Chowspace - Space Renting Platform

Phusith Suktrakul
Virayut Sandhu
Atichat Lappanopakon

Bachelor of Engineering in Software Engineering
International College
King Mongkut's Institute of Technology Ladkrabang
Academic Year 2018
KMITL-2019-IC-B-003-006

Thesis - Academic Year 2018

Bachelor of Engineering in Software Engineering
International College
King Mongkut's Institute of Technology Ladkrabang

Title: Chowspace: Space Renting Platform

Authors:

1. Phusith Suktrakul Student ID: 58090031
2. Virayut Sandhu Student ID: 58090035
3. Atichat Lappanopakon Student ID: 58090043

Approved for Submission

Veera Boonji
.....

(Assoc.Prof.Dr. Veera Boonjing)
Advisor

Date 26/06/19

Acknowledgments

This project could not have been completed without great advice from advisors and would like to express our deepest gratitude to Assoc.Prof.Dr.Veera Boonjing, the authors also would like to thank Dr.Isara Anatavrasilp, Dr.Ukrit Watchareeruetai and friends for giving suggestions in presentation and useful discussions. Furthermore, the authors are thankful to lecturers at International College for providing support and also we would like to give special thanks to families for their support, motivations, and encouragement. Lastly, the authors would like to express a sense of gratitude to all, who directly or indirectly, have a part in this venture.

Abstract

Renting space has always been a cumbersome task. Renter has difficulty finding space to rent while the space's owner lacks a common channel to advertise their space. Once the renter has found space to rent, the procedure that follows like acquiring the details and confirmation of payment to reserve space are spread out across different services. Moreover, it is also difficult for the space's owner to keep track of the space that is rented out. A single service that provides all the mentioned functionality is much needed.

Chowspace is a space renting platform that provides a clear communication flow between the renter and the space's owner. Chowspace also provides an advertisement area for the space owner to showcase their space and a management system for the space owner to view and manage their reservation. Chowspace provides details of the space to the renter and also a communication system to acquire additional details.

Table of contents

List of figures

List of tables

Chapter 1
Introduction

1.1 Problem Descriptions

The Internet is one of the main factor in making life more convenient. With the help of Internet people are able to get information, communicate and socialise. One of the more recent trend on internet is online shopping. Online shopping alleviate the need to physically go to the store, view and compare the product, stand in line and pay for the product. Online shopping is a one stop service where all of the procedure is done sitting in front of a computer. It makes people's life much more convenient.

Nevertheless, one such area where with the help of Internet could make people's life more convenient is renting spaces. Spaces for setting up offices and businesses, organizing events like exhibition, marriage or seminar, are difficult to find. Moreover, the traditional method of renting spaces requires looking through various advertisement, making contact with the owner to find more details, making payment and sending the receipt to reserve the booking. These steps are spread out and the renter has to keep track of the process on their own. Similarly, the space's owner has to find advertisement area for their space and also keep track of all the information of the renter which becomes extremely hard when the number of renter grows.

As a result, there clearly is a need for a platform that connects both renter and space's owner together in order to smooth out the communication flow between them, also provides advertisement area for the space, management system for the owner to easily view various reservation. The platform is guarantee to make renting space much more convenient.

The desirable features of space renting platform should include area where the owner can showcase their spaces and ability to search for specific space. Moreover, it should also allow the renter to reserve directly via the system and the ability to commu-

nicate with the owner should the need arise. The space must also be able to be reviewed by the previous renter sharing their experience about the space. The system must have place management system for the owner to keep track of various reservation occuring within the system and views various reservation statistics such as number of reservation made per month, etc.

Most existing space renting platform doesn't have all the mentioned features. It only have the feature where it allows the owner to showcase their spaces and the renter can specifically search for a space. Any reservation and communication between the renter and the owner are done using the traditional approach that is the renter would directly contact the owner. Our application tries to make the space renting platform that would greatly benefit both the renter and space's owner.

1.2 Objective and scope

In order to indicate the success of the project, several goals have been set. The following list shows all the goals which are needed to be satisfied.

- Provides web application for space renting platform for the renter and space's owner.
- Provides clear communication flow between renter and space's owner.
- Provides management system for space's owner to manage various places, spaces and reservations.
- Provides review system for the renter to share their experience about the space.

The proposed system is designed to work under the following scope.

- The space renting platform does not include any form of payment to the space's owner. All transaction between the renter and the space's owner are done outside of the platform.

Chapter 2
Literature Review

There are numerous product on the market regarding the space renting platform. This chapter focuses on describing the various product in the market and comparing them with the proposed system. The features that space renting platform should provide are Advertisement area, Space validation, Smart search, Google map integration, Rating & Review, Reservation system, Message system, Management system, Device support and Facebook integration. This chapter is focused on which features that these various products are provided.

2.1 ThaiFranchiseCenter

ThaiFranchiseCenter is one of the top website containing franchise business information in Thailand. This website is a platform providing easy search of franchise business for users to compare and decide the right franchise for them. This website allows business owner to use as an advertisement area. Also, the website contain statistical information of the franchise business.

2.2 Thumle Khaikhong

Thumle Khaikhong is a website directly relating with renting space. The website provides an advertisement area but mostly focusing on retail space. The website allows anyone to advertise their space by posting space information regarding their space and they can categorise their space accordingly. Thumle Khaikhong provides similar features as ThaiFranchiseCenter but focuses on retail space. Contacting space's owner still depends on the information provided by them.

2.3 Kaidee

Kaidee is a platform for buying and renting of items but focuses mainly on second-hand items in Thailand. The most posted on Kaidee.com are vehicles, amulets, smartphones, tablets, motorcycles and properties which relate to renting and buying space. Kaidee provides a lot of feature such as smart search, item validation, notification system, promotion system, live chat and also item management system.

2.4 Event Banana

Event Banana is an online marketplace for event venue covering various event types. There are seminar room, meeting room, marriage place, restaurant and bar in their categories. Event Banana includes more information regarding renting the space for a specific event like seminar or marriage. There are smart search, space validation, notification system, promotion system, space management system, rating and review system and booking system. However, they do not provide communication channel with the space's owner on their platform.

2.5 DDproperty

DDproperty is also an online marketplace that relate with renting spaces on both short term and long term and even with buying spaces. Beside their main focuses on residence spaces such as condo and houses, there are also commercial spaces like retail, office, warehouse and factory space. DDproperty provides most of the functionality that the other website has to offer, and also offer more control in management system with things like statistic dashboard, booking management system and inbox management system for space's owner, however, there is no live chat, rating and review system.

2.6 Facebook

Facebook is also an online marketplace for renting property. Facebook provides an only smart search feature that use filters such as housing type, price range, bedrooms, bathrooms to find what user want, however, Facebook can efficiently reaching renter on Facebook platform where they're already looking for a home for rent and use messenger application to be clear communication channel between renter and owner.

	thaifranchisecenter	ทำเลทองคอม	kaidee	eventbanana	ddproperty	facebook	chowspace
Features || Tools							
Advertisement Area	Yes	Yes	Yes	Yes	Yes		Yes
Space Validation			Yes	Yes	Yes		Yes
Smart Search			Yes	Yes	Yes	Yes	Yes
Google Map Integration				Yes	Yes	Yes	Yes
Rating & Review				Yes			Yes
Reservation System				Yes			Yes
Message System					Yes	Yes	Yes
Management system							
Space Management System	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Reservation Management System					Yes	Yes	Yes
Device Support							
Website	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Mobile Application			Yes	Yes	Yes	Yes	Yes
Social Media							
Facebook Integration			Yes	Yes	Yes	Yes	Yes

Figure 2.1: Literature review summary table

Chapter 3
Background Knowledge

This chapter provides background knowledge essential for the project. The background knowledge part covers the basics of microservice software architecture.

3.1 Microservice Architecture

Instead of building a single monstrous, monolithic application, the idea is to split application into a set of smaller, interconnected services. A service typically implements a set of distinct features or functionality, such as order management, customer management, etc. Each microservice is a mini–application that has its own architecture consisting of business logic along with various adapters and databases. Some microservices would expose an API that's consumed by other microservices or by the application's clients.[1]

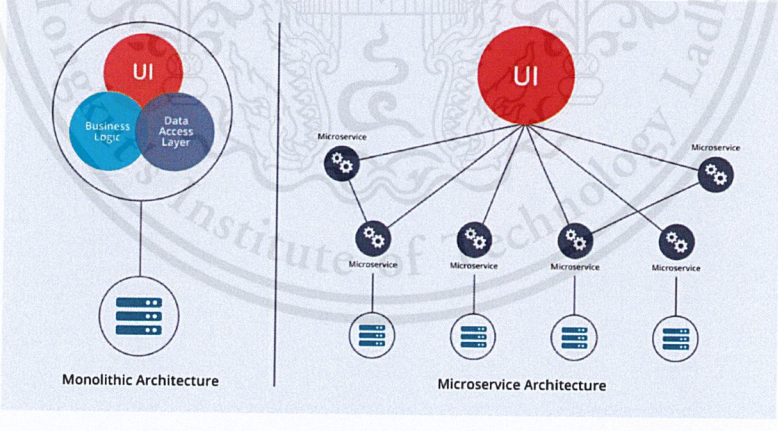

Figure 3.1: Monolithic and Microservice architecture [2]

The Benefit of microservices is that it tackles the problem of complexity. It decomposes a monstrous monolithic application into a set of services. While the total amount of functionality is unchanged, the application has been broken up into manageable chunks or services. Consequently, individual services are much faster to develop

and much easier to understand and maintain. Moreover, each service can be scaled independently through replication.

Despite the benefits that it provides, a microservices application is a distributed system. It provides a new set of challenges that needs to be taken care, such as client–microservice interaction, inter–process communication, service discovery, distributed transaction, and service deployment. The following describes the solution to the challenges.

3.2 Client-Microservice Interaction

This challenge arises when deciding on how the application's clients will interact with the microservices. With a monolithic application, there is just one set of endpoints. In a microservices architecture, however, each microservice exposes a set of endpoints itself. Consider, a web page showing various information such as product details, reviews, product's inventory, and shipping details.

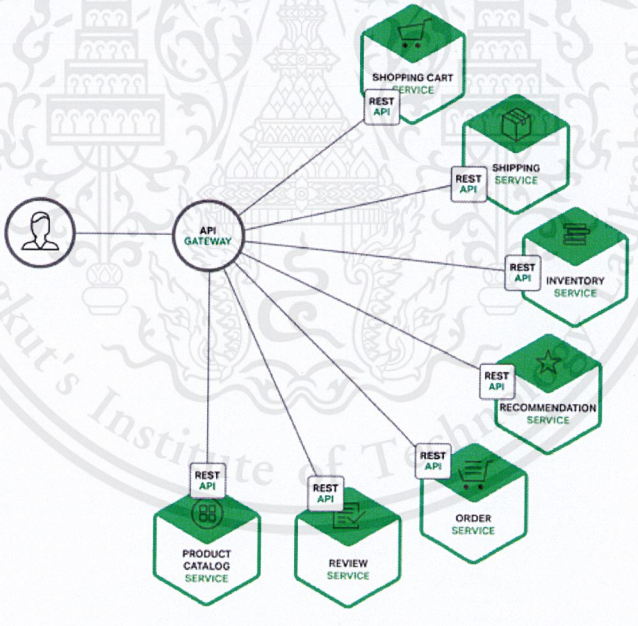

Figure 3.2: API Gateway encapsulates the underlying various services [3]

This information is gathered from various services. A client could make requests directly to each of the services but there are limitations to this approach. Firstly, the client has to make several separate calls to render a page. This approach is significantly inefficient over the public Internet and would definitely be impractical over a mobile

network. This approach also makes the client code much more complex. Furthermore, this approach also makes it difficult to refactor the microservices. However, using an API Gateway solves these issues.

