Developing a Web Application For Cocktail Community Service: "Cocktell"

Eunbi Kang[1], Taehoon Kim[2], Byeoungkyu Park[3], Youji Cho[4], and Eunseo Choe[5]

[1] Sungkyunkwan University, Department of Statistics
[2] Sungkyunkwan University, Department of Mathematics
[3] Sungkyunkwan University, Department of Sports Science
[4] Sungkyunkwan University, Department of Physics
[5] Sungkyunkwan University, Department of Global Leader
https://github.com/silverwest8/MixBowl

Abstract. This paper presents Cocktell, a web application designed for cocktail beginners and enthusiasts. Unlike existing platforms, Cocktell integrates cocktail recipes, a cocktail bar map, and a reliable community in one platform. Cocktell offers various pages, including registration, login, home, recipe, community, cocktail bar map, and my-page. Users can browse recipes, post their own recipes, filter and sort recipes, and report inaccurate information. The community section allows users to view and contribute posts, categorized by recommendations, reviews, questions, and a free bulletin board. The cocktail bar map displays nearby bars and reviews, and users can also search for bars by location. The my-page section allows users to modify their profile and verify their bartender certificate. Cocktell provides a reliable and user-friendly platform for cocktail enthusiasts to explore recipes, engage in a community, and locate cocktail bars.

Keywords: Cocktail · Recipe · Community · Cocktail bar map · Bartender

1 Introduction

"Cocktell" is a web application for cocktail beginners and enthusiasts. As people's interest in Cocktail has recently been increased, there were some cocktail recipe apps and cocktail community website, but there is no app or website provided both recipe information and community. Moreover, the information the existing community site provided isn't reliable and worthy enough. The main goal of "Cocktell" is to provide the cocktail recipes, the cocktail bar map, and the easily accessible cocktail community for beginners and enthusiasts with reliability. Additionally, to provide reliable and high-quality information in the community, we decided to attract bartenders to our service by dividing membership levels and giving a high level to user who verified bartender certificate.

To develop the web application, we used the React, NodeJS, Express, and MySQL. The React library is used to develop the web pages. The NodeJS is the JavaScript runtime required to develop the server. The Express is the framework for developing the server. The MySQL is the relational database management system. We used it to define and manage precise data schemas, and use efficient query capabilities.

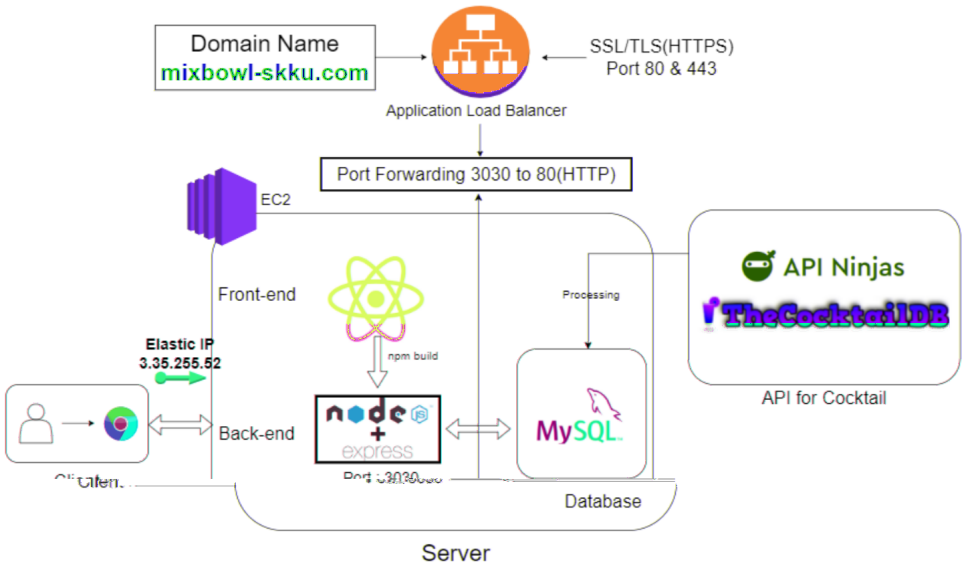
"Cocktell" has seven types of pages: Registration page, Login page, Home page, Recipe page, Community page, Cocktail bar map page, My-page. We implemented the registration and login function. Also, we used the Json Web Token for authorization. In the home page, the user can see this week's popular cocktails, questions needed answers to, cocktail bars near the user, and this week's popular community posts. In the cocktail recipe page, the various recipes are displayed. We obtained cocktail recipe

