Sri Lanka Institute of Information Technology

Programming Applications and Frameworks (IT3030)

Continuous Assignment – 2025, Semester 2

Initial Document

GROUP ID: Y3S2-WE-147

Work Distribution

Student ID	Name	Contribution
IT22271396	Udawela U K R M R N	• Modified & created the final assignment document. • Architecture diagram for client web application & REST API. • Writing the project description
IT22137050	Malshan W A D D I	• Identified the functional requirements for REST API & Client web applications. • Helped modify the assignment document.
IT22261250	Lidapitiya L G S L H B	• Identified the Non-Functional requirements for REST API & client web application • Helped modify the assignment document • Gantt chart
IT22196224	Piyatissa M S N	• Overall architecture diagram for the entire system • Adding References

Contents

1. Project Description

Power Fitness: Your Ultimate Fitness Friend is now available.

Fitness enthusiasts everywhere are seeking a community where they can connect with likeminded people, share their experience, and receive inspiration in a world where health and wellness are of utmost importance. Introducing Power Fitness, a state-of-the-art social media network created just for those who are enthusiastic about exercise, fitness, and leading healthy lives.

Power Fitness allows users to easily publish their fitness accomplishments, wholesome meals, and motivational progress images and videos to a single, central portal. Power Fitness gives you the ability to demonstrate your commitment and inspire others while you work out, whether you're smashing a new personal record at the gym or preparing a tasty, high-protein meal in your home.

Our platform is for encouraging and support, not simply sharing. Make connections with other fitness enthusiasts, follow their adventures, and react to their postings by leaving comments and likes. Power Fitness ensures accessibility for all users by providing user-friendly features and simple interfaces that appeal to people with varying technical backgrounds. Power Fitness also places a high priority on security and privacy, giving users easy access to edit their material and limiting who may engage with their postings. Furthermore, users may easily join the Power Fitness community because of the platform's seamless connection with well-known social network accounts, which makes logging in a snap.

Join Power Fitness to start your fitness journey like never before—a place where community, motivation, and progress all come together. Come grow with us today and reach new levels of fitness.

2. Functional Requirements

2.1. Functional requirements for the Client Web Application

1. **Authorization of Users:**
    - Users must be able to sign up and log in securely.
    - Implement authentication using email/password and social logins.
    - Role-based access control (Admin, User, etc.).

2. **User management**
    - Users can update their profiles (name, age, preferences, etc.).
    - Admin can view, edit, and delete user accounts.
    - Password reset and account recovery functionality.

3. **Skill sharing post management**
    - Users can upload up to 3 photos or short videos (max 30 sec) with a description.
    - Users can view posts from other users in their feed. User receive reminders for meal plans.
    - Users can edit the description of their posts.
    - Users can remove their posts if needed.

4. **Meal plan and workout plan management**
    - Users can create, view, and modify personalized meal plans and work out plans.
    - The system suggests meal plans based on dietary preferences.
    - Users receive reminders for meal plans.

5. **Learning Progress Updates**
    - Users can post updates on their learning journey using predefined templates.
    - Users can browse progress updates from others.
    - Users can modify their progress updates if needed.
    - Users can remove progress updates if outdated.

6. **Post Engagement (Likes ,Comments and Notification Management)**
    - Users can like posts and add comments
    - Users can see who liked a post and read comments.
    - Users can edit their comments.
    - Users can remove their own comments, and post owners can delete comments on their posts.
    - Users can receive notifications when a post receives a like or comment.

2.2. Functional requirements for the REST API

1. User Authorization & Authentication

• Users must be able to sign up and log in securely
router.post('/users/register', (req, res) => {}); // User Registration
• Implement authentication using email/password and social logins
router.post('/auth/login', (req, res) => {}); // Login with JWT
• Role-based access control (Admin, User, etc.)
router.get('/auth/role', (req, res) => {}); // Get user role
• Password reset and account recovery functionality
router.post('/auth/reset-password', (req, res) => {}); // Password Reset

