**EECS4413 Fall 2020**

**Project Report**

**December 4, 2020**

Rui Yang Huang

214773279

[huangr@my.yorku.ca](mailto:huangr@my.yorku.ca)

Farhan Latif

215145592

[farhan95@my.yorku.ca](mailto:farhan95@my.yorku.ca)

Youngjin Ko

214876924

[Youngjin.ko@youngjinko.me](mailto:Youngjin.ko@youngjinko.me)

**Table of Contents**

**Part 1: Diagrams p.2**

UML Diagram p.3

Use Cases Diagram p.4

Sequence Diagram #1 p.5

Sequence Diagram #2 p.5

**Part 2: Design Decisions p.6**

Design Decisions p.6

Model-View-Controller (MVC) p.6

Data Access Object (DAO) p.6

Event-Driven p.7

**Part 3: Implementation p.7**

Angular and Spring boot p.7

HTML-typescript-service/model-router (Angular) p.7

model-repo-controller-application (Spring boot) p.8

Decisions p.8

Database p.9

Tradeoffs p.10

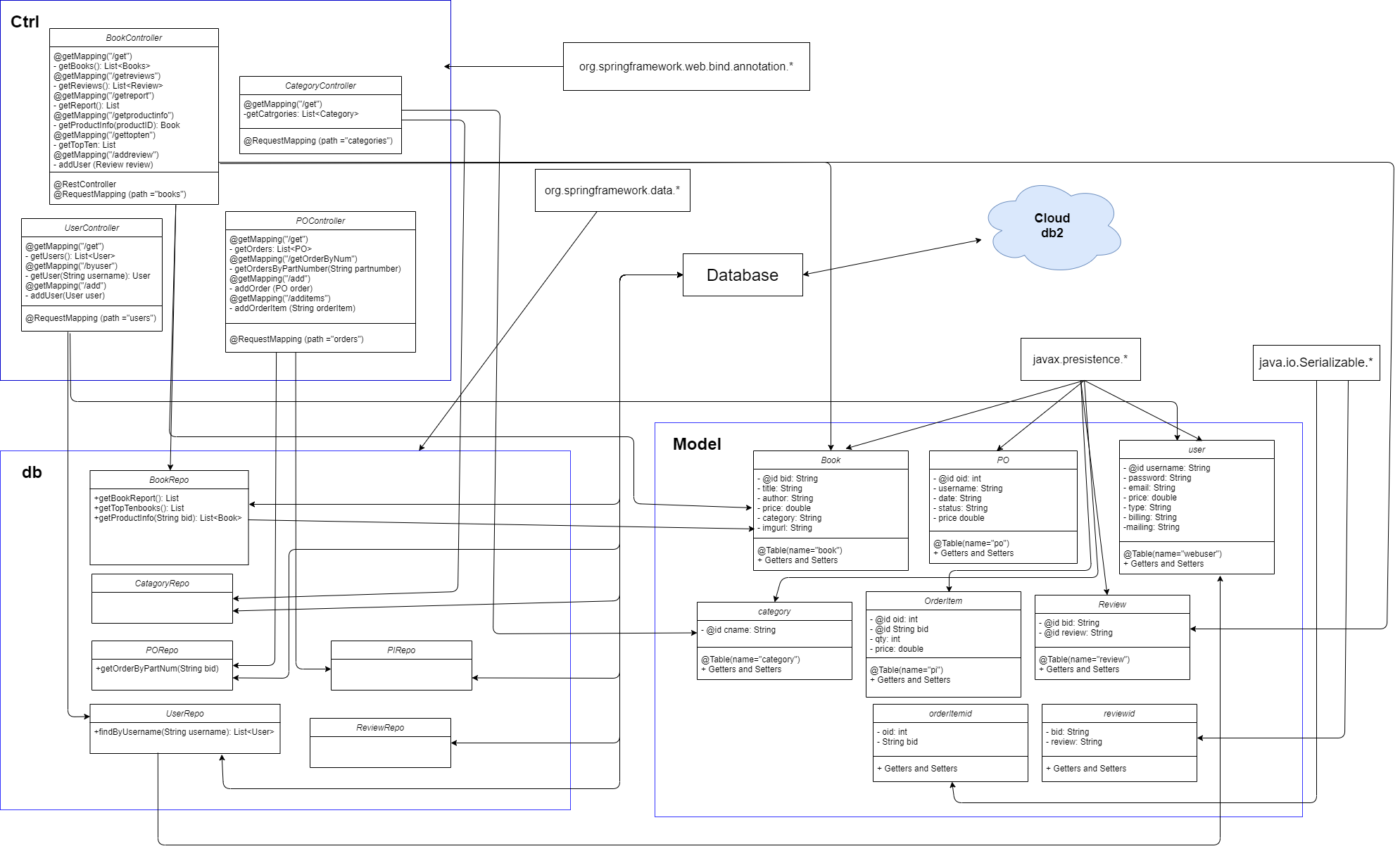
Instructions to turn on the angular project in your machine p.10

