Sri Lanka Institute of Information Technology

Programming Applications and Frameworks (IT3030)

Continuous Assignment – 2025, Semester 2

Initial Document

Group ID: DS-63

ID Number	Name
IT22363770	A.D.Oshadhi Vibodha
IT22026484	Ekanayake S D S
IT22026620	Kabisek S
IT22029690	T SANJAYAN

Table of Contents

Contents

Project Description 3
Functional Requirements 4
Non-Functional Requirements 6
Designs 8
Overall System Architecture Diagram 8
Detailed REST API architecture diagram. 9
Detailed front-end architecture diagram. 10
Work Distribution 11
Gantt Chart 12

Project Description

LearnSpark is a web application that designed to facilitate learning and sharing of skills, track progress, and interact with others. It offers a collaborative learning space where users can exchange posts, plans, and educational tips on various topics. Users can upload updates, images, short videos, and articles, and engage with others through likes, comments, and following features. The platform also includes community groups, messaging, interactive learning challenges, and progress tracking to enhance engagement and motivation.

Key features include skill-sharing posts, learning progress updates, learning plan creation, interaction, real-time messaging(chatbot), and notifications. The technology stack includes Java Spring Boot (REST API), React.js, MongoDB, JWT (JSON Web Tokens) for secure access control, and WebSocket for instant messaging and notifications.

The development process includes planning, designing, developing both frontend and backend, integrating third-party services, testing, deployment, and ongoing maintenance. LearnSpark aims to create an engaging, user-friendly, and interactive platform that fosters a community-driven learning environment, supports self-improvement, and provides the necessary tools to help users achieve their learning goals effectively.

Functional Requirements

2.1. Functional Requirements for the Client Web Application

1. **User Account Management:**
    - Users can use the standard registration form to sign up and log in.
    - React Router is used to control navigation between authentication pages.
    - For authentication procedures, Axios will manage communication with the backend.
    - Users can update and share about their learning progress in their profiles.
    - Users will be able to sign up and log in with ease through Google Authentication.
2. **Collaborative Learning Plan Management:**
    - Users can create, modify, and share structured learning plans.
    - Topics, instructional materials, and due dates are all included in learning plans (To do list).
3. **Content Sharing and Media Upload:**
    - Up to three photos or brief videos (not more than 30 seconds) can be uploaded by users for each post to convey instructional content.
    - A description will be included with every post to give background information on the shared resources.
4. **Learning Progress Tracking:**
    - By sharing updates on their progress, users can record their educational journey.
    - Users can format updates on finished tutorials and newly learned abilities with the aid of predefined templates.
5. **Community-Driven Feedback and Engagement:**
    - By liking and commenting on posts, users can engage with them.
    - The author can edit or remove comments.
    - Unwanted comments on a post can be removed by the post owner.
    - Users will be able to comment on learning plans through a review and rating system.

6. **Version Control for Learning Plans:**
    - A version history feature will be used to track modifications to learning plans.
    - If needed, users can restore earlier upgrades and view earlier iterations.
7. **User Profiles and Social Features:**
    - Each and every user gets a profile page that showcases their learning activities and contributed.
    - To customize their feed to their favorite material, users can follow and unfollow other users.
    - Public access to profiles encourages cooperation and exchange of information.
8. **Notifications:**
    - Users will receive notifications when their posts receive likes or comments.
    - Notifications will also alert users to updates on learning plans they are following.

2.2. Functional Requirements for the REST API

1. **Comprehensive CRUD Operations:**
    - The API will make it easier to Create, Read, Update, and Delete posts, learning plans, user accounts, and progress reports.
2. **Standard HTTP Methods:**
    - To manage different resources, the system will support the GET, POST, PUT, and DELETE methods.
3. **Robust Error Handling:**
    - For incorrect requests or unexpected issues, the API will clearly display error messages.
4. **Secure Authentication and Authorization:**
    - Open Authorization authentication will ensure secure access control.
    - Depending on user responsibilities and permissions, access to the API will be limited.
5. **Versioning and Hypermedia Support (HATEOAS):**
    - Updates to the learning plan will be version-controlled in the API.
    - Links to relevant sites will be included in API answers to enhance discoverability and effectiveness.

