CHAPTER 1

ABOUT THE COMPANY

1.1 Introduction of Company

TAP EdTech Pvt. Ltd. is a Bengaluru-based edtech company focused on training students in full-stack web development, aptitude, and communication skills. It offers hands-on, project-based learning that bridges the gap between academic knowledge and industry needs. Collaborating with colleges across India, TAP EdTech delivers scalable programs that boost employability and technical expertise. The company empowers learners with real-world skills to succeed in today’s tech-driven job market.

TAP EdTech fosters a dynamic learning environment through live coding sessions, hackathons, mock interviews, and personalized mentorship. The company continuously evolves its training methodologies to stay aligned with the latest industry trends and technologies, ensuring that learners gain not just theoretical knowledge but practical expertise. By combining technical training with a focus on confidence-building and career readiness, TAP EdTech empowers individuals to succeed in the competitive tech landscape and build fulfilling, future-proof careers.

1.2 Company Profile

Company Name:	TAP EdTech PVT LTD
Address:	4, Outer Ring Rd, near McDonald's, BTM 2nd Stage, Kuvempu Nagar, Stage 2, BTM Layout, Bengaluru, Karnataka 560076.
Telephone Number:	+91 8123464489
Email Address:	enquiry@thetapacademy.com
Type of Company:	Private Limited Company
Founded:	2022
Industry:	Full-Stack Web Development.
Products and services:	Full stack, Placement, E-learning, Projects, Hackathons, Mentorship.

1.3 Vision and Mission

- To bridge the gap between academic learning and industry requirements.
- To empower students with practical, job-ready tech skills.
- To deliver high-quality, mentor-led training in full-stack development and aptitude.
- To make career-oriented education accessible across India.
- To foster continuous learning through coding challenges and real-world projects.
- To stay updated with evolving industry trends and technologies.
- To create confident, skilled professionals ready for the tech workforce.

1.4 Awards and Recognition

TAP EdTech Pvt. Ltd. has been recognized as a leader in technology-driven education, receiving multiple prestigious awards for its innovative approach to skill development. The company was honored with the Industry Excellence Award 2024 and named Best EdTech Startup 2023 for its impactful training programs and commitment to personalized mentorship. TAP EdTech's dedication to quality education and learner success has also earned it the Innovation in Skill Development Award and the Customer Choice Award. Additionally, the company holds ISO 9001 certification, underscoring its adherence to high standards in educational service delivery.

Fig 1.1 Company Logo

Chapter 2

INTRODUCTION

2.1 Overview

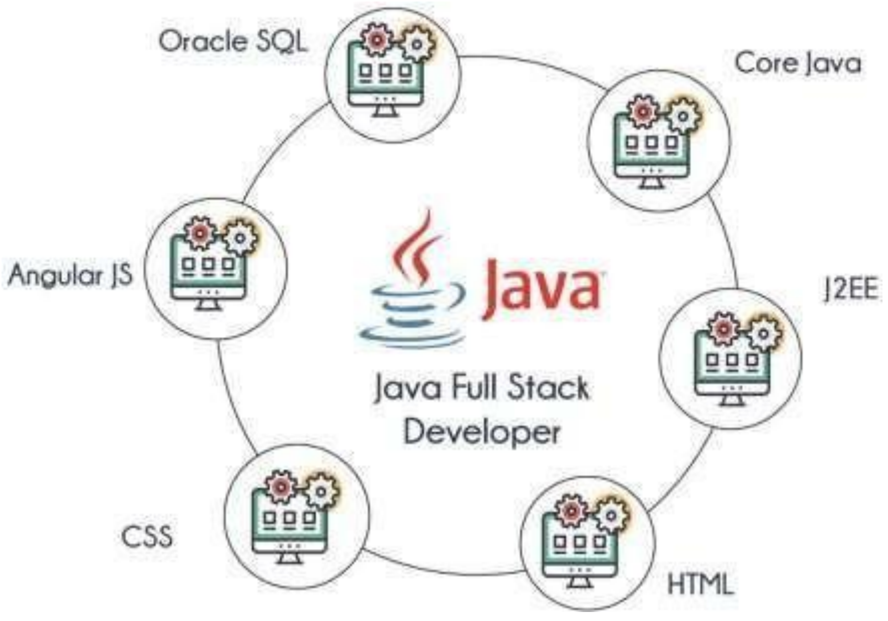


Fig2.1 Full Stack

Full-Stack Web Development represents the end-to-end process of designing, building, and deploying interactive web applications that seamlessly connect rich, browser-based front ends with robust, scalable back ends. At **TAP EdTech PVT LTD**, this discipline unites user-centric interface design—using **HTML**, **CSS**, **JavaScript**, and **React**—with server-side business logic and data management powered by **Java**, **Spring Framework**, and **Spring Boot,** and **SQL**. The program also emphasizes working with relational databases, where learners design schemas, write optimized SQL queries, and integrate database operations into backend services to build fully functional, data-driven web applications.

In this internship, I learned how modern web applications are structured as decoupled front-end and back-end services, using HTML, CSS, JavaScript, and React—with robust, scalable back-end development powered by Core Java, Advanced Java, Spring Framework, and Spring Boot. The curriculum is structured to reflect modern development practices, where applications are built as decoupled front-end and back-end services communicating through RESTful APIs.

This combined approach not only fosters a clear separation of concerns-Interns gain hands-on experience building responsive UIs with React, managing state, routing, and creating reusable components. On the server side, they model entities, create service layers, and expose secure APIs using Spring Boot, while implementing concepts such as dependency injection, auto-configuration, and Spring Security for authentication and authorization. The program also emphasizes with relational databases using **SQL**, enabling students to design schemas, write queries, and integrate database operations with Java applications using JDBC and Hibernate.

This internship provided a comprehensive learning experience by blending core concepts of web development. It offered valuable insights into building intelligent web applications from the ground up—starting from frontend design to backend logic and smart model integration. By working on real- time use cases, this internship also helped strengthen problem-solving, debugging, and deployment skills, which are essential for developing professional-grade, web solutions.

2.2 Introduction to Front-End Technologies

HTML (Hypertext Markup Language) forms the structural skeleton of every web page, defining elements like headings, paragraphs, forms, and navigation. **CSS** (Cascading Style Sheets) then styles these elements—controlling layouts, typography, colors, and responsive breakpoints so interfaces look polished across devices. JavaScript, the language of the browser, brings pages to life by manipulating the DOM, handling user events, and performing asynchronous data fetches. Building on vanilla JavaScript, React provides a component-based framework that streamlines UI development. By breaking the interface into reusable components (each with its own props and state), React promotes cleaner code organization and easier maintenance.

2.3 Introduction to Back-End Technologies

On the server side, **Java** remains a rock-solid, type-safe language for enterprise-grade applications. The Spring Framework—and its convention-over-configuration extension, **Spring Boot**—offers rapid development of production-ready services with minimal boilerplate. Key concepts I explored include:

1. **Spring Boot** Auto-Configuration: Automatically wiring up common components (data sources, web servers) based on dependencies.
2. **Dependency Injection**: Decoupling components via annotations (**@Component**, **@Service**, **@Autowired**) to improve testability and modularity.
3. **REST Controllers**: Defining **@RestController** classes with **@GetMapping**, **@PostMapping**, etc., to expose JSON endpoints.
4. **JPA** & **Hibernate**: Mapping Java entities to relational tables, writing repository interfaces for CRUD operations, and using JPQL for custom queries.
5. **Spring Security** & **JWT**: Securing **APIs** by configuring authentication filters, generating JSON Web Tokens on login, and validating them on subsequent requests.

2.4 Full-Stack Workflow and Best Practices

During the internship, I followed an Agile-inspired workflow:

1. Requirement Analysis & Design: Breaking down client stories into front-end and back-end tasks, sketching component hierarchies, and drafting API contracts with OpenAPI.
2. Parallel Development: Working on UI mockups and prototype components while back-end services were scaffolded, then integrating both strands via API calls.
3. Testing: Writing unit tests for Java services (using JUnit) and component tests for React (using Jest), plus end-to-end smoke tests.
4. Version Control & CI/CD: Collaborating through Git branches, performing code reviews, and seeing automated builds pass on every pull request.
5. Deployment & Monitoring: Deploying services to a development server, examining logs and performance metrics, and iterating on feedback.

2.5 Libraries and Frameworks

In this internship, I leveraged a range of front-end and back-end libraries to accelerate development, enforce best practices, and ensure maintainability across the full-stack web application.

