Sri Lanka Institute of Information Technology

Programming Applications and Frameworks (IT3030)

Continuous Assignment – 2024, Semester 2

Initial Document

GROUP ID: **JUN_WE_158**

Name with Initials	Registration Number
IT21228094	Mendis A.R. P
IT21225024	Bhagya P. S
IT21215292	Madhusanka J.A. A
IT21231100	Sandaruwan W.M.I.M

Content

1. Project Description

The project assigned to our group is to create a fitness-focused social media network. This website attempts to give users a specific area where they can share exercises, fitness journeys, and healthy living advice. Users can post workout status updates, trade training programme, and even reveal their meal plans—complete with recipes and dietary details—by uploading images and videos that highlight their actions through a smooth interface. Our objective is to enable people to monitor their advancement, motivate others, and establish connections with a community of like-minded fitness lovers, all while emphasizing ease of use and personalization.

These are the functions that related to our web application.

- Functionality for Uploading Media and Adding Descriptions

With this feature, users can share up to three images or videos (with a maximum duration of 30 seconds) that highlight their exercises, fitness routines, nutritious meals, and advancements. It should be possible for users to choose and upload media files straight from their devices, along with the ability to add captions to their media posts. For users to share their fitness journeys and accomplishments with the community, this tool is essential.

- Functionality for Sharing Plans and Workout Status:

This feature allows users to use pre-made templates to generate and post updates on their current workout status, including distance run, number of pushups accomplished, weight lifted, etc. It should also be possible for users to share their training schedules, which should include sets, repetitions, routines, and exercises. These plans can be updated at any moment. Sharing exercise programme with others and keeping track of progress both depend on this function.

- Functionality for Sharing and Managing Meal Plans:

This feature enables users to exchange meal plans with resources for meal planning and documentation, including recipes, nutritional data, and portion sizes. It must to be possible for users to group their meal plans according to specific dietary requirements, such vegetarian, vegan, keto, etc. In order to share recipes and healthy eating practices with the community, users must utilize this function.

- Functionality for Community Engagement and Interaction:

By enabling social sharing, this option lets users join a variety of fitness challenges and activities with friends and family. It promotes a sense of community and inspiration by encouraging users to share their results and accomplishments. The development of a community of encouragement among fitness lovers depends on this feature.

2. The functional requirements for the REST API and the client web application.

2.1. REST API

- Endpoints for Resources:
  Specify the URLs, HTTP methods, and request/response formats for every resource.
  As an example, /users (POST, GET), /users/{id} (PUT, DELETE, GET).

- Rate restriction
  In order to deter misuse and guarantee equitable use, the API ought to incorporate a mechanism that restricts the quantity of requests that can be submitted in a given amount of time.

- Post Management
  Posts ought to be created, retrieved, updated, and deleted by users. and Different forms of posts (pictures, videos, workout status updates, meal plans) should be supported.

- Error Handling
  The API For different situations (such as failed authentication, server problems, and client issues), define error codes and messages. and
  Indicate the error messages that clients should get (e.g., HTTP status codes, error objects).
  Example: Include informative error messages in the response body and use common HTTP status codes (401 Unauthorized, 404 Not Found, etc.).

- Documentation
  Give specifications for the API documentation, such as usage recommendations, request/response samples, and endpoint descriptions.
  Example: Use open API or Swagger to create interactive API documentation.

2.2. Client Web Application

- User Authentication
  Let users to Establish a system that allows users to sign up, log in, and log out. Provide support for generating passwords for secure storage. and Give users the option to reset forgotten passwords.

- User Profile management
  Let users to access and modify their personal profiles. Provide options for updating personal data, including a profile picture and email address.

- Dashboard
  Allow Create a dashboard that shows a summary of the user's activity or personalized information. And add modules or widgets to provide easy access to important information or functions.

- Searching and filtering
  Make search features available so users can find particular information or data.
  Enable the filtering options to customize search results according to different standards.

- Notifications
  Install a notification system to inform users of changes or significant events. Give users the option to view and modify their notification preferences.

- Data Visualization
  Use interactive tables, graphs, and charts to effectively convey data. as well Permit users to alter visualizations according to their own preferences.

- Error handling and Feedback
  When mistakes occur, give users relevant error messages and comments. To assist users in fixing errors, include validation messages and guidelines.

3. The non-functional requirements for the REST API and the client web application.

3.1. REST API

- Usability
  The API must conform to RESTful principles and have a standardized, user-friendly interface. Developers integrating with the API should be able to quickly understand standard error messages and status codes.

- Performance
  Even with a high volume of traffic, the API should reply to queries within a given amount of time. and it is important to optimize response times for frequent tasks like fetching postings.

