**High-level Design Document**

**Project Title: ChatU**

**Document Version: 1.0.0**

**Group ID: B4**

**Print date: 04/02/2023**

**Group Members:**

**DENG Haopeng 1155138409**

**Gu Zhehao 1155141578**

**LIU Zhixian 1155141612**

**TANG Yufei 1155141604**

**ZHANG Haoxiang 1155141702**

**Selected Application Choice: Twitter**

**CSCI3100**

**Department of Computer Science and Engineering**

**The Chinese University of Hong Kong**

**Table of Content**

[**1. INTRODUCTION**](#_dzr2azrjgru8) **2**

[1.1. Project Overview](#_f8qqqgearfix) 2

[1.2. System Feature](#_vraekui20wg) 2

[**2. SYSTEM ARCHITECTURE**](#_2swknetj94z) **5**

[2.1. Technologies](#_f829zgur838b) 5

# INTRODUCTION

## Project Overview

For this project, our objective is to create a simplified version of Twitter, a dominant microblogging and social networking platform, with the underlying blogging and interactive framework being that of Twitter but implemented in the way we know about the working mechanism and paradigms of social networking services.

## System Feature

Basic Application Requirements:

* Client-Server Architecture

The application should consist of two major components, namely the client and the server. For every feature of the application that a user invokes, the application should send the corresponding request from the client side to the server side, and the server, after processing the request, will perform the requested feature for the user or send the requested information to the client side.

* Global Database

The application should have access to a remote database storing user information, tweet contents, comments, number of likes for each tweet, and so on, so that the server will create, read, update, delete (C.U.R.D.) data from the database in accordance with the needs of the software.

* Graphic User Interface Design

The graphical user interface design of the application should be user friendly, meaning that it should be clear, informative, understandable, and consistent, so that anyone can use the application without the assistance of the developers.

* User Account

In our application, a user can use his/her email to sign up for a unique account with a username and password, and then create a user profile for it. Before a user logins to his/her account, our Twitter will only provide some random tweets for the logged-out user, and also a logged-out user cannot like or dislike a post or follow anyone. After logging into his/her account, all the above restrictions will be gone and our Twitter will display tweets of the followed accounts on the main page. The user can after all log out from his/her account as well, so as to prevent others from getting their hands on his/her account.

* Admin Account

The application should have an admin account that has an UI different from that of ordinary user accounts. This admin account allows the admin user to view all ordinary users’ information and tweets, to create, retrieve, update or delete an account or tweet, or to ban an account from publishing tweets, writing comments, and private messaging.

* User Operations

Similar to Twitter, the application should support the following user operations:

* + The operation to search for users

The application should allow any user to search for other registered users’ accounts using their usernames or their unique userIDs at a search bar.

* + The operation to follow other users

Our application should allow logged-in users to follow any other registered user that he/she is interested in. And for those who have followed others, they should find the accounts that he/she has followed in their “following” pages. And for those who are followed by others, they should find their followers in their follower page.

* + The operation to like or dislike a tweet

Each tweet in the application has a like button, a dislike button, and a like counter. The like counter records the number of likes the tweet receives minus the number of dislikes it receives (and will be set to zero if the result is negative). If for a tweet, a user clicks its like button, then the like counter should increase by one, and if he/she clicks its dislike button, then the like counter should decrease by one (but will not further decrease if it is already zero).

* + The operation to comment a tweet

Our Twitter should allow logged-in users to comment on a tweet. A comment will show the commentator's icon, commentator’s name and userID, the post date and the sub-comment to the comment. Also we will entitle the tweet editor the right to delete any comment.

* + The operation to post a tweet

Our Twitter should allow logged-in users to post their tweets, and also they can delete and comment on their own tweets.

* + The operation to retweet a tweet

The application should allow a user to perform an operation called “retweet”, which is to quote a selected tweet in a new tweet for which all the followers of this user can see.

* + The feature of showing tweets of accounts being followed on home page

Any user should see the tweets of accounts he/she follows on the home page.

