businesses to manage their information, create their placements and define their available schedule.

3. Develop an admin website to approve business registration to our platform, and manage information updates.

4. Develop an on-site application which will be used by employees to validate electronic tickets of customers.

5. To develop an online seat reservation system that supports real-time updates to the customers mobile application when there are multiple concurrent users.

1.3 Scope of Work

The scope of this project are listed as follows:

- Use Flutter to develop the mobile application according to the material design guidelines.

- Develop the seating placement component by yourself by using HTML canvas to draw graphics on the web page.

- Develop an on-site application which will run on a tablet and validate electronic tickets.

- To use gRPC streaming to provide real-time updates of seat status to the mobile and on-site clients.

1.4 Thesis Structure

This thesis consists of seven chapters which are arranged as follows:

- Chapter 1 Objectives and Problem description - refers to the motivation, objectives, scope of work, and thesis structure of this thesis.

- Chapter 2 Related work – proposes the Literature survey, various software that are relevant to this project, and comparison.

- Chapter 3 Background knowledge - explains the knowledge and technology necessary for the reader to understand the thesis.

- Chapter 4 Requirement analysis/ System Architecture and Design – presents the requirement of the system, the use case diagram, and relevant system architecture diagram.

- Chapter 5 Development - explains the concepts, tools, and techniques that are used in developing the project.

- Chapter 6 Results - refers to the results of software demonstration, which consists of the user interfaces of the software and real-world applications.

- Chapter 7 Conclusion and Future work - is the chapter that talks about the conclusion, future work and improvements to the project.

Chapter 2
Related works

This chapter describes, presents and compares existing related software relevant to the project. These include SF Cinema mobile application, QueQ mobile application and Floorplanner web application. It is important to note that these example applications have somewhat similar features however they are not intended to be seen as direct competitors as their services do not align with our objective.

2.1 SF Cinema

SF Cinema is a mobile solution that allows the users to access services provided by SF Cinema from anywhere with internet access. The services include: searching for movie showtime, viewing details of the showtime, booking specific seat(s), and using the generated QR code as an electronic ticket.

The mobile application follows a common interface seen in most online reservation solutions. The general action taken to book seats with SF Cinema mobile application are as defined below:

1. The users start by selecting their desired location and view the list of movies available for booking.
2. The user selects a movie and an available show time.
3. A 2D seating placement view is rendered onto the application. This view shows the list of taken seats and the remaining available seats for booking.
4. Users can then select from the available seats and proceed to checkout. Once the checkout process is started, the user is given a limited time to complete the purchase.
5. During the checkout process, the system ensures that no other users may proceed with the selected seats. If the user cancels the

process, only then may other users proceed with the cancelled seats.

One of SF Cinema's strong points is how it handles concurrent access during seat selection. The following describes what happens when two users have selected the same seat:

1. There are two users, UserA and UserB, concurrently accessing the same movie and show time seat booking view.
2. UserA proceeds with checkout having selected seat A1 and A2.
3. At the same time UserB has yet to proceed with the checkout process but have also selected seats A2 and B4.
4. At this point, UserB is notified that seat A2 has been taken. The view is updated and seat A2 is removed from UserB selection automatically while still retaining seat B4 which is still available.

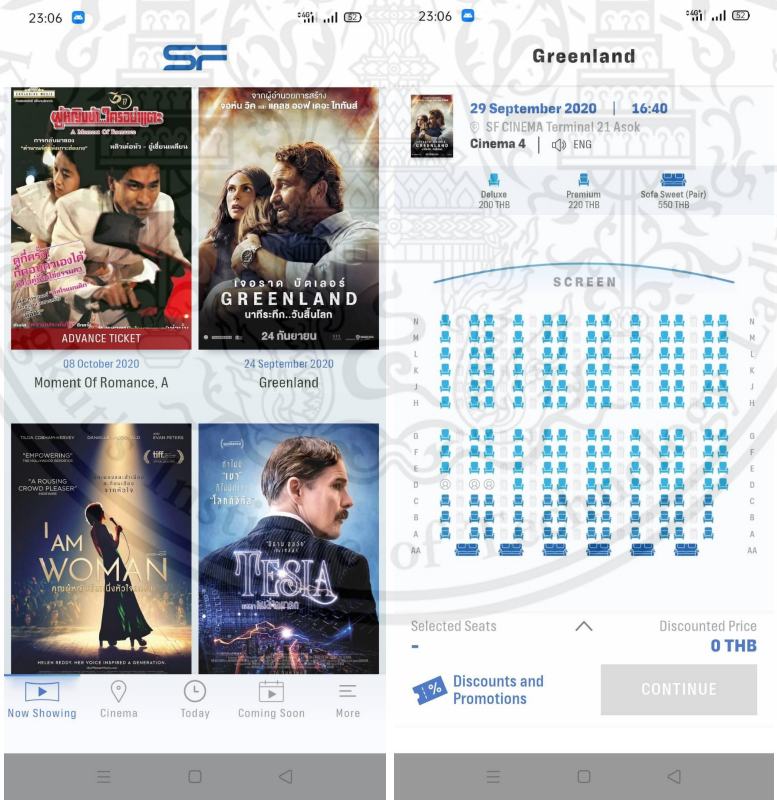


Figure 2.1: SF Cinema application movie listing and seat selection

2.2 QueQ

QueQ is an online queue reservation mobile application mostly seen in mid-tier restaurants. The objective of QueQ is to eliminate the need for users to wait in line for their queue. Most interaction with QueQ are described as follows:

1. The users start by viewing nearby restaurants, based on the user's location (Location service permission is a requirement for QueQ).
2. The user selects their restaurant and number of seats needed.
3. The user is given a queue number and ticket that can be viewed through the mobile application.
4. Users can view the status of their queue via the mobile application or at the restaurant itself.
5. The user can use the electronic ticket as proof of their queue number at the restaurant.

Although queue is an online reservation process, it focuses on reservations made usually no more than an hour before the actual time. Users do not have the ability to make any selections, it is simply a queuing system used in place of standing in line.

2.3 Floorplanner

Floorplanner describes itself as a lightweight editor that runs on the browser. The web application allows users to create 2D floor plans placed under their accounts, which can be accessed from anywhere with internet access. Users can also render a 3D view of their 2D plan, place cameras and view it from different perspectives. The general steps to start a floor planning project with Floorplanner are as follows:

1. The user starts by creating a new project and clicking on it to enter into the floor editor page for that particular project.
2. The user can then place down structures and furniture describing their designs (they can switch between 2D and 3D view here).
3. Clicking on what they placed gives the user options to include extra descriptions indicated by a menu drawer linking to the selected object.
4. Once the user has finished placing everything, they can click save to persist the plan.

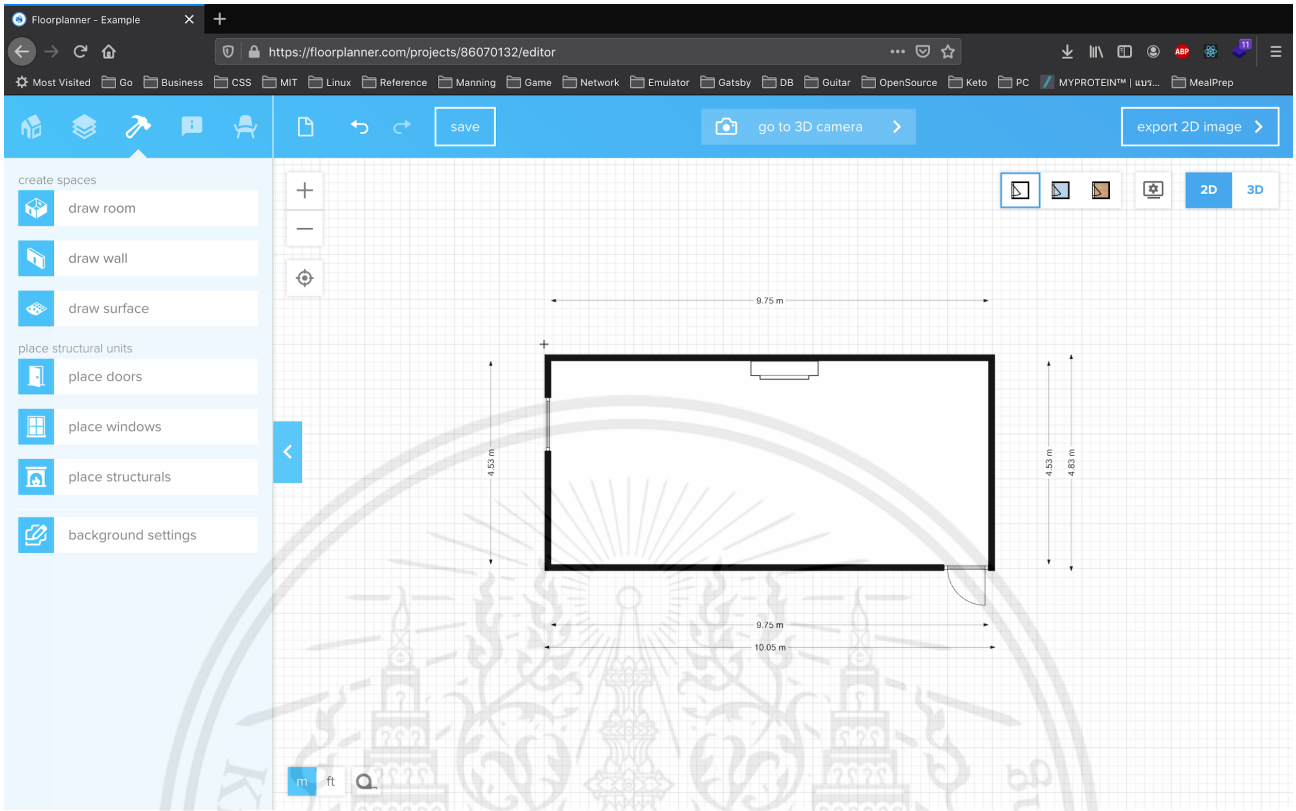


Figure 2.3: Floorplanner's editor view

Chapter 3
Background Knowledge

This chapter provides relevant background knowledge on the technology used in order to get a better understanding of the project.

3.1 MongoDB

MongoDB is a document-oriented database, it stores data in documents, similar to JSON objects. We chose mongo because the reservation placement seating is difficult to represent in a relational database.

3.1.1 Data Modeling

With MongoDB document data structure, data modeling methods in MongoDB revolve around embedding sub documents or storing references to another document entirely. Both of these methods have their own strengths and weaknesses. Some benefits of embedding documents in MongoDB include simpler queries, better performance and single document atomicity. On the other hand, storing references can be seen as a normalized data model. The general rule of thumb when choosing between the two strategies is the potential size of the number of references between the two documents.

When modeling application data for MongoDB various operational factors come into play. The things to take into consideration are the size of the document (MongoDB has a 16 megabytes maximum size limit per document), query performance, atomicity, the nature of your read (retrieve small subset each time or retrieve several) and write operations, and indexing overhead.

Unlike a relational database, by default MongoDB does not require the documents in a collection to have the same schema. To ensure that documents will share a similar structure MongoDB have included the support for schema

validation during update and insert operations. These validation rules are applied to an entire collection in JSON schema format.

3.1.2 Concurrency Control and Transaction

MongoDB allows multiple clients to read/write data at the same time. In order to ensure consistency MongoDB uses multiple granularity locking and optimistic concurrency control. MongoDB concurrency control mechanism guarantees that a write to a single document either occurs in full or not at all.

With multiple granularity locking, locks are set on objects that contain other objects. A lock on the object will lock the object as well as the object it contains (this structure can be seen as a tree-like structure where each object can be denoted in terms of nodes in a tree). In addition to shared (S) and exclusive (X) locks, MongoDB uses granularity locking concept of intent shared (IS) and intent exclusive (IX) locks to provide finer granularity control. In order to lock a node, with S or X, it has to acquire IS or IX lock (depending on the operation) on all of its ancestors. In order to optimize throughput, MongoDB will grant all non-conflicting lock requests in the queue (move them up) at the same time.

In older versions of MongoDB, only single document operations were compliant with the ACID properties. With MongoDB version 4.0 onwards MongoDB now supports multi-document transactions on replica sets. Furthermore, with the recent version 4.2, MongoDB now supports multi-document transactions on sharded clusters. Operations supported in transactions in MongoDB include CRUD operations on documents.

By default MongoDB isolation levels do not permit outside reading during a transaction. If clients wish to read without consistency guarantee, they can edit the “read concern” property which can result in the clients seeing the result of writes before data is made durable.

3.2 HTML Canvas

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

The canvas element is part of HTML5 and allows for dynamic, scriptable rendering of 2D shapes and bitmap images. It is a low level, procedural model that updates a bitmap and does not have a built-in scene graph, but through WebGL it allows 3D shapes and images to be displayed. HTML5 Canvas also helps in making 2D games.

Konva is another HTML extension that will allow the user to interact with objects inside the canvas.

A canvas consists of a drawable region defined in HTML code with height and width attributes. JavaScript code may access the area through a full set of drawing functions similar to those of other common 2D APIs, thus allowing for dynamically generated graphics. Some anticipated uses of canvas include building graphs, animations, games, and image composition.

3.3 Next js (React)

React is a JavaScript library that is used for helping developers build graphical user interfaces, or GUIs. In web applications, GUI or UI is a component that users can interact with in order to use the application, the example of the menu, search bar and buttons.

Before React JS, developers were stuck building UIs by hand with “vanilla JavaScript” or with less UI-focused React predecessors like jQuery. That meant longer development times and plenty of opportunities for errors and bugs. In addition to providing reusable React library code, React comes with two key features that add to its appeal for JavaScript developers: JSX Virtual DOM

During this process, browsers create something called a Document Object Model (DOM), a representational tree of how the web page is arranged. Developers can then add dynamic content to their projects by modifying the DOM with languages like JavaScript. JavaScript extension(JSX) is a React extension that makes it easy for web developers to modify their DOM by using simple, HTML-style code.

Next.js is a react framework for production. It allows the best developer experience with all the features needed for production: hybrid static & server rendering, TypeScript support, smart bundling, route prefetching, and more. No additional configuration is needed.

3.4 Flutter

Flutter is a free and open-source mobile UI framework created by Google. It allows developers to create a native mobile application with only one codebase. This allows the developer to use one programming language and one codebase to create two different apps, iOS and Android.

The central idea behind Flutter is the use of widgets. It's by combining different widgets that developers can build the entire UI. Each of these widgets defines a structural element, a stylistic element, a layout aspect, and many others.

Flutter doesn't use OEM widgets, but provides developers with its own ready-made widgets that look native to Android or iOS apps. However, developers are allowed to create their own widgets as well.

Flutter also provides developers with reactive-style views. To avoid performance issues deriving from using a compiled programming language to serve as the JavaScript bridge, Flutter uses Dart. It compiles Dart ahead of time (AOT) into the native code for multiple platforms.

That way, Flutter can easily communicate with the platform without needing a JavaScript bridge that involves a context switch between the JavaScript realm and the native realm. As you can imagine, compiling to native code also boosts the app startup time.

3.5 gRPC

gRPC is an open source general purpose RPC framework originally developed at Google (now part of the CNCF) that can run in any environment. It is most often used with Protocol Buffers, as its Interface Definition Language (IDL) and as an interchangeable message format.

gRPC allows client applications to directly call a method on the server (can be on a different or same machine) as if it were calling a method locally. When working with gRPC, you define methods which can be called remotely with their parameters and return types. On the server side, the server implements an interface for the service (collection of defined methods) and runs a gRPC server to handle client requests. On the client side, a stub is provided which provides the same method implemented by the server side - allowing the clients to make calls to a server as if it were a normal local function.

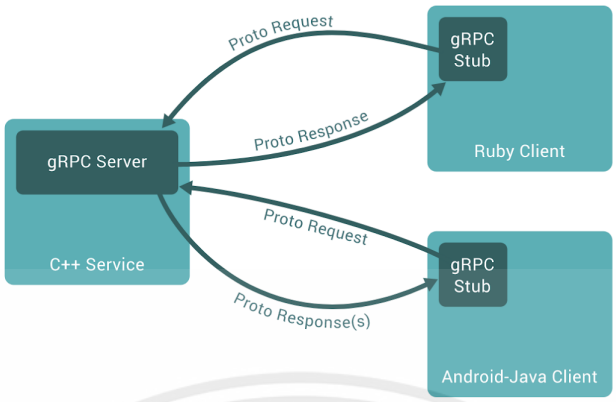


Figure 3.1 gRPC

One of the most important aspects of gRPC is that gRPC servers and clients can be written in several different languages and will still be able to interact with each other through the defined interface.

Currently, there are supported types of service methods that can be defined in gRPC.

- **Unary RPCs** - Like normal function calls, a single request is made and a single response is returned.
  ```
  rpc SayHello(HelloRequest) returns (HelloResponse);
  ```

- **Server streaming RPCs** - With this type of method, the client sends a request to a server and gets a stream of messages to read. The client reads until there are no more messages, message ordering is guaranteed within individual RPC calls.
  ```
  rpc LotsOfReplies(HelloRequest) returns (stream HelloResponse);
  ```

- **Client streaming RPCs** - In this case the client sends a stream of messages to the server and once the server has finished reading the sequence of messages it returns a response.
  ```
  rpc LotsOfGreetings(stream HelloRequest) returns (HelloResponse);
  ```

- **Bidirectional Streaming RPCs** - This is when both the client and server send sequences of messages to each other (the streams are separate). Clients and servers can read/write in whatever order they like. For example, the server could do one read then write or it could do all the read before writes etc.
  ```
  rpc BidiHello(stream HelloRequest) returns (stream HelloResponse);
  ```

Chapter 4
Requirement analysis / Design / Architecture

4.1 Requirement Analysis

This section describes all the requirements necessary for our application. The requirement will be both functional and non-functional. The functional requirement will be an action the user of our system must be able to do. It will describe what the action the user will take and a basis of how the action leads the user through our application.

For our project, there will be four main actors.

1. Business Managers
2. Customer
3. Employee
4. Platform Admin

4.1.1 Functional Requirement

The first main actor, the business manager. This actor will be the user who uses a web application to manage their business information. The managers must login to their account before they can create or edit any information about the business. The manager can change the business information such as name and business location. The manager can create a seating placement that represents their actual seats/tables. The Placement will contain information such as the available space per seating, location, and name.

- Managers must be able to submit request to modify their business information
- Managers must be able to create placement view for their business
- Managers must be able to create time slots which define an available time range in which customers can make a reservation for that time slot
- Managers must be able to cancel a reservation time slot that they have

created

- Managers must be able to view which seats have been reserved for a time slot that they have created
- Managers must be able to create menu items that will be displayed alongside the business details
- Managers must be able to edit their menu items
- Managers must be able to create employee user account
- Managers must be able to edit their employee user account

The second actor will be the customer. The customer will be using the mobile application for seat reservation. The customer must be able to access the basic information about the business such as name, location and pictures of the place. The basic information will help the customer to make a decision if they should make a reservation. The customer must be able to reserve a specific seat through the mobile application. If desired, the customer must be able to cancel the reservation. After a successful reservation, the customer must get a QR ticket that can be validated by the employee, on-site.

- The customer must be able to view details of businesses
- The customer must be able to search for businesses with the following parameters: location, name, date range, and type
- The customers must be able to view available time slots of a business given a date time range
- The customers must be able to select and reserve specific seats.
- The customers must be able to see real time updates of seatings availability of a schedule while viewing a business schedule (time slot)
- The customers must be able to their orders (which is their tickets will reside)
- The customers must be able to cancel their reservation
- The customer must be able to view their QR ticket

The third actor is the employee, who acts as a receptionist or a waiter/waitress. The employee will use a scanner to validate the customer's electronic QR ticket and get the information on the customer's reservation.

- Employees must be able to view a list of time slots (reservations) that have been created by their managers
- Employees must be able to scan the QR ticket which will validate the ticket
- Employees must be able to view ticket information of the customer after validation

The last actor is the admin, which overviews the system and checks business requests. They are allowed to view the business request and choose to approve or refuse the request.