data using the external APIs and modified it to fit our recipe table attributes. The users can filter and sort the recipes with various condition. They can post/edit/delete their own cocktail recipe with images. Also, they can report once per recipe. In the community page, the users can see the posts categorized by the cocktail recommendation, question, cocktail review, and free board. Also, they can search the posts by keyword. They can post/edit/delete the posts including images, and post/edit/delete the comments. Also, they can report once per post. Those functions in the community page is possible by implementing the CRUD APIs. In the cocktail bar map page, the user can see the cocktail bars and their reviews within 1 kilometer centered at their current location. To get the user's current location, we used the Geolocation API. To display the map, we used the Kakao Map Javascript SDK[1]. To get the cocktail bar information, we used the Kakao Local Search API.[2] Also, we used the Google places API and Google geocoding API to implement the function to search cocktail bars by location. The users can post/edit/delete the reviews with rating, review text, keywords, and images. In the my-page, the users can modify their nickname, verify the bartender certificate, and see their activity histories. We implemented the API to verify the bartender certificate by using the site called Q-net.[5]

The usability test was performed by asking one member of each team. We received the feedback the overall UI was luxurious and had a young sensibility to match the concept of cocktail. It is also said that both the desktop and mobile versions work well. There were also opinions that it was good that it was divided into several bulletin boards, such as recommendation/review/question/free bulletin board in the community.

2 Design for the Proposed System (Service)

Fig. 1. Our Service's System Architecture



2.1 Overall Architecture

We deployed our application in a cloud-based web environment using the widely known AWS (Amazon Web Services) platform. We utilized EC2 (Elastic Compute

Cloud) as our server infrastructure. Within the EC2 server, we set up the environment for the frontend, backend, and database components. We built the frontend using the React library, and after the build process, we delivered the frontend files as static files to an Express server powered by Node.js, thereby connecting the frontend and backend.

We assigned port number 3030 for the backend server within the EC2 instance. To allow external access to the application over the HTTP protocol on port 80, we set up port forwarding within the EC2 instance. However, to ensure security, we needed to apply HTTPS. To achieve this, we utilized ACM (AWS Certificate Manager) provided by AWS to obtain an SSL/TLS certificate for our domain. We easily applied the certificate using an Application Load Balancer, which automatically redirects traffic from port 80 to port 443, enabling users to access the application securely via HTTPS.

In terms of DNS, we obtained our domain name "mixbowl-skku.com" through AWS's service called Route 53. We connected the domain name to the Application Load Balancer, allowing us to apply our domain to the application.

Overall, this deployment architecture leveraged AWS services such as EC2, ACM, Route 53, and Application Load Balancer to provide a secure and scalable cloud-based web environment.

2.2 Framework (Programming Language) and Database

Except for HTML and CSS, both the frontend and backend code are written in JavaScript. We chose JavaScript as the primary language for both parts to integrate development processes, enabling faster development. For the frontend, we utilized React, a JavaScript library, while for the backend, we utilized Node.js along with the Express server framework. This unified JavaScript approach allowed us to share code and maintain consistency across the application.

To ensure structured data storage and retrieval, we employed MySQL, a widely used relational database management system (RDBMS). By using MySQL, we were able to define and manage precise data schemas, ensuring data integrity and efficient querying capabilities.

Overall, our development stack consisted of HTML, CSS, JavaScript (React for frontend, Node.js with Express for backend), and MySQL for the database, enabling a fast and integrated development experience.

2.3 Challenges

Initially, when we started using the EC2 server, we took advantage of the free tier provided by AWS, which offered a Linux operating system with 1GB of RAM and an 8GB volume size. However, we encountered issues where the server would go down during the process of fetching the source code and downloading related development packages. To address this, we utilized Linux's swap memory. Swap memory is a mechanism that uses disk space to supplement physical RAM when it becomes fully utilized. By utilizing swap memory, we were able to overcome the memory limitations and stabilize the server.

