CS30700

Design Document

Team 6 Members：

Yuanfei Song

Congtian Wu

Yichen Dai

Ekaterina Tszyao

Coordinator：

Jiayi Liu

Index

Purpose:

With the growth of social media and e-commerce, users often need help finding reliable and comprehensive product comparisons due to fragmented and biased reviews. Our Judge Everything addresses this by enabling users to create product profiles set vital parameters, and rate or review products freely and independently. Users can also compare multiple products and see consolidated scores, making it easier to make informed decisions. Unlike traditional platforms, Judge Everything is entirely user-driven, offering various products and real-time rankings based on authentic feedback, combining social media functions and allowing users to add some events or memes to be rated just for sharing.

Functional Requirements:

User Accounts:

1. As a user, I want to create an account so that I can securely access the app's features.
2. As a user, I want to log in to my created account to access the app securely with my previous history.
3. As a user, I want to reset my password by email so I can access my account even if I forget it.
4. As a user, I want to modify my account details (e.g., password, avatar, preferences, etc.) to reflect my current interests.
5. As a user, I want to be able to delete my account so that I can stop sharing my personal info with Judge Everything after I'm no longer interested in it.
6. As a user, I want an introductory page after registration to help me learn how to navigate the website.
7. As a user, I would like to access a contact page that lists all the necessary contact information (e.g., address, phone number, working hours).

8. As a user, I want to have the option to chat directly with a support officer and request real-time assistance from a support agent.

User Client:

- Rating & Reviewing System:

9. As a user, I want to view and browse product profiles created by other users to gather information that interests me.
10. As a user, I want to see an average score for each product based on its reviews to make an informed decision.
11. As a user, I want to be able to report inappropriate or wrong information about product profiles created by other users.
12. As a user, I want to be able to rate products on a 5-star scale using pre-defined or custom parameters so that I can share my opinions with others.
13. As a user, I want to be able to write comments for products to share more details of my opinions with others.
14. As a user, I want to view and access my comments and rating history, which will allow me to see replies to my comments and edit my previous comments or ratings.
15. As a user, I would like to create new product profiles with relevant details (e.g., name, category, images, description) so that I and other users can "judge" the product.
16. As a user, I would like to create custom parameters for product ratings, add them to pre-defined parameters, and rate the product according to my criteria.
17. As a user, I want to manage my posted products by editing the content, adding pictures, and modifying the attributes.

- **Searching, Ranking System & Compare:**

18. As a user, I want a convenient search system that allows me to quickly find specific information by entering a few keywords.
19. As a user, I want product profiles organized into basic categories (e.g., electronics, furniture, hygiene products) to navigate to the products relevant to my specific needs easily.

20. As a user, I want the option to view products in different listing orders (e.g., ascending by number of comments, descending by the number of ratings, most trending, most relevant, or by the initial post time).
21. As a user, I want to see product rankings across various categories (e.g., most popular, highest rated) to identify the top-rated products quickly.
22. As a user, I want to see product rankings within the same categories(e.g., Nike or other sports brands) to identify top-rated products in specific categories quickly.
23. As a user, I want to select some products and get a direct diagram or information comparison, including critical parameters, top reviews, and average scores, to see which product is better for me overall.

- **Recommendation System**:

24. As a user, I want to see content that aligns with my interests, based on my past activities, to provide me with the most relevant content.
25. As a user, I want to receive updates on new information, trending content, and personalized recommendations through emails and system notifications.
26. As a user, I want to see related products when viewing a specific product, providing more relevant information I might need.

- **History:**

27. As a user, I want to see my past search history displayed under the search bar so I can easily access information I may want to revisit.
28. As a user, I want to see my browsing history in case I need to revisit previously viewed content.
29. As a user, I want to view the history of my posts or my comments to other users.

- **Chat & Social Channels:**

30. As a user, I want to chat with other users via a live chat system to discuss products, events, or other shared interests.
31. As a user, I want to be able to follow other users so that I stay updated on their reviews and ratings in my feed.
32. As a user, I want to rate and share events and memes (non-product entries) in a separate section (Social Channel) to engage in social interactions.

33. As a user, I want to comment on other users' reviews so that I can engage with their content.
34. As a user, I want to like/dislike other users' reviews to engage with their content.
35. As a user, I want to receive notifications related to my account (e.g., new followers and chat messages)
36. As a user, I want to receive notifications when others interact with my posts (e.g., likes and comments) so that I stay updated.

- **Audit:**

37. As a user, I want to receive a notification if my content is deleted or edited by an administrator because the content I posted is inappropriate or incorrect.
38. As a user, I want the option to appeal if my content is deleted or edited by an administrator so that my proper content isn't accidentally deleted/modified.
39. As a user, I want to be notified that my report has been processed.

Head Admin Accounts:

40. As a head admin, I would have a head administrator account that can manage all the administrator accounts.
41. As a head admin, I would have a search system for administrator accounts.
42. As a head admin, I want to be able to create new administrator accounts.
43. As a head admin, I want to be able to delete any administrator account.

Admin Accounts and Accessibility:

44. As an admin, I should be able to log in to a separate management interface using my administrator account.
45. As an admin, I should be able to approve or deny user requests to create, edit, or delete product profiles to ensure data quality.
46. As an admin, I want the platform to automatically detect and flag sensitive or inappropriate review content to help me maintain a respectful community.
47. As an admin, I should be able to manage and moderate flagged content (e.g., inappropriate reviews) to maintain a safe and respectful platform.

48. As an admin, I want the platform to automatically detect when a product profile has been reported more than a certain number of times and inform me that I need to review it again.
49. As an admin, I want access to user activity logs to monitor the platform's usage and enforce rules when necessary.

Non-functional requirements:

1. Architecture and performance: The application will use React JS for the front end, Express JS for the back end, and Node JS for the data, which will be hosted with the Firebase Realtime Database.

- Scalability: The platform supports up to 100 simultaneous connections.
- Latency: The application responds to user requests within 500ms.

2. Security: The application supports independent security and data security based on Firebase rules. The browsing and comment histories will also be stored in the Cloud Firestore to keep each user's private history secure and separate. Our sensitive word system will also be able to block users from exchanging other social media accounts (if we have time).
3. Usability:

- Our platform can fit different window sizes on various devices.
- Our platform supports the mobile app version. (if time allows)
- The platform should have a clear and understandable UI design.
- The platform should be easy to navigate for users.

4. Hosting and Development:

- Judge Everything's hosting will be handled by Firebase, which hosts the Cloud Firestore data. The free "Spark" Plan of Firebase allows for 10 GB stored, 10 GB transferred, 1 hour/day's virtual device testing, and 30 minutes /day's physical device testing. Authentication services such as emails and identity platforms (google, etc.)
- For the amount of monthly active users, Firebase supports 50k/month for the free plan.

Design Outline

High-Level Overview

This project will be a web platform that allows users to create and publish reviews on various products while maintaining complete authenticity. The platform will enable users to post entries with detailed ratings on various product parameters and leave comments. Users can also browse, search, and filter through these entries using a robust search function, keyword, and tag system to find products of interest efficiently. The platform will feature a recommendation system that suggests entries based on user behavior and profiles and a trending feature that highlights popular entries.

The application will use a client-server model, where Node.js and Express backend handles simultaneous requests from multiple users. Data will be managed and stored in a Firebase, and the frontend will be developed using React.js. The server will process user requests, manage product reviews and ratings, and provide necessary data to the client. Advanced features such as a recommendation algorithm, automated review mechanisms, and user-friendly navigation tools will enhance the platform's functionality and ensure an engaging experience for users.