1. **React:** A component-based UI library that allows declarative building of interactive interfaces with JSX, reusable components, and a virtual DOM for efficient rendering.
2. **React Router:** Enables client-side routing in single-page applications, managing URL history and dynamic route matching without full page reloads.
3. **Spring Boot Starters:** Opinionated dependency descriptors that automatically configure common components—such as web servers (spring-boot-starter-web), data access (spring-boot-starter-data-jpa), and security (spring-boot-starter-security).
4. **Spring Data JPA:** Simplifies database interactions by providing repository abstractions over JPA and Hibernate, enabling CRUD operations and query derivation from method names.
5. **Spring Security:** Offers a robust framework for authentication and authorization, including support for JWT, OAuth2, and method-level security annotations.

2.6 Introduction to Java and Spring Boot

In this internship, Java served as the primary language for back-end development, prized for its stability, strong typing, and extensive ecosystem. Originally released by Sun Microsystems in 1995, Java has become a cornerstone of enterprise software due to its "write once, run anywhere" philosophy, which is made possible by the Java Virtual Machine (JVM). Its object-oriented design enforces clear abstractions—classes, interfaces, and inheritance—that help organize complex business logic into maintainable modules.

Building on Java, Spring Boot dramatically accelerates the creation of production-ready RESTful services by embedding an application server (like Tomcat) and providing sensible defaults for common dependencies. Spring Boot's auto-configuration mechanism inspects your class path and beans to wire up components automatically, which means you can focus on writing business code instead of boilerplate setup. Key features include:

1. Starter Dependencies (spring-boot-starter-web, spring-boot-starter-data-jpa, etc.) that pull in and configure common libraries in one go.
2. Embedded Server support, so applications can be run with a simple java -jar command.
3. Actuator Endpoints for health checks, metrics, and environment information—vital for monitoring and production readiness.
4. Externalized Configuration via application. Properties or application, allowing separate profiles (dev, test, prod) and secure management of secrets.
5. Define REST Controllers—annotating classes with @RestController and mapping HTTP verbs (@GetMapping, @PostMapping) to Java methods that handle client requests and return JSON responses.
6. Model Entities and Repositories—creating Java classes annotated with JPA annotations (@Entity, @Id, @OneToMany) and interfaces extending JpaRepository to perform CRUD operations without writing SQL.
7. Implement Service Layers—encapsulating business rules in @Service classes, injected into controllers to keep concerns separated and support unit testing.

2.7 Introduction to Web Development

HTML (Hypertext Markup Language)

HTML is the fundamental language used to structure and display content on the web. It defines elements such as headings, paragraphs, links, lists, and multimedia content like images and videos. HTML uses tags, such as <h1> for headings, <p> for paragraphs, and <a> for links, to create a web page’s structure. The latest version, HTML5, brought many new features like native support for audio and video, canvas for drawing graphics, and new APIs for modern web applications. HTML serves as the skeleton of every webpage, providing essential structure and semantics that browsers use to render content. As the backbone of web development, HTML is essential for defining the content layout and ensuring accessibility across devices.

CSS (Cascading Style Sheets)

CSS is a style sheet language that defines how the HTML elements should be displayed. It allows developers to set visual styles for elements, such as colors, fonts, margins, padding, and positioning. CSS enables responsive web design by making pages adapt to different screen sizes, using media queries to change layouts based on device characteristics. With the evolution of CSS, features like Flexbox and Grid have simplified the process of creating complex, responsive layouts. Advanced CSS techniques include animations, transitions, and transformations, adding interactivity and visual flair to websites. CSS can be external, internal, or inline, with external stylesheets being the most efficient method for managing styles across multiple pages. It plays a crucial role in the aesthetic appeal of a website, enhancing user experience.

JAVA SCRIPT

JavaScript is a high-level programming language that brings interactivity and dynamic behavior to web pages. It allows developers to create features like form validation, interactive maps, real-time updates, and animations without reloading the page. JavaScript runs in the browser, enabling fast and seamless interactions with users. With the advent of libraries and frameworks like React, Vue.js, and Angular, JavaScript has become the cornerstone of modern web development, allowing the creation of single-page applications (SPAs) and complex user interfaces. It also works server-side via Node.js, enabling full-stack development. JavaScript can be used for everything from creating interactive games to building complex data-driven applications. Its versatility and growing ecosystem make it an indispensable tool in web development.

Fig 2.2 Web Development Logo

2.8 Introduction to MySQL

Fig 2.3 SQL

As a Full-Stack Web Development intern at TAP Academy, I gained hands-on experience on structured Query Language (SQL) is a standard programming language used for managing and manipulating relational databases. It allows users to create, read, update, and delete data (CRUD operations) stored in a structured format. SQL is essential in backend development, as it enables efficient communication between applications and databases. Through commands like SELECT, INSERT, UPDATE, and DELETE, developers can retrieve and modify data based on specific conditions. SQL also supports advanced features such as joins, indexing, transactions, and views, which help in organizing and optimizing data access. In full-stack development, SQL plays a crucial role in building data-driven applications by ensuring reliable and secure data storage and retrieval.

Role of SQL in Full Stack Web Development

- **Data Management:** SQL allows developers to store, retrieve, update, and delete data efficiently in relational databases, which is essential for building dynamic web applications.
- **Backend Integration:** It works seamlessly with backend technologies like Java and Spring Boot, enabling smooth interaction between the application logic and the database.
- **Dynamic Functionality:** SQL helps provide personalized and real-time content by handling user inputs, queries, and transactions behind the scenes.
- **Data Relationships and Structure:** SQL supports relational database design, allowing developers to create structured data models with proper relationships between tables.
- **Performance and Security:** SQL features like indexing, filtering, user access control, and transactions ensure secure and high-performing applications.

Chapter 3

TASKS PERFORMED

Learn Sphere is an innovative, technology-driven learning platform designed to enhance and personalize the educational experience for students and professionals. The project aims to create an interactive ecosystem where learners can access a wide range of courses, learning materials, and skill-building tools, all tailored to their individual needs and pace

3.1 Goals and Scope of the Project

The goal of Learn Sphere is to provide a user-friendly, secure, and scalable platform for both trainers and students. Trainers can create and manage their courses, while students can browse, purchase, and interact with the learning materials.

1. Trainer Functionalities: Create courses and lessons.
2. Upload various learning materials (e.g., videos, quizzes).
3. Monitor student progress.
4. Student Functionalities: rowse available courses.
5. Enroll in courses and make payments. Access course materials and track progress.

3.2 Technologies Used

The success of Learn Sphere relies heavily on the choice of technologies that form the backbone of the platform. Each technology was carefully selected based on its ability to meet specific project requirements, such as scalability, security, performance, and ease of use. Below, we will provide a detailed explanation of the core technologies used in this project.

- Spring Boot (Backend Framework)
- Spring Boot is the chosen backend framework for Learn Sphere, and it plays a vital role in simplifying the development process of web applications. Built on top of the Spring Framework, it helps developers create stand-alone, production-ready applications with minimal setup and configuration.

Why Spring Boot: Simplicity and Efficiency: One of the key features of Spring Boot is its ability to significantly reduce the amount of boilerplate code required to set up a Spring application. With Spring Boot, you don't need to worry about configuring servers, databases, or other components manually. Instead, it offers auto-configuration, so the application is ready to run with minimal setup. o Integrated Development: Spring Boot integrates well with the Spring Ecosystem, making it easier to incorporate features like Spring Security, Spring Data JPA, and Spring MVC. It also integrates smoothly with Thyme leaf for rendering dynamic HTML content and managing the view layer. Microservice Architecture Support: Though Learn Sphere is not currently a microservice-based platform, Spring Boot’s inherent flexibility allows it to scale to a microservices architecture in the future. This makes it easy for the platform to handle growing demand as it expands and needs to manage a larger number of users and services. Why It Was Chosen for Learn Sphere: Given Learn Sphere's need for rapid development, maintainability, and scalability, Spring Boot was the ideal choice for the backend. It supports building robust APIs, handles user authentication with Spring Security, and integrates with MySQL to manage relational data efficiently.

2. MySQL (Database Management System)

MySQL is used as the relational database management system (RDBMS) for Learn Sphere. It is an open-source RDBMS known for its reliability, speed, and ease of use. MySQL stores all critical data related to users, courses, lessons, payments, and student progress.

Why MySQL?

Structured Data Storage: Since Learn Sphere deals with structured data, such as user profiles, courses, lessons, and payment history, a relational database is the best choice. MySQL provides robust support for relational data modeling, ensuring that the data is organized and easy to manage.

1. **Performance:** MySQL is highly optimized for read and write operations, making it perfect for applications like Learn Sphere, where real-time data access and transaction management are critical. It ensures that data queries for course content and user activity are processed efficiently, even with large datasets**.**

2. **Scalability:** MySQL's ability to scale with increasing data volumes makes it suitable for future growth. As Learn Sphere expands, MySQL can handle the growing data needs of students, trainers, and courses.