**Part 4: Team Member Contributions p.12**

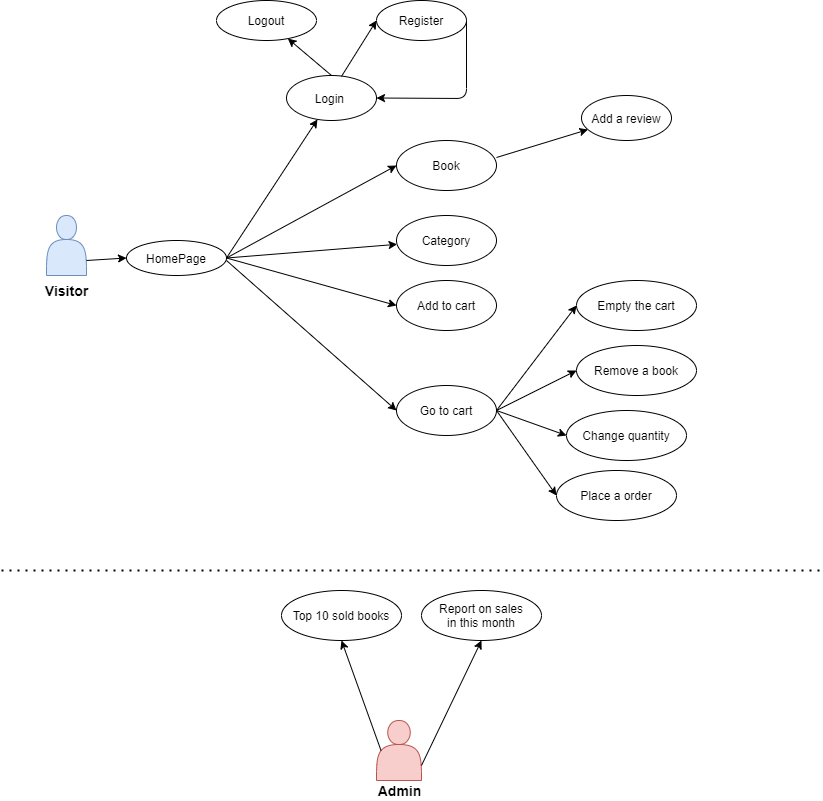
**PART I –DIAGRAM**

There are four diagrams for our project, these diagrams are listed on the next pages.

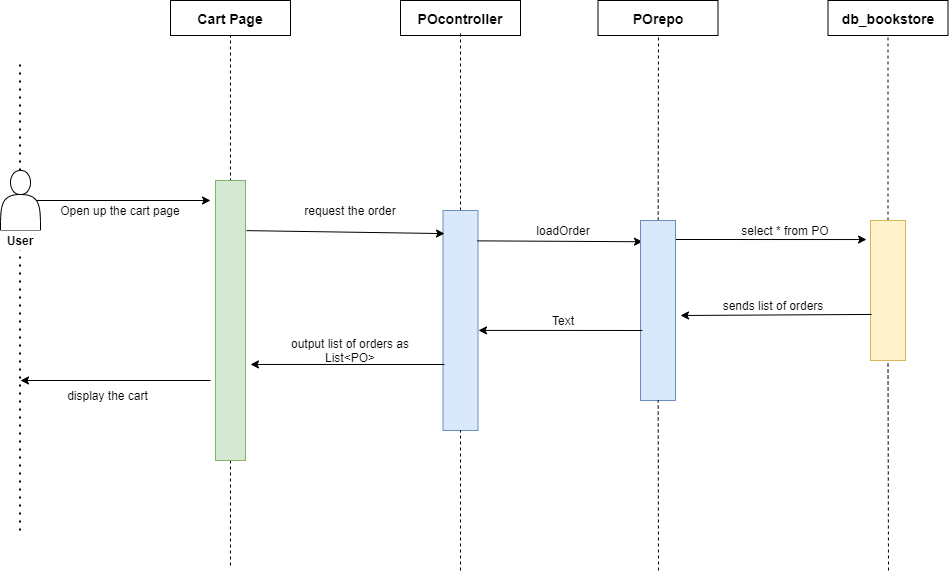
1. **UML Diagram**
2. **Use Case Diagram**
3. **Sequence Diagram #1**
4. **Sequence Diagram #2**