However, we encountered another problem when building the frontend code. The server went down again. When we checked the volume using the 'df -h' command, we found that there was only around 760MB of available space. Considering the operational requirements of images and multiple database insertions during server operation, we realized this was insufficient. Taking both memory and overall volume size into consideration, we decided to upgrade to a better-spec EC2 server, t3.medium. This upgrade allowed us to smoothly carry out the deployment process.

Through these steps, we were able to overcome the initial limitations and successfully deploy the application.

3 Implementation

3.1 User

(i) Front-end

In the user registration page, user can sign up entering the email, password, and nickname. The user must verify that the e-mail is his/her own by entering an authentication number. The password must be at least 8 and less than 16 characters, and must include at least one alphabet and at least one number. The nickname entered by user is checked whether it is duplicated with other users'. After completing all these validations, the user can sign up.

In the login page, user can sign in by authenticating with entered email and password. After signing in, we use the Json Web Token for authorization. The token is a string created by the server after login. When the login succeeds, the server issues two tokens: access token and refresh token. The access token is used to authorize the user for protected operations and expires after one hour. It is reissued using the refresh token which does not contain user information and expires after one day. The access token and the refresh token are stored in local storage and cookies, respectively.

(ii) Back-end

The implementation related to user management heavily relies on the JWT (JSON Web Token) module. Therefore, instead of explaining the basic CRUD functionality of the database, We will focus on the implementation aspects related to JWT.

When a user logs in, the backend verifies the existence of the user's information in the database and checks if the provided password matches. If the verification is successful, the backend issues an access token and a refresh token. The issuance process is carried out using the 'jsonwebtoken' library, and the key required for issuing the tokens is securely stored in the '.env' file.

To handle token issuance and verification, a module is defined with functions such as issuing access and refresh tokens upon initial login, verifying the validity of the access token, checking the refresh token to renew the access token upon expiration, and obtaining a new refresh token when the current one expires. These functions allow continuous updating of the user's login status.

Furthermore, to ensure that certain categories require user authentication, middleware is used. Three middleware functions are employed: one to verify the access token and check if the user is logged in, another to check the refresh token if the access token has expired, and finally, a middleware function that allows access to the category regardless of whether the user is logged in, but also parses and provides the user's information if they are logged in. These middleware functions enable the implementation of features such as checking user permissions for modifying posts and restricting access to certain functionalities for non-logged-in users.

Through these implementation approaches, the system can effectively manage user login states and provide appropriate access control based on JWT tokens.

3.2 Home

(i) Front-end

Firstly, the top section of the homepage features three types of banners. These banners automatically rotate every 3 seconds.

Below the banners, there is a section displaying unanswered community questions. Clicking on a question takes users to the respective question page where they can provide answers. Additionally, the popular cocktail recipes of the week

are showcased. Up to 10 recipes are displayed based on the number of recommendations received during that week. The answers and recipes are implemented as slides, allowing users to navigate using buttons. On mobile screens, the buttons are removed, and users can easily navigate through the slides by swiping.

Below the recipes, nearby cocktail bars are shown. This functionality is similar to the cocktail bar map page on the website's map page.

At the bottom of the map, the popular community posts of the week are displayed. Users can preview the top 4 posts with the highest number of recommendations for the week.

(ii) Back-end

There is an API to find questions posts waiting answers, popular cocktails, and surrounding cocktails bars. First of all, the question post will be included in the list if the article has a category of 2, that is, a question, and there is no answer in the comment. Next, the popular cocktail recipes for the week are shown in the order of the highest number of 'likes', so use the COUNT function. Last, the cocktail bars around me appear in almost the same format as the 'Cocktail Bar Map' tab, so it will be explained in that tab section. Finally, community posts, like popular cocktails, are shown regardless of the community category, in the order of the highest number of 'likes' for the week.

3.3 Recipe

(i) Front-end

The recipe website features an edited cocktail API that show approximately 100 basic recipes. The website incorporates infinite scrolling and displays user nicknames, ratings, recommendation counts, and review counts as individual items. Additionally, there is a filtering function available to search for recipes. Users can filter by search term, ABV, color, recommendation count, and latest additions.