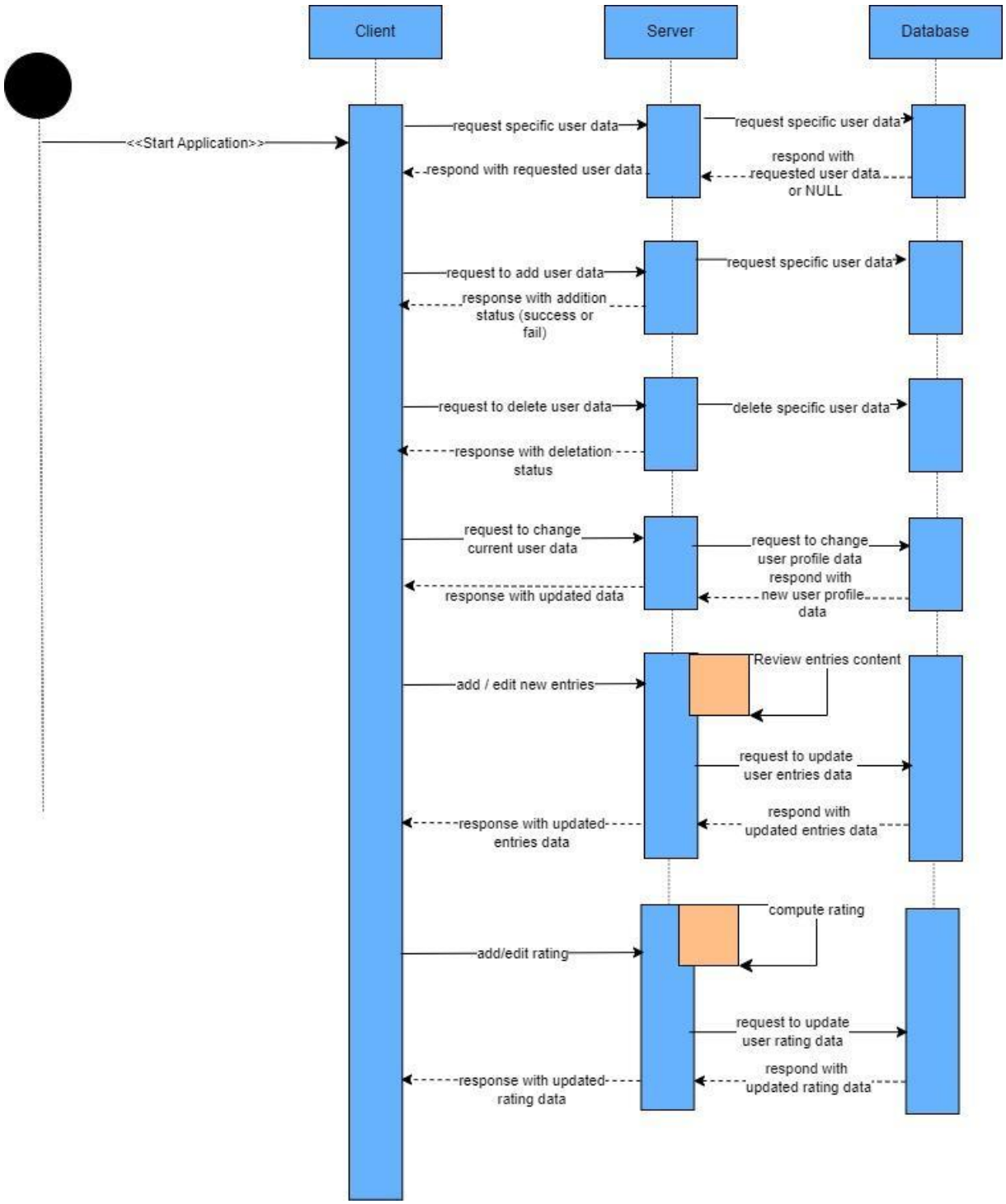
1. **Client**
    a. The client is a web interface built using React.js. It allows users to interact with the platform by browsing, searching, rating, and posting product reviews.
    b. To retrieve or submit data, the client sends asynchronous HTTP requests (such as GET and POST) to the server using modern methods like fetch() or Axios.
    c. The client receives responses from the server, interprets the data, and updates the user interface dynamically, reflecting changes such as displaying new reviews, personalized recommendations, or trending products.
2. **Server**
    a. The server is built using Node.js and Express and handles client requests, such as retrieving product reviews, managing user preferences, and processing new entries.
    b. It validates and processes the requests, interacting with Firebase to store, update, or retrieve data as needed. Firebase's real-time capabilities can also enable live updates to the client interface.
    c. After processing the data, the server sends the appropriate response back to the client, ensuring efficient data management and user interaction.
3. **Database (Firebase)**
    a. Firebase stores all application data, including product reviews, user profiles, ratings, and comments. It supports real-time data synchronization, providing users with instant updates.
    b. The Firebase database responds to queries from the server, sending the requested data back for display or processing by the client. Firebase also allows easy data storage and retrieval, making it ideal for handling user-generated content.

Sequence of Events Overview:

The sequence diagram for the "Judge Everything" platform illustrates the typical interaction between the client, server, and Firebase database. The sequence starts when a user launches the web app. Upon initiating the login process, the client sends a request to the server. The server processes this request, verifies the user's credentials, and queries Firebase for the necessary user data. Once the login is successful, the server returns the results to the client, allowing the user to access the platform.

After logging in, the user can perform various actions, such as managing their profile, searching for products, creating product entries, submitting ratings and comments, or following other users. Each of these actions sends requests from the client to the server. The server processes these requests and interacts with Firebase to store, modify, or retrieve the necessary data. Firebase returns the requested or updated data to the server, which is then relayed back to the client, updating the user interface dynamically based on the user’s actions.

For content moderation, when admins make changes to product profiles or user comments, the server updates the necessary records in Firebase and sends notifications to the affected users, allowing them to appeal any modifications made to their content.


Client
Server
Database
<<Start Application>>
request specific user data
request specific user data
respond with requested user data
respond with requested user data or NULL
request to add user data
request specific user data
response with addition status (success or fail)
request to delete user data
delete specific user data
response with deletation status
request to change current user data
request to change user profile data
response with updated data
respond with new user profile data
Review entries content
add / edit new entries
request to update user entries data
response with updated entries data
respond with updated entries data
compute rating
add/edit rating
request to update user rating data
response with updated rating data
respond with updated rating data

Design Issues

Functional Issues:

1. What information do we need for registering an account?

- **Option 1**: Username and password only
- **Option 2**: Username, password, email address
- **Option 3**: Username, password with email address as an optional input

Choice: Option 3
Justification: We will only require a username and password to speed up the registration process and minimize user frustration. Long registration forms are often tedious and can discourage users from completing the process. By making the email address optional, we allow users who value account security or want to receive notifications to add their email during registration or later. This approach balances convenience with flexibility, catering to casual users and those who prefer more secure accounts.

2. How should we manage content to ensure quality?

- **Full auto censorship**
- **Manual supervision**
- **Both**

Choice: Both
Justification: Automated censorship and manual supervision are the most effective approaches to ensuring content quality. Automated filters can help flag inappropriate content, reducing the workload for manual moderators. However, fully automated systems can be too rigid and mistakenly flag valid content. Manual supervision allows for more nuanced decisions, ensuring that legitimate content is not unfairly removed. This hybrid approach provides a balance between efficiency and a balance between efficiency and accuracy, maintaining quality while reducing moderation overhead.

3. How should we manage the storing of data to improve efficiency?

- **Simply use pointers to reference the data**
- **Create a general managing class, “Id”, to improve simplicity and readability**

Choice: Create a general managing class, “Id”
Justification: Using a general "Id" class improves the code's simplicity and readability by providing a unified system to manage and reference data throughout the application. This makes tracking, maintaining, and extending the codebase easier as the project grows. Relying only on pointers can lead to clarity and consistency, particularly when scaling the project or managing complex data relations.