An API Gateway is a server that is the single entry point into the system. The API Gateway encapsulates the internal system architecture and provides an API that is tailored to each client. It might have other responsibilities such as authentication, monitoring, load balancing, caching, request shaping and management, and static response handling.[3] All requests from clients first go through the API Gateway. It then routes requests to the appropriate microservice. The API Gateway will often handle a request by invoking multiple microservices and aggregating the results. It can also translate between web protocols such as HTTP and WebSocket and other web–unfriendly protocols that are used internally.

3.3 Inter-Process Communication

In a monolithic application, components invoke one another via language–level function calls. In contrast, a microservices–based application is a distributed system running on multiple machines. In order to select the most suitable form of communication between services, the interaction styles between services must be thought about first. Interaction styles can be categorized along two dimensions. The first dimension is whether the interaction is one–to–one or one–to–many and the second dimension is whether the interaction is synchronous or asynchronous. One–to–one and one–to–many interaction tell whether the request is processed by one service or many services. The synchronous and asynchronous interaction tells whether the response is either expected right away or eventually. The following table shows the interaction styles:

	one-to-one	one-to-many
Synchronous	Request/Response	-
Asynchronous	Notification Request/Async Response	Publish/Subscribe Publish/Async Responses

Table 3.1: Inter-Process communication [4]

Synchronous request/response can be met by using protocols such as REST or Thrift. Each of the protocol differs in the communication protocol and message data format. REST uses HTTP protocol and JSON and XML as the message data format. While Thrift provides a larger set of communication protocol and supports binary data

format to provide faster transfer of information.

Asynchronous communication involves three main roles, producer, who produces messages, consumers, who consume messages, and message broker, who deliver messages. Messages are exchanged over channels maintained by the message broker. Any number of producers can send messages to a channel. Similarly, any number of consumers can receive messages from a channel. Both the producer and consumer are not aware of each other so as to decouple each service from each other as much as possible.

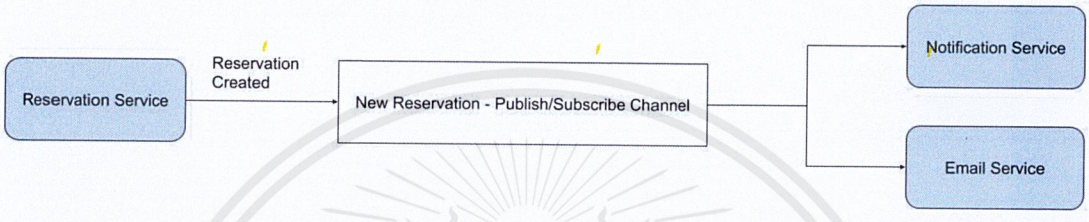

Figure 3.3: Reservation service publishing new reservation event and notification, email service receiving the event.

3.4 Service Discovery

In microservice architecture, service instances have dynamically assigned network locations. Moreover, the set of service instances changes dynamically because of autoscaling, failures, and upgrades. In order to make a request, the service needs to know the network location (IP address and port) of other service instances. In a traditional application, the network locations of service instances are relatively static.

A key part of service discovery is the service registry. The service registry is a database of available service instances. A service can request for the location of other services through the service registry. Service instances are registered with and deregistered from the service registry. Some system component must handle the registration and deregistration of the service instance created. In some deployment environments, service discovery is built in. For example, Kubernetes provides a built in support for service discovery.[5]

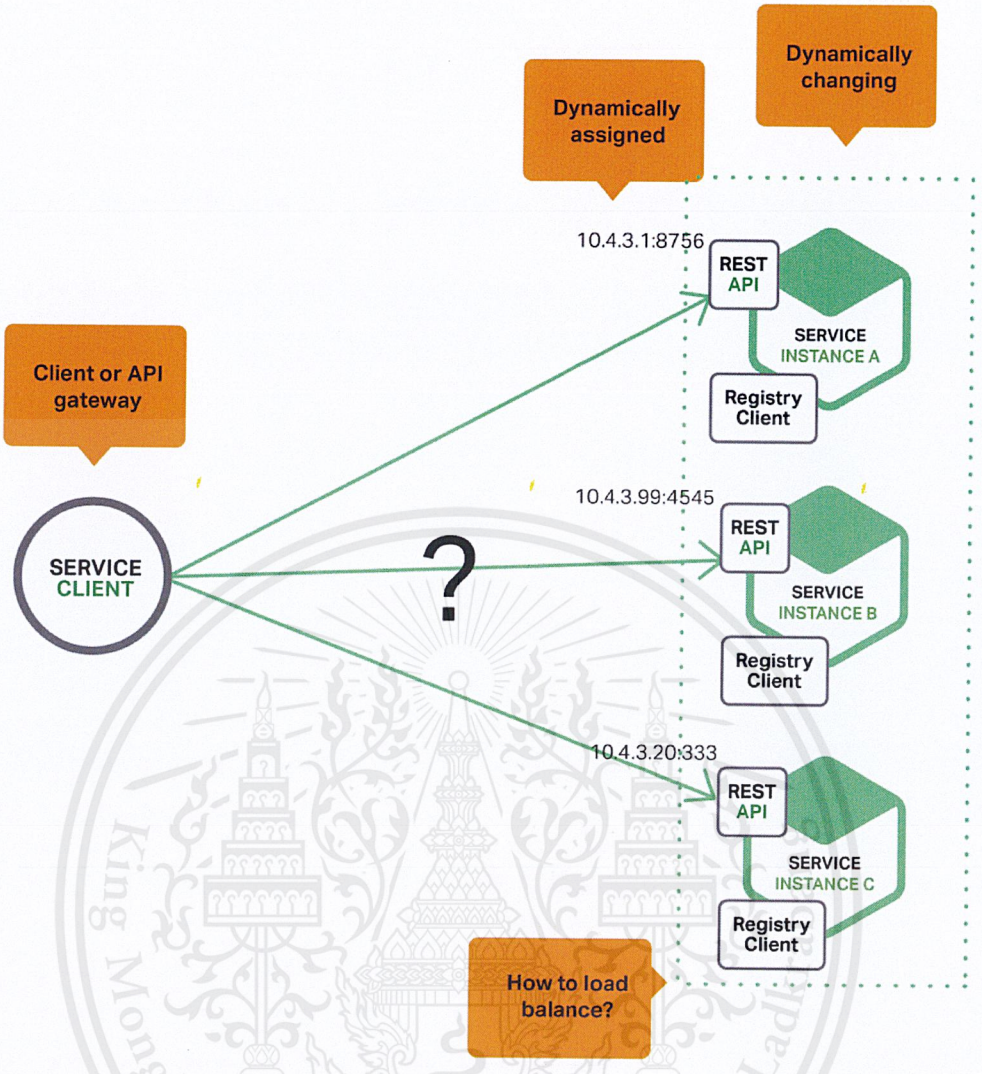

Figure 3.4: The figure shows the problem where the service client is unable to find the dynamically created services. [5]

3.5 Distributed Transaction

A monolithic application typically has a single database. But in microservices, the data owned by each service is private to that service and can only be accessed via its API. The challenge that arises is how to implement business transactions that maintain consistency across multiple services. Different services often use different kinds of databases. So, two phase commit functionality provided by relational databases is not feasible. [6] To remedy this problem, event driven architecture along with saga pattern is used. A saga is a sequence of local transactions on various services. Each local transaction updates the database and publishes a message or event to trigger the next local

transaction in the saga. If a local transaction fails because it violates a business rule then the saga executes a series of compensating transactions that undo the changes that were made by the preceding local transactions.

3.6 Service Deployment

A microservices application consists of tens or even hundreds of services. Services are written in a variety of languages and frameworks. Each one is a mini–application with its own specific deployment, resource, scaling, and monitoring requirements. Microservices are usually deploy using service instance per container pattern. In this pattern, each service instance runs in its own container. Containers are a virtualization mechanism at the operating system level. A container consists of one or more processes running in a sandbox. From the perspective of the processes, they have their own port namespace and root filesystem. Container's memory and CPU resources can be limited. Some container implementations also have I/O rate limiting. Examples of container technologies include Docker and Solaris Zones.[7]

A service is package as a container image. A container image is a filesystem image consisting of the applications and libraries required to run the service. Some container images consist of a complete Linux root filesystem. Others are more lightweight. Once the service has been packaged as a container image, it is then launch as one or more containers. Multiple containers are run on a single physical or virtual host. Running containers on multiple host can be easily achievable using a cluster manager such as Kubernetes or Marathon to manage the containers. A cluster manager treats the hosts as a pool of resources. It decides where to place each container based on the resources required by the container and resources available on each host.

Chapter 4
Requirement Analysis / Design / Architecture

This chapter provides the definition of user requirements in FURPS aspect. The basic structure of the system are shown in use-case diagram and activity diagram.

4.1 Requirement Analysis

In this system, user can be both space's owner and renter after they register on the web application. The space's owner can publish their own space to the system by providing all the information required by our system in order to make their space available for reservation from other people who are interested. The renters can choose the space that they are interested in and make a reservation for that place. The following are the explanation of some of the processes available to get the gist of the system.

4.1.1 Publishing Space

Owner input required information of the space in the system. The space will be published making it available for reservation. The published space will have validation status as pending until the space is verified as legitimate.

4.1.2 Booking Space

Renters searches for the space that they are interested in. The system shows information of the selected space. User can see the available date of the place to make a reservation. In case of space that is not available on certain date, user will not be able to make a reservation on that day.

4.1.3 Messaging

Renter is able to communicate with the space's owner by sending messages through the messaging system inquiring for more specific detail.

4.1.4 Rating and Review

Renter who have reserved the space is able to give feedback to the space's owner or future renter sharing their experience. This features helps the space's owner to improve their facility further and helps the renter makes a better decision before booking the space.

4.1.5 Verifying Space

After publishing the space, admin has to verify the space that it is valid or not. If the information of the space is valid, admin can mark the space as verified but if it is not valid, admin can withdraw the space from the system.

4.1.6 Notification

Renter and owner is able to recognize when the state of activity, that is a part of themselves, changes with the notification of the system.

4.1.7 Email

Our system provide the email service that will sent email to our customers for the event that will happen in the system.