On the recipe write page, users can submit their own recipes. Required fields include cocktail name, color, ingredients, ABV, and recipe description. If an image is attached, it will be displayed; otherwise, a default image will be shown. The submitted recipes are added to the recipe list. If a user has created a recipe, they can also modify or delete it. In case of an incorrect or inappropriate recipe, there is a reporting function. If multiple reports are received, an admin will delete the recipe.

The recipe detail page provides detailed information about the recipe, such as ABV, color, list of ingredients, and recipe description. Users can write reviews, recommend or report the recipe from this page as well.

(ii) Back-end

1. Database

Cocktail data was collected using the OPEN API that can obtain cocktail information. Two APIs were used, first 70 cocktail information were taken from 'Ninjas' and 24 from 'TheCocktailDB'.[3], [4] Because data was collected from two different APIs, the structure that described the material information and the manufacturing method was very different. Therefore, it was difficult to unify the structure to implement the recipe function. Therefore, rather than making OPEN API requests in real time when users request, the method of storing pre-received data in the MySQL Database was used.

The 'Cocktail' table stores cocktail names, alcohol levels, colors, instruction, and image paths, and the database structure is set up by referring to recipes and color tables, and the imported data is transferred. In this process, the 'Recipe' table was divided into ingredient names, amount, and unit.

Information on cocktail color and alcohol levels was not included, so it was collected using 'chat GPT'. Cocktails come in a total of 12 colors: red, orange,

yellow, green, blue, purple, pink, black, brown, gray, white, and colorless. And the data in 'TheCocktailDB' included an thumbnail image link, but the images for 'ninjas' data was collected directly because there was no thumbnail and the path was saved. Lastly, all the data was collected in English, so it was translated into Korean with the help of 'chat GPT'.

2. Recipes APIs

a. CRUD

Among the APIs related to recipes, Recipe CRUD will be described first. Originally, POST, GET, PUT, and DELETE methods should be used respectively, but Recipe contains photos, so POST must be used to receive images through 'Formdata' format, so 'Update' was implemented through POST method.

Internally, the user-entered recipe information is validated. Check the database constraints to see if the cocktail name does not duplicate with the existing cocktail, and if the ingredients names do not duplicate in one cocktail. Regarding images, because members can optionally upload photos when post Recipe, the photo of cocktails uploaded without photos will be replaced by the default photos, logo photos.

b. Cocktail List Filtering

The following features are related to filtering and sorting in the cocktail list. With about 100 cocktails registered by default, this feature, including search, is essential to find the cocktails you want easily and quickly. There are three options: "Color," "Alcohol Level," and "Keyword." "Keyword" is applicable if any of the cocktail names, ingredients, and instructions correspond to the keyword. If users select more than one option to search, the same option is filtered through 'OR' and the other option is filtered through 'AND' operation. These filtered lists are sorted by last order, or by the order of 'likes' received, to be determined by the list.

In addition, to implement infinite scrolling, another form of Paging, 'offset' and 'limit' are set to search up to 12 cocktails from the specified offset once loaded.

c. Cocktail Detail Information

The recipe details should show ingredient information and instruction, the number of likes, whether the user likes, and related posts, so data are searched and displayed in various tables based on the cocktail's ID, which is 'CNO'. In the case of the 'Like' feature, the correct number can be updated by sending the cumulative number in response. The same API implements the click-off function so that user don't have to press 'like' repeatedly. Finally, the 'report' feature is also restricted from allowing one person to report more than once in a single post.

3.4 Community

(i) Front-end

The community pages on our website can be categorized into four main sections: the Community Home page, the Board page, the Detail page, and the Posting page.

The Home page features a Hot Posting section that displays the top three posts of the week based on the number of likes they have received. Additionally, there is a New Posting section which presents a grid format of four mini boards, each sorted by category. There are 4 categories in total, which are cocktail recommendation, question, cocktail review, and free.