3. **Why It Was Chosen for Learn Sphere:** MySQL was chosen because it aligns perfectly with the requirements of Learn Sphere. It offers strong transactional support for handling payments via Razorpay and ensures the relational integrity of data as students enroll in courses, interact with content, and track their progress.

1. **Input Features**:

The dataset includes features such as Age, Duration, Frequency, Location, Character, Intensity, Nausea, Vomit, Phonophobia, Photophobia, and more—used to describe patient symptoms.

2. **Output**:

   The dataset does **not contain an explicit class column**, so you may need to add or define the output labels (e.g., Migraine Type) based on available data or medical categorization.

3. **Dataset Size and Structure**:

   The dataset contains **400 rows** and **24 columns**, representing patient records and their respective symptom details.

3. Razorpay (Payment Gateway Integration)

Razorpay is an online payment gateway that allows businesses to accept, process, and disburse payments with ease. It is integrated into Learn Sphere to handle secure course payments and subscription fees.

1. Why Razorpay?

- Ease of Integration: Razorpay provides easy-to-use APIs and SDKs for seamless integration into web applications. Its comprehensive documentation and developer friendly tools make the integration process fast and simple.
- Secure Transactions: Security is a priority when dealing with payments, and Razorpay provides encryption and fraud detection systems that ensure all transactions are secure and reliable. It also supports multiple payment methods like credit/debit cards, net banking, UPI, and wallets.
- Global Payment Support: While Learn Sphere primarily targets the Indian market, Razorpay's ability to support international payment methods ensures that the platform can scale globally, allowing users from different countries to easily make payments.
- Why It Was Chosen for Learn Sphere: Razorpay was chosen because of its ease of use, broad payment method support, and robust security features. It simplifies the payment process for students purchasing courses, making the overall user experience smoother and more trustworthy.

4. Frontend Technologies (HTML, CSS, JavaScript, Thymeleaf)

The frontend of Learn Sphere is developed using HTML, CSS, JavaScript, and Thymeleaf. These technologies allow for the creation of a dynamic, responsive user interface that enhances user interaction and engagement.

• Why HTML, CSS, and JavaScript?

- HTML: The foundation of the web, HTML is used to define the structure and content of the web pages. In Learn Sphere, it is responsible for laying out the course catalog, course pages, dashboards, and other key elements.
- CSS: CSS is used to style the HTML elements, ensuring that the platform is visually appealing and easy to navigate. Using responsive design principles, CSS ensures that the platform adapts seamlessly across all screen sizes, whether on mobile or desktop devices.
- JavaScript: JavaScript is used for enhancing the interactivity of the platform. It allows for dynamic content updates, such as course filtering, live progress tracking, and interactive elements on the user interface. Additionally, JavaScript is used for form validation, ensuring that user inputs are correctly formatted before submission.

2. Why Thyme leaf?

- Thyme leaf is a modern server-side Java templating engine that integrates well with Spring Boot. It enables dynamic content rendering, which allows the platform to inject real-time data (such as course descriptions, progress bars, and payment statuses) into HTML pages.
- Why It Was Chosen for Learn Sphere: Thymeleaf is tightly coupled with Spring Boot, making it the ideal choice for rendering HTML pages that require dynamic content. It allows the backend to send real-time data to the frontend, ensuring that the user sees the most up-to-date information, whether viewing a course or tracking their progress.

5. Spring Security (Security and Authentication)

Spring Security is a powerful and customizable authentication and authorization framework used to secure Learn Sphere. It ensures that all user interactions with the platform are secure, and only authorized users have access to the appropriate features.

5 Why Spring Security?

- Comprehensive Authentication: Spring Security provides out-of-the-box solutions for user authentication, including support for form-based login, basic authentication, and JWT (JSON Web Token) authentication. It allows users to log in securely, preventing unauthorized access.

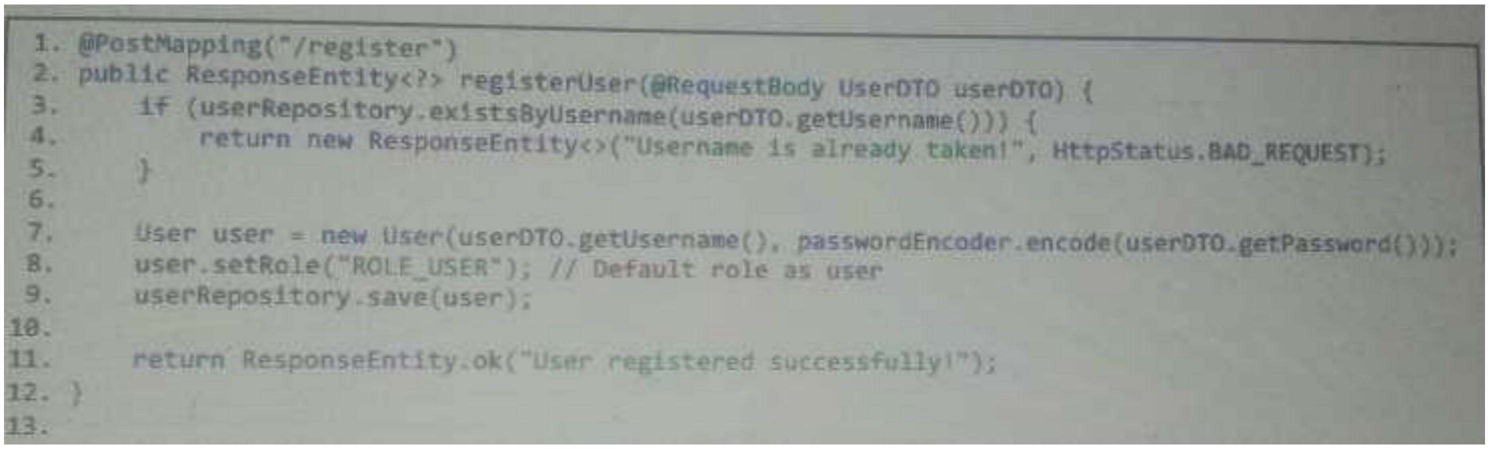
3.3 Implementation:

- **User Management Module (Learn Sphere):**

The User Management Module in Learn Sphere is crucial for managing user authentication, registration, and maintaining the security of the platform. The Learn Sphere project is built using Spring Boot and MySQL, and this module focuses on handling the registration, login, and role management of users (Admin, Trainer, and Student). Let's break down the specific code implementation related to User Management in Learn Sphere.

User Registration

The user registration functionality is responsible for allowing new users (trainers, students, admins) to sign up for the platform. It ensures that only valid users can be registered and adds them to the database with encrypted passwords.



```
@PostMapping("/register")
public ResponseEntity<?> registerUser(@RequestBody UserDTO userDTO) {
    if (userRepository.existsByUsername(userDTO.getUsername())) {
        return new ResponseEntity<>("Username is already taken!", HttpStatus.BAD_REQUEST);
    }

    User user = new User(userDTO.getUsername(), passwordEncoder.encode(userDTO.getPassword()));
    user.setRole("ROLE_USER"); // Default role as user
    userRepository.save(user);

    return ResponseEntity.ok("User registered successfully!");
}
```

Fig 3.1: User Registration

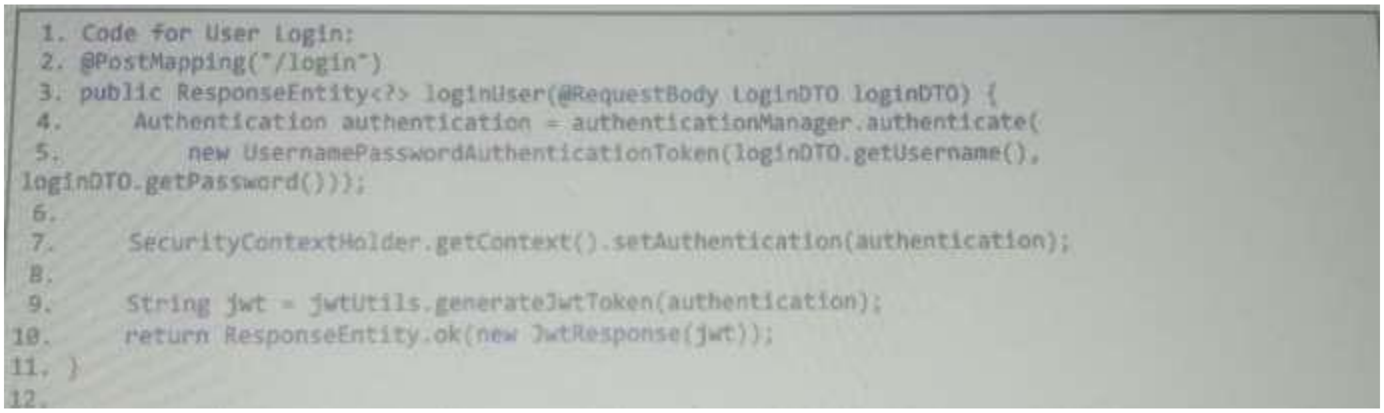
Detailed Explanation of the Code:

1.POST Method for Registration:

- The @PostMapping("/register") annotation maps the method to handle POST requests made to the /register endpoint. This method handles the incoming request to register a new user on the platform.
- UserDTO: @RequestBody UserDTO userDTO binds the incoming request body to a UserDTO object. This object contains data such as username and password which the user enters while registering.
- Check for Existing User: userRepository.existsByUsername(userDTO.getUsername()) checks whether a user with the provided username already exists in the database. If the username is taken, the API returns a BAD_REQUEST (400) response with a message saying "Username is already taken!". This is a simple validation to prevent duplicate users from being created.
- Setting Default Role: The default role for all newly registered users is set as ROLE_USER. This ensures that users will have the basic user permissions, but this can be changed later by an admin.

