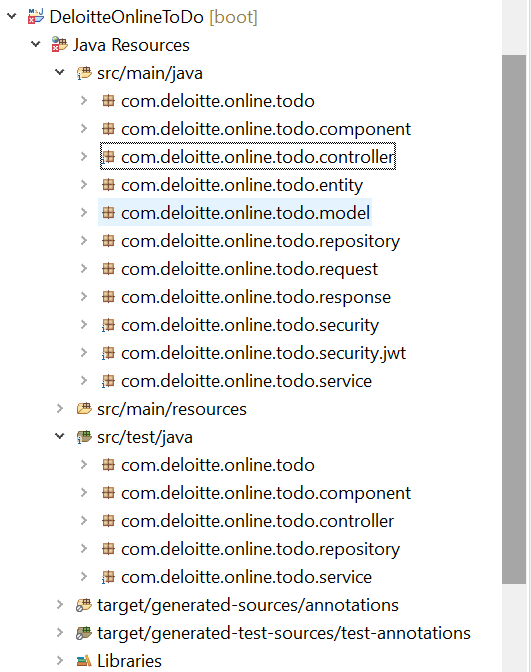
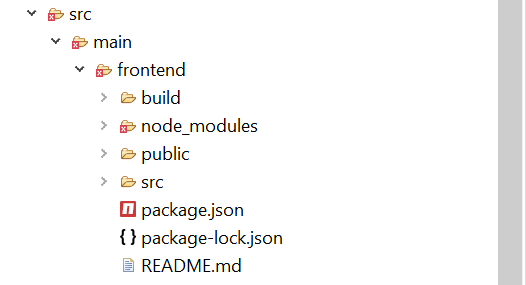
I analysed the assignment and segregated into two parts:-

1. Java, Spring Boot, MySQL db (I will attach the schema) for backend side. I have done as much Junit test cases I could within stipulated time period; however that part could be enhanced more.

