**PerScholas Capstone**

**Objective Statement:**

To create a comprehensive, user-friendly learning platform that enables users to learn web development skills through interactive lessons, quizzes, and projects. The platform will include features such as content management, progress tracking, and assessment, as well as user authentication and authorization to ensure secure access.

**Table of Contents:**

* Case Study Rubrics
* Roadmap
* Presentation Slides
* Daily Progress
* User Stories
  + Admin
  + User
* Technical Architecture
* Wire Frame
* Schema Diagram
* Jira(Project Management)
* Technical Challenges
* Lessons Learned
* Future Improvements

**Roadmap:**

Phase 1: Planning and Setup

In this phase, you will define the project scope and requirements for a learning platform to help people learn web development. You will choose a technology stack that includes Java, Spring Boot, MySQL/PostgreSQL, and Angular/React, and set up the development environment. You will also create the initial database schema and data models, including user authentication and authorization to ensure secure access to the platform.

* Define project scope and requirements
* Choose technology stack (Java, Spring Boot, MySQL/PostgreSQL, Angular/React)
* Create project structure and setup development environment
* Create initial database schema and data models
* Develop user authentication and authorization system

Phase 2: Content Management

In this phase, you will develop the tools for creating and managing course content, including lesson creation and content management tools. You will implement content structure and organization and create tools for users to track their progress and enroll in courses. You may also develop features to enable users to browse and search for courses based on keywords, topic, or other criteria.

* Develop lesson creation and content management tools
* Implement content structure and organization
* Create tools for users to track their progress and enroll in courses

Phase 3: Progress Tracking and Assessment

In this phase, you will develop the progress tracking and reporting features for users. This includes tools for users to track their progress in courses, view course completion status, and take assessments. You will create a custom quiz builder with various question types (e.g., multiple choice, true/false, essay), as well as tools for users to view their scores.

* Develop progress tracking and reporting features for users
* Create custom quiz builder with various question types (e.g., multiple choice, true/false, essay)
* Implement tools for users to take assessments and view their scores

Phase 4: User Interface and Experience

In this phase, you will develop the user interface (UI) for the platform, with a focus on creating an intuitive and user-friendly experience for users. This includes optimizing the UI for mobile devices and accessibility, as well as implementing search and filtering features for courses and content. You may also develop features such as a course review and rating system, and a social network for users to connect with one another.

* Develop intuitive and user-friendly user interface (UI) for users
* Implement search and filtering features for courses and content
* Optimize UI for mobile devices and accessibility

Phase 5: Deployment and Testing

In this final phase, you will perform comprehensive testing to ensure quality and performance, deploy the platform to a staging environment for further testing and feedback, and address any issues or bugs identified during testing. Once the platform is ready, you will deploy it to a production environment and provide ongoing maintenance and support.

* Perform comprehensive testing to ensure quality and performance
* Deploy the platform to a staging environment for further testing and feedback
* Address any issues or bugs identified during testing
* Deploy the platform to a production environment and provide ongoing maintenance and support

**Timeline:**

|  |  |
| --- | --- |
| **Phase** | **Allotted Time** |
| Phase 1: Planning and Setup | 2-3 weeks |
| Phase 2: Content Management | 3-4 weeks |
| Phase 3: Progress Tracking and Assessment | 3-4 weeks |
| Phase 4: User Interface and Experience | 2-3 weeks |
| Phase 5: Deployment and Testing | 1-2 weeks |