2. User Login:

The login authentication ensures that only registered users can access the platform. When a user logs in, their credentials (username and password) are validated. If correct, the system generates a JWT (JSON Web Token), which is used for authenticating future requests.



```
Code for User Login:
@PostMapping("/login")
public ResponseEntity<?> loginUser(@RequestBody LoginDTO loginDTO) {
    Authentication authentication = authenticationManager.authenticate(
        new UsernamePasswordAuthenticationToken(loginDTO.getUsername(),
loginDTO.getPassword()));

    SecurityContextHolder.getContext().setAuthentication(authentication);

    String jwt = jwtUtils.generateJwtToken(authentication);
    return ResponseEntity.ok(new JwtResponse(jwt));
}
```

Fig 3.2: User login

Detailed Explanation of the Code:

- POST Method for Login: The @PostMapping("/login") annotation maps this method to handle POST requests made to the /login endpoint. This method is responsible for authenticating the user with their username
- Response: o The generated JWT token is returned as part of the JwtResponse object in the response body. The frontend stores this token and sends it with every request that requires authentication.

3. Role-Based Access Control (RBAC)

RBAC ensures that users are assigned roles (e.g., Admin, Trainer, Student) and can access only the resources.

1. @EnableWebSecurity: The @EnableWebSecurity annotation enables Spring Security configuration for the web application. It activates the security mechanisms, such as authentication and authorization.

2. Role-based Authorization: antMatchers("/admin/**") has Role ("ADMIN") restricts access to any URL starting with /admin/ to users with the ADMIN role. o .antMatchers("/trainer/**").has Role("TRAINER") restricts access to URLs starting with /trainer/ to users with the TRAINER role. o .antMatchers("/student/**").hasRole("STUDENT") ensures that only STUDENT users can access URLs starting with /student/. o anyRequest().authenticated() ensures that all other requests to the application are only accessible to authenticated users.

3. Login and Logout Configuration: formLogin().permitAll() allows anyone to access the login page, regardless of their role. o .logout().permitAll() allows all users to log out, even if they are not authenticated.

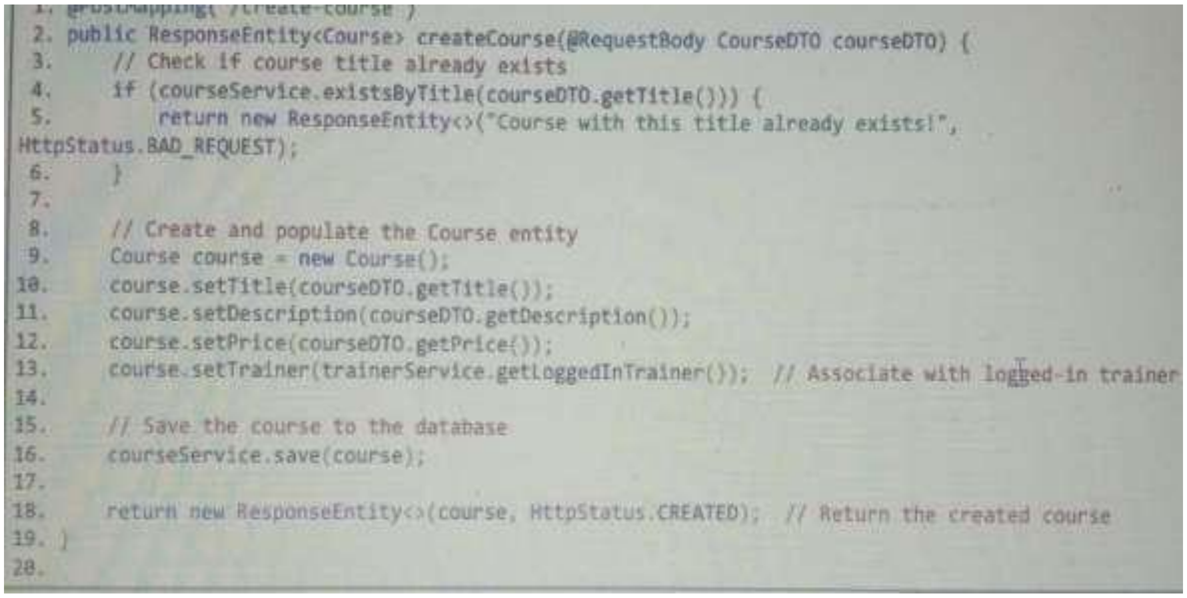
2. Course Management Module:

The Course Management Module is a fundamental part of the Learn Sphere platform. This module allows Trainers to create, update, and manage courses and their associated lessons. It also provides Students with the ability to explore, enroll, and track their progress in various courses.

Key Features of the Course Management Module:

- Course Creation: Trainers can create new courses, set the course title, description, and assign relevant lessons.
- Lesson Management: Trainers can add lessons to courses, including text-based content, videos, and assignments.
- Course Visibility: Trainers can set courses as public or private, allowing for control over which students can access them.
- Course Enrollment: Students can view available courses, enroll in them, and track their progress.
- Progress Tracking: Students can see their progress in each course, such as lessons completed and quizzes taken.
- **Course Creation:**

  The Course Creation functionality allows Trainers to create and manage new courses. This involves setting up the course title, description, price (if applicable), and other relevant details. The course is then saved in the database, and its visibility can be managed.



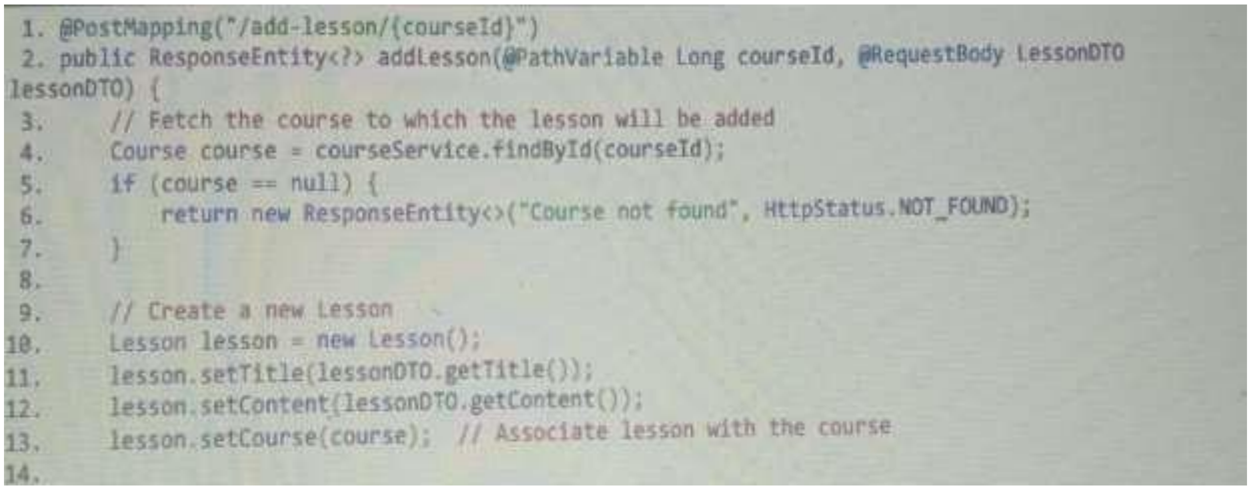
```
@PostMapping("/create-course")
public ResponseEntity<Course> createCourse(@RequestBody CourseDTO courseDTO) {
    // Check if course title already exists
    if (courseService.existsByTitle(courseDTO.getTitle())) {
        return new ResponseEntity<>("Course with this title already exists!",
HttpStatus.BAD_REQUEST);
    }

    // Create and populate the Course entity
    Course course = new Course();
    course.setTitle(courseDTO.getTitle());
    course.setDescription(courseDTO.getDescription());
    course.setPrice(courseDTO.getPrice());
    course.setTrainer(trainerService.getLoggedInTrainer());  // Associate with logged-in trainer

    // Save the course to the database
    courseService.save(course);

    return new ResponseEntity<>(course, HttpStatus.CREATED);  // Return the created course
}
```

Fig3.3: Course Creation

Explanation of the Code:

- POST Method for Creating a Course: o The @PostMapping("/create-course") annotation defines an endpoint that handles POST requests made to the /create-course path. This endpoint is used by trainers to create new courses.
- CourseDTO: o @RequestBody CourseDTO courseDTO binds the incoming JSON request body to the CourseDTO object. This object contains the title, description, and price of the course provided by the trainer.
- Course Existence Check: The course Service. Exists By Title(course DTO. Get Title()) checks whether a course with the same title already exists in the database. If the title is not unique, a BAD_REQUEST (400) response is returned with an appropriate message.
- Creating a Course Entity: o A new Course object is created, and the details from the CourseDTO object are used to populate the Course entity (i.e., title, description, and price).