On the Board page, users can utilize a search bar and category selection menus to filter the displayed postings. This search functionality works on both

the Community Home page and the Board page. Each item in the list is linked to the Posting Detail page, where users can access more information about a specific post.

Within the Posting Detail page, if the currently logged-in user is the author of the post, they have the ability to modify or delete it. For users who are not the authors, these options are replaced with a report button, which opens a report modal to notify the moderators of any issues. Up to three images associated with the post are displayed, and users can view all images by opening the image modal. Users can also write, edit, and delete replies to the post. The design of replies may vary depending on whether the post falls under the question category. To provide a more user-friendly experience, the time of each post is displayed as n hours ago, or n days ago, instead of the exact date. When editing a reply, the original content is pre-filled in the reply input field, along with the cancel and confirm buttons. When deleting posts or replies, a confirmation modal is displayed to ensure that the user's decision is intentional before permanently removing the content.

Both the Community Home page and the Board page feature a posting button, which becomes visible once the user logs in. On the Posting page, users can select the category for their post and upload up to five images, along with providing a title and content. The title and content fields are mandatory, except for posts in the question category, where the title field is hidden. Instead, the posting text for question category posts is displayed in bold and cannot be modified or deleted. In the cocktail review category, the title field offers an autocomplete feature with cocktail names to prevent random inputs from users. This category also includes an option for users to indicate whether they would recommend the cocktail or not. This recommendation status is later shown in the posting list with a corresponding icon.

(ii) Back-end

The community feature involved repetitive CRUD operations. We utilized "multer" library for uploading image. During the database design phase, we went through a process of decomposing tables to achieve database normalization. The 'POST' table, for instance, had a primary key representing the unique post ID, and it included other essential and mandatory attributes such as user information, post category, title, content, creation date, and modification date. In addition, since each post has a unique image, we also made a 'IMAGE_COMMUNITY' table to retrieve which image is related. While we aimed for normalization, there were additional attributes specific to the cocktail review category, such as cocktail information and recommendations, which prevented achieving perfect normalization. However, considering the complexity of programming, we designed the tables with a focus on development efficiency.

In addition to the 'POST' table, there were other tables to store related information. These included the 'POST_LIKE' table to store information containing who likes the post, the 'POST_REPLY' table to store who write the comment, and the comment information, and the 'POST_REPORT' table to store post reporting information. We established referential integrity by defining relationships between multiple tables, ensuring the consistency and integrity of the data.

Through this design, we aimed to strike a balance between normalization principles and development efficiency, allowing for effective data management in the community feature.

3.5 Map for Cocktail Bar Information

(i) Front-end

We use the Geolocation API to ask location access permission to user. If the user's current location cannot be retrieved because the user blocks location

access, the cocktail bars within 1 kilometer centered at the Hyehwa-dong, Seoul will displayed on the map . If the user's current location can be retrieved, cocktail bars within 1 kilometer centered at the user's current location will displayed on the map. Also the name, address, review keywords, and the latest two review of cocktail bars are displayed next to the map. Kakao Map JavaScript SDK is used to display the map on the page.[1] To use this SDK, we insert the script tag to the HTML file. The user can search cocktail bars with name or location. To search cocktail bars with location, address must be converted to coordinates, and Korea address list data must be available. To implement this, we use the Google places API to get the Korea address list, and the Google geocoding API to translate the address to the coordinates. If the user clicks the marker or the place name, the user can see the cocktail bar detail page. In the cocktail bar detail page, the user can see the whole review list. The user can post the review with the rating, review text, keywords, and images.

(ii) Back-end

1. Data, Database, Kakao APIs

   In order to implement a map for cocktail bars, the location information of the cocktail bars is required. To collect this information, the OPEN REST API provided by Kakao was used. It is called Kakao Local Search APIs.[2] When the authentication key is set to the header, up to 45 locations can be retrieved, and additional search with specific latitude, longitude, and range can be performed. Keywords containing region names are also supported by the API itself, allowing users to have a more convenient search experience.

   Since only location information for the cocktail bar should be provided, only companies classified as 'cocktail bars' were separately filtered from the metadata of the information received from the API. The information provided is the company name, address, road name address, phone number, latitude and longitude. Since this API is used in real time while the service is provided, access to cocktail bar information across the country is possible, which is continuously updated by the Kakao service.

