Sri Lanka Institute of Information Technology

Programming Applications and Frameworks (IT3030)

Continuous Assignment – 2024, Semester 1

Initial Document

GROUP ID: ***IT_JUN_WE_47***

Gangabadage U.D	**IT21157400**
Kumara H.M.I.M	**IT21183904**
Disanayka D.M.W.B.T	**IT21164194**
Anjana E.K.C	**IT20641542**

1.Contents

This document should give a brief introduction to the assessment panel on how you are planning to implement the assignment. This document should **not** exceed seven (7) pages in length. The following details are expected.

1. A very brief project description
2. The functional requirements for the REST API and the client web application separately.
3. The non-functional requirements for the REST API and the client web application separately.
4. An overall architecture diagram for the entire system you are contracted to design and implement (you can ignore the mobile applications).
5. A detailed architecture diagram(s) for the REST API.
6. A detailed architecture diagram(s) for the client web application. You should decide upon suitable front-end architecture.
7. References

2.Brief description

Our project aims to develop a comprehensive social network tailored for fitness enthusiasts, fostering a supportive community where users can share their fitness journeys, training routines, and dietary strategies. Through streamlined features for media uploads, training and nutrition plan exchanges, and workout status sharing, the platform promises a user-friendly experience.

The media sharing function enables users to exhibit their workouts, healthy meals, fitness activities, and progress through up to three 30-second photos or videos. Utilizing the platform's interface, users can effortlessly select and upload media content from their devices, accompanied by descriptive captions.

For tracking workout progress, users can post updates on metrics such as weights lifted, pushups completed, and distances covered, with pre-designed templates for ease. Each post provides a succinct overview of the user's fitness achievements.

Furthermore, users can exchange workout programs comprising sets, reps, routines, and exercises. The platform offers customizable templates for users to design and adapt their workout plans in line with evolving fitness goals and preferences.

Facilitating the sharing of meal plans, the website accommodates portion sizes, recipes, and nutritional information. Users can input ingredients, cooking instructions, and images of their meals, categorizing plans based on dietary preferences like vegan, vegetarian, and keto options.

3.Functional Requirements

3.1 Functional requirements for REST API

Post Creation Endpoint:

Allow users to generate posts containing uploaded media files.

Permit users to include descriptions alongside their posts.

Verify input data to ensure correct formatting and adherence to length restrictions.

Media Upload Endpoint:

Permit users to upload a maximum of three photos or videos displaying fitness activities.

Restrict the duration of videos to no more than 30 seconds.

Enable the direct acceptance of media files from user devices.

Workout plan Management Endpoint:

Enable users to generate, modify, and remove workout plans.

Offer templates to assist users in specifying routines, exercises, sets, and repetitions.

Facilitate the customization of workout plans to align with changing fitness objectives.

Workout Status Update Endpoint:

Allow users to generate and distribute updates regarding their ongoing workout progress.

Offer pre-established templates for inputting workout metrics.

Incorporate a concise description of the fitness accomplishment with each update.

Meal Plan Management Endpoint:

Allow users to generate, modify, and remove meal plans.

Offer tools for users to input recipes, nutritional details, and serving sizes.

Incorporate functionalities for organizing meal plans according to dietary preferences.

Data validation and Error Handling

Verify input data to maintain consistency and integrity.

Incorporate error-handling mechanisms to deliver informative error messages to users.

User Authentication and Authorization

Incorporate authentication mechanisms to authenticate user identities.

Enforce authorization rules to limit access to specific endpoints according to user roles and permissions.

4.Functional Requirements

Functional requirements for Client Web Application

Responsive Design:

Ensure the web application is fully responsive and accessible across various devices and screen sizes.

Optimize performance to enhance loading times and deliver a smoother user experience.

User Registration and Authentication:

Enable users to register new accounts or log in using existing credentials securely.

Employ password hashing and robust authentication mechanisms to safeguard user accounts.