- Reliability
  Clients should be able to receive informative error messages from the API and it should have procedures in place to handle mistakes graciously. With the proper backup and recovery procedures in place, it should be resilient to failures.

- Security
  Industry-standard techniques should be used to protect user authentication and data transfer. And Sensitive gateway access, such as updating user profiles, needs to be secured with the right authorization procedures.

- Scalability
  An increasing number of users and postings should not cause the API's performance to noticeably decline. In order to add extra servers or instances as needed, it should be designed to develop dynamically.

3.2. Client Web Application

- Performance
  Reaction time: The web application must load rapidly and react to user input in a timely manner.
  Efficiency: The system should be able to manage a specific quantity of users or requests at once without experiencing any performance impairment.
  Scalability: In order to handle increased load or usage, the application must be able to scale both horizontally and vertically.

- Security

  Data encryption: To avoid unwanted access or interception, sensitive data sent between the client and server should be encrypted.
  Authorization and authentication: To guarantee that only authorized users may access particular features or data, the application should impose access control and securely authenticate users.
  Compliance: The system is bound by all applicable security guidelines and laws, including PCI DSS, GDPR, and HIPAA.

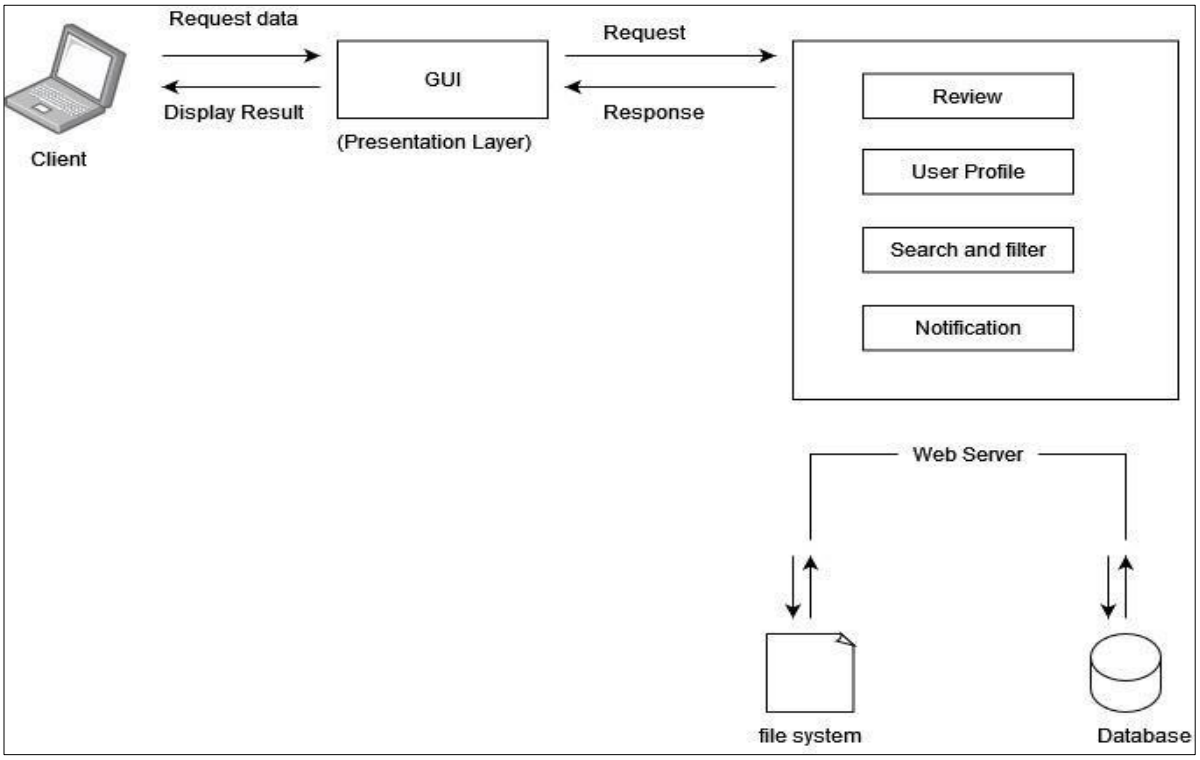
- Usability

  User interface design: The programme must to have an easy-to-use interface that is simple to navigate and understand.
  Accessibility: The application must adhere to WCAG principles and standards for accessibility in order to be used by people with impairments.
  Compatibility: There should be no significant problems with the application's ability to function consistently across a variety of web browsers, hardware, and operating systems.

