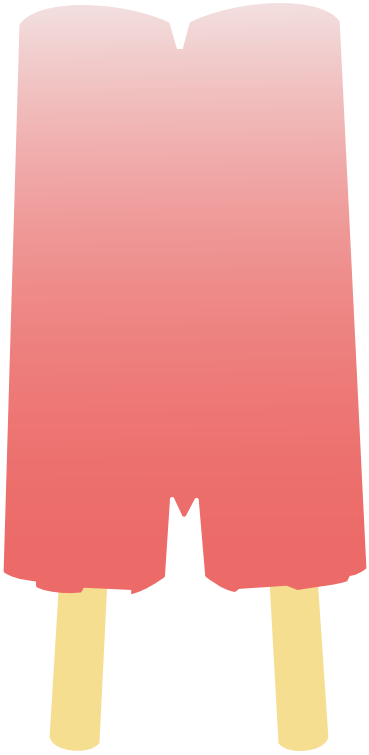


Final Design Report

CSCI3100 - Group: E6 - mATE

Ao Kin Pong 1155096429
Hui Lok Hin 1155133188
Lam Pui Hong 1155093597
Lao Kam Fung 1155096606
Wong Chun Wai Tom 1155109240

Table of Contents

1. Introduction

1.1 Project Overview

mATE is a social networking dining application specifically designed for optimal restaurant sharing and communication experience.

Combining restaurant searching functions and social networking functions into one, mATE allows users to create and build their very own personal network with reliable restaurants and friends, in which they can discover, participate in, and share their dining experience in various restaurants. By a focus on photos, tags, and ratings, mATE also ensures it is not the companies or restaurants that define the presentation of themselves, but the users, thus creating a truly user-based environment that users can rely on to build their own circle of desirable restaurants.

mATE undoubtedly provides one of the best restaurant-searching and sharing experience among all the restaurants-rating app currently in the market.

1.2 Objective

mATE is all focused in user experience. We strive our best to create a normal restaurant-searching/rating application with a twist of social networking experience. Hence, there are two main objectives:

1) between users and restaurant, to allow users search, discover, and rate restaurants
2) inter-users, to allow users to communicate and share said restaurants – or even users, depends on what the hidden gems are – to other users.

For the former, mATE aims to provides a page where hot and popular restaurants and users are recommended to users. Users can search for entities by specifying fields like name, types, tags, and location; upon found, posts can be created with respect to the restaurant and are visible to all parties.

For the latter, mATE aims to provides a feed where recent activities, i.e., posts or shares by other users, are displayed to users. Relevant notifications are sent to users, who can then receive information from their trusted partners to slowly build up their own network of restaurants that suits their taste.

Aside from the above, restaurant creation and claiming ownership of restaurants are provided in cases where restaurants are not in the database yet.

1.3 Expected Customers and Market

Our target customers would be 15-40 years old Internet generation who love spending time finding delicious food and their favourite restaurants. They are not content with the current searching system of existing restaurant finding and reviewing applications, deeming them full of anonymous reviews and unable to correctly reflect the restaurants pros and cons.

Meanwhile, they find the current social network, as a substitute for searching in the dining apps, provides too little information on restaurants and too vague to use. As pleasing as the photos and information they find in the network, when they stumble onto a potential restaurant and wish to know more reviews and photos, they have to search again, transfer to another application, or use online searching engine.

mATE aim to target these enthusiasts who wish to get the benefits of both platforms, while share and gain these knowledge, places, and dishes to and from a community they built around (free promotion!).

1.4 System features and Interfaces

mATE has a general interface of a social network application with a twist on what the users can access. There are two types of entities: users and restaurants.

Users 1) have their own profile, 2) can like and comment posts, 3) can follow and share other entities; while restaurants only have their own page that users can follow and save.

mATE has in total five main features:

1) **Activity feed**: All recent activities – namely restaurants reviews, restaurant/user sharing, and restaurant check-ins – of followed entities are displayed. Users can interact with these posts as in normal social networks.

2) **Search and Discover**: It displays popular restaurants and users in the area. If users decide to search, the page will display the searching result.

3) **Profile**: It displays the user's own information and posts which users can edit. When there are new activities or interactions from following entities, it notifies the user (if they desire to be notified).

4) **Post or Review Creation**: Users can choose to write a review for a restaurant or check into a restaurant. If the restaurant they desire to review are not included in the server, they can create it, waiting for owners to claim (to further edit restaurant information).

5) **Reservation**: Users can make reservation on restaurants' page.

2. Background

Currently there are only two main restaurant-searching and rating applications in Hong Kong. These applications allow people to search for specific cuisines in certain area to find their desired restaurants. Often, users are required to go through a lot of comments to determine whether the restaurant is up to their expectation, let alone what food or dish is the best the restaurant can provide. Worst still, these comments are made by anonymous people where a portion of which might be paid to write opinions that are in favor for the restaurants.

Meanwhile, an increasing number of people, ranging from teenagers to middle-aged adults, start to switch their attention to social media and use their tagging function. An array of photos is presented to the users, providing them much more insight than comments. Embedded with description and posted by accounts that users can engage and verify, these posts created another platform achieving the same purpose (albeit not being the intended use of the application). However, highly customizable tags simply mean there are too much to choose from; a lot of information such as location and dish type are condensed into a single hashtag – the only one user can search, with no additional information or filters. To acquire more information, comments, and reviews of that restaurant, another queries, again with vague searching methods and troublesome to user, are required.

All in all, the current restaurant-searching applications fail to let users engage in a network, while social networks fail to provide a suitable follow-up on the enquiry of restaurants.

Thus, mATE is created.

It serves the function of both applications above and offers more. Search has never been easier: for one, user can still search normally by location, cuisine, and name; on top of that, user can search by multiple tags (including but not limited to description, food types, and price) and filter to find the food they so wanted. Restaurants only consists of basic information and the rest are reviews of the restaurants. Within the reviews centered in photos and tags, users are provided with visual representation of the food to decide their view towards the restaurant without having to read much description – an option nevertheless that are still available to the user.

Implementation of a social network encourages user to find entities of their interest. No matter with friends or families, users can follow their accounts and check if any interesting restaurants they have recently visited; then, one can head to the restaurant info page in one click, check the reviews, and add them to their list if one desires. Gradually, users create their own circle of restaurants and users, discovering new places to dine in while encouraging others to explore more.

No more anonymous comments, empty accounts, or paid reviews when all the users' reviews are available for one to check, all while maintaining a trustworthy network to share personal experiences or simply a check-in.

3. System Architecture Design

3.1 System Architecture

3.1.1 Registration

Collecting users' username, email and password. mATE will send an email to welcome every new user and add the account data to the backend database.

3.1.2 Sign In

Users can login by using their username and password. Then mATE will check if the account exists and check if the password corrects in the database.

3.1.3 Change Personal Information

Users can change their profile picture and nickname in the profile page.

3.1.4 Search Entities

Users can input string in the search bar to search for users or restaurants. The system will list out the matched results.

3.1.5 Un/follow Entities

Users can follow other users or restaurants. When the users you followed open a new post or publish a new review, it will show on your main page.

3.1.6 Add Post

Users can check-in or leave reviews for the restaurant. A review includes rating, comment and hashtag.

3.1.7 Like and Comment

Users can like, dislike, and leave comments on other users' posts or reviews.

3.1.8 Explore

Up to 10 random restaurants will be represented in the explore page.

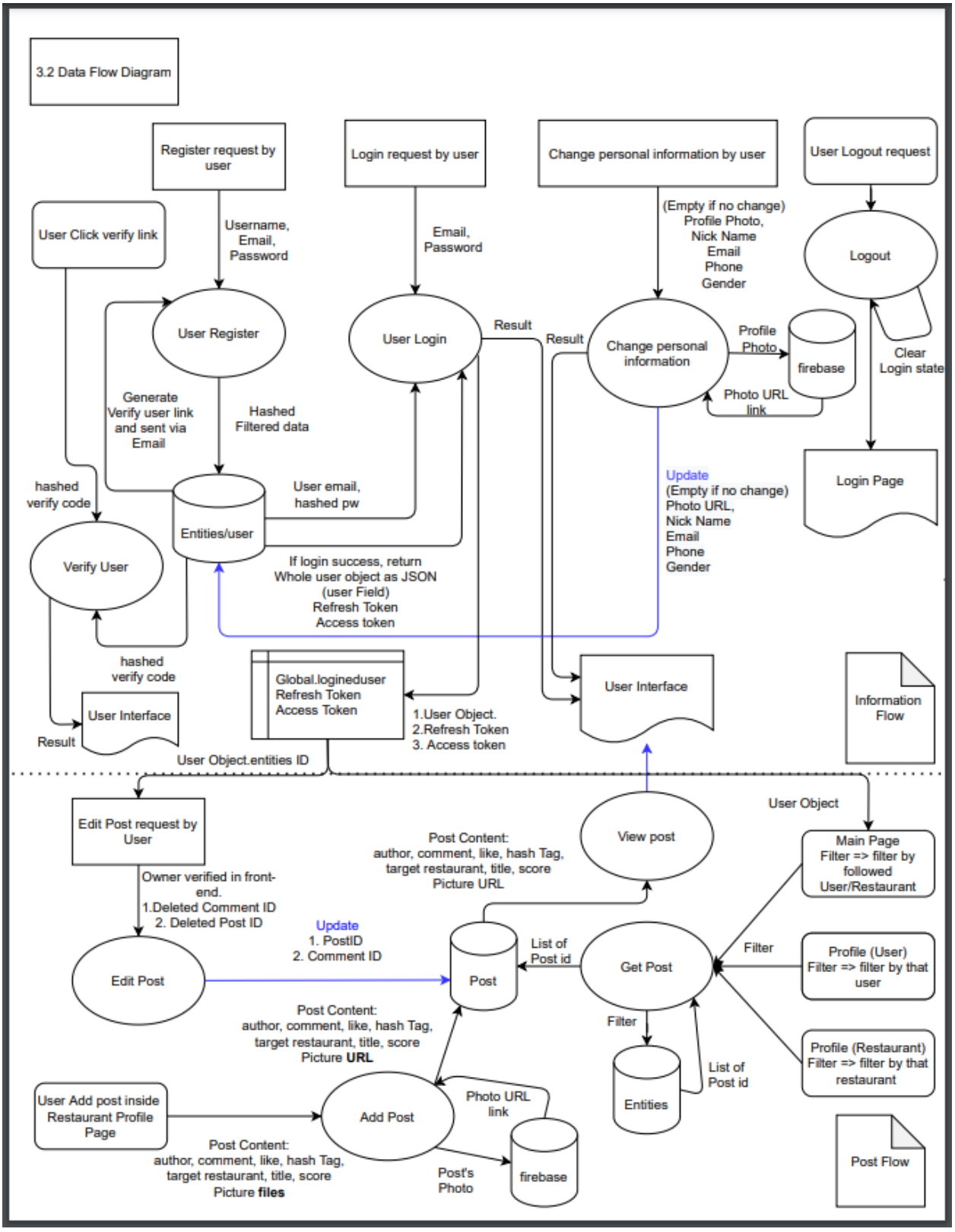
3.1.9 Logout

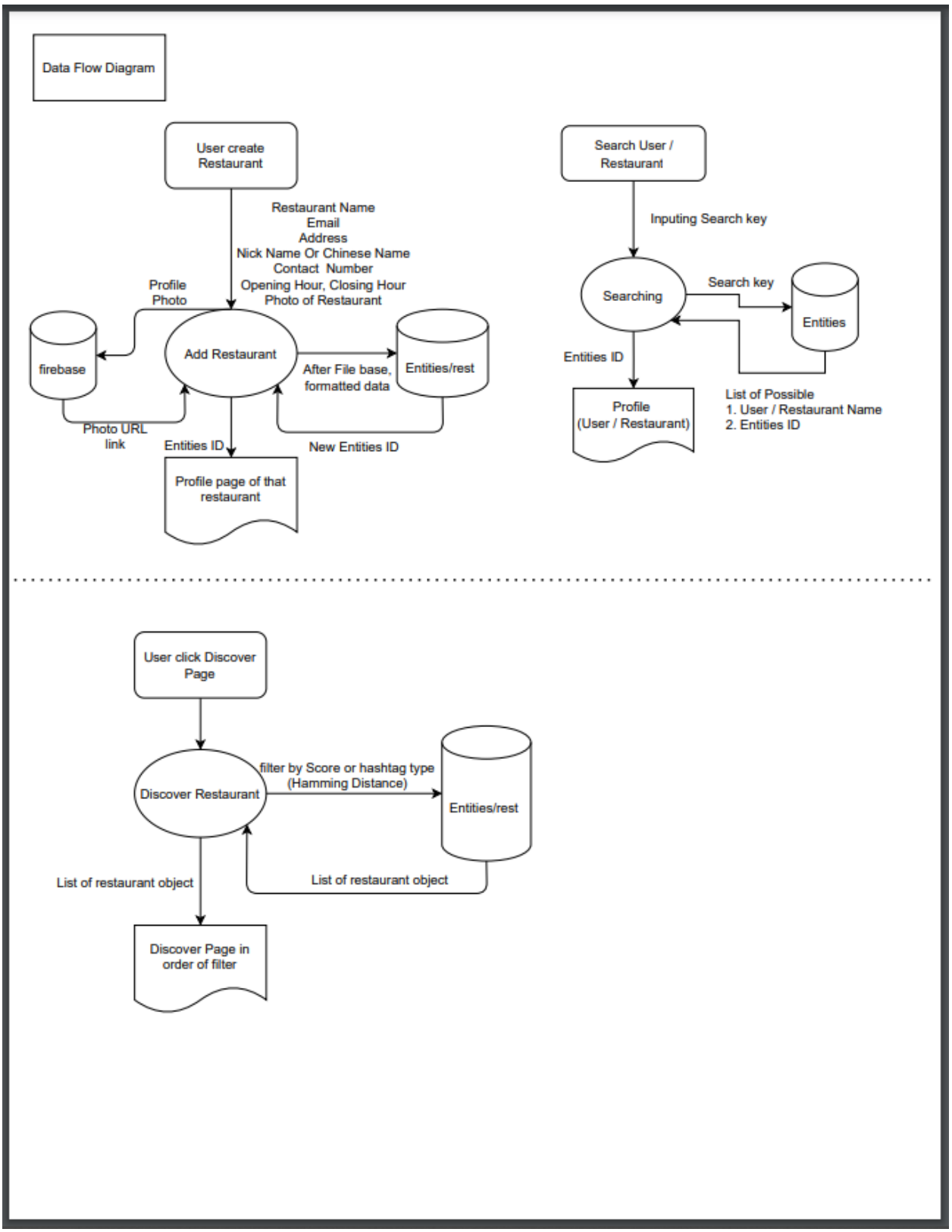
Users can logout their accounts.