4. How should we store and create data IDs to prevent repeated data?

- **Check for overlap when creating a new ID**
- **Design the ID declaration to ensure no duplicates**

Choice: Design the ID declaration to ensure no duplicates
Justification: Preventing the creation of duplicate IDs at the design level ensures the system is more robust and avoids the need for costly checks after the fact. By ensuring unique IDs from the moment they are declared, we prevent potential errors and reduce the complexity of the data management process. This proactive approach minimizes the risk of conflicts or duplication in the database, improving overall system integrity.

5. **How should users and admins interact regarding modification of user content?**

- **Option 1**: Admins modify or delete content without formal notice, then notify users post-modification, allowing appeals
- **Option 2**: Users are notified about required modifications, and if no changes are made within a given time, admins will modify or delete the content.

Choice: Option 1
Justification: To improve the efficiency of content moderation, admins will be allowed to modify or delete user content without prior notice. After the modification, users will be informed of the changes made to their posts, giving them the opportunity to appeal if necessary. This approach ensures that inappropriate content is addressed swiftly, without unnecessary delays,

while still providing users with the chance to contest the decision if they believe their content was wrongfully modified.

Non-Functional Issues

1. What database/framework should we use?

- **MySQL**
- **MongoDB**
- **Firebase**

Choice: Firebase
Justification: Firebase offers real-time data synchronization, which is ideal for a dynamic platform like "Judge Everything," where user-generated content is constantly updated. It also provides authentication features, seamless scaling, and integration with modern web technologies like React.js, making it well-suited for the requirements of this project. Additionally, Firebase is easy to manage and does not require complex server maintenance compared to relational databases like MySQL.

2. What frontend framework should we use?

- **React**
- **Angular**
- **Vue**

Choice: React
Justification: React is well-suited for building interactive user interfaces due to its component-based architecture, which allows for reusable UI elements. Its popularity and extensive community support mean that the team will have access to numerous resources and libraries. React also integrates well with Firebase, and its virtual DOM ensures high performance, which is crucial for a dynamic platform like "Judge Everything."

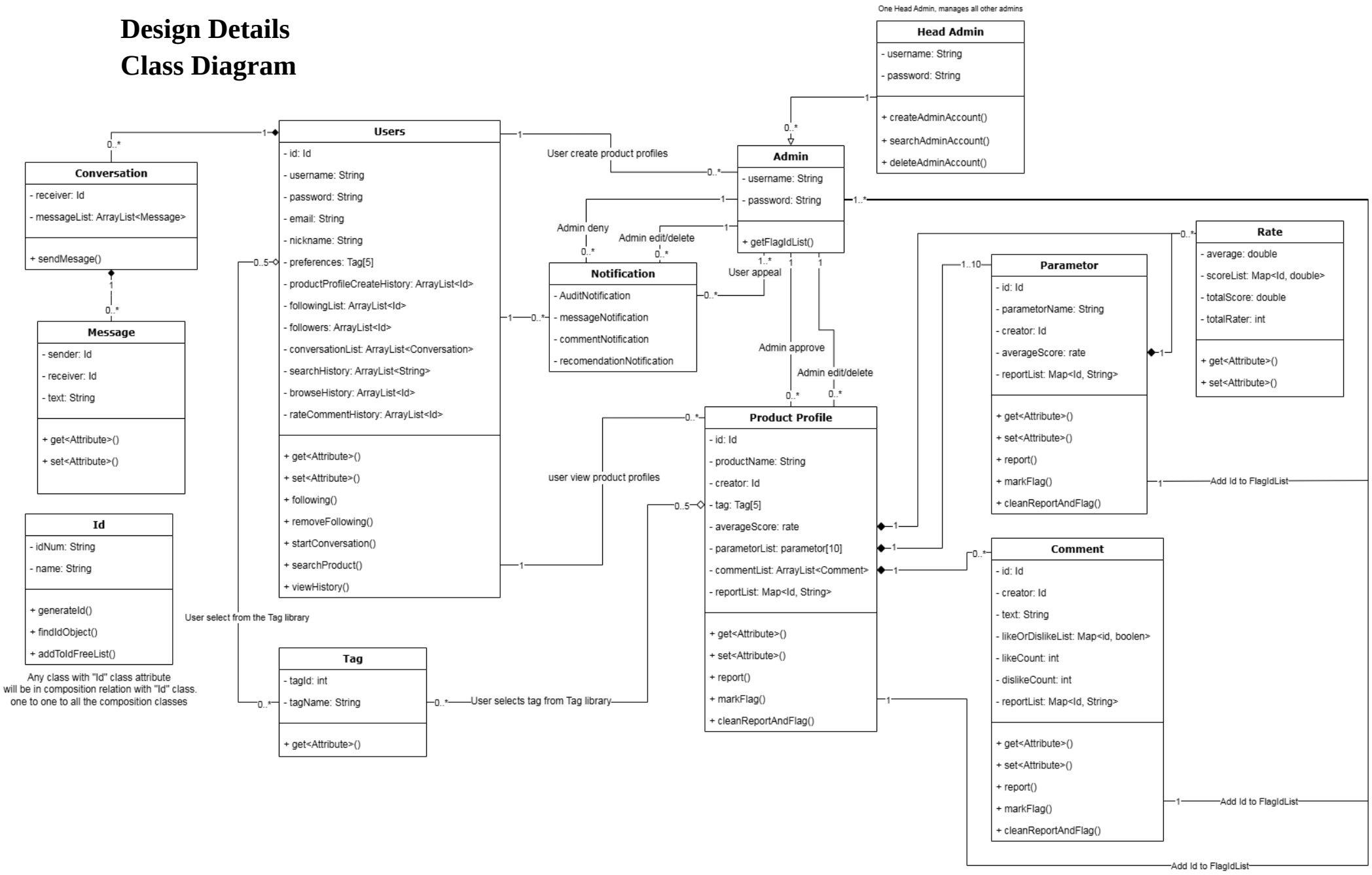
3. What backend framework should we use?

- **Django**
- **Spring Boot**
- **Node.js**

Choice: Node.js

Justification: Node.js, combined with Express, provides a lightweight and efficient backend framework that allows for fast development and scalability. Given that the frontend is built with React, using Node.js ensures that the entire application is JavaScript-based, simplifying development and improving communication between the client and server. Additionally, Node.js offers excellent support for RESTful APIs, which will be critical for managing user reviews and ratings.

Design Details
Class Diagram



Descriptions of Classes and Interaction between Classes

The classes are designed based on the objects in our application. Each class has a list of attributes which is the characteristics owned by each object.

1. **Id**:
    a. The Id class manages unique identifiers within the system.
    b. Each Id object is associated with an idNum and a name.
    c. Key methods include generateId() for creating a new unique ID, findIdObject() for locating objects by ID, and addToFreedList() for handling IDs that are no longer in use.
2. **User**:
    a. The User class represents a user account in the system.
    b. Each user has a unique id, username, password, and optionally an email.
    c. Additional attributes include a nickname, tags (used for search and recommendations), and lists for managing following, followers, search and browsing history, and rating/comment history.
    d. Key methods:
        i. getAttributes(): Retrieves user details.
        ii. setAttribute(): Modifies user attributes.
        iii. searchProduct(): Allows users to search for products.
        iv. following(), removeFollowing(): Manage the user's following list.
        v. viewHistory(): Displays user browsing and search history.
        vi. startConversation(): Starts conversation with another user.
3. **Message**:
    a. The Message class handles private communication between users.
    b. Each message contains a sender (identified by id), a receiver, and the content text.
    c. Messages allow users to communicate with each other based on interactions on the platform.
4. **Notification**:
    a. The Notification class manages alerts and updates for users.
    b. It has attributes such as adminNotification, messageNotification, commentNotification, and recommendationNotification.
    c. Notifications are used to inform users about interactions on their content, recommendations, or admin actions on their posts.
5. **Admin**:
    a. The Admin class represents an administrative user who manages content on the platform.
    b. Admins have a username and password, similar to regular users.
    c. They have access to advanced actions, such as approving user appeals, modifying content, and deleting inappropriate posts.

