**VIETNAMESE-GERMAN UNIVERSITY**



**MULTI-DISCIPLINARY PROJECT REPORT**

**TRACK ME RUN**

Major: Computer Science

**Instructor : TRUONG TUAN ANH, PhD**

**Student 1 : Đào Gia Hưng (10423048)**

**Student 2 : Lê Mai Bảo Linh (10423072)**

**Student 3 : Lê Quang Minh Khoa (10423057)**

**Student 4 : Nguyễn Đức Khang (10423054)**

**Student 5 : Võ Khánh Trân (10423177)**

**---o0o---**

HO CHI MINH CITY, 07/2024

**ABSTRACT**

Contents

[CHAPTER 1](#_heading=h.gjdgxs)  INTRODUCTION 1

[I.](#_heading=h.30j0zll) Context 1

[II.](#_heading=h.3znysh7) System description 2

[CHAPTER 2](#_heading=h.2et92p0) SYSTEM ANALYSIS 3

[I.](#_heading=h.tyjcwt) System stakeholders 3

[II.](#_heading=h.3dy6vkm) System requirements 3

[1.](#_heading=h.1t3h5sf) Functional requirements 3

[2.](#_heading=h.4d34og8) Non-functional requirements 3

[III.](#_heading=h.2s8eyo1) Use case Diagram 4

[IV.](#_heading=h.3rdcrjn) Use case Description 5

[V.](#_heading=h.1ksv4uv) Activity Diagram 7

[VI.](#_heading=h.3j2qqm3) Class Diagram 10

[VII.](#_heading=h.4i7ojhp) Sequence Diagram 12

[CHAPTER 3](#_heading=h.2bn6wsx) SYSTEM DESIGN 16

[I.](#_heading=h.qsh70q) Architectural design 16

[II.](#_heading=h.3as4poj) Database design 16

[III.](#_heading=h.1pxezwc) UI design 16

[CHAPTER 4](#_heading=h.49x2ik5) SYSTEM IMPLEMENTATION 17

[CHAPTER 5](#_heading=h.2p2csry) SYSTEM TESTING 18

[CHAPTER 6](#_heading=h.147n2zr) SYSTEM DEPLOYMENT 19

[CHAPTER 7](#_heading=h.3o7alnk) CONCLUSIONS 20

[REFERENCES 21](#_heading=h.23ckvvd)

**LIST OF TABLES**

[Table.1: Use case Description - Sign in 5](#_heading=h.26in1rg)

[Table.2: Use case Description - Log out 6](#_heading=h.lnxbz9)

[Table.3: Use case Description - Reset Password 7](#_heading=h.35nkun2)

**LIST OF FIGURES**

[Figure 1: Use case Diagram 4](#_heading=h.17dp8vu)

[Figure.2: Activity Diagram - Sign in 8](#_heading=h.44sinio)

[Figure.3: Activity Diagram - Log out 9](#_heading=h.2jxsxqh)

[Figure.4: Activity Diagram - Reset Password 10](#_heading=h.z337ya)

[Figure.5: Class Diagram 11](#_heading=h.1y810tw)

[Figure 6: Sequence Diagram - Sign in 12](#_heading=h.2xcytpi)

[Figure 7: Sequence Diagram - Log ou 14](#_heading=h.1ci93xb)

[Figure 8: Sequence Diagram - Reset Password 15](#_heading=h.3whwml4)

# 

# CHAPTER 1 INTRODUCTION

# Context

Due to higher demands in personal fitness training nowadays, many fitness enthusiasts and service providers are constantly looking for ways to enhance physical well-being and make exercising products more accessible. In recent years, a particular type of workout has emerged as one of the most preferable means: running. In fact, according to World Economic Forum, running is the second most popular form of weekly exercise, chosen by 19 percent of 21,000 surveyed adults across 29 markets. Evidently, more adults are paying attention to running and jogging as an effective health maintenance method.

In order to improve running workout quality and motivate runners to keep on their practice, there is a need for runners to interact with their workout statistics in a meaningful way. One practical and easy to implement tracking data is calorie consumption. The number of calories burnt is the primary point of focus for many running participants. By tracking it, runners are able to proactively build a reasonable diet to ensure a balance between calorie intake and consumption, thereby maintaining their physical fitness.

There are numerous approaches when it comes to collecting calorie consumption data among runners. For seasoned and dedicated runners, they usually opt for a wearable fitness tracker, such as an Apple Watch or Google Fitbit. These gadgets offer a significantly high level of precision and are well-trusted by many. However, since there is a considerable barrier of entry to these products for newbies, they typically utilize the tracking sensors available on their smartphones. Therefore, developing accessible and cost-saving mobile software to meet this need is a solution in great demand. For this project, the team has decided to utilize the sensor capabilities of a smartphone and IoT platform support of Arduino Cloud to develop a mockup program to track users’ positions during a running session and calculate calorie consumption from that data.

# System description

The aim of our website is to manage and keep track of the information of the users’ running sessions. Such information as calories burnt, total distance, total time, the maximum and average speed is tracked, stored in our database, and accessible by users. This system can help users with running training and their physical health building journey. After each of the users’ sessions, they can check the statistics of their performance, which allows for necessary changes in running methods or in future plans.

# CHAPTER 2 SYSTEM ANALYSIS

# System stakeholders

* Standard users: register an account and get access to the data of such account (the data collected from the GPS device that the GPS holder holds).
* GPS Holders: whose training sessions are tracked and the statistics of this person are stored in the system (this person can also be the one whose role is the standard user).
* Admin: responsible for managing the system and database and maintaining and upgrading the website.

# System requirements

## Functional requirements

The system provides the following functional requirements:

* As an admin, he/she can access to the database and user profiles
* As a registered user, he/she can create personal accounts to have their own data saved and view the calories burnt, max speed, average speed, total distance, and total time he/she has run in a particular session between the start and the end of the programme.

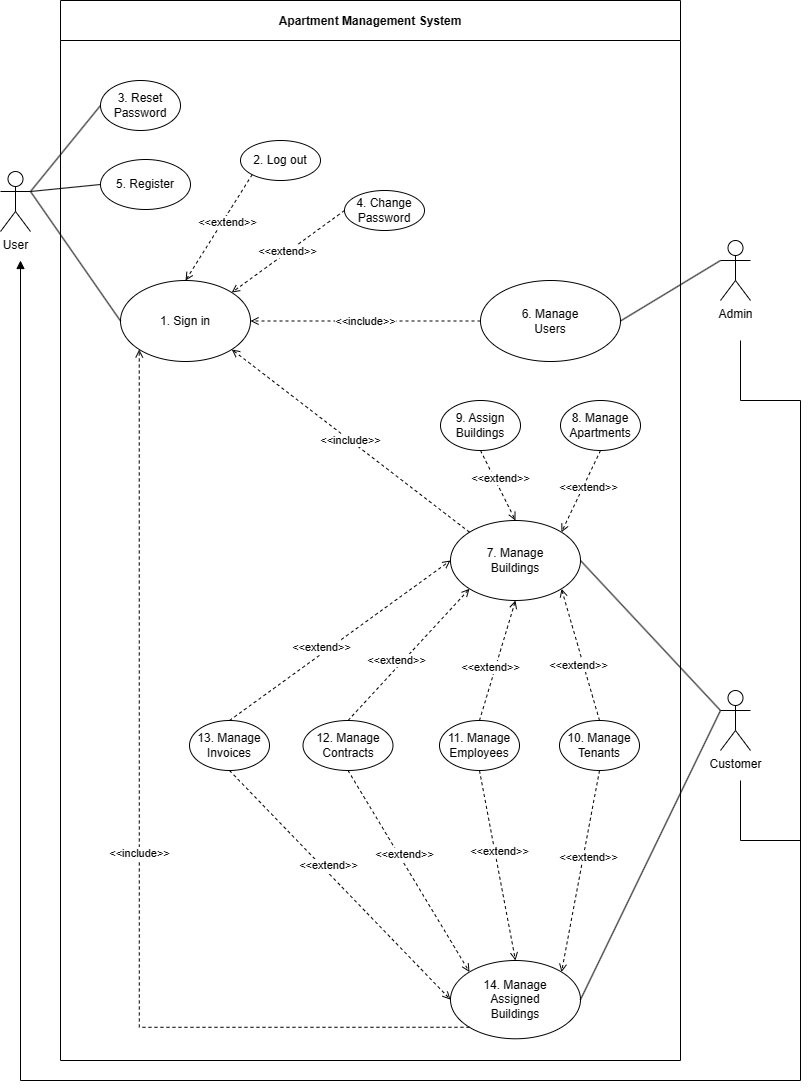
## Non-functional requirements

The system provides the following non-functional requirements:

* The user data is secured by being encrypted by the system with every movement executed.
* English is the main language used on the web application.
* User-friendly, users should be able to view the information within 2 clicks.
* The response time is less than 2 seconds.
* The system ensures 24/7 operation. Downtime shall not exceed 2 hours per year.

# Use case Diagram

The use case diagram of the system is depicted in Figure 1.



