Programming Applications and Frameworks (IT3030)
Assignment – 2025, Semester 1

Important Details

- This assignment carries **30%** of the final mark for the IT3030 module.
- This assessment is to be carried out as **group work**, but each member's contribution will be assessed individually. Marks may differ among group members based on their performance.
- **Assignment release date:** 11th March 2025
- **Submission deadline:** 11.45 PM, 02nd May 2025 (GMT + 5.30) via Courseweb

Assignment Description

The goal of this assignment is to **design and implement:**

1. A **Java (Spring Boot) REST API** for the given business scenario using best practices.
2. A **React-based client web application** that allows users to interact with the system.

Business Scenario

Your team has been hired to develop a **Skill-Sharing & Learning Platform**, where users can **share and learn different skills** like coding, cooking, photography, and DIY crafts. The platform should provide the following functionalities:

Features:

- **Skill Sharing Posts:**
  - Users can upload up to **3 photos or short videos (max: 30 sec)** per post.
  - Allow users to add **descriptions** for their shared content.
- **Learning Progress Updates:**
  - Users can post updates on their learning journey.
  - Predefined templates to help users input their progress (e.g., completed tutorials, new skills learned).
- **Learning Plan Sharing:**
  - Users can create structured **learning plans**, including topics, resources, and completion timelines.
  - Plans can be updated as users progress.
- **Interactivity & Engagement:**
  - Users can **like** and **comment** on others' posts.
  - Comments can be edited or deleted by the user.
  - Post owners can delete comments on their own posts.

- **User Profiles & Social Features:**
  - Each user has a **profile page** displaying their skill-sharing posts and activities.
  - Users can **follow other users** to see their posts.
  - Profiles are **publicly visible** to encourage interaction.
- **Notifications:**
  - Users receive **notifications** for likes and comments on their posts.
- **Authentication:**
  - Users can log in using their **existing social media accounts** (OAuth 2.0).

Tasks in the Assignment

1. Identify:

- **Functional requirements** for the REST API and the client web application.
- **Non-functional requirements** (e.g., security, scalability, performance).

2. Architecture Design:

- **Overall system architecture diagram** (excluding mobile applications).
- **Detailed REST API architecture diagram.**
- **Detailed front-end architecture diagram.**

3. Development & Implementation:

- Develop the **REST API** using **Spring Boot** and ensure it follows RESTful principles.
- Develop the **React client web application** for user interaction.

4. Testing & Validation:

- Verify that the **REST API works independently and with the client application** as per the requirements.
- Ensure the **client application functions correctly** with good UI/UX.

Technical Requirements

1. **Spring Boot** for the REST API implementation.
2. **React** for the client web application.
3. **Spring Security & OAuth 2.0** for authentication.

Other Requirements

1. The project must be **version-controlled** using Git and hosted on GitHub with **GitHub Workflow**.
2. Teams have creative freedom in designing the system, ensuring good **UI/UX**, security, and maintainability.
3. Each member must create at least **four (4) REST API endpoints** with different HTTP methods (GET, POST, PUT, DELETE).

Marks Breakdown

Documentation (15 Marks)

- Initial document (17th March) **5 Marks**
- Final documentation (02nd May) **10 Marks**

In-Class Progress Review (10 Marks)

- Progress review (one week starting 24th April) **10 Marks**

REST API (30 Marks)

- Proper endpoint naming **5 Marks**
- Follows REST architectural principles **10 Marks**
- Correct HTTP methods and status codes **5 Marks**
- Code quality (Java/Spring conventions) **5 Marks**
- Satisfying all requirements **5 Marks**

Client Web Application (15 Marks)

- Proper architectural design **5 Marks**
- Satisfying all requirements **5 Marks**
- Good UI/UX **5 Marks**

Version Control (10 Marks)

- Proper Git usage (commits, branching, etc.) **5 Marks**
- GitHub Workflow implementation **5 Marks**

Authentication (10 Marks)

- OAuth 2.0 authentication implementation **10 Marks**

Creativity (10 Marks)

- Unique features, additional enhancements **10 Marks**

Total: 100 Marks

Special Notes

- Academic integrity and honesty are strictly required.
- The assignment tests the ability to build a modern web application with best practices.
- Each team can divide work among the members, but individual grading will be applied.
- **AI-generated code (Gemini, ChatGPT, etc.) is allowed, but usage must be disclosed in documentation and progress reviews.**
- Submissions must be made as a **.zip file** containing the final report, source code, and documentation.
- **Submission deadline:** 11.45 PM, 02nd May 2025.

Marking Rubric:

Criteria	Excellent	Good	Needs Improvement	Not Acceptable
DOCUMENTATION (15 MARKS)				
Initial Document (5 Marks | Grp)	Deep understanding of the task with a clear WBS **(5)**	Solid understanding of the task but not with a clear WBS **(3-4)**	Basic understanding of the task **(1-2)**	Key concepts may be misunderstood or under-explained **(0)**
Final Document (10 Marks | Grp)	Clear, logical flow with well-structured sections **(8-10)**	Generally well-organized with minor issues but could be improved **(5-7)**	Sections are present but may be poorly structured **(1-4)**	Content is largely irrelevant, and not structured **(0)**