Project Structure

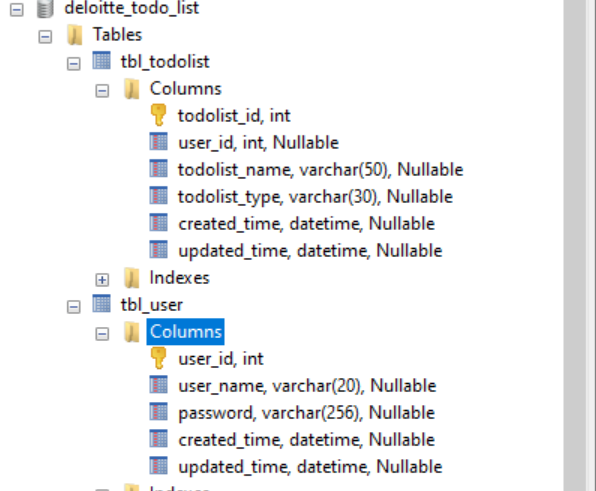


1. React js for Front end.

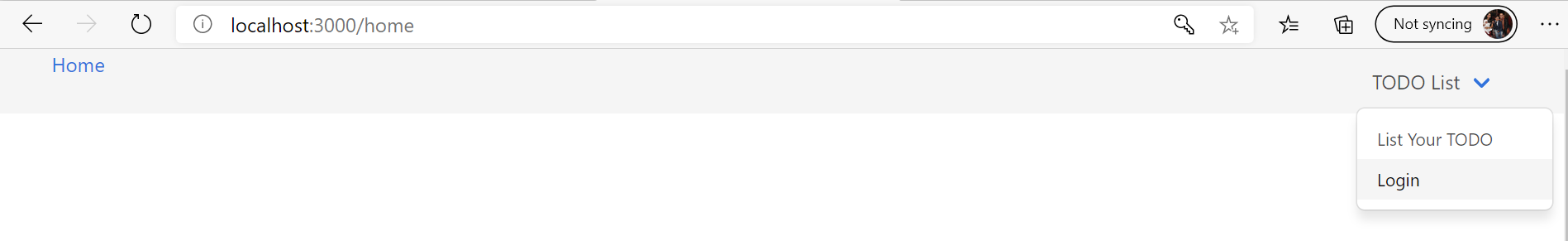


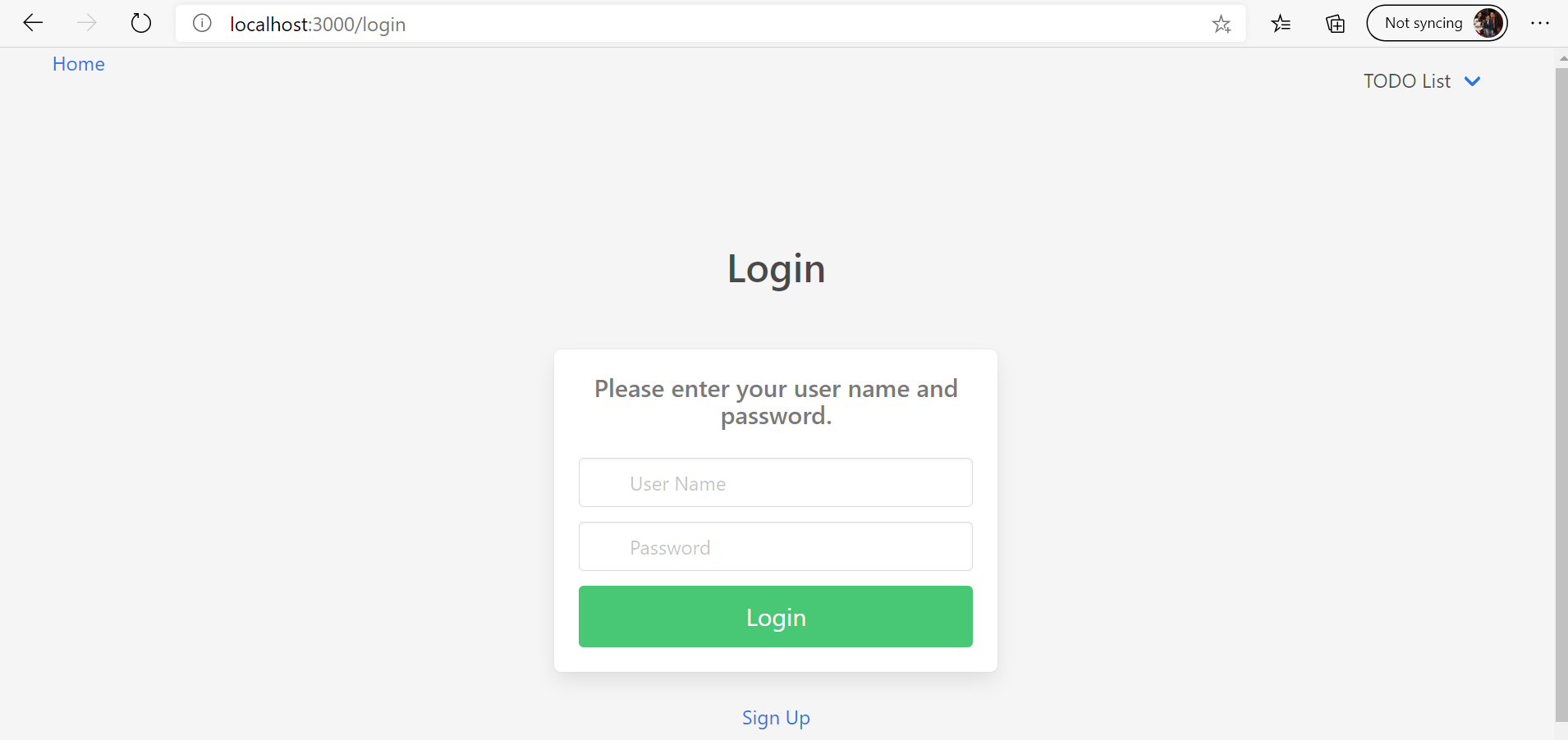