Non-Functional Requirements

3.1. For the Client Web Application

1. **Security:**
    - Implement strong safeguards in place to protect user data and communications, such as input validation, HTTPS, and XSS/CSRF protection.
2. **Reliability:**
    - Maintain consistent user experiences by implementing efficient error handling and ensuring high availability with less downtime.
3. **Performance:**
    - Make sure that responsiveness and resource loading are optimized for quick and seamless interactions.
4. **Usability:**
    - Create an interface that is easy to use and understand for both technical and non-technical users.
5. **Scalability:**
    - Construct the system to effectively handle increasing user counts and data quantities without seeing a decline in performance.
6. **Compatibility:**
    - Ensure seamless functioning on a range of hardware, operating systems, and widely used web browsers.
7. **Accessibility:**
    - Follow accepted accessibility guidelines (such as WCAG) to guarantee that people with disabilities can use the application.

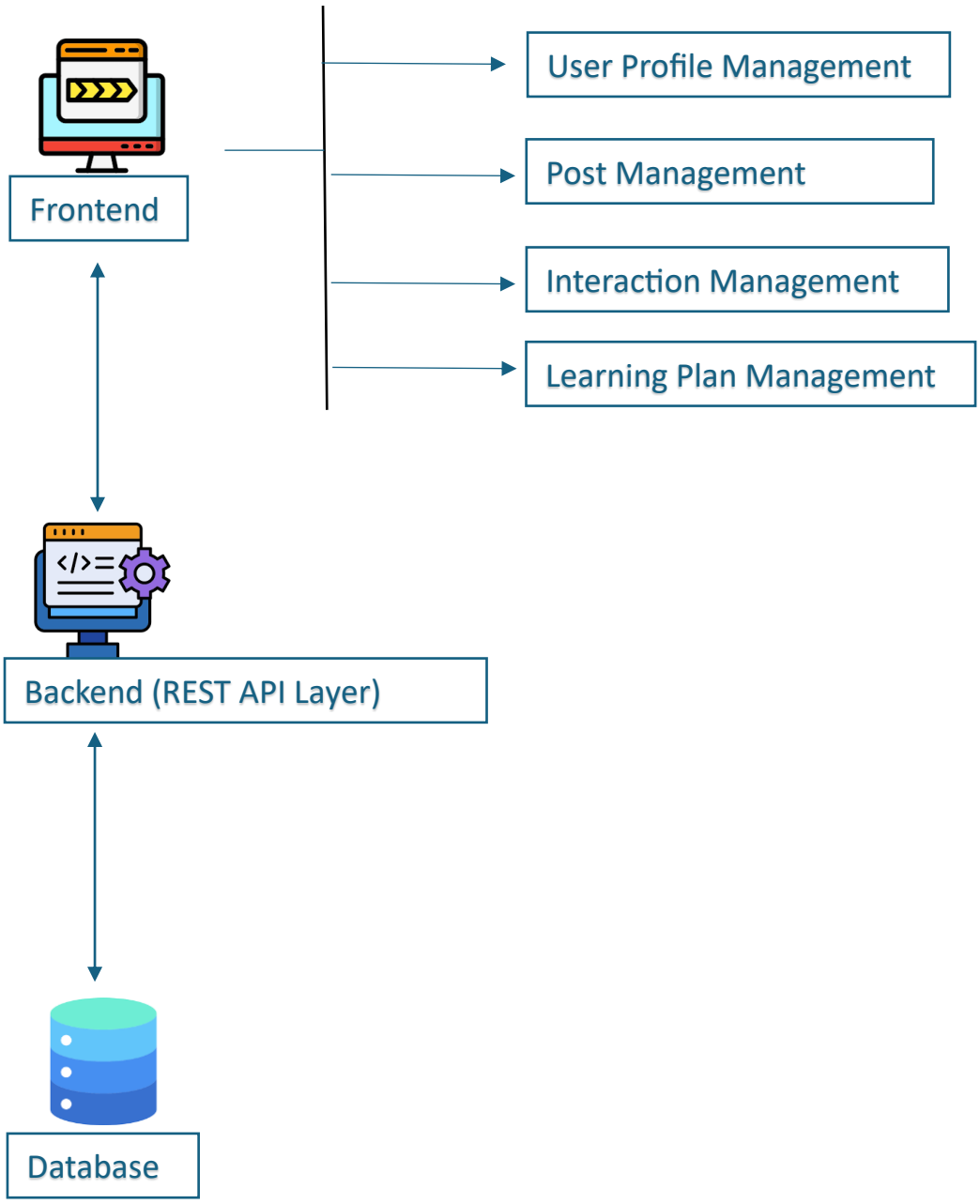
3.2. For the REST API

1. **Security:**
    - Using protocols like Open Authorization, secure API endpoints enforce strict permission and encrypt data while it's in transit.
2. **Performance:**
    - Utilize strategies like caching, effective query design, and indexing to optimize the API to manage large request volumes with minimal latency.
3. **Reliability:**
    - Use strong error handling, rate limitation, and redundancy to maintain steady uptime and responsiveness.