*Figure 1: Use case Diagram*

# Use case Description

* **Use case 1. Sign in**

| **Use case: Sign in** | | |
| --- | --- | --- |
| Mục đích: | Để đăng nhập vào Website | |
| Mô tả: | Thực hiện đăng nhập bằng Email và Password | |
| Tác nhân: | Người dùng và công ty | |
| Điều kiện trước: | Đang ở trong giao diện đăng nhập, nhập đầy đủ Email và mật khẩu | |
| Điều kiện sau: | Email và Password khớp với cơ sở dữ liệu, hệ thống cho phép truy cập vào hệ thống | |
| Luồng sự kiện chính (Basic flows) | Người dung | Hệ thống |
| 1. Nhập email, mật khẩu vào Form và chọn nút đăng nhập. | 2. Gửi dữ liệu đăng nhập về Server để kiểm tra tính hợp lệ dữ liệu.  3. Cho phép truy cập vào Website. |
| Luồng sự kiện phụ (Alternative Flows): | 3.1 Hệ thống hiển thị thông báo “Đăng nhập thất bại, sai Email hoặc mật khẩu, vui lòng nhập lại” | |

*Table.1: Use case Description - Sign in*

* **Use case 2. Log out**

| **Use case: Log out** | | |
| --- | --- | --- |
| Mục đích: | Để đăng xuất khỏi hệ thống | |
| Mô tả: | Thực hiện đăng xuất khỏi hệ thống | |
| Tác nhân: | Người dùng và công ty | |
| Điều kiện trước: | Đang ở trạng thái đã đăng nhập | |
| Điều kiện sau: | Người dùng gửi yêu cầu thoát bằng cách bấm nút đăng xuất trên giao diện | |
| Luồng sự kiện chính (Basic flows) | Người dung | Hệ thống |
| 1. Bấm nút đăng xuất trên Giao diện. | 2. Gửi yêu cầu đăng xuất khỏi hệ thống đến Server.  3. Đăng xuất khỏi hệ thống. |
| Luồng sự kiện phụ (Alternative Flows): | 3.1 Hệ thống Cập nhật giao diện chưa đăng nhập | |

*Table.2: Use case Description - Log out*

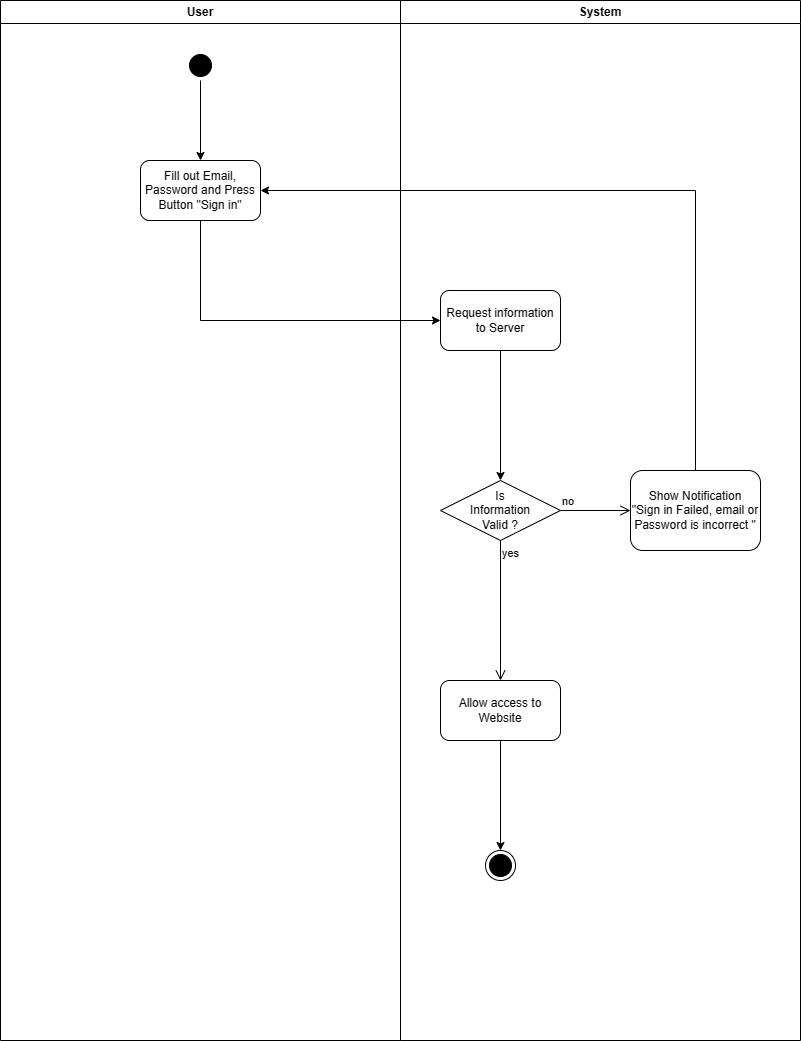
* **Use case 3. Reset Password**

| **Use case: Reset Password** | | |
| --- | --- | --- |
| Mục đích: | Để thay đổi mật khẩu khi quên mật khẩu | |
| Mô tả: | Thực hiện mật khẩu thay đổi mật khẩu khi quên mật khẩu | |
| Tác nhân: | Người dùng và công ty | |
| Điều kiện trước: | Đang ở trong giao diện Quên mật khẩu, nhập Email để lấy mã xác thực | |
| Điều kiện sau: | Nhập mã xác thực chính xác và nhập mật khẩu mới hợp lệ , hệ thống cho phép đổi mật khẩu mới | |
| Luồng sự kiện chính (Basic flows) | Người dung | Hệ thống |
| 1. Nhập email vào Form và chọn nút lấy mã xác thực.  3. Nhập mã xác thực và mật khẩu mới. | 2. Gửi dữ liệu về Server để kiểm tra tính hợp lệ dữ liệu.  4. Gửi thông tin mật khẩu mới về Server để kiểm tra.  5. Hệ thống thông báo “Đổi mật khẩu thành công” và chuyển về trang chủ. |
| Luồng sự kiện phụ (Alternative Flows): | 1.1. Hệ thống hiển thị thông báo “Email không tồn tại”  3.1. Hệ thống hiển thị thông báo “Đổi mật khẩu thất bại, vui lòng nhập lại”. | |