2. User Management

• Users can update their profiles
router.put('/users/:id', (req, res) => {}); // Update User Profile
• Admin can view, edit, and delete user accounts
router.get('/admin/users/:id', (req, res) => {}); // View User
router.put('/admin/users/:id', (req, res) => {}); // Edit User
router.delete('/admin/users/:id', (req, res) => {}); // Delete User
• Password reset and account recovery functionality
router.post('/auth/reset-password', (req, res) => {}); // Password Reset

3. Skill sharing post management

• Users can upload photos/videos and descriptions
router.post('/posts', (req, res) => {}); // Create Post
• Users can view posts
router.get('/posts', (req, res) => {}); // View Posts
• Users can edit their posts
router.put('/posts/:id', (req, res) => {}); // Edit Post
• Users can remove their posts
router.delete('/posts/:id', (req, res) => {}); // Delete Post

4. Meal plan and workout management

• Users can create, view, and modify meal/workout plans
router.post('/meal-plan', (req, res) => {}); // Create Meal Plan
router.put('/meal-plan/:id', (req, res) => {}); // Modify Meal Plan
router.get('/meal-plan/:id', (req, res) => {}); // View Meal Plan
• System suggests meal plans based on dietary preferences
router.get('/meal-plan/suggestions', (req, res) => {}); // Meal Plan Suggestions
• Users can receive reminders for meal plans
router.post('/meal-plan/reminder', (req, res) => {}); // Set Meal Plan Reminder

5. Learning progress update

• Users can post updates on their learning journey
router.post('/learning-progress', (req, res) => {}); // Post Learning Progress
• Users can browse progress updates from others
router.get('/learning-progress', (req, res) => {}); // View Learning Progress
• Users can modify their progress updates
router.put('/learning-progress/:id', (req, res) => {}); // Edit Learning Progress
• Users can remove progress updates
router.delete('/learning-progress/:id', (req, res) => {}); // Delete Learning Progress

6. Post engagement (Likes, Comments and Notifications Management)

- • Users can like posts
  router.post('/posts/:id/like', (req, res) => {}); // Like Post
  • Users can add comments
  router.post('/posts/:id/comment', (req, res) => {}); // Add Comment
  • Users can edit their comments
  router.put('/posts/:id/comment/:commentId', (req, res) => {}); // Edit Comment
  • Users can remove their comments
  router.delete('/posts/:id/comment/:commentId', (req, res) => {}); // Delete Comment
  • Users can receive notifications when a post receives a like or comment
  router.get('/notifications', (req, res) => {}); // Get Notifications
  .

7. General API Requirements

- All responses should be in JSON format.
- Error handling with appropriate status codes (e.g., 400 Bad Request, 404 Not Found).
- Pagination for listing endpoints.
- Secure API with rate limiting and input validation

3. Non-Functional Requirements

3.1. Non-Functional Requirements for the Client Web Application

1. **Usability:**
    - Ensure a user-friendly interface suitable for non-technical users.

2. **Performance:**
    - Minimize latency for swift user interactions.

3. **Responsiveness**
    - The platform is easily accessible from PCs, laptops, tablets, and smartphones, increasing user happiness and accessibility.

4. **Accessibility:**
    - Accessibility ensures that all users, including those with disabilities, can access and use the platform effectively, promoting inclusivity and compliance

5. **Scalability:**
    - Ensure the application can handle increased traffic without performance degradation.

6. **Security:**
    - Adhere to strict security measures to prevent unauthorized access.

7. **Compatibility:**
    - Ensure compatibility with various hardware, software, and browsers.

8. **Documentation:**
    - By facilitating user adoption and developer cooperation, documentation makes it possible for developers to easily interact with the REST API and for users to use the platform to its full potential.

9. **Reliability:**
    - Maintain operational reliability with minimal downtime.

3.2. Non-Functional Requirements for the REST API

1. **Performance:**
    - Make sure that requests and answers are handled effectively, and that you can scale as needed.

2. **Reliability:**
    - Ensure constant availability and response.

