Software Design and Engineering

Lab Document

High Level Purpose Statement:	Pt 1. My goal for this lab is to create a full stack authentication system using Spring Boot with Java and React. I will also use MongoDB for the database. Maven is used for the backend dependency manager, and NPM is used for dependencies on the frontend. This project will allow a user to login/register an account to the DB. Pt 2. My goal for this lab is to create the Stripe API integration for the Connext Game using Spring Boot and ReactJS. I will follow boilerplate examples and append my frontend to implement the api.
Experimental Design:	The goal for this lab is to make a similar authentication system that I did for Postgres and MongoDB, only this time I will use Spring Boot for the backend instead of ExpressJS. I will use MongoDB for the database. I will use Maven and NPM for full stack dependencies I will follow boiler plate code for Spring Boot authentication I will re-use my frontend from Lab 4. I will use stripes API for payments
Resources Available:	Resources that I heavily used: ● https://www.youtube.com/watch?v=5PdEmeopJVQ&t=9167s ● https://github.com/bezkoder/spring-boot-security-jwt-auth-mongodb/blob/master/src/main/java/com/bezkoder/spring/jwt/mongodb/controllers/AuthController.java ● https://github.com/fhsinchy/movieist/blob/master/src/main/java/dev/farhan/movieist/movies/Movie.java ● https://github.com/GretiCani/spring-boot-stripe-payment-integration/blob/master/src/main/resources/templates/subscription.html ● https://github.com/neerajparkashsharma/spring-boot-stripe-payment/blob/main/pom.xml
Time Estimate:	Pt 1. I will spend about 10 hours on this project. I will follow along with the YouTube video to gain an understanding of Spring Boot compared to ExpressJS. This time, I'll review the boilerplate code

	for the backend, and plan my authentication system accordingly. I plan to use the Roles model in the future lab for processing Admin pages/authorities. Pt 2. I will spend about six hours on this part of the project. I will follow along youtube videos and boiler plate code on github for integrating stripe in to a Spring Boot backend.
Experiment Notes:	● Spring Boot is very similar to ExpressJS ○ Lots of controllability ○ Connection Strings w/ .env ○ Backend frameworks w/ easy implementation ● MongoDB is easy to use with Spring Boot because of using a .env w/ connection string ● NoSQL makes it very easy to implement code without the headache of tables ● Cross Origin issues make no sense
Results:	Pt 1.The project now properly runs using NPM and the MERN stack to authenticate login or registration through a React form field. NPM acts a lot like Maven or Gradle in this case because it is used for dependency management and run configuration. Once I configured the scripts within NPM all the user has to do to run this project is open the terminal and type “npm start”. The project now properly runs using Maven and NPM. I am using a front and backend (but it’s weirdly setup). ● **In order to start the project** ○ **mvn spring-boot:run** ○ **(OPEN NEW TERMINAL)** ■ **cd frontend** ■ **Npm install** ■ **Npm start** Pt 2. The project runs, but when you try to click on a plan under the no ads tab on the home page, you get CORS errors. I tried to resolve this error over the span of four hours, but no matter what I tried to do, nothing would resolve it. On the other hand, let's imagine this was working, you, the user, would be redirected to a pre designed stripe session page where you can “test” the payment plan. Stripe allows for easy integration, when CORS doesn’t get in the way, for apps to accept payment plans.
Consequences for the Future:	In the future, I should update the Role model for testing an admin system with payments. Reconfiguring the modules instead of nesting them for better modularity. I should also fix more time into resolving CORS issues when the api endpoint for stripe subscription is being called. I took about four hours on the same error, yet I still couldn’t get it to work. ← This is why MERN or NextJS is better imo. All you have to do is call “app.use(cors())” and everything works for you. SB on the other hand has more

	controllability, yet it takes a lot more in depth thinking about origin points and access for other web endpoints.