- The admin must be able to view a list of pending business registration approval request
- The admin must be able to reject or approve of a business registration request
- The admin must be able to view a list of pending business information change request
- The admin must be able to reject or approve of a business information change request

4.1.2 Non-Functional Requirement

The following list the non-functional requirements in our system.

- Business managers should receive email notification when their business has been approved.
- Business managers should receive email notification when their change request has been approved.
- Customers should get an email notification if the business has

cancelled one of their reservations in which they have already made a reservation for.

4.2 Usecase Diagram

The system provides both web and mobile applications. The web application allows the managers to create menu and placement to use when creating a reservation schedule. While the mobile application is used by a customer to make a reservation or view information on the business.

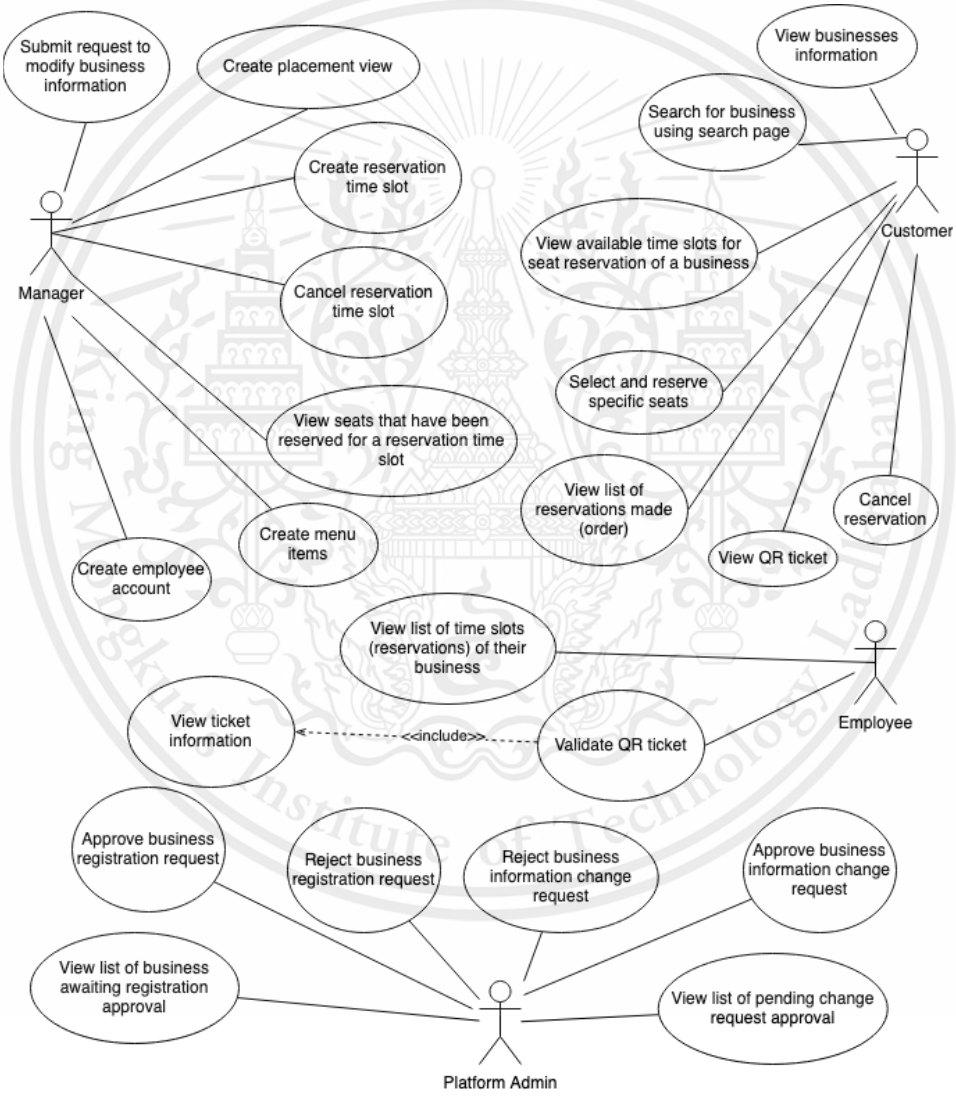


Figure 4.1 Usecase diagram

4.2.1 Fully dressed usecase

The following table will provide more information into each of the

usecase presented in figure 4.1.

Use Case Section	**Description**
Use case name	Submit request for business information modification
Scope	Business System
Level	User goal
Primary Actor	Manager
Stakeholder and interests	The manager can edit business information about the store through a web application. (Eg. store location)
Precondition	Must have an account
Success guarantee	Change of business information to a newly edited one
Main scenario	1) Click edit information 2) Change the information and save it 3) Request will be send to the admin automatically, the change will only apply after admin accept the request
Extensions	If the admin rejects the request the information will not change and there will be a notification to the user.

Table 4.1 Fully dressed “Submit request for business information modification” use case

Use Case Section	**Description**
Use case name	Create placement
Scope	Business System
Level	User goal
Primary Actor	Manager
Stakeholder and interests	Managers are allowed to create Placements which define their businesses floor design (seating placement, stage area etc.).
Precondition	Must have an account
Success guarantee	Created design of the store
Main scenario	1) Manager click create new placement which will redirect to a page with canvas and sidebar

	2) The manager can drag and drop item(table,chair,wall) from sidebar into the canvas to create layout of the restaurant 3) Click save to save the current design of the Placement

Table 4.2 Fully dressed “Create placements” use case

Use Case Section	**Description**
Use case name	Create reservation time slot
Scope	Reservation system
Level	User goal
Primary Actor	Manager
Stakeholder and interests	Managers are allowed to select a period of time in which customers will be able to make reservations.
Precondition	Must have an account
Success guarantee	Created a reservation schedule with selected date
Main scenario	1) Manager click create new schedule, and select the date and time period for this reservation schedule 2) The manager select date 3) The manager select time range 4) The manager click create to create the new reservation time slot

Table 4.3 Fully dressed “Create reservation time slot” use case

Use Case Section	**Description**
Use case name	Cancel reservation time slot
Scope	Reservation system
Level	User goal
Primary Actor	Manager
Stakeholder and interests	Managers are allowed to select a period of time and cancel the time slot in the schedule page.
Precondition	Must have available time slot
Success guarantee	Remove the time slot from being reservable

Main scenario	1) Manager choose a time slot 2) Click “cancel reservation”
Extensions	An email notification is sent to the customers who have made a reservation to the particular reservation time slot.

Table 4.4 Fully dressed “Cancel reservation time slot” use case

Use Case Section	**Description**
Use case name	View reserved seat
Scope	Reservation System
Level	User goal
Primary Actor	Manager
Stakeholder and interests	Managers are allowed to view information on previously created reservation time slots.
Precondition	Must have created a reservation time slot
Success guarantee	View updated information of the reservation schedule
Main scenario	1) Manager click on the calendar 2) The calendar will show the time slot of the month 3) The manager click on an individual reservation, more information on the particular reservations schedule is retrieved 4) The manager clicks on “view placement” to see the reserved table in visual format.

Table 4.5 Fully dressed “View reserved seat” use case

Use Case Section	**Description**
Use case name	Create menu
Scope	Business System
Level	User goal
Primary Actor	Manager
Stakeholder and interests	Managers are allowed to create Menu which define the menu items which will be displayed on the mobile application

Precondition	Login to account
Success guarantee	Created a list of menu items
Main scenario	1) Manager click create new menu which will redirect to a page where managers can add items to the menu 2) The manager can add new items by filling in the form and pressing add 3) The manager can edit or delete an existing item by selecting it from a list of added menu items

Table 4.6 Fully dressed “Create menu” use case

Use Case Section	**Description**
Use case name	Create employee account
Scope	Employee System
Level	User goal
Primary Actor	Manager
Stakeholder and interests	Managers are allowed to create employee account for the employee to use with an on-site application
Precondition	Login to account
Success guarantee	Created new employee account
Main scenario	1) Manager click create account which will redirect to a page where managers can add new employee account 2) The manager can add new account by filling in the form and pressing “save” 3) The manager can edit or delete an existing account by selecting it from a list of employee account

Table 4.7 Fully dressed “Create employee account” use case

Use Case Section	**Description**
Use case name	View Business Information
Scope	User System
Level	User goal
Primary Actor	Customer

Stakeholder and interests	Customer able to view business information such as menu, pricing, and store location
Precondition	Downloaded the application and have an account
Success guarantee	User able to view all the information
Main scenario	1) The customer clicks on a business card component 2) The application will show the detail of the business in a new Business Detail Page

Table 4.8 Fully dressed “View Business Information” use case

Use Case Section	**Description**
Use case name	Cancel reservation
Scope	Reservation system
Level	User goal
Primary Actor	Customer
Stakeholder and interests	Cancel a reserved seat
Precondition	Reserved a table/seat
Success guarantee	Reserved seat is cancelled
Main scenario	1) The customer go into the application and click view order 2) The page will show a list of previously booked seat/table 3) Customer click on the individual item, information on the order is shown 4) Customer click cancel

Table 4.9 Fully dressed “Cancel reservation” use case

Use Case Section	**Description**
Use case name	Select and reserve specific seats
Scope	Reservation system
Level	User goal
Primary Actor	Customer
Stakeholder and	The user can select specific seat/table for reservation

interests	
Precondition	The store owner must have made a reservation time slot
Success guarantee	The customer can make a reservation for specific seat
Main scenario	1) The customer click a reservation time slot card component 2) The customer click at a blue seat (available) 3) The customer the confirm to make a reserve, the seat will turn grey to make sure the seat is reserved and qr code will be generated for the user
Extensions	If two customers book the same table, the first customer will make a successful reservation and the error message will be shown on the second customer application and ask the customer to reselect and make the booking again.

Figure 4.10 Fully dressed “Select and reserve specific seats” use case

Use Case Section	**Description**
Use case name	Search for Business
Scope	User System
Level	User goal
Primary Actor	Customer
Stakeholder and interests	The user can search for a list of business by selecting providing some search criteria
Precondition	Downloaded the application
Success guarantee	List of businesses is presented to the user
Main scenario	1) The user click on the search on the navigation drawer 2) User provide some information to the search form 3) The application fetch list business based on the given information

Table 4.11 Fully dressed “Search for Business” use case

Use Case Section	**Description**
Use case name	View available time slot for seat reservation
Scope	User System
Level	User goal

Primary Actor	Customer
Stakeholder and interests	The user can view available seats and time slots for the business.
Precondition	Downloaded the application
Success guarantee	List of businesses is presented to the user
Main scenario	1) After discover the business of interest, the user click into the business 2) Scroll down the page the user will be able to see available time slot 3) Click the time slot and it will show the available seats to the customer

Table 4.12 Fully dressed "View available time slot for seat reservation" use case

Use Case Section	**Description**
Use case name	View list of reservation made
Scope	User System
Level	User goal
Primary Actor	Customer
Stakeholder and interests	The user will be able to see the history of the reservation he had made.
Precondition	Make a reservation before
Success guarantee	The user can see the list of reservations made
Main scenario	1) Click "order" from the navigation drawer 2) A list of reservations made is displayed

Table 4.13 Fully dressed "View list of reservation made" use case

Use Case Section	**Description**
Use case name	View QR ticket
Scope	User System
Level	User goal
Primary Actor	Customer

Stakeholder and interests	The user will be able to see the QR ticket of the reservation he had made.
Precondition	Made a reservation
Success guarantee	The user can view the QR ticket
Main scenario	1) Click the order card component 2) Detailed view of the reservation is shown including the QR code

Table 4.14 Fully dressed “View QR ticket” use case

Use Case Section	**Description**
Use case name	View list of time slot for their business
Scope	Reservation system
Level	User goal
Primary Actor	Employee
Stakeholder and interests	The employee can see time slots of the business
Precondition	Employee login to the account created by the manager
Success guarantee	Show the table the customer reserved
Main scenario	1) The employee login 2) They will have the list of reservation time slots displayed on the home page.

Table 4.15 Fully dressed “View list of time slot for their business” use case

Use Case Section	**Description**
Use case name	Verify Ticket
Scope	Reservation system
Level	System goal
Primary Actor	Employee
Stakeholder and interests	Verify the QR ticket issue to the customer and bring the customer to the reserved table

Precondition	Customer show the QR to the employee
Success guarantee	Show the table the customer reserved
Main scenario	1) The customer show the received qr ticket to the employee 2) The employee select the corresponding reservation time slot 3) The employee scan the QR code to verify and retrieve ticket information
Extensions	The QR ticket does not match selected reservation time slot

Table 4.16 Fully dressed “Verify Ticket” use case

Use Case Section	**Description**
Use case name	View Ticket information
Scope	Reservation System
Level	User goal
Primary Actor	Employee
Stakeholder and interests	The employee can get the reservation information and lead the customer to the right table
Precondition	The employee has scanned the employee ticket for verification
Success guarantee	The reservation information is shown
Main scenario	1) The employee scan the QR ticket from customer mobile phone 2) The ticket information is shown

Table 4.17 Fully dressed “Scan Qr ticket” use case

Use Case Section	**Description**
Use case name	View the list of awaiting business registration
Scope	Register business information
Level	User goal
Primary Actor	Platform Admin
Stakeholder and interests	See list of new registration from businesses
Precondition	Login to the system

Success guarantee	Allow the admin to view all the registration from businesses
Main scenario	1) Login to the admin page 2) Check new registration page
Extensions	none

Table 4.18 Fully dressed “View the list of awaiting business registration” use case

Use Case Section	**Description**
Use case name	Approve await business registration requests
Scope	Registration business information
Level	User goal
Primary Actor	Admin
Stakeholder and interests	Approve business and add the business to the system
Precondition	New registration is made
Success guarantee	Allow the admin to view all the request from businesses
Main scenario	1) Login to the admin page 2) Check new registration page 3) Click to see the information 4) Click approve business registration.
Extensions	none

Table 4.19 Fully dressed “Approve await business registration requests” use case

Use Case Section	**Description**
Use case name	Reject await business registration requests
Scope	Reject business information
Level	User goal
Primary Actor	Admin
Stakeholder and interests	Reject the registration and send and email to the business

Precondition	New registration is made
Success guarantee	Reject the business into the system
Main scenario	1) Login to the admin page 2) Check new registration page 3) Click to see the information 4) Click reject business registration.
Extensions	none

Table 4.20 Fully dressed “Reject await business registration requests” use case

Use Case Section	**Description**
Use case name	View the list of pending business requests
Scope	Updating business information
Level	User goal
Primary Actor	Admin
Stakeholder and interests	See the request from businesses
Precondition	Login to the system
Success guarantee	Allow the admin to view all the request from businesses
Main scenario	1) Login to the admin page 2) Check new request page
Extensions	none

Table 4.21 Fully dressed View the list of pending business request“” use case

Use Case Section	**Description**
Use case name	Approve the request
Scope	Updating the business information
Level	System goal
Primary Actor	Admin
Stakeholder and interests	Update business information after the admin approve

Precondition	Request show in new request page
Success guarantee	Approve - customer business's information update
Main scenario	1) View the new request 2) Click to see the information 3) Click approve

Table 4.22 Fully dressed "Approve request" use case

Use Case Section	**Description**
Use case name	Reject the request
Scope	Updating the business information
Level	System goal
Primary Actor	Admin
Stakeholder and interests	Update business information after the admin approve
Precondition	Request show in new request page
Success guarantee	Reject - Notify the customer and no change is made
Main scenario	1) View the new request 2) Click to see the information 3) Click reject

Table 4.23 Fully dressed "Reject request" use case

4.3 Architecture

This section describes the architecture of our backend services, mobile client application, and web application. The following subsections attempt to explain the architecture used by each respective platform. We will also briefly describe the observable pattern we use in our backend to send real time updates.

4.3.1 Backend

Figure 4.2 shows the packages within our backend system. Packages ending with “pb” are packages that have been generated with protocol buffers and gRPC, they contain boilerplate code for a gRPC server, request and response data structure, and server interface with methods corresponding to our gRPC endpoints. Packages starting with “grpc” contain implementation of the server interface generated in “pb” packages, they contain the business logic for the gRPC services. Packages ending with “api” are packages that have been generated from an OpenAPI specification file version 3, they contain boilerplate code for a HTTP server, request and response data structure, and server interface with methods corresponding to our HTTP endpoints. The packages starting with “http” contain implementations of the server interface generated in “api” packages, they contain the business logic for the http services.

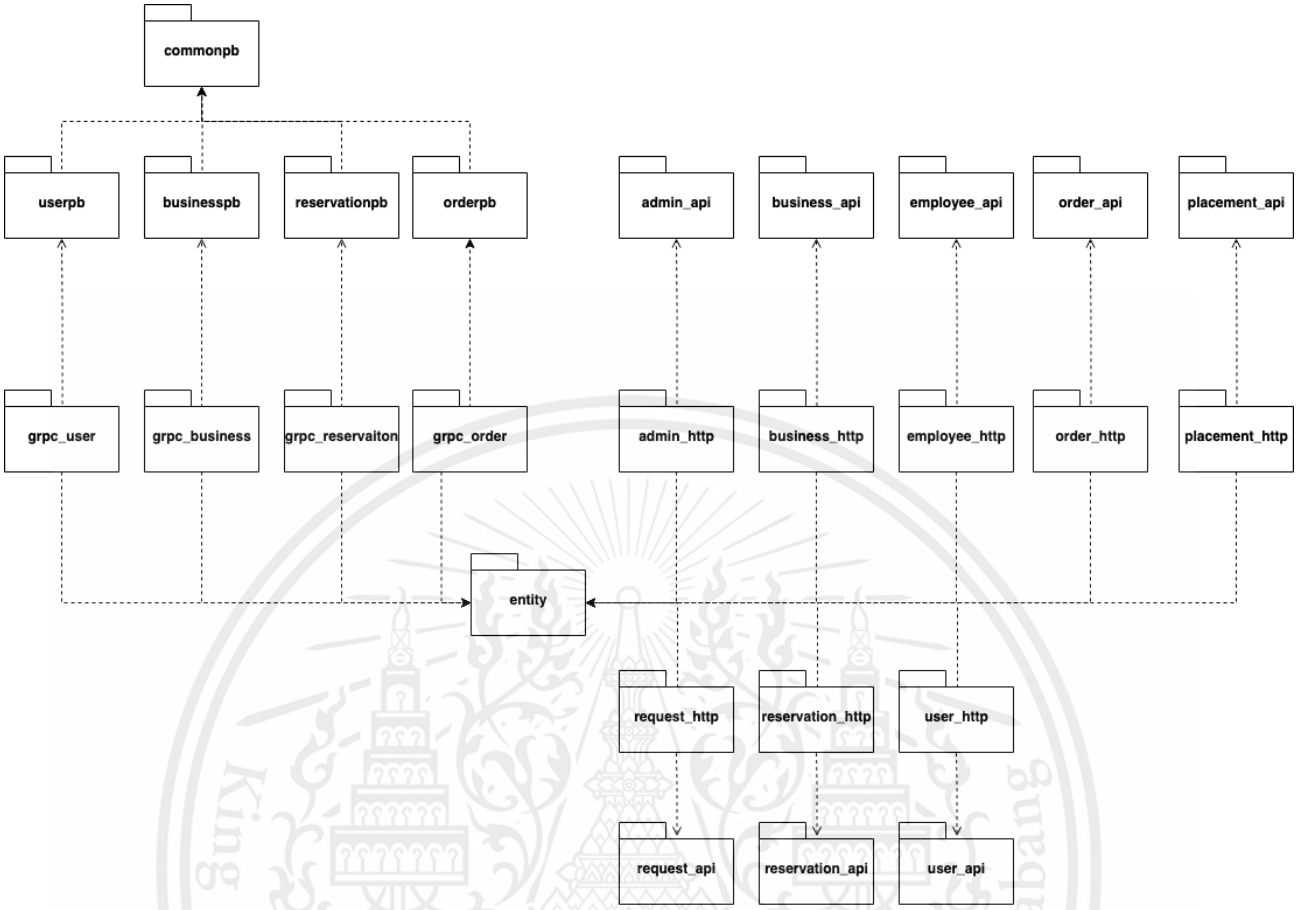


Figure 4.2 Backend packages

Figure 4.3 shows the class diagram within the "entity" package. This package is meant to simply share common reusable classes/objects that exist within our system, they are almost identical to the document structure inside the database.