*Table.3: Use case Description - Reset Password*

# Activity Diagram

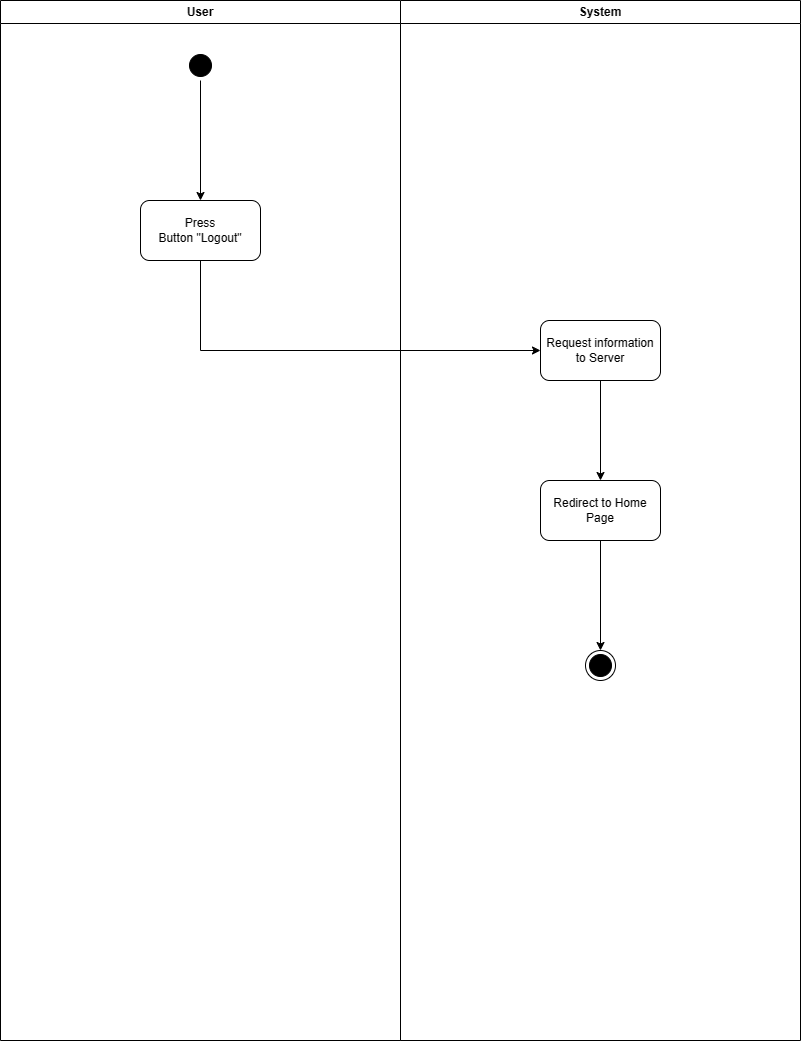
* **Activity 1. Sign in**



*Figure.2: Activity Diagram - Sign in*

**Description:** Short description for this activity diagram…

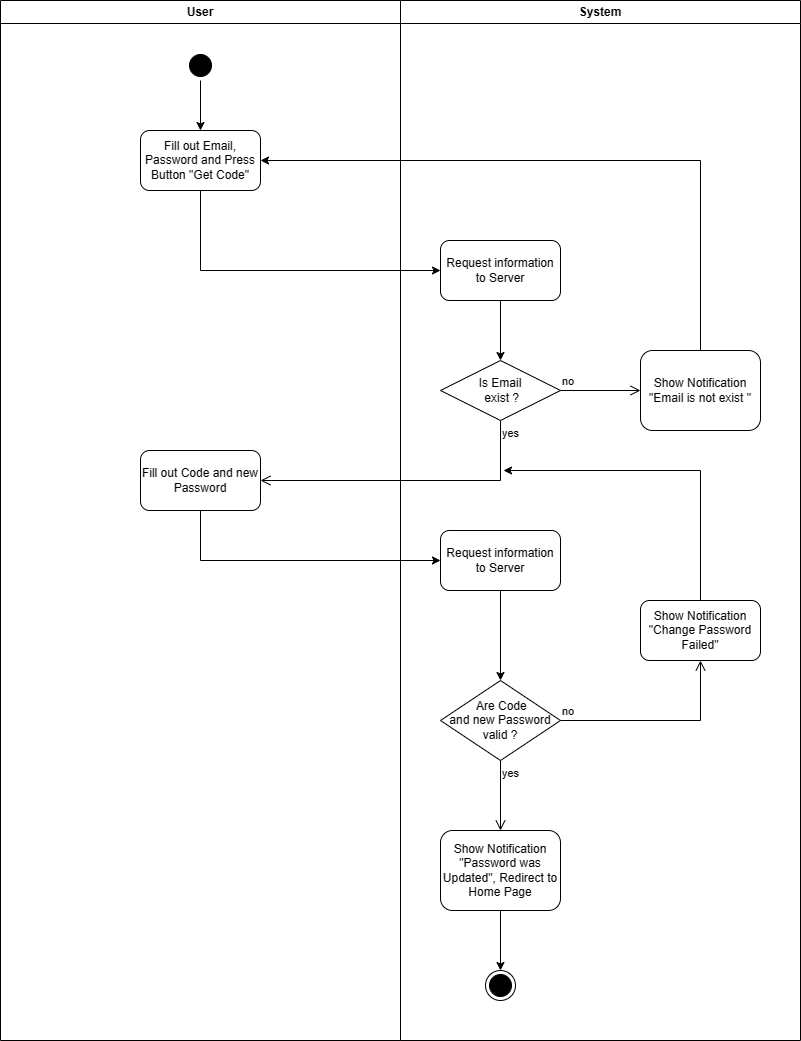
* **Activity 2. Log out**



*Figure.3: Activity Diagram - Log out*

**Description:** Short description for this activity diagram…

* **Activity 3. Reset Password**

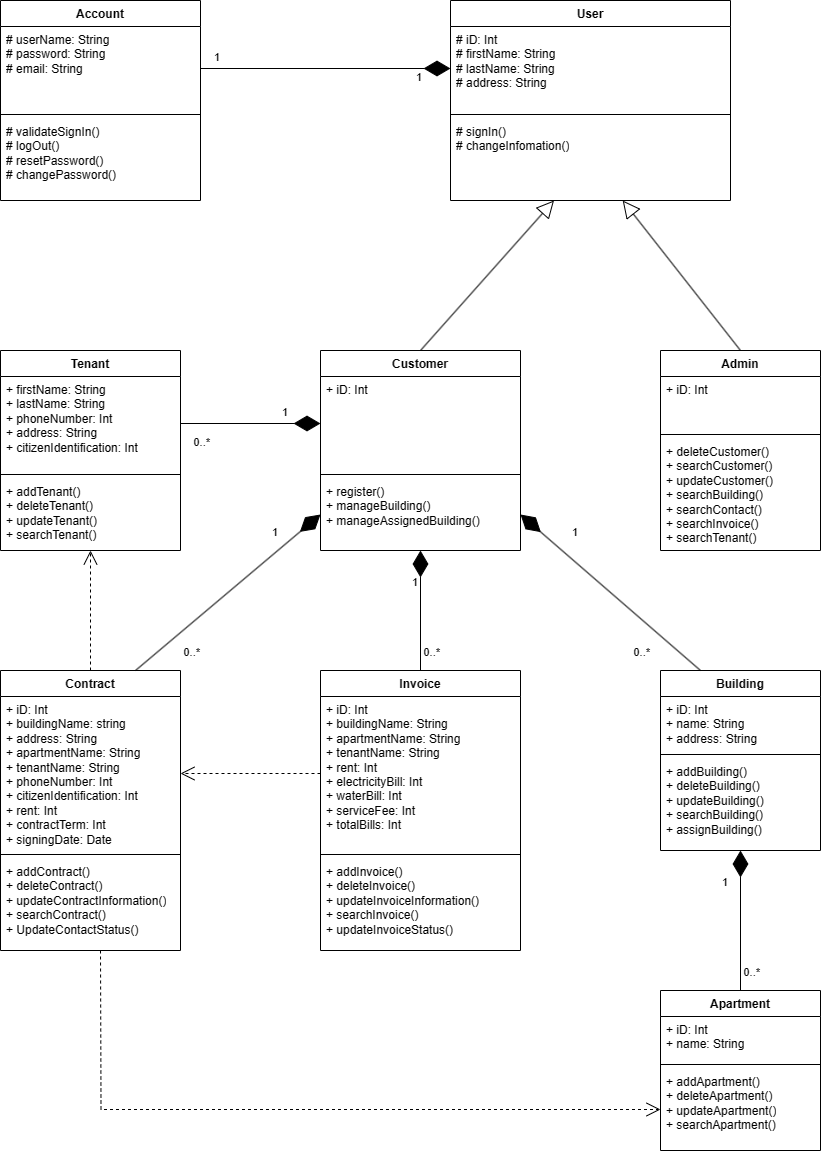


*Figure.4: Activity Diagram - Reset Password*

**Description:** Short description for this activity diagram…

# Class Diagram

The class diagram for the system is as follows:

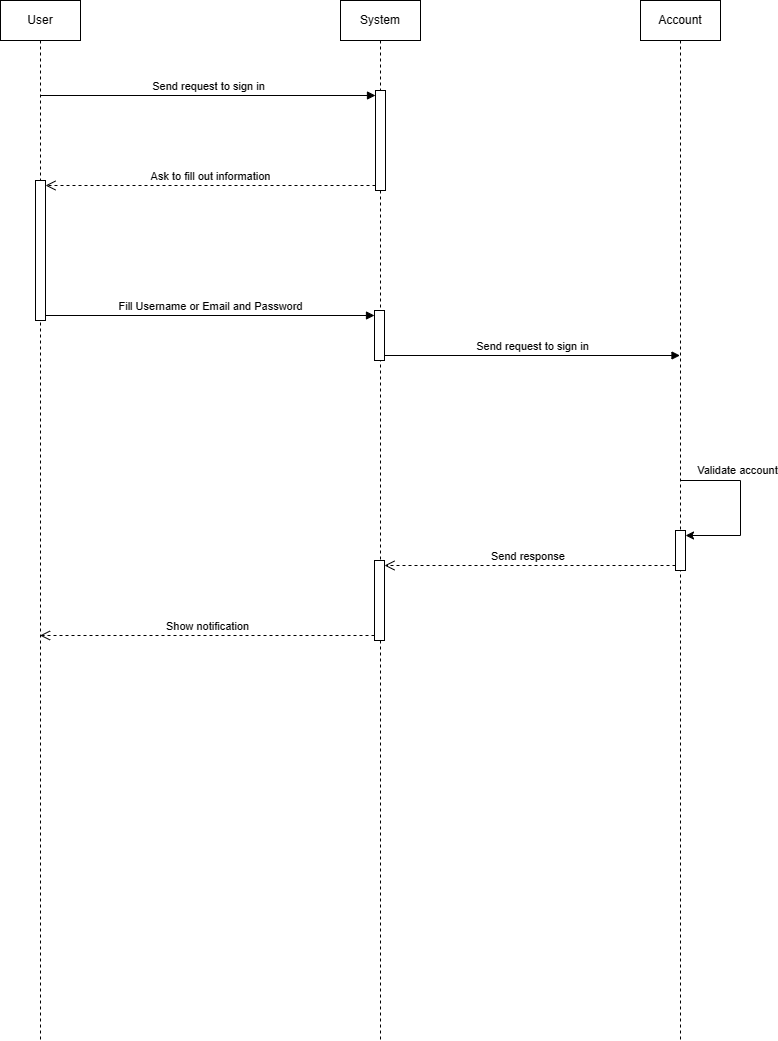


*Figure.5: Class Diagram*

**Description:** Short description for this class diagram…

# Sequence Diagram

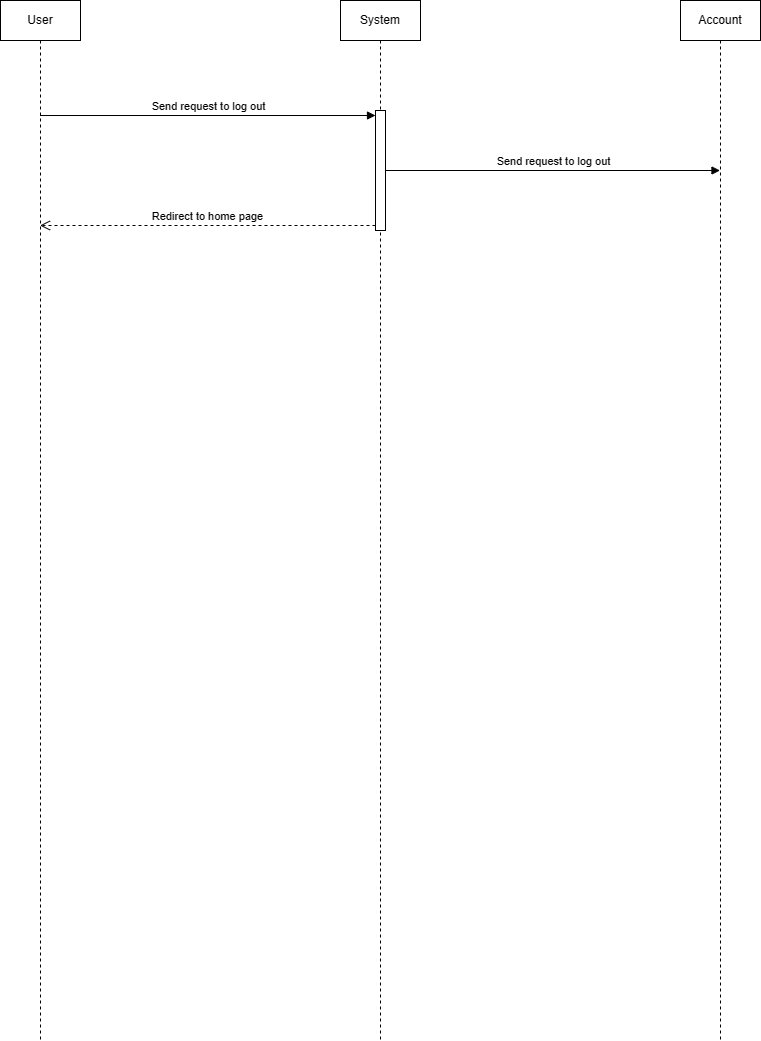
* **Sequence 1. Sign in**



*Figure 6: Sequence Diagram - Sign in*

**Description:** Short description for this sequence diagram…

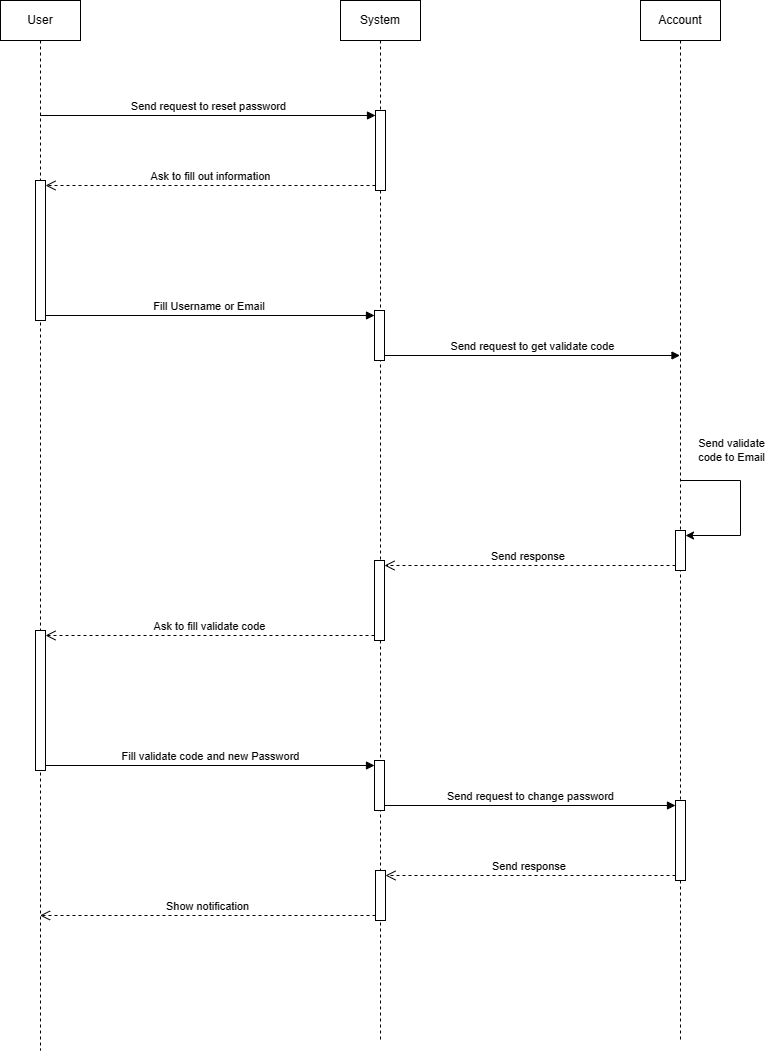
* **Sequence 2. Log out**



*Figure 7: Sequence Diagram - Log ou*

**Description:** Short description for this sequence diagram…

* **Sequence 3. Reset Password**



*Figure 8: Sequence Diagram - Reset Password*

**Description:** Short description for this sequence diagram…

# CHAPTER 3 SYSTEM DESIGN

# Architectural design

…….

…….

# Database design

…….

…….

# UI design

……

……

# CHAPTER 4 SYSTEM IMPLEMENTATION

# CHAPTER 5 SYSTEM TESTING

# CHAPTER 6 SYSTEM DEPLOYMENT

# CHAPTER 7 CONCLUSIONS

# REFERENCES

[1] React. “LEARN REACT”. 2023, từ: https://react.dev/learn/

[2] Redux. “API Reference”. 2023, từ: https://redux.js.org/api/api-reference/

[3] Redux-Saga.“API Reference”. 2023, từ: https://redux-saga.js.org/docs/api/

[4] PostgreSQL. “Documentation”. 2023, từ: https://postgresql.org/docs/

[5] JSON Web Token. “Introduction”. 2023, từ: https://jwt.io/introduction/

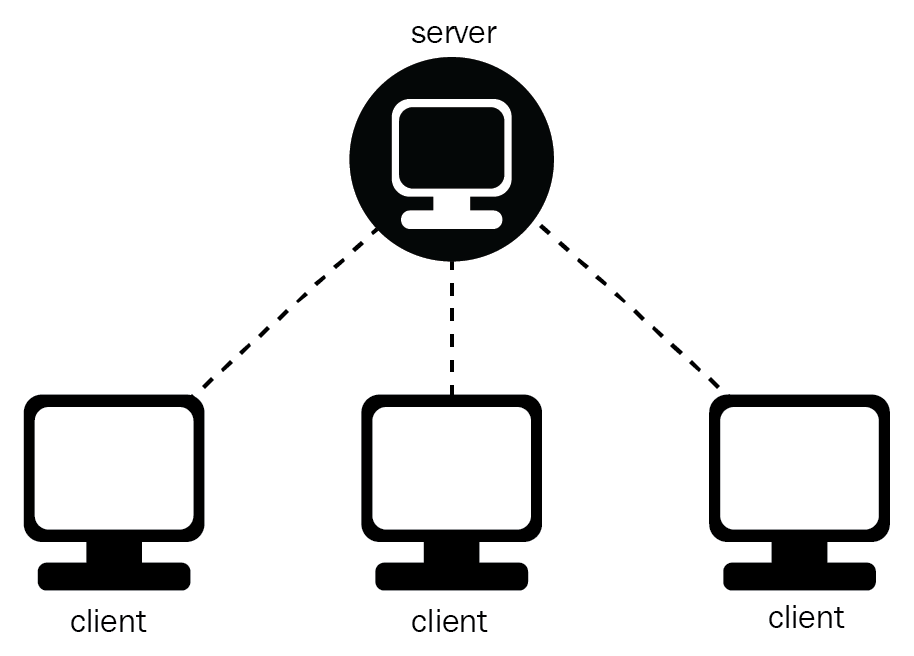
[6] Express. “API”. 2023, từ: https://expressjs.com/en/5x/api.html/

[7] W3school. “Learn to Code”. 2023, từ: https://w3schools.com/

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Mô hình mạng Client - Server

Mô hình Client Server là mô hình mạng máy tính trong đó các máy tính con được đóng vai trò như một máy khách, chúng làm nhiệm vụ gửi yêu cầu đến các máy chủ. Để máy chủ xử lý yêu cầu và trả kết quả về cho máy khách đó.



Hình 2.1.1: Mô hình Client – Server

**Nguyên tắc hoạt động**

Trong mô hình Client Server, server chấp nhận tất cả các yêu cầu hợp lệ từ mọi nơi khác nhau trên Internet, sau đó trả kết quả về máy tính đã gửi yêu cầu đó

Máy tính được coi là máy khách khi chúng làm nhiệm vụ gửi yêu cầu đến các máy chủ và đợi câu trả lời được gửi về.

Để máy khách và máy chủ có thể giao tiếp được với nhau thì giữa chúng phải có một chuẩn nhất định, và chuẩn đó được gọi là giao thức. (Giao thức là gì thì các bạn có thể tham khảo tại đây) Một số giao thức được sử dụng phổ biến hiện nay như: HTTPS, TCP/IP, FTP,...

Nếu máy khách muốn lấy được thông tin từ máy chủ, chúng phải tuân theo một giao thức mà máy chủ đó đưa ra. Nếu yêu cầu đó được chấp nhận thì máy chủ sẽ thu thập thông tin và trả về kết quả cho máy khách yêu cầu. Bởi vì Server - máy chủ luôn luôn trong trạng thái sẵn sàng để nhận request từ client nên chỉ cần client gửi yêu cầu tín hiệu và chấp nhận yêu cầu đó thì server sẽ trả kết quả về phía client trong thời gian ngắn nhất.