****

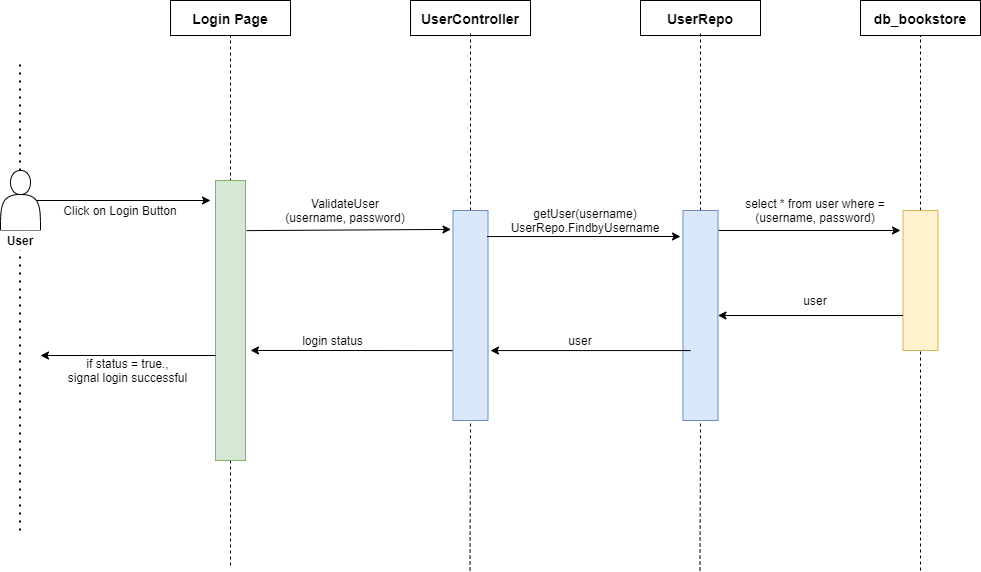
**Figure 1:** UML Diagram

****

**Figure 2:** Use Cases Diagram

****

**Figure 3:** Sequence Diagram 1 **(Visiting the cart)**

****

**Figure 4:** Sequence Diagram 2 **(Attempt to log in)**

**Part II – DESIGN PATTERNS**

Instead of using the servlet and apache tomcat, which we learned during the class, we implemented an application based on Angular and Spring boot for this project. These platforms are structured based on several important design patterns for web development such as MVC and DAO patterns. Here is the list of design patterns that we used.

**I) Design Decisions**

**a) Model-View-Controller (MVC)**

The Model View Controller is the software design pattern commonly used for web development that dividing into three interconnected elements. This pattern consists of three components: model, view, and controller. see Figure 5.

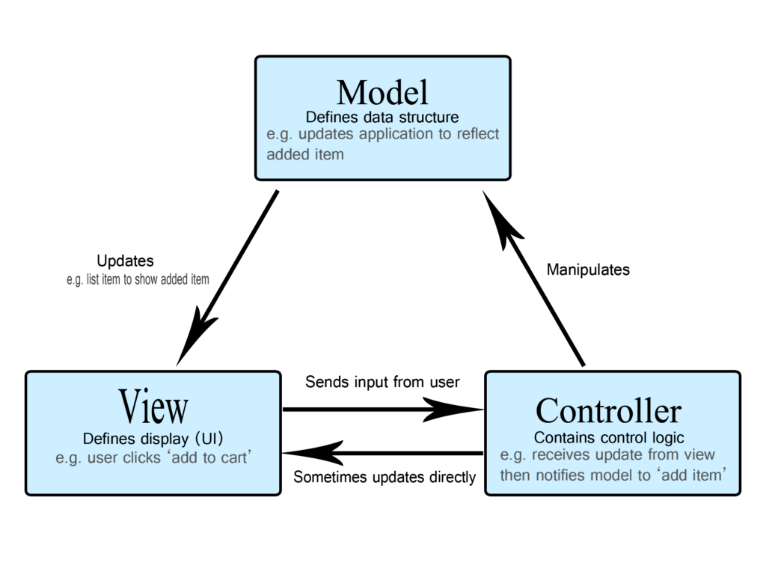


Figure 5: Basic architecture of MVC pattern.

The structure of our bookstore application is designed based on the Model-View-Controller (MVC) pattern, the foundation for responsive, effective, and fast application. The view (front end component in Angular) contains all the web contents that display to the user, controller (ctrl package in the backend), gets the input from the user through view and manipulate with the model. Lastly, the model (db package in the backend) interacts with the database and update the database when needed.