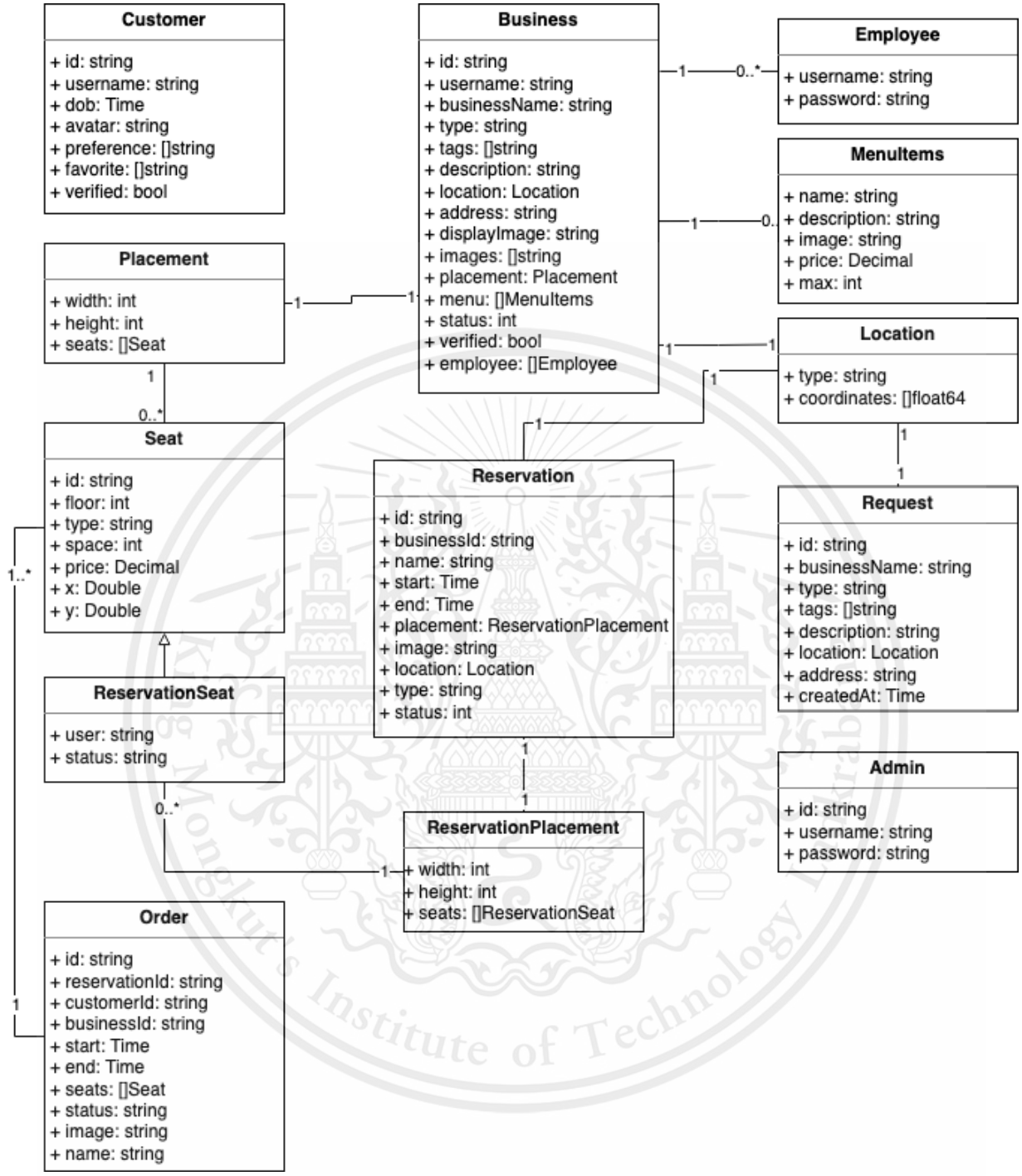


Figure 4.3 Backend entity package

Figure 4.4 shows the classes within one of the “grpc” packages, it is a simple package. Inside this package is the concrete implementation of the generated server interface, a small interface that represents interaction with the data source (database), and the mock/concrete implementation of that interface. All of the packages which implement the server interface all have a similar structure.

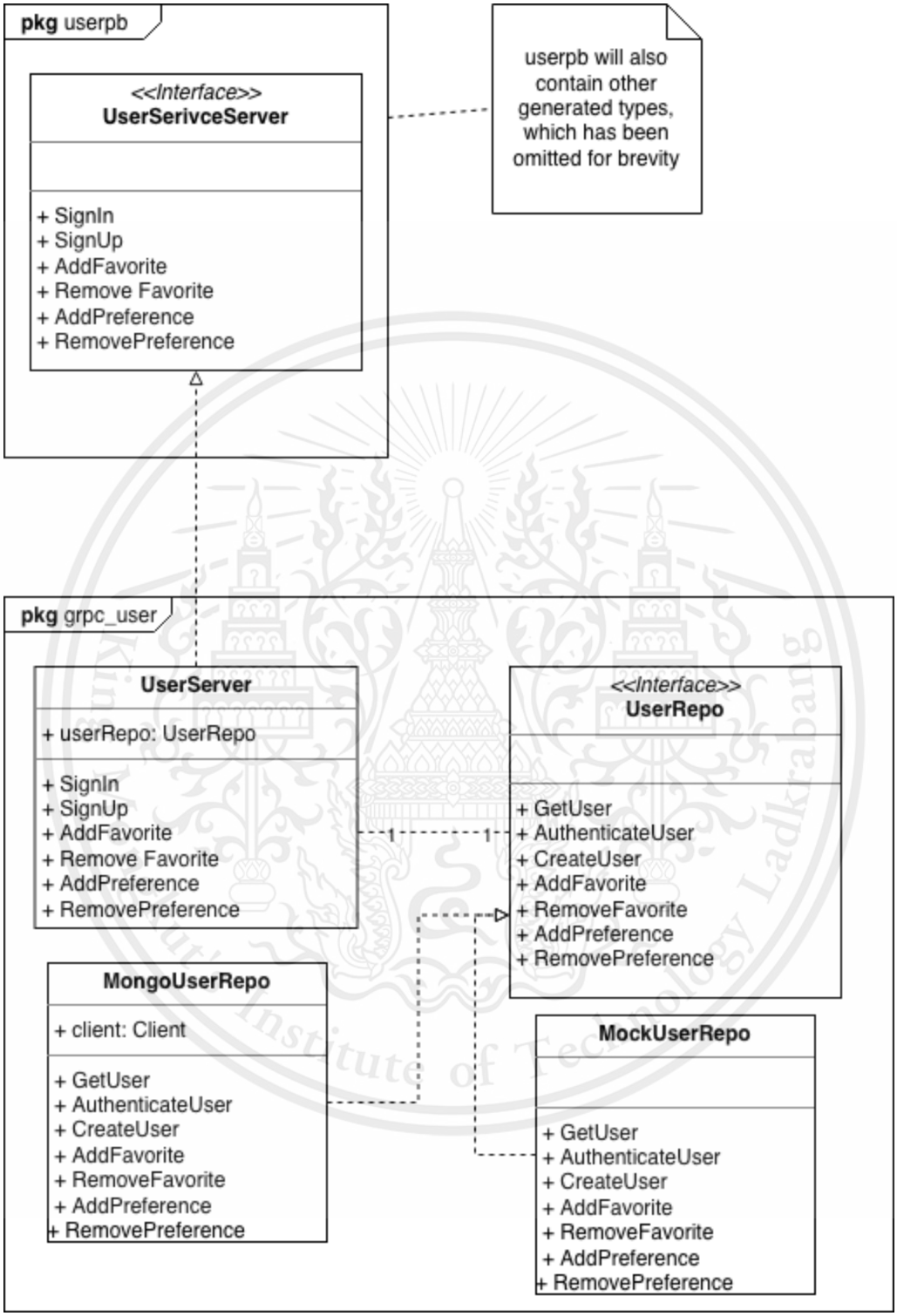


Figure 4.4 Backend grpc_user package

4.3.2 Customer Mobile Application

Figure 4.5 shows the package within the customer mobile client. The genproto package contains client stubs that have been generated with protocol buffers based from the same definition file as our backend gRPC services. The packages ending with "repository" act as the intermediate layer between our API and the user interface. The entity package contains data representation of the entities within our applications, it also serves as a common shared type between our user interface and repository. Our user interface contains Flutter widgets and BLoC (Business Logic Component) classes which contain state shared across the application.

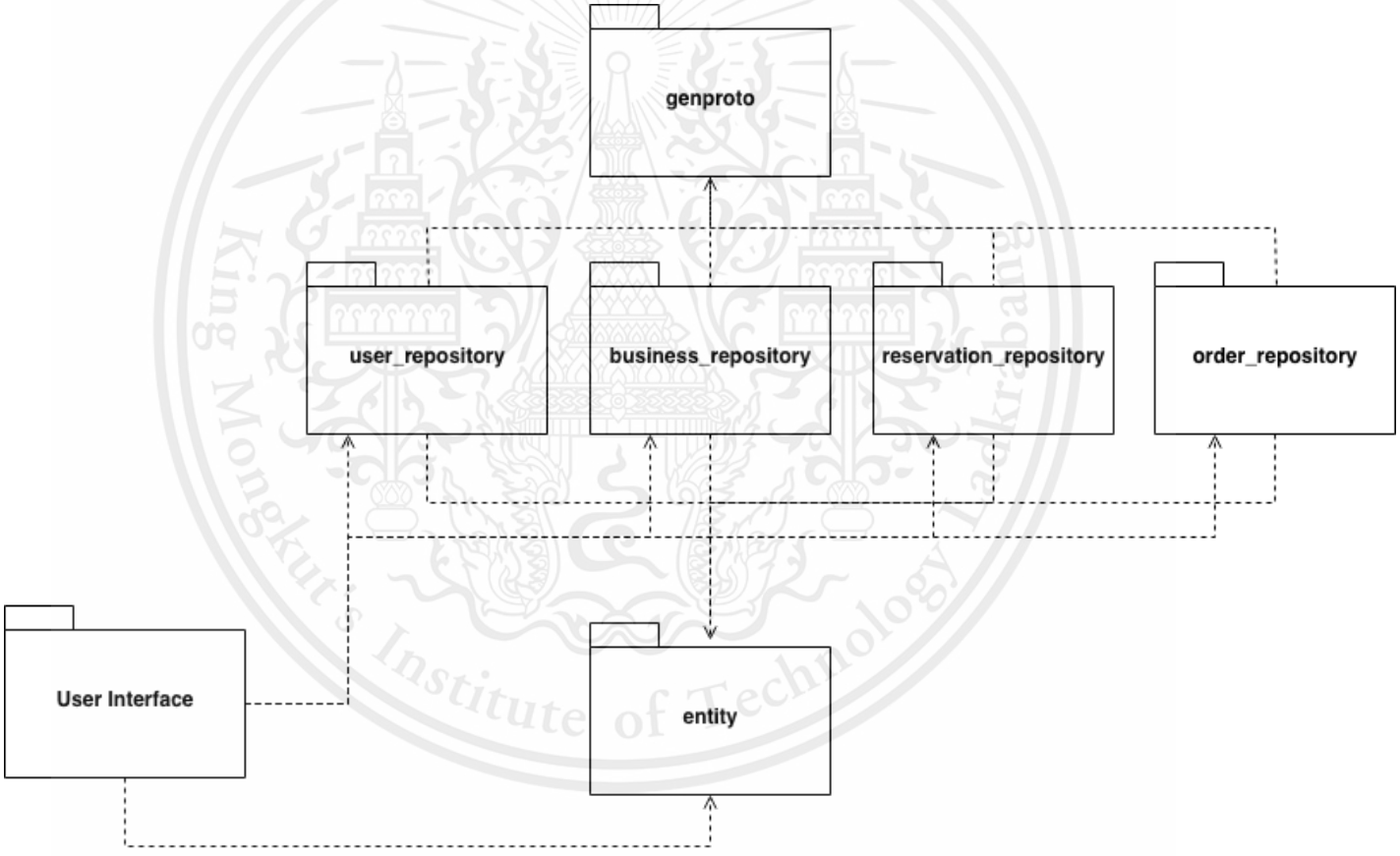


Figure 4.5 Customer Mobile packages

Figure 4.6 describes the classes within the entity package. It is similar to the entity package in the backend, the key difference being that the entity package of the mobile client only contains attributes from the perspective of the customer.

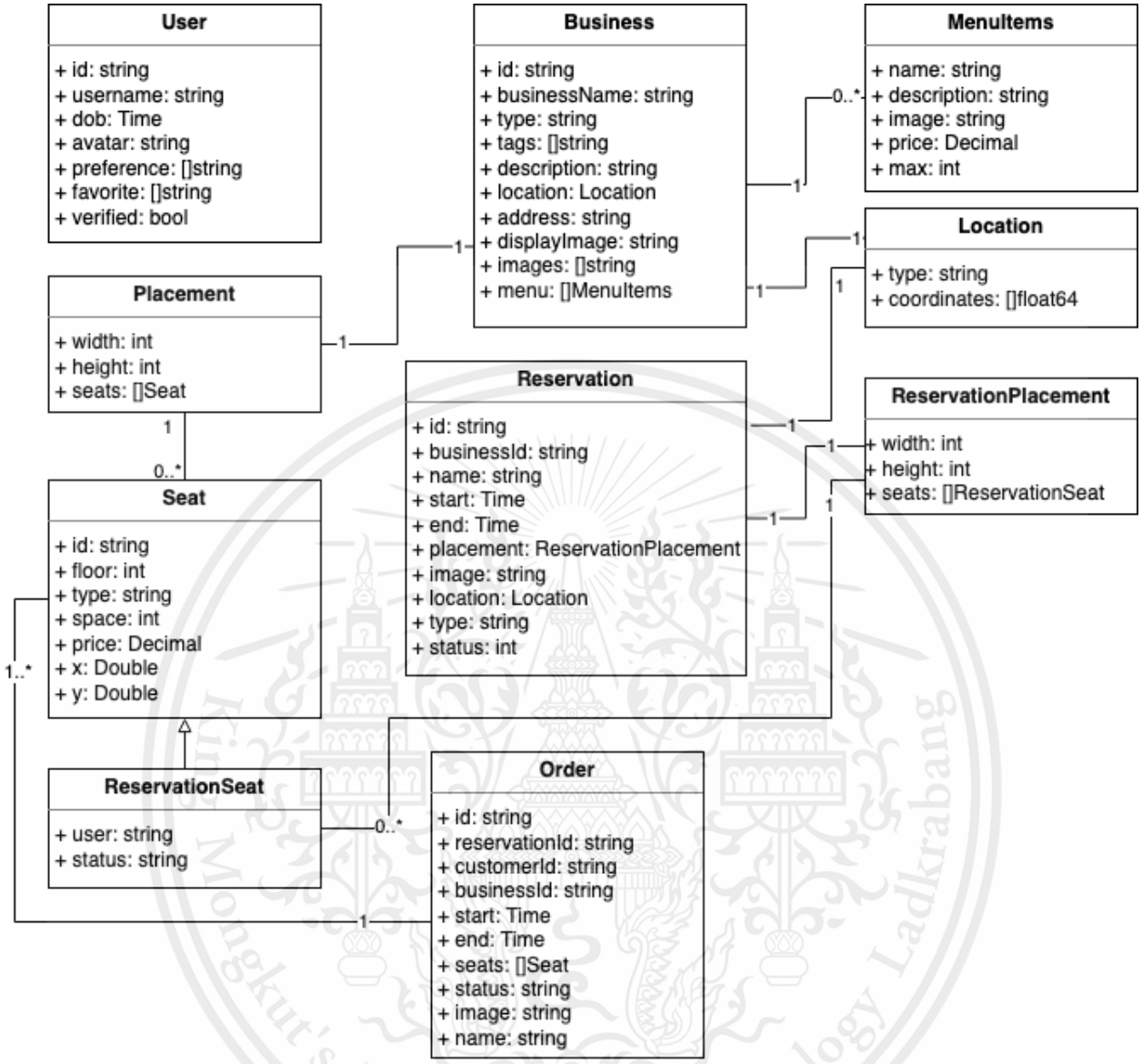


Figure 4.6 Customer Mobile entity package

Figure 4.7 shows the classes within the user_repository package, which is one of the "repository" packages. The package contains an interface of the repository which contains a gRPC client stub and methods to methods to call the endpoints. The package contains two implementations of these interfaces, a mock version for development and one which makes real gRPC calls to the backend.

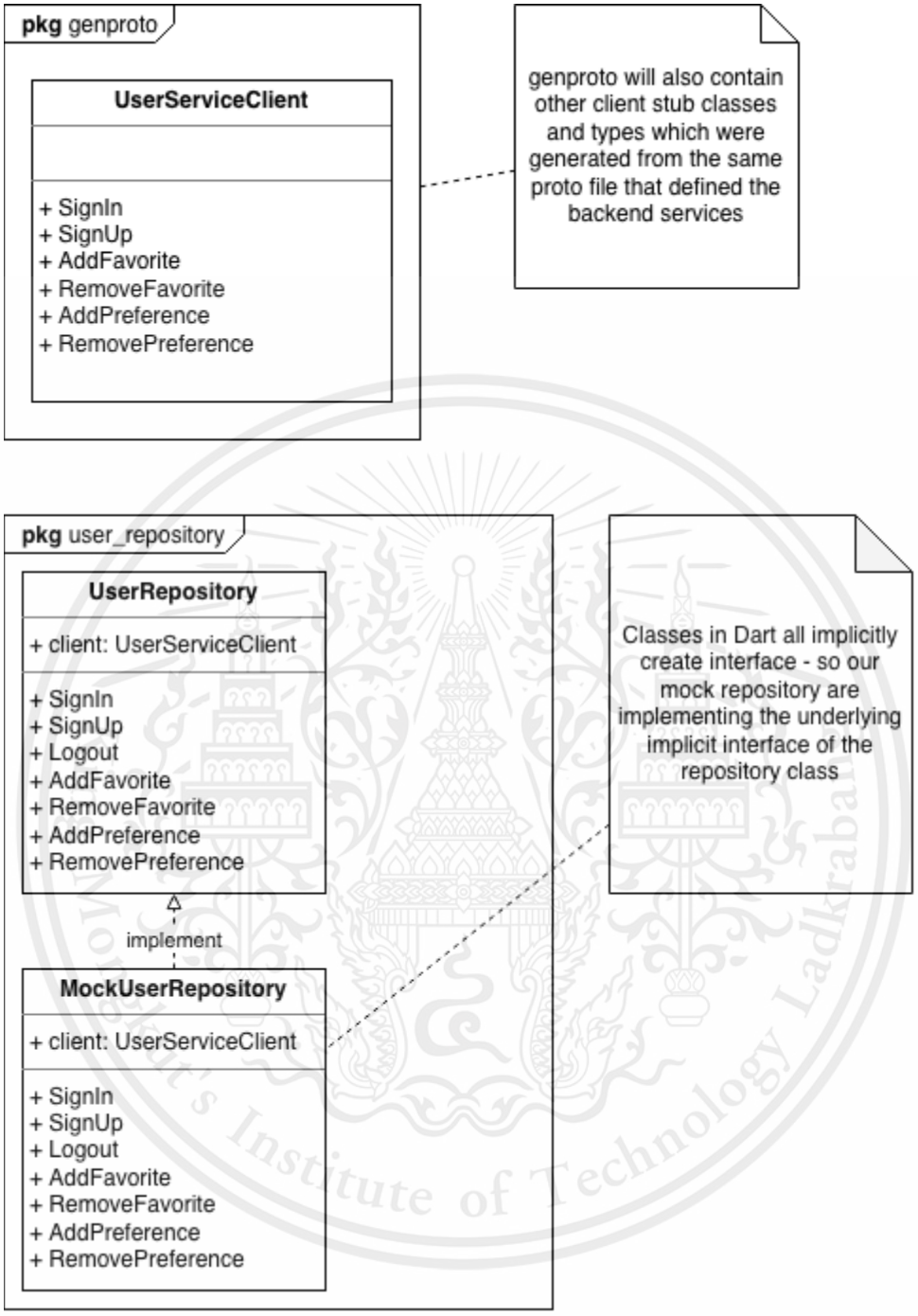


Figure 4.7 Customer Mobile user_repository package

Figure 4.8 shows the favorite BloC pattern used in our user interface package. The BloC pattern allows our application to share state and react to changes by providing them through the context of Flutter application. Note that the Business type seen in the package is from the entity package. The event type defines the events that we can add to via the Bloc add method (add to the Bloc's event stream), and the state is what is yielded as a reaction to the event that

occurs. Flutter widgets that rely on these data then listens to the changes that occur and re-renders as necessary.