3.1.10 Reservation

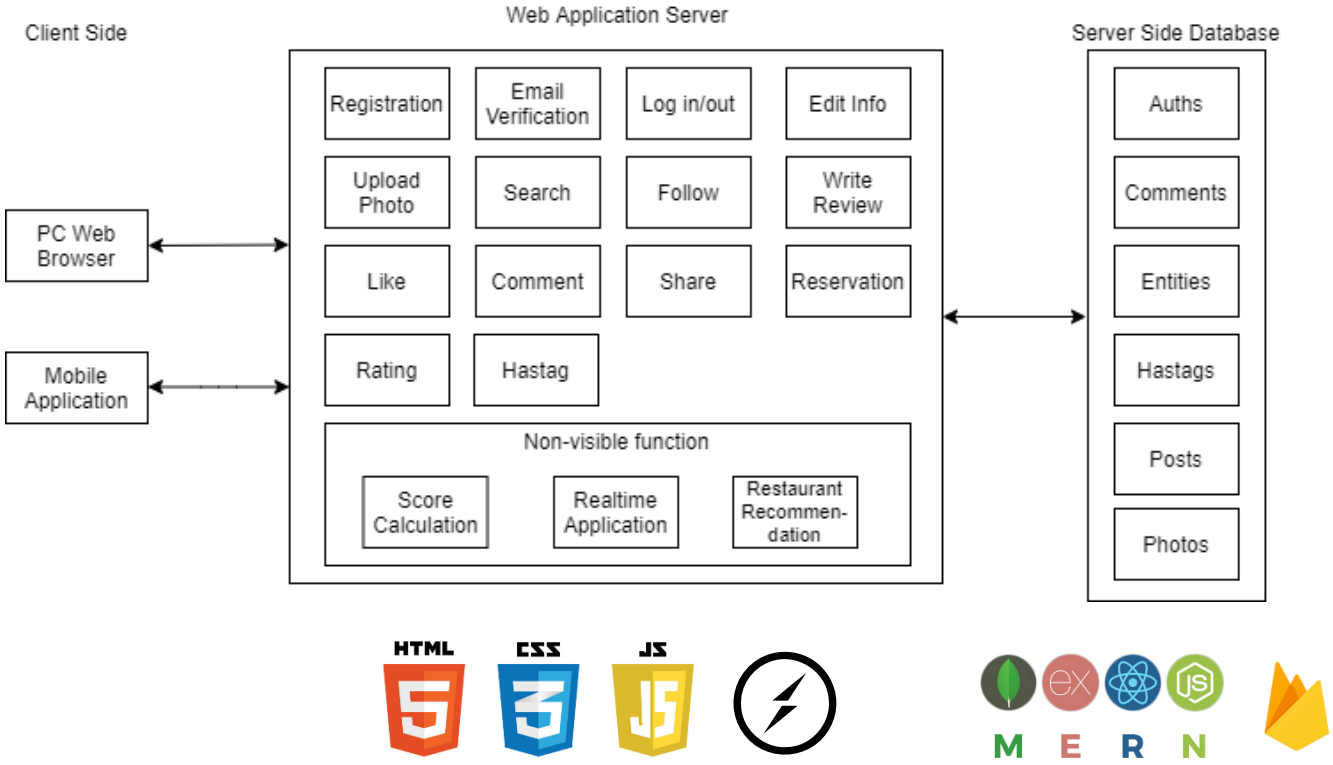
Users provide relevant information: arrival time, username, Phone number, Remarks. System will Sent email to both restaurant and user for Reservation

3.2 Data Flow Diagrams (DFDs)

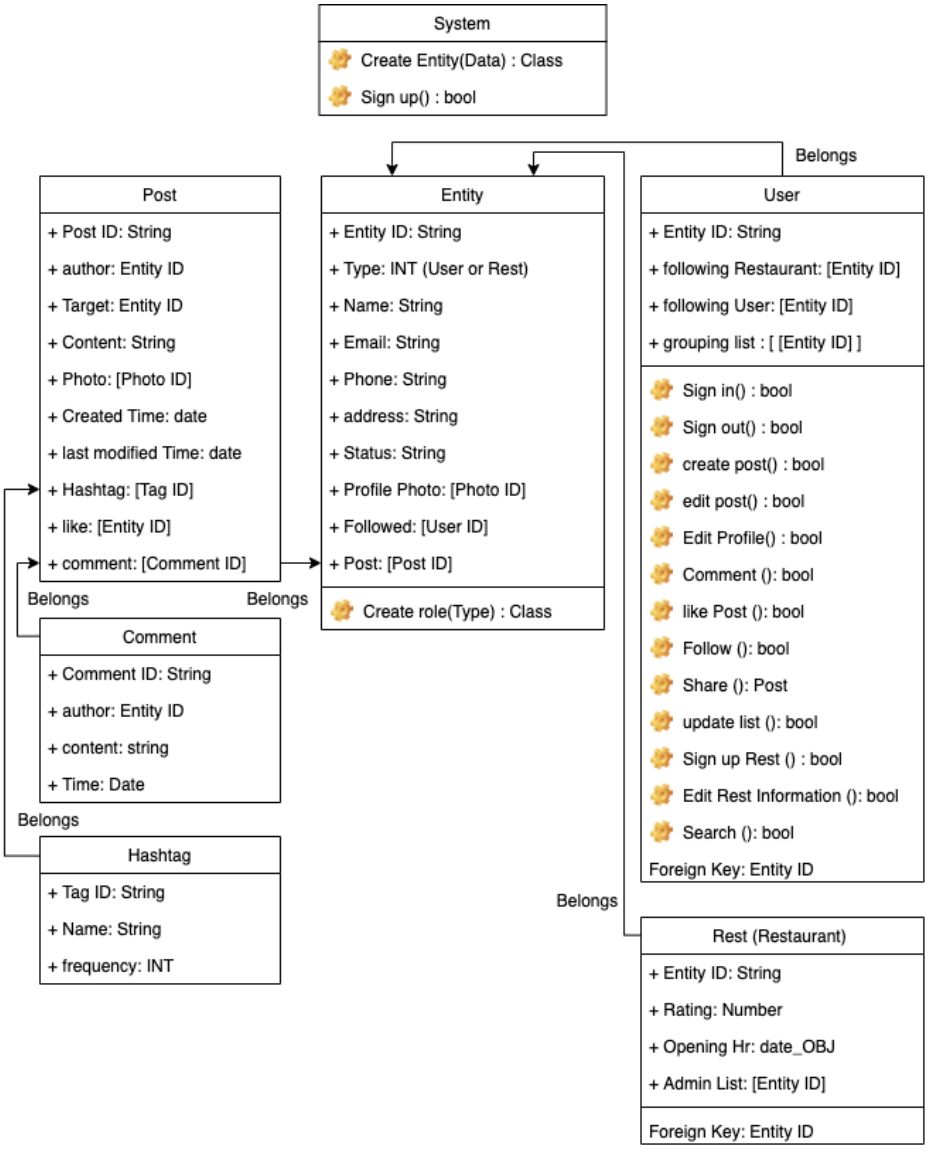



Data Flow Diagram
User create Restaurant
Restaurant Name
Email
Address
Nick Name Or Chinese Name
Contact Number
Opening Hour, Closing Hour
Photo of Restaurant
Profile Photo
firebase
Add Restaurant
After File base, formatted data
Entities/rest
Photo URL link
Entities ID
New Entities ID
Profile page of that restaurant
Search User / Restaurant
Inputing Search key
Searching
Search key
Entities
Entities ID
Profile (User / Restaurant)
List of Possible
1. User / Restaurant Name
2. Entities ID
User click Discover Page
Discover Restaurant
filter by Score or hashtag type (Hamming Distance)
Entities/rest
List of restaurant object
List of restaurant object
Discover Page in order of filter