**Phase 1:**

1. Define project scope and requirements
   1. User Authentication and Authorization
      1. Users must be able to register for a new account, log in, and log out.
      2. Users should only be able to access certain parts of the platform if they are logged in and authorized to do so.
      3. Users should be able to reset their password if they forget it.
   2. Course Content Creation and Management
      1. Platform administrators should be able to create new courses, lessons, and quizzes.
      2. Platform administrators should be able to edit and delete existing courses, lessons, and quizzes.
      3. Course content should be organized into logical sections or units.
   3. User Progress Tracking and Reporting
      1. Users should be able to view their progress within a course, including completed and incomplete lessons and quizzes.
      2. Users should be able to track their overall progress across all courses.
      3. Users should be able to view their quiz scores and overall course completion status.
   4. Course Discovery and Enrollment
      1. Users should be able to browse and search for courses based on keywords, topic, or other criteria.
      2. Users should be able to enroll in courses and track their progress within them.
   5. User Interface and Experience
      1. The platform should be intuitive and user-friendly, with a clean and modern design.
      2. The platform should be optimized for use on mobile devices.
      3. The platform should include features such as a search bar, navigation menu, and course categories.
   6. Other Requirements
      1. The platform should be secure and protect user data.
      2. The platform should be scalable and able to handle a large number of users and courses.
      3. The platform should be deployed on a reliable hosting environment.
2. Choose technology stack (Java, Spring Boot, MySQL/PostgreSQL, Angular/React)
   1. HTML, CSS, JavaScript, React, Java, Sprint Boot, MySQL
3. Create project structure and setup development environment
   1. Create a new directory for your project, let's call it "learning-platform".
   2. Inside the "learning-platform" directory, create two subdirectories: "client" and "server". The "client" directory will contain all of your front-end code, while the "server" directory will contain your back-end code.
   3. Within the "client" directory, create a new React app using the create-react-app tool. You can do this by running the command "npx create-react-app client" in your terminal.
   4. Within the "server" directory, create a new Spring Boot application using the Spring Initializer tool. You can do this by going to the Spring Initializer website and selecting the appropriate dependencies for your project, such as "Spring Web" and "Spring Data JPA".
   5. Once you have generated your Spring Boot application, you can import it into your preferred Java IDE, such as Eclipse or IntelliJ IDEA.
   6. Within your Spring Boot application, create a new package called "controllers". This package will contain your RESTful API endpoints for communicating with the front-end.
   7. In your React app, you can start building out your front-end components within the "src" directory. You may want to organize your components into subdirectories based on their functionality, such as "components/auth" for authentication-related components.
   8. To connect your front-end and back-end, you will need to make HTTP requests from your React components to your Spring Boot API endpoints. You can use the Axios library for this purpose.
4. Create initial database schema and data models
   1. **Database Schema**
      1. Users Table: Contains user information such as ID, email, password, and role.
      2. Courses Table: Contains course information such as ID, title, description, and creator.
      3. Lessons Table: Contains lesson information such as ID, title, content, and course ID.
      4. Quizzes Table: Contains quiz information such as ID, title, description, and course ID.
      5. Quiz Questions Table: Contains quiz question information such as ID, content, and quiz ID.
      6. Quiz Options Table: Contains quiz answer option information such as ID, content, and question ID.
      7. Quiz Answers Table: Contains quiz answer information such as ID, user ID, quiz ID, question ID, and option ID.
   2. **Data Models**
      1. User Model: Contains user attributes such as ID, email, password, and role.
      2. Course Model: Contains course attributes such as ID, title, description, and creator.
      3. Lesson Model: Contains lesson attributes such as ID, title, content, and course ID.
      4. Quiz Model: Contains quiz attributes such as ID, title, description, and course ID.
      5. Quiz Question Model: Contains quiz question attributes such as ID, content, and quiz ID.
      6. Quiz Option Model: Contains quiz answer option attributes such as ID, content, and question ID.
      7. Quiz Answer Model: Contains quiz answer attributes such as ID, user ID, quiz ID, question ID, and option ID.
   3. **Entity Relational Diagram**

Text

Description automatically generated

1. Develop user authentication and authorization system
   1. Spring Security provides a wide range of features, including authentication and authorization, role-based access control, and support for a variety of authentication mechanisms such as form-based login, HTTP Basic authentication, and OAuth. It also integrates easily with other Spring components such as Spring MVC, Spring Boot, and Spring Data, which makes it a popular choice for building modern web applications.
   2. To get started with Spring Security, you can add the Spring Security dependency to your project's build file (e.g., pom.xml or build.gradle) and then configure the security settings in a Spring Security configuration class. You can also customize the authentication and authorization settings according to your specific requirements.