4.2 Requirements as FURPS

No.	Requirement	Type
1	User can be both space's owner and renter after registering.	Functional
2	User can acknowledge the activity with the notification.	Functional
3	Renter can reserve the space to rent.	Functional
4	Renter can search the space by name, category or available date.	Functional
5	Renter can see the available date for the space.	Functional
6	Renter can rate the spaces they have reserved.	Functional
7	Renter can review the spaces they have reserved.	Functional
8	Renter can send messages to the space's owner to communicate.	Functional
9	Renter is able to report spaces that false advertise.	Functional
10	Space's owner can publish their space for renting on the web application.	Functional
11	Space's owner can see how many people has made a reservation with their space.	Functional
12	Space's owner can upload images of their spaces.	Functional
13	Space's owner can edit the detail of their spaces.	Functional
14	Space's owner can select which category their space belongs to.	Functional
15	Space's owner can see statistical data of their space.	Functional
16	Space's owner can accept or reject renter's reservation.	Functional
17	Admin can verify spaces to be valid or not.	Functional
18	Admin is able to ban every user account.	Functional
19	Admin can add and update the category of the space.	Functional
20	System can send the email to user for the event that will happen.	Functional
21	User interface is easy to use.	Usability
22	System always make the available date for the space up to date at all time.	Reliability
23	System is able to soft delete the space that is invalid.	Reliability
24	System must respond to user quickly.	Performance
25	Web application support both PC and mobile devices.	Support ability

Table 4.1: Requirements as FURPS

4.3 Use Case Diagram

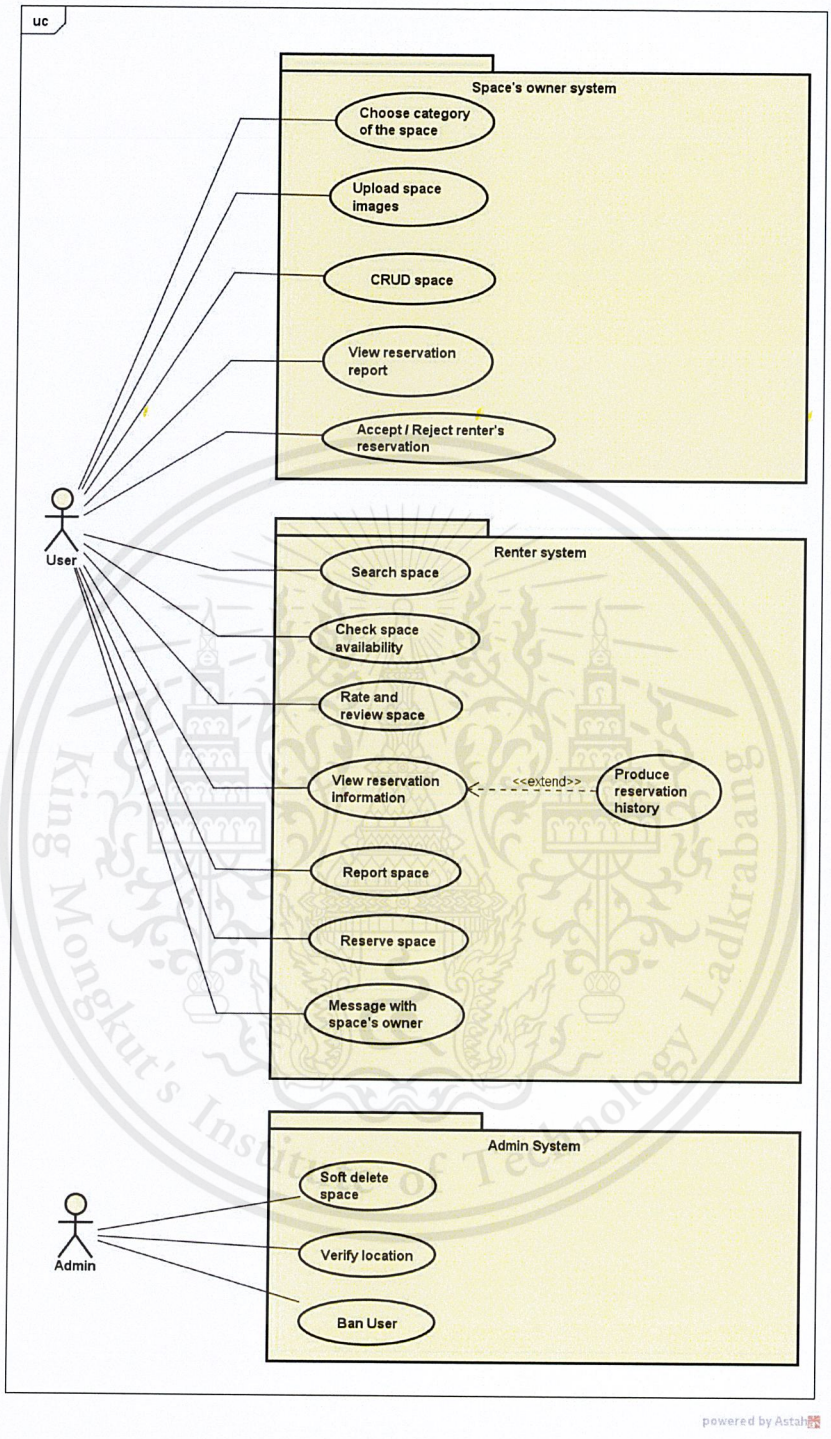

Figure 4.1: Use case diagram

4.3.1 Brief Use Case

The following are the description of the use cases in brief.

Choose category of space

Space's owner can choose the categories of space that the system provides.

Upload space images

Space's owner can upload images of the spaces to showcase their spaces to the renter.

CRUD space

Space's owner is able to create, read, update or delete the space. Newly created space are automatically publish within the system.

View reservation report

Space's owner can see statistical data of reservation for their space.

Accept / Reject renter's reservation

Space's owner can choose whether to accept or reject each renter's request for reservation.

Reserve space

Renter can make reservation for spaces.

Search space

Renter is able to search the space that they are interested in.

Check space availability

Renter can check the available date of the space before making reservation. If the date is not available then reservation can't be made.

Rate and review location

Renter can give rating score as well as a comment to the space they have made reservation.

Message with space's owner

Renter can communicate with the space's own through the messaging system to inquire more detail regarding the space.

Reservation information

Renter can view reservation information.

Produce reservation history

The system stores all the current and previous reservation made within the system.

Report space

Renter is able to report the space that they find to contain misleading information.

Verify space

Admin is able to verify that the space information is valid or not.

Soft delete space

Admin can delete the space that are invalid. The system must not permanently delete the space but rather soft delete it.

Ban user

Admin can ban users violating the standard rules supplied by the system.

4.4 System Architecture

The following diagram shows how the user's request is handled behind the scenes. The user's request will be routed to the appropriate service based on the request.

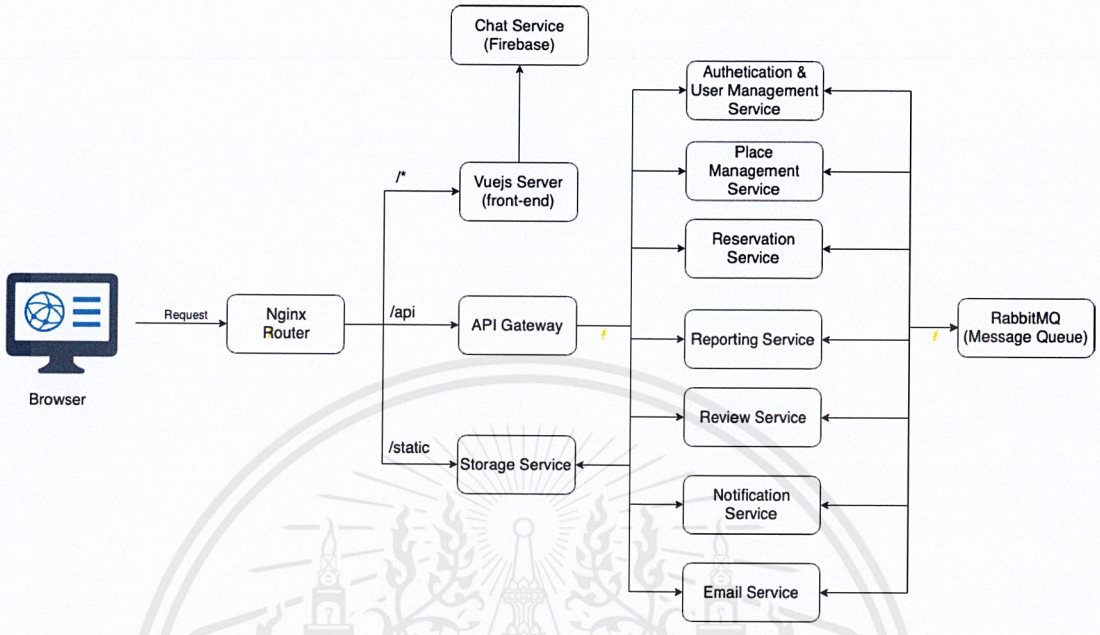

Figure 4.2: System architecture

4.4.1 Nginx Router

This component is responsible for the routing request based on the url format to the correct back-end services. Url with /api will be routed towards api gateway for invoking various other services within the system and response accordingly. Url with /static will be routed to the storage service which is responsible for serving static files and images. Other url format will routed to Vuejs server.

4.4.2 Vuejs Server

Vuejs server is responsible for serving html and javascript files responsible for rendering web pages on browsers.

4.4.3 API Gateway

API gateway is the entry point into the back-end services. It is responsible for invoking various internal services based on the request and then aggregating the result and returns the response back to the client.

4.4.4 Storage Service

Storage service is used for storing various images of places and spaces. It is responsible for serving those static images to the client.

4.4.5 Authentication & User Management

User's account information and user's profile is maintained by this service. It is responsible for authenticating user and authorisation of user's actions.

4.4.6 Space Management Service

This service is responsible for the management of places and spaces. Information of places and spaces is created, stored, updated using this service.

4.4.7 Reservation Service

Reservation service as the name suggest is responsible for reserving spaces for user.

4.4.8 Review Service

This service is responsible for handling user's review of the place.

4.4.9 Notification Service

Notification service is responsible for notifying user through browser regarding various actions that occurs within the system.

4.4.10 Chat Service

Chat service provides the functionality for the users to communicate with each other.

4.4.11 Email Service

Email service is responsible for sending emails to the users based on the various action the occurs within the system.

4.4.12 Reporting Service

This service is responsible for gathering report information of various users, places, and spaces for the admin of the system to further act upon.

4.4.13 Message Queue

This service acts as a message broker for various services to communicate with each other. It is responsible for receiving messages from the producer and reliably sending the message to the consumer.

4.5 Domain Class Diagram

The following diagram shows the models and their relationship within the system.