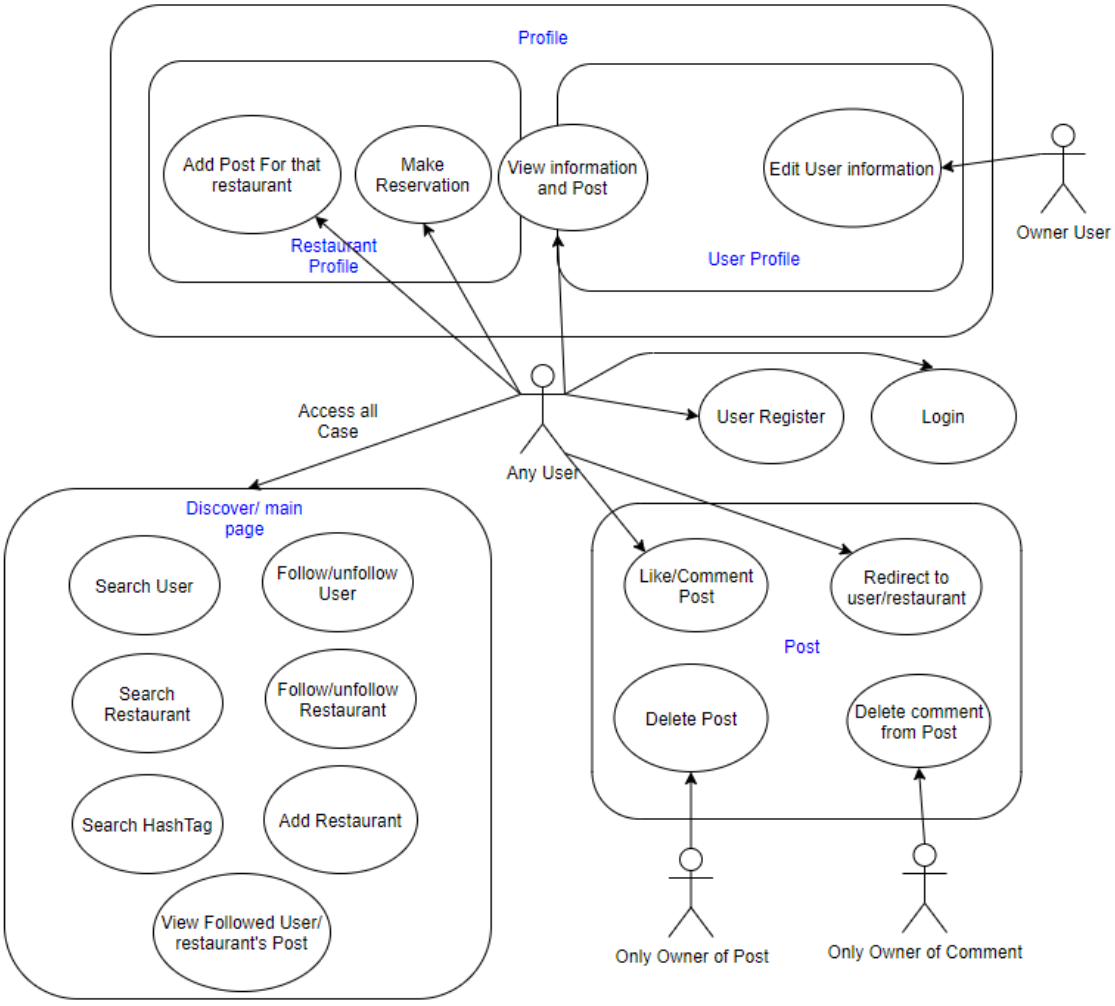
3.3 Architecture Diagram



3.4 Class diagram

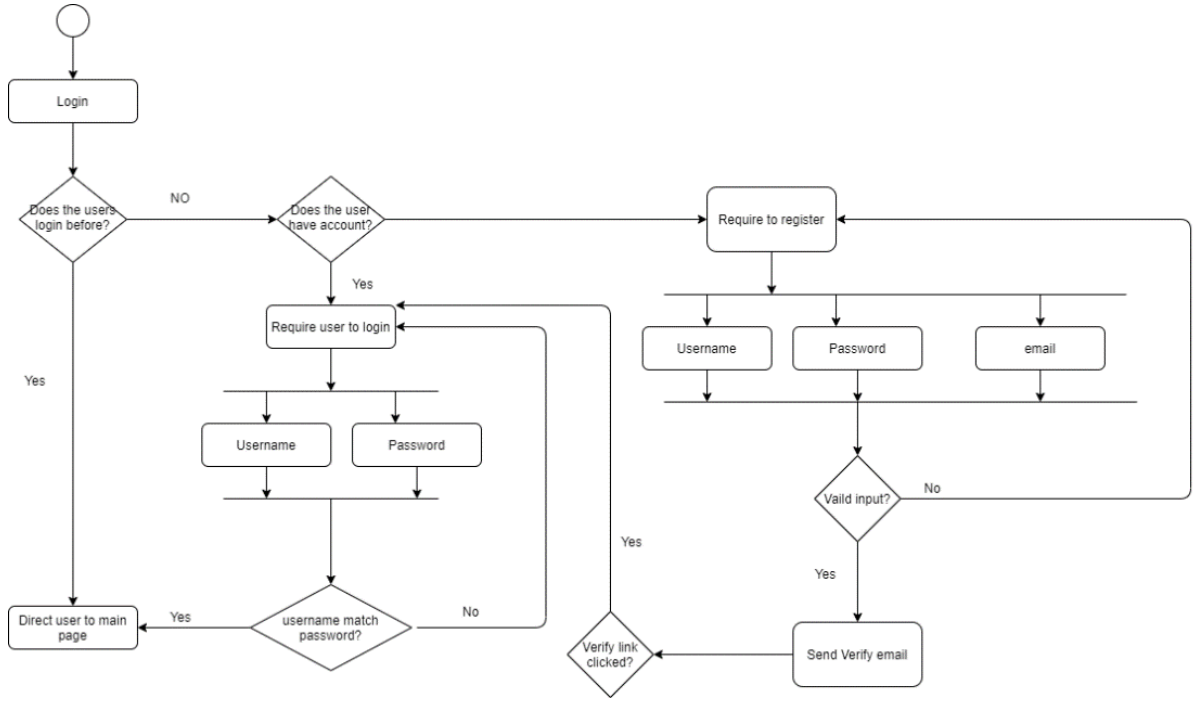


3.5 Use Case Diagram



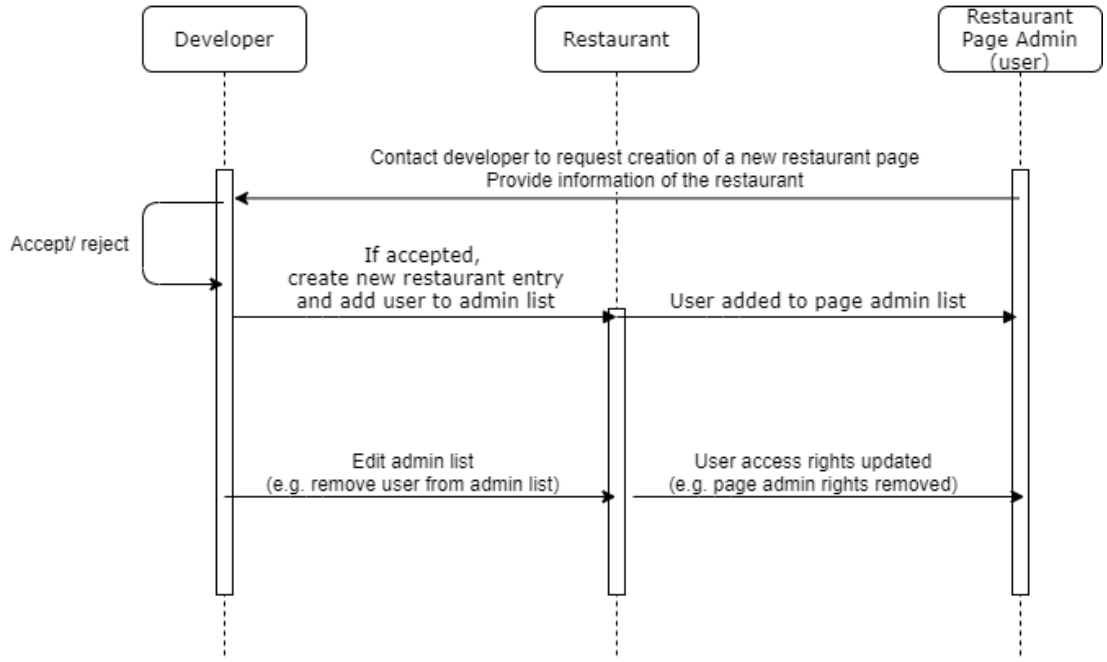
4. Components Description

4.1 Login and Sign-in system



The above UML diagram showing the registration and login procedures. All users have to login their own account in order to access mATE platform. Users that do not have an account will be directed to create a new one. To create a new account, users are required to provide username, password and email. Verification code will be sent to the user's email for registration purpose. User with an existing account only needs to provide username and password to login. Only logged in users will be directed to the main page.

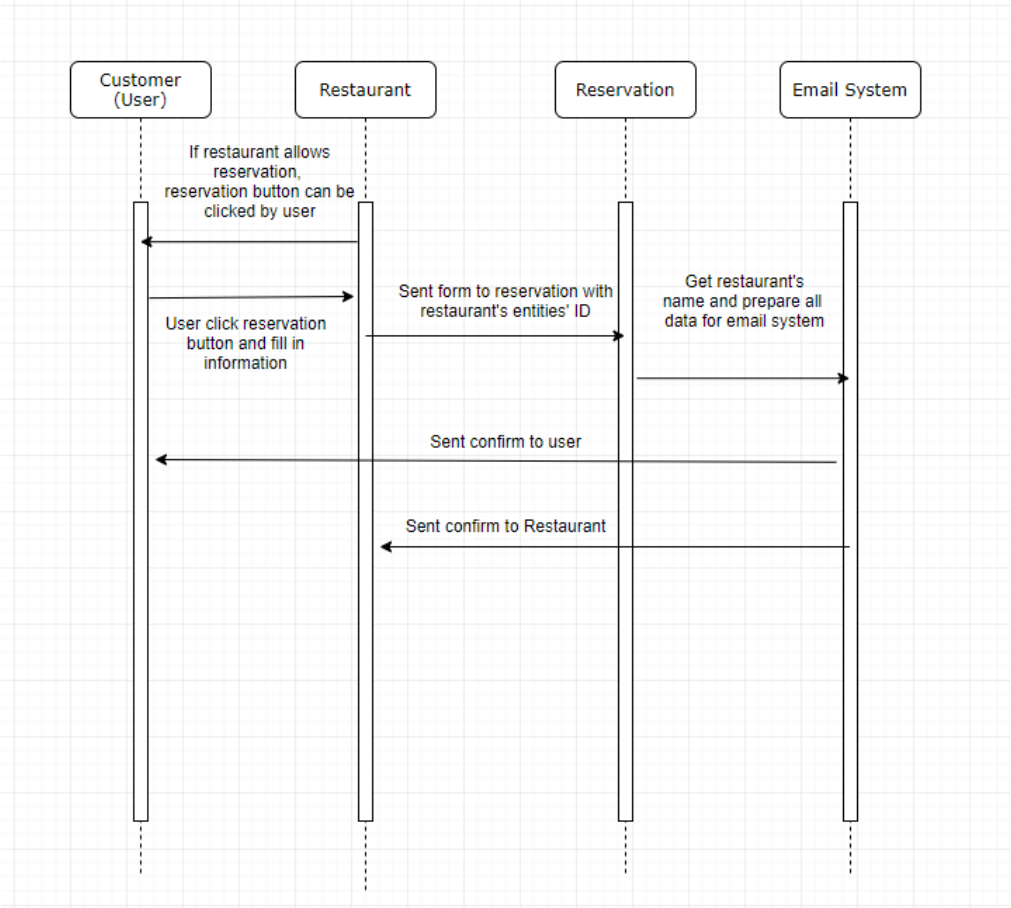
4.2 Creating a new restaurant page



The above UML diagram showing the procedures of how a restaurant owner, which originally is a normal user at the web page, can create their own restaurant page in mATE. First the owner has to send the request to the platform developer and provide relevant information of the restaurant (location, phone number) to the developer. The request will be processed by the developer. If accepted, a new restaurant page will be created and the restaurant database will be updated accordingly. The account of the restaurant owner will be added to the admin list of that restaurant page, so that they can manage their own restaurant page.

The platform developer reserves the right to edit/remove the restaurant page at any time.

4.3 Reservation:



The above figure shows the procedures of making an online reservation at a restaurant.

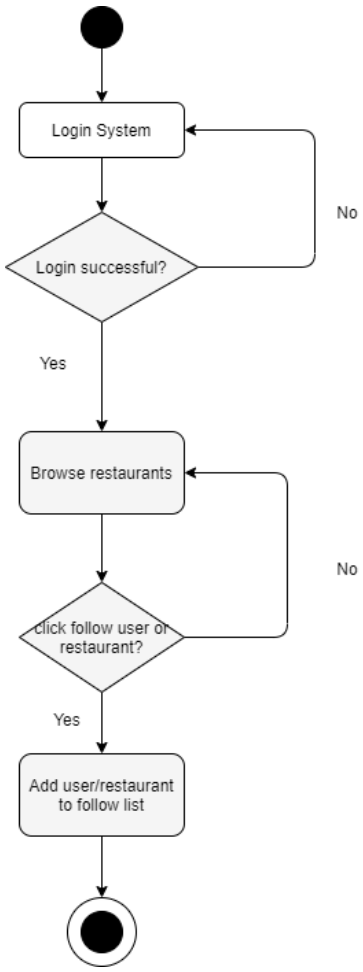
First the user opens the page of the restaurant, where they can find the section for making reservation.

To make a reservation, users provide relevant information (arrival time, username, Phone number, Remarks, etc) at the reservation page inside restaurant, which will be sent to the reservation system.

The reservation request will be processed, and system will get restaurant's name and sent to Email system.

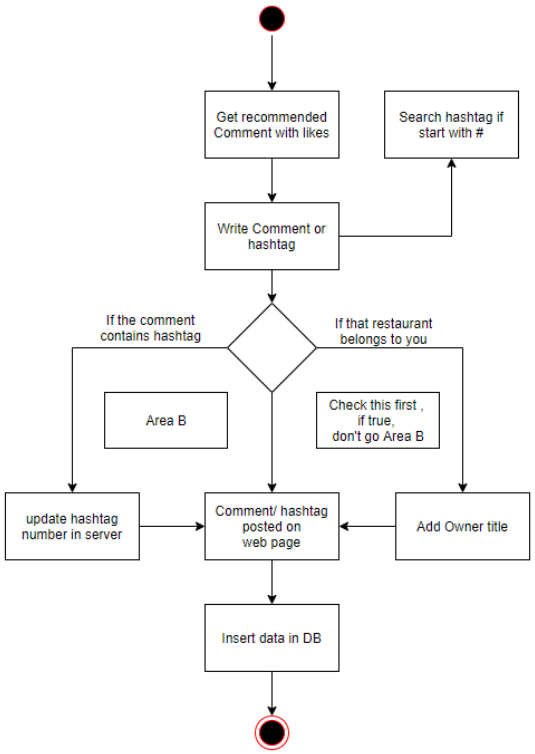
After that, Email system will send email to both user and restaurant's email.

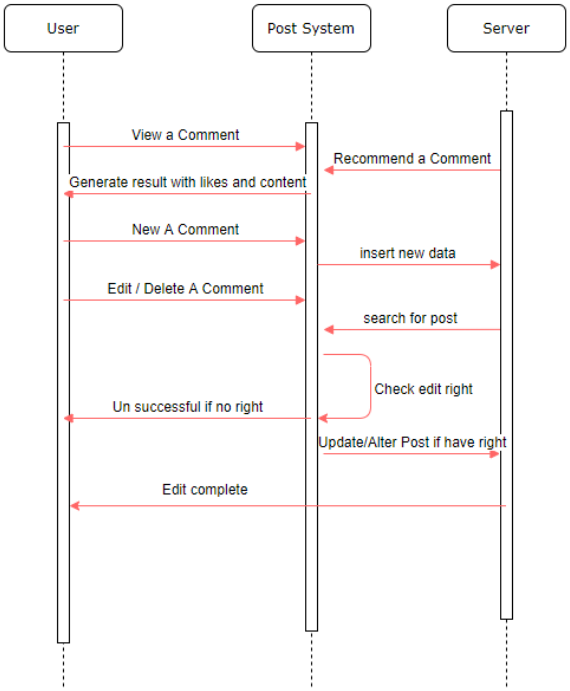
4.4 Follow Restaurant/ User



The above UML diagram showing how a user can add a restaurant to their favorite list. First, the logged in user can browse the restaurant pages to look for the restaurant they like. If the user would like to add the restaurant to their favorite list, they can click the button “Follow”. Restaurants that are in the favorite list will be displayed to the user at the section “Followed List”.

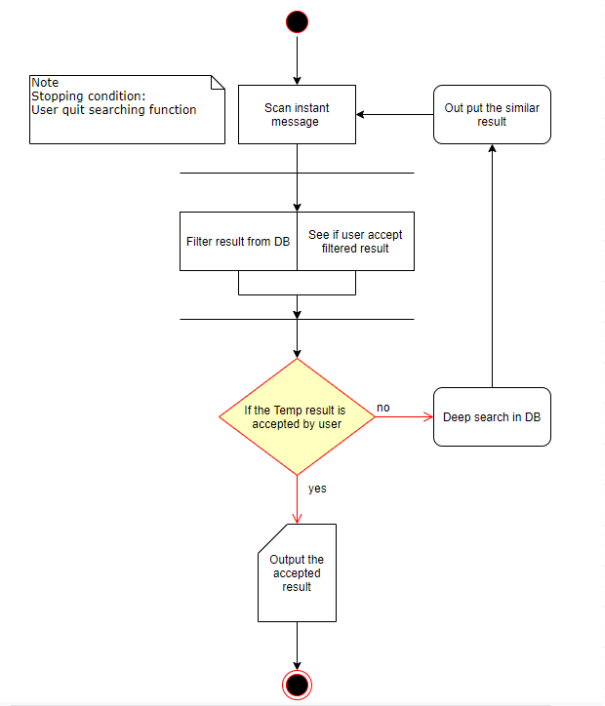
4.5 Creating Comment or Hashtag

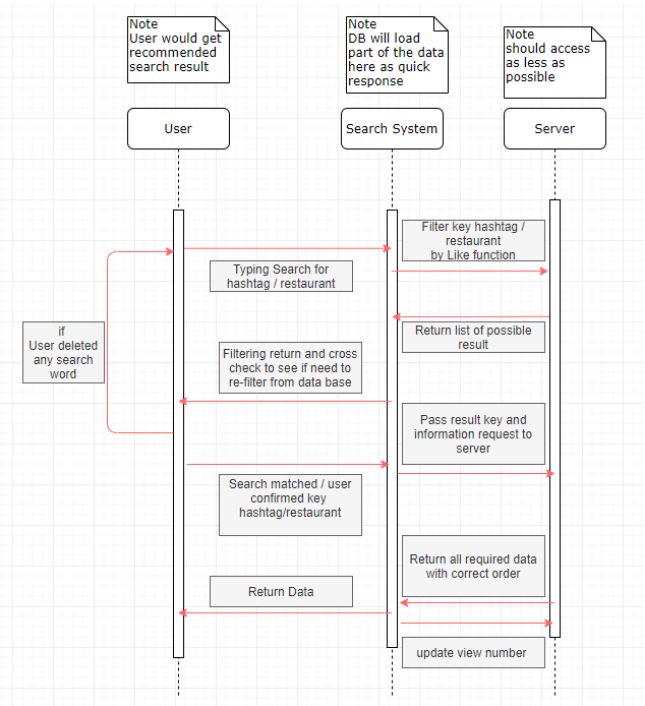




When a user views a post, we will generate recommended comment belongs to this post reference to the comment likes and views number. User may comment regarding the post or a specific comment. Whenever user type in #, hashtag mode would be activated. System will grab data from DB with user's typing. When user wrote a comment and clicked submit button. System will check if the user is the owner of that restaurant. If yes, we will add owner in front of the username of the comment and those hashtags will not be counted as tag increase under that comment.