   This service manages the location information collected from the API and stores or updates the information received by the API on the PLACE table based on the `PLACE_ ID` which is the primary key of the table, at that time to connect to the review data corresponding to each place.

   This allows users to write or inquire cocktail bar reviews based on keywords, images, and text for information focused on cocktails that are not provided by other services.

2. Review

   To facilitate posting reviews for cocktail bars listed on the map, a 'REVIEW' table has been designed. This table includes attributes such as REVIEW_ID (a primary key that uniquely identifies each review), UNO (a foreign key referencing the 'USER' table to represent the unique user number), PLACE_ID (a foreign key referencing the 'PLACE' table to provide information about the cocktail bar), TEXT (for storing the content of the review post), RATING (for capturing the rating of the cocktail bar), createdAt (for storing the date when the review was created).As same as Community post, User can upload images using the multer library. Furthermore, to accommodate keywords associated with each review, a separate 'KEYWORD' table has been created, allowing for a maximum of three keyword rows per review in the 'KEYWORD' table.

3.6 My-Page

(i) Front-end

On the My-Page, users have access to various features to manage their account. They can modify their nickname, view user level information, verify bartender certificate, and review their contributions, including recommended cocktail recipes, postings, replies, and reviews. Up to three entries are displayed for each category, and users can view more entries in each respective category. Non-clickable arrows indicate the absence of entries. The detailed pages for each category also implement infinite scroll, allowing users to easily navigate through the content.

There are a total of three basic member grades, consisting of Level 1 achieved by signing up, Level 2 achieved by posting more than 10 posts, and Level 3 achieved by posting more than 30 posts. In addition, bartender authentication becomes a bartender level. The user's level automatically upgrades once met the conditions, except for the bartender level. To verify bartender certificate, users must provide their name, birth date, certificate number, issue date, and certificate paper number. The form submission is not possible if any of these inputs are left empty.

For changing their nickname, users are presented with a nickname change modal that requires a new nickname. The new nickname must be between 1 and 10 characters, consistent with the nickname setting during the signup process. Additionally, users have the option to delete their account directly from the My-Page.

(ii) Back-end

1. User History and Level

   The API related to My Page is basically based on the previous activity record of the logged-in user. They include cocktails marked "like" by users, community posts registered by users, comments written by users, and reviews left by users. All APIs operate based on when the user is logged in, so they do not respond if there is no token in the header.

   Like many other pages, it shows the user's activity history in the form of a list, so if there are many corresponding items, paging, or infinite scrolling, is required. Therefore, based on 10, more data is loaded only when requested, so users do not have to wait for a lot of data to load.

   My Page is in charge of the user's member rating management function as well as the operation of the activity details. When you access My Page and load member information, the rating is adjusted by automatically counting the number of posts of the member. However, there is a rule that once the grade is raised, it does not fall.

2. Bartender Certification

   For bartender certification, There is no OPEN REST API that verifies bartender qualifications, so the API was created indirectly by referring to the network request used by the site(Q-net) that verifies the authenticity of the bartender.[5]

   When accessing the certificate authenticity verification site, the user enters the name, date of birth, license number, date of issue, and certificate number to verify the authenticity of the certificate. By analyzing the network request to do this, the required field names `hgulNm, resdNo1, lcsNo, qualExpDt, lcsMngNo` were identified and used. Also, 'iconv-lite' module was used to encode because the name must be encoded using 'euc-kr' to operate normally. Since html is returned as a response, the 'cheerio' module is used to parse it. If the submitted information is a correctly issued certificate, the class value of the span tag of the response html is `ff_zh`, so by bringing the text of the tag, the certification is confirmed.

3.7 Admin

(i) Front-end

We implemented the pages for the administrator. If someone who is not authenticated requests the home page, the page redirects to the login page. If not, the administrator can see the home page. In the home page, the administrator can see the reported recipes and community posts, the reason for the report and who reported it. When the administrator clicks the title of content, a link goes to the "Cocktell" distribution site. Also, the administrator can delete the content.

