Document Version 1.01

High-Level Design Document

Group ID: A1

Group Members:
LAM Man Ho (1155159171)
MA Yu Ying (1155157879)
CHEUNG Ka Ho (1155158622)
FUNG Ngai Man (1155158312)
LI, Eric John (1155159116)

Department of Computer Science and Engineering (The Chinese University of Hong Kong)

Printing Date: 17/02/2023

Table of Contents

1. INTRODUCTION *2*

1.1 Project Overview **2**

1.2 System Features **2**

1.2.1 User Interface (UI) 2

1.2.1.1 User Management 2

1.2.1.2 User Operations 3

1.2.1.3 Administrator Management 3

1.2.1.4 Recommendations 3

2. SYSTEM ARCHITECTURE *4*

2.1 Technologies **4**

2.2 Architecture Diagram **4**

2.3 System Components **4**

1. INTRODUCTION

1.1 Project Overview

The objective of this project is to build a web application to provide microblogging and social networking services. With this web application, users can post messages, images, and videos (which are known as “buzz”) to have real time interaction with one another. People accessing to the application can read public buzzes, but most operations are forbidden until they register an account. This high-level design document:

- Describes the system features.
- Lists the technologies used.
- Presents the system architecture with diagram.

1.2 System Features

This section describes the system features such as the design features and functionalities of the application.

1.2.1 User Interface (UI)

The user interface should have a clear graphical UI design so that the user can find it easy to understand the meaning of the symbols of different icons/symbols, which make them find the application user-friendly. Standard UI designs such as using a cogwheel icon to represent the “setting” function, representing the “Like” function with a heart or thumbs up icon, etc. are examples of clear graphical UI design and similar conventions can help to make a user-friendly user interface. In addition, the UI should adopt responsive web design to enhance usability and include some animations such as 3D fade out effect on commonly used function buttons to ensure satisfactory.

The user interface should not only have the features previously mentioned, but also include some useful functionalities which helps the user to post buzzes and manage their account.

1.2.1.1 User Management

The application should include basic user management functionalities so that users can manage their own account.
Basic management functionalities include letting users to

- Sign up
- Login and logout
- Reset password
- Modify their own information in their profile
- Choose their favourite categories (Affect the buzzes and users' recommendations)

Other than the basic functionalities, the application should also include advanced functionalities to improve user experience.

Advance management functionalities should allow users to

- Set privacy control on the visibility of their buzzes (users can set who are allowed to see their buzzes)

- Manage personal settings (if application enables customized display settings)

1.2.1.2 User Operations

Users should be allowed to operate application to post contents that they want to share.
User operations allow the user to

1. Search for other users (based on usernames or unique userIDs)
2. Follow other users
   - new posts of followed users should be pushed to the user
3. Like/Dislike a buzz
   - a user can increase/decrease the buzz's associated like counter by the like/disklike button
4. Comment a buzz
   - a user can leave comment with message
   - a user can reply to others' comments
   - a user can delete his/her own comment
5. Rebuzz a buzz
   - a user can share someone's buzz
6. Post a buzz
   - a user can include text, images, and videos in their buzz
   - a user can include hashtags regarding the content of the buzz
7. Show other users' buzzes
   - for a user, the application should show the buzzes posted by users he/she followed
8. Start a private chat with other users
9. Mention other users with a @ symbol
   - a user mention others by @<username>

1.2.1.3 Administrator Management

The application should allow administrators to manage the operating of the application with their Buzzer administrator accounts have more operations other than general user.
The Buzzer administrator accounts are simply Buzzer accounts with the administrator role, the administrators are responsible for ensuring the posts in the community follows the written regulations of Buzzer, thus the administrators have the following additional operations:

- Delete a user
- Delete a buzz
- Verify a user
  - The administrator can add a blue checkmark next to a user's username
  - The blue checkmark helps to distinguish celebrities/organizations from phonies

1.2.1.4 Recommendations

To elevate user experience, the application should make recommendation to users according to their interest.

- User recommendation
  - Recommend users of interest for a user to follow
  - Recommendation is made based on the category a user is interested in
- Buzz recommendation
  - Recommend buzzes of interest for a user to read
  - Buzzes are recommended according to user favours

2. SYSTEM ARCHITECTURE

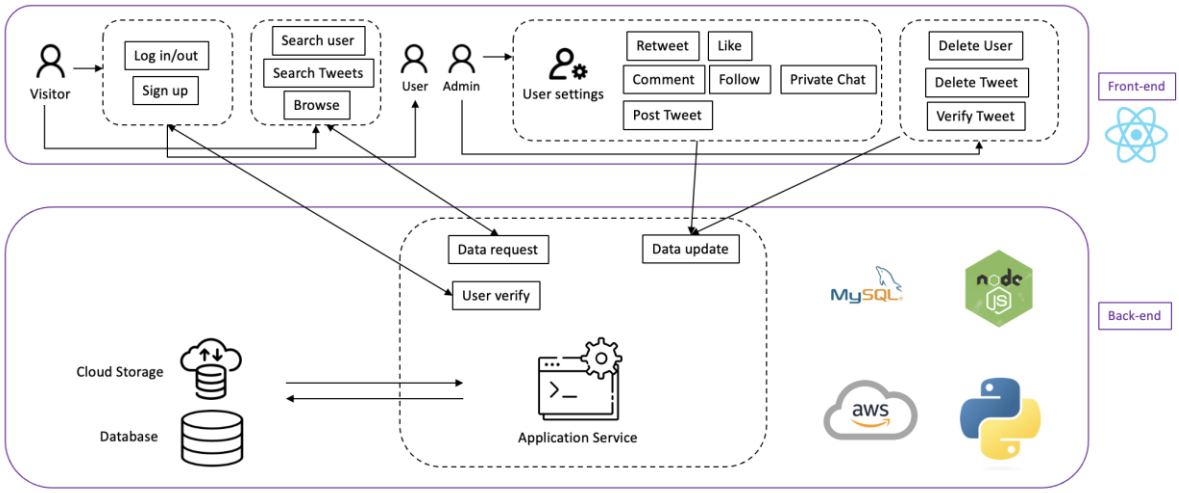
2.1 Technologies

This section states the technologies for developing the application:

- Front-end: HTML, CSS, and JavaScript
- Front-end framework: React.js
- Back-end: Node.js, Python
- Database: MySQL

2.2 Architecture Diagram

The web application follows a client-server architecture, and the diagram below illustrates the interaction between components in the architecture:



Details regarding how the front-end, back-end, storage system in the architecture build the application and accomplish the functionalities and features will be discussed in the next section.

2.3 System Components

- Front-end
  The front-end of the application provides web service through the user interface (UI). It is written in HTML, CSS, and JavaScript, which is built by the framework React.js. It collects users' inputs from the UI and sends the requests to the backend server through fetch. After receiving the data in JSON format, it will render the data and then present it on the UI. Furthermore, React Native will be used to handle the responsive rendering to consider the users' potential desire to access the application through mobile devices.

- Back-end
  The back-end of the application will be written in Python and Node.js, which are built by Django and Express respectively. It provides services including verification in user login and registration, user subscription and user searching. To make the services realistic, it will handle the requests from the front-end, communicate with the database, package the data in form of JSON, and then

respond to the front-end. Furthermore, it also handles the features like real-time notifications and private chatting.

- Database & Cloud Storage
  The database will be used for dealing with data storage. In the application, we will use a relational database, MySQL to manage the data. Considering the limitations of the database, we would use the AWS (Amazon Web Services) cloud storage services to store the visual media, specifically images and videos. The following figure shows the relationship of data in our database.