**Ưu điểm**

Ưu điểm mô hình Client - Server bao gồm:

Giúp chúng ta có thể làm việc trên bất kì một máy tính nào có hỗ trợ giao thức truyền thông. Giao thức chuẩn này cũng giúp các nhà sản xuất tích hợp lên nhiều sản phẩm khác nhau mà không gặp phải khó khăn gì.

Có thể có nhiều server cùng làm một dịch vụ, chúng có thể nằm trên nhiều máy tính hoặc một máy tính.

Chỉ mang đặc điểm của phần mềm mà không hề liên quan đến phần cứng, ngoài yêu cầu duy nhất là server phải có cấu hình cao hơn các client.

Hỗ trợ người dùng nhiều dịch vụ đa dạng và sự tiện dụng bởi khả năng truy cập từ xa.

Cung cấp một nền tảng lý tưởng, cho phép cung cấp tích hợp các kỹ thuật hiện đại như mô hình thiết kế hướng đối tượng, hệ chuyên gia, hệ thông tin địa lý (GIS).

**Nhược điểm**

Nhược điểm mô hình Client - Server đó là:

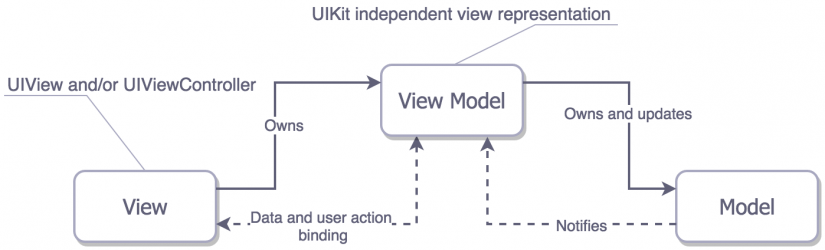
Vấn đề bảo mật dữ liệu thông tin đôi khi còn chưa được an toàn lắm vì do phải trao đổi dữ liệu giữa 2 máy tính khác nhau ở 2 khu vực địa lý cách xa nhau. Và đây cũng nhược điểm duy nhất của mô hình này.

Tuy nhiên vấn đề này thì có một số giao thức đã hỗ trợ bảo mật dữ liệu khi truyền tải. Giao thức được sử dụng phổ biến như HTTPS.

Mô hình Model - View - ViewModel

Mô hình MVVM là một mô hình định nghĩa cấu trúc ứng dụng, được phát triển dựa trên kiến trúc của MVP. Mô hình này cho phép người dùng tách biệt giữa Model (dữ liệu), ViewModel (mã thực thi) và View (giao diện người dùng).

Khác với các mô hình truyền thống, mô hình MVVM không cho phép người dùng xử lý sự kiện Click và các hoạt động khác bằng cách viết mã trực tiếp trên một Button. Nguyên nhân là vì trong MVVM, các control như ListView, Button hay SearchBar,… không thể kết nối với dữ liệu một cách trực tiếp. Thay vào đó, bạn sẽ phải sử dụng thuộc tính Command (thuộc kiểu ICommand) để kết nối các hành động với ViewModel.



Hình 2.2.1: Mô hình Model - View - ViewModel (MVVM)

**View**

View được hiểu là thành phần giao diện mô tả dữ liệu của một ứng dụng, đồng thời đây cũng là thành phần duy nhất cho phép người dùng tương tác trong chương trình.

View trong MVVM được đánh giá là tích cực hơn so với các mô hình khác nhờ vào khả năng thực hiện hành vi và cung cấp phản hồi cho người dùng với một số tính năng nổi bật như Command, Binding,…

**Model**

Giống với mô hình MVC, Model là đối tượng cho phép bạn truy xuất dữ liệu và thao tác trên dữ liệu thật sự.

**ViewModel**

ViewModel là lớp trung gian của View với Model. Thành phần này tương đương với Controller trong mô hình MVC, có nhiệm vụ chứa mã lệnh cần thiết để thực hiện các lệnh Command, Data Binding,…

Cơ sở dữ liệu quan hệ

Cơ sở dữ liệu quan hệ là một loại cơ sở dữ liệu tổ chức dữ liệu thành một hoặc nhiều bảng hoặc quan hệ, mỗi bảng có một tên duy nhất và bao gồm một tập hợp các hàng và cột. Dữ liệu trong cơ sở dữ liệu quan hệ được cấu trúc và tổ chức, giúp dễ dàng tìm kiếm, truy xuất và quản lý.

Dữ liệu thường được lưu trữ ở dạng chuẩn hóa trong cơ sở dữ liệu quan hệ. Dữ liệu được chia thành các bảng nhỏ hơn, có liên quan với nhau, mỗi bảng có khóa hoặc mã định danh duy nhất. Mối quan hệ giữa các bảng này được xác định thông qua việc sử dụng các khóa ngoại, khóa này liên kết dữ liệu trong một bảng với dữ liệu trong một bảng khác.

Cơ sở dữ liệu quan hệ được sử dụng rộng rãi trong các ứng dụng khác nhau, bao gồm hệ thống kinh doanh và tài chính, nghiên cứu khoa học và thương mại điện tử. Chúng cung cấp một cách linh hoạt và có thể mở rộng để lưu trữ và quản lý lượng lớn dữ liệu trong khi vẫn đảm bảo tính toàn vẹn và nhất quán của dữ liệu thông qua các ràng buộc, chẳng hạn như khóa chính và khóa ngoại.

**Cấu trúc**

Cơ sở dữ liệu quan hệ được cấu trúc bằng cách sử dụng bảng, còn được gọi là quan hệ. Mỗi bảng bao gồm các hàng và cột, với mỗi hàng đại diện cho một bản ghi hoặc phiên bản dữ liệu và mỗi cột đại diện cho một thuộc tính hoặc trường cụ thể của dữ liệu. Một tập hợp các thuộc tính hoặc kiểu dữ liệu, chẳng hạn như văn bản, số, ngày hoặc Boolean, xác định các cột trong bảng. Mỗi cột cũng có một tên duy nhất, giúp xác định kiểu dữ liệu được lưu trữ trong cột đó.

Các hàng trong bảng đại diện cho các bản ghi hoặc phiên bản dữ liệu riêng lẻ. Mỗi hàng có một mã định danh duy nhất, được gọi là khóa chính. Khóa chính được sử dụng để liên kết các bản ghi trên các bảng khác nhau trong cơ sở dữ liệu. Mối quan hệ giữa các bảng trong cơ sở dữ liệu quan hệ được xác định bằng khóa ngoại. Khóa ngoại là một cột trong một bảng tham chiếu đến khóa chính của một bảng khác. Điều này cho phép dữ liệu liên quan được liên kết và truy cập từ các bảng khác nhau trong cơ sở dữ liệu.

Ngoài các bảng, cơ sở dữ liệu quan hệ cũng sử dụng các ràng buộc để đảm bảo tính toàn vẹn và nhất quán của dữ liệu. Các ràng buộc là các quy tắc hoặc điều kiện phải được đáp ứng trước khi dữ liệu có thể được chèn, cập nhật hoặc xóa khỏi cơ sở dữ liệu. Ví dụ về các ràng buộc bao gồm khóa chính, khóa ngoại, ràng buộc duy nhất và ràng buộc kiểm tra.

**Mô hình quan hệ**

Mô hình quan hệ là một mô hình dữ liệu được sử dụng để thiết kế và quản lý dữ liệu trong cơ sở dữ liệu quan hệ. Mô hình quan hệ được Edgar F. Codd giới thiệu vào năm 1970 và kể từ đó nó đã trở thành mô hình dữ liệu được sử dụng rộng rãi nhất cho cơ sở dữ liệu hiện đại.

Mô hình quan hệ dựa trên khái niệm bảng, còn được gọi là quan hệ. Mỗi bảng trong cơ sở dữ liệu đại diện cho một tập hợp dữ liệu có liên quan và mỗi hàng trong bảng đại diện cho một bản ghi hoặc phiên bản của dữ liệu đó. Mỗi cột trong bảng đại diện cho một thuộc tính hoặc trường cụ thể của dữ liệu.

Mối quan hệ giữa các bảng trong cơ sở dữ liệu được xác định bằng các khóa. Khóa chính là một cột hoặc tập hợp các cột trong bảng xác định duy nhất từng hàng trong bảng đó. Khóa ngoại là một cột trong một bảng tham chiếu đến khóa chính của một bảng khác, cho phép dữ liệu liên quan được liên kết trên các bảng khác nhau trong cơ sở dữ liệu.

Mô hình quan hệ cũng hỗ trợ các hoạt động truy vấn và thao tác dữ liệu trong cơ sở dữ liệu, chẳng hạn như CHỌN, CHÈN, CẬP NHẬT và XÓA. Các thao tác này được thực hiện bằng một ngôn ngữ đặc biệt gọi là Ngôn ngữ truy vấn có cấu trúc ( SQL ), ngôn ngữ này xác định các truy vấn và câu lệnh tương tác với cơ sở dữ liệu.

Một trong những lợi ích chính của mô hình quan hệ là tính linh hoạt và khả năng mở rộng của nó. Các bảng có thể được thêm, xóa hoặc sửa đổi để phù hợp với các yêu cầu dữ liệu đang thay đổi và mối quan hệ giữa các bảng có thể được xác định hoặc cập nhật dễ dàng khi cần. Ngoài ra, mô hình quan hệ cung cấp một cách tổ chức và quản lý dữ liệu nhất quán và chuẩn hóa, giúp dễ dàng duy trì và cập nhật cơ sở dữ liệu lớn và phức tạp theo thời gian.