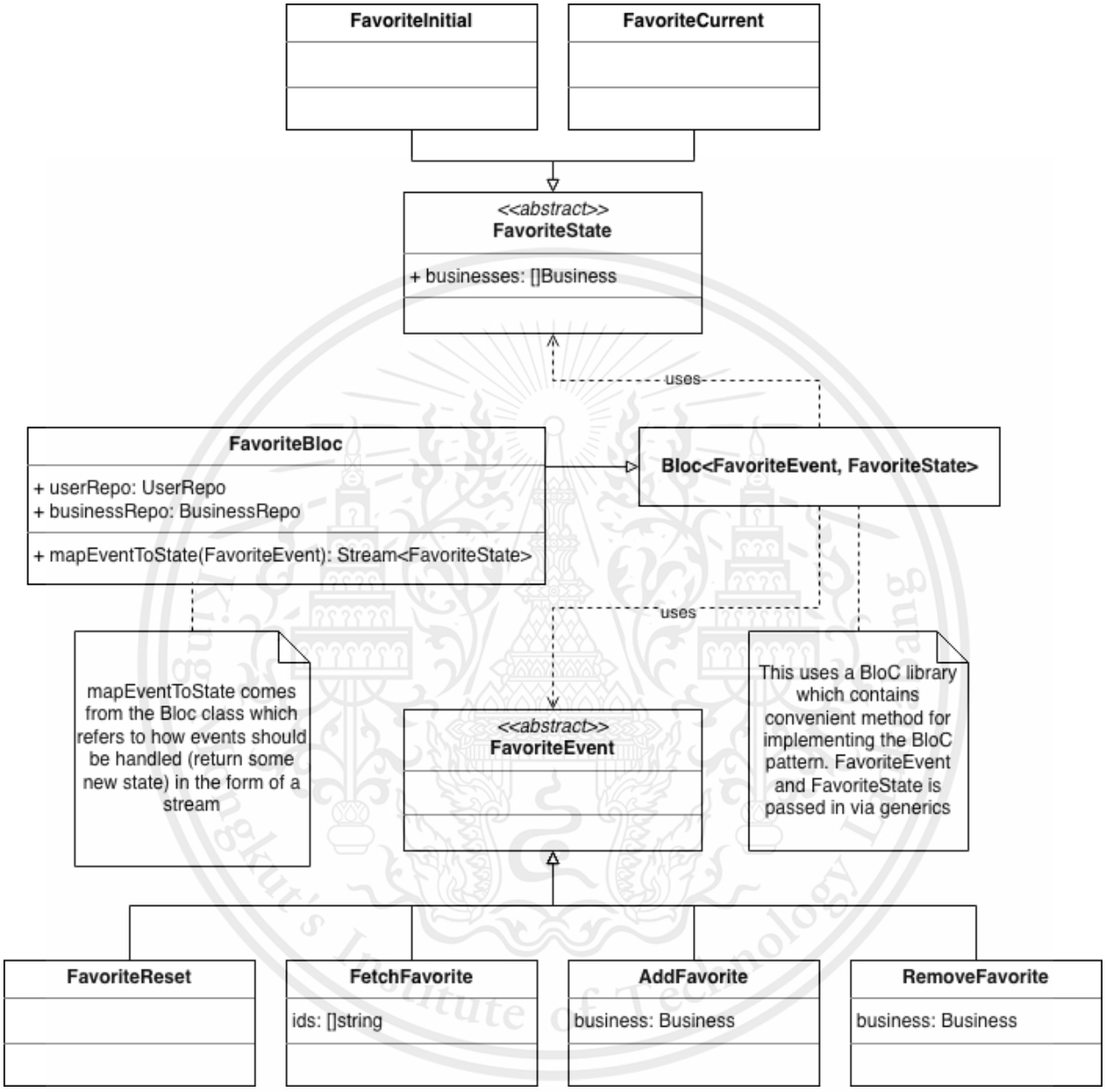


Figure 4.8 Favorite Bloc classes

4.3.3 Business Web Application

It is important to note that the Javascript type system does not include classes, but only syntactic sugar for making new objects using the prototypal pattern. The only reason the web platform has classes is to enforce object creation through the constructor with our own type checking validation.

Figure 4.9 describes the package within the frontend web application. It contains the entity package which contains structure used throughout the components within the system. The repository contains collections of functions which will make calls to the HTTP endpoints. For React state management, we simply use the native concept of lifting state upwards and use the context api for common state. For brevity we will not be including the structure of the components as this is often changing.

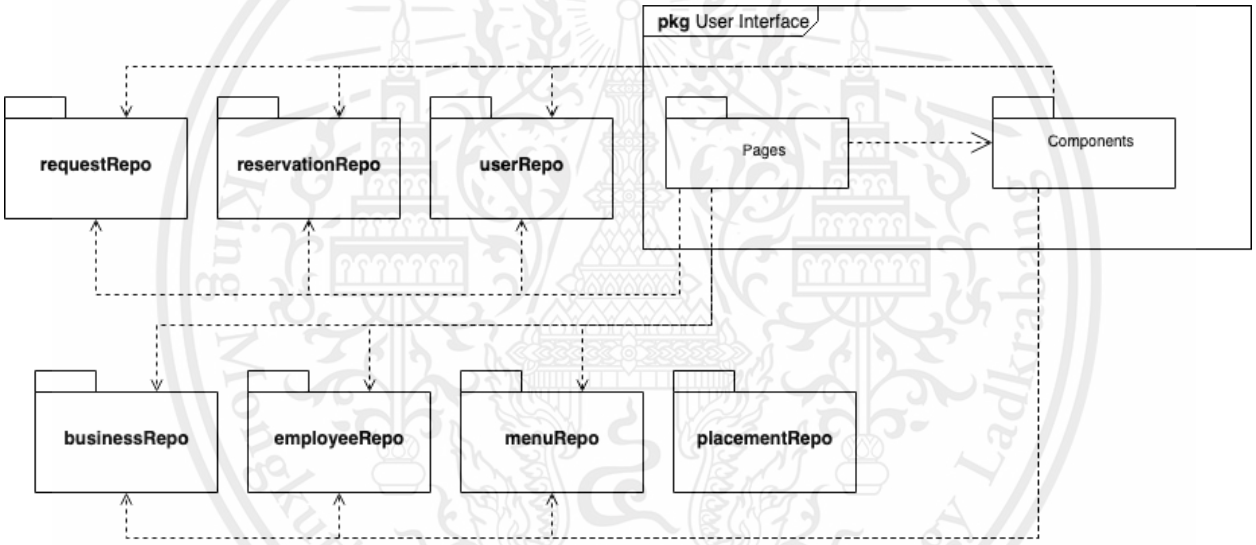


Figure 4.9 Web package

Figure 4.10 shows the class diagram for the entity package, it is similar to the previous platform entity package, but since this is for the web platform, a lot of the customer details are not needed as they are private to the customer.

4.3.4 Platform Admin Web Application

Since the web application for platform admin actor is quite simple and does not include many external API calls or complex state modification, no particular state management pattern is used. State management simply follows the principle of lifting state up to the most common ancestor of the component.

4.3.5 Employee Mobile Application

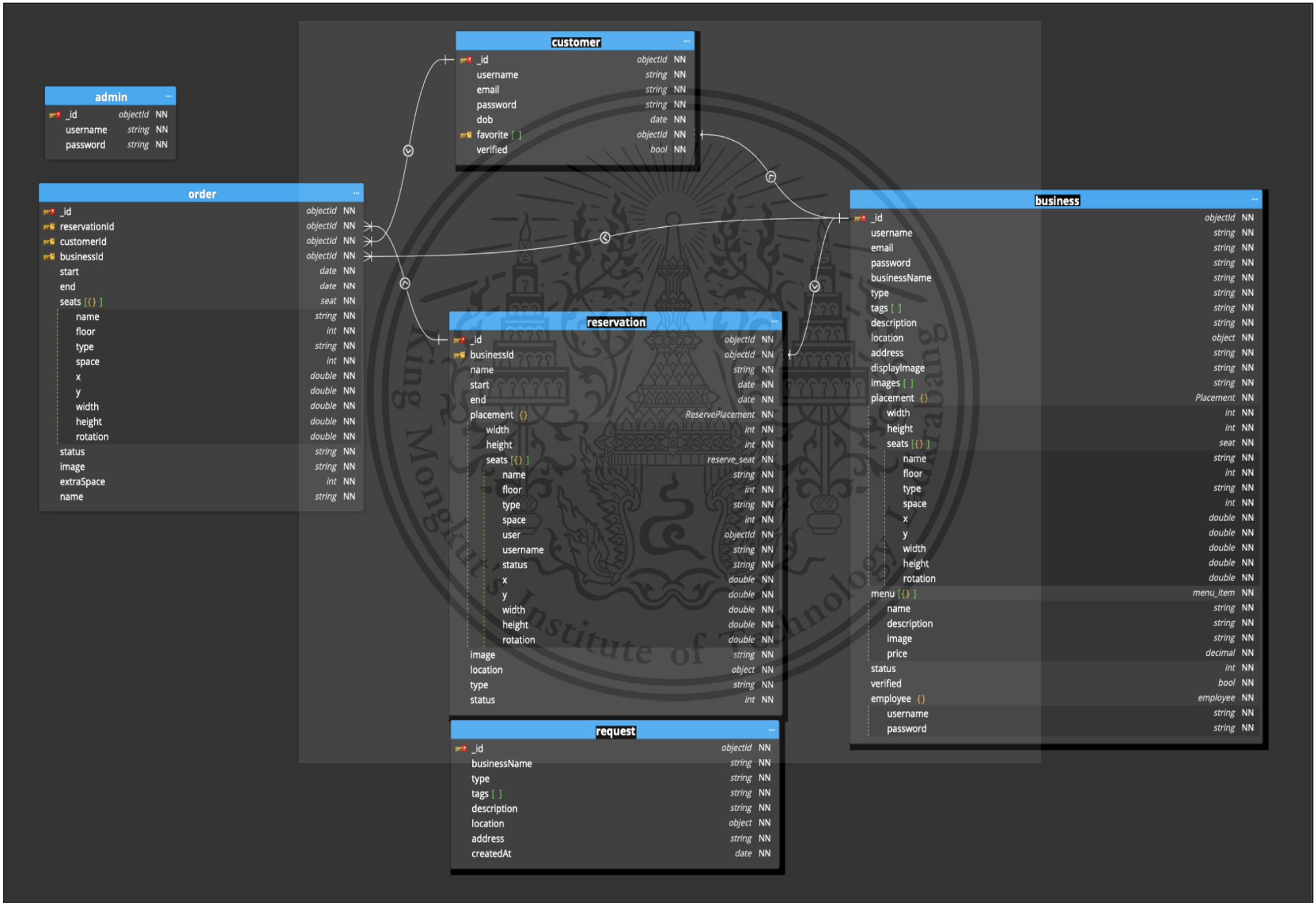
The employee mobile application is also much simpler when compared to the customer mobile application, and thus Flutter natvie state management alongside the use of a simple provider pattern for certain values to be exposed to the entire application.

4.3.6 Backend Observable Pattern

In our backend server we used the observable pattern to notify our mobile client of updates to the reservation that they are lookin at. We have a gRPC streaming endpoint. When the client calls the endpoint the server will register an observer to a map structure based on the particular reservation identification number (we use the Go's programming language concept of channel and communication sequential process to achieve this). Whenever a change happens the streaming gRPC handler is notified via a message passed through Go channel created during the previous step. The gRPC streaming endpoint then does the necessary encoding and sends the updates to the mobile client. The mobile client can then update the placement view.

4.4 Database Schema

Figure 4.11 is the system's current database. There are a total of 6 collections: admin, customer, business, request, reservation, and order. The admin collection contains information on our admin platform user. The customer collection contains information on the user who will be making reservations through the mobile application. The business collection contains the business account and the business itself. The request collection contains a list of information change requests from businesses that are pending approval. The reservation collection contains information on each reservation schedule (eg. time, available seats etc.). The order collection contains the individual reservation made by the customer, and can be seen as the reservation record of the customer. Note that some details of the database schema are omitted for brevity.


admin
_id objectId NN
username string NN
password string NN
customer
_id objectId NN
username string NN
email string NN
password string NN
dob date NN
favorite [ ] objectId NN
verified bool NN
order
_id objectId NN
reservationId objectId NN
customerId objectId NN
businessId objectId NN
start date NN
end date NN
seats [{}] seat NN
name string NN
floor int NN
type string NN
space int NN
x double NN
y double NN
width double NN
height double NN
rotation double NN
status string NN
image string NN
extraSpace int NN
name string NN
reservation
_id objectId NN
businessId objectId NN
name string NN
start date NN
end date NN
placement {} ReservePlacement NN
width int NN
height int NN
seats [{}] reserve_seat NN
name string NN
floor int NN
type string NN
space int NN
user objectId NN
username string NN
status string NN
x double NN
y double NN
width double NN
height double NN
rotation double NN
image string NN
location object NN
type string NN
status int NN
request
_id objectId NN
businessName string NN
type string NN
tags [ ] string NN
description string NN
location object NN
address string NN
createdAt date NN
business
_id objectId NN
username string NN
email string NN
password string NN
businessName string NN
type string NN
tags [ ] string NN
description string NN
location object NN
address string NN
displayImage string NN
images [ ] string NN
placement {} Placement NN
width int NN
height int NN
seats [{}] seat NN
name string NN
floor int NN
type string NN
space int NN
x double NN
y double NN
width double NN
height double NN
rotation double NN
menu [{}] menu_item NN
name string NN
description string NN
image string NN
price decimal NN
status int NN
verified bool NN
employee {} employee NN
username string NN
password string NN

Chapter 5
Development

This chapter presents and describes tools that are used to develop this project and how we implement the project. We use different tools for each part such as web server, web and mobile application.

5.1 Development tools

5.1.1 General

Visual Studio Code

Visual studio code is a text editor program that is used in our front-end project development. It has plugins and support for several tools we use in development. This includes web preview based on our OpenAPI specification files, protocol buffer syntax highlighting etc.

5.1.2 Backend Server

gRPC Protocol Buffer plugins

We define our services using Protocol Buffer. gRPC code generation isn't included with Protocol Buffer by default. If we want to generate gPRC server and client stubs we need to use gRPC plugins. For our project we use two officially supported gRPC plugins, grpc-go and grpc-dart.

DeepMap's oapi-codegen

For our backend server we define and document our HTTP API endpoints

by following the OpenAPI specification. We then use DeepMap's oapi-codegen to generate request/response structure, server interface and other such boilerplate in the Go programming language.

Echo

Our backend server uses the Go programming language and the Echo framework. Echo is a minimal web framework that is built on top of Go's standard library. DeepMap's oapi-codegen uses the Echo framework by default.

5.1.3 Mobile Development

Flutter

Flutter is an open source mobile software development kit(SDK). It allows developers to create a native Android and iOS application with the same code base.

Flutter apps are written in the Dart language and make use of many of the language's more advanced features. On Windows, macOS, and Linux Flutter runs in the Dart virtual machine, which features a just-in-time execution engine. While writing and debugging an app, Flutter uses Just In Time compilation, allowing for "hot reload", with which modifications to source files can be injected into a running application. Release versions of Flutter apps are compiled with ahead-of-time (AOT) compilation on both Android and iOS, making Flutter's high performance on mobile devices possible.

bloc

Flutter bloc is a library that, as the name suggests, helps developers implement the BLoC pattern. It is built on top of RxDart, which is built on top of Dart's Stream class in the official async library. It contains helper builder widgets allowing widget rebuilding through streams, and accessing state/repository through Flutter's native context.

5.1.4 Web Development

React

First appearing in 1995, Javascript is as of today, the most popular programming language for web development. JavaScript is a client-side

programming language that allows developers to do web application development and make dynamic and interactive web pages by implementing custom client-side scripts.

Developers can also utilize cross platform runtime engines like Node.js to write server-side code in JavaScript. These allow developers to create multi-platforms web applications by combining JavaScript, HTML5, and CSS3. Javascript also supports multiple Frameworks such as AngularJS, ReactJS, Vue Js, Ruby on rail, and NodeJS.

For this project, we choose to use React, a javascript framework. React allows a to buy front end web pages by breaking down complex HTML into smaller components and utilizing it in various parts of the project which in turn reduce redundancy of the coding. It also allows fast updating and rendering of ideal components when data changes.

Next.js

Next.js is an open source framework built on top of React. Most existing React components can work with Next.js. Like most frameworks it comes with framework specific functionality. It handles things such as SSG, SSR, URL based routing, image optimization, code splitting and many more features. All of this is done without any additional configuration.

Flutter

Flutter also supports web application development and our platform admin web application also uses Flutter for the web application. Refer to section 5.1.3 for more information on Flutter.

5.2 Development method

The system consists of several web and mobile applications. The business web application allows the managers to create menu, placement and create a reservation schedule period.The customer mobile application is used by a customer to make a reservation or view information on the business. The employee mobile application for employees to verify tickets. Lastly, the platform admin web application to manage the businesses on the platform themselves.

5.2.1 Requirement Analysis

The requirement and understanding of this project is being done in the beginning project. We commit research on the common reservation systems

and make our application design based on them. Functional and non-functional requirements are also being made during this phase. However, there have been many changes to the requirement during the duration of this project, an example of this is the addition of the platform admin web application and the removal of the payment feature.

5.2.2 System Design

The overall system design takes place during the first semester and although there have been some changes to the system, the core components and implementation of the system remains the same as it was first decided. The overall design of the system is kept very simple and it is reflected upon the fact that the project only includes one monolithic backend server.

5.2.3 Backend Server Development

Our approach to the backend is to first discuss the API definition before implementation. We use gRPC (Protocol Buffer file format) and OpenAPI (YAML) as a method of documenting our available API to the mobile and web application. In addition, we use code generation based on these files, ensuring that when we implement the services, the API definition stays consistent and is documented along the way.

5.2.4 Mobile development

We have started implementing the layout of the mobile application according to our design. Our approach requires that we have discussed the API definition before proceeding with the implementation, this is to minimize code refactoring in the future. We start by writing the repository package that will make calls to the backend services, including the mock repository. Then we add widgets and state to incorporate the new part of the application we are implementing.

For mobile development we use dependency injection during development to use the mock repository. The mock repository simply returns some predetermined data when it is used. This method meant that we did not have to have our backend server running or implemented at all. For each section of the application, we develop them in short cycles, in parallel with defining the API service definition.

5.2.5 Web development

Our web application takes a similar approach to our mobile development. It uses mock repository and dependency injection the same way our mobile client development method does. Components are left as empty placeholders until we are ready to implement them. We implement each component in small cycles at a time.