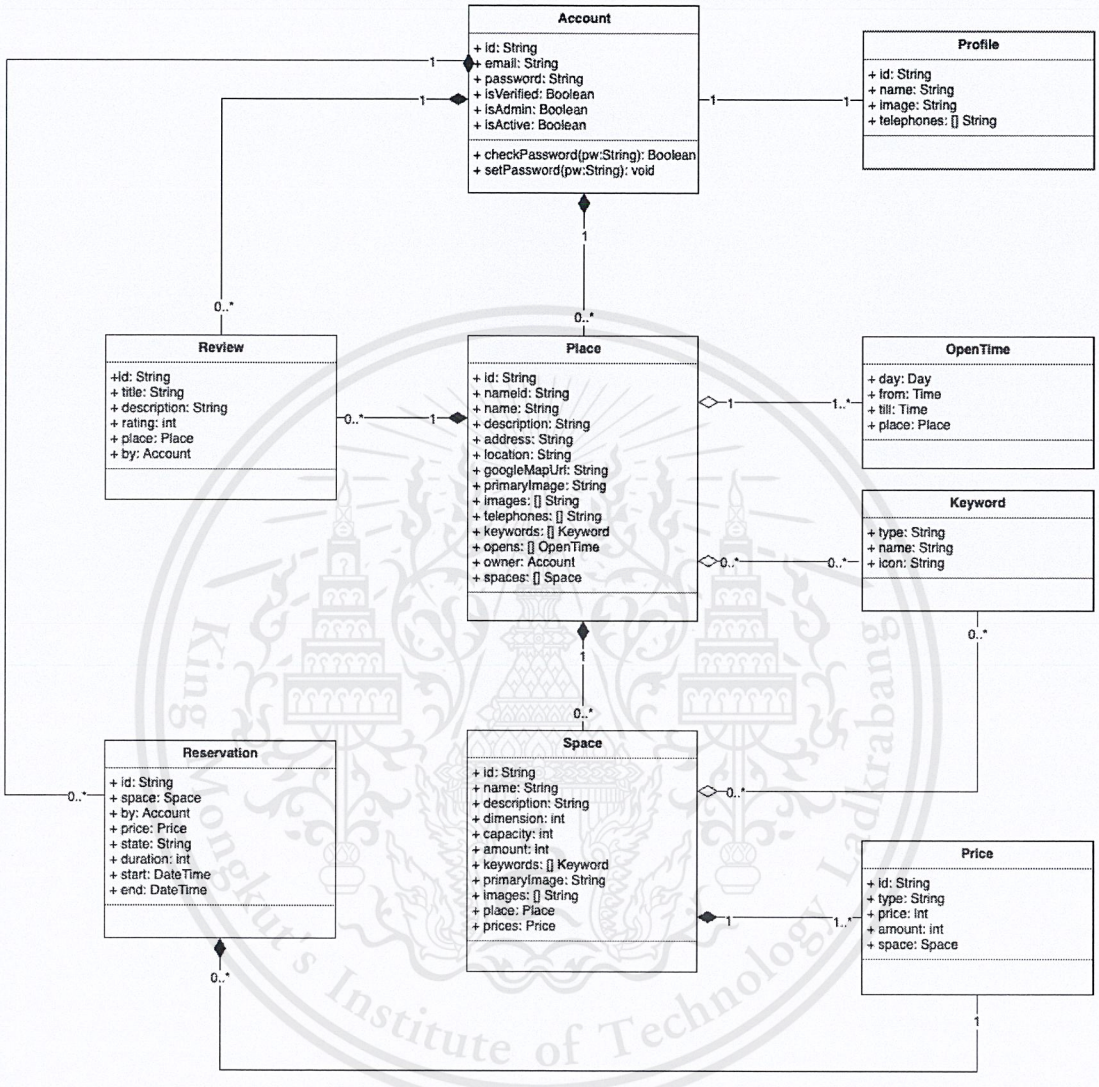

Figure 4.3: Domain class diagram

4.6 Authentication Service Class Diagram

The following shows the class diagram of the authentication and user management service.

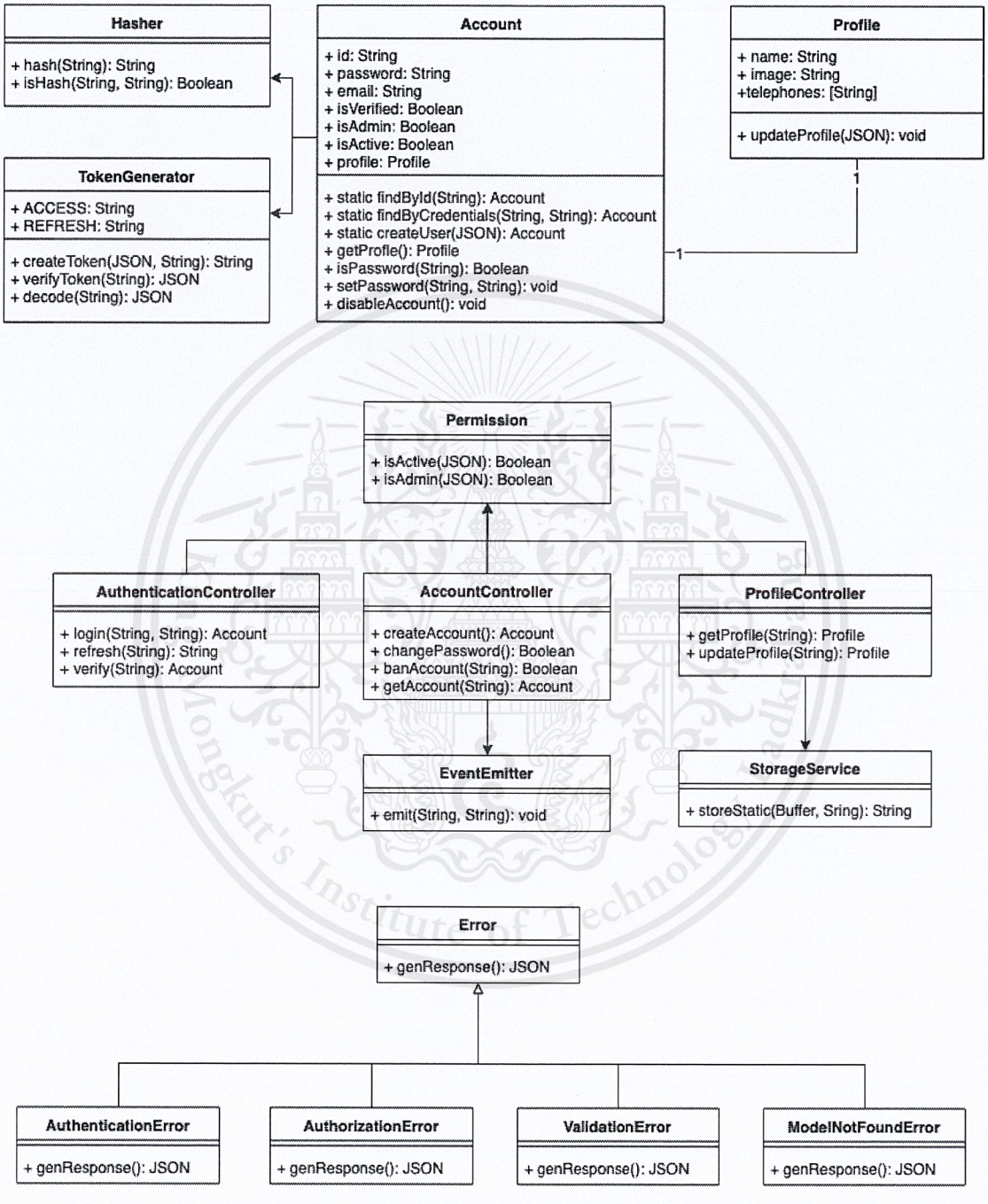

Figure 4.4: Authentication service class diagram

- The Controller classes handles the business logic of the application and is mapped to a particular HTTP URL endpoints. It invokes various function of the models.
- The Account class represent the account information of the user within the system and has a profile which contain the public information of each user.
- The Hasher is responsible hashing the user's password stored in the database.
- The TokenGenerator generates and validates token of the user. The token will be used by various services to validates the user's authentication.
- Each of the models can throw various business logic errors to the controller and the Error classes generates appropriate responses for the user.

Chapter 5
Implementation

This section elaborates the actual development process of the project which describes the development tools and how the system works with the users. For the development tools, Vue.js is used as a front-end, Node.js is used as a back-end and many other tools which make the project that we build as a web application become completely perfect. Also, this chapter describes how the system component communicates between the client side and the server side.

5.1 Development Tools

In the development tools, the system is separated into client-side(front-end) and server-side(back-end). Our system architecture is designed to be the microservices. Our system will be divided into many services so we need to have tools that can help our software performance become powerful. Starting with the front-end, we use the Vue.js framework for the development of the user interface of the website. The server side we use the Node.js framework to implement each of our services for handling between client's requests and the server's response. Docker and Kubernetes come to help with deployment if the services that we gave divided as a microservice. Nginx and RabbitMQ work as a middle man in the server side to direct the client's request to the service, which depends on each request's objective and handles the real-time streaming data respectively. Lastly, PostgreSQL, MongoDB, and Firebase (Real-time database) is a database in this system that use to store all the data inside our system.

5.1.1 Vue.js

Vue.js is an open-source JavaScript framework for building user interfaces. [8] Vue focuses on making some of the best ideas in web UI development (components, declarative UI, hot-reloading, time-travel debugging).[9] Vue is designed to be incrementally adoptable making integration into projects that use other JavaScript libraries

simplified. The core library of Vue focuses on declarative rendering and component composition embedded into HTML pages. Advanced features such as routing, state management and build tooling which are offered via supporting libraries and packages. [10] Vue is also capable of powering advanced single-page applications.

5.1.2 Node.js

Node.js is an open-source and cross-platform JavaScript runtime environment [11] that executes JavaScript code outside of a browser. Since we decide our software architecture to be microservices, Node.js is the perfect framework that matches with microservices design because it supports multiple microservices frameworks which enables the development of applications based on microservices strategy and it is one of the best solutions for creating high-performance, real-time web applications. Combining with a microservice-based application, it can handle an extreme amount of load with low response times. For productivity, Node.js uses NPM (node package manager) which comes up with tons of ready to use modules that speed up the development process. Node.js uses javascript as interface language so that the same language can be used for the back end and front end. This saves lots of development time[12] and it is also has a community that supports millions of developers across the globe.

5.1.3 Docker

Docker is an open source tool has been designed to create applications as a small container on any machine. By using docker development, deployment is too easy is for developers especially for microservice architecture which has many services that must be deployed. We can say this is very light-weight in size which includes minimal OS and your application. In a way, Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. This gives a significant performance boost and reduces the size of the application.[13]

5.1.4 Kurbernetes

Kubernetes is a powerful system, developed by Google, for managing containerized applications in a clustered environment. By using Kubernets we can manage Docker containers very smoothly as well as we can control over the scaling, monitoring, and automation.[13]

5.1.5 Nginx

Nginx is open source software for web serving, reverse proxying, caching, load balancing, media streaming, and more. It started out as a web server designed for maximum performance and stability.[14] We use Nginx to direct the client's request to the front-end service or back-end services depend on the objective of the request.