**Lợi ích của hệ quản trị cơ sở dữ liệu quan hệ**

Hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS) mang lại nhiều lợi ích, một số lợi ích bao gồm:

Tính toàn vẹn của dữ liệu : RDBMS sử dụng nhiều ràng buộc khác nhau như khóa chính, khóa ngoại và ràng buộc kiểm tra để đảm bảo tính toàn vẹn của dữ liệu, giúp duy trì tính chính xác và nhất quán của dữ liệu.

Khả năng mở rộng : RDBMS có thể xử lý một lượng lớn dữ liệu và có thể dễ dàng tăng hoặc giảm quy mô khi cần. Họ cũng có thể hỗ trợ đồng thời nhiều người dùng và ứng dụng.

Tính linh hoạt : RDBMS cung cấp một cách linh hoạt để tổ chức và lưu trữ dữ liệu, vì các bảng có thể được thêm, xóa hoặc sửa đổi để đáp ứng các yêu cầu dữ liệu thay đổi.

Dễ sử dụng : Ngôn ngữ SQL được sử dụng trong RDBMS rất dễ học và dễ sử dụng, đồng thời cung cấp một cách tiêu chuẩn và nhất quán để tương tác với cơ sở dữ liệu.

Bảo mật dữ liệu : RDBMS cung cấp các tính năng bảo mật tích hợp như kiểm soát truy cập và xác thực người dùng để đảm bảo rằng chỉ những người dùng được ủy quyền mới có thể truy cập và sửa đổi dữ liệu.

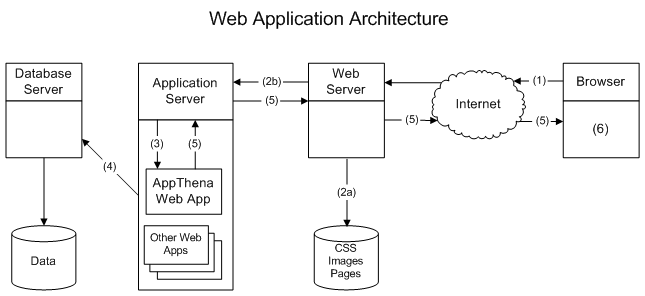
Tính nhất quán của dữ liệu : RDBMS sử dụng các giao dịch để đảm bảo rằng dữ liệu nhất quán và đáng tin cậy, ngay cả khi hệ thống bị lỗi hoặc gián đoạn.

Chia sẻ dữ liệu : RDBMS có thể chia sẻ dữ liệu trên các ứng dụng và nền tảng khác nhau, cải thiện sự hợp tác và năng suất trong các tổ chức.

RDBMS cung cấp một cách mạnh mẽ và đáng tin cậy để quản lý dữ liệu và được sử dụng rộng rãi trong nhiều ứng dụng, bao gồm các hệ thống kinh doanh và tài chính, nghiên cứu khoa học và thương mại điện tử.

Kiến trúc hệ thống

Kiến trúc ứng dụng Website (Website application architecture) mô tả sự tương tác giữa các ứng. dụng, cơ sở dữ liệu (database) và các hệ thống phần mềm trung gian trên web. Kiến trúc này đảm bảo nhiều ứng dụng có thể hoạt động đồng thời và trơn tru.



Hình 2.4.1: Kiến trúc ứng dụng Website

**Cách hoạt động**

Với bất kỳ ứng dụng Website điển hình nào, bạn cũng sẽ cần hai mã (chương trình con) khác nhau chạy song song. Đó là:

Code từ phía máy khách (frontend): Đoạn code này nằm trong Browser và phản hồi input (đầu vào) của người dùng.

Code phía máy chủ (backend): Code nằm trên máy chủ và phản hồi các yêu cầu HTTP

Một nhà phát triển web sẽ quyết định xem mã trên phía máy chủ sẽ làm gì với mã trong trình duyệt - phía máy khách. Để viết mã từ phía máy chủ, bạn sẽ cần sử dụng các ngôn ngữ lập trình như C#, Java, JavaScript, Python, PHP, Ruby,...

Bất kỳ code nào có thể đáp ứng các yêu cầu HTTP đều có khả năng chạy trên phía máy chủ. Mã phía máy chủ chịu trách nhiệm tạo trang mà người dùng yêu cầu cũng như lưu trữ các loại dữ liệu khác nhau, bao gồm hồ sơ người dùng và đầu vào của họ.

Sự kết hợp của CSS, HTML, JavaScript được sử dụng để viết mã phía máy khách. Mã này sẽ được trình duyệt website phân tích cú pháp. Không giống mã phía máy chủ, mã phía máy khách có thể được người dùng nhìn thấy và chỉnh sửa. Mã phía máy khách chỉ giao tiếp thông qua các yêu cầu HTTP và không thể đọc trực tiếp các tệp từ máy chủ.

**Các thành phần**

Khi nhắc đến các thành phần web application, chúng ta có thể hiểu đó là bất kỳ phần nào trong số 2 thành phần sau:

Thành phần UI/UX của ứng dụng Website: Bao gồm nhật ký hoạt động, trang tổng quát, thông báo, cài đặt, thống kê,... Các thành phần này không liên quan gì đến hoạt động của kiến trúc ứng dụng web. Thay vào đó, chúng là một phần của sơ đồ bố cục giao diện của ứng dụng web.

Các thành phần cấu trúc: Hai thành phần cấu trúc chính của Website là phía máy khách (Frontend) và máy chủ (Backend):

Thành phần phía máy khách: Các component phía máy khách (frontend) được phát triển trong CSS, HTML và JS. Vì nó tồn tại trong trình duyệt web của người dùng nên không cần có những điều chỉnh liên quan đến hệ điều hành hoặc thiết bị. Phần frontend đại diện cho chức năng của ứng dụng web mà người dùng tương tác cùng.

Thành phần phía máy chủ: Phần backend có thể được xây dựng bằng cách sử dụng một hoặc kết hợp nhiều ngôn ngữ lập trình cùng các framework, bao gồm các ngôn ngữ như Java, .NET, NodeJS, PHP, Python, Ruby on Rails,...Component của máy chủ phải có ít nhất hai thành phần là logic ứng dụng và cơ sở dữ liệu. Logic ứng dụng là trung tâm điều khiển chính của ứng dụng web trong khi cơ sở dữ liệu là nơi lưu trữ tất cả các dữ liệu một cách liên tục.

Ngôn ngữ

**HTML**

HTML hay HyperText Markup Language - Ngôn ngữ đánh dấu siêu văn bản, là ngôn ngữ được sử dụng cho các tài liệu web.

Trên thực tế, đây không phải là một ngôn ngữ lập trình, HTML giống như một ngôn ngữ xác định đâu là ý nghĩa, mục đích và cấu trúc của một tài liệu. Cùng với CSS và JavaScript, HTML tạo ra bộ ba nền tảng kỹ thuật cho các website.

**Vai trò của HTML**

Một website thường chứa nhiều trang con. Mỗi trang con sẽ có một tập tin HTML riêng. Dù bạn lập trình bằng bất cứ ngôn ngữ nào, trên bất cứ Framework nào, khi chạy trên nền website, chúng đều được biên dịch ra ngôn ngữ HTML. Đa số các trình soạn thảo văn bản trên website đều có 2 chế độ xem: HTML và văn bản thường.

Chức năng chính của HTML là xây dựng cấu trúc siêu văn bản trên một website, hoặc khai báo các tập tin kỹ thuật số (media) như hình ảnh, video, nhạc. HTML thường được dùng để phân chia các đoạn văn, heading, links, blockquotes,…

***Với HTML, bạn sẽ có thể:***

Thêm tiêu đề, định dạng đoạn văn, ngắt dòng điều khiển.

Tạo danh sách, nhấn mạnh văn bản, tạo ký tự đặc biệt, chèn hình ảnh, tạo liên kết.

Xây dựng bảng, điều khiển một số kiểu mẫu

Như đã đề cập ở trên, HTML không phải là ngôn ngữ lập trình. Do đó, HTML cũng không thể tạo ra các chức năng “động” được. Hiểu đơn giản, HTML giống như Microsoft Word, dùng để bố cục và định dạng trang web.

Tóm lại, HTML là ngôn ngữ markup, dễ học, dễ hiểu, dễ áp dụng. Tuy nhiên, một website được viết bằng HTML rất đơn giản, nếu không muốn nói là nhàm chán. Để gây hứng thú với người truy cập, website cần có sự hỗ trợ của CSS và JavaScript.

Nếu HTML cung cấp cấu trúc cơ bản của các trang web thì CSS và JavaScript sẽ cải tiến và sửa đổi website sinh động hơn.

**CSS**

CSS là viết tắt của cụm từ “Cascading Style Sheets”, tạm dịch: ngôn ngữ tạo phong cách cho trang web. Ngôn ngữ lập trình này quy định cách các thành phần HTML của trang web thực sự sẽ xuất hiện trên frontend như thế nào.

Hiểu đơn giản, CSS sẽ giúp webmaster xác định styles và định nghĩa nhiều loại nội dung của website. CSS được tạo ra để kết hợp với ngôn ngữ markup HTML nhằm tạo nên phong cách cho trang web.

***Có 3 loại style CSS chính:***

Style CSS Internal: là style được tải lên mỗi khi trang web được refresh.