Chapter 6
Results

6.1 Customer Mobile Application

This section contains screenshots of each page of the customer mobile application. Note that the pictures presented here are mock data. The placeholder images are retrieved from a placeholder image web service and the location data are all made up.

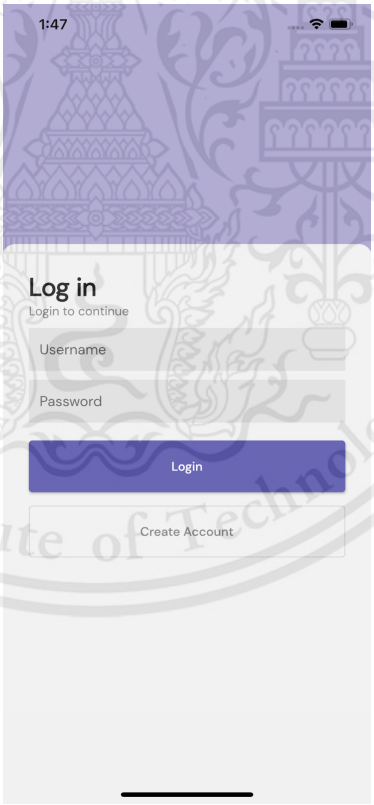


Figure 6.1 Customer Mobile Login Page

Figure 6.1 shows the login page for the mobile client, which is where the customer will authenticate themselves. This is also the first page that customers will see.

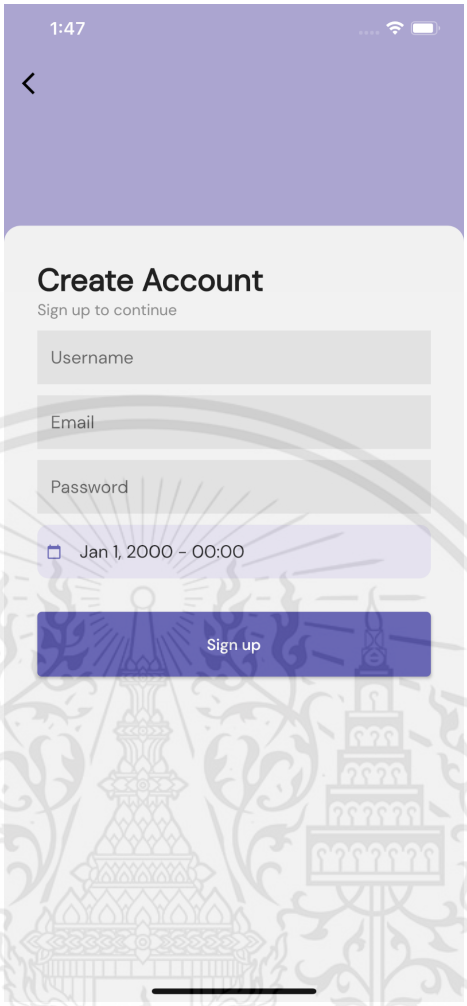


Figure 6.2 Customer Mobile Sign Up Page

Figure 6.2 shows the sign up page which allows customers to create an account to use the customer mobile application.

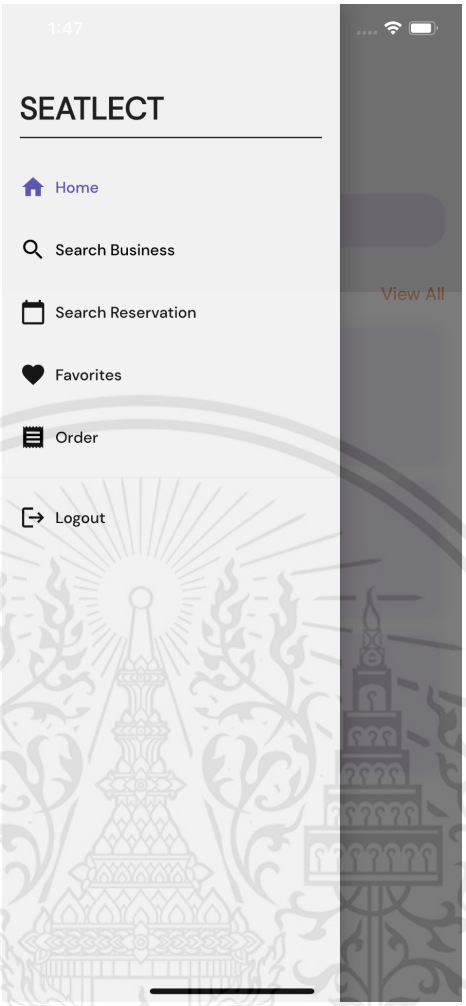


Figure 6.3 Customer Navigation Drawer

Figure 6.3 shows the navigation drawer which allows customers to navigate the customer mobile application.

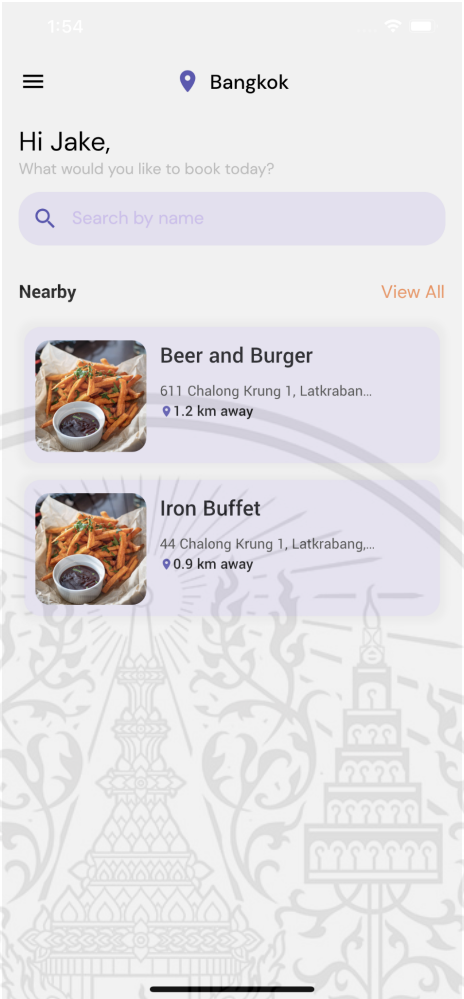


Figure 6.4 Customer Mobile Home Page

Figure 6.4 shows the home page of the mobile application. The application bar shows the current location of the user (if the user has given permission to the application). The search bar receives text input and is a shortcut to the search page which will use the input data as arguments for searching. The "Nearby" will show nearby businesses of "restaurant" type for the customers, given that they have given the application permission to use their location.

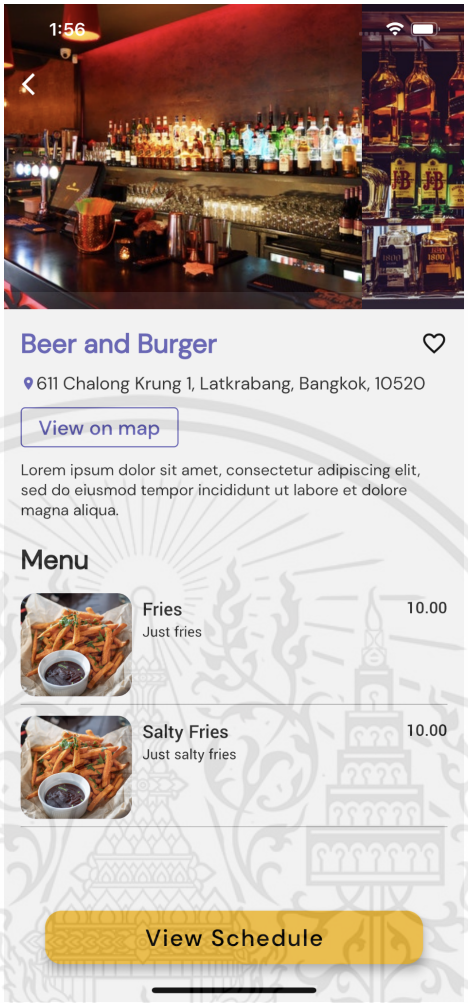


Figure 6.5 Customer Mobile Business Detail Page

Figure 6.5 is the detailed individual page corresponding to the business. The page contains the name, location, description, menu, the favorite's button, and the reserve button. Tapping the heart icon will either add the business to the user's favorite list or remove them. Tapping the "view schedule" will allow the customers to query the reservation time slot made by the business. Tapping "view on map" will allow the customers to see the location of the business on the map.

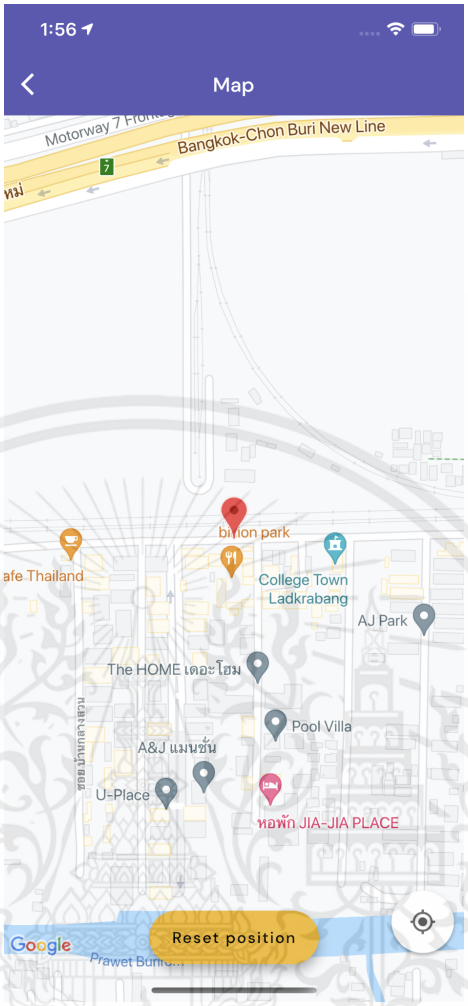


Figure 6.6 Customer Mobile Business Location Page

Figure 6.6 shows the page after tapping the “view on map” button in the Business Detail Page. It displays the location of the business with a marker.

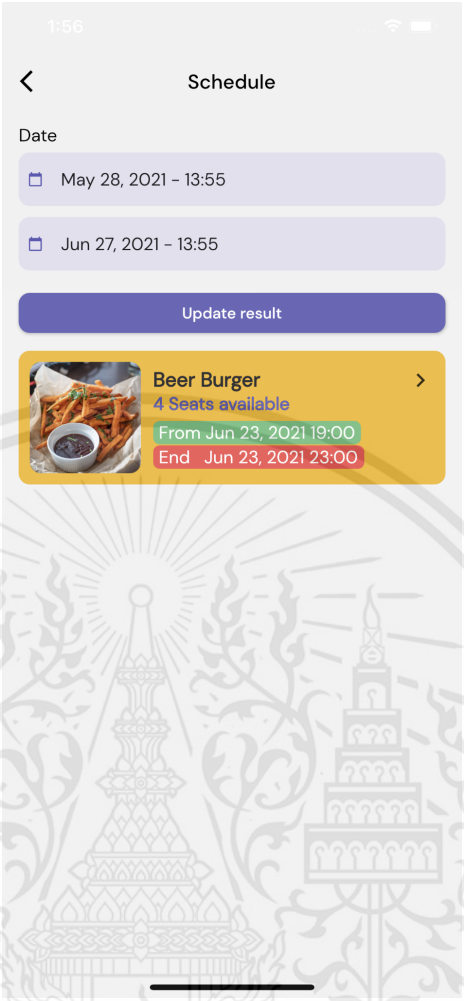


Figure 6.7 Customer Mobile View Schedule Page

Figure 6.7 shows the page after tapping “view schedule” from the Business Detail Page. It displays the reservation time slot made by the business given a date range which can be modified at the top of the page.

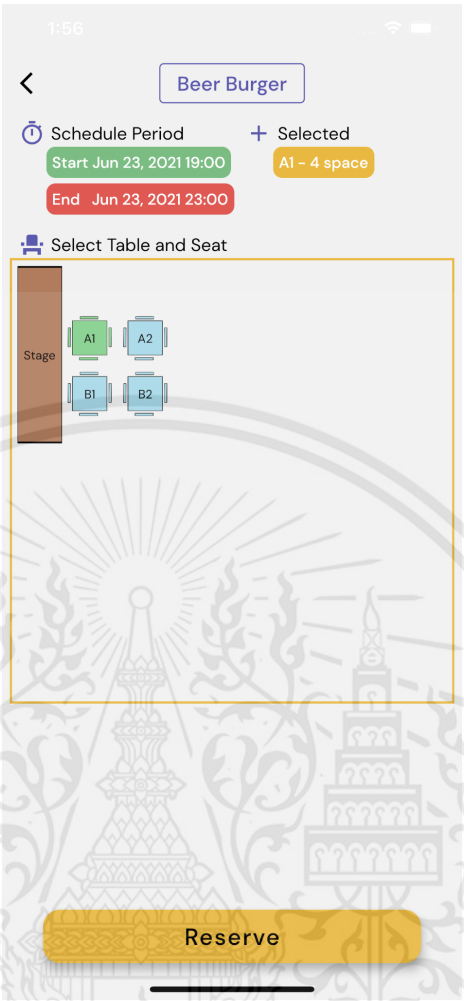


Figure 6.8 Customer Mobile Reservation Page

Figure 6.8 shows the page where the customer will select and confirm seat reservation. Green highlights on the seats indicate currently selected seats, blue indicates seats available for selection, and red indicate unavailable seats.

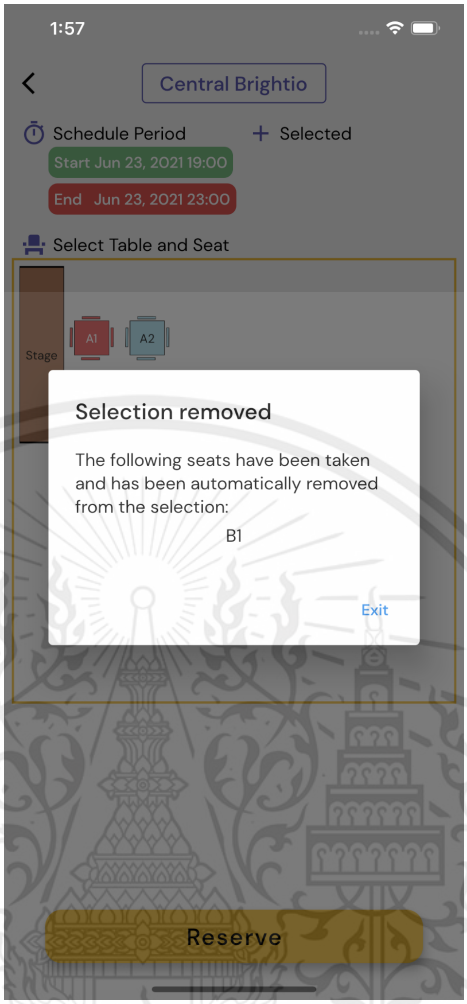


Figure 6.9 Customer Mobile Seats Update

Figure 6.9 shows an example of an alert that the customer will receive if one of their currently selected seats have been reserved before they have confirmed their reservation.

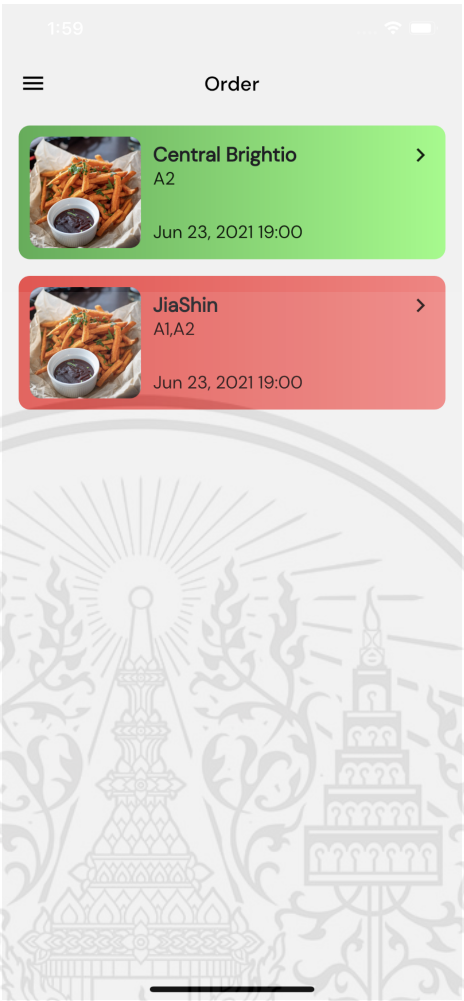


Figure 6.10 Customer Mobile Order Page

Figure 6.10 shows the list of "orders" (or more intuitively known as a reservation) made by the customers. The green indicates available tickets that has not been used, and the red indicates an order where the business themselves have cancelled the reservation time slot.

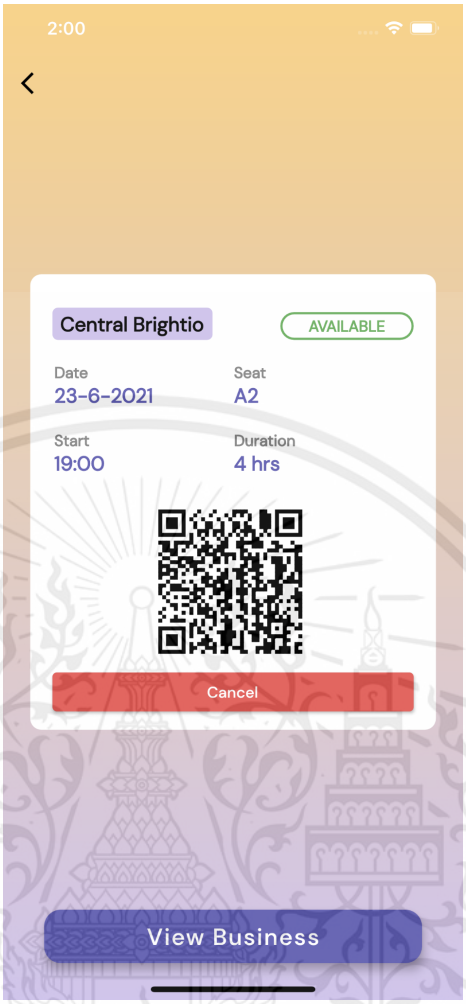


Figure 6.11 Customer Mobile Order Detail Page

Figure 6.11 shows the detailed view of an order. It contains the information of the order and a QR code to be scanned by the employee mobile application. The customer can also tap cancel to cancel the order.

Figure 6.12 Customer Mobile Search Reservation Page

Figure 6.12 shows the search option for searching for a list of reservations. Customers can search with a name, location (default to middle of Bangkok), type, and date range.

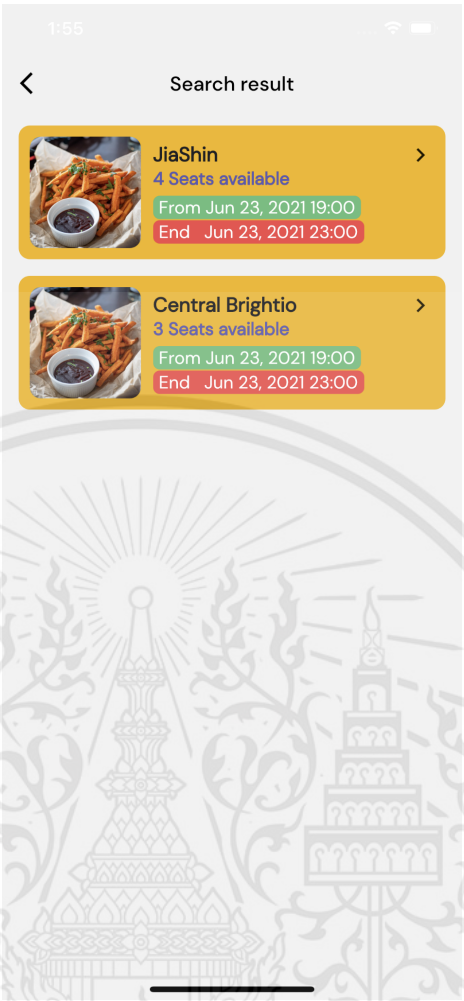


Figure 6.13 Customer Mobile Search Reservation Result

Figure 6.13 shows the result of the Search Reservation Page. It shows a list of reservation time slots from potentially several businesses fitting the criteria provided in the Search Reservation Page.

