launch code

Unit 2 Project Brief: Full-Stack Application

Introduction

It's time to put everything that you've learned together! In this project, you will build a full-stack web application! You can either take your existing front-end application and add a back-end, or you can create an entirely new project with both a front-end and back-end.

Learning Goals

- Learn by creating a full-stack web application.
- Build effective planning material and execute on the material.
- Practice all the topics covered over Unit Two thus far including server side engineering utilizing Java and Spring Boot with MVC convention.
- Combine and apply the skills you've learned from both Unit One and Unit Two by building a full-stack web application from the ground up.
- Build a polished, published website you can share in your portfolio.
- Practice web development best practices such as code reusability and code organization
- Learn to debug and problem-solve coding issues
- Understand the importance of good layout and intuitive design, making the app easy to navigate
- Gain experience with version control using Git

Project Task

Create a full-stack web application with a complete data model, web services, and front-end user interface, including:

1. At least two working models/database tables.
2. Complete RESTful CRUD Web Services using Spring Boot.
3. A complete React front-end that communicates with the back-end using JavasScript fetch requests.
4. At least three pages in your application, including the home/landing page.

Project Requirements

Technical Requirements

- Fully functioning full-stack application created with an MVC-structure using the React, Java and Spring Boot stack.
- Minimum TWO database models with a relationship between them that supports the features you selected.
- Full CRUD must be demonstrated across your app's resources. Not every individual resource needs to support all four CRUD operations, but the app as a whole must include Create, Read, Update, and Delete functionality.
- To keep within time constraints, we recommend designing your database with the “code first” methodology that you learned in Unit 2 with an empty database and wired model classes. (You are **not required** to take the “database first” approach or make a database-creation script).
- RESTful Routes - Design your routes for CRUD endpoints in your Spring Boot controllers using the REST convention.

- Git 50+ commits. Commit early, commit often. Tell the story of the app with your commits. Each message should provide a clear description of what you changed.
- Code must be:
    - Clean
        - No unused or commented-out code
        - Proper spacing and indentation
        - Intuitive and conventional naming of classes, methods and variables.
    - Appropriately commented / documented
        - Proper use of comments to plan or document, removal of unnecessary comments for professional portfolio
        - Professional looking and easy to read
- **BONUS - Utilize Authentication/Authorization**
    - Implement a registration and login system (authentication) as described in Unit 2
    - Apply appropriate authorization attributes to your end-points, and use Principal parameters to return data appropriate for the logged in user
- **BONUS - Include a Database Creation SQL Script**
    - Create a SQL script for doing the initial creation and population of your database
- **BONUS - Deploy your Full-Stack Project**
    - Choose a Cloud Provider and deploy your application
        - A couple of our favorite cloud providers:
            - https://www.digitalocean.com/

        - https://aws.amazon.com/
    - This will require the following:
        - A functioning front-end and back-end application, and database.
        - Virtual Computer and Storage (Server and Database).
    - **Note:** All cloud providers and services (like retrieving API data!) have an associated cost. Please be mindful when creating and adding additional resources to your account.

Content Requirements

- A working full-stack application.
- Completion of the technical requirements above.
- A **README.md** file that serves as your project documentation and includes:
    - A paragraph-long description (elevator pitch) of your project.
    - A list of the technologies used.
    - A list of installation steps for the app itself and any dependencies - how would another developer run your site locally?
    - A link to your wireframes.
    - A link to your entity relationship diagrams.
    - Descriptions of any unsolved problems or future features.

Presentation Requirements

- 10-minute presentation demonstrating:
    1. Website functionality
    2. Code explanation of one CRUD feature
    3. Main challenges and solutions

4. Future enhancement plans

- Q&A session (2-3 minutes)

Planning Requirements

- **User Stories**: Define what actions a user can take within your application to help guide the development process. User stories determine the features or functionalities of an application from the perspective of the end-user:
    - Those needed to achieve MVP in the two week time frame.
    - Those that could be classified as future enhancements.
    - You can learn more about user stories here.
- **Wireframes** that sketch each screen's user interface for the major functionality of the application. You can sketch the wireframes by hand or use a digital tool of your choice.
    - You can learn more about wireframes here.
- **ERD Data Models** that illustrate all data models and their relationships (associations) within your project.

  You can learn more about ERD's here.
- **Daily Standups** in small groups to share resources and troubleshoot each other's blockers.

Projected Timeline

- Milestone 1: Planning deliverable approval
- Milestone 2: Creating back-end database, and Java server project with models
- Milestone 3: Writing Controllers/Routes
- Milestone 4: Creating a new front-end or preparing your existing front-end.
- Milestone 5: Connecting your front-end and back-end

- Milestone 6: Testing and debugging
- Milestone 7: Final adjustments and ReadME
- Milestone 8: Presentations

Project Submission

To submit your project, complete the following steps:

1. Make a copy of the submission sheet: here.
2. Fill out the sheet completely.

Submit the url for your submission sheet into the Canvas project submission.

Resources

Learning Materials

- Java Style Guide - Coding Standards
- MDN
- W3Schools
- StackOverflow

Development Tools

- IntelliJ
- MySql WorkBench
- Grepper
- Github Desktop
- DevTools (Chrome and Firefox)

Design Resources

- Google Fonts
- Unsplash
- FontAwesome icons

Deployment Resources

- Deploying an App from a Linux Machine
- AWS Basics