Style CSS Inline: bạn có thể chỉnh sửa một yếu tố nào đó mà không cần truy cập trực tiếp vào file CSS.

External style: bạn có thể tạo phong cách ở file khác áp dụng CSS vào trang bạn muốn. External style sẽ cải thiện thời gian tải trang rất nhiều.

*Đặc biệt, bạn có thể sử dụng External Style CSS để tạo phong cách cho nhiều trang cùng lúc.*

**Vai trò của CSS**

Tạo phong cách và định kiểu cho những yếu tố được viết dưới dạng ngôn ngữ đánh dấu, như HTML.

Tiết kiệm công sức của lập trình viên nhờ điều khiển định dạng của nhiều website

Phân biệt cách hiển thị của trang web với nội dung chính của trang bằng cách điều khiển bố cục, màu sắc và font chữ.

Nếu HTML cung cấp các công cụ thô cần thiết để cấu trúc nội dung trên một website thì CSS sẽ giúp định hình kiểu nội dung này để trang web xuất hiện trước người dùng theo một cách đẹp hơn.

**Javascript**

JavaScript là ngôn ngữ lập trình được nhà phát triển sử dụng để tạo trang web tương tác. Từ làm mới bảng tin trên trang mạng xã hội đến hiển thị hình ảnh động và bản đồ tương tác, các chức năng của JavaScript có thể cải thiện trải nghiệm người dùng của trang web. Là ngôn ngữ kịch bản phía máy khách, JavaScript là một trong những công nghệ cốt lõi của World Wide Web. Ví dụ: khi duyệt internet, bất cứ khi nào bạn thấy quảng cáo quay vòng dạng hình ảnh, menu thả xuống nhấp để hiển thị hoặc màu sắc phần tử thay đổi động trên trang web cũng chính là lúc bạn thấy các hiệu ứng của JavaScript.

**Vai trò của Javascript**

Trước đây, các trang web có dạng tĩnh, tương tự như các trang trong một cuốn sách. Một trang tĩnh chủ yếu hiển thị thông tin theo một bố cục cố định và không làm được mọi thứ mà chúng ta mong đợi như ở một trang web hiện đại. JavaScript dần được biết đến như một công nghệ phía trình duyệt để làm cho các ứng dụng web linh hoạt hơn. Sử dụng JavaScript, các trình duyệt có thể phản hồi tương tác của người dùng và thay đổi bố cục của nội dung trên trang web.

Khi ngôn ngữ này phát triển hoàn thiện, các nhà phát triển JavaScript đã thiết lập các thư viện, khung và cách thức lập trình cũng như bắt đầu sử dụng ngôn ngữ này bên ngoài trình duyệt web. Ngày nay, bạn có thể sử dụng JavaScript để thực hiện hoạt động phát triển cả ở phía máy khách và máy chủ:

**JavaScript phía máy khách**

JavaScript phía máy khách đề cập đến cách thức JavaScript hoạt động trong trình duyệt của bạn. Trong trường hợp này, công cụ JavaScript nằm trong mã trình duyệt. Tất cả các trình duyệt web phổ biến đều có các công cụ JavaScript tích hợp riêng.

Các nhà phát triển ứng dụng web viết mã JavaScript với các hàm khác nhau được liên kết với nhiều loại sự kiện, chẳng hạn như nhấp chuột hoặc khi di chuột đến. Các hàm này thực hiện các thay đổi đối với HTML và CSS.

**JavaScript phía máy chủ**

JavaScript phía máy chủ đề cập đến việc sử dụng ngôn ngữ viết mã trong logic máy chủ back-end. Trong trường hợp này, công cụ JavaScript nằm trực tiếp trên máy chủ. Hàm JavaScript phía máy chủ có thể truy cập cơ sở dữ liệu, thực hiện các phép toán logic khác nhau và phản hồi nhiều loại sự kiện do hệ điều hành của máy chủ kích hoạt. Ưu điểm chính của viết tập lệnh phía máy chủ là bạn có khả năng tùy chỉnh linh hoạt phản hồi của trang web dựa trên yêu cầu và quyền truy cập của mình cũng như theo yêu cầu thông tin từ trang web.

Công nghệ sử dụng

**Node.js**

Node.js là một platform (nền tảng) độc lập được xây dựng trên môi trường Javascript Runtime - trình thông dịch ngôn ngữ Javascript cực nhanh trên trình duyệt Chrome. Dựa trên Node.js, chúng ta có thể xây dựng các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng.

Node.js được thiết kế bởi Ryan Lienhart Dahl - một kỹ sư phần mềm người Mỹ vào năm 2009 và phát triển dưới sự bảo trợ của Joyent.

Phần Core bên dưới của Nodejs được viết hầu hết bằng C++ nên cho tốc độ xử lý và hiệu năng khá cao.

Các ứng dụng được tạo ra bởi Node.js có tốc độ xử lý nhanh, realtime thời gian thực.

Node.js sẽ thích hợp cho các sản phẩm có lưu lượng truy cập lớn, cần sự thay đổi về công nghệ và cần được mở rộng hoặc tạo ra các dự án Startup một cách nhanh nhất có thể.

**Vai trò của Node.js**

Các ứng dụng Nodejs được viết bằng javascript, ngôn ngữ này là một ngôn ngữ khá thông dụng. Theo tác giả của ngôn ngữ Javascript, Ryan Dahl: “Javascript có những đặc tính mà làm cho nó rất khác biệt so với các ngôn ngữ lập trình động còn lại, cụ thể là nó không có khái niệm về đa luồng, tất cả là đơn luồng và hướng sự kiện.”

Nodejs chạy đa nền tảng phía Server, sử dụng kiến trúc hướng sự kiện Event-driven, cơ chế non-blocking I/O làm cho nó nhẹ và hiệu quả.

Có thể chạy ứng dụng Nodejs ở bất kỳ đâu trên máy Mac - Window - Linux, hơn nữa cộng đồng Nodejs rất lớn và hoàn toàn miễn phí. Các bạn có thể thấy cộng đồng Nodejs lớn như thế nào tại đây, các package đều hoàn toàn free.

Các ứng dụng NodeJS đáp ứng tốt thời gian thực và chạy đa nền tảng, đa thiết bị.

**Express**

Express hay còn được viết là Express Js, Express.js. Đây là một framework mã nguồn mở miễn phí cho Node.js. Express được sử dụng trong thiết kế và xây dựng các ứng dụng web một cách đơn giản và nhanh chóng.

Vì Express chỉ yêu cầu ngôn ngữ lập trình Javascript nên việc xây dựng các ứng dụng web và API trở nên đơn giản hơn với các lập trình viên và nhà phát triển. Express cũng là một khuôn khổ của Node.js do đó hầu hết các mã code đã được viết sẵn cho các lập trình viên có thể làm việc.

Nhờ có Express mà các nhà lập trình có thể dễ dàng tạo các ứng dụng một website, nhiều web hoặc kết hợp. Do có dung lượng khá nhẹ, Express giúp cho việc tổ chức các ứng dụng web thành một kiến trúc MVC có tổ chức hơn.Để có thể sử dụng được mã nguồn này, chúng ta cần phải biết về Javascript và HTML.

Express cũng là một phần của công nghệ giúp quản lý các ứng dụng web một cách dễ dàng hơn hay còn được gọi là ngăn xếp phần mềm MEAN.Nhờ có thư viện Javascript của Express đã giúp cho các nhà lập trình xây dựng nên các ứng dụng web hiệu quả và nhanh chóng hơn. Express cũng được sử dụng để nâng cao các chức năng của Node.js.

Trên thực tế, nếu không sử dụng Express, bạn sẽ phải thực hiện rất nhiều bước lập trình phức tạp để xây dựng nên một API hiệu quả. Express đã giúp cho việc lập trình trong Node.js trở nên dễ dàng hơn và có nhiều tính năng mới bổ sung.

**Vai trò của Express**

Phát triển máy chủ nhanh chóng: Express cung cấp nhiều tính năng dưới dạng các hàm để dễ dàng sử dụng ở bất kỳ đâu trong chương trình. Điều này đã loại bỏ nhu cầu viết mã từ đó tiết kiệm được thời gian.

Phần mềm trung gian Middleware: Đây là phần mềm trung gian có quyền truy cập vào cơ sở dữ liệu, yêu cầu của khách hàng và những phần mềm trung gian khác. Phần mềm Middleware này chịu trách nhiệm chính cho việc tổ chức có hệ thống các chức năng của Express.

Định tuyến - Routing: Express cung cấp cơ chế định tuyến giúp duy trì trạng thái của website với sự trợ giúp của URL.

Tạo mẫu - Templating: Các công cụ tạo khuôn mẫu được Express cung cấp cho phép các nhà xây dựng nội dung động trên các website bằng cách tạo dựng các mẫu HTML ở phía máy chủ.

Gỡ lỗi - Debugging: Để phát triển thành công các ứng dụng web không thể thiết đi việc gỡ lỗi. Giờ đây với Express việc gỡ lỗi đã trở nên dễ dàng hơn nhờ khả năng xác định chính xác các phần ứng dụng web có lỗi.

**PostgreSQL**

PostgreSQL là một hệ thống quản trị cơ sở dữ liệu quan hệ-đối tượng (object-relational database management system) có mục đích chung, hệ thống cơ sở dữ liệu mã nguồn mở tiên tiến nhất hiện nay.