Post Creation Interface:

Empower users to craft posts enriched with uploaded media files.

Integrate fields for adding descriptions and tags to enrich the content of the posts.

Workout Plan Management Interface:

Create a user-friendly interface for the creation, modification, and deletion of workout plans.

Include forms or step-by-step wizards to guide users in defining their exercise routines.

Social Features:

Implement interactive features like liking, commenting on, and sharing posts within the platform.

Introduce a news feed or timeline to showcase posts from followed users.

Media Upload Interface:

Develop a user-friendly interface allowing seamless upload of photos and videos from user devices.

Incorporate progress indicators to keep users informed about the upload status.

Meal Plan Management Interface:

Offer robust tools for users to create, modify, and remove meal plans.

Implement functionality for categorizing meal plans according to dietary preferences.

Workout Status Update Interface:

Provide users with intuitive templates for inputting workout metrics.

Enable users to share concise descriptions of their fitness accomplishments alongside their updates.

Error Handling and Feedback:

Deliver informative error messages and feedback to users in case of invalid inputs or errors.

Utilize loading indicators or progress bars during data fetching or processing operations to enhance user experience.

5.Non - Functional Requirements

Non - Functional requirements for REST API

Scalability:

Ensure the API can scale horizontally, accommodating increased loads by adding more resources or instances.

Implement load balancing mechanisms to evenly distribute incoming requests across multiple server instances.

Security:

Implement authentication mechanisms like JWT (JSON Web Tokens) or OAuth to restrict API access to authorized users.

Utilize HTTPS encryption to secure data transmission and thwart eavesdropping or man-in-the-middle attacks.

Apply input validation and output encoding to prevent injection attacks such as SQL injection or Cross-Site Scripting (XSS).

Performance:

Guarantee that the API responds to requests within a predefined time limit, typically measured in milliseconds.

Handle concurrent requests efficiently, maintaining consistent response times across varying loads and traffic patterns.

Reliability:

Ensure the API maintains high availability, minimizing downtime and swiftly recovering from failures.

Implement fault tolerance mechanisms to gracefully handle unexpected errors without disrupting the entire system.

Compatibility:

The API should be compatible with a wide range of platforms, programming languages, and frameworks to maximize its accessibility and adoption.

Compatibility testing should be conducted regularly to ensure seamless integration with various client applications and systems.

Interoperability:

The API should adhere to industry standards and protocols to facilitate interoperability with other systems and services.

Integration with third-party APIs and services should be supported through well-defined interfaces and protocols.

6.Non - Functional Requirements

Non - Functional requirements for Client Web Application

Performance:

The application should load quickly, with minimal latency, to provide a seamless user experience.

Response times for user interactions, such as button clicks or form submissions, should be instantaneous.

Smooth animations and transitions should be implemented to enhance user engagement.

Scalability:

The application should be able to handle a growing user base and increased traffic without compromising performance.

It should be designed to scale horizontally or vertically to accommodate spikes in usage or data volume.

Reliability:

The application should be stable and dependable, with minimal downtime or service disruptions.

Error handling mechanisms should be in place to gracefully manage unexpected errors and prevent crashes.

Security:

Data transmitted between the client and server should be encrypted using HTTPS to prevent interception.

Measures such as input validation and output encoding should be implemented to mitigate security threats like XSS or CSRF attacks.

Authentication mechanisms should be robust to prevent unauthorized access to sensitive information or functionalities.

Usability:

The user interface should be intuitive and easy to navigate, with clear labeling and consistent design patterns.

Accessibility features should be incorporated to ensure the application is usable by people with disabilities.

Error messages should be informative and user-friendly, helping users understand and resolve issues effectively.

Compatibility:

The application should be compatible with a wide range of devices, browsers, and operating systems.

Compatibility testing should be conducted to ensure consistent performance and functionality across different platforms.

Maintainability:

The codebase should be well-organized and documented, making it easy for developers to understand and modify.