2. Adding Lessons to a Course



```
@PostMapping("/add-lesson/{courseId}")
public ResponseEntity<?> addLesson(@PathVariable Long courseId, @RequestBody LessonDTO
lessonDTO) {
    // Fetch the course to which the lesson will be added
    Course course = courseService.findById(courseId);
    if (course == null) {
        return new ResponseEntity<>("Course not found", HttpStatus.NOT_FOUND);
    }

    // Create a new Lesson
    Lesson lesson = new Lesson();
    lesson.setTitle(lessonDTO.getTitle());
    lesson.setContent(lessonDTO.getContent());
    lesson.setCourse(course);  // Associate lesson with the course

```

Fig3.4: Adding Lessons to a Course

After a course is created, trainers need to add lessons. This can involve adding text, videos, quizzes, and assignments. Lessons are linked to a specific course, and each lesson can contain multimedia content.

Explanation of the Code:

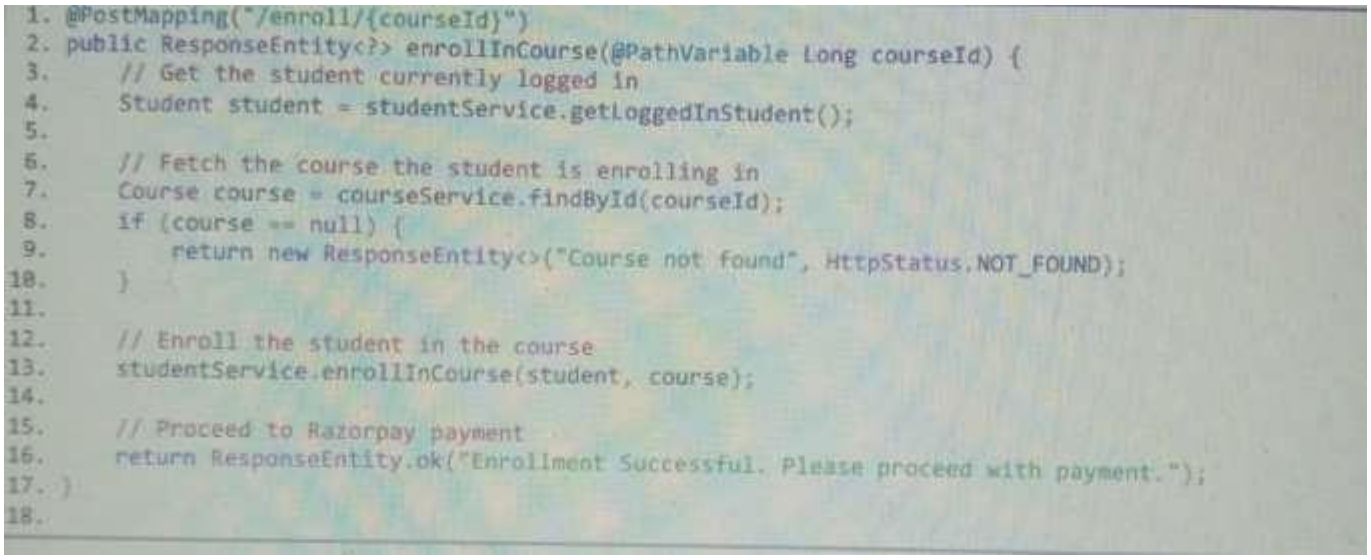
1. POST Method for Adding Lessons: The @PostMapping("/add-lesson/{courseId}") annotation maps this method to the POST request made to the /add-lesson/{courseId} endpoint. This endpoint allows trainers to add lessons to a specific course, which is identified by the courseId path variable.

2. Fetching Course: The course Service. Find By Id(courseId) method is used to fetch the course to which the lesson will be added. If the course does not exist, the backend returns a NOT_FOUND (404) response.

3. Creating a Lesson: A new Lesson object is created using the details from the Lesson DTO object, which contains the lesson's title and content.

4. Associating the Lesson with the Course: The lesson is associated with the course by setting the course reference. This ensures that the lesson is linked to the correct course in the database.

5. Saving the Lesson: The lesson Service. save(lesson) method saves the lesson to the MySQL database, where it is linked to the course and available for students to view. 6. Success Response: o After the lesson is successfully created, the backend sends the created lesson as part of the response with a CREATED (201) HTTP status.



```
@PostMapping("/enroll/{courseId}")
public ResponseEntity<?> enrollInCourse(@PathVariable Long courseId) {
    // Get the student currently logged in
    Student student = studentService.getLoggedInStudent();

    // Fetch the course the student is enrolling in
    Course course = courseService.findById(courseId);
    if (course == null) {
        return new ResponseEntity<>("Course not found", HttpStatus.NOT_FOUND);
    }

    // Enroll the student in the course
    studentService.enrollInCourse(student, course);

    // Proceed to Razorpay payment
    return ResponseEntity.ok("Enrollment Successful. Please proceed with payment.");
}

```

Fig3.5: Course Enrollment

3. Course Enrollment:

Once a course is created, students need to be able to enroll in it. The enrollment process includes selecting a course.

Explanation of the Code:

1.POST Method for Enrollment: The @PostMapping("/enroll/{courseId}") annotation maps this method to handle POST requests made to the /enroll/{courseId} endpoint. This endpoint handles the enrollment of students in courses

2. **Getting the Logged-in Student:** The student Service. Get Logged In Student() method fetches the currently logged-in student. This is crucial because the enrollment should be associated with the student who is making the request.
3. **Fetching the Course:** The course Service. Find By Id(courseId) method retrieves the course the student is trying to enroll in. If the course does not exist, a NOT_FOUND (404) response is returned.
4. **Student Enrollment:** The student Service. Enroll In Course(student, course) method enrolls the student in the course by adding an entry in the database that associates the student with the course.
5. **Redirecting to Razorpay:** o After enrollment, the system prepares to proceed with payment using Razorpay