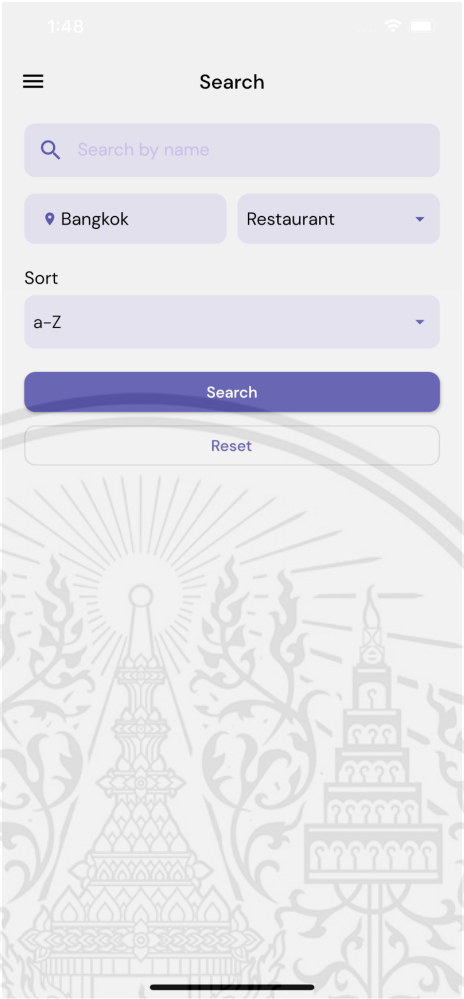


Figure 6.14 Customer Mobile Search Business Page

Figure 6.14 shows the search option for searching for a list of businesses. The difference between Search Business and Search Reservation is that Search Business doesn't do date range and will mostly be used when a customer is looking for a particular business (and might have not decided on a particular date yet).

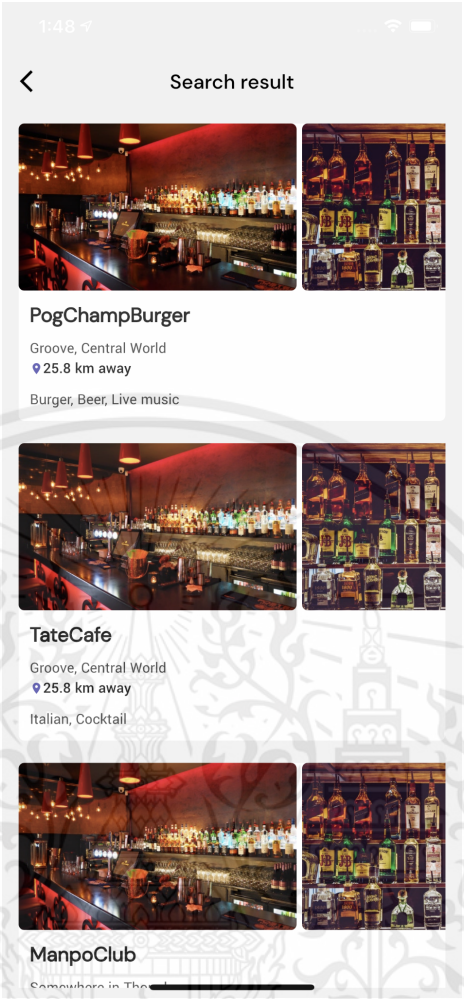


Figure 6.15 Customer Mobile Search Business Result

Figure 6.15 shows results from the Search Business Page, it shows a list of businesses that matches the search criteria. Tapping on the cards will show the Detailed Business Page.

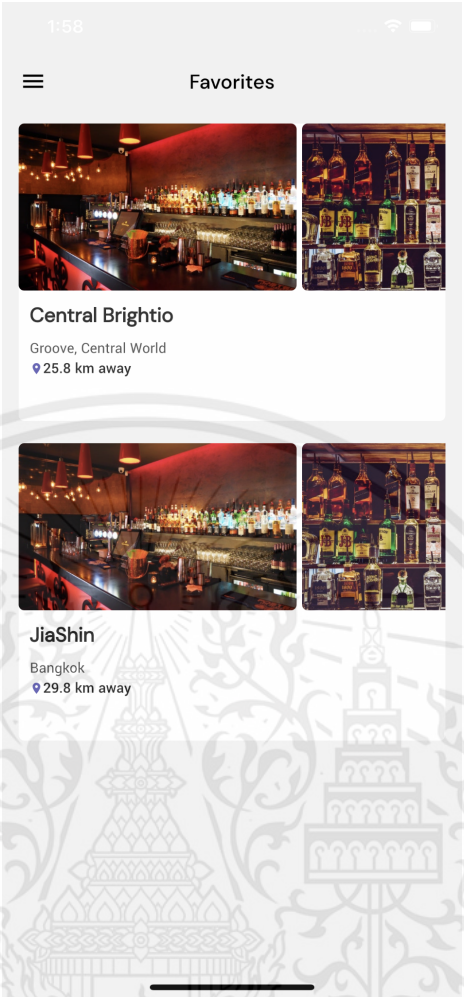


Figure 6.16 Customer Mobile Favorites Page

Figure 6.16 shows a list of businesses which have been favorited (via the heart icon) of the customer.

6.2 Business Web Application

This section contains screenshots of each page in the business web application and description of each page. Note that the pictures presented here are mock data. The placeholder images are retrieved from a placeholder image web service and the location data are all made up.

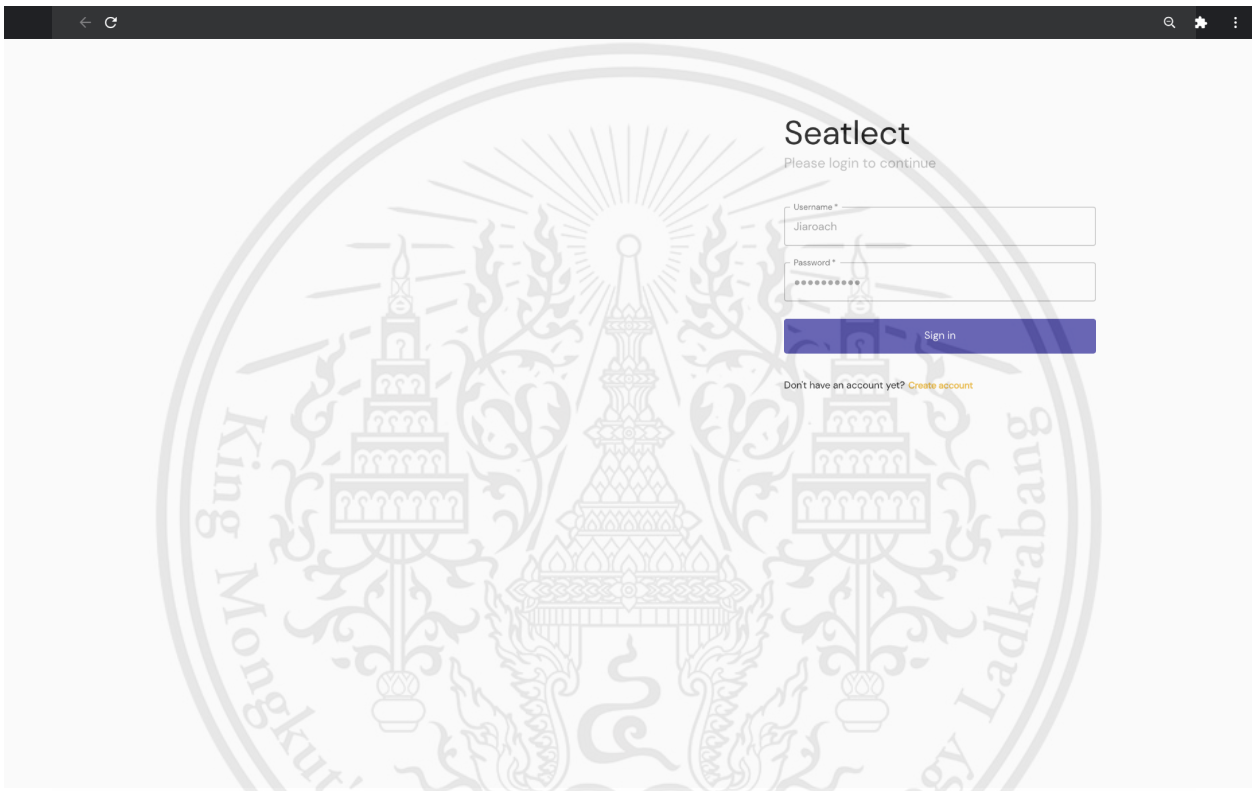


Figure 6.17 Business Web Login Page

Figure 6.17 shows the login page for the Business web application.

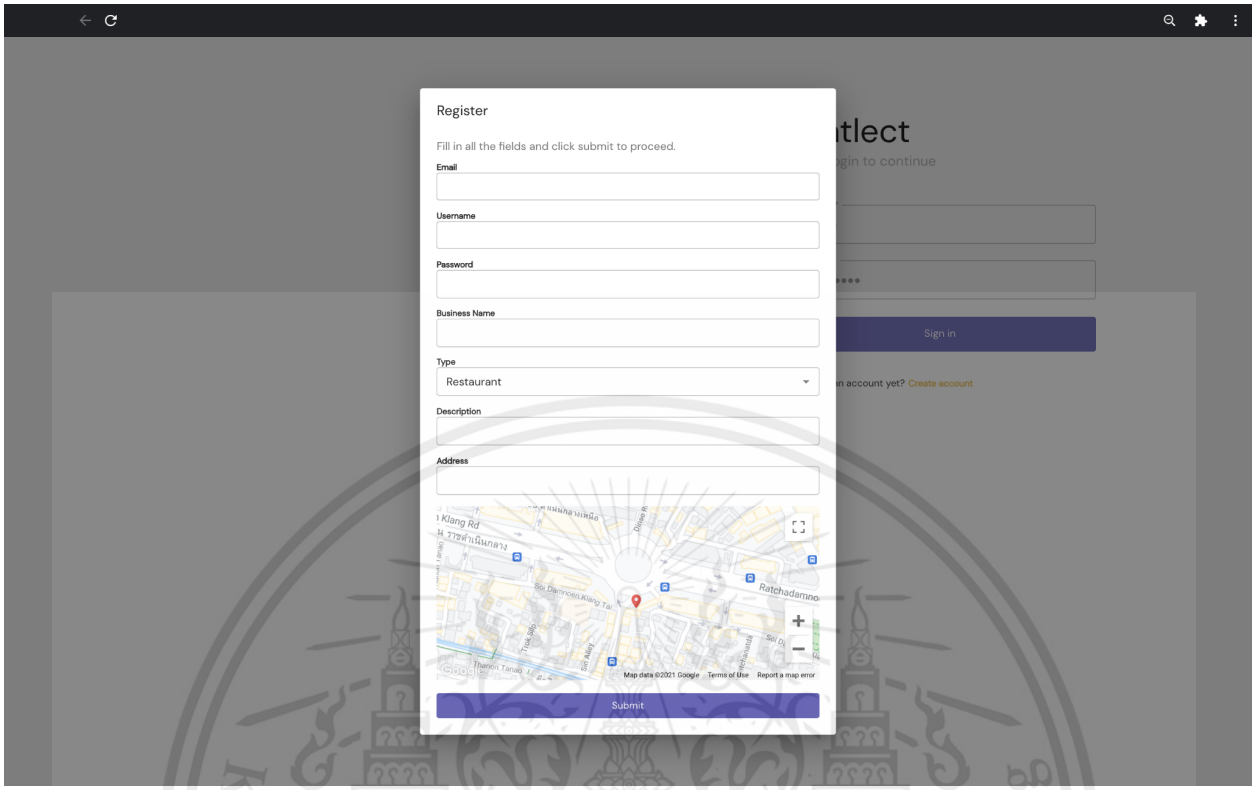


Figure 6.18 Business Web Sign Up Page

Figure 6.18 shows the sign up page, where businesses can submit their business to be added to the platform (they will be able to login using their credentials once their business registration request has been approved).

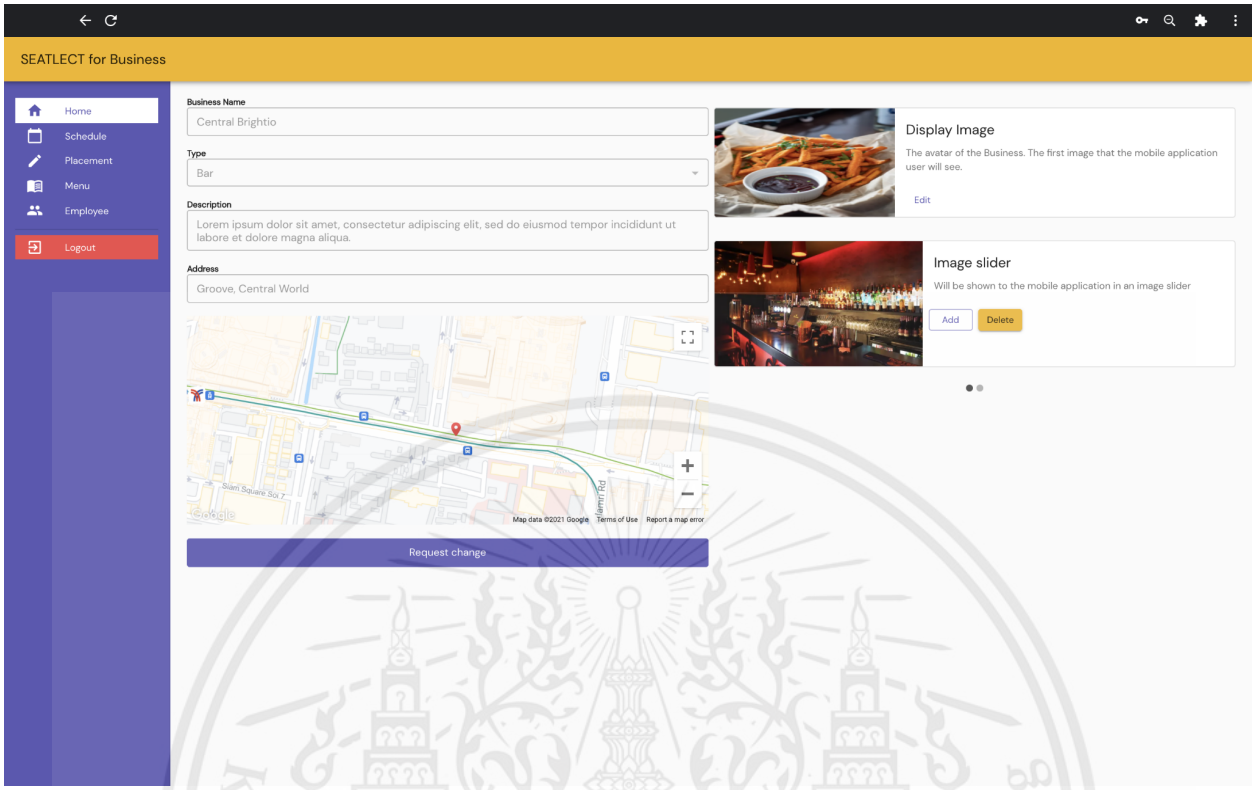


Figure 6.19 Business Web Home Page

Figure 6.19 shows the home page, it displays the business information. Images can be changed on the right side of the page (a modal to choose an image will appear) and tapping “request change” will open a modal where business managers can enter updated information that they want our platform admin to approve (information won’t be updated before approval).

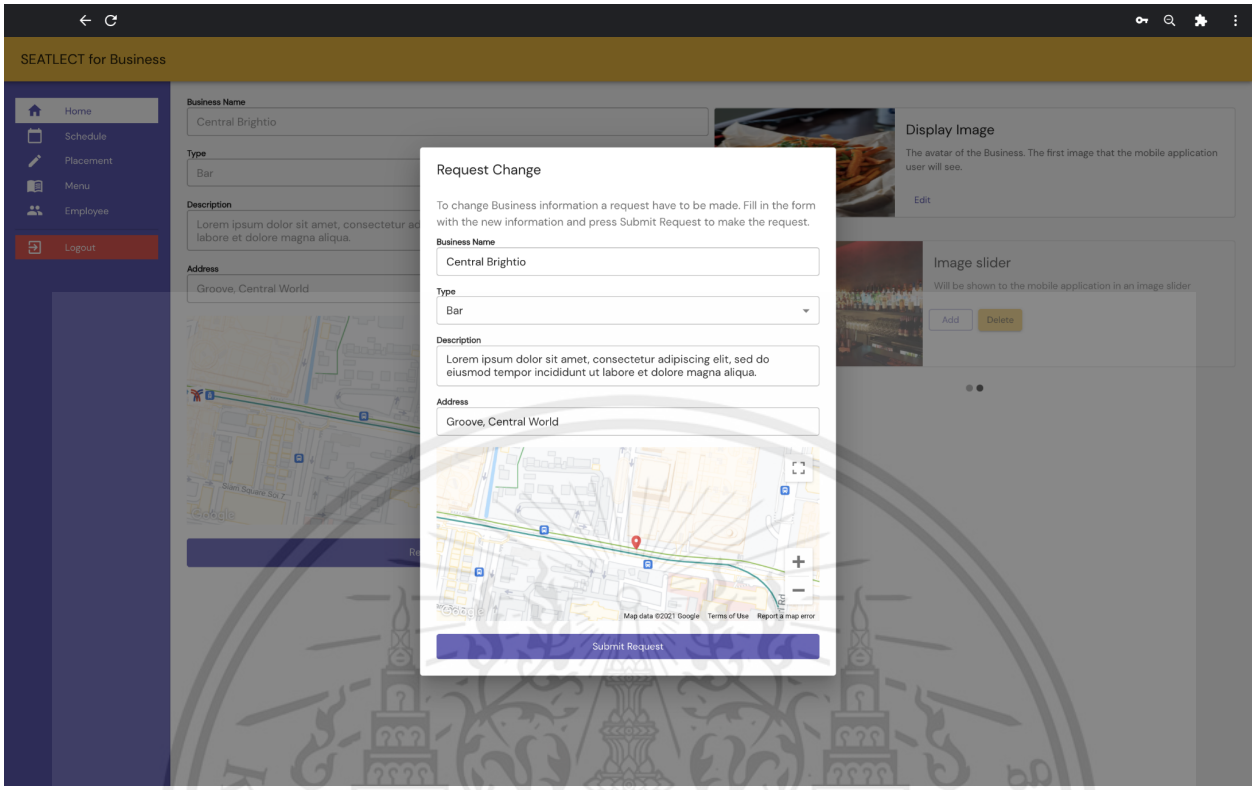


Figure 6.20 Business Web Request Change Modal

Figure 6.20 shows the request change modal, where the business managers can enter new information and send a request to the platform admin for approval.