**b) Data Access Object (DAO) Pattern**

In addition to the MVC pattern, we also implemented the Data Access Object (DAO) pattern, which provides the abstract interface to the device. In our application, the DAO object is stored in the model package in the backend. Each model stores the data elements from the database table and acts as an intermediary between the application and the database. The primary advantage of using this pattern including separation between two important parts of an application that can but should not know anything of each other, and which can be expected to evolve frequently and independently, so that detail of the database can be hidden and protected. We provide a service on Angular to provide another step of separation and to enable data exchange between the model and the view

**c) Event-Driven**

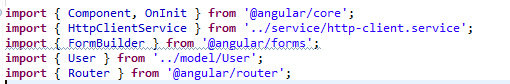


Figure 6: Pre-built libraries from angular is imported on each component’s TypeScript

The event-driven architecture was used on this application, based on an event generated in the view component, the TypeScript controller will call the functions and handle the event by calling various unique functions or the service based on which component is currently viewed. Angular provides pre-built libraries (see figure 6), providing convenience for developers since they do not need to build the procedure from scratch and able to use widely used functions from the Angular library.

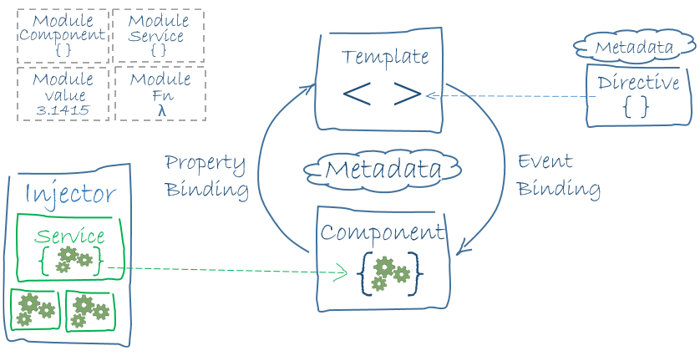
**Part III – IMPLEMENTATIONS**

**I) Implementation platform - Angular and Spring boot**

As mentioned above, we developed the application based on angular and spring boot as development tools for this project. In both the frontend and backend, we focused to make the application easily maintainable (such as not having redundancy) and provide secure access to the bookstore by providing the application in MVC structure.

**html-typescript-service/model-router (Angular)**

The Angular is the framework built and maintained by google used on developing dynamic web and mobile applications. Angular has gain popularity in the past few years by having an advantage. For example, the pre-developed functions (framework) by the angular developer team on TypeScript, which is a superset of JavaScript. This allows keep their code clean (reduce redundancy), reduce the workload, and expect to have better security and higher code quality.

  
Figure 7: The basic structure/concept of the Angular application.

The angular consisting of a module, where the developer defines the entire domain, workflow, and the components that are deployed independently and connecting with the root element. Components are like independent pieces of modules which can be dependency injected into one another.

**model-repo-controller-application (Spring boot)**

The Spring Boot is an open-source and Java-based framework that is used to create the microservices, architecture that developers to develop and deploy the services independently. This convenient framework brought advantages to the developers, provide the pre-built structure that allows the developers able to start effectively without wasting time on preparing and configuring the environment. Spring provides flexible XML configurations, robust batch processing, database transactions, easy workflow, along a wide variety of tools for development.

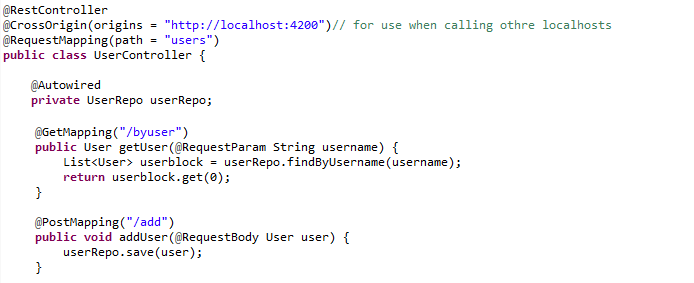


Figure 8: RESTful web services are being applied in each controller.

The Spring boot supports the REST controller programming, annotations such as @RestController, and @GetMapping, @PostMapping is used as a getter and setters to create RESTful web services using Spring MVC. See Figure 8. It takes care of mapping request data to the defined request handler method. Once the response body is generated from the handler method, it converts responses to JSON or XML format.

**II) Decisions**

The following are the implementation decision that we have made with this project to make the application’s architecture is easily maintainable and not having redundant components.