Version control should be used to track changes and facilitate collaboration among team members.

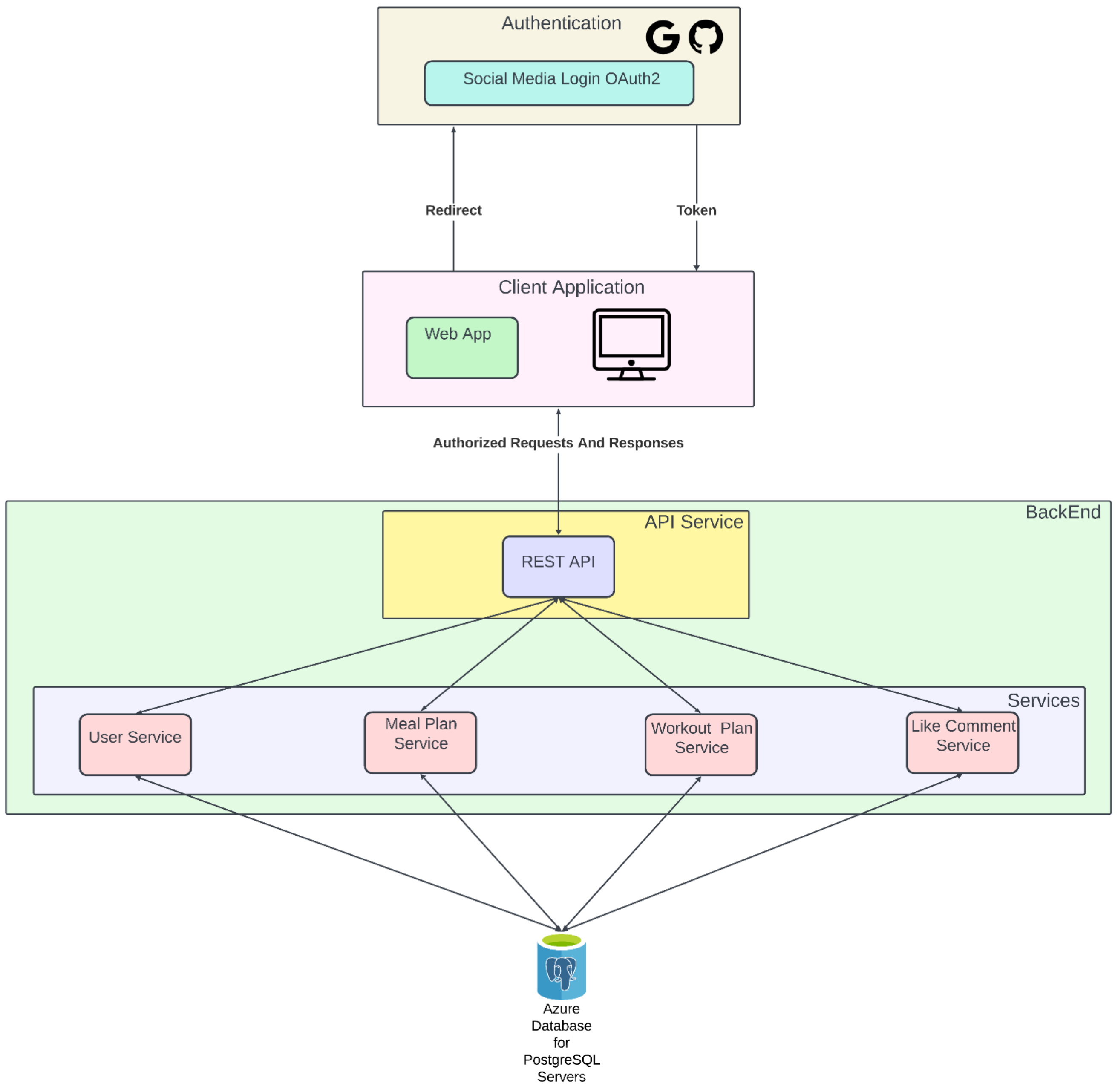
Automated testing and continuous integration practices should be implemented to detect and fix issues early in the development process.