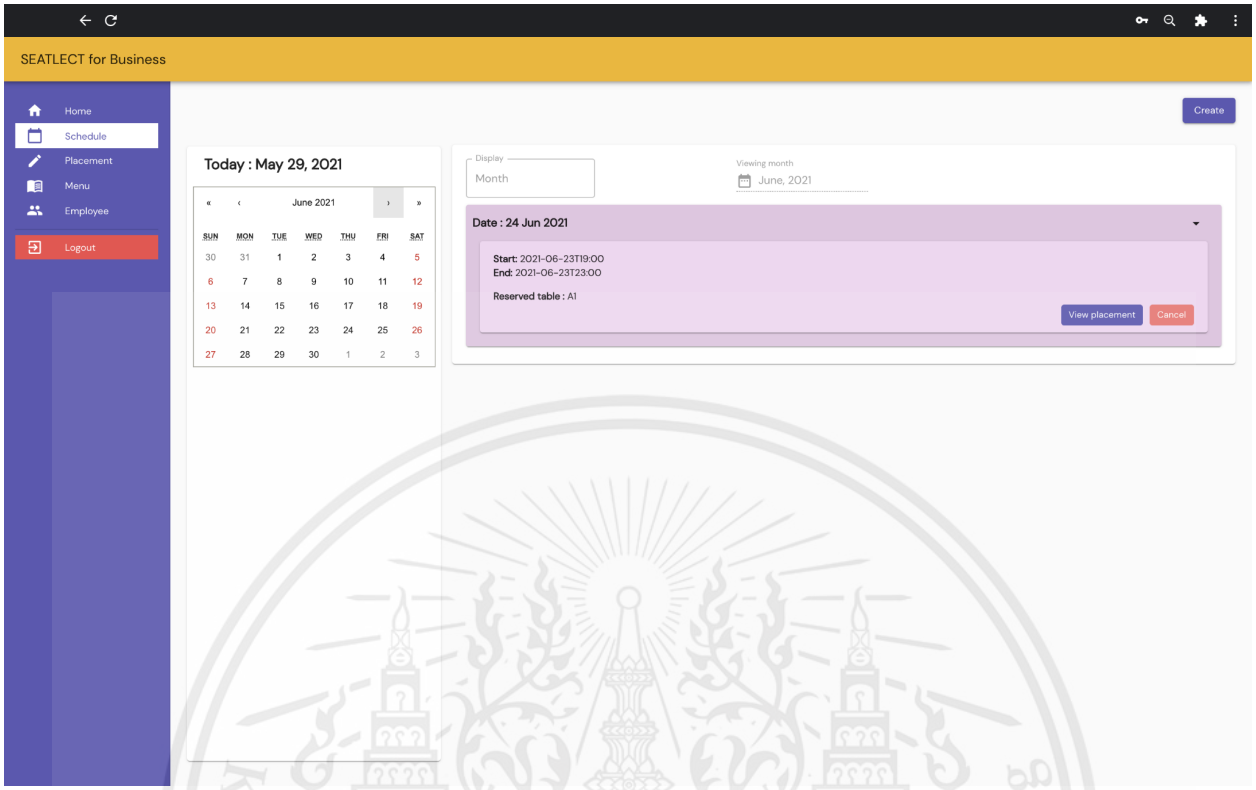


Figure 6.21 Business Web Schedule Page

Figure 6.21 shows the schedule page, the page displays a list of reservation time slots by month and allows the business managers to cancel or create reservation time slots. Tapping “view placement” will also visually display the reservation placement of the particular time slot.

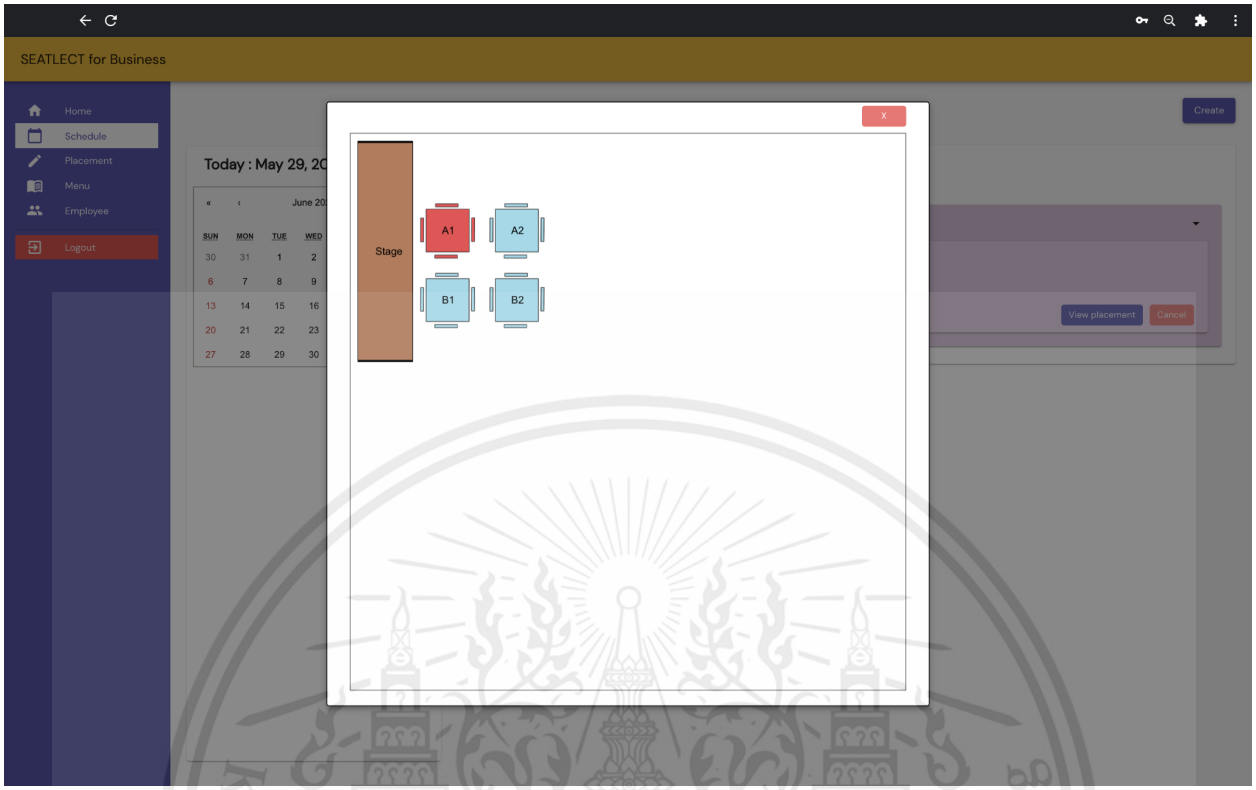


Figure 6.22 Business Web Schedule Reservation Placement

Figure 6.22 shows the reservation placement view of a particular reservation time slot that can be accessed by tapping the “view placement” button of a reservation time slot from the schedule page. Red color indicates that someone has made a reservation for that seat.

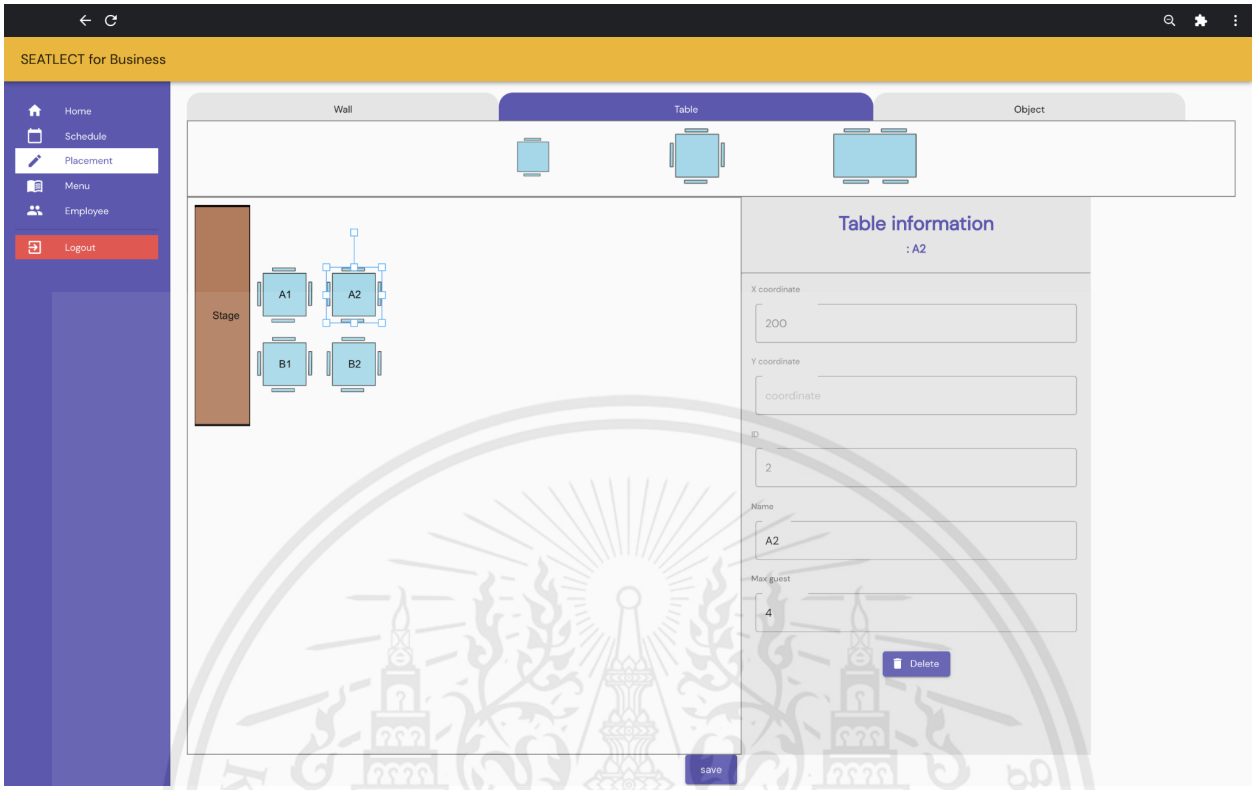


Figure 6.23 Business Web Placement Page

Figure 6.23 shows the placement view of the business, this allows the business managers to create a placement view that represents their business. Updates to the placement will be shown only in the newly created reservation time slot after the updates. Old reservation time slot placement will not be affected.

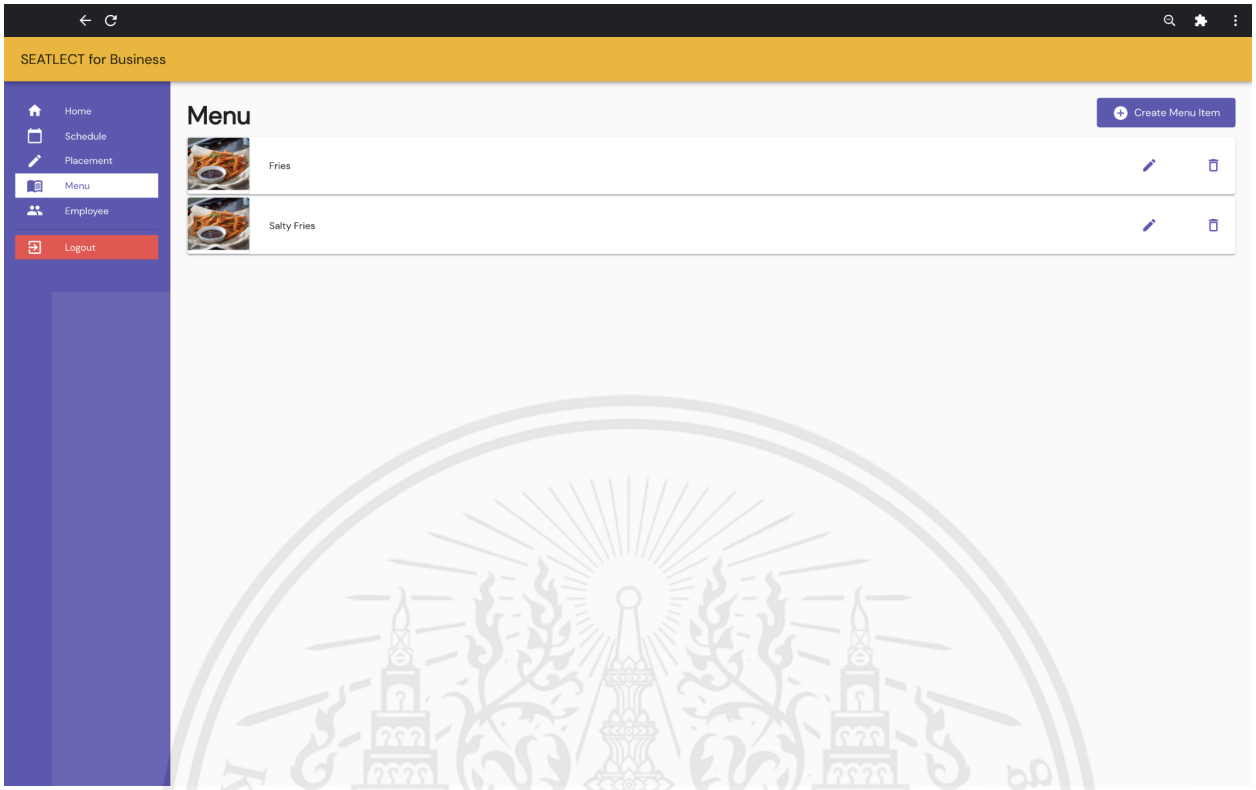


Figure 6.24 Business Web Menu Page

Figure 6.24 shows the menu page that allows the business managers to add and delete menu items to be displayed to the customers through the mobile application. Tapping “Create Menu Item” will display a modal that allows them to add a new menu item.

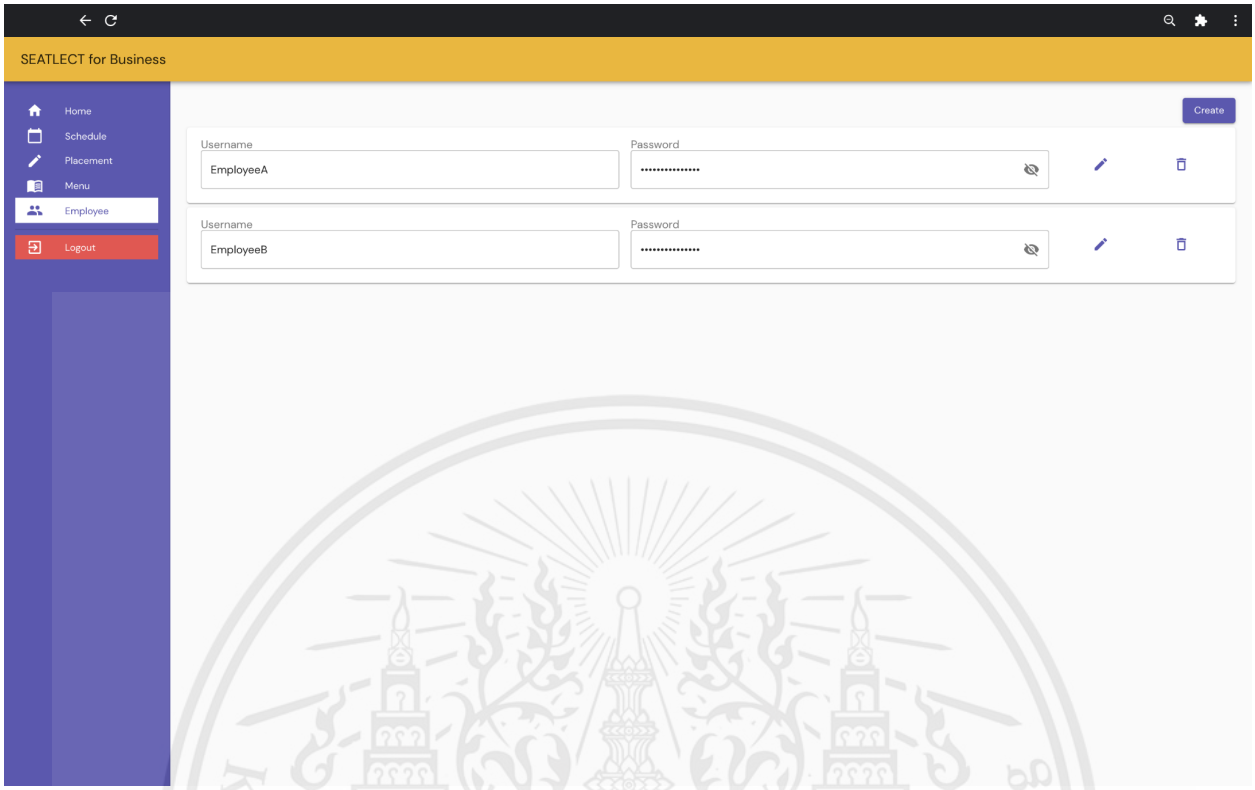


Figure 6.25 Business Web Employee Page

Figure 6.25 shows the employee page where the business managers can create and delete credentials for employees to authenticate in the Employee mobile application.

6.3 Platform Admin Web Application

This section contains screenshots of each page of the platform admin web application. Note that the pictures presented here are mock data. The placeholder images are retrieved from a placeholder image web service and the location data are all made up.

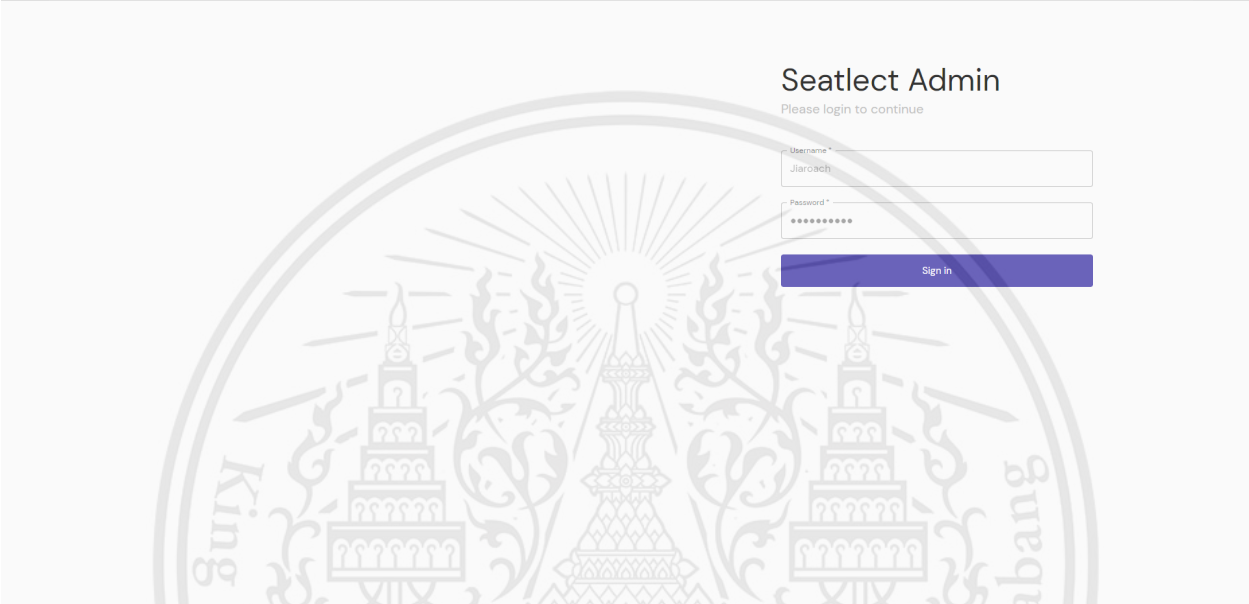


Figure 6.26 Homepage of Admin Web Application

Figure 6.26 shows the platform admin web application page.