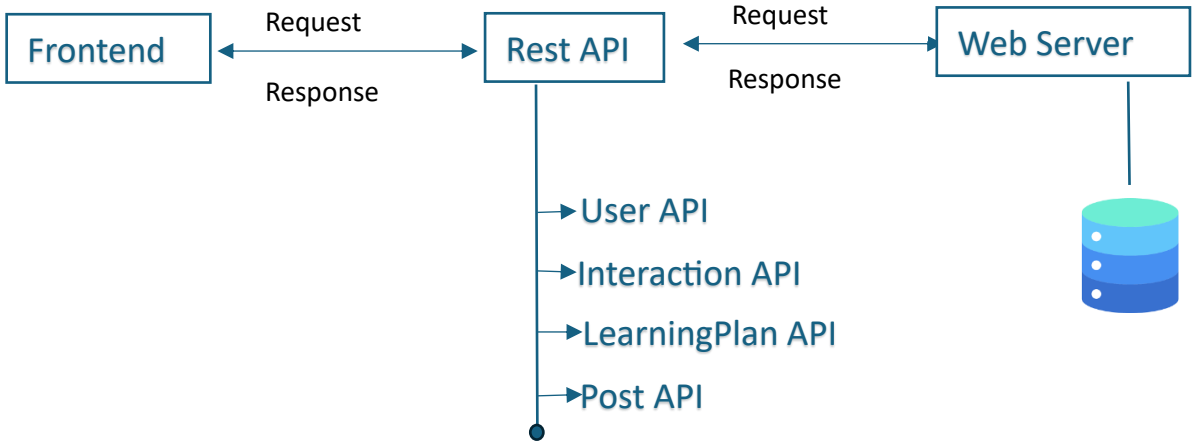
4. **Scalability:**
    - Without losing performance, build the infrastructure to handle growing loads and data volumes with ease.
5. **Compliance:**
    - Make sure that all data processing and handling abide with industry, legal, and regulatory standards (such as the GDPR).
6. **Interoperability:**
    - Stick to industry standards (such JSON formatting and RESTful principles) to facilitate easy connection with external systems.
7. **Developer Usability:**
    - To make developer interactions with the API easier, provide thorough error messages, integration instructions, and clear, detailed documentation.
8. **Extensibility:**
    - Without facing major change, design the API to support upcoming improvements and new features.