3.4 Result and Snapshot:

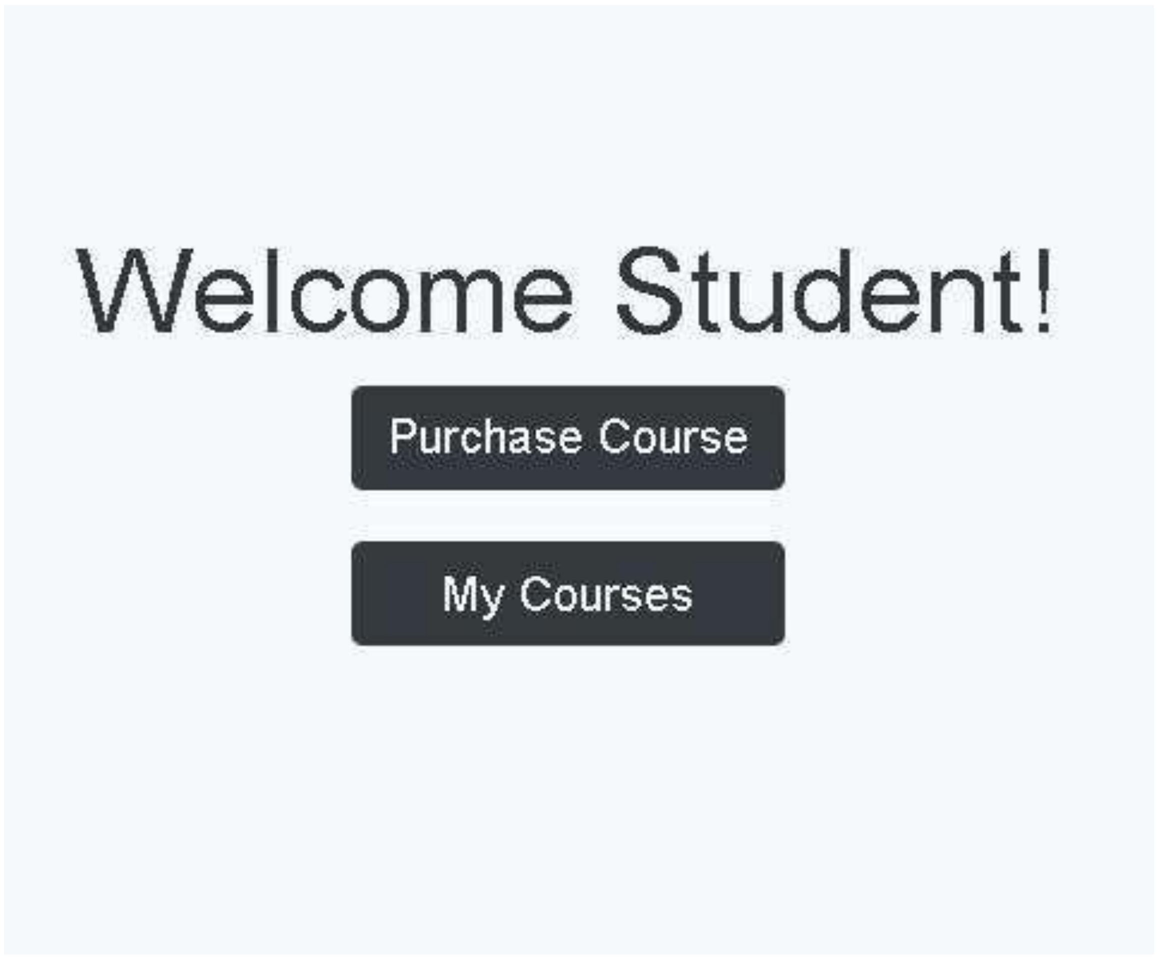
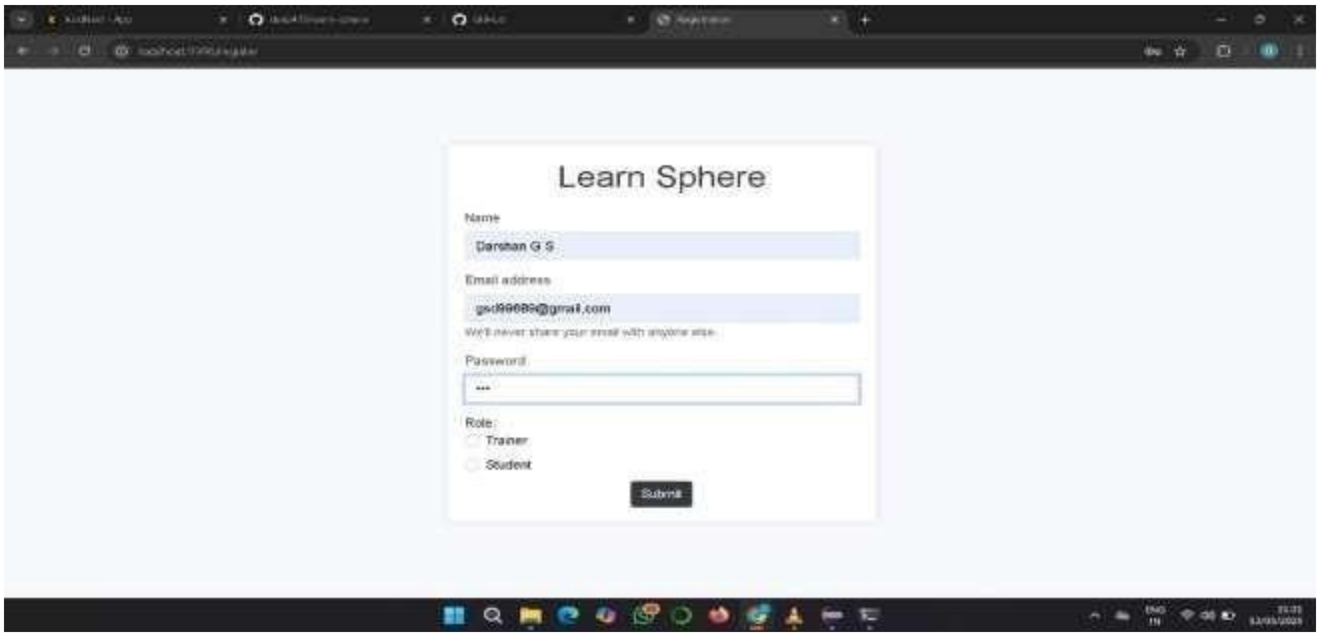


Fig 3.6: Student Home page



Learn Sphere

Name

Darshan G S

Email address

gsd99689@gmail.com

We'll never share your email with anyone else.

Password

***

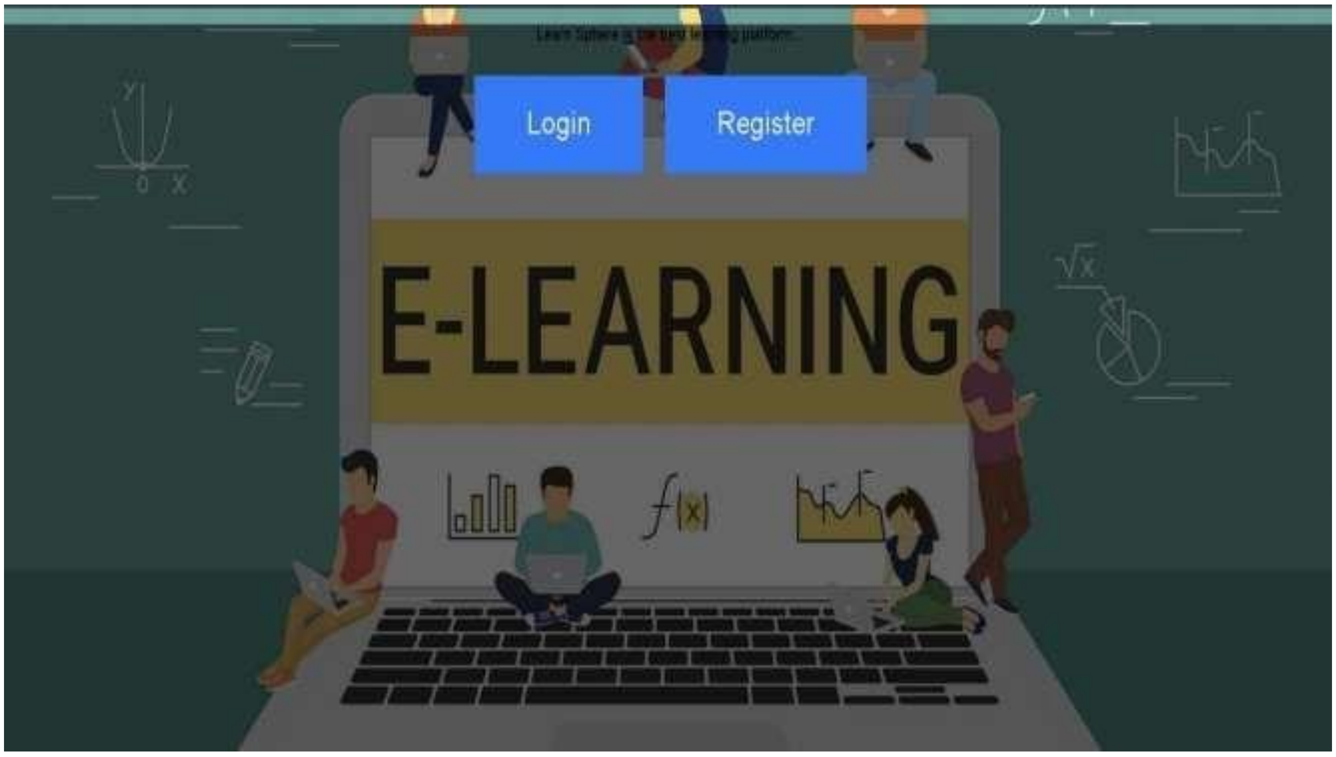
Role:

Trainer

Student

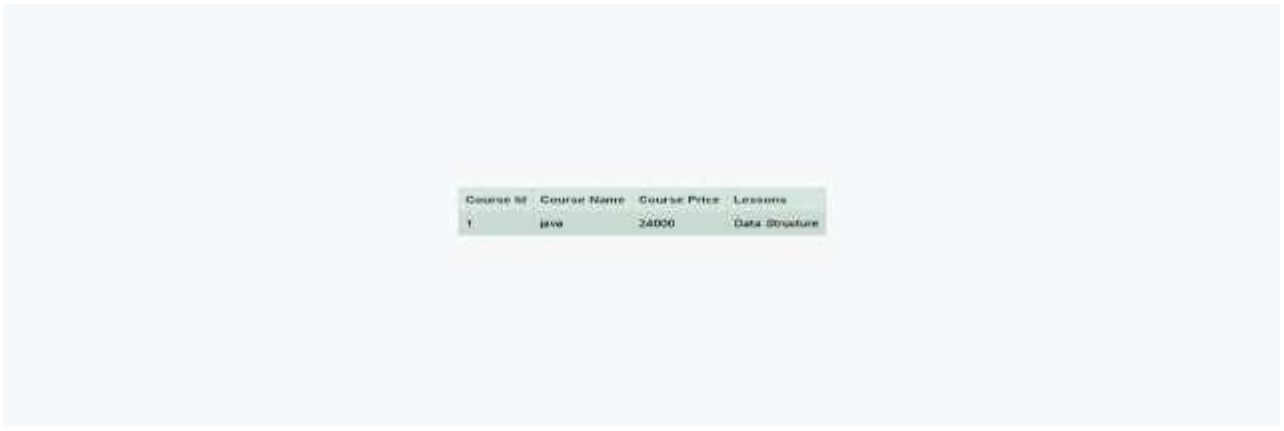
Submit

Fig 3.7: Login page



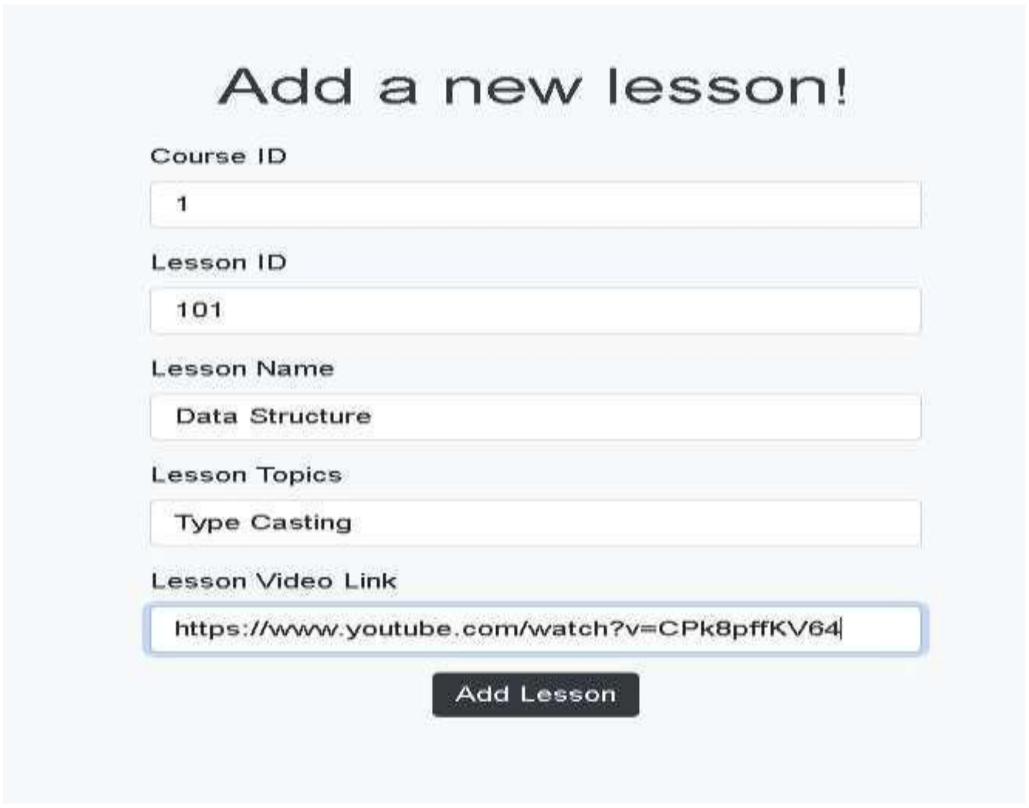
Learn Sphere is the best learning platform...

Fig 3.8: Home Page



Course Id	Course Name	Course Price	Lessons
1	java	24000	Data Structure

Fig3.9: Data set page



Add a new lesson!

Course ID	1
Lesson ID	101
Lesson Name	Data Structure
Lesson Topics	Type Casting
Lesson Video Link	https://www.youtube.com/watch?v=CPk8pffKV64

Fig 3.10: Course adding page

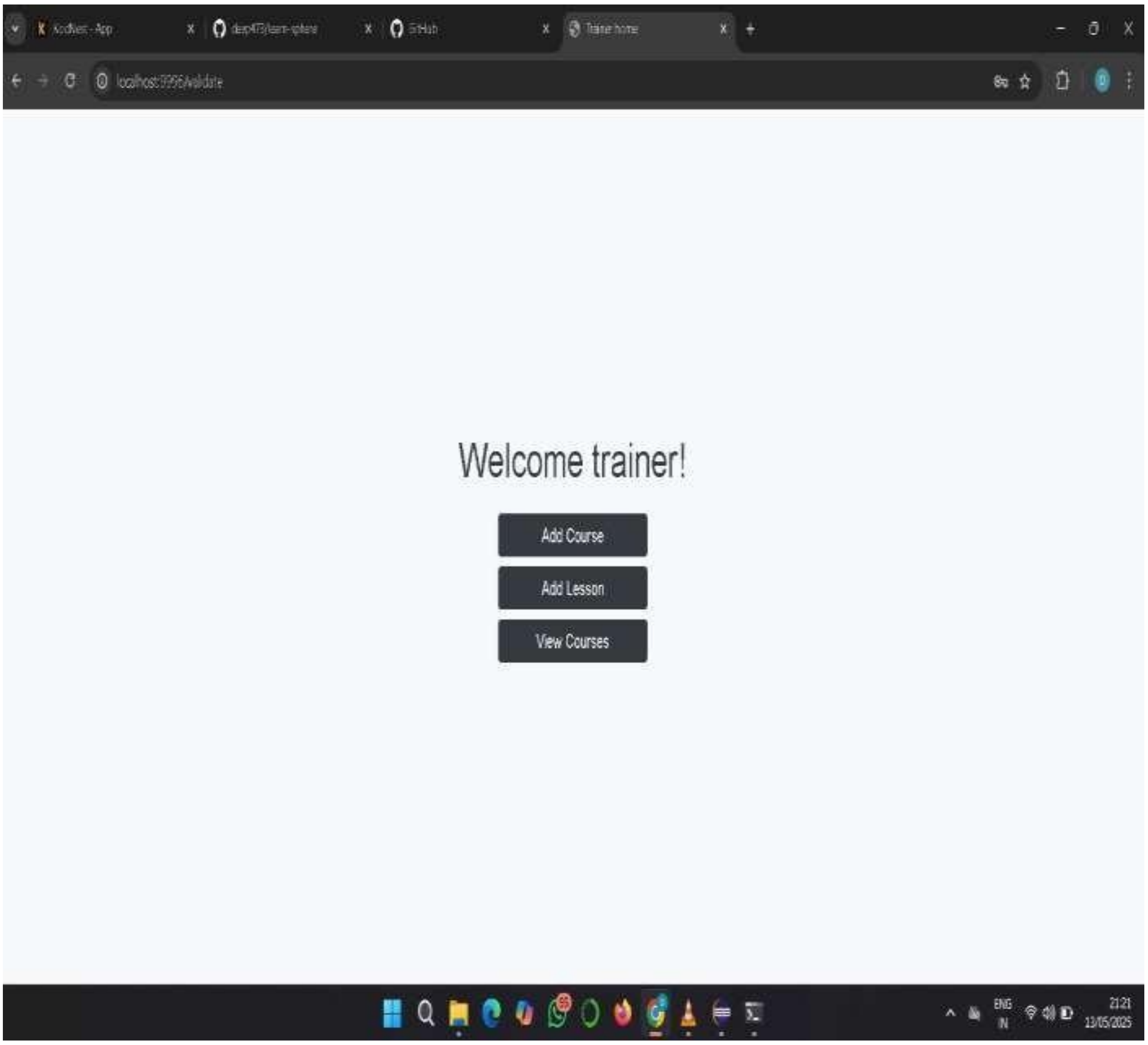
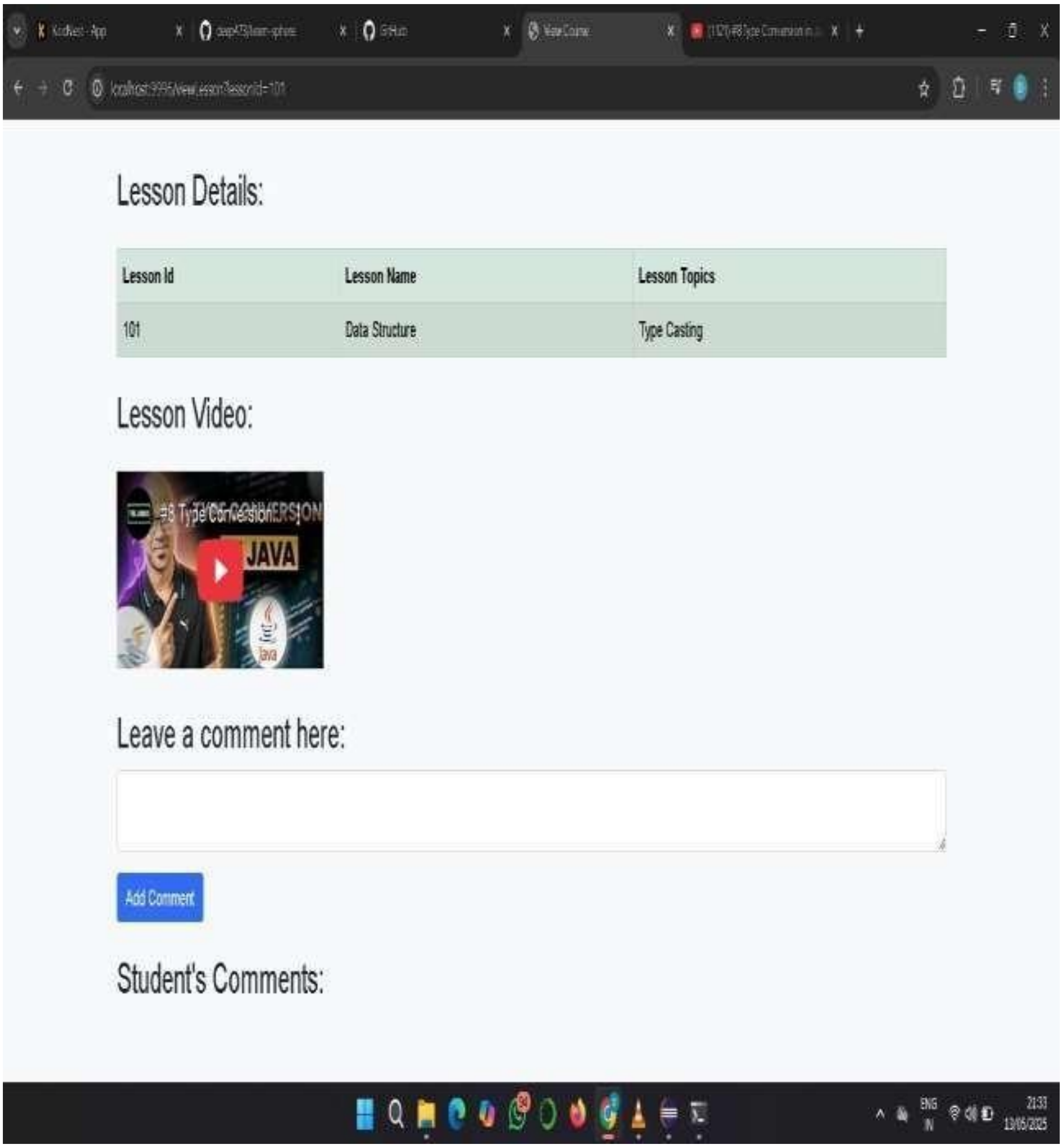