Advanced Functionalities and Features:

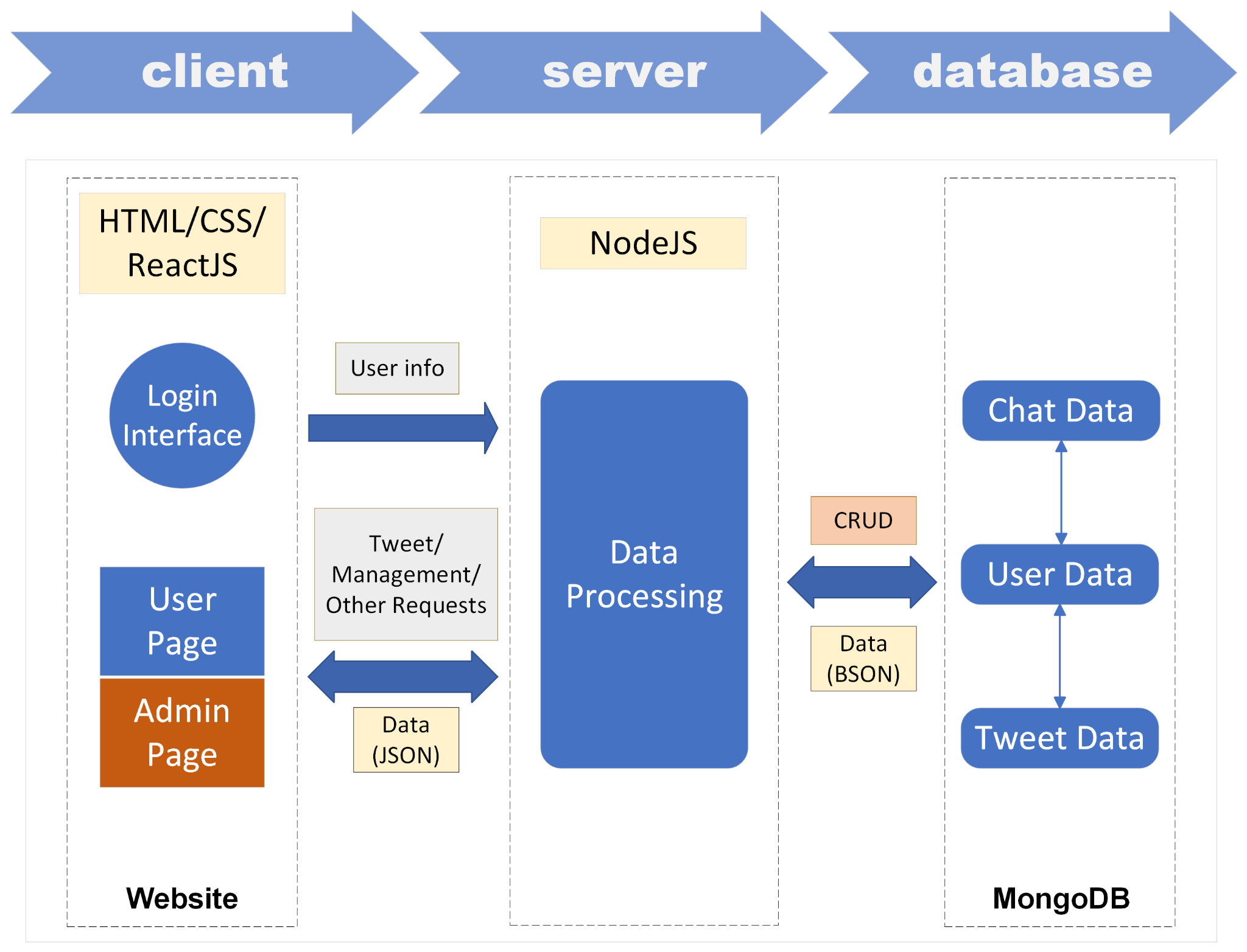
* Tags: Before a user posts a tweet, the application should require him/her to select a tag for this tweet from a given list of tags that indicate the category of contents that the tweet falls under, and once the user successfully publishes the tweet, the application should store the selected tag as one of the attributes of the tweet and associate it along with the tweet in the database. Moreover, immediately after a user registers an account, the application should prompt the user to select the tag(s) that he/she shows interest for.
* Recommendation of potential users to follow: On the personal information page of a user, the application should additionally generate several accounts that he/she does not follow at that time and display those accounts at the bottom of this page. The mechanism of this feature is to randomly select several accounts that the user does not follow but some of his/her following accounts follow, and this list of selected accounts will be displayed as the recommended accounts that the user is suggested to follow. In case if this list is empty or inadequate, it will be replenished with, for example, accounts that are not followed by the user and have the most followers in the application.
* Recommendation of potential tweets of one’s interest (a.k.a. feeds): On the main page of the application, it should interweave following accounts’ tweets with tweets from accounts that the user does not follow. The working mechanism can be, for example, choosing tweets that come with the user’s preferred tags and the highest number of likes.
* Private chat: The application may allow a user to initiate an online chat with another user, during which private messages can be sent between the two users.
* Favorite tweet: Each user is allocated with storage space named “favorite”, so that once the user finds a tweet that he/she deems special and would like to bookmark, he/she can add this tweet to this storage space, after which the added tweet can always be found in this storage with being lost in the feeds.
* Privacy and visibility control: Before publishing a tweet, the application allows the user to choose the followers to which the tweet will be made invisible.
* Banning users: The application should allow a user to ban another user, after which the tweets of the banned user will no longer be delivered to the main page of the first user and the banned user can no longer send private messages to the first user.
* Trend: The application maintains an hourly updated tweet popularity rank which shows the ten tweets that have the highest like counts.
* Sending feedback from application: The application provides an interface whereby the user can send feedback, deliver suggestions, or report bugs to the developers in the form of an email.
* Pretty user interface: A more aesthetically appealing user interface can be incorporated into the application while keeping the UI to be clear and informative.
* Video tweets: The application can allow users to add video contents to a tweet that he/she will publish.