5.1.6 RabbitMQ

There are many ways of communication between the service of microservice such as the synchronous and asynchronous way. The message queue is the asynchronous communication that is commonly use and RabbitMQ is also one of the tools that easy to use and popular. RabbitMQ is an open source message broker that supports multiple messaging protocols. The way RabbitMQ routes messages depend upon the messaging protocol it implements. In general, a broker is a person who facilitates trades between a buyer and a seller. Similarly, if we want to trade messages between two distributed software components, we need a mediator. This mediator is known as the message broker. It receives incoming messages from a sender and sends them to a recipient. This way the sender and receiver can be totally isolated.[15]

5.1.7 PostgreSQL

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. PostgreSQL comes with many features aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help manage data no matter how big or small the dataset.[16]

5.1.8 Mongo DB

MongoDB is a cross-platform and open-source document-oriented database, a kind of NoSQL database. As a NoSQL database, MongoDB shuns the relational database's table-based structure to adapt JSON-like documents that have dynamic schemas which it calls BSON. This makes data integration for certain types of applications faster and easier. MongoDB is built for scalability, high availability and performance from single server deployment to large and complex multi-site infrastructures.[17] We use MongoDB to keep the data in our system that does not need the relational between data.

5.1.9 Firebase (Realtime database)

Firebase real-time database is a NoSQL cloud database that stores data in JSON format and synchronization the data in realtime with every device within a short period of time. Firebase real-time database supports offline mode (the data will be stored in local storage until the connection is back to normal then it will sync the data automatically). It also has security rules that we can decide the data access rules for both read and write operations.

5.2 Client side

For this system, the users are separated into two actors. The one who can publish their own space on the website is called the owner and the other one who can select the space for renting is called renter. Both two actors can interact and manage their spaces on the website.

5.3 Server side

The server is implemented in JavaScript using Node.js framework for each of the service. It is the center of the system between the client and the database, that handles requests from users interacting with the web application. The primary role for this side is to carry out the request from the client side such as patching a data into a database, creating a new data or getting existing data out to display.

Chapter 6
Preliminary Result

This chapter provides the result of the development of the project. It describes each pages of the website on its working and functionality for users to interact with.

6.1 Web Application

6.1.1 Home Page

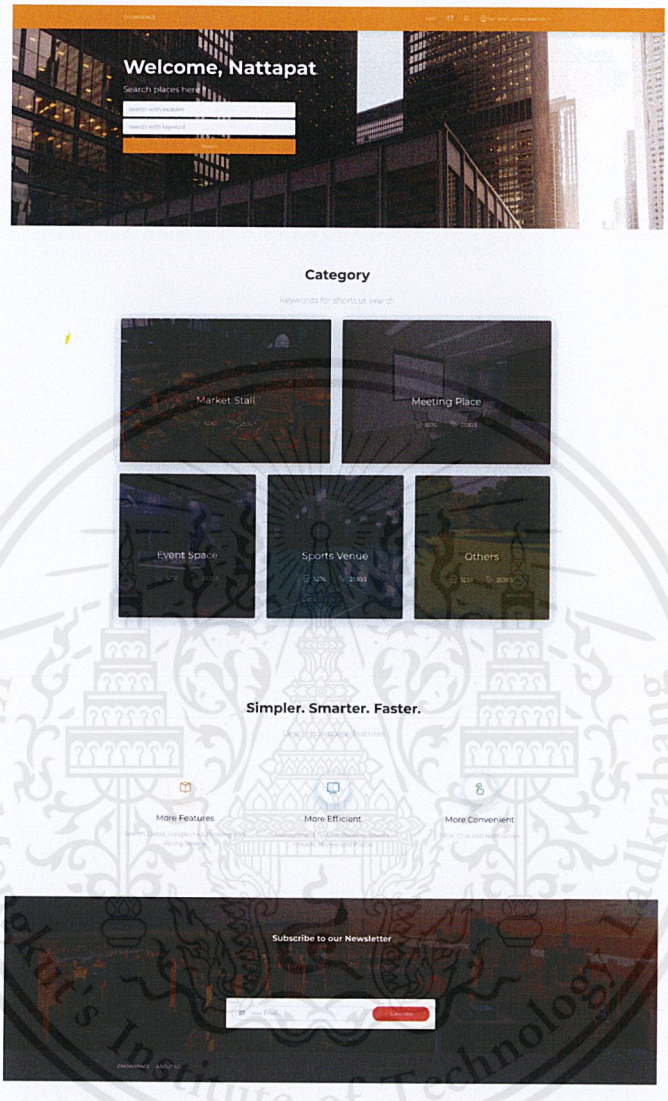

Figure 6.1: Home page

Home page shows the overview of what the system provides for the users. It consists of the search functionality where users can search the place they are interested in. This page also provides the category of the place that they are interested in to narrow down the place they are looking for. After filling in the information or selecting the category, the user will be linked to the search page that shows the result of the information that has been filled in by the user.

6.1.2 Login Page

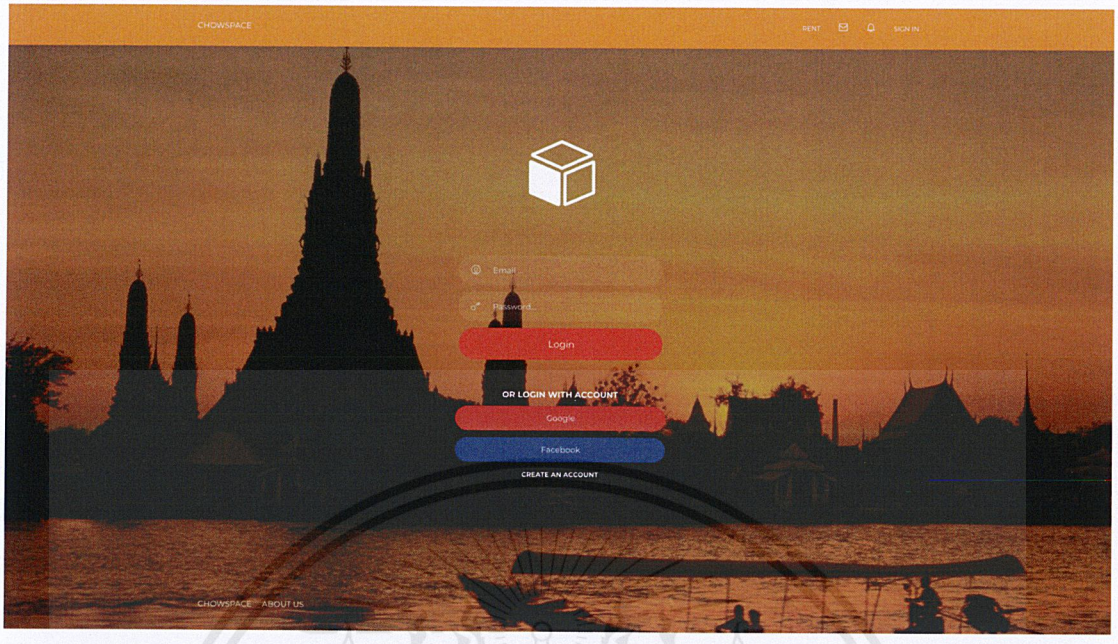

Figure 6.2: Login page

Login page lets a user to login into the system then the user will receive permission into the management system.

6.1.3 Create An Account Page

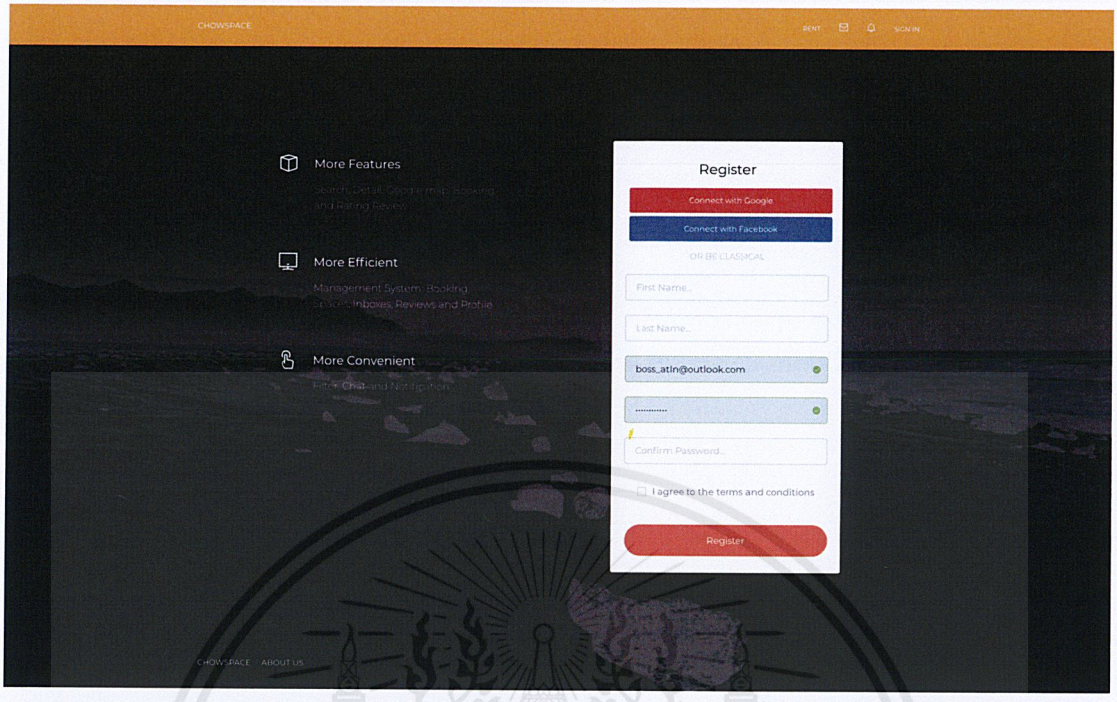

Figure 6.3: Create an account page

This page requires information of the user to create an account.