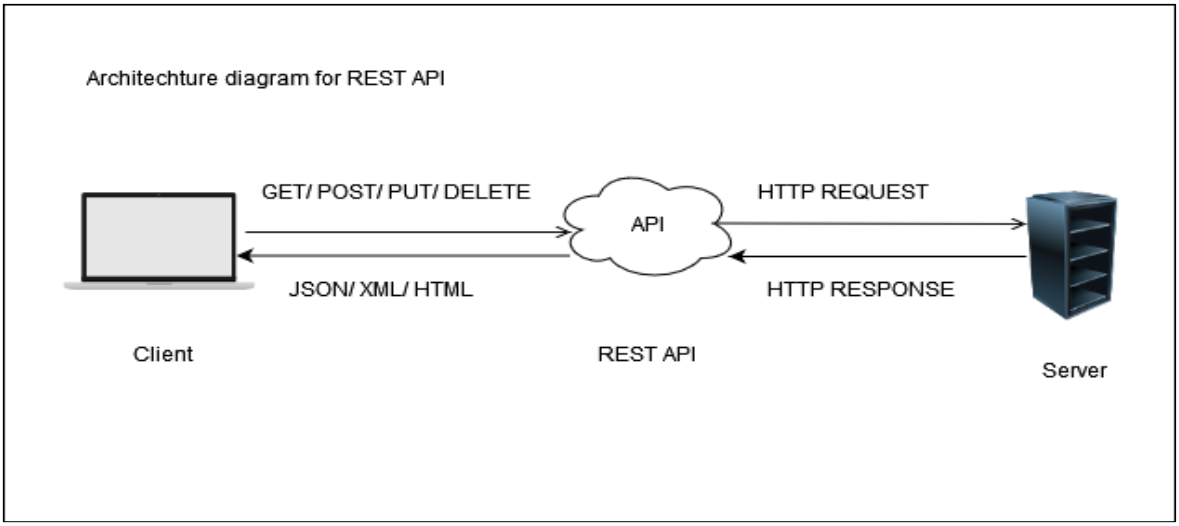
- Maintainability

  Code maintainability: To enable continuous maintenance and upgrades, the application code needs to be flexible, well-documented, and well-structured.
  Version control: Git and other version control systems, along with well-defined branching and merging procedures, should be used to manage the source code.
  Logging and monitoring: To facilitate troubleshooting and monitoring, the application should log pertinent events and metrics. It should also have tools in place to analyze and address problems.

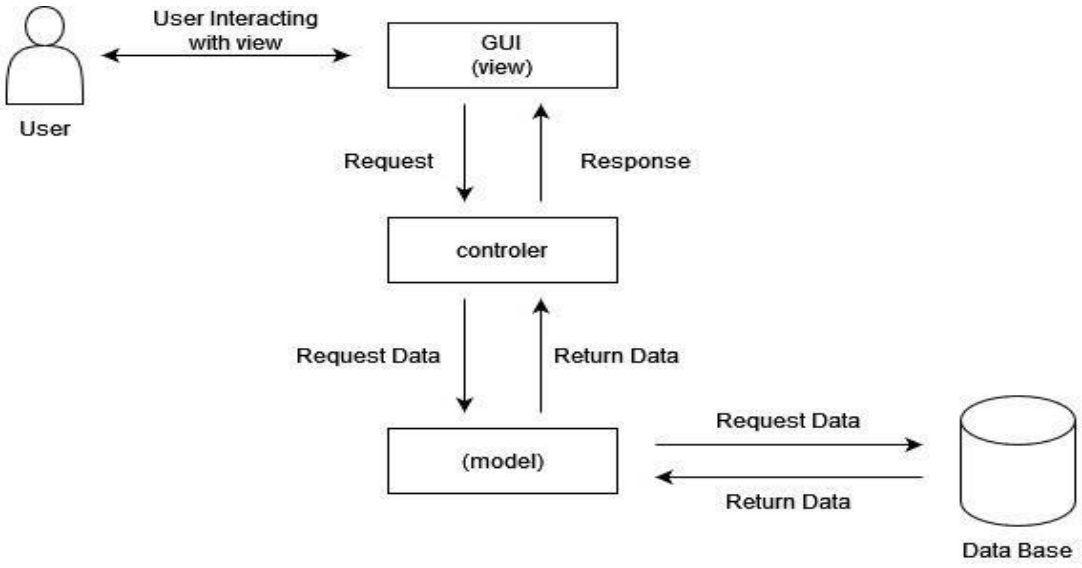
4. Architecture Diagram for the System



5. Architecture Diagram for REST API



6. Architecture Diagram for Client Web Application



7. References

1. What is an API?
   https://www.wallarm.com/what/the-concept-of-an-api-gateway
2. Difference between client web application & client/server application
   https://stackoverflow.com/questions/715063/what-is-the-difference-between-a-web-application-and-a-client-server-application
3. How to draw Architecture Diagrams
   https://nulab.com/learn/software-development/architectural-diagrams-what-to-know-and-how-to-draw-one/