3. **Scalability:**
    - Control growing loads without compromising dependability

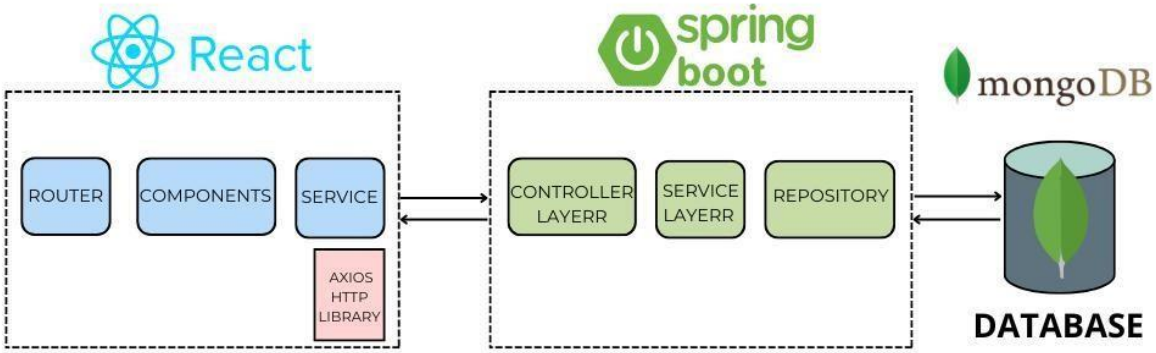
4. **Usability:**
    - To make things easier to use, include explicit documentation and error management.

5. **Interoperability:**
    - Respect industry norms to ensure interoperability across various technologies.

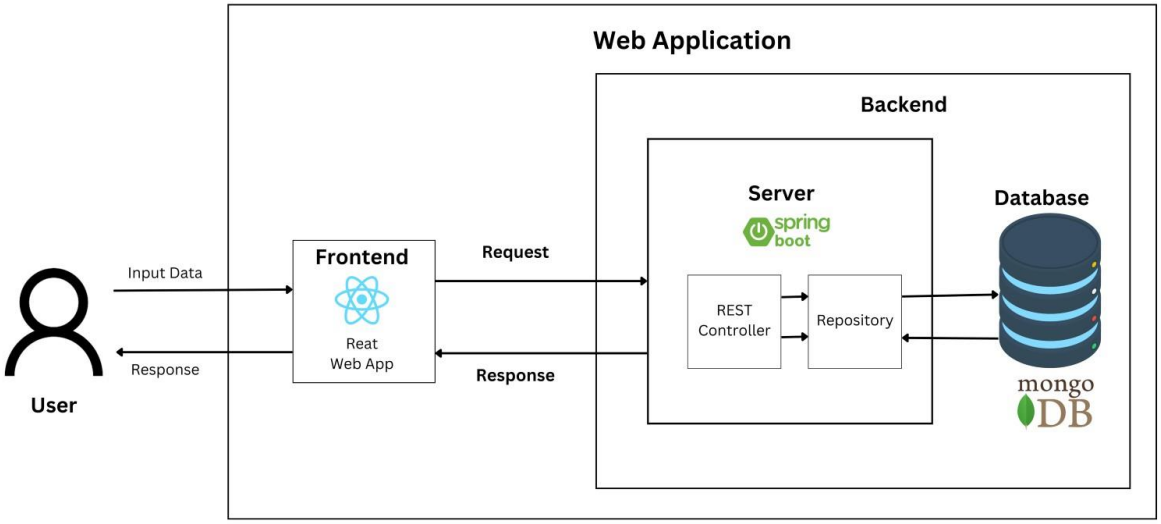
6. **Extensibility:**
    - Create the API with future mobile application updates in mind.

7. **Compliance:**
    - Make sure all applicable laws and requirements are followed

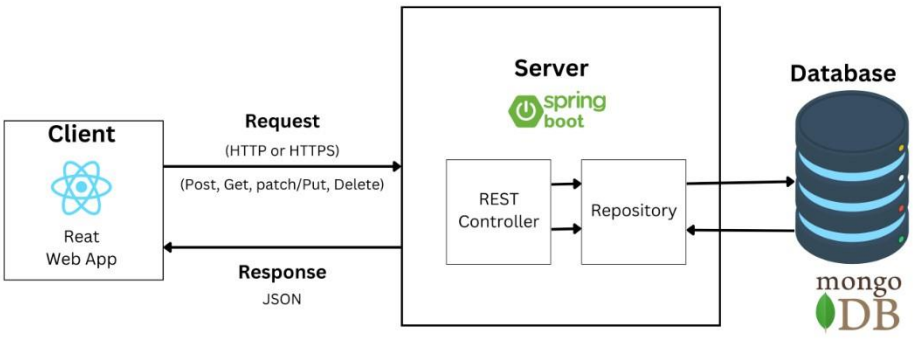
4. Overall architecture diagram for the entire system