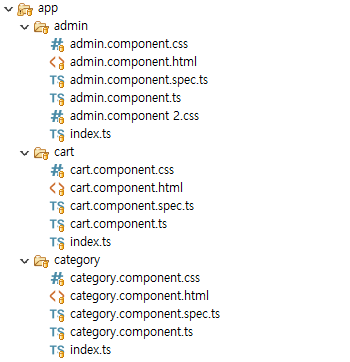


Figure 9: Structure of each component

* Each component in the frontend has its distinct controller (such as *cart.component.ts* for the controller of the cart component), its functionality brought advantage to maintenance as separation brought to reduce the complexity. By separating the code of each part, it can be easily corrected if a problem occurs in a specific part. Figure n shows the structure of each component.
* To protect the database, the database is only accessible via each repository (repo), which contains the statements that are already prepared. These statements are set up for each dedicated task that we need on our webpage.
* Use the model objects that store the data of each column that is found from the table and accessible via getters/setters.

**III) Database**

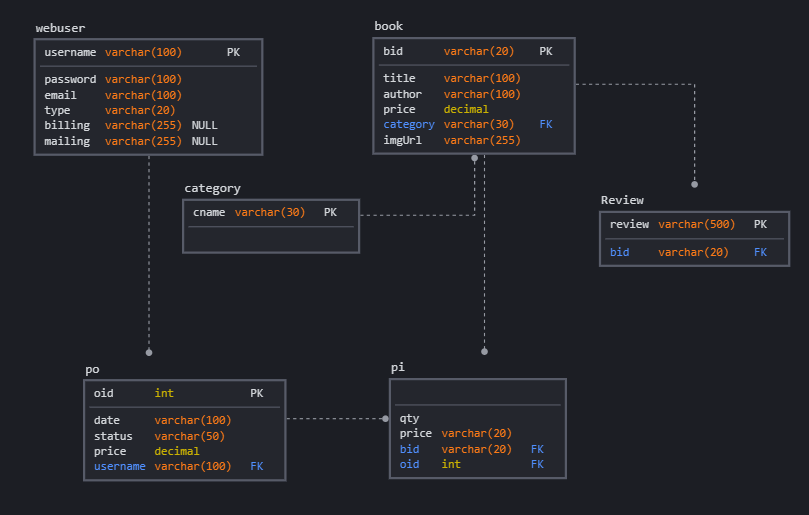


Figure 10: The database structure diagram for this application.

The database for our website is run in the online server at IBM cloud, which stores the database online using the IBM cloud makes to manage the database is more efficiently managed as well as improves security by preventing accidents such as accidental database loss. Figure n shows a sketch of the database that was implemented for this project.

Figure 10 shows the structure of the database that we designed and deployed for our application and deployed on the IBM Cloud.

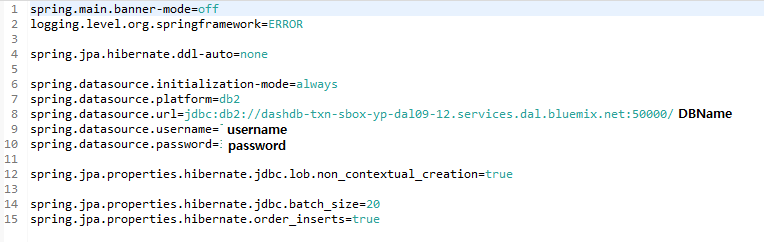


Figure 11: connection is made in application.properties to make a connection between spring boot and db2 database in the cloud.