    d. Admins can interact with users through the Notification system.
    e. Admins can view falged content using getFlagIdlist(), to acquire the Ids of flagged object.
6. **Head Admin**:
    a. The Head Admin is a special class for top-level administrative users.
    b. They have the ability to createAdminAccount(), searchAdminAccount(), and deleteAdminAccount().
    c. This class holds ultimate control over administrative users on the platform, enabling them to manage the admin hierarchy.
7. **Tag**:
    a. The Tag class is used to categorize and organize products.
    b. Each tag has a unique id and a tagName.
    c. The getAttributes() method allows retrieval of tag information.
    d. Users can search for products using tags, and tags help organize content for both users and admins.
8. **Product Profile**:
    a. The Product Profile class represents a product entry on the platform.
    b. Each product profile has a unique id, a productName, and the username of the creator.
    c. Tags (Tag), average scores (Rate), and lists of parameters (Parameter[10]) and comments (ArrayList<Comment>) are associated with the product profile.
    d. Also contains reportList to store all the report information for the admin to review.
    e. Methods such as get/setAttributes() allow retrieval of product data, report() and markFlag() lets users report problematic content. cleanReportAndFlag() is for admin uses.
9. **Parameter**:
    a. The Parameter class stores detailed attributes that are rated for each product.
    b. Each parameter has a parameterName, the creator_username, and its average score (averageScore).
    c. Methods such as get/setAttributes() allow retrieval of parameter data, report() and markFlag() lets users report problematic content. cleanReportAndFlag() is for admin uses.
    d. Parameters allow users to rate products based on multiple factors.
10. **Comments：**
    a. The Comment class handles user comments on product profiles.
    b. It includes the creator_id, the text of the comment, and likeCount/dislikeCount to track user engagement.

c. Methods such as get/setAttributes() allow retrieval of comment data, report() and markFlag() lets users report problematic comment. cleanReportAndFlag() is for admin uses.

11. **Rating**:
    a. The Rating class is used to aggregate and store the rating information of products or parameters.
    b. It holds values such as the average rating, a scoreList (mapping users to their scores), and totalScore.
    c. Methods allow retrieval (getAttributes()) and modification (setAttribute()) of rating information.
12. **Conversation**:
    a. The Conversation class is responsible for managing the communication between two users on the platform.
    b. receiver: Id - The unique ID of the user receiving the messages in the conversation.
    c. messageList: ArrayList<Message> - A list of Message objects representing the conversation history between the users.
    d. sendMessage() - This method allows a user to send a message to the receiver. It appends the new message to the messageList, updating the conversation history.

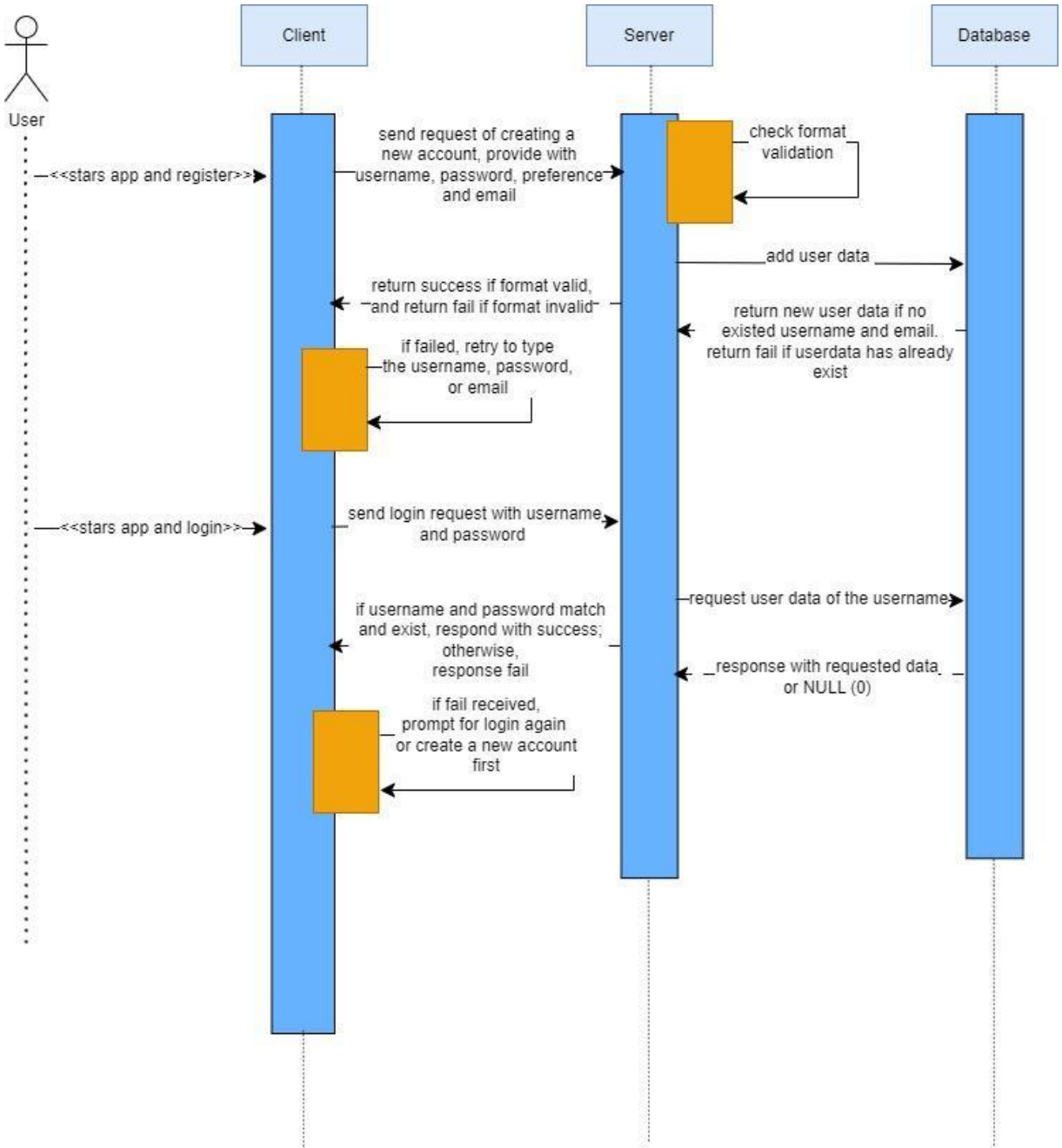
Sequence Diagram Overview for "Judge Everything"

The following sequence diagrams depict five key interactions in the "Judge Everything" platform:

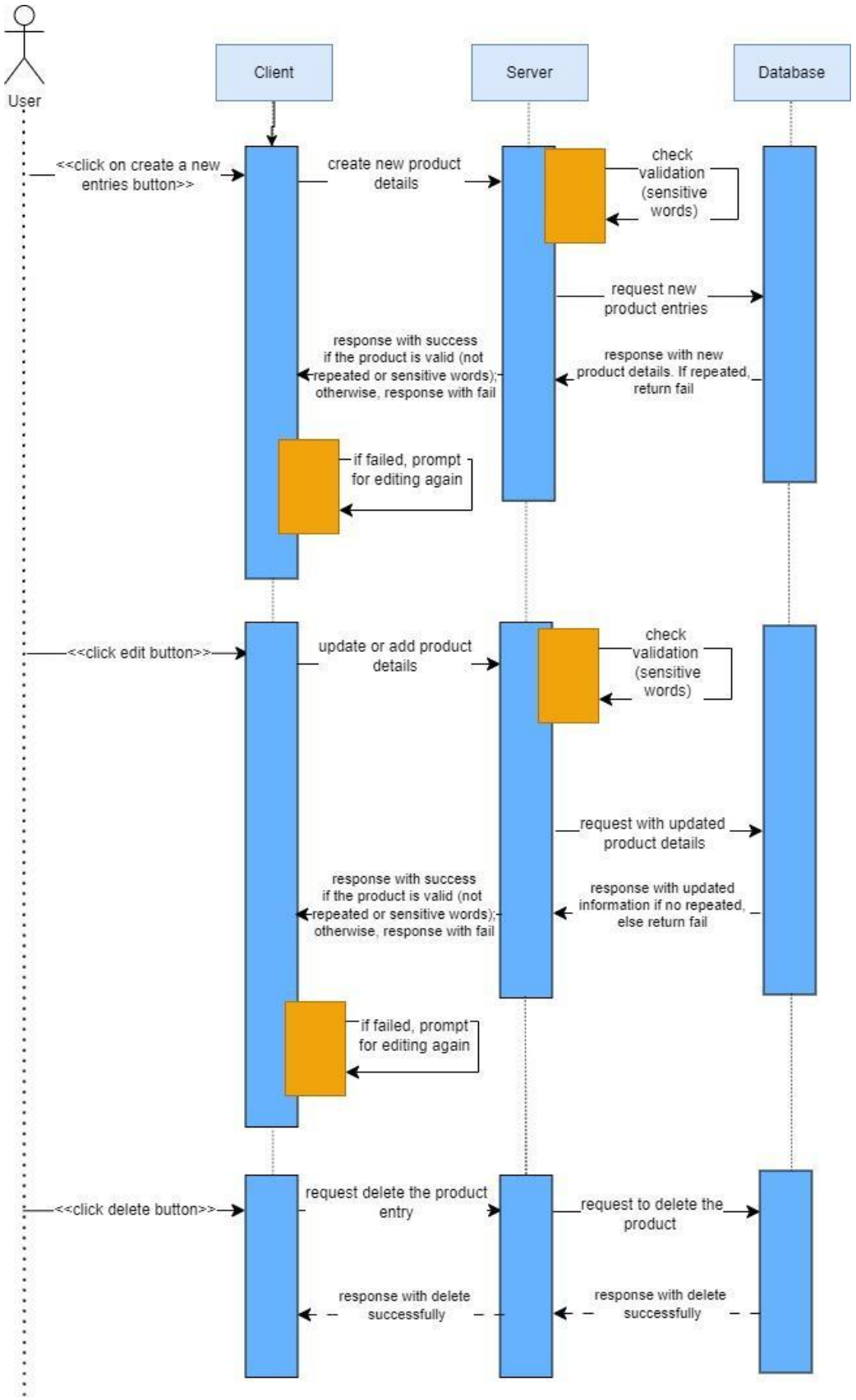
1. **Login and Register Process**
2. **Creating/Deleting a Product Entry**
3. **Search System**
4. **Adding/Modifying/Deleting Comments and Reporting a Product**
5. **Admin Creating and Deleting User Accounts**

Each sequence demonstrates how the client, server, and Firebase database interact in a client-server model to process user actions.