Interoperability:

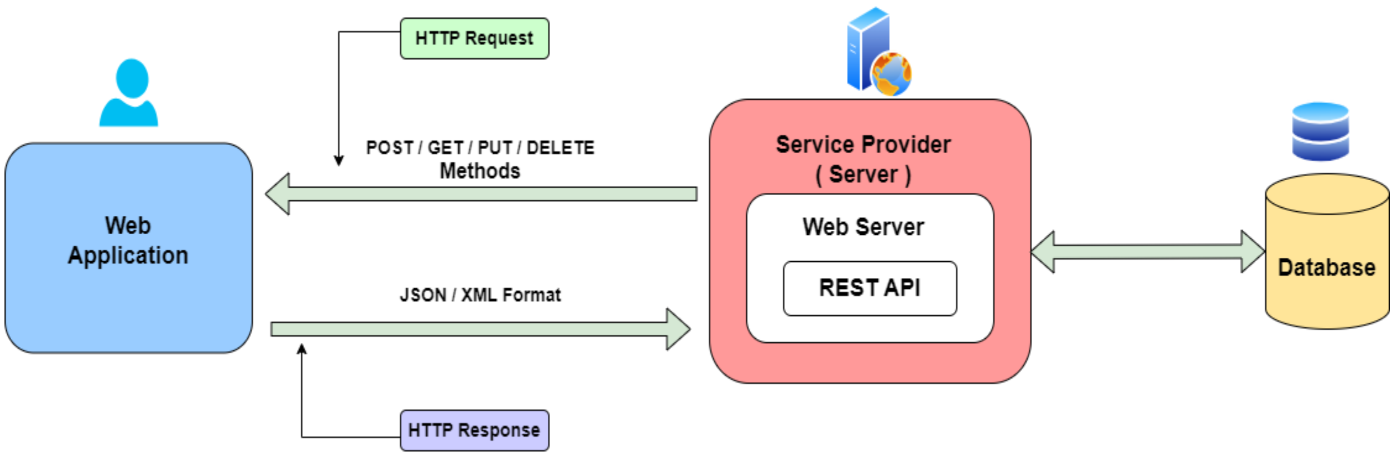
The application should be able to interact seamlessly with other systems and services through well-defined APIs or protocols.

Integration with third-party libraries or frameworks should be supported to leverage existing functionalities and resources.