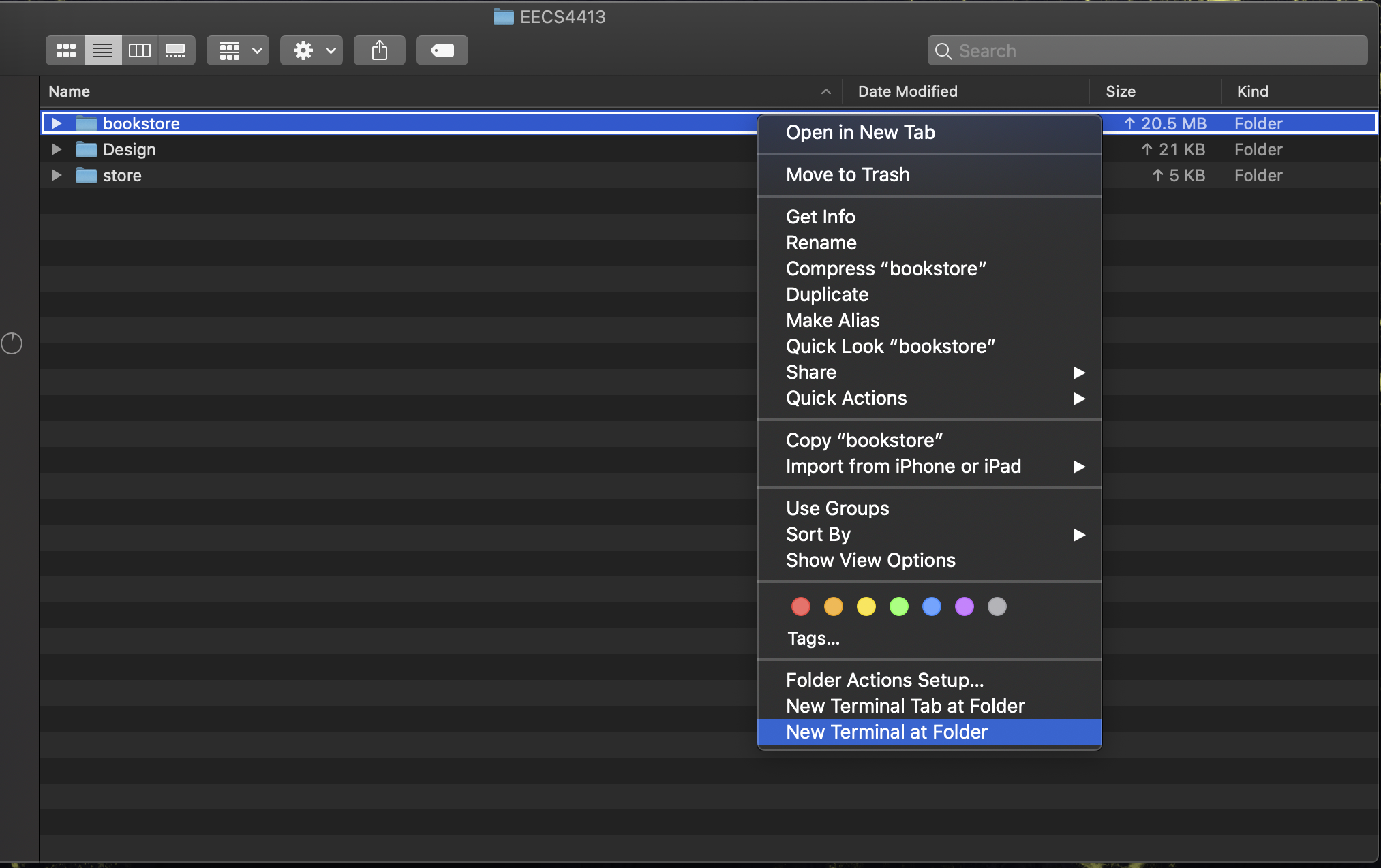
The connection between the database stored in IBM cloud and Spring boot (where handle the events and connecting between the frontend and the database) can be done by setting up the properties in the application.properties file. Figure n shows the way we configure to connect with the database.

**IV) Tradeoffs**

The decision that implementing the frontend using Angular, as the Angular application is developing based on the TypeScript, means the visitors of our website must enable the JavaScript and not able to see the website without it. This reduces compatibility since the application is unusable in environments where JavaScript cannot be able to enable.

While using the Angular and Spring boots provides a convenient and effective developing environment to us, these platform installs many extra dependencies (often not used) that bundled with the platform. This increases the size of the distribution file, makes a large amount of time needed to start the server, increases the space it takes up time and reduces efficiency on the file system. It can also lead to a performance tradeoff since the browser will be busy manipulating the dom elements and will take a lot of the compute resources. This will impact some users with old devices that having older devices, which may take a long processing time to run heavy applications.

To run the Angular application, it is required to install the Angular to function properly, The next section below provides the instruction to run the application that developed based on the Angular application. This can be offset by fully integrating the Angular application to a cloud and generate a URL that can constantly be serving the application.

**V) Instructions to turn on the Angular project in your machine**

1. Make sure You have Angular installed on your machine. If you do not have Angular installed, you can follow the instructions here and install on your machine

<https://www.youtube.com/watch?v=8yL2-ZJfzQ0>

If you are using Windows, you may also need to install the nodejs since windows do not primarily support the terminal.

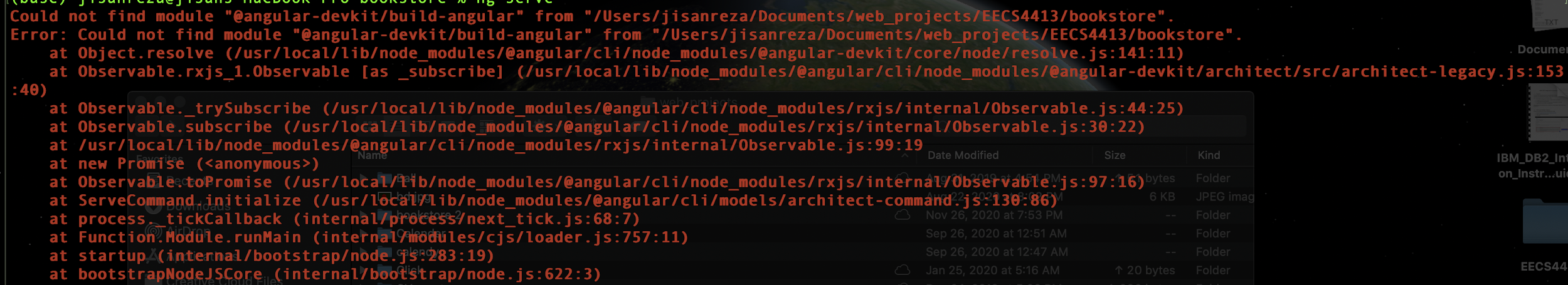
2. In your machine go to the repository and into the folder “bookstore” and open the terminal in the bookstore folder.