(ii) Back-end

We create the ADMIN table which distinguish from the USER table. It has the id, email, and password attributes. We implemented the admin login API for authentication. The email and password in the request body is compared with the email and password in the ADMIN table. If both have matching admin accounts, an access token is issued. If not, return an error. We implemented the API for getting the report information and the API for deleting the reported content. To check whether there is permission to access report information and delete content, We implemented the middleware that checks if there is a token issued by the server in the Authorization header.

4 Evaluation

4.1 Usability Test

The Usability test was performed with the instruction below.

1. Home, Some Recipe Features, Community Available When Not Logged In
2. Other functions can be used only after signing up and logging in
3. How to use cocktail recipes
   a. Using search, alcohol level, and color filtering / Using sorting functions
   b. Recipe Details: "Like" Function, "Report" function, review writing, inquiry, modification, deletion function
4. How to use the community
   a. Posts can be created, viewed, modified, or deleted by category
   b. Upload up to 5 photos
   c. "Like", "Report", and write comments
5. How to use the cocktail bar map
   a. Location acceptance required
   b. Search Cocktail Bar
   c. View, create, modify, and delete reviews
   d. Upload up to 5 photos
6. How to use My Page
   a. Check the membership level
   b. Certification function of the bartender
   c. Modifying Nickname
   d. Check my activity history

4.2 Result

The Usability test was conducted based on several questions. A summary of the answers to each question is as follows.

1. What was your overall impression and usage?
   – It was good that the overall UI was luxurious and had a young sensibility to match the concept of cocktail

– The basic composition and design of the web, which should be equipped for community-related pages, are well equipped.
– It feels neat and easy to use, but the quality of the design is disappointing in some details (font, dark background).

2. Was it okay to use both desktop and mobile?
– mobile environtment

It works well. Even though it was a web page, it was easy to use as if the app was used, and the design was neat. It would be much easier to click if the size of the handwriting or button was a little larger.

– desktop environtment

It works well.

3. Does the function specified in the instruction work properly?

The majority of the respondents said they all worked well. Errors are addressed in the following questions.

4. Were there any errors?

Small errors were reported, such as not reflecting the number of reviews, toggling when pressing the like button, not reflecting the line change, and leaving the previous content when opening a new post after writing, but all were fixed.

5. Good point
– It's cool that the image of the banner part changes in real time.
– It is good that it is divided into several bulletin boards such as recommendations/reviews/question answers/free boards within the community.

6. Points to supplement
– It would be good to enable to check the "Recipe List I Made" on My Page.
– It would be good to further subdivide the bulletin board by age group or region.

5 Limitation and Discussions

5.1 Admin

We received feedback on how administrator can always manage pages. It was feedback that it would be better if the page was automatically maintained without an administrator. However, upon further consideration and discussions, we realized that having an administrator is essential for effective monitoring and management. Almost all websites now have administrators to monitor. We also judged that it is impossible to maintain all situations automatically. Instead, we added an admin page and reporting function. This allows users to report any issues or inappropriate content, which can then be reviewed and resolved by the administrator. Our team will process these reports periodically on the admin page. We solved the feedback with this alternative.

5.2 Alcohol Level

At first, we thought of a way to calculate the alcohol level of cocktails automatically. we found a formula for calculating the alcohol level of cocktails, and we were going to calculate the alcohol level of cocktails entered by the user according to the formula and show it to the users. However, we decided that it was not easy to calculate the formula. First of all, the marking units of the materials are so diverse. we decided that it was not easy to show them all because many people used so many different units, such as ml, oz, drop, dash, and ts. So we divided the alcohol level into three units: low, medium, and high. Three degrees are classified and shown to users: 0 to 5 degrees is low, 6 to 15 degrees is normal, and 16 to 99 degrees is high. This simplified classification enables users to filter and search for cocktails based on their desired alcohol level. Although this approach may not provide precise alcohol measurements, it offers a practical solution considering the complexity of ingredient units and enhances the usability of the platform.

5.3 Image Right

We found cocktail images using search and stored them in the database passively. However, the copyright of these images is unclear. We will find an alternative to this part and revise it.