4.6 Searching

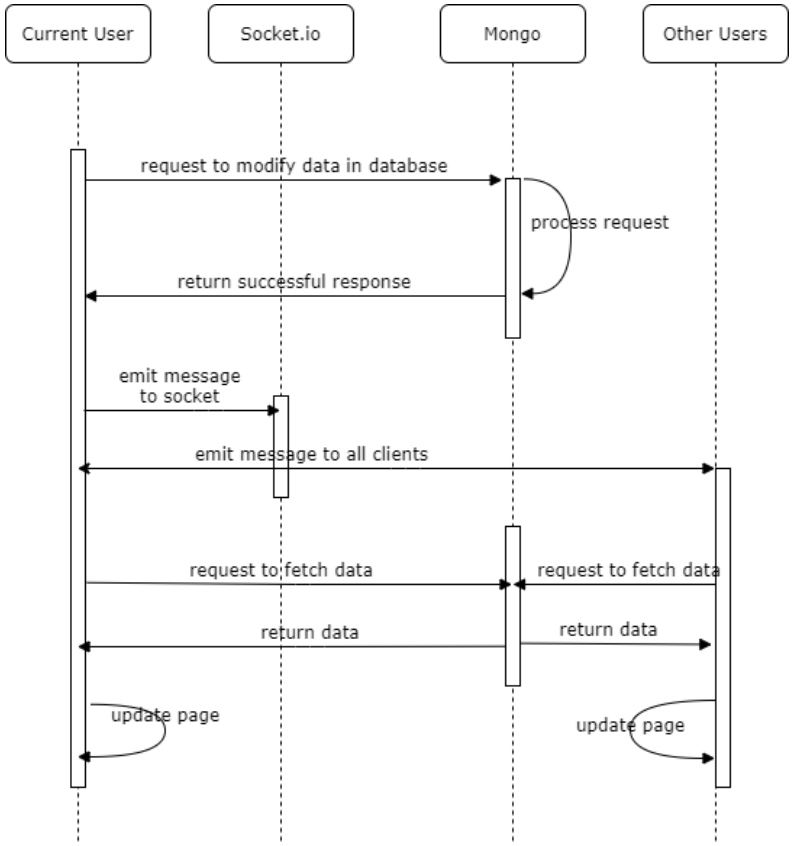




When user activate searching function. System will scan the current input and grab a list of similar result from DB. Those data would be stored locally and keep matching and recommending searching result to user. If user accepted search result, result will output the required data and function end.

In some cases, if users delete a word and that ruins the searching condition from DB. System will grab data from DB again and repeat the step above. The search is ended when search complete or user end the searching function. The matching between user and local will commit once user change searching input.

4.7 Socket.io



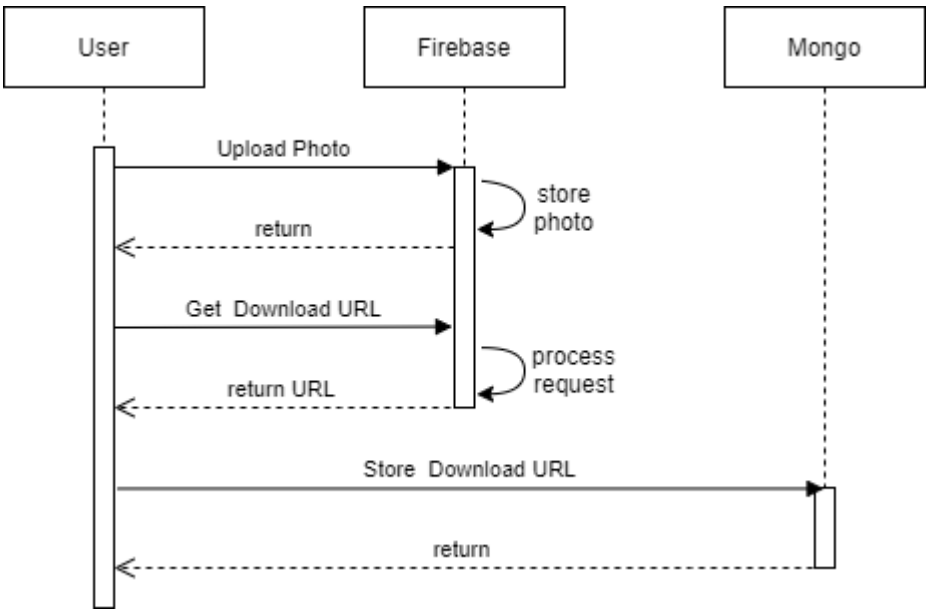
The above UML sequence diagram describing the interactions between users, mongo and socket.io when a user requests to modify data in a database.

In this project, socket.io is used to achieve synchronous communication between users. The main advantage is that users can see the changes done by other users immediately without refreshing the page, for example by adding comments under a post.

The above UML describes the communication process. Without loss of generality, the current user denotes the user that modifies data in a database, e.g., create/modify/delete action to entity/post/comment database. After the mongo backend finishes the request sent by the current user, if successful, a successful response will be sent to the current user. Then the current user will emit a message to the socket which will in turn emit another message to all clients, including the current user. Upon receiving the message, all users will fetch data from mongo and update the page accordingly.

However, the current method creates unnecessary loading to the server. One scenario is when two users that do not have following/followed relationship and hence cannot see each other's posts, the action done by one user should not affect the other user, but the user will still fetch data even when another user creates a post, which is unnecessary. To improve, instead of sending the socket.io emit message to all clients, the message should be sent to the relevant clients only.

4.8 Firebase



Sequence Diagram: Upload Photo

In this project, Firebase Storage is used for storing the photos uploaded by users in different posts.

The above sequence diagram describes the full sequence of how a photo will be uploaded. Upload multiple photos at the same time is also supported, the input for the upload photo request will become an array of photos instead of a single photo.

Firstly, the user uploads the photo to the Firebase Storage. After finishing the upload request (photo is successfully stored), Firebase Storage will return a response. Secondly, get download URL request will be submitted. The Firebase database will process the request and return the download URL of the photo. The download URL will then be stored in the Mongo database. After finishing the request (URL is successfully stored), Mongo database will return a successful response.

5. User Interface Design

5.1 Description of General UI/UX Design

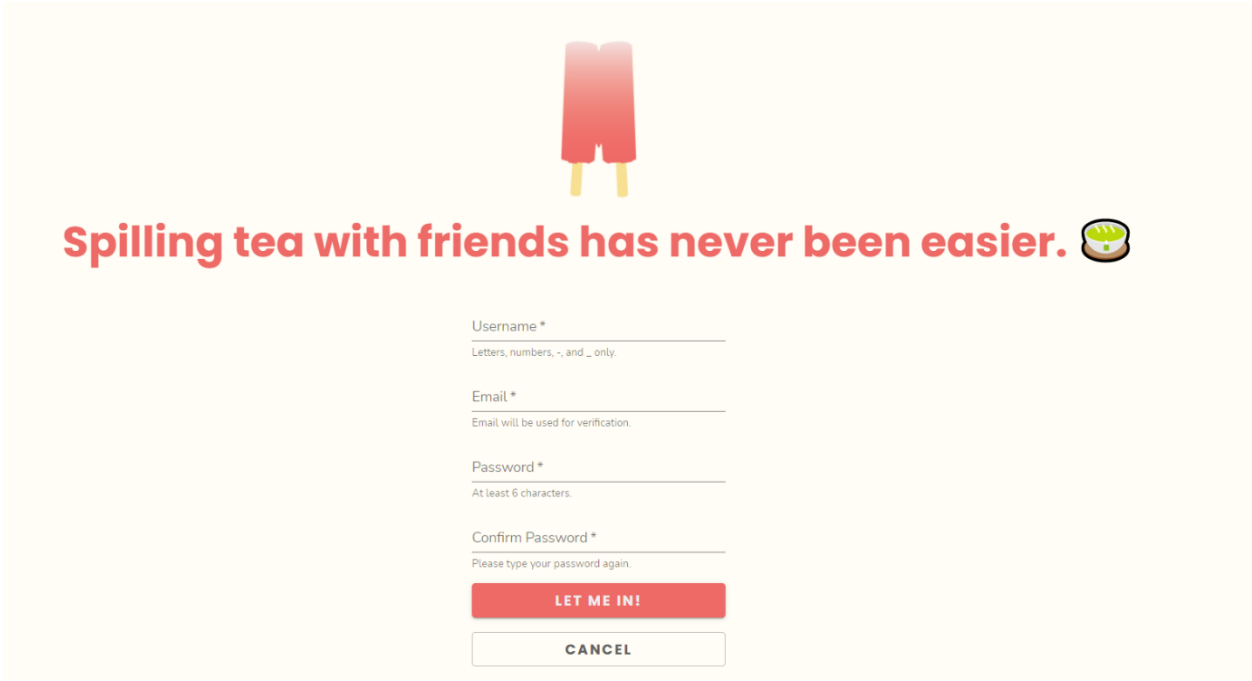
mATE put user experience as the utmost important feature. Therefore, we try to keep everything consistent and predicable throughout the whole application.

The logo is a Hong Kong food double popsicle (孖條) in a shape of the letter “M”, reminding users of food, warmth, and vigor.

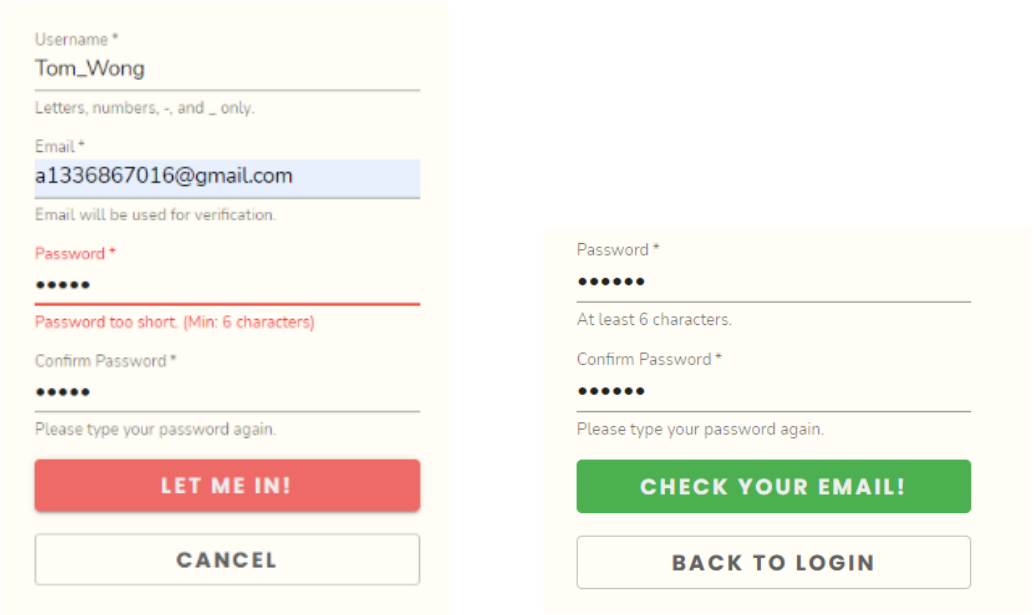
The UI and styles are mostly maintained by a base theme built in MaterialUI, on which we built and customize different components. The base theme uses the primary color #EE6A67, with dark (#B7393C) and light #FAF1F0 color variant. The background is a light beige #FFFDF5. The title font is **Poppins** and body font is **Nunito**, both sans-serif fonts and being responsive. In the future, if the base theme is modified, say, to change the main color from dark pink to dark blue, to implement a “dark mode” function, or to change font, most of the scheme will be changed accordingly. In fact, most of the UI/UX design follows the guideline of MaterialUI, for example, main buttons are configured to have solid background and secondary buttons outlined.

The whole application (after login) is consisted of a navigation bar and below a content box, which content is dependent on different webpages. While consistent to allow users pick up the logistics of the webpage quickly, the application made some notable difference whenever necessary. For example, in the profile page, restaurants have circle avatar and users square.

5.2 Registration Page

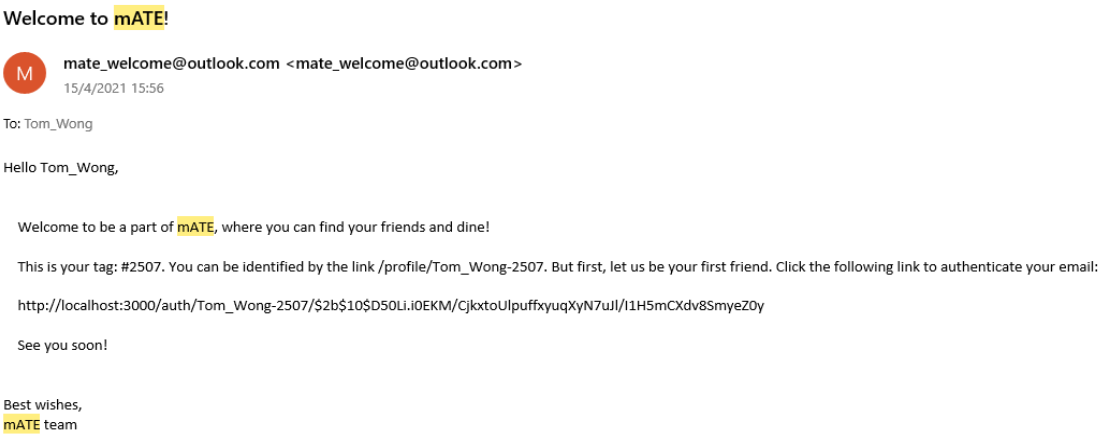


Users are required to input username, email, and password to sign up.