cd (project directory)/bookstore

ex) EECS4413-proj/bookstore/

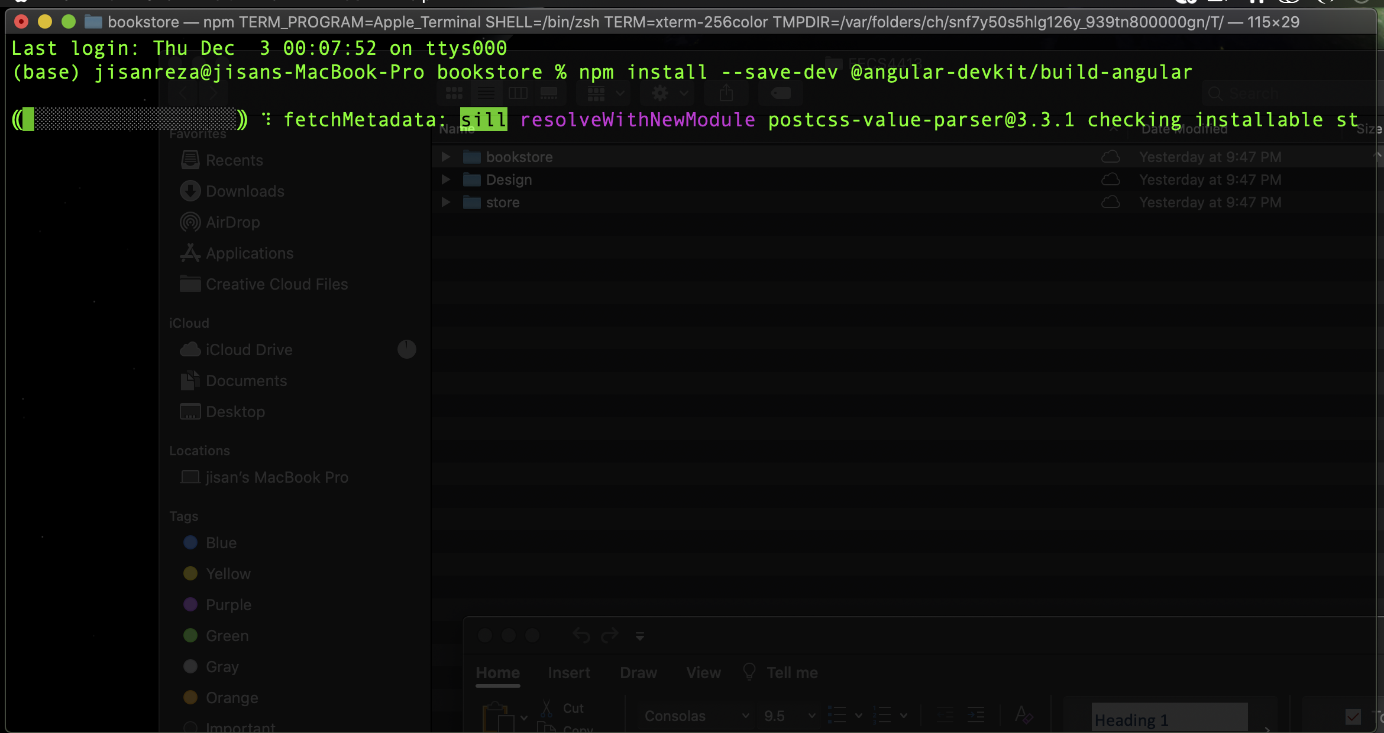
3. Once the terminal and reach the directory that is installed. Type this command in the terminal:

ng serve



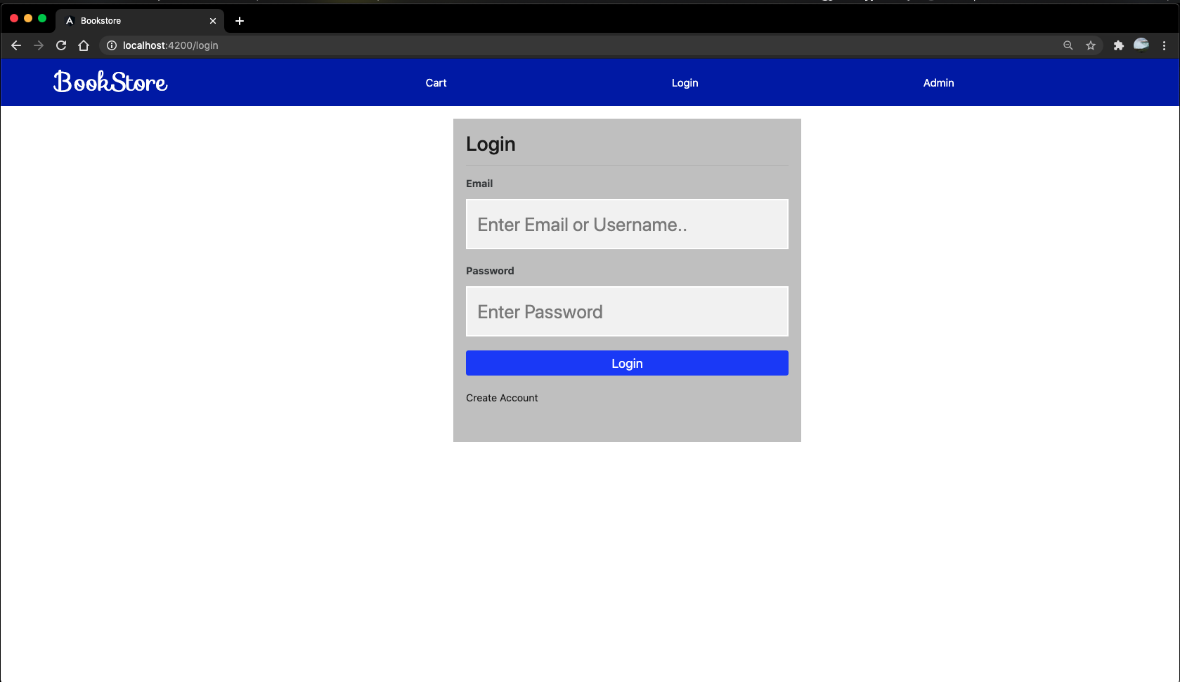
4. You may have error message that starting with “could not find module…”. This is because the version specified in package.json was not in the expected format. to solve this problem, type this command in the terminal.

npm install --save-dev @angular-devkit/build-angular



This command will recompile the angular application and turn the fix the version issue and get the app ready to run on your machine

5. Finally, on your terminal type ng serve again and the Angular Live Development server will run and display the live application on your machine by typing the URL: <http://localhost:4200> on any browser on your device. By this command, the application will start in your browser



**Part III – TEAM MEMBER CONTRIBUTIONS**

While this project is developed based on collaborative efforts, we separated the workload of the project into three parts. There is a person who was mainly responsible for the front-end developing, back-end developing, and deploy the database and design diagrams. Since the COVID-19 brought us restriction on a face-to-face meeting (also note that Youngjin currently resides in South Korea), we mainly communicate with each other through group discord and share the codes through the GitHub repository when any phases of work are done.

Rui was responsible for developing the backend of the project which included the database schema as well as the models, dao, and controllers for the spring boot application to provide various REST services to external clients that may call for it. Also included in this is the testing client to test the web services and verify the REST requests received the correct responses. He also developed the TypeScript (javascript) for the Angular frontend with both the various services, for the Angular application to receive the responses from the server that spring boot would provide, and the various functions that allowed each page of the website to extract and utilize the information provided from the database given the necessary information from the page. GET and POST requests were also handled as well as creating a token system to identify a user and their level of authority on the website on top of validating and sanitizing the inputs. Furthermore, they also developed the dynamically changing cart as well as the payment order process to handle and insert orders into the database. They were also directly involved in integrating these functionalities into the HTML so that the website had events that could occur as well as the various scopes that needed to be managed. He was also the main tester for the website testing inputs, outputs for angular, spring boot, and the (local and cloud) database to ensure everything functioned as per the requirements.

Farhan was mainly responsible for delivering the frontend of our website. He designed most pages on our webpage (in Angular). He also contributed to the backend by helping Rui in making the connection between the frontend and backend.

Youngjin was responsible for check the application complies with the design patterns that are commonly used in web development, and design and deploying the database in the IBM cloud, and working on the documents (such as UML diagrams). He also contributed and gave some advice to other team members and did help for the improvements in the frontend if requested.

**Signatures**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Youngjin Ko | Farhan Latif | Rui Yang Hwang |