6.1.4 Create New Place Page

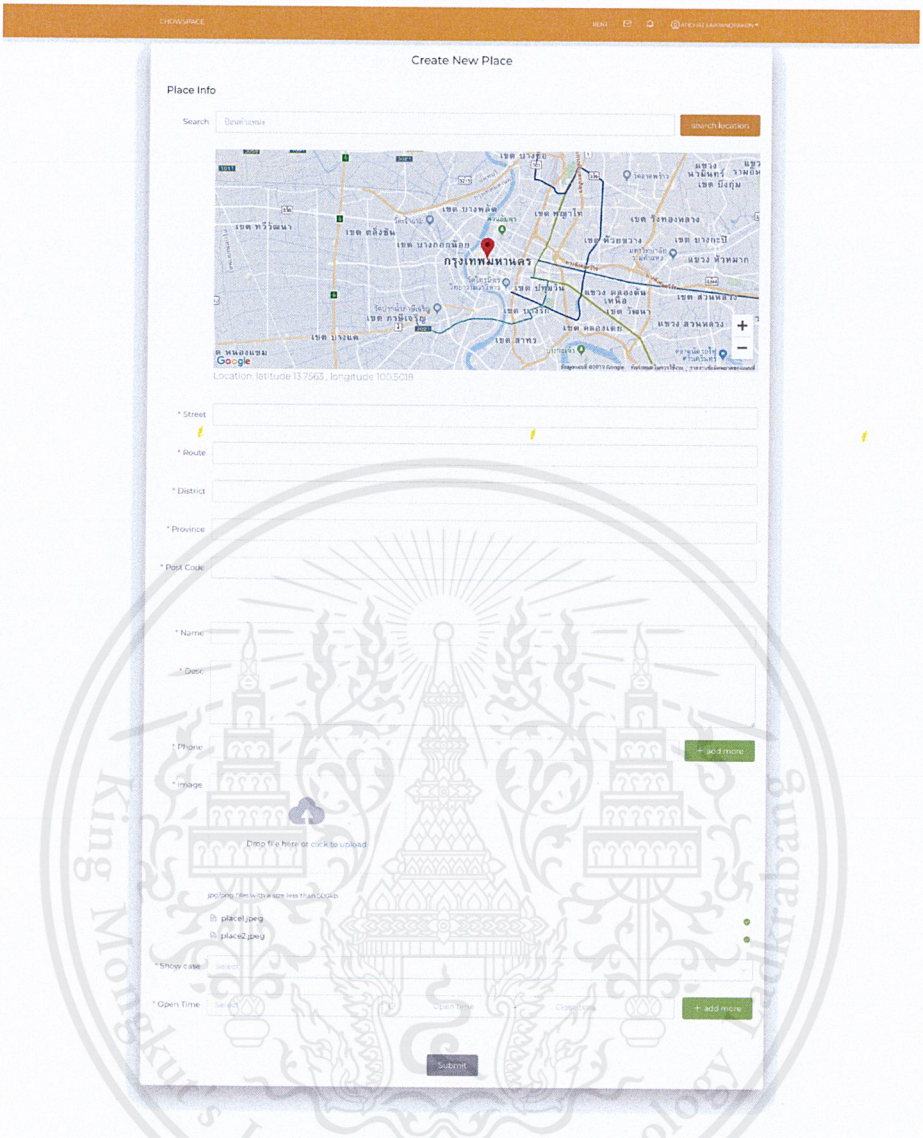

Figure 6.4: Create new place page

This page requires information of the place to be filled in by the owner of the place. This information is utilized by the admin of the system to verify the validity of the place. An example of information that the owner needs to fill in are the name of the place, address, contact, and images of the place. After submitting, the place will be published into our system immediately but they have to wait for the admin to verify the validity of the information. If the admin accepts, your place will be given the verified badge from the system.

6.1.5 Create New Space Page

Figure 6.5: Create new space page

This page requires information of the space to be filled in by the owner of the space in the place. Example of information that the owner needs to fill in are the name of the space, price, and image. After submitting, your space will be available to rent immediately.

6.1.6 Search Page

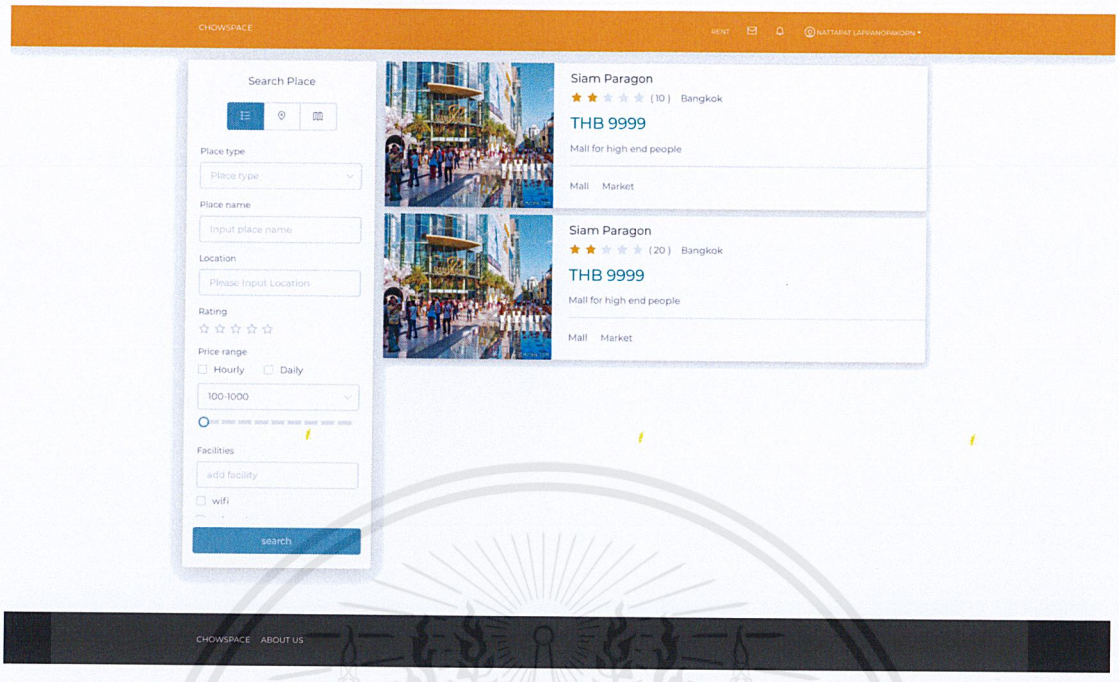

Figure 6.6: Search page

Search page let the users search for the place in the system in more specific detail than the Home page. This allows the user to specify the place that they are looking for more accurately. After finishing the search, list of all the result matching the information will be displayed.

6.1.7 Place Detail Page

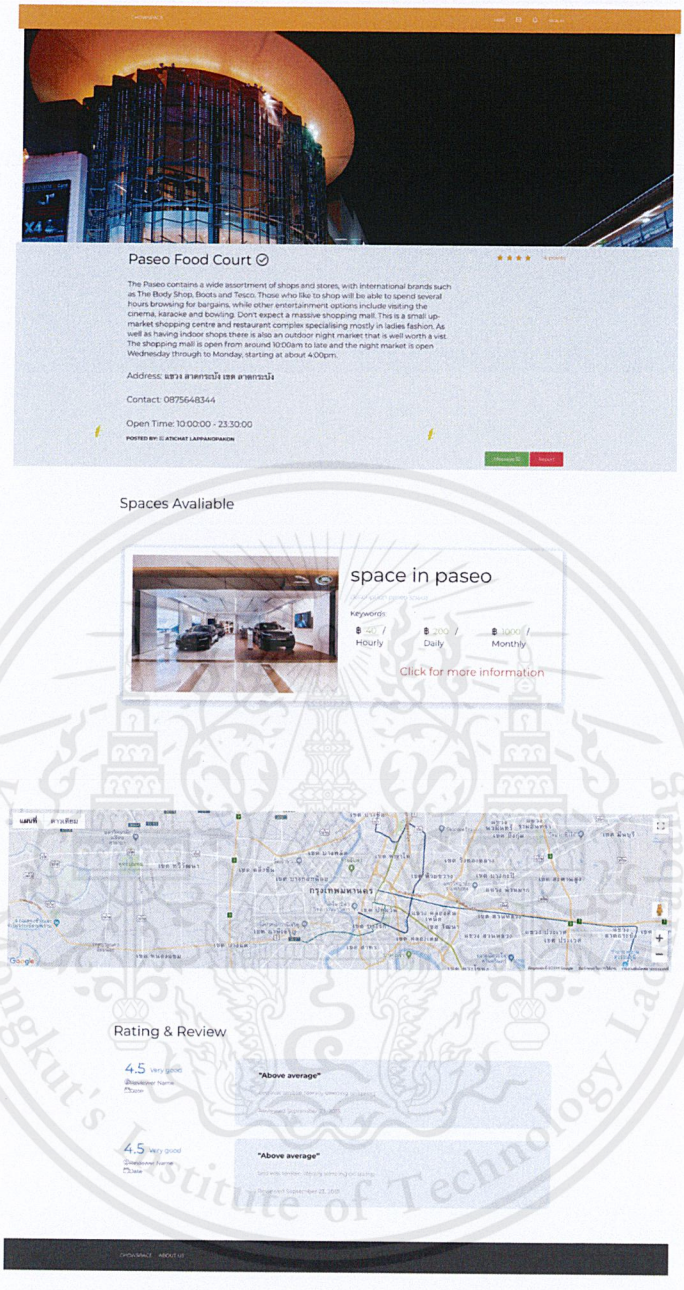

Figure 6.7: Place detail page

Upon selecting the place from the search page, it directs the user to the place detail page. In place detail page, it provides all the information of the place that is selected and all the available spaces in the place, also the location(map), images, rating and reviews from other users of the place.

6.1.8 Reservation Page

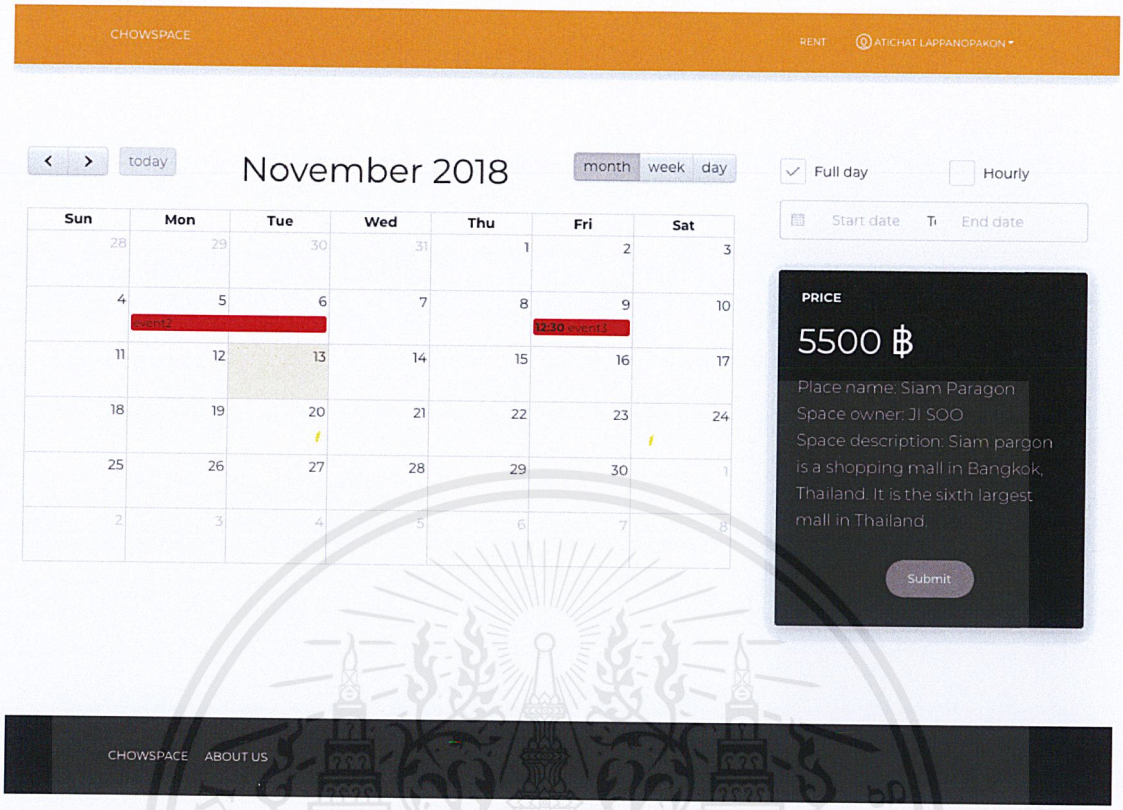

Figure 6.8: Reservation page

Upon selecting to rent the space from space detail page, it directs the user to booking page where user can view all the available date or hour of the space and the price that are calculated from the range of date or hour that the user picks. This page consist of a calendar, contact information of the owner, space description and price result. After finish renting, your reservation result will be shown in reservation management system page.