PostgreSQL được phát triển dựa trên POSTGRES 4.2 tại phòng khoa học máy tính Berkeley, Đại học California.

PostgreSQL được thiết kế để chạy trên các nền tảng tương tự UNIX. Tuy nhiên, PostgreSQL sau đó cũng được điều chỉnh linh động để có thể chạy được trên nhiều nền tảng khác nhau như Mac OS X, Solaris và Windows.

PostgreSQL là một phần mềm mã nguồn mở miễn phí. Mã nguồn của phần mềm khả dụng theo license của PostgreSQL, một license nguồn mở tự do. Theo đó, bạn sẽ được tự do sử dụng, sửa đổi và phân phối PostgreSQL dưới mọi hình thức.

PostgreSQL không yêu cầu quá nhiều công tác bảo trì bởi có tính ổn định cao. Do đó, nếu bạn phát triển các ứng dụng dựa trên PostgreSQL, chi phí sở hữu sẽ thấp hơn so với các hệ thống quản trị dữ liệu khác.

**JSON Web Token (JWT)**

JWT (JSON Web Tokens) là một chuỗi mã hoá hay một tiêu chuẩn mở RFC 7519 được sử dụng như một phương tiện đại diện nhỏ gọn có khả năng thông qua chuỗi JSON để truyền đạt thông tin giữa server và client.

JSON là một dạng dữ liệu được sử dụng theo một quy luật chung nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể dễ dàng đọc và tìm hiểu. Nó được xem là một tiêu chuẩn mở với mục đích sử dụng là trao đổi dữ liệu trong website.

Token là một chữ ký điện tử được mã hoá thành các con số khác nhau và các con số này tạo thành một dãy số ấn tượng. Do được tạo dưới dạng OTP nên loại mã số này chỉ được tạo ngẫu nhiên và sử dụng trong một lần cho từng lần giao dịch khác nhau.

Các thông tin trong chuỗi JWT được định dạng bằng JSON đồng thời được đảm bảo tính bảo mật và gia tăng mức độ tin tưởng dựa trên chữ ký điện tử. Chữ ký điện tử này sẽ sử dụng các Public Key (khóa công khai) và Private Key (khóa riêng tư) có ứng dụng ECDSA và RSA với thuật toán HMAC.

**Lợi ích của việc sử dụng JWT:**

Ủy quyền - Authorization: Đây là trường hợp nên sử dụng JWT. Khi người dùng đã đăng nhập, mỗi request tiếp theo được gửi từ Client sẽ bao gồm JWT, cho phép người dùng access vào routes, services, and resources được phép với token đó. Single Sign ON là tính năng sử dụng JWT rộng rãi hiện nay, vì chi phí thấp và dễ dàng sử dụng trên các domains khác nhau.

Trao đổi thông tin - Information Exchange: JSON Web Tokens là một cách tốt để truyền thông tin an toàn giữa các bên Client và Server. Vì JWT có thể signed. Ví dụ, sử dụng các cặp public/private key, bạn có thể biết chắc người gửi. Ngoài ra, vì signature được xác định dựa vào header và payload, bạn cũng có thể xác minh rằng nội dung chưa bị giả mạo.

**Reactjs**

ReactJS được hiểu nôm na là một thư viện mã nguồn mở chứa nhiều JavaScript và người tạo ra ReactJS chính là ông trùm với cái tên quen thuộc Facebook, được ra đời vào năm 2013. Mục đích của việc tạo ReactJS là tạo ra các ứng dụng web nhanh, hiệu quả và hấp dẫn với nỗ lực viết mã tối thiểu. Mục tiêu chính của ReactJS là bất kỳ trang web nào sử dụng ReactJS phải mượt mà, nhanh chóng, khả năng mở rộng cao và dễ thực hiện.

Nhìn chung, các tính năng và điểm mạnh của ReactJS thường đến từ việc tập trung vào các phần riêng lẻ. Do đó, khi làm việc với web thay vì toàn bộ ứng dụng của một trang web bằng ReactJS, các nhà phát triển tính năng có thể tách rời và chuyển đổi giao diện người dùng từ những cách phức tạp và biến nó thành những phần đơn giản hơn. Điều này có nghĩa là kết xuất dữ liệu không chỉ ở phía máy chủ. Thực hiện với các vị trí, nhưng bạn cũng có thể thực hiện tại vị trí khách hàng khi sử dụng ReactJS.

**Redux**

Nói một cách đơn giản, Redux là một công cụ quản lý trạng thái. Mặc dù nó được sử dụng chủ yếu với React, nhưng nó có thể được sử dụng với bất kỳ khung hoặc thư viện JavaScript nào khác. Nó rất nhẹ ở mức 2KB (bao gồm cả phụ thuộc), vì vậy bạn không phải lo lắng về việc nó làm cho ứng dụng của bạn có kích thước lớn.

Với Redux, trạng thái ứng dụng của bạn được giữ trong một "store" và mỗi thành phần có thể truy cập bất kỳ trạng thái nào mà nó cần từ "store"này. Sâu hơn một chút để xem tại sao bạn có thể cần một công cụ quản lý trạng thái.

**Redux - Saga**

Redux saga là một thư viện giúp các side effect của ứng dụng tức là những thứ không đồng bộ như tìm nạp dữ liệu và những thứ không tinh khiết như truy cập bộ nhớ cache của trình duyệt dễ quản lý hơn, thực thi hiệu quả hơn, dễ kiểm tra và xử lý lỗi tốt hơn.

Redux-saga là một phần mềm trung gian của redux (Redux là một thư viện JavaScript mã nguồn mở để quản lý trạng thái ứng dụng. Nó được sử dụng phổ biến nhất với các thư viện như React hoặc Angular để xây dựng giao diện người dùng). Phần mềm Redux này có thể được bắt đầu, tạm dừng và hủy khỏi ứng dụng chính với các hành động redux bình thường.

Redux saga có quyền truy cập vào trạng thái ứng dụng redux đầy đủ và gửi các hành động Redux.

Redux saga sử dụng một tính năng ES6 được gọi là Generators khiến các luồng không đồng bộ đó dễ đọc, ghi và kiểm tra. Bằng cách sử dụng Redux saga, các luồng không đồng bộ này trông giống như mã JavaScript đồng bộ tiêu chuẩn của bạn.

**Ant Design**

Trong quá trình phát triển các ứng dụng trên máy tính, có nhiều thông số thiết kế và hệ thống xử lý khác nhau rất phức tạp, điều này gây khó khăn gấp đôi cho các nhà thiết kế và lập trình viên, đồng thời sẽ làm giảm hiệu quả cho việc phát triển này. Sau khi có cho mình các dự án lớn và sơ lược lại, đội ngũ Ant UED đã cho ra mắt ngôn ngữ thiết kế Ant Design dành cho giao diện của ứng dụng, nhằm mục đích nhất quán các thông số giao diện người dùng trên nền tảng dự án nội bộ, giảm chi phí không cần thiết của việc triển khai ý tưởng và cung cấp nhiều tài nguyên cho việc thiết kế và lập trình front-end.

Ant Design đặc biệt được tạo ra cho các ứng dụng máy tính nội bộ, cam kết trải nghiệm của người dùng và các nhà thiết kế sản phẩm. Những người thiết kế giao diện và thiết kế trải nghiệm cho người dùng được gọi chung là thiết kế sản phẩm, Ant Design sẽ xóa bỏ ranh giới giữa việc quản lý sản phẩm, thiết kế tương tác, thiết kế hình ảnh, phát triển giao diện và quản trị dữ liệu người dùng. Nhờ việc tận dụng sự đồng nhất về đặc điểm kỹ thuật, Ant Design đơn giản hóa việc thiết kế và lập trình cho những người đang thực hiện Project, điều này sẽ giúp họ có thêm nhiều kinh nghiệm và tăng hiệu quả phát triển giao diện của sản phẩm và ứng dụng.

**Material UI**

Material Design là một phiên bản phát triển hơn so với Flat Design (thiết kế phẳng) và được các designer vô cùng yêu thích trong thời gian gần đây. Không giống hoàn toàn với Flat Design khi mang đến cảm giác cả thiết kế là một mặt phẳng, Material Design thực chất là các lớp Flat xếp chồng lên nhau để tạo điểm nhấn và chiều sâu cho thiết kế của bạn.

Đối với phong cách thiết kế này, các designer thường tập trung chủ yếu vào các chi tiết như đường nét tối giản hay màu sắc được chọn là màu đậm để mang lại sự nổi bật cho tổng thể thiết kế. Ngoài ra các yếu tố đồ họa tạo cảm giác 3D cũng được sử dụng để tạo điểm nhấn cần thiết cho thiết kế, khiến người xem có cảm giác như hiệu ứng của bạn nổi lên trên giao diện.

Material Design tập trung vào đường nét tối giản và màu sắc đậm nét

Không chỉ gây ấn tượng với người xem bằng những yếu tố đồ họa 3D, Material Design còn sử dụng những chuyển động tự nhiên, ví dụ như các menu hay các nút hiện trên màn hình. Nhờ vậy mà người dùng có thêm những trải nghiệm mới mẻ và gần gũi hơn với đời sống thực.

**Font Awesome**

Font awesome là một thư viện để tạo ra những icon tiện lợi, nhẹ nhàng có tính tùy chỉnh cao trong website. Thay vì bạn phải mất công design icon bằng file ảnh nặng nề thì giờ đây có hàng ngàn icon miễn phí sử dụng nhanh chóng.