Fig 3.11: Trainer home page



Lesson Details:

Lesson Id	Lesson Name	Lesson Topics
101	Data Structure	Type Casting

Lesson Video:

Leave a comment here:

Student's Comments:

Fig 3.12: Student learn page

Chapter 4

REFLECTION NOTES

4.1 Technical Outcomes of Internship

1) Enhanced Front-end Development Skills

Gained proficiency in building responsive, interactive UIs with HTML5, CSS3, and modern JavaScript. Learned semantic structuring, Flexbox and Grid layouts, and optimized assets for performance. Developed modular, maintainable styles and enhanced user experience through dynamic DOM manipulation using Vanilla JS.

2) Mastery of React for Client-side Applications

Built reusable components with React, managed state via hooks and context, and implemented client-side routing with React Router. Understood lifecycle methods and optimized render performance through memorization and lazy loading. Integrated third-party libraries (e.g., Axios for HTTP requests) and applied best practices for component-driven architecture.

3) Strong Command of Java and Spring Ecosystem

Deepened understanding of Java fundamentals—collections, streams, generics—and object-oriented design principles. Applied the Spring Framework and Spring Boot to rapidly bootstrap microservices: configured IoC containers, leveraged dependency injection, and employed Spring Data JPA for ORM with PostgreSQL.

4) Design and Implementation of RESTful APIs

Designed and implemented REST endpoints in Spring Boot using @RestController, @RequestMapping, and @ExceptionHandler for robust error handling. Employed DTOs and Model Mapper to separate internal entities from API contracts. Secured endpoints with Spring Security and JWT-based authentication, enabling stateless, token-driven sessions.

5) Full-stack Integration and State Management

Connected React front-end to Spring Boot back-end through Axios-based HTTP clients. Managed asynchronous data flows and loading states in the UI. Implemented form validation on both client (using formik + yup) and server sides. Ensured CORS configuration and proper error propagation for smooth user interactions.

6) Database Modeling and Persistence

Modeled relational entities (Users, Roles, Products, Orders) and defined one-to-many/many-to-many relationships in JPA. Wrote JPQL and Criteria API queries for complex data retrieval. Tuned database performance via indexing strategies and pagination.

7) Version Control and Collaboration

Used Git and GitHub for branching, pull requests, and code reviews. Followed GitFlow workflow to manage feature development and releases. Collaborated daily with teammates via stand-up meetings, tracking tasks in Jira.

8) Problem-Solving and Debugging Expertise

Strengthened ability to debug across the stack: used browser DevTools for front-end issues, logging and breakpoints in IntelliJ for back-end exceptions, and SQL explain plans for performance bottlenecks. Broke down complex bugs into reproducible test cases and applied systematic troubleshooting.

4.2 Non-Technical Outcomes of Internship

1) Team Work

Teamwork is crucial for achieving success in various domains, including business, sports, education, and more. It harnesses the collective strengths and talents of individuals, promotes collaboration and innovation, and enhances productivity and satisfaction. By working together as a team, individuals can accomplish goals that would be challenging or impossible to achieve individually, leading to shared success and growth.

2) Communication Skills

Good communication skills are a hallmark of effective leaders. Leaders who can clearly articulate their vision, inspire others, and convey their expectations create a positive impact on their teams and organizations. Effective communicators are adept at motivating and influencing others, fostering a collaborative and productive work environment.

3) Time Management

Enhanced focus and concentration: Time management involves eliminating distractions and creating a conducive environment for focused work. By dedicating specific time slots for important tasks and minimizing interruptions, individuals can improve their focus and concentration. This increased focus leads to higher quality work and greater efficiency.

Opportunity for personal growth: Efficient time management allows individuals to allocate time for self-improvement and personal technologies. It provides opportunities for learning new skills, pursuing hobbies, and engaging in activities that contribute to personal growth. By dedicating time to continuous learning and self-reflection, individuals can enhance their knowledge, skills, and overall well-being.

4) Creativity

Creative skills help you view challenges in new ways. These skills allow you to examine all aspects of a situation and consider new possibilities that challenge the status quo. Creativity is important to many employers because it often leads to innovation that pushes the company in new directions. By learning new things, we can do new things with our creativity.

5) Collaboration

Collaboration skills relate to how well you work with others on a project to achieve a shared goal. These skills help you create a team-first mindset to focus on shared success rather than individual success. Collaboration skills are especially important when you may work on a team with members in other areas, requiring you to adapt to different situations or use various communication styles.

Chapter 5

CONCLUSION

The Full Stack Web Development internship at TAP Academy provided a comprehensive and practical learning experience, bridging theoretical knowledge with real-world application. Through hands-on projects and mentor guidance, I developed strong skills in both front-end technologies like React and back-end frameworks including Java, Spring Boot, and SQL. This internship deepened my understanding of building scalable, secure, and maintainable web applications following industry best practices. Overall, the experience has significantly enhanced my technical abilities, problem-solving skills, and confidence, preparing me to contribute effectively as a full stack developer in a professional environment.

Fig 5.1 Internship Logo

REFERENCES

[1] www.tapedtech.com.

[2] https://tapedtech.com/full-stack-web-development-course

[3] https://tapedtech.com/blog/full-stack-web-development-guide

[4] https://tapedtech.com/resources/web-development-tutorials