Designs

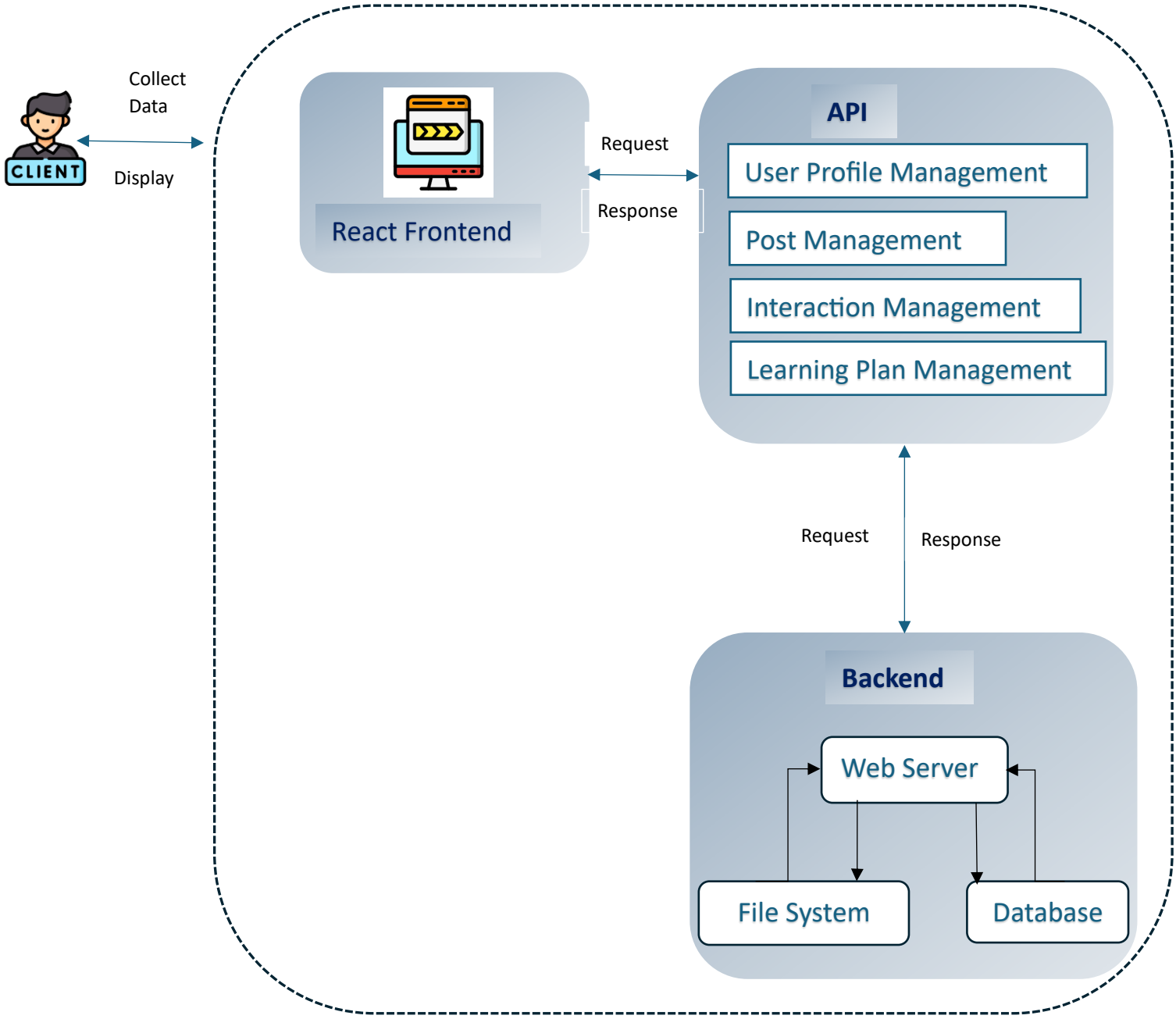
Overall System Architecture Diagram



Detailed REST API architecture diagram.



Detailed front-end architecture diagram.



Work Distribution

IT Number	Name	Work Allocated
IT22363770	A.D.Oshadhi Vibodha	• **Interaction Model:** Build models for user interactions, likes, comments, follows, and other actions. • **Interaction Repository:** Put in place the database's interaction data management repository. • **Interaction Controller:** Create the API endpoints for the Interaction Controller to manage user interactions, such as content, following users, and retrieving interactions.
IT22026484	Ekanayake S D S	• **Learning Plan Entity:** Specify the format of learning plans, including sections for subjects, materials, progress monitoring, and due dates. • **Learning Plan Repository:** Manage learning plans in the database by putting CRUD operations into practice. • **Learning Plan Controller:** Provide API endpoints that allow users to add, edit, remove, and retrieve lesson plans. • **Learning Plan Management Service Application:** To launch and establish the Learning Plan Management Service, set up the Spring Boot application.
IT22026620	Kabisek S	• **Post Entity:** Establish the framework of posts, comprising elements such as description, location, image path, hashtags, and user ID. • **Post Repository:** Set up the required CRUD operations to manage posts in the database. • **Post Controller:** Establish REST API endpoints for generating, retrieving, modifying, and removing posts, in addition to obtaining user-specific posts. • **Post Management Service Application:** Set up the Spring Boot application to execute the Post Management Service.
IT22029690	T Sanjayan	• **User Entity:** Specify the format of user profiles, incorporating necessary data such as roles, email, password, and username. • **User Repository:** To handle user data, develop CRUD operations techniques. • **User Controller:** Configure API endpoints for profile management, login,

		user registration, and user-specific operations like account deletion or password changes.

Gantt Chart