6.1.9 Places Management System Page

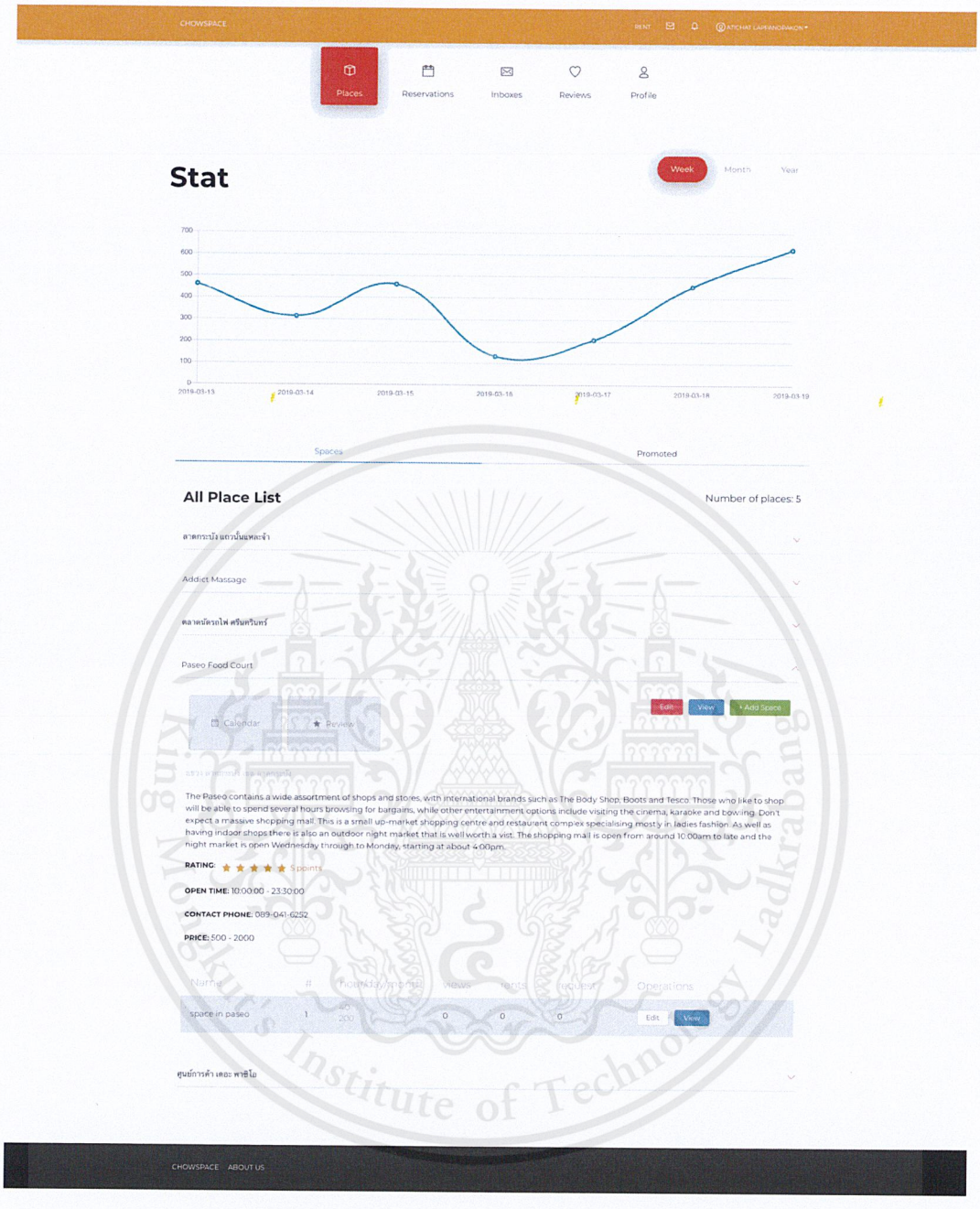

Figure 6.9: Place management system page

This page shows a statistic of all place in the list of the owner and provides a lot of features that allow the owner to manage place and space such as edit place, add new space, calendar and review shortcut.

6.1.10 Reservations Management System Page

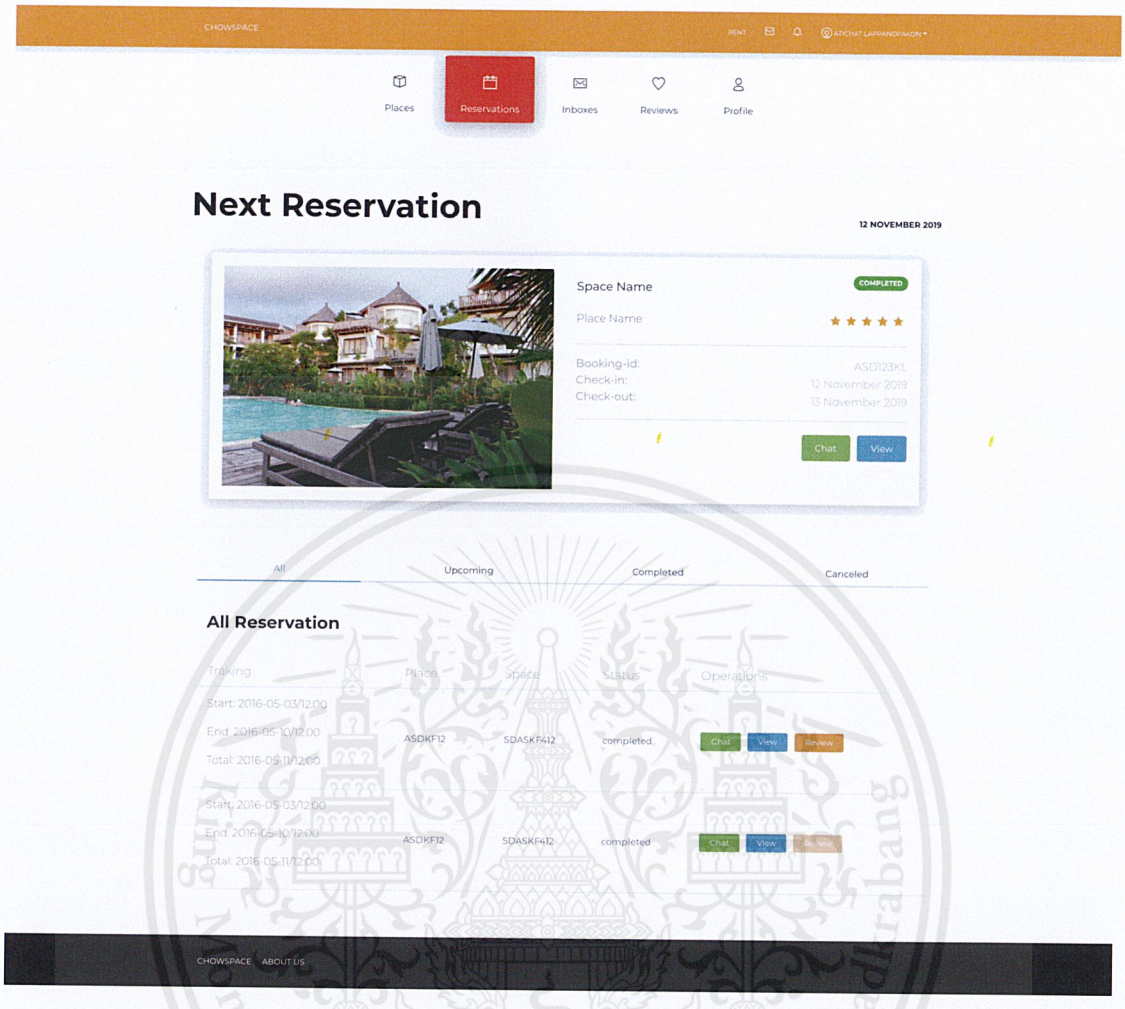

Figure 6.10: Reservations management system page

This page shows a list of reservation of a user reserved and this page also provides filter and shortcut to chat with the owner of that place, to view the receipt and to review that place.

6.1.11 Inboxes Management System Page

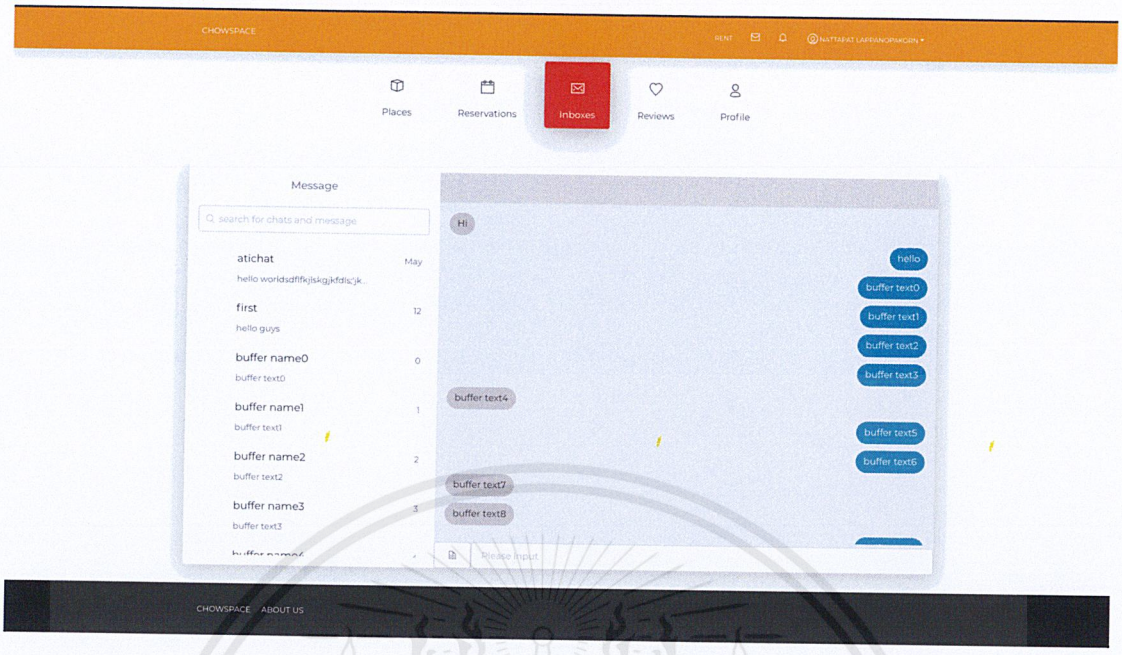

Figure 6.11: Inboxes management system page

This page lets a user communicate with others in our platform that provide a clear convenient channel between owner and renter.