Username may only contain letters, numbers, “-“, and “_” only. Password should be made up of at least 6 characters, which must agree with the “Confirm Password” field. Any failure to meet such requirements are prompted with a helper error message under corresponding fields. Once a valid set of input is sent, a verification email will be sent to the user’s mailbox.

The verification email contains information on their server-generated 4-character tag, a partial link to their own account, and a full verification link. Registration is completed once the verification link is entered, which will then redirect the user to the login page.



Welcome to mATE!

M **mate_welcome@outlook.com <mate_welcome@outlook.com>**
15/4/2021 15:56

To: Tom_Wong

Hello Tom_Wong,

Welcome to be a part of mATE, where you can find your friends and dine!

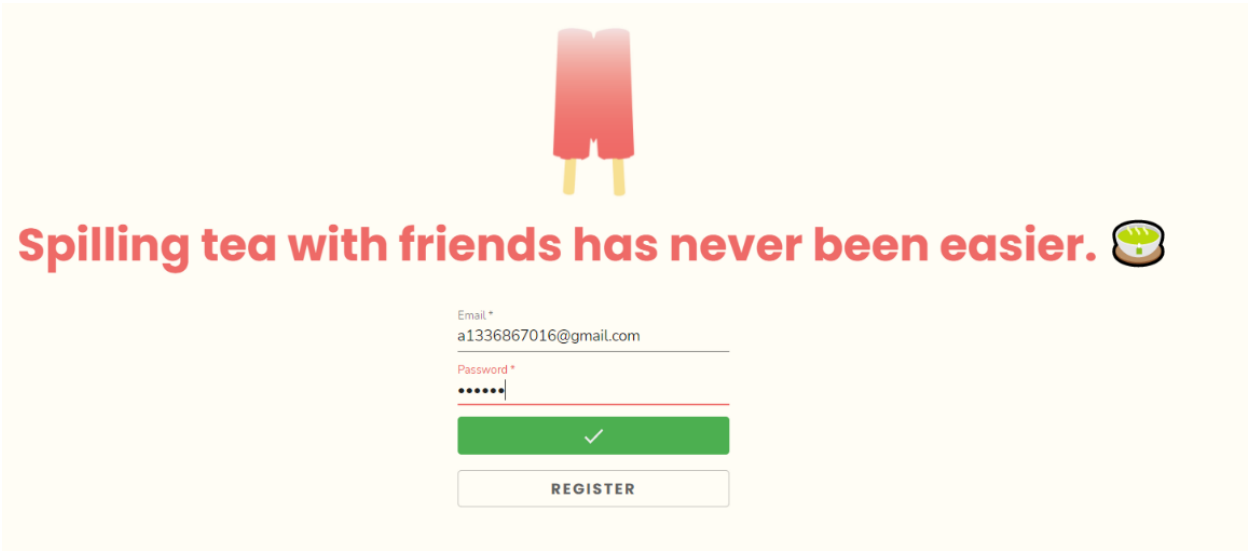
This is your tag: #2507. You can be identified by the link /profile/Tom_Wong-2507. But first, let us be your first friend. Click the following link to authenticate your email:

http://localhost:3000/auth/Tom_Wong-2507/$2b$10$D50Li.i0EKM/CjkxtoUlpuffxyuqXyN7uJl/I1H5mCXdv8SmyeZ0y

See you soon!

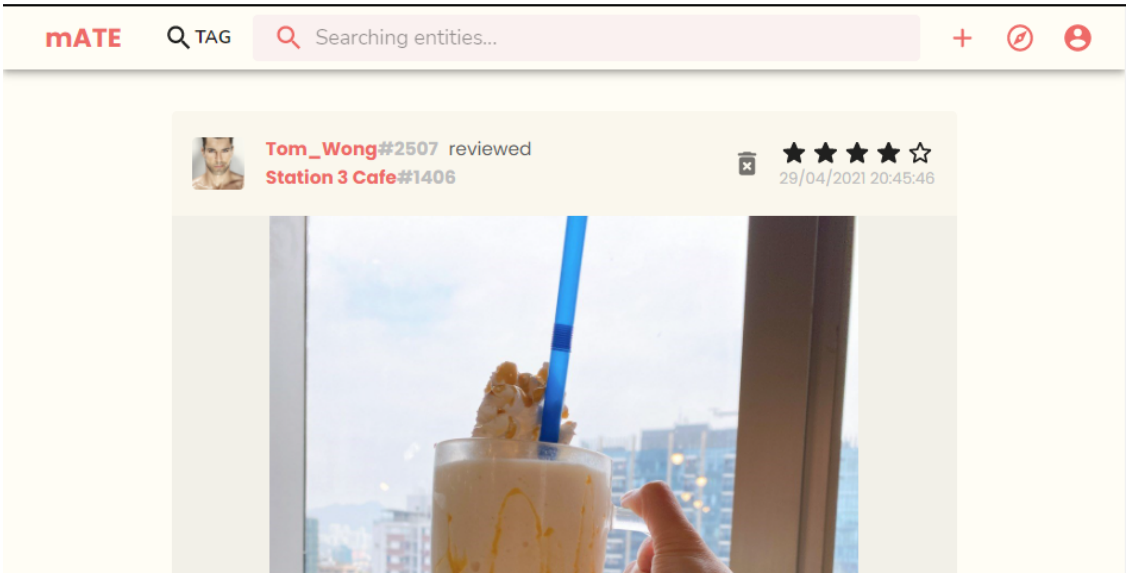
Best wishes,
mATE team

5.3 Login Page

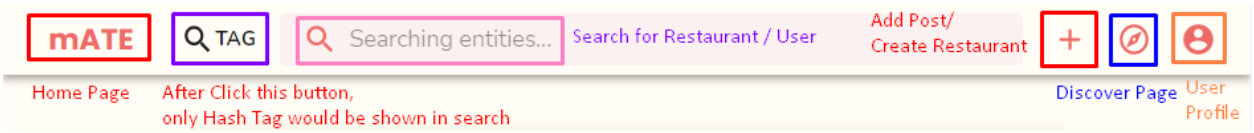


In the login page, users are asked to input a correct email and password pair to login. Any failure such as inputting a non-existing email account or a wrong password are prompted with a helper error message under corresponding fields. Once a valid email-password pair is sent, user will be logined and redirected to mATE's main page.

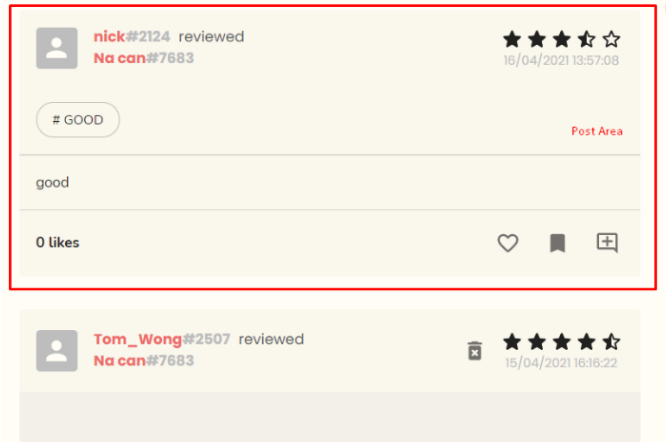
5.4 Main Page



The main page is composed of two major components: the navigation bar and the content area. This layout is consistent throughout the application to allow easy navigation between pages.

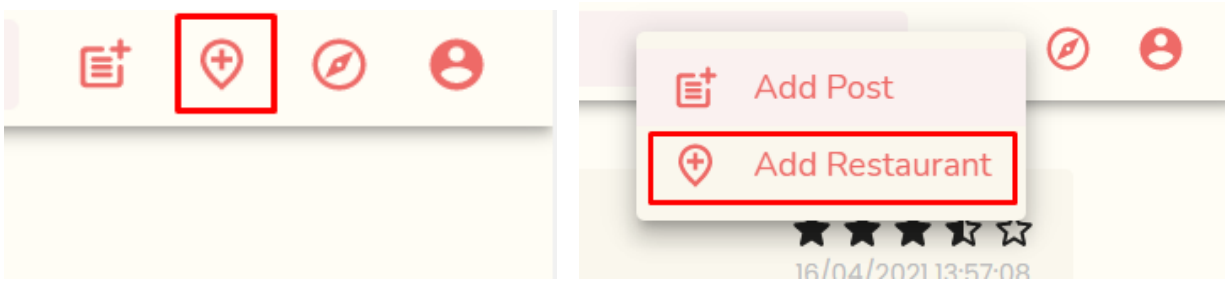


The navigation bar consists of three parts: a button with our application name mATE to redirect to home (main) page, a search bar to search for particular entities, and four action buttons on the right. The buttons lead to the pages for adding post, adding restaurant, discovering new restaurants, and user profile, in which the first two may be merged into one depending on the size of the viewport.

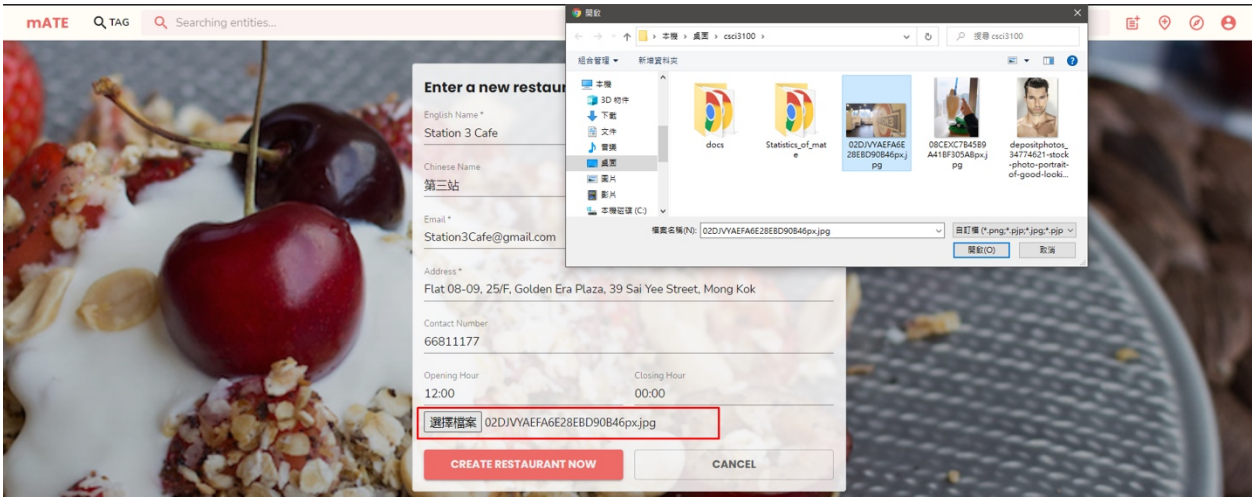


As for the content area of the main page, all recent posts created by the users or restaurants that the logined user has followed will be listed in chronological order, starting from the latest posts.

5.5 Create Restaurant Page



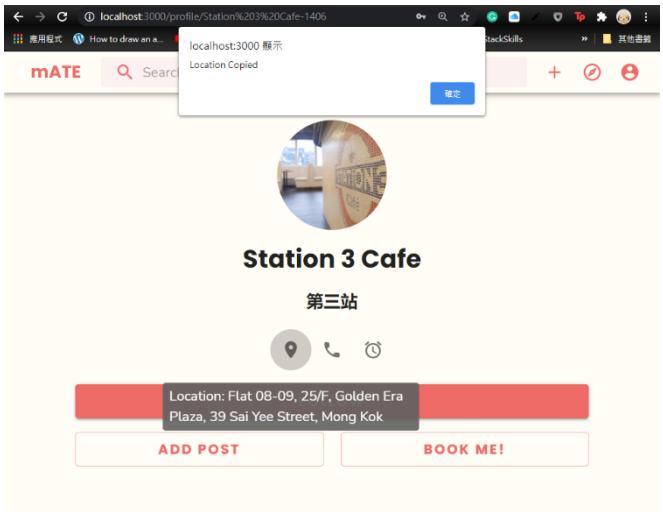
Depending on the viewport, the create restaurant page can be accessed by these buttons.



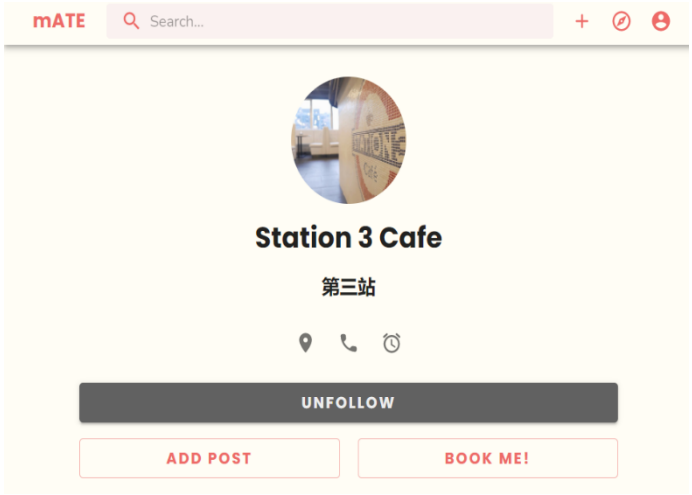
Fill in the required information to add a restaurant to the database. A profile photo can also be attached for the restaurant here. After a successful creation, the user is redirected to the restaurant profile page.