# SYSTEM ARCHITECTURE

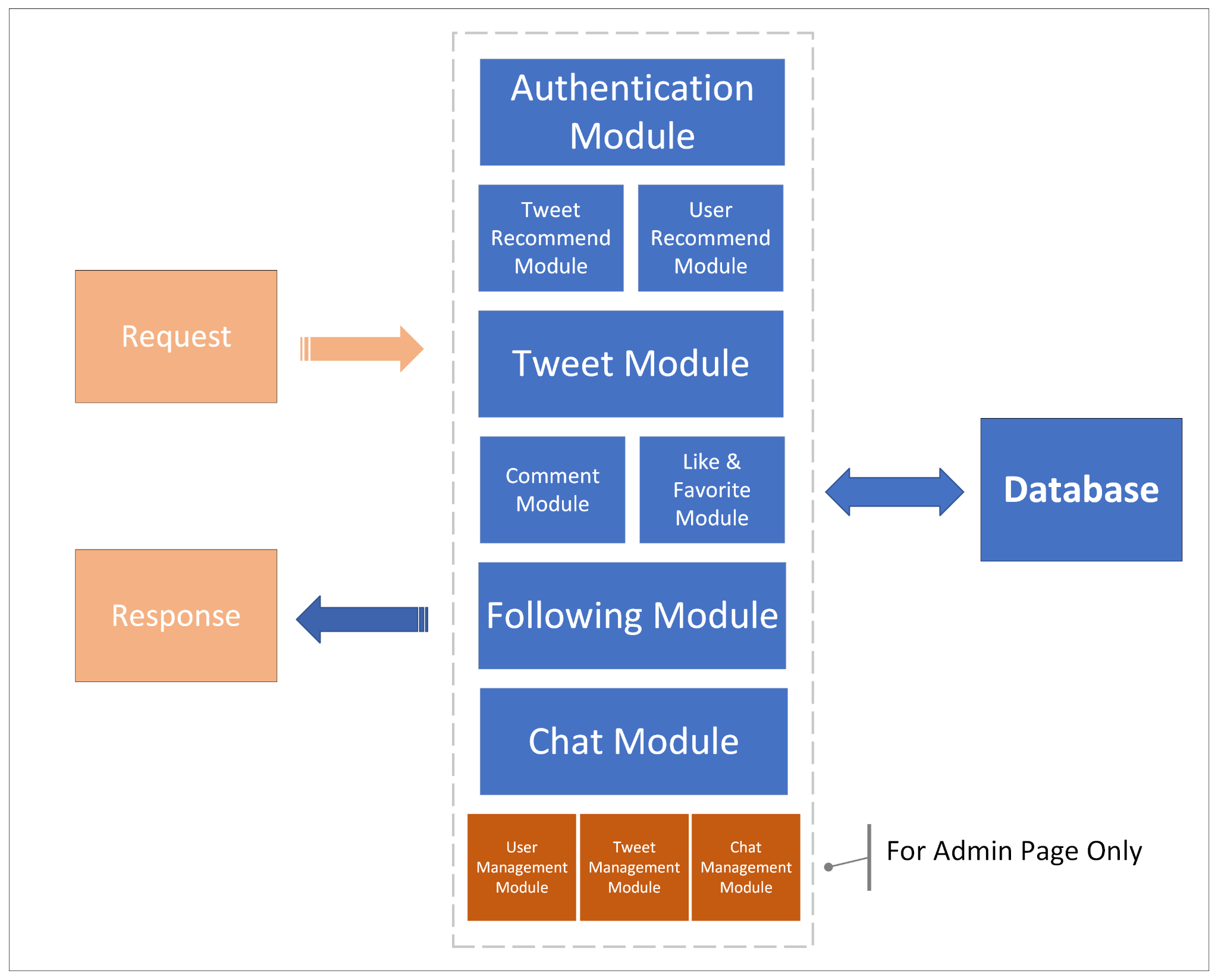
## Technologies

We will use Bootstrap, ReactJS for the front-end, NodeJSs for the back-end development. The database system we will use is Mongodb. We will mainly use css to design and generate the user interface with the help of Bootstrap framework and Figma. The programming languages that our software will be written in are HTML and JavaScript.

Architecture Diagram and System Components



As can be seen in the architecture diagram above, our Twitter application uses the client-server-database structure. The client part is essentially the website on which our twitter application is hosted, and it is composed of the login interface, the pages that an ordinary user can access, and the pages that only the admin can access. The codes for these parts will be written in a combination of HTML, CSS, and JavaScript and ReactJS. The server part is essentially an abstraction of the many modules responsible for processing data received from the client side and database side, where in the diagram above is depicted as a whole system called the data processing system, and the code of which will be written in NodeJS. The database part we will use MongoDB, and design three data schemas, namely the Chat, User, and Tweets, respectively. After a user inputs the username and password of his/her own account and clicks log-in on the website of our Twitter application, the client will send the inputted username and password to the server, then the data processing system will check whether the inputted username/password match the correct one stored in the database. After successful login, the user will be navigated to the admin page if the account is an admin account or to the user page of ordinary users if it is not the case. Then any operation being requested there, including publishing new tweets or managing tweets and user accounts (in the case of an admin account), the corresponding request will be sent from the client side to the server side through the internet, and the response will be sent back with corresponding data in JSON form. The data processing system has the right to Create, Read, Update and Delete (CRUD) the data (in BSON form) stored in the database, and use them to generate the data to be sent to the client.



The diagram above gives the inner details of the data processing system, which will be a highly modularized system. We have separated admin users from ordinary users, so that in the admin page we can access the User, Tweet and Chat management modules that contain the functions to manage basically everything of our Twitter. The Authentication module makes the log-in process of any logged-out user possible, and a logged-out user’s client is not permitted to send requests to Tweet/User Recommend modules as well as the Comment, Like & Favorite, Following, and Chat modules. All the modules in the diagram above function properly by sending requests to the database and subsequently obtaining from it the information required by the operations they support.