**Phase 2: Content Management:**

1. Develop lesson creation and content management tools
   1. Create a Lesson model: In order to store the lessons in the database, you need to define a Lesson model that includes fields such as lesson title, content, author, and date created.
   2. Implement CRUD functionality: You need to create a backend API that provides CRUD (create, read, update, and delete) functionality for the Lesson model. This allows users with the appropriate permissions to create, edit, and delete lessons.
   3. Add user roles and permissions: You should add roles and permissions to the application to ensure that only authorized users can create, edit, and delete lessons. For example, only users with the "instructor" role may have permission to create or modify lessons, while other users can only view them.
   4. Build a lesson creation form: You need to create a frontend form that allows instructors to create and edit lessons. This form should include fields for lesson title, content, and other relevant metadata.
   5. Implement rich text editing: You may want to include a rich text editor in the lesson creation form to allow instructors to format and style the lesson content.
   6. Implement media management: You should also consider how instructors can add media such as images or videos to the lessons and implement functionality to manage this media.
   7. Build a lesson view page: You need to create a frontend page that displays the lesson content to students. This page should be designed to be easy to read and navigate.
   8. Implement lesson search and filtering: As the number of lessons grows, you may need to implement search and filtering functionality to make it easier for students to find the lessons they need.
2. Implement content structure and organization
   1. Define content types: In order to organize the lessons and other content on the platform, you need to define the types of content that will be available. For example, you may have lessons, quizzes, and exercises.
   2. Create a content model: Once you have defined the content types, you need to create a content model that includes fields such as title, description, content type, and date created.
   3. Implement CRUD functionality: You need to create a backend API that provides CRUD functionality for the Content model. This allows users with the appropriate permissions to create, edit, and delete content.
   4. Add user roles and permissions: You should add roles and permissions to the application to ensure that only authorized users can create, edit, and delete content. For example, only users with the "instructor" role may have permission to create or modify content, while other users can only view it.
   5. Build a content creation form: You need to create a frontend form that allows instructors to create and edit content. This form should include fields for content title, description, content type, and other relevant metadata.
   6. Implement media management: You should consider how media such as images or videos will be organized and managed on the platform and implement functionality to manage this media.
   7. Build a content view page: You need to create a frontend page that displays the content to students. This page should be designed to be easy to read and navigate.
   8. Implement content search and filtering: As the amount of content on the platform grows, you may need to implement search and filtering functionality to make it easier for students to find the content they need.
   9. Create content categories: You may want to create categories to organize content by subject or topic. For example, you may have categories such as HTML, CSS, JavaScript, and React.
3. Create tools for users to track their progress and enroll in courses
   1. Define user progress tracking: You need to define what metrics will be used to track user progress, such as course completion percentage or quiz scores.
   2. Create a progress tracking model: Once you have defined the metrics to track, you need to create a progress tracking model that includes fields such as user ID, course ID, lesson ID, and completion percentage.
   3. Implement CRUD functionality for progress tracking: You need to create a backend API that provides CRUD functionality for the progress tracking model. This allows users to track their progress and for instructors to view the progress of their students.
   4. Build a progress tracking dashboard: You need to create a frontend dashboard that displays progress information for users. This dashboard should display progress metrics for each course or lesson and allow users to view their progress over time.
   5. Implement course enrollment functionality: You need to create functionality that allows users to enroll in courses. This can be done through a form or button on the course page.
   6. Create a course catalog page: You need to create a frontend page that displays all available courses. This page should allow users to filter courses by category or search for courses by keyword.
   7. Implement course completion certificates: You may want to create certificates for users who complete a course. This can be done automatically once the user completes all the lessons in the course.
   8. Create a user profile page: You need to create a frontend page that displays user information, including their progress, enrolled courses, and completed courses.
   9. Implement progress notifications: You may want to send notifications to users when they reach certain milestones or complete a course. This can be done through email or push notifications.