IN-CLASS PROGRESS REVIEW (10 MARKS)				
Progress Review (50 – 75%) **(10 Marks | Ind)**	Web Services, and REST API with GitHub flow **(8-10)**	Web Services with GitHub flow **(5-7)**	Working on Web Services with GitHub flow **(1-4)**	Not or just started without Version Controlling **(0)**
REST API (30 MARKS)				
Proper Endpoint Naming **(5 Marks | Ind)**	Follows standard conventions (RESTful principles), meaningful, and consistent naming (e.g., /users/{id}, /orders/{id} for resources) **(5)**	Mostly follows proper conventions but with minor inconsistencies in naming **(3-4)**	Endpoint naming is inconsistent, lacks clarity, or does not fully follow RESTful principles **(1-2)**	Poor or no adherence to RESTful principles, unclear and ambiguous endpoint names **(0)**
Follows the Six REST Architectural Styles **(10 Marks | Ind)**	Fully adheres to all six REST architectural constraints (Client-Server, Stateless, Cacheable, Uniform Interface, Layered System, Code-on-Demand) **(8-10)**	Adheres to most REST constraints but has minor deviations **(5-7)**	Partially follows REST constraints but lacks key elements **(1-4)**	Does not follow REST principles or ignore major constraints **(0)**
Proper usage of HTTP methods and status codes **(5 Marks | Ind)**	Correct and consistent use of HTTP methods (GET, POST, PUT, DELETE) with appropriate status codes (200, 201, 204, 400, 404, etc.) **(5)**	Mostly correct, but with minor issues in HTTP method selection or status code usage **(3-4)**	Some incorrect HTTP methods or status codes used inconsistently **(1-2)**	HTTP methods and status codes are used incorrectly or not considered **(0)**
Good code quality following Java/ Spring coding conventions **(5 Marks | Ind)**	Code is clean, well-structured, follows Java and Spring best practices, with proper indentation, naming conventions, and documentation **(5)**	Mostly follows conventions, but minor issues in structure, naming, or documentation **(3-4)**	Some violations of Java/Spring coding standards, lacks readability and maintainability **(1-2)**	Poor code quality, does not follow Java/Spring conventions, difficult to read and maintain **(0)**

Satisfying all requirements (5 Marks | Ind)	Fully implements all specified API functionalities, including authentication, CRUD operations, and validations, ensuring seamless integration with the client **(5)**	Implements most functionalities but may have minor missing features or incomplete validation **(3-4)**	Partially satisfies the requirements but lacks key functionalities or has major issues in implementation **(1-2)**	Does not meet the API requirements, missing critical functionalities or entirely non-functional **(0)**
CLIENT WEB APPLICATION (15 MARKS)				
Proper Architectural Design and Implementation (5 Marks | Ind)	Well-structured architecture, modularized components, follows best practices in React development, ensuring maintainability and scalability **(5)**	Mostly well-structured but with minor architectural flaws or less modularization **(3-4)**	Basic structure implemented but lacks modularization, making it difficult to maintain **(1-2)**	Poorly structured or non-functional application, does not follow best practices **(0)**
Satisfying all Requirements (5 Marks | Ind)	Fully implements all required features, ensuring smooth functionality and seamless integration with the REST API **(5)**	Implements most features but may have minor missing functionalities or UI/UX inconsistencies **(3-4)**	Partially satisfies the requirements but lacks key features or has major usability issues **(1-2)**	Poorly Does not meet the application requirements, missing critical features or entirely non-functional **(0)**
Good UI/UX (5 Marks | Ind)	Excellent user interface design, visually appealing, intuitive layout, smooth navigation, and great user experience **(5)**	Good UI/UX but with minor inconsistencies in design, layout, or usability **(3-4)**	Basic UI/UX with several usability or aesthetic issues affecting the user experience **(1-2)**	Poor UI/UX, difficult to use, cluttered design, lacks visual appeal or usability considerations **(0)**

VERSION CONTROLLING (10 MARKS)				
Proper Usage of Git **(5 Marks | Grp)**	Uses Git effectively with meaningful commit messages, proper branching strategies, and collaborative workflows **(5)**	Mostly follows Git best practices but with minor inconsistencies in commits or branching **(3-4)**	Basic Git usage with occasional missing commit messages, poor branching structure **(1-2)**	Poor or no use of Git, lacks version control practices **(0)**
Proper Usage of the GitHub Workflow **(5 Marks | Grp)**	Fully utilizes GitHub Workflow for deployment with well-defined workflows **(5)**	Mostly uses GitHub Workflow effectively but may have minor deployment inefficiencies **(3-4)**	Basic use of GitHub Workflow for deployment or improper setup **(1-2)**	No implementation of GitHub Workflow **(0)**
AUTHENTICATION (10 MARKS)				
Implementing OAuth 2.0 Authentication **(10 Marks | Grp)**	Fully implements OAuth authentication, ensuring secure login with proper token handling, user roles, and session management **(8-10)**	Implements OAuth authentication but may have minor security or integration flaws **(5-7)**	Partial OAuth implementation with missing features or security concerns **(1-4)**	No OAuth authentication implemented, or it is non-functional **(0)**
INNOVATION/OUT OF THE BOX THINKING (10 MARKS)				
Overall Creativity **(10 Marks | Grp)**	Demonstrates unique and innovative features, enhancing user engagement and functionality beyond basic requirements **(8-10)**	Includes some creative elements but mostly follows standard implementations **(5-7)**	Limited creativity, minimal enhancements beyond the basic requirements **(1-4)**	No creativity, only implements basic requirements with no additional innovation **(0)**

End of the Assignment