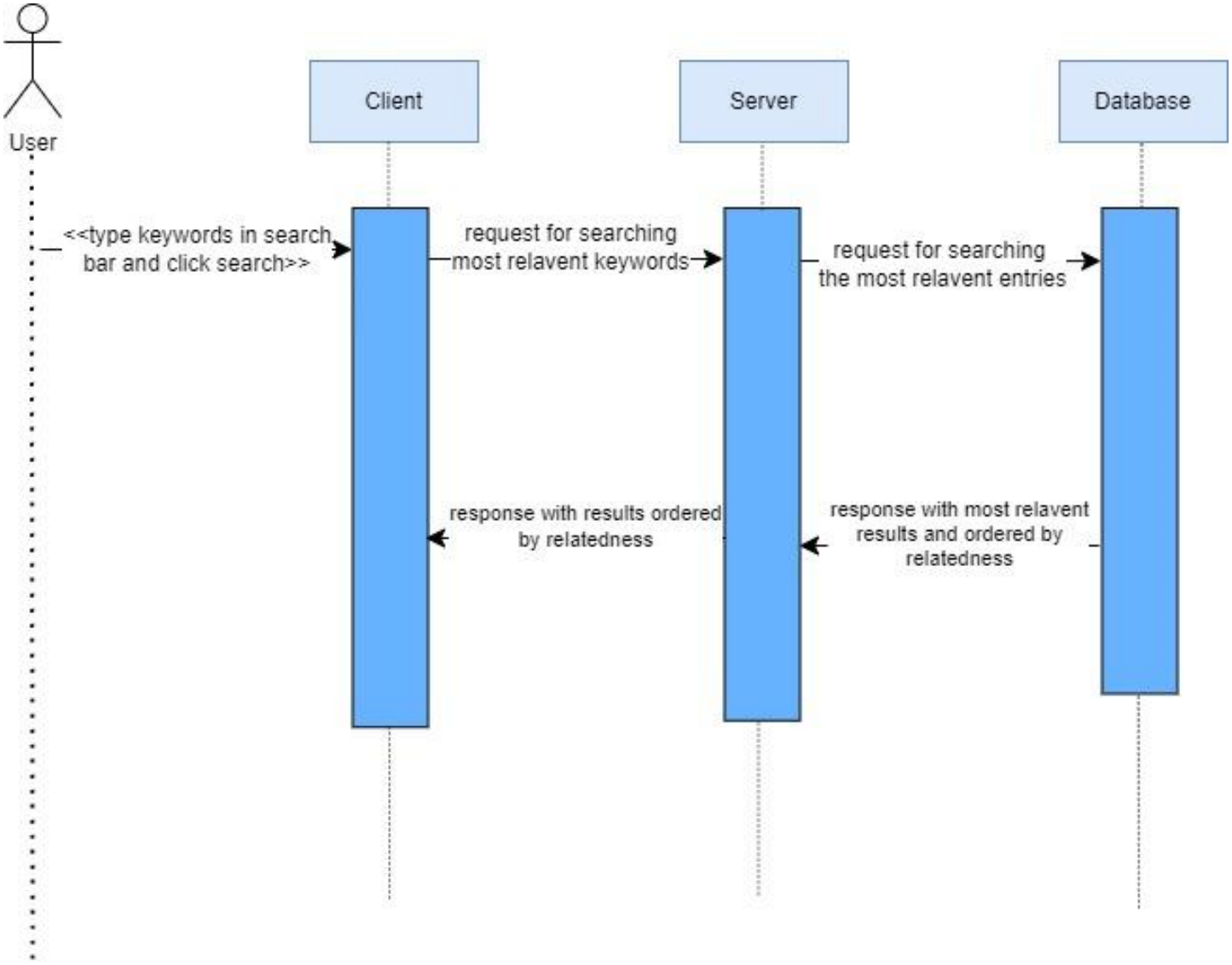
1. **Login and Register Process**



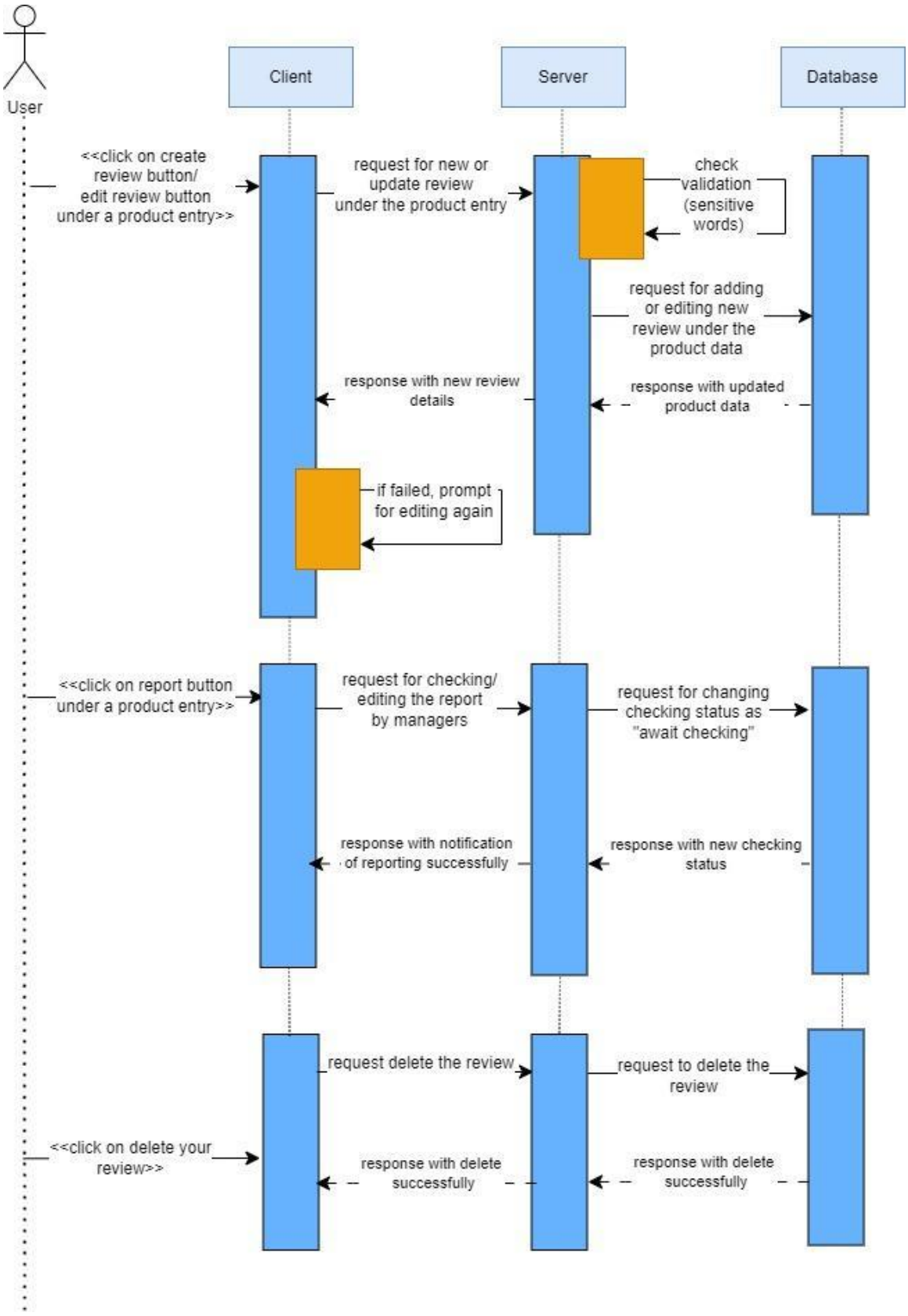
2. Creating/Deleting a Product Entry



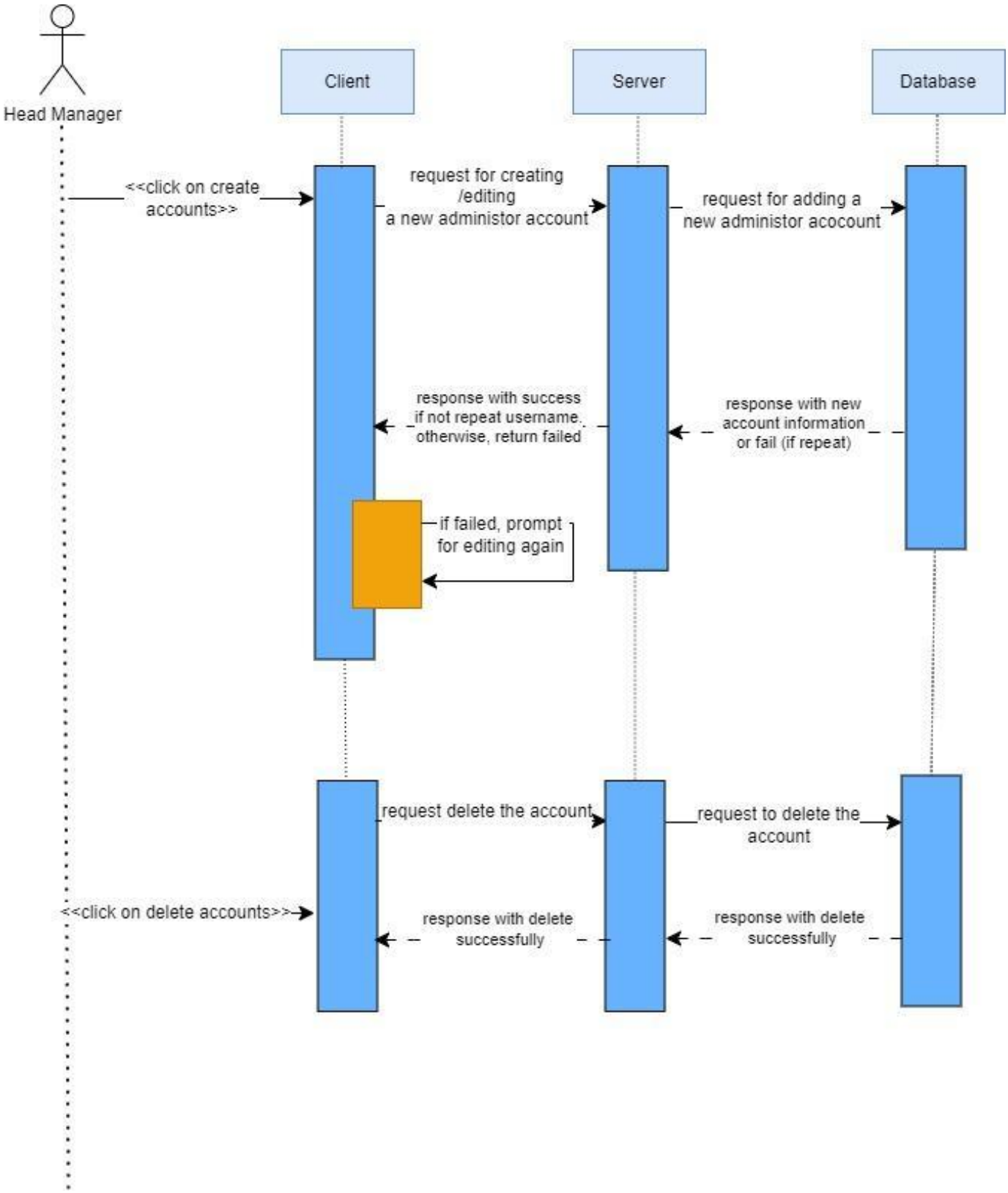
3. Search System



4. Adding/Modifying/Deleting Comments and Reporting a Product

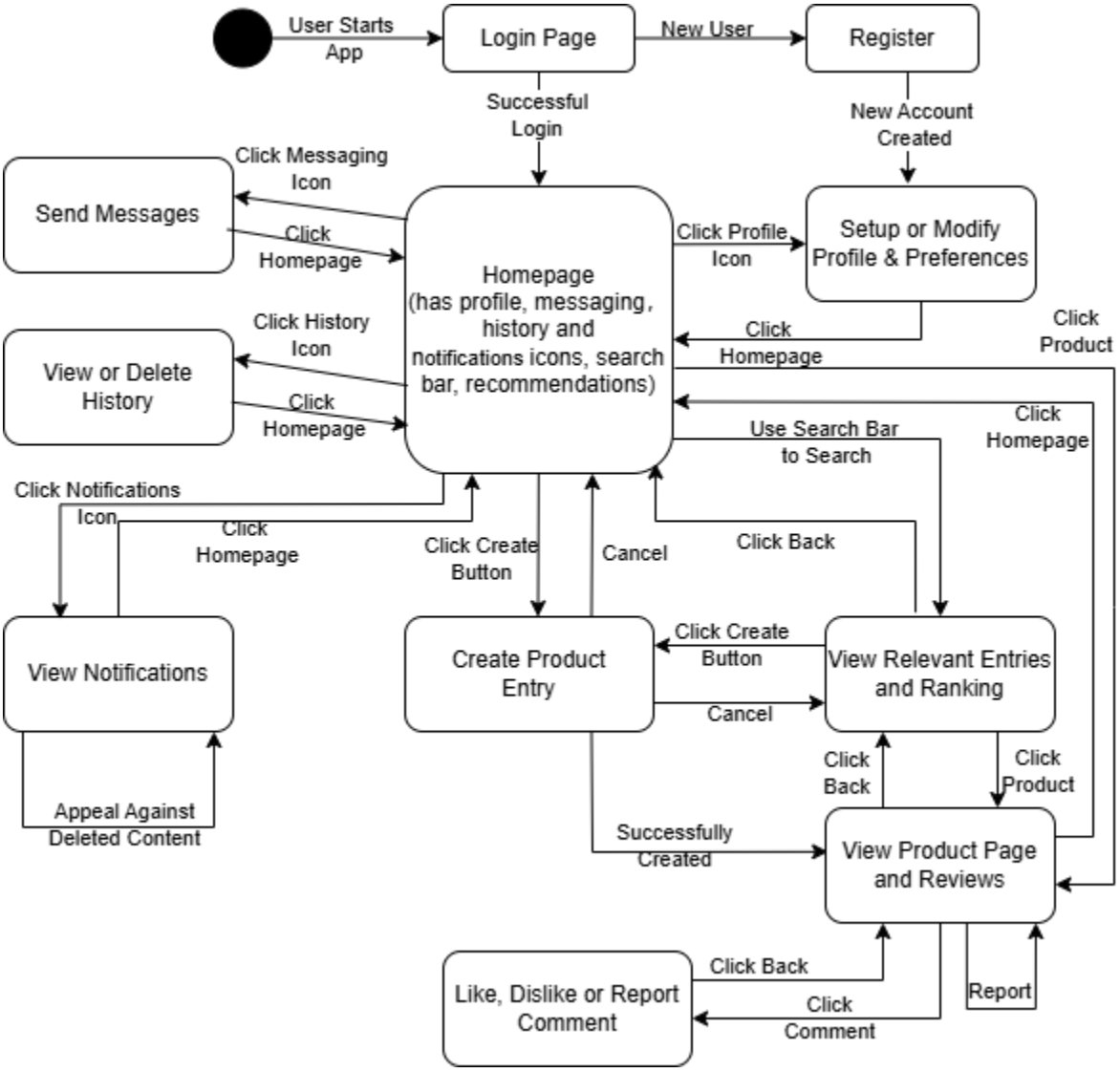


5. Admin Creating and Deleting User Accounts

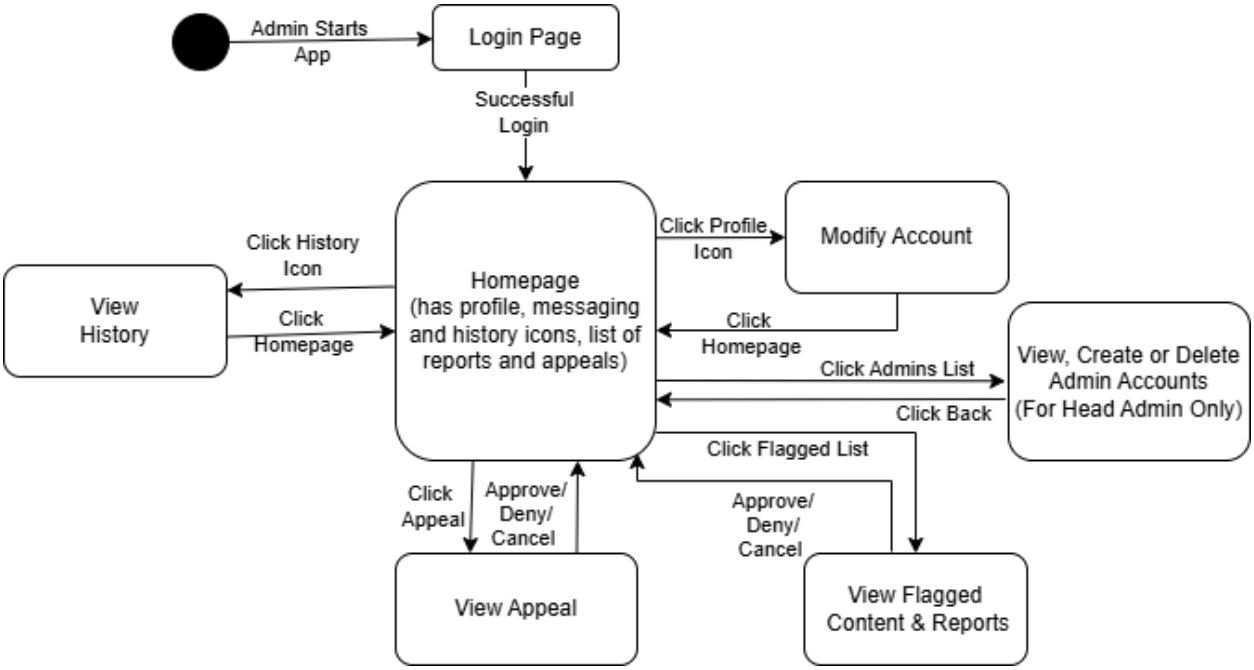


Navigation Flow Map

1. User Flow Map



2. Admin Flow Map

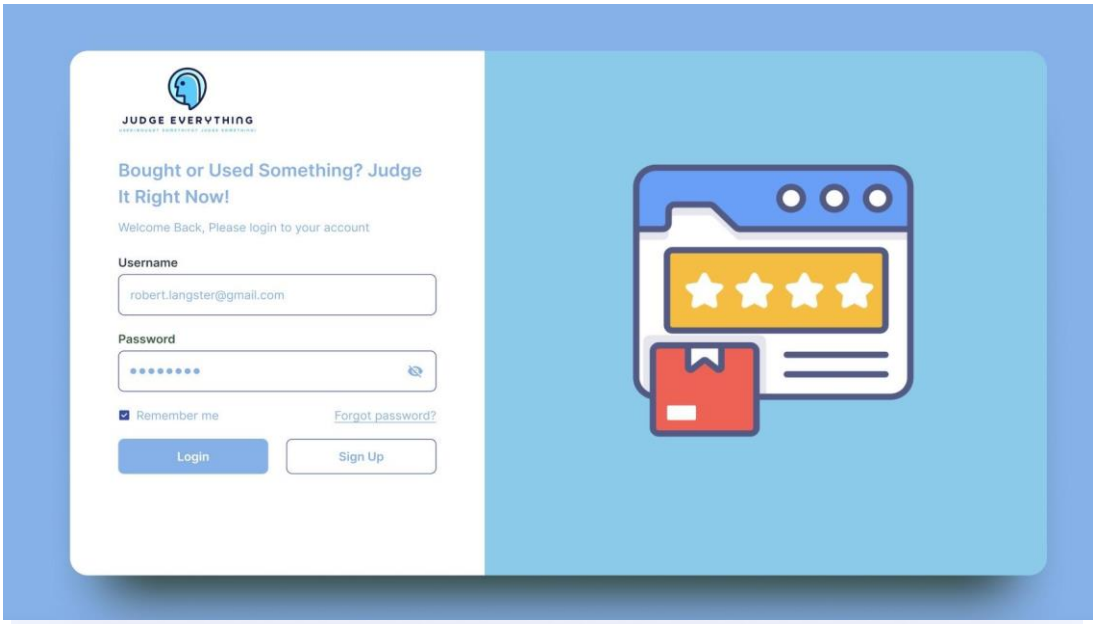


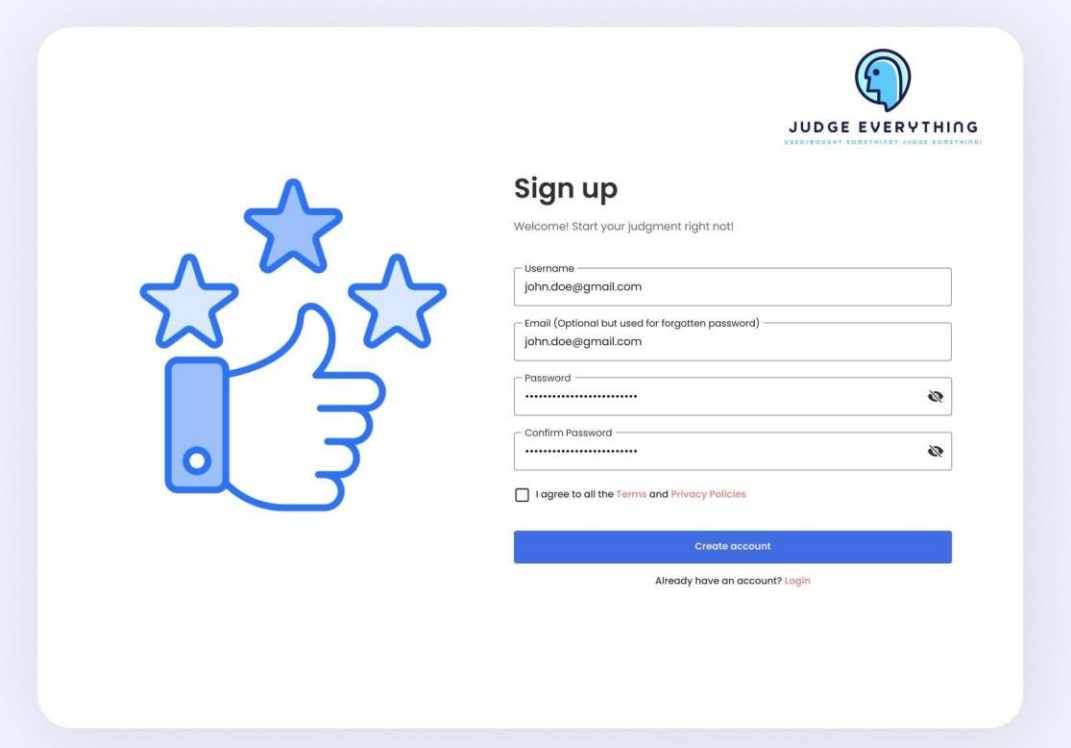
UI Mockup

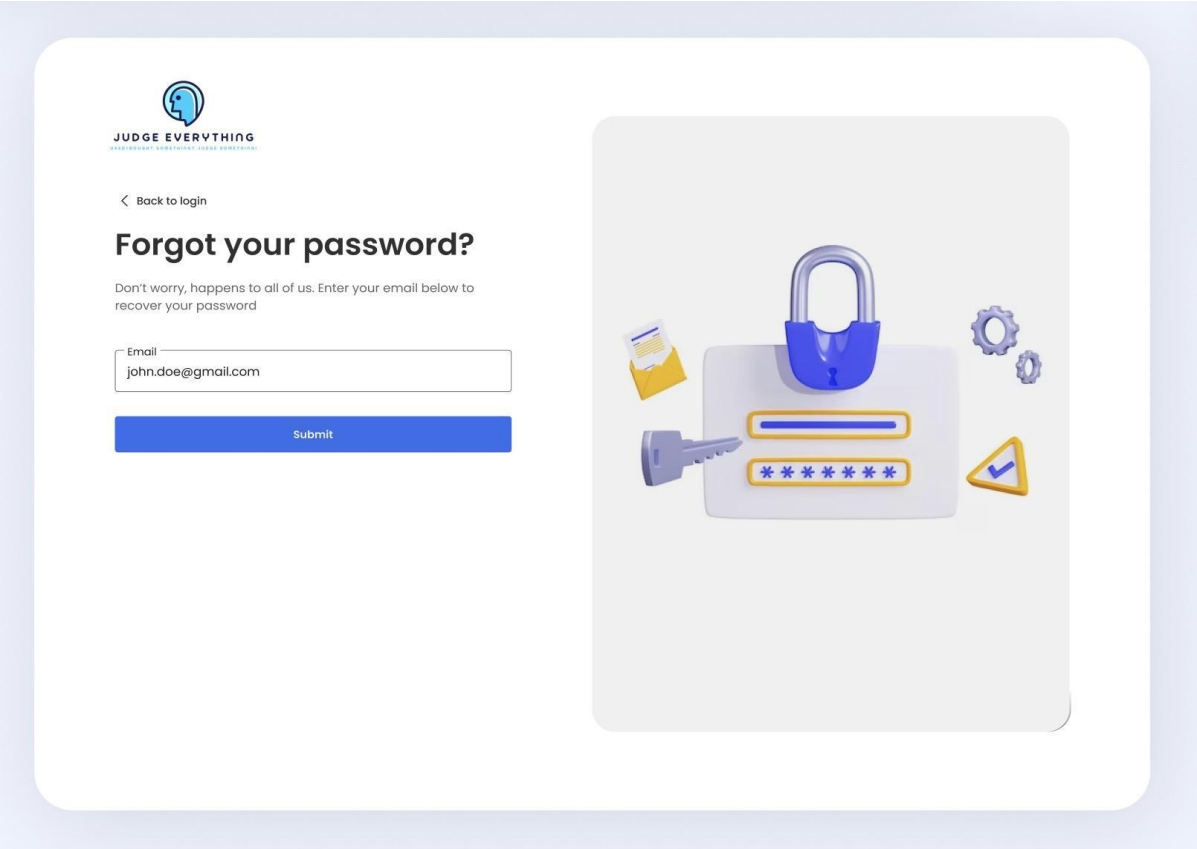
Login page and sign up page:

Users can log into an existing account using username and password, or create a new account using the sign-up page. Providing email is optional but strongly recommended, because it is used to reset password in case a user forgets their password. Users can add their email at any time.

User







Homepage:

Users can see the homepage after starting the web app, which displays the most popular entries of the week, updated every Tuesday, and personal recommendations based on the user's browsing history within the app. From the homepage, users can create entries, directly view product profiles or history, using corresponding buttons, or browse entries using search bar. Personal profile, messaging and notification icons are displayed after clicking the bar icon in top right, and only available after the user logs in. Users can modify their account information and preferences at any time using the profile icon.

Found in 2024

Judge Everything

Bought/Used something? Share it!

Create a New Entry

Ranking

Most Popular Entries This Week

will update every Thursday 11:59 p.m. EST

iPhone 16
Electronics
View

Dress
Clothes
View

Standing Cat
MEME
View

Stanley Cup
Water Bottle
View

Jacket
Clothes
View

LOAD MORE ENTRIES

Recommendations

The Products You May Like...

Change your preference in you account setting anytime!

iPhone 16
Electronics
View

Dress
Clothes
View

Standing Cat
MEME
View

Stanley Cup
Water Bottle
View

Jacket
Clothes
View

LOAD MORE ENTRIES