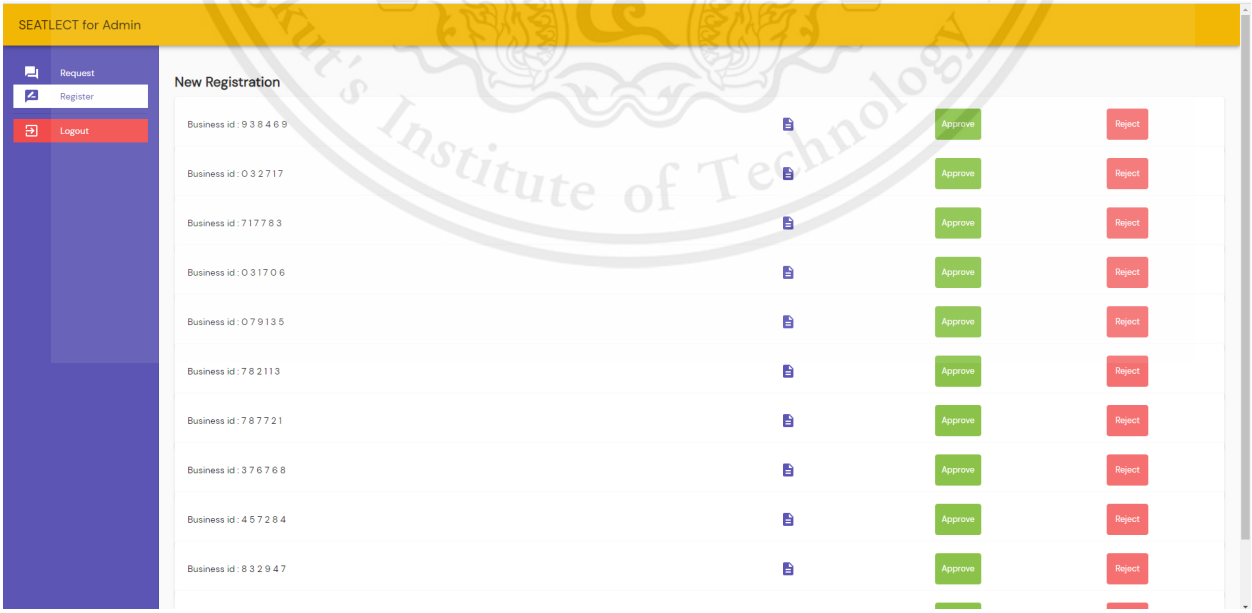


Figure 6.27 Detail page of Admin Web Application

Figure 6.27 shows the list of businesses awaiting approval after registering to the platform.

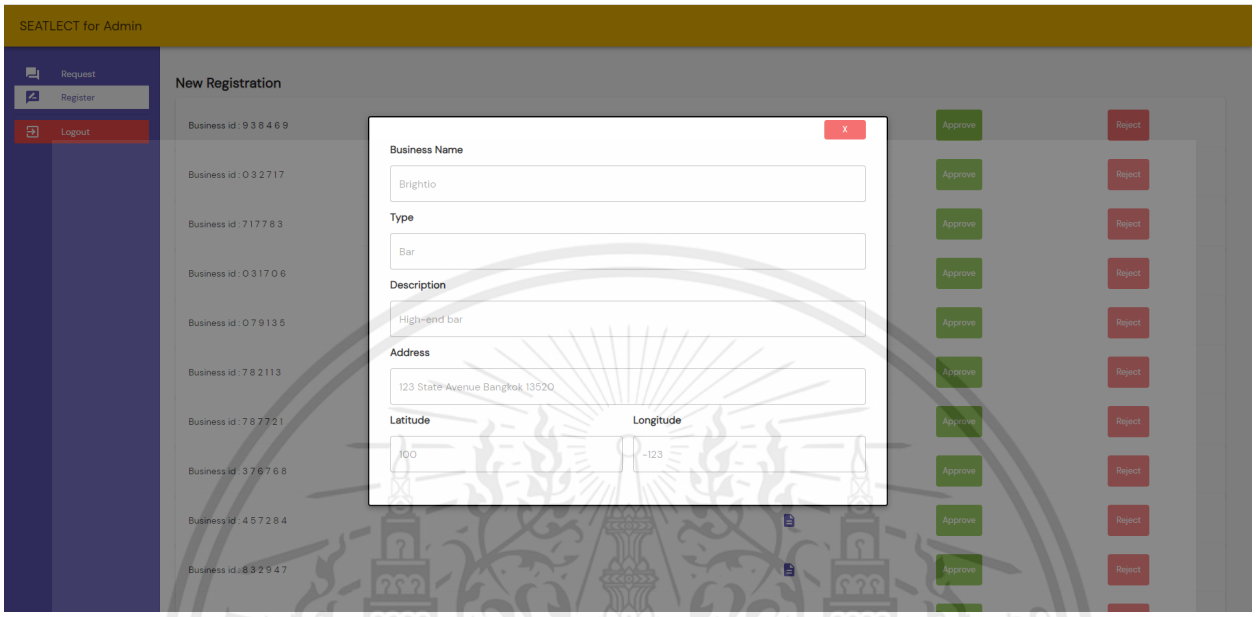


Figure 6.28 Admin Web Registration Detail Page

Figure 6.28 displays detailed information of the business awaiting approval. The platform admin can either reject or approve of the registration.

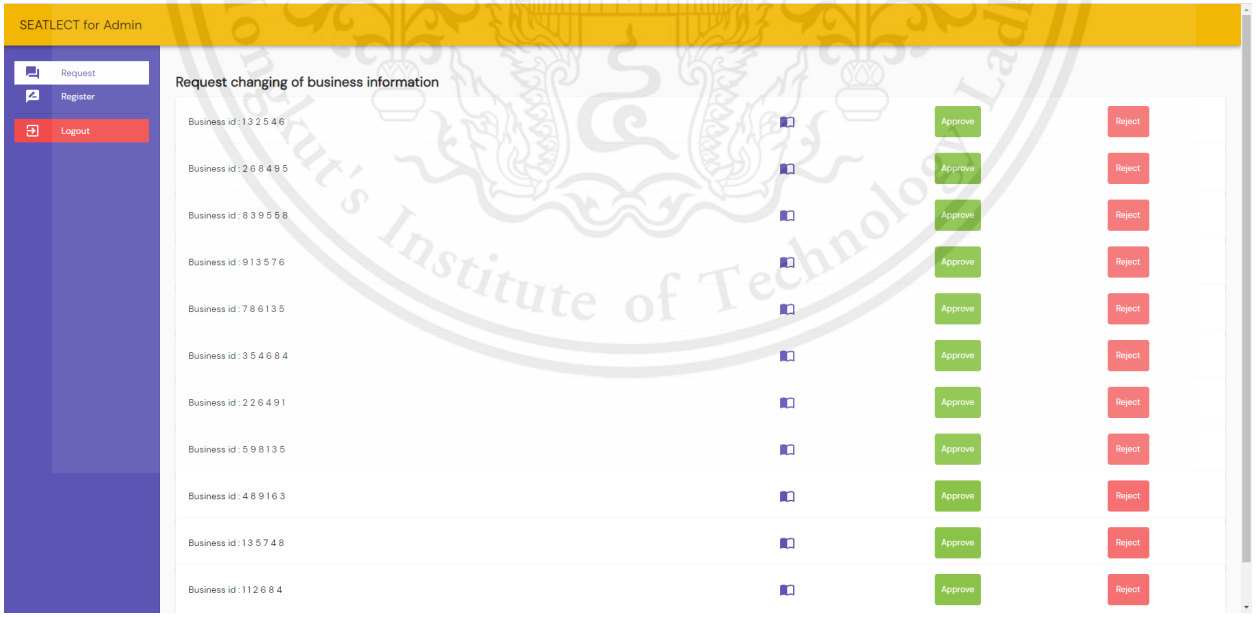


Figure 6.29 Admin Web Change Request List Page

Figure 6.29 displays the list of change requests submitted by the business.

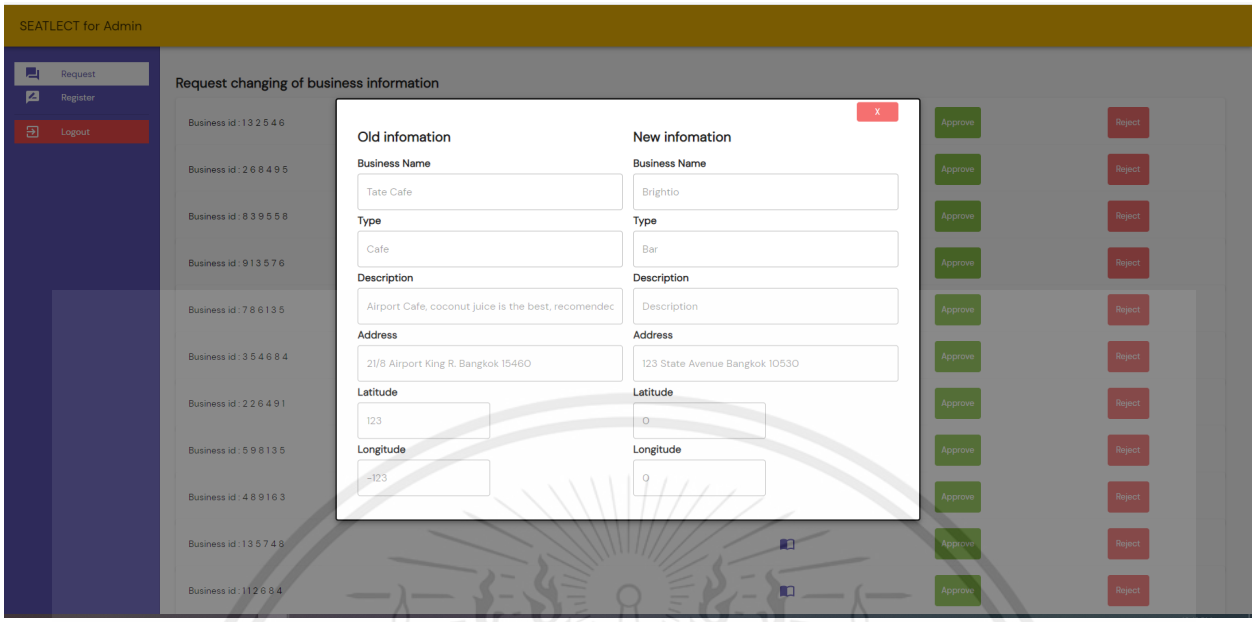


Figure 6.30 Admin Web Change Request Detail Page

Figure 6.30 displays the detailed updated information that the business wants to update. The platform admin can either reject or approve of the change request.

6.4 Employee Mobile Application

This section contains the screenshot and description of each page in the employee application. Note that the pictures presented here are mock data. The placeholder images are retrieved from a placeholder image web service and the location data are all made up.

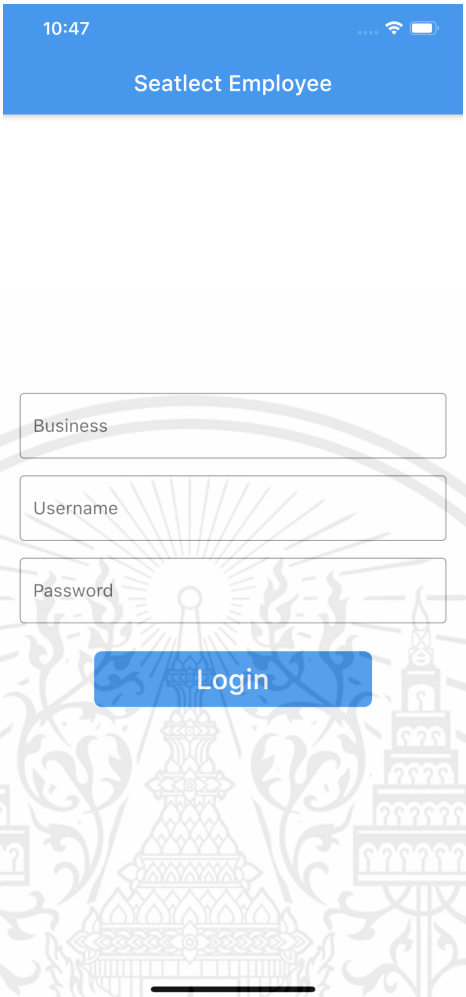


Figure 6.31 Employee Mobile Login Page

Figure 6.31 shows the login page for the employee application.

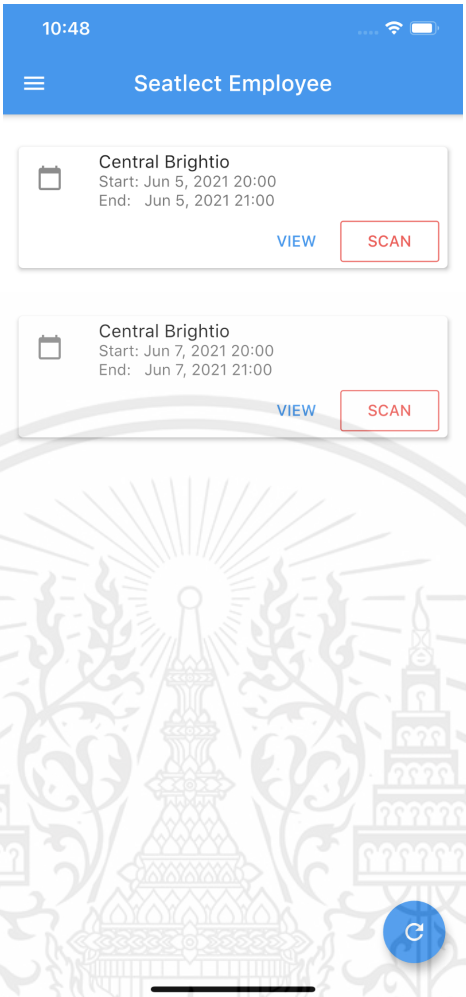


Figure 6.32 Employee Mobile Reservation List

Figure 6.32 shows the list of reservation time slots of the business.

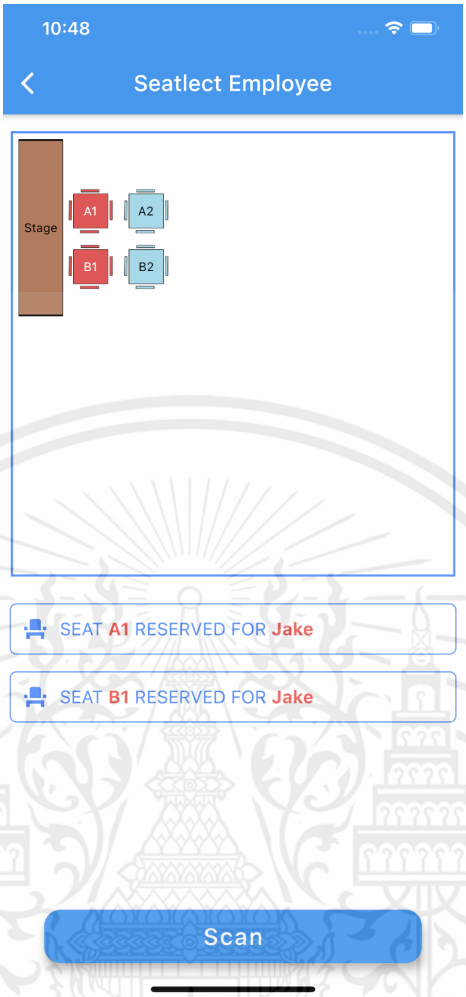


Figure 6.33 Employee Mobile Reservation Detail

Figure 6.33 shows the detailed view of the reservation time slot. Tapping the "scan" button will allow the employee to scan and validate the QR code provided by the customer.

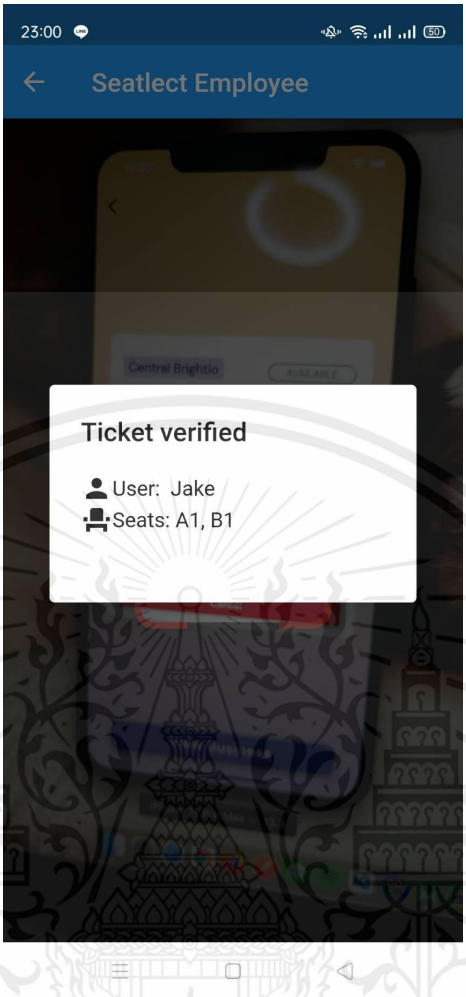


Figure 6.34 Employee Mobile Scanned Ticket

Figure 6.34 shows the ticket information of the customer if successful verification.

Chapter 7
Conclusion

7.1 Summary

The core features and functionality of the Seatlect system is described in detail in the “Requirement Analysis” section. In brief, the system allows businesses to manage their business and reservations (in a time slot manner), customers to discover and make reservations, employees to verify electronic tickets, and platform admin to manage the businesses on the platform. The functional requirements of the project have been implemented successfully, but still leaves room for many improvements.

The project includes several points of interaction, consisting of two mobile applications and two web applications. Although the architecture between frontend and backend themselves is simple, the work put into the project is spread rather thin because it consists of several separate frontend applications. The separation of application makes sense but for a two person project, this means that certain user experiences such as error alerting are not handled as well as they should be.

7.2 Future Work

In this section we will describe the problems with the system and the possible improvements to the system. We separate possible improvements into three categories, user experience improvements, improvements built on existing features, and new features.

For user experience improvements this mostly refers to error handling and reporting on both the backend server, and frontend application. For the backend

server, possible improvement would be a more flexible error error reporting mechanism and attaching a dedicated error logging middleware. For the frontend application, it would be a much needed improvement to provide the user with a more detailed error alerting as opposed to the current generic error alerting. Another improvement for the business web application would also be a more flexible placement seating design tool as well as some sensible default options such as auto generated seat name etc.

For improvements built on existing system this refers to extending existing features to cover more usecase. One improvement would be providing better control over the change requests feature (and also similarly with the registration process). Possible improvement to the change requests includes:

- Better management of change requests made such as cancellation and progress tracking for business managers
- Better rejection system for the platform admin such as providing more detail to the business as to the rejection reasons.

Another improvement on existing features would be a better search engine for the system, the current system relies on constructing complex mongodb queries by ourselves thus restricting what could possibly be accomplished. Although this improvement is included in improvement on existing systems, implementing a search engine functionality would require substantial architectural change to the system such as the addition of an inverted index data storage solution.

Lastly, we shall discuss possible improvement from adding a new feature to the system. There are three major new features that come to mind. It is important to note that none of them are minor new features that would've been possible to properly implement within the time frame of the project.

- The first is a reputation system that incentivize businesses and customers not to take advantage of the system (such as making a reservation then cancelling repeatedly), and also to punish those who misuses the system (eg. lower the business discoverability if there is a pattern of bad behavior).

- The second is an optional payment system that businesses can either use or not. Providing the business with an optional payment system would open the platform to several more markets. This is because certain businesses require reservations to be paid beforehand (a certain amount) as they may not allow walk-in.

- The third is an internal notification system. Currently the system uses Google's gmail smtp server to send mail to notify users. Building a notification system is not trivial but having a notification built into the application rather than using an email client improves user experience.

Bibliography

1) gRPC URL : https://grpc.io/docs/
2) gRPC Protocol Buffer plugins URL : https://grpc.io/docs/languages/go/quickstart/
3) DeepMap's oapi-codegen : https://pkg.go.dev/github.com/deepmap/oapi-codegen/pkg/codegen
4) Echo
5) Flutter Documentation URL : https://flutter.dev/docs
6) BLoC Document - https://pub.dev/packages/flutter_bloc
7) React Document - https://reactjs.org/docs/getting-started.html
8) Konva Document- https://konvajs.org/docs/react/index.html
9) Next.js - https://nextjs.org/docs