5.6 Restaurant Page

We know that every time you may want to share some specific information of the restaurant to your friend. We have implemented a feature for your convenience.



(Follow a restaurant)



(Copy restaurant information)

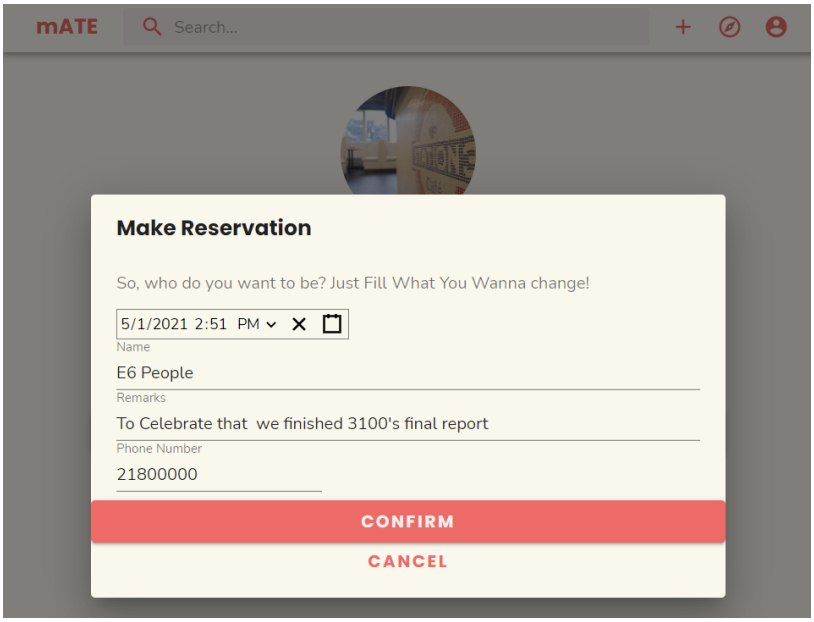
When your cursor just moves to the location Icon. mATE will show the location of restaurant. When you click that icon. Location will be copied, and you can share it with your friends. This feature also works for phone number and opening hour of the restaurant. You can also follow that restaurant in order to see the reviews/post of that restaurant at your main page.

For restaurant that already have user reviews, we would have Hash Tag Area to show the Top Six Hash Tag that the restaurant obtained from users.

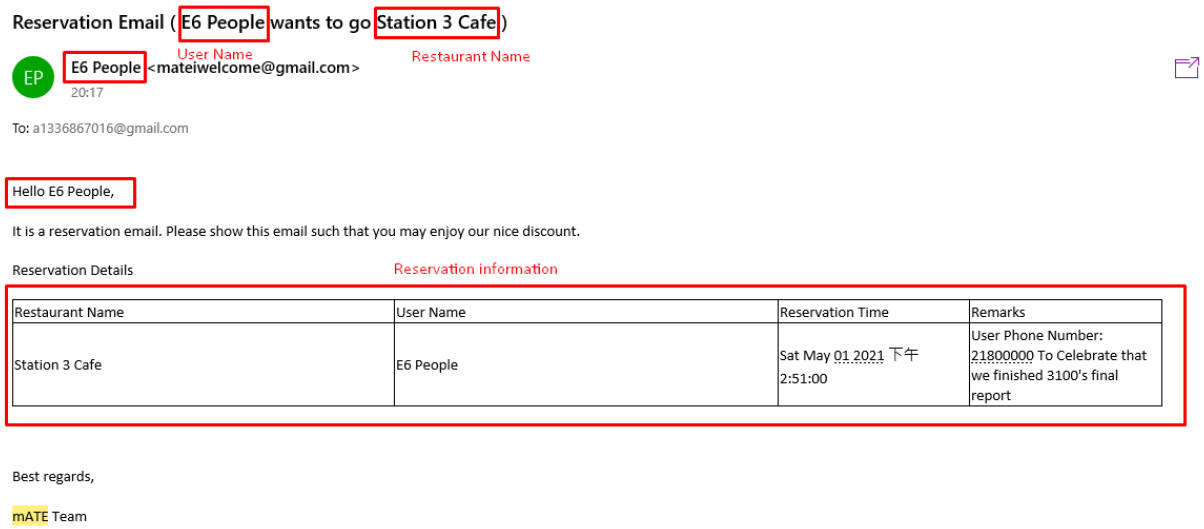
We also allow user to grade the restaurant for their service and food quality. We can see NA can obtained 4 stars that is 8 scores for the average of all post's score.

Therefore, Na can may be a good choice for your reference. Actually, I like their lunch set the most!

5.7 Make Reservation



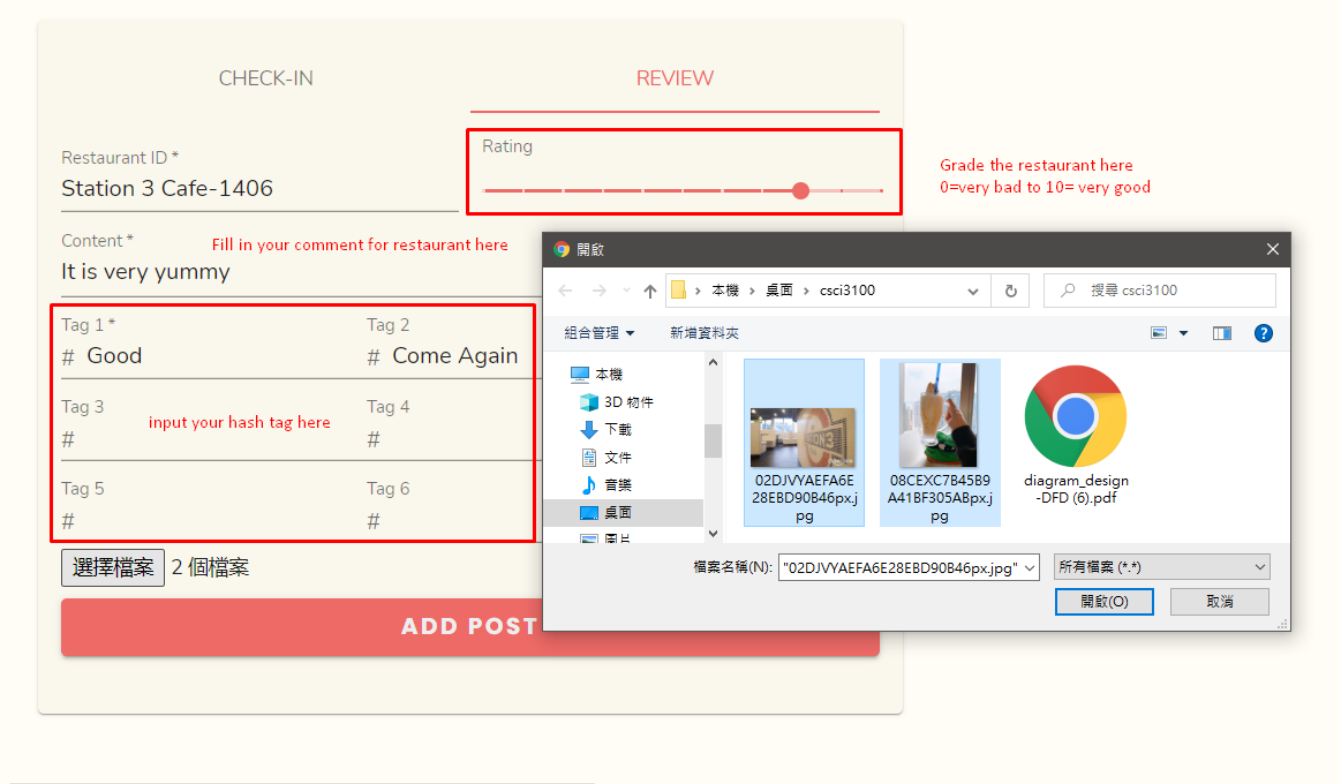
After reviewing other users' beautiful photo. You may want to have a taste on that restaurant, and we offer the "make reservation" feature by clicking the **BOOK ME!** button. Just fill in the basic information and the optional remarks. You will be able to confirm the reservation.

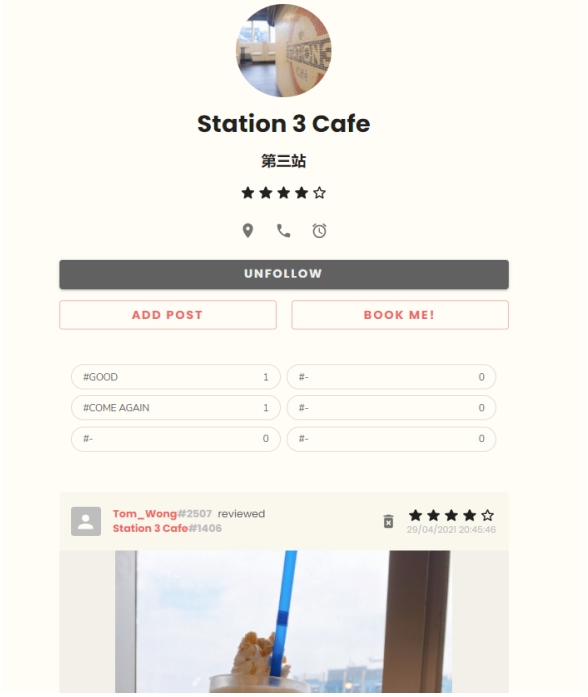


After user confirm reservation, both the restaurant and user will receive an email from mATE for reservation confirmation.

5.8 Add Post

After enjoyed the service from the restaurant. You may want to add your own post / review for that restaurant. You may add your post by clinking the **ADD Post**

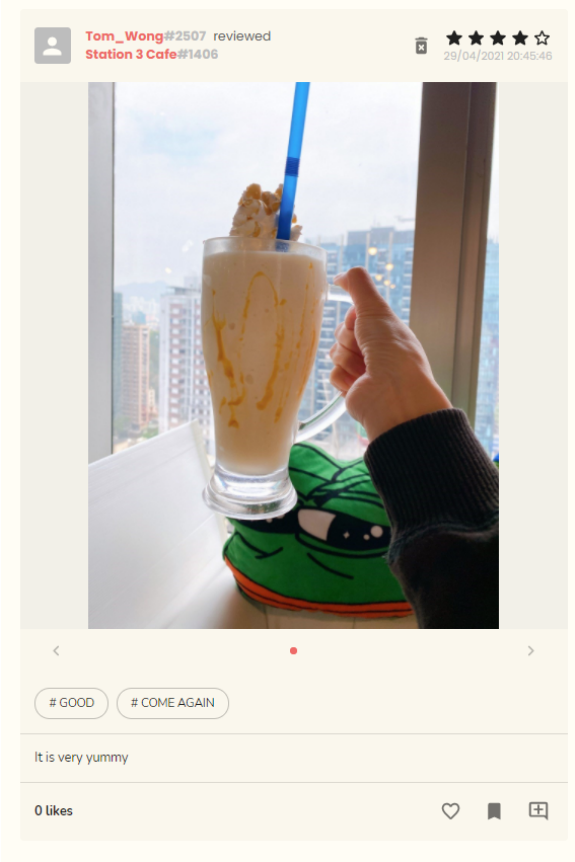


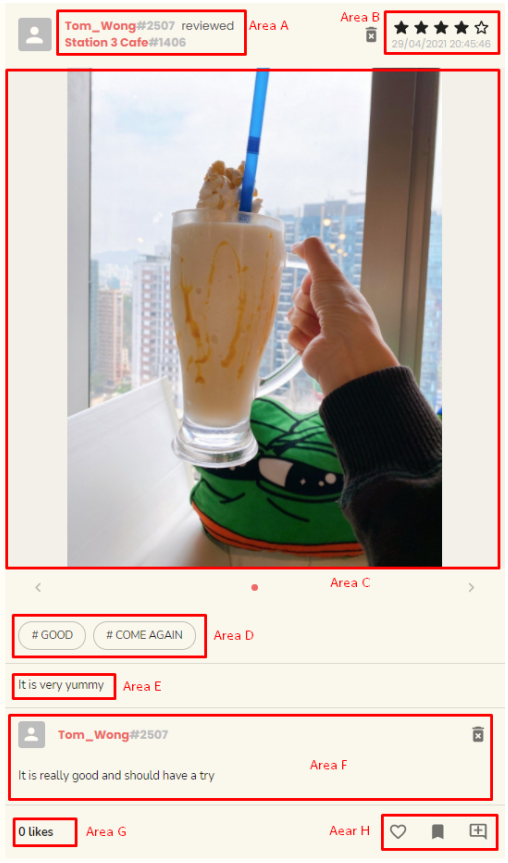


After adding post, the profile of restaurant will be updated including the

1. Top Six Hash Tag
2. Average Stars of the restaurant
3. Hottest post for this restaurant

5.9 Post Interface



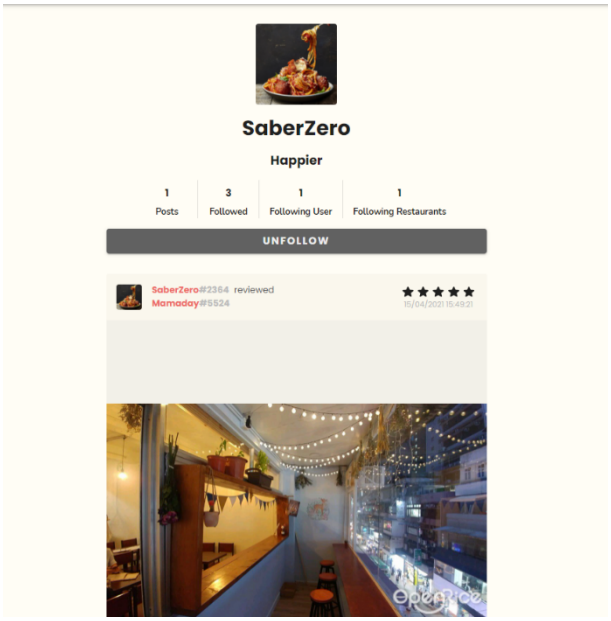


We separate the post system into 8 Area:

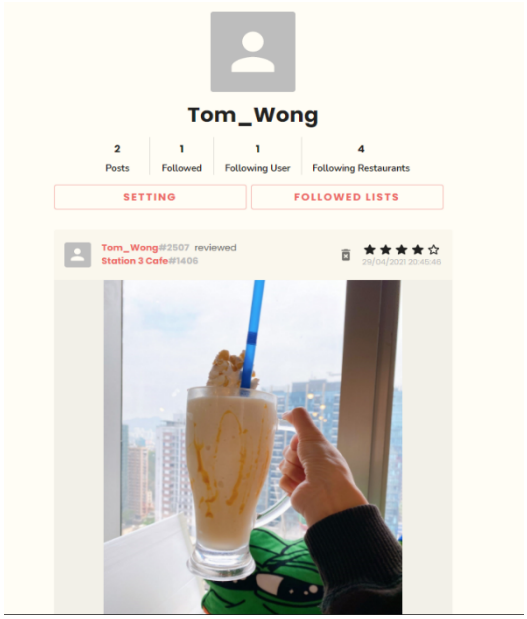
- Area A: Tom_Wong wrote a post for restaurant Station 3 Café. You may click the username or restaurant name to view the profile of Tom_Wong or Station 3 Café.
- Area B: Rating system from 1 star to 5 stars (1 - 10 marks).
- Area C: Photo uploaded for this post
- Area D: Hashtag function for user to express their feelings and allow us to collect data.
- Area E: Caption area for user to write anything.
- Area F: Show the latest 10 user's comment
- Area G: Total number of likes from other users
- Area H: You can **Like**, **Follow** this restaurant and **Comment** this post by clicking these 3 buttons respectively.