**Phase 3: Progress Tracking and Assessment:**

1. Develop progress tracking and reporting features for users
   1. Define progress tracking metrics: You need to define what metrics will be used to track user progress, such as course completion percentage or quiz scores.
   2. Create a progress tracking model: Once you have defined the metrics to track, you need to create a progress tracking model that includes fields such as user ID, course ID, lesson ID, and completion percentage.
   3. Implement CRUD functionality for progress tracking: You need to create a backend API that provides CRUD functionality for the progress tracking model. This allows users to track their progress and for instructors to view the progress of their students.
   4. Build a progress tracking dashboard: You need to create a frontend dashboard that displays progress information for users. This dashboard should display progress metrics for each course or lesson and allow users to view their progress over time.
   5. Implement course completion certificates: You may want to create certificates for users who complete a course. This can be done automatically once the user completes all the lessons in the course.
   6. Create a user profile page: You need to create a frontend page that displays user information, including their progress, enrolled courses, and completed courses.
   7. Implement progress notifications: You may want to send notifications to users when they reach certain milestones or complete a course. This can be done through email or push notifications.
   8. Implement reporting features: You need to create reporting features that allow instructors to view progress metrics for their students. This could include reports on course completion rates, quiz scores, and time spent on each lesson.
2. Create custom quiz builder with various question types (e.g., multiple choice, true/false, essay)
   1. Define progress tracking metrics: You need to define what metrics will be used to track user progress, such as course completion percentage or quiz scores.
   2. Create a progress tracking model: Once you have defined the metrics to track, you need to create a progress tracking model that includes fields such as user ID, course ID, lesson ID, and completion percentage.
   3. Implement CRUD functionality for progress tracking: You need to create a backend API that provides CRUD functionality for the progress tracking model. This allows users to track their progress and for instructors to view the progress of their students.
   4. Build a progress tracking dashboard: You need to create a frontend dashboard that displays progress information for users. This dashboard should display progress metrics for each course or lesson and allow users to view their progress over time.
   5. Implement course completion certificates: You may want to create certificates for users who complete a course. This can be done automatically once the user completes all the lessons in the course.
   6. Create a user profile page: You need to create a frontend page that displays user information, including their progress, enrolled courses, and completed courses.
   7. Implement progress notifications: You may want to send notifications to users when they reach certain milestones or complete a course. This can be done through email or push notifications.
   8. Implement reporting features: You need to create reporting features that allow instructors to view progress metrics for their students. This could include reports on course completion rates, quiz scores, and time spent on each lesson.
3. Implement tools for users to take assessments and view their scores
   1. Define progress tracking metrics: You need to define what metrics will be used to track user progress, such as course completion percentage or quiz scores.
   2. Create a progress tracking model: Once you have defined the metrics to track, you need to create a progress tracking model that includes fields such as user ID, course ID, lesson ID, and completion percentage.
   3. Implement CRUD functionality for progress tracking: You need to create a backend API that provides CRUD functionality for the progress tracking model. This allows users to track their progress and for instructors to view the progress of their students.
   4. Build a progress tracking dashboard: You need to create a frontend dashboard that displays progress information for users. This dashboard should display progress metrics for each course or lesson and allow users to view their progress over time.
   5. Implement course completion certificates: You may want to create certificates for users who complete a course. This can be done automatically once the user completes all the lessons in the course.
   6. Create a user profile page: You need to create a frontend page that displays user information, including their progress, enrolled courses, and completed courses.
   7. Implement progress notifications: You may want to send notifications to users when they reach certain milestones or complete a course. This can be done through email or push notifications.
   8. Implement reporting features: You need to create reporting features that allow instructors to view progress metrics for their students. This could include reports on course completion rates, quiz scores, and time spent on each lesson.