6.1.12 Review Management System Page

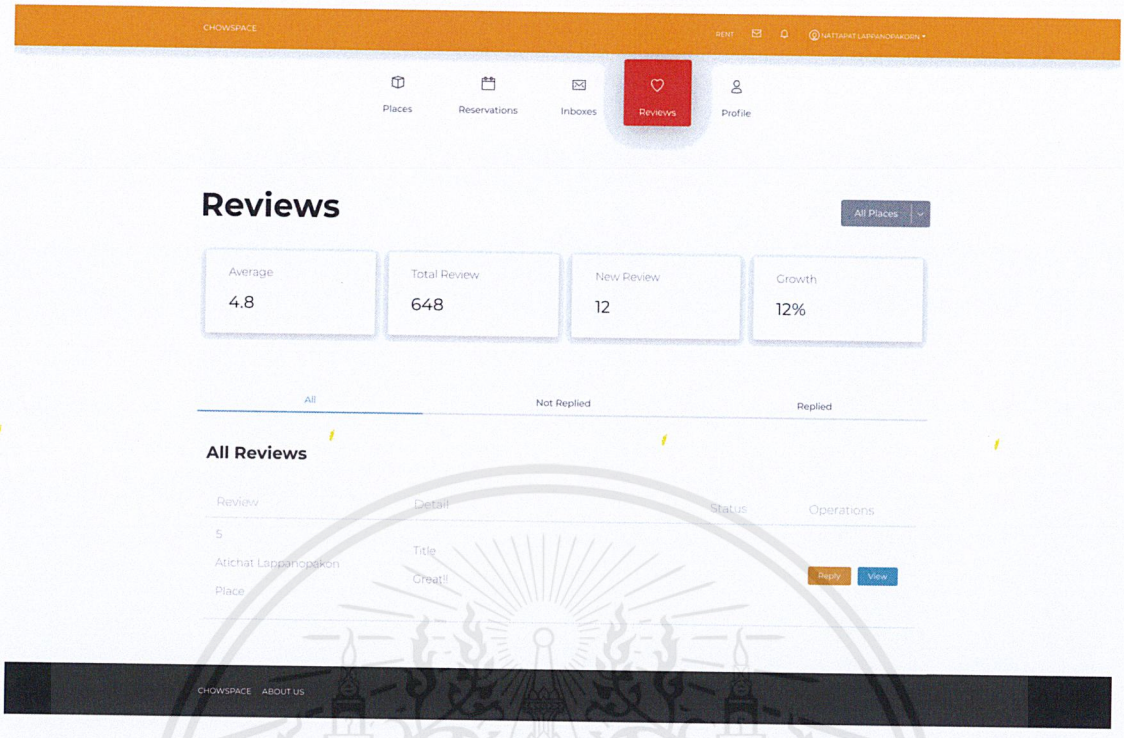

Figure 6.12: Review management system page

This page shows essential statistic number that lets the owner monitor their post easier and this page also show review on their places that allow owner can reply back to the renter in that post.

6.1.13 Profile Management System Page

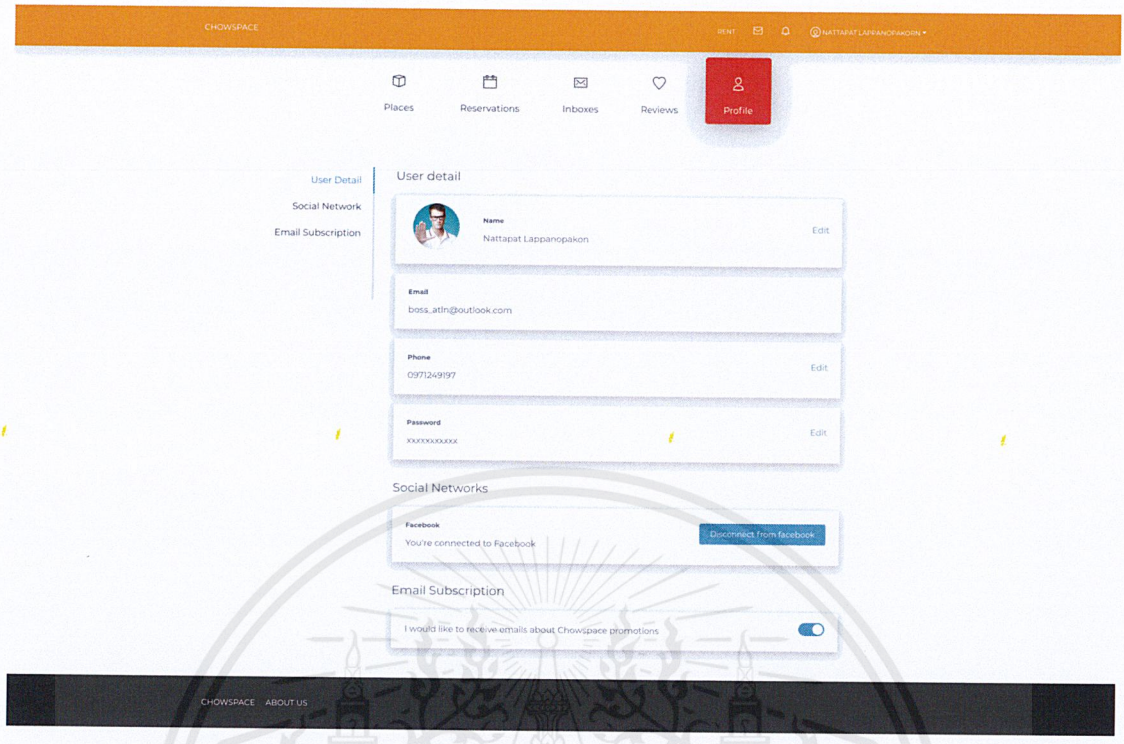

Figure 6.13: Profile management system page

This page shows user information and allows user can update their information which includes name, password, profile picture and contact information.

Chapter 7
Conclusion

This thesis proposes a renting platform website for both owner and renter that focuses on short term renting such as market stall, event space, meeting place and others. The system is based on a web application that allows user to rent a place or publish their own available space to our platform and this system will work as a middle man between renter and owner by advertising the space published by the owner of the space and providing information for renter who looking for a place to run their business. By using this platform, both the owner and renter life will be easier with this convenient way of renting.

The thesis's main contribution is the microservice architecture which composes of 9 services (authentication & user management, place management, reservation, review, notification, chat, email, reporting, storage). Each service is working individually by itself so if one of these nine services is down the systems can continue working with other services that still running. With microservice architecture design, it makes demanding services can be deployed in multiple servers to enhance performance in high scalability. Since each service is working individually, it has less chances that are going to impact other services.

7.1 Future Work

7.1.1 Authentication & User Management Service

At present, authentication & user management service is using our authentication only which might not be convenient to users that they have to fill in their information by themselves. So the suggestions were made to using other social login API. The examples of social login API that can be integrated with the system are Google, Line or Facebook that they already provide the information of the user already on their API.

7.1.2 Reservation Service

Currently, the reservation service can not do the payment yet which might make concern for the user who wants to do the payment through our platform. In the future, the system should provide the payment features to make the reservation of the renting process become more powerful.

Bibliography

[1] Chris Richardson. "Introduction to Microservices". [Online]. Available: https://www.nginx.com/blog/introduction-to-microservices [Accessed: 12-February-2019].

[2] Bhagwati Malav. "Microservices vs Monolithic architecture". [Online]. Available: https://medium.com/startlovingyourself/microservices-vs-monolithic-architecture-c8df91f16bb4 [Accessed: 12-Februrary-2019]

[3] Chris Richardson. "Building Microservices: Using an API Gateway". [Online]. Available: https://www.nginx.com/blog/building-microservices-using-an-api-gateway [Accessed: 12-February-2019].

[4] Chris Richardson. "Building Microservices: Inter-Process Communication in a Microservices Architecture". [Online]. Available: https://www.nginx.com/blog/building-microservices-inter-process-communizcation [Accessed: 12-February-2019].

[5] Chris Richardson. "Service Discovery in a Microservices Architecture". [Online]. Available: https://www.nginx.com/blog/service-discovery-in-a-microservices-architecture [Accessed: 15-February-2019].

[6] Chris Richardson. "Event-Driven Data Management for Microservices". [Online]. Available: https://www.nginx.com/blog/event-driven-data-management-microservices [Accessed: 15-February-2019].

[7] Chris Richardson. "Choosing a Microservices Deployment Strategy". [Online]. Available: https://www.nginx.com/blog/deploying-microservices [Accessed: 15-February-2019].

[8] Evan You. "What is Vue.js?". [Online]. Available: https://vuejs.org/v2/guide/#What-is-Vue-js [Accessed: 12-November-2018].

[9] Evan You. "Evan is creating Vue.js". [Online]. Available: https://www.patreon.com/evanyou [Accessed: 12-November-2018].

[10] Wikipedia contributors. "Vue.js". [Online]. Available: https://en.wikipedia.org/wiki/Vue.js [Accessed: 12-November-2018].

[11] Node.js contributors. "Node.js dev". [Online]. Available: https://nodejs.dev/ [Accessed: 17-March-2019].

[12] Node.js contributors. "Node.js with microservices". [Online]. Available: http://www.tothenew.com/blog/microservices-and-node-js/ [Accessed: 17-March-2019].

[13] Docker & Kubernetes contributors. "Docker & Kubernetes". [Online]. Available: https://www.quora.com/What-are-Docker-and-Kubernetes-What-are-they-used-for/ [Accessed: 17-March-2019].

[14] NGINX contributors. "NGINX". [Online]. Available: https://www.nginx.com/resources/glossary/nginx/ [Accessed: 17-March-2019].

[15] RabbitMQ contributors. "RabbitMQ". [Online]. Available: https://medium.freecodecamp.org/rabbitmq-9e8f78194993 [Accessed: 24-May-2019].

[16] PostgreSQL contributors. "About PostgreSQL". [Online]. Available: https://www.postgresql.org/about/ [Accessed: 12-November-2018].

[17] MongoDB contributors. "MongoDB". [Online]. Available: https://www.techopedia.com/definition/30340/mongodb [Accessed: 17-March-2019].