5.10 User Profile:

You may Search or click username from post to go any user's profile page or click the  to go my user profile



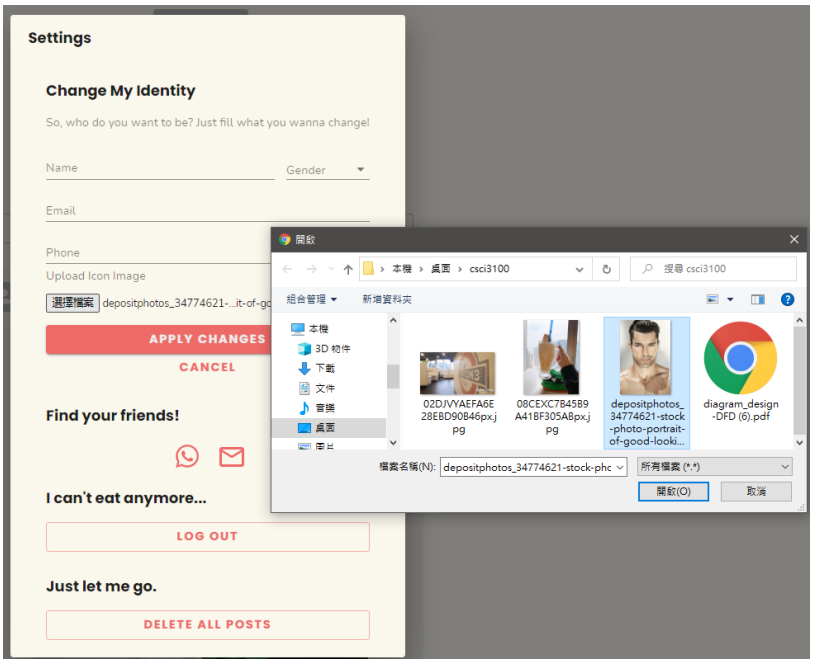
(Other user's profile)



(My user profile)

There are 4 information available in profile page: Posts wrote, Followed user, Following user and Following restaurant. There are two buttons for your own profile: Setting & Followed Lists.

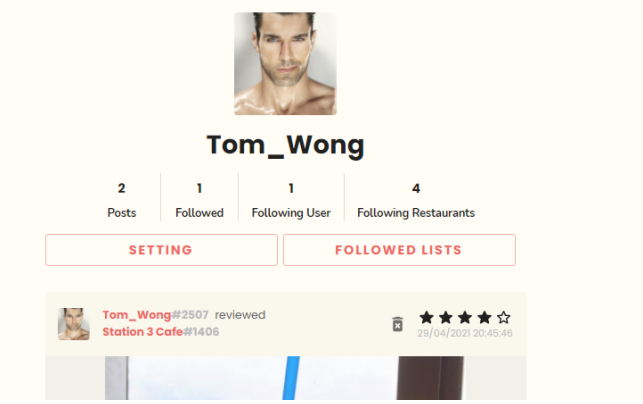
5.11 User Setting Interface

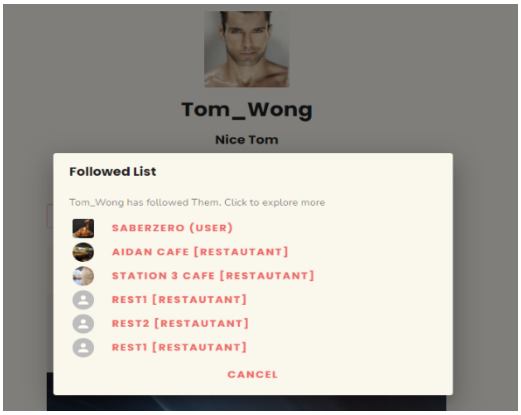


You are able to change the Name, Gender, Email, Phone and most importly ==> your icon in mATE.

You may also share your profile via WhatsApp and email by Find your friends 's buttons

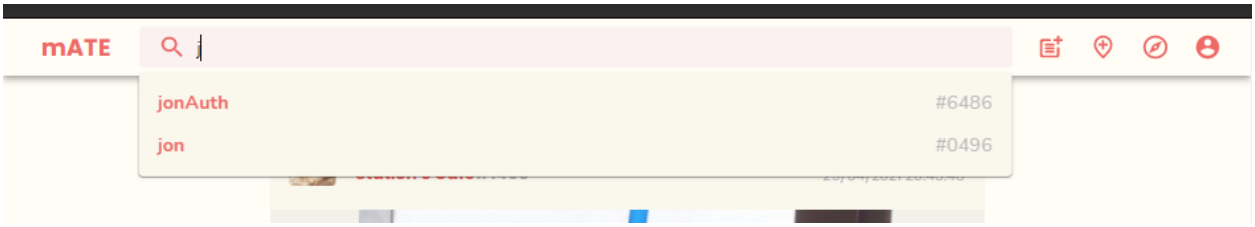
After Applied changes on icon





Tom_Wong become a handsome guy in mATE and all his icon in Post system will also be updated. You may also view the Followed List of user and restaurant in followed Lists button.

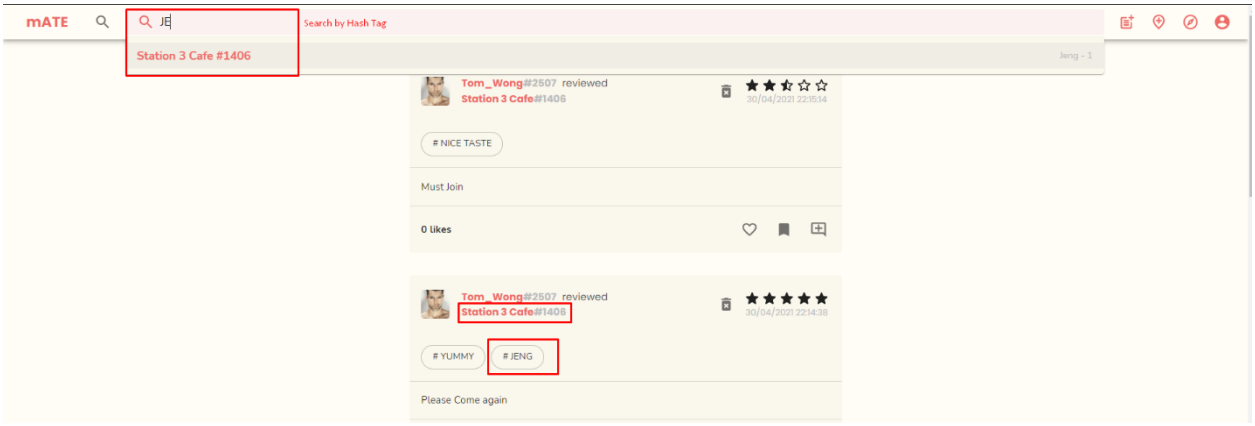
5.12 Search bar



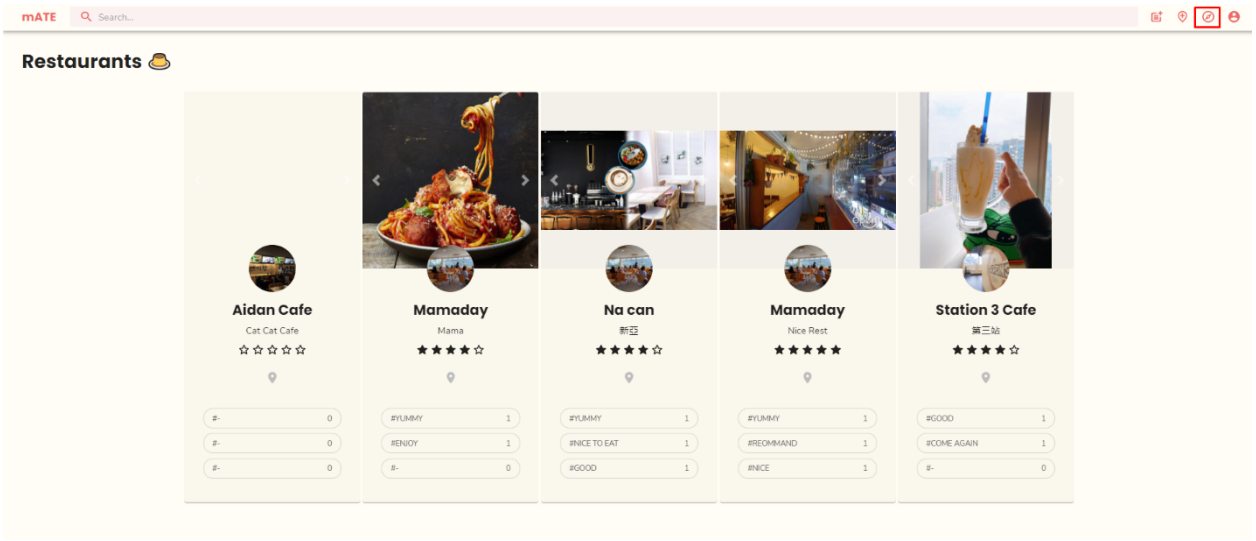
We offer search bar at the button of web page. You are able to search for user and restaurant there.

5.13 Search by Hash Tag

In our most updated version. We offer Hash Tag searching System. You may enter some HashTag and system will recommend restaurant for you with reference to the hash tag.



5.14 Discover Page



You may click the compass icon in bar to enter discover page. Inside this page, users are able to discover new restaurants and you can see to most used comment hashtag for the restaurant in this page.

6. TEST CASE

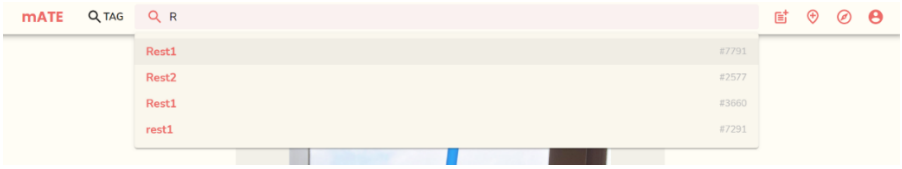
6.1 Login Test Case

User Email	User Password	Remarks of Input	System Response
a1336867016@gmail.com	hello1	Correct Email & Password & user verified	Login Success
a1336867016@gmail.com	P@ssw0rd	Wrong Password	**403** (Forbidden) "Incorrect password."
Abcdefg@gmail.com	Any	Email does not register	**404** (Not Found) "Entity not found."
Empty	Any	Missing Email	Rejected in frontend and users are notified
a1336867016@gmail.com	Empty	Missing Password	Rejected in frontend and users are notified
Abcdefg	Any	Email should contain @	Rejected in frontend and users are notified
ohMyLove@gmail.com	computer	Email not verified by user	**403** (Forbidden) "Not verified."

6.2 Register Test Case

Username	Email	Password	Confirm Password	Remarks of Input	System Response
Anyone Is empty				All fields are required	Frontend: Please fill in the field
IronMan	Tony@marvelDC.com	IamIronMan	IamIronMan	Normal	Register Success
IronMan	a1336867016@gmail.com	IamIronMan	IamIronMan	Email already exists	Request failed with status code **409**
IronMan	Tony@marvelDC.com	IamIronMan	IamInevitable	Password & confirm Password does not match	Frontend Reject And notify user
IronMan	Tony@marvelDC.com	Iam	Iam	Password too short	Frontend Reject And notify user
Tom_Wong	Tony@marvelDC.com	IamIronMan	IamIronMan	Username Already exist	Register Success (expected)

6.3 search function



Input parameter: input string in search bar

Expected display result: username of entity whose username/entityID matches the input string **starting from the first character**

Example of input parameter = 'R'

Below the search bar is the display result

Existing entity in database

	username	entityID
Entity1	Rest1	Rest1-7791
Entity2	Rest1	Rest1-3360
Entity3	rest1	rest1-7291
Entity4	Rest2	Rest2-2577
Entity5	SaberZero	SaberZero-2364

Results

Input parameter	Display result (entityID)	Remarks
R OR r	Rest1-7791 Rest1-3360 Rest2-2577 rest1-7291	Not case-sensitive
Rest1	Rest1-7791 Rest1-3360 rest1-7291	
Rest1-7	Rest1-7791 rest1-7291	Matches entityID, but not username
Rest 1 (Space between t and 1)	No options	Space is also counted as one character
Rest1 (A space before R)	No options	Space is also counted as one character

6.4 Search Hashtag

Existing hashtags in database

hashtag	Frequency	Restaurant ID
taga	3	Rest1-7791
tagA	1	
delicious	1	
de_li_cious remark: (two space)	1	
tagA	1	Rest2-2577

Results

	Display results		Remark
Input	Username #tag	hashtag - frequency	
taga	Rest1 #7991 Rest1 #7991 Rest1 #7991 Rest2 #2577	taga -3 tagA -1 tagA -1 tagA -1	Not capital letter sensitive
de	Rest1 #7991 Rest1 #7991	delicious -1 de li cious-1	Match starting from the first character
de_l (Space between e and l)	Rest1 #7991	de li cious-1	Space Is counted as one character

6.5 Create post: check-in

Results

Input parameter	Display result	Remark
Any restaurant ID (empty value, existing/non-existing entityID) Content: empty	Please fill in this field (content)	Both restaurant ID and content are required
RestaurantID: empty Content: non-empty	Please fill in this field (resturantID)	Both restaurant ID and content are required
RestaurantID: non-existing restaurant Content: empty or non-empty	Invalid restaurant ID.	Cannot create post for non-existing restaurant
RestaurantID: existing restaurant Content: non-empty	Created!	