**Phase 4: User Interface and Experience:**

1. Develop intuitive and user-friendly user interface (UI) for users
   1. Define progress tracking metrics: You need to define what metrics will be used to track user progress, such as course completion percentage or quiz scores.
   2. Create a progress tracking model: Once you have defined the metrics to track, you need to create a progress tracking model that includes fields such as user ID, course ID, lesson ID, and completion percentage.
   3. Implement CRUD functionality for progress tracking: You need to create a backend API that provides CRUD functionality for the progress tracking model. This allows users to track their progress and for instructors to view the progress of their students.
   4. Build a progress tracking dashboard: You need to create a frontend dashboard that displays progress information for users. This dashboard should display progress metrics for each course or lesson and allow users to view their progress over time.
   5. Implement course completion certificates: You may want to create certificates for users who complete a course. This can be done automatically once the user completes all the lessons in the course.
   6. Create a user profile page: You need to create a frontend page that displays user information, including their progress, enrolled courses, and completed courses.
   7. Implement progress notifications: You may want to send notifications to users when they reach certain milestones or complete a course. This can be done through email or push notifications.
   8. Implement reporting features: You need to create reporting features that allow instructors to view progress metrics for their students. This could include reports on course completion rates, quiz scores, and time spent on each lesson.
2. Implement search and filtering features for courses and content
   1. Define progress tracking metrics: You need to define what metrics will be used to track user progress, such as course completion percentage or quiz scores.
   2. Create a progress tracking model: Once you have defined the metrics to track, you need to create a progress tracking model that includes fields such as user ID, course ID, lesson ID, and completion percentage.
   3. Implement CRUD functionality for progress tracking: You need to create a backend API that provides CRUD functionality for the progress tracking model. This allows users to track their progress and for instructors to view the progress of their students.
   4. Build a progress tracking dashboard: You need to create a frontend dashboard that displays progress information for users. This dashboard should display progress metrics for each course or lesson and allow users to view their progress over time.
   5. Implement course completion certificates: You may want to create certificates for users who complete a course. This can be done automatically once the user completes all the lessons in the course.
   6. Create a user profile page: You need to create a frontend page that displays user information, including their progress, enrolled courses, and completed courses.
   7. Implement progress notifications: You may want to send notifications to users when they reach certain milestones or complete a course. This can be done through email or push notifications.
   8. Implement reporting features: You need to create reporting features that allow instructors to view progress metrics for their students. This could include reports on course completion rates, quiz scores, and time spent on each lesson.
3. Optimize UI for mobile devices and accessibility
   1. Define progress tracking metrics: You need to define what metrics will be used to track user progress, such as course completion percentage or quiz scores.
   2. Create a progress tracking model: Once you have defined the metrics to track, you need to create a progress tracking model that includes fields such as user ID, course ID, lesson ID, and completion percentage.
   3. Implement CRUD functionality for progress tracking: You need to create a backend API that provides CRUD functionality for the progress tracking model. This allows users to track their progress and for instructors to view the progress of their students.
   4. Build a progress tracking dashboard: You need to create a frontend dashboard that displays progress information for users. This dashboard should display progress metrics for each course or lesson and allow users to view their progress over time.
   5. Implement course completion certificates: You may want to create certificates for users who complete a course. This can be done automatically once the user completes all the lessons in the course.
   6. Create a user profile page: You need to create a frontend page that displays user information, including their progress, enrolled courses, and completed courses.
   7. Implement progress notifications: You may want to send notifications to users when they reach certain milestones or complete a course. This can be done through email or push notifications.
   8. Implement reporting features: You need to create reporting features that allow instructors to view progress metrics for their students. This could include reports on course completion rates, quiz scores, and time spent on each lesson.