6 Related work

After conducting a thorough investigation of various mobile cocktail applications and cocktail-related websites, we identified several critical features that were lacking in those platforms, which we have implemented in our application. Specifically, we examined Masilang, Shaker, Cocktail Flow, and My Bar among the mobile applications. While some of these apps had basic functionalities such as recommendations, ingredient information, and recipe descriptions, they were missing essential features like user-added cocktail recipes, like/dislike functionality, bookmarking, and advanced filtering. Moreover, none of them provided a means for communication, quotation, or sharing of reviews about cocktail bars, which significantly impacted the reliability of the recipes.

In contrast, our application, Cocktell, offers all of these crucial features. To facilitate communication, we have implemented login, sign-up, and community features, enabling users to freely post content with images, engage in discussions, and provide replies or answers. The application also includes information about cocktail bars and reviews on the Map Page, facilitating the sharing of valuable information among users. To ensure reliability, we have introduced a bartender level system with verification, instilling confidence in the accuracy of the information presented. Additionally, Cocktell boasts advanced filtering capabilities, providing 12 color categories and ABV-based sorting options.

We also conducted investigations into cocktail-related communities on the web, including the DC Inside Cocktail Gallery and DC Inside World Liquor Gallery. These communities operate within the larger DC Inside site and offer subject-specific forums for users. However, their formats and structures are not optimized for cocktail-related communication and primarily serve as basic community platforms. Furthermore, the high degree of freedom within the site often hinders user engagement, and there is a lack of guaranteed information reliability.

In contrast, Cocktell provides a user-friendly interface with sorted and filtered cocktail recipes, making it easier for users to find specific recipes. The categorized community page further facilitates organized postings and discussions. Not to mention Cocktell provides trusty information.

In conclusion, our application, Cocktell, surpasses existing mobile cocktail applications and cocktail-related websites by offering a comprehensive set of features that address the shortcomings of other platforms. Its robust communication features, reliable information sources, advanced filtering options, and user-friendly interface make it an ideal choice for cocktail enthusiasts seeking a comprehensive and trustworthy application.

7 Conclusion

The web application Cocktell provides a comprehensive solution for cocktail beginners and enthusiasts, offering cocktail recipes, a cocktail bar map, and a reliable cocktail community. Unlike existing apps and websites, Cocktell combines recipe information with a community platform, addressing the need for a reliable and high-quality resource in the cocktail community which can be provided by verified bartenders.

The web application consists of seven main pages: Registration, Login, Home, Recipe, Community, Cocktail Bar Map, and My-page. Users can register or login using JSON Web Token, and check the issues in the home page. In recipe page, users can filter, sort, and contribute their own cocktail recipes with images, as well as report any inaccuracies. The Community page categorizes posts, and users can search posts by keywords, create, edit, and delete their own posts and comments. The Cocktail Bar Map page allows users to locate nearby cocktail bars and read, write, modify, and delete reviews within a 1-kilometer radius of their current location, by utilizing the Kakao Map API and Google Places API, Google Geocoding API. The My-page provides users with the ability to modify their nickname, verify their bartender certificate, and view their activity histories.

Overall, Cocktell effectively caters to the needs of cocktail beginners and enthusiasts, offering reliable recipe information, a dedicated community, and a convenient cocktail bar mapping feature. The positive feedback from usability testing further validates the usability and appeal of the web application. With its comprehensive features and user-friendly interface, Cocktell contributes to enhancing the cocktail experience for users at all levels of expertise.

References

1. Kakao Maps API, accessed Jun 7. 2023, https://apis.map.kakao.com/web/documentation/
2. Kakao local REST API, Kakao Developers, accessed Jun 3. 2023, https://developers.kakao.com/docs/latest/ko/local/dev-guide
3. thecocktaildb API, thecocktaildb, accessed Jun 3. 2023, https://www.thecocktaildb.com/api.php
4. Cocktail API, API Ninjas, accessed Jun 3. 2023, https://api-ninjas.com/api/cocktail
5. Certification Verification, Q-net, accessed Jun 3. 2023, https://www.q-net.or.kr/qlf006.do?id=qlf00601&gSite=Q&gId=

Acronyms

CRUD Create Read Update Delete. 2, 4, 6, 7

SDK Software Development Kit. 2, 8