6.6 Create post: review

Input parameter				Display result	Remark
RestID	Content	Tag1	Photo		
Empty	Any	Any	Any	Please fill in this field	
Existing	**Empty**	Any	Any	Please fill in this field	
Existing	Non-empty	**Empty**	Any	Please fill in this field	
Non-existing	Any	Any	Any	Invalid restaurantID!	
Existing	Non-empty	Non-empty	**>5MB**	Invalid file type	Multiple photos can be uploaded, but each of them should be <5MB
Existing	Non-empty	Non-empty	**File type: non image**	Invalid file type	If images and files are uploaded at the same time, none of them will be successfully uploaded
Existing	Non-empty	Non-empty	< 5MB & File type: image	Post created	

6.7 Comment

Create comment

Input Comment field	Display result
empty	No comment created
Non-empty	Comment created

Delete comment

Target comment	Display result
Current user's comment	Can delete
Other user's comment	Cannot delete

6.8 Follow

Action	Display result
Follow an unfollowed entity	Unfollow-->follow
Unfollow a following entity	Follow-->unfollow

All of the above Test result are with In our expectation

7. Lesson Learned

7.1 Methodology

Throughout the project, we were using Extreme programming. We first documented the structure of data and the data flow & UI design of webpage. However, we found that what we coded and what we designed has a large difference due to the lack of experience. So, with a well-defined project goal in mind, we simply dive right in and start coding with reference to what we have learnt. We soon got familiar with the technologies and worked on the basic functions; these functions created the prototype of the application and, in several modifications, more enhanced functions and products are made, which we improve continuously.

7.2 Technologies

The application is composed mainly of the MERN stack, MongoDB (Mongoose), Express.JS, React, and Node.js. To facilitate cooperation between members, we have created a Github repository and a virtual machine using GCloud.

7.2.1 Backend – Node.js, Express.JS, ...

For the **backend**, we learned how to use Node.js and Express.JS to create suitable RESTful APIs for the frontend. Multiple functions, mainly concerning of the basic CRUD (Creation, Read, Update, and Delete) functions, are first set up to handle the logistics between the data and databases. These functions can be used individually as normal JavaScript functions to manipulate data in the database before the frontend is built.

In the process, we discovered that some classes rely heavily on other classes. For example, the Post component (or class) stores the reference from the Entity, Hashtag, Comment components. Even some of the basic searching functions require a User to be fetched (specifically their ObjectId in Mongo databases) before the desired result can be found. Considering this, we reviewed the functions, checked which functions are initiated by the users or are related to other class, and decided instead of manipulating all databases in different modules, we specify each module would mainly concern of only one database, calling other functions in other modules or passing properties (prop) if necessary. This in turns improves the maintainability of the codes and produces a loose coupling but high cohesion modules.

After the functions are well-established, according to their CRUD functions, we set up respective GET (Read), POST (Read/Create), PUT (Update), DELETE (Delete), and PATCH (Update) APIs and routes for the frontend. Again, these routes are separated into different routers to make sure the interconnectivity between modules are well-established.

7.2.2 Database – MongoDB, Mongoose, JWT, OAuth, Firebase, …

For the **databases**, we learned how to use MongoDB and Mongoose (JS package) to set up data. As for photo uploading, we used Firebase for photo storage. There are mainly two problems we have encountered.

The first is the extended classes of Entity - User and Rest. In the beginning, we used a local field "type" to separate the two. We still have a User and Rest databases, in which each document stores the ObjectId reference to the Entity. This worked out fine but each time we have to fetch an Entity, we needed to access two databases at the same time; and since User/Rest and Entity are two objects and only User/Rest documents store a reference to their corresponding Entities, a second query is often needed to get a full document. After some lengthy investigations, we discovered MongoDB has a discriminator function that more or less does what we desired.

The second problem is the separation of the User and Auth Mongo databases, or more gene rally, the separation of the resource server(s) and the authentication server. In the beginning, we masked the password field, although hashed, when frontend sent a fetch request on a User. This had proved to be successful until we decided to include much stronger authentication protocol for the application. We decided to use JSON Web Token and implement an OAuth protocol with refresh and access tokens. This took a long time for us to learn and configure the logistics behind. Nonetheless, we landed on separating the databases and this proved to be much more maintainable (and user-friendly for the login component of user) than simply masking a highly confidential field.

7.2.3 Frontend – React, EmailJS, History (npm), MaterialUI, Socket.io, …

For the **frontend**, we learned how to use React to create a basic responsive webpage. In the beginning we learned how to build React class components, only to find out that React has updated and recommending using functional components instead. We searched and discussed about the differences between the two components and learnt many new ways to improve our components. Hooks are one of the main features we have learned and tried to use through useState(), useEffect(), and also custom made hooks such as useWindowDimensions(). A transition from single page application to multi page applications using React.Router and the History package is also new to us.

Another technology we used is EmaiUS to send out reservation emails and verification emails, which the latter is later moved to backend to increase security (the verification link or the generation of it should not be private and not shown to users).

As for photo uploading, we used Firebase for photo storage. We encountered a problem related to the asynchronous communication between the client and the firebase storage. When we upload multiple photos at the same time, we encountered a problem in passing the download URL to the mongo database. The result always has a null value. After different trials and errors, we discovered that the problem is caused by asynchronous communication. Therefore, we

solved the problem by returning a promise (array of download URLs) after receiving all the promises (download URL of each photo) from firebase.

As for UI design, we used the common Bootstrap (React Bootstrap) which is later changed to MaterialUI. As none of us have experience in using this before, a lot of configurations as general as a global theme to as specific as the MaterialUI. AutoComplete API are ones we have spent many hours investigating into, all while learning their connections to the React interfaces.

To provide synchronous communication between users and within the application, aside from using React built-in states, we also used socket.io to achieve a real-time application.

7.3 Code Legibility and Documentation

Since we are working as a team and every module is interlinked with one another in some way, code legibility and documentation is one of the most important aspect to a smooth and successful project. In backend, we learned that it is essential to document every route developed to allow frontend to call the APIs, instead of relying on teammates to search through lines and lines of code. Throughout the development process, it also became apparent that it is crucial to name all variables with meaning and develop (and negotiate with other teammates) a consistent coding behavior for our own and for others' sake.

The last thing we discovered (through pain) is that documentation should be built up gradually instead of being the last stage of the development process. No matter if it is a single-line comment or a complete JSDoc, it helps significantly to other teammates or future developers to debug and maintain the codes. We learned how to use JSDoc to document modules, set up functions and classes, and refer to other modules, etc; using external packages to formulate such documents is much better and much more eligible than using readme.md to communicate between members and back-front ends.

7.4 Looking Back... – A Review

We believe this project put things into perspective and we are always trying to be self-critical. Looking back to the whole development process, there are something that can be improved.

First and foremost, our team should communicate better and be decisive on the packages and coding format we are going to use. Although it is a great learning opportunity to learn from the mistakes, some of the aforementioned processes, such as transitioning from backend storage to Firebase and separating User and Auth databases to accommodate OAuth authentication process, can be avoided had we decided on what extent the project is going to achieve. Different people have different expectations and habits - some have used SQL databases but not NoSQL databases, some prefer Bootstrap over MaterialUI - but as a team we should have put more time and thought into the planning process so that we can work truly as a team.

Secondly, our team should work on the basic, create maintainable and reusable codes, and only then start to expand our project. We used to have a huge plan for this project at the very

beginning – actually, when one views the setup of the project's backend and database, one can see a lot of classes such as GroupList (lists of lists of user's saved entities) and Resv (reservation) are written but never used. The time spent on developing these functions can be first spent on creating a basic application, onto which we will base and produce a larger and more comprehensive application. Also, functions with similar purposes should be grouped into a multi-usage function to ensure reusability of codes and components. One of such examples is the Rating function; a counter-example would be the Hashtag function scattered in three different files.

7.5 Looking Forward – Future development

We believe mATE is a project with great potential. There is nothing similar yet in the market and the trend of using social media platform as a restaurant-searching application is growing rapidly.

First off, we would like to implement the things we had set our minds into in the initial design and already have support in the backend. GroupLists, a list of restaurant lists where user can save their favourite restaurants, is crucial in inter-user experience. They can browse and also share their favourite lists to friends that not only can keep tap on restaurants they really like or have not been to, but also as an incentive to use this app. Resv, the class which stores reservation details, is also essential to the communication between Users and Rests. User who is an owner of a restaurant can claim their place and then review the reservations in-app, instead of using emails. Discover page should also display random posts to allow users discover more entities out of their usual scope. Some small functions like deleting accounts and updating comments should be included as well.

Secondly, we put a lot of thoughts into the hashtag functions. This is the distinctive feature of our application. After the demo, we have implemented and displayed some of the potential of the tags, including searching entities by tags. In the future, we would like to create auto statistics to evaluate the "positiveness" of a tag, such as "good" being positive, "disgusting" being negative, and "ice-cream" being neutral. This metric will also partially influence the rating of the restaurant and in doing so, the rating will more accurately present the restaurants.

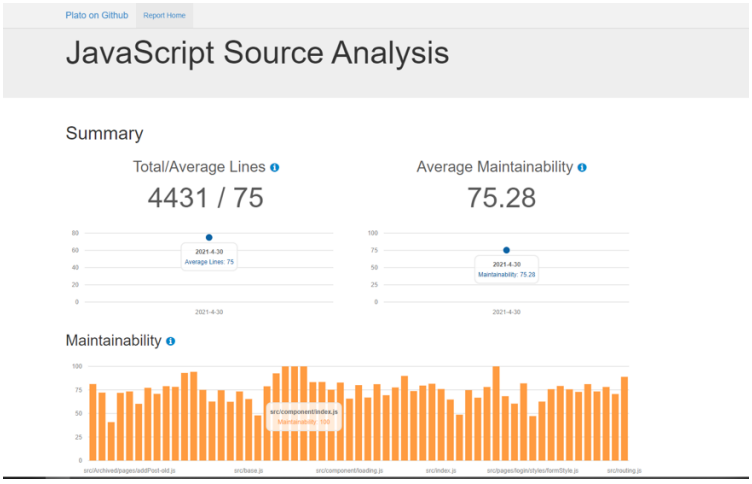
Thirdly, albeit a bit far-fetched, we would like to cooperate with restaurants to provide coupons and discounts to our users. The goal of our application is to create a user-friendly and user-guided environment for both users and restaurants. In allowing restaurant to promote themselves without having to compromise the integrity of a user-rating-based application (since it is mostly social-media based than restaurant-searching based), we believe mATE can achieve what the two types of application – social media and restaurant searching applications – can, but more.

8. Conclusion

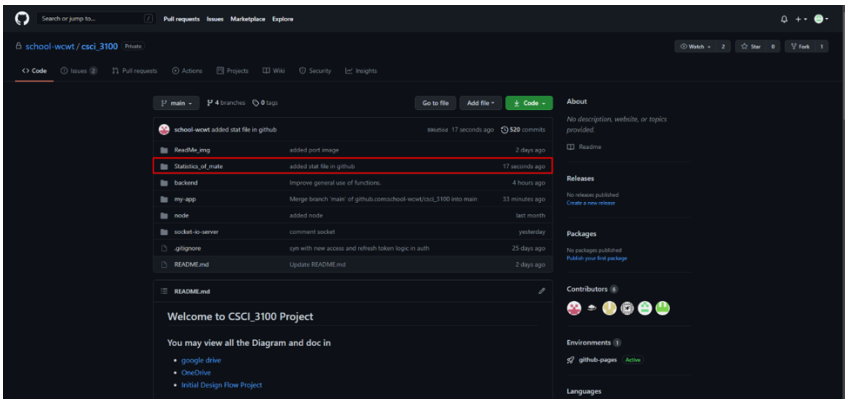
In mATE this project, we aim and successfully provide an environment to search and communicate between users who like to eat and share their experience. The responsive web-based application is set up mainly with MERN stack, decorated by MaterialUI, with many assisting packages such as Firebase. Within this app, users will be able to make posts, discover new restaurants, share entities to their friends, and all-in-all create an excellent community for users and restaurants likewise. The current mATE is also developed to withstand vigorous future developments and tests. We firmly believe mATE will be the perfect place to connect between users and between user and restaurants.

9. APPENDIX

9.1 Statistics



We have implemented a html page using Plato which indicates the maintainability / expected Warning. Avaliable in GitHub page.



9.2 Reference

1. https://getbootstrap.com/
2. https://www.w3schools.com/
3. https://material-ui.com/
4. https://reactjs.org/
5. https://firebase.google.com/
6. https://socket.io/
7. https://www.mongodb.com/
8. https://nodejs.org/en/
9. https://expressjs.com/