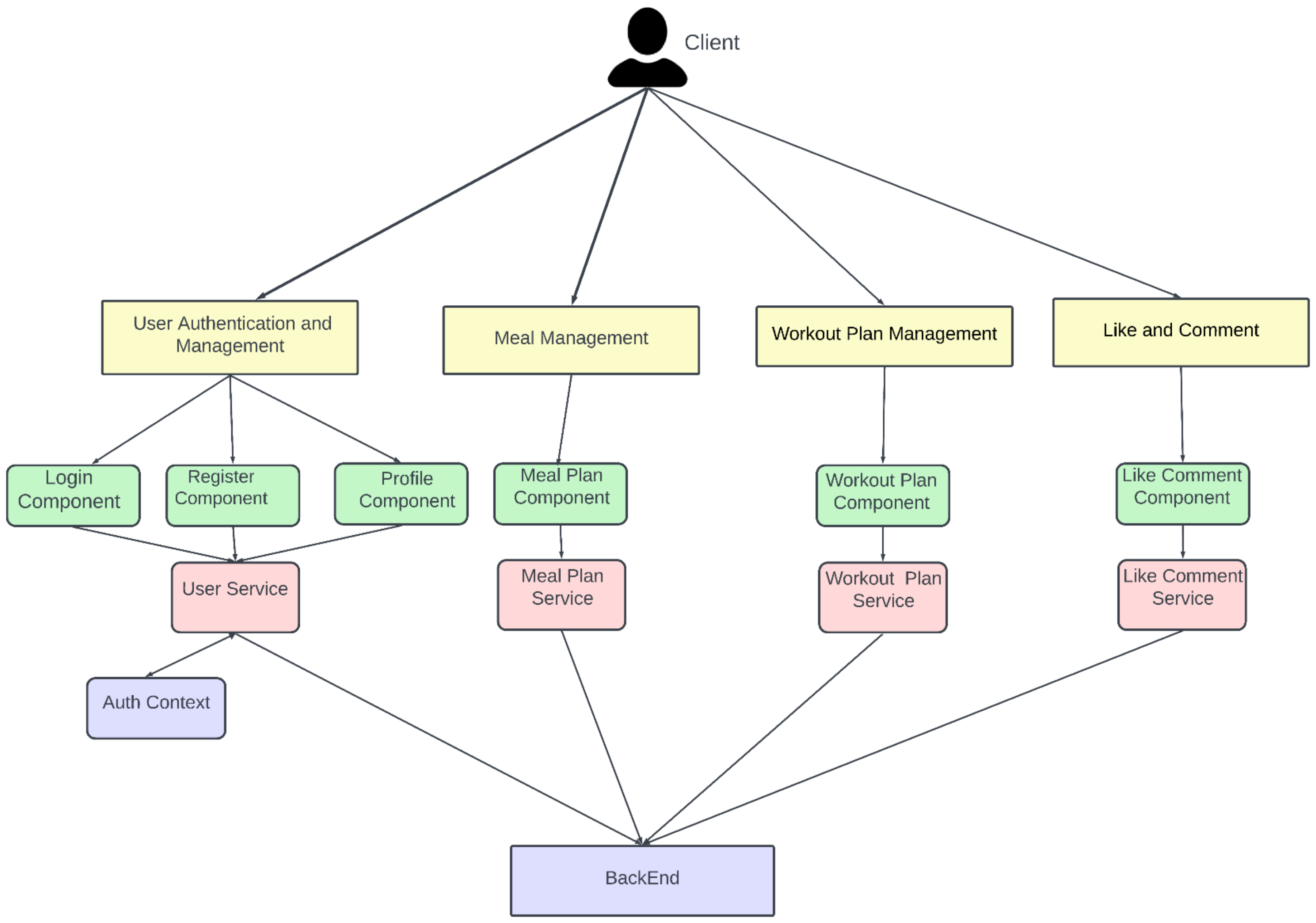
7.Overall architecture diagram for the entire system



8.Detailed Architecture Diagrams for the REST API



9.Detailed architecture diagrams for the client web application



10.References

1. *Getting started: Building Rest Services with spring* (no date) *Getting Started | Building REST services with Spring*. Available at: https://spring.io/guides/tutorials/rest (Accessed: 30 March 2024).
2. Madushan, I. (2024) *Web application architecture: The latest guide 2024 ;, ClickIT*. Available at: https://www.clickittech.com/devops/web-application-architecture/ (Accessed: 30 March 2024).
3. Prakash, V. (2021) *Web application architecture*, *Medium*. Available at: https://medium.com/geekculture/web-application-architecture-800d3ecd8019 (Accessed: 30 March 2024).
4. *What is a rest api?* (no date a) *IBM*. Available at: https://www.ibm.com/topics/rest-apis#:~:text=the%20next%20step-,What%20is%20a%20REST%20API%3F,transfer%20(REST)%20architectural%20style. (Accessed: 30 March 2024).
5. *What is a rest api?* (no date b) *IBM*. Available at: https://www.ibm.com/topics/rest-apis#:~:text=the%20next%20step-,What%20is%20a%20REST%20API%3F,transfer%20(REST)%20architectural%20style. (Accessed: 30 March 2024).
6. *What is rest?* (2023) *REST API Tutorial*. Available at: https://restfulapi.net/ (Accessed: 30 March 2024).