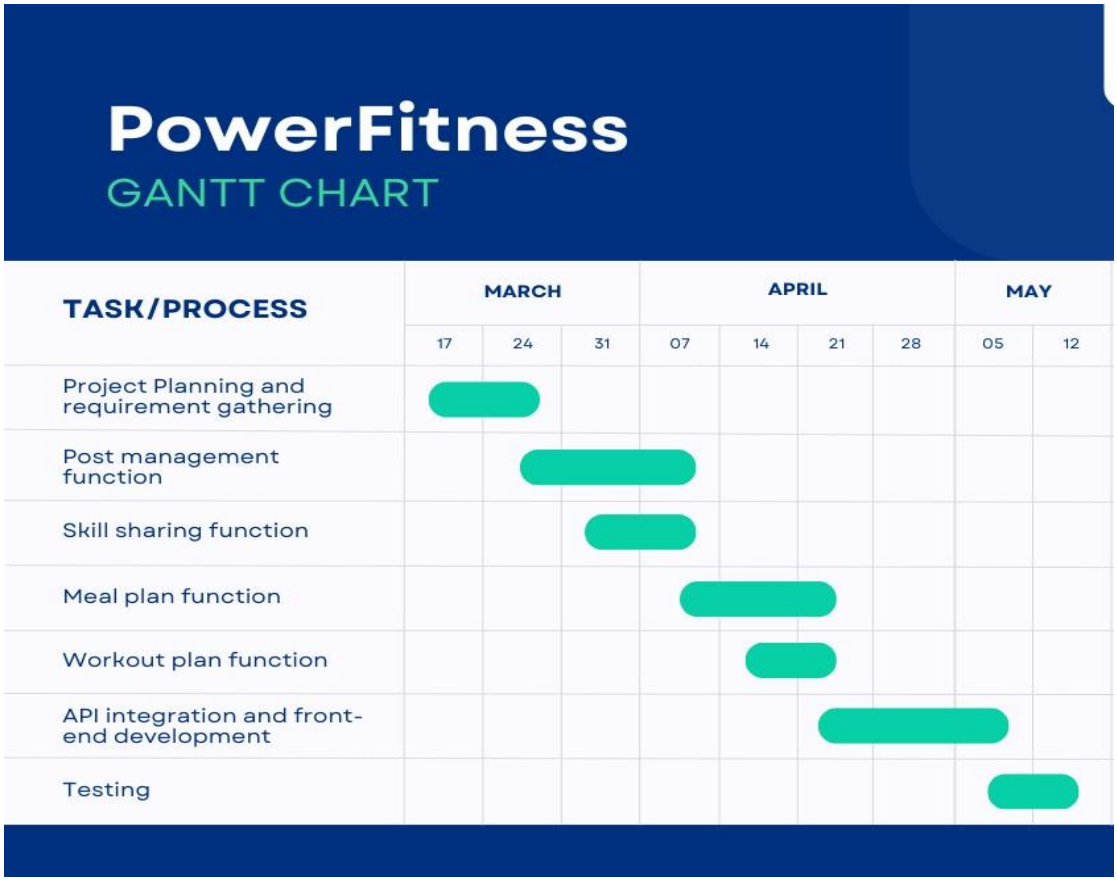
5. Architecture Diagram for the client web application



6. Architecture diagram for the REST API



7. Gantt Chart



References

1. **“Spring Security Tutorial,” [Online]. Available:** Getting Started | Securing a Web Application
2. **“Building REST Services with Spring,” [Online]. Available:** Spring | Spring
3. **What is the Rest API?**
   What is a REST API? | IBM
4. **How to draw an architecture Diagram?**
   Diagramming System Architecture in 5 Simple Steps | Tandem