**Phase 5: Deployment and Testing:**

1. Perform comprehensive testing to ensure quality and performance
   1. Define progress tracking metrics: You need to define what metrics will be used to track user progress, such as course completion percentage or quiz scores.
   2. Create a progress tracking model: Once you have defined the metrics to track, you need to create a progress tracking model that includes fields such as user ID, course ID, lesson ID, and completion percentage.
   3. Implement CRUD functionality for progress tracking: You need to create a backend API that provides CRUD functionality for the progress tracking model. This allows users to track their progress and for instructors to view the progress of their students.
   4. Build a progress tracking dashboard: You need to create a frontend dashboard that displays progress information for users. This dashboard should display progress metrics for each course or lesson and allow users to view their progress over time.
   5. Implement course completion certificates: You may want to create certificates for users who complete a course. This can be done automatically once the user completes all the lessons in the course.
   6. Create a user profile page: You need to create a frontend page that displays user information, including their progress, enrolled courses, and completed courses.
   7. Implement progress notifications: You may want to send notifications to users when they reach certain milestones or complete a course. This can be done through email or push notifications.
   8. Implement reporting features: You need to create reporting features that allow instructors to view progress metrics for their students. This could include reports on course completion rates, quiz scores, and time spent on each lesson.
2. Deploy the platform to a staging environment for further testing and feedback
   1. Define progress tracking metrics: You need to define what metrics will be used to track user progress, such as course completion percentage or quiz scores.
   2. Create a progress tracking model: Once you have defined the metrics to track, you need to create a progress tracking model that includes fields such as user ID, course ID, lesson ID, and completion percentage.
   3. Implement CRUD functionality for progress tracking: You need to create a backend API that provides CRUD functionality for the progress tracking model. This allows users to track their progress and for instructors to view the progress of their students.
   4. Build a progress tracking dashboard: You need to create a frontend dashboard that displays progress information for users. This dashboard should display progress metrics for each course or lesson and allow users to view their progress over time.
   5. Implement course completion certificates: You may want to create certificates for users who complete a course. This can be done automatically once the user completes all the lessons in the course.
   6. Create a user profile page: You need to create a frontend page that displays user information, including their progress, enrolled courses, and completed courses.
   7. Implement progress notifications: You may want to send notifications to users when they reach certain milestones or complete a course. This can be done through email or push notifications.
   8. Implement reporting features: You need to create reporting features that allow instructors to view progress metrics for their students. This could include reports on course completion rates, quiz scores, and time spent on each lesson.
3. Address any issues or bugs identified during testing
   1. Define progress tracking metrics: You need to define what metrics will be used to track user progress, such as course completion percentage or quiz scores.
   2. Create a progress tracking model: Once you have defined the metrics to track, you need to create a progress tracking model that includes fields such as user ID, course ID, lesson ID, and completion percentage.
   3. Implement CRUD functionality for progress tracking: You need to create a backend API that provides CRUD functionality for the progress tracking model. This allows users to track their progress and for instructors to view the progress of their students.
   4. Build a progress tracking dashboard: You need to create a frontend dashboard that displays progress information for users. This dashboard should display progress metrics for each course or lesson and allow users to view their progress over time.
   5. Implement course completion certificates: You may want to create certificates for users who complete a course. This can be done automatically once the user completes all the lessons in the course.
   6. Create a user profile page: You need to create a frontend page that displays user information, including their progress, enrolled courses, and completed courses.
   7. Implement progress notifications: You may want to send notifications to users when they reach certain milestones or complete a course. This can be done through email or push notifications.
   8. Implement reporting features: You need to create reporting features that allow instructors to view progress metrics for their students. This could include reports on course completion rates, quiz scores, and time spent on each lesson.
4. Deploy the platform to a production environment and provide ongoing maintenance and support
   1. Set up a server infrastructure to host the application, configuring the necessary security measures such as firewalls and SSL certificates, and deploying the codebase to the production server.
   2. Ongoing maintenance and support would involve monitoring the system for issues and bugs, providing regular updates and improvements, and addressing any user feedback or support requests. This would also include ensuring the scalability and performance of the system as the user base grows over time. It may also involve creating a system for user feedback and support, such as a ticketing system or a knowledge base.
   3. It's important to have a plan for ongoing maintenance and support from the beginning of the project to ensure that the platform remains functional, secure, and up to date. This could involve assigning a team or individual responsible for maintenance and support, setting up monitoring and alerting systems, and establishing a process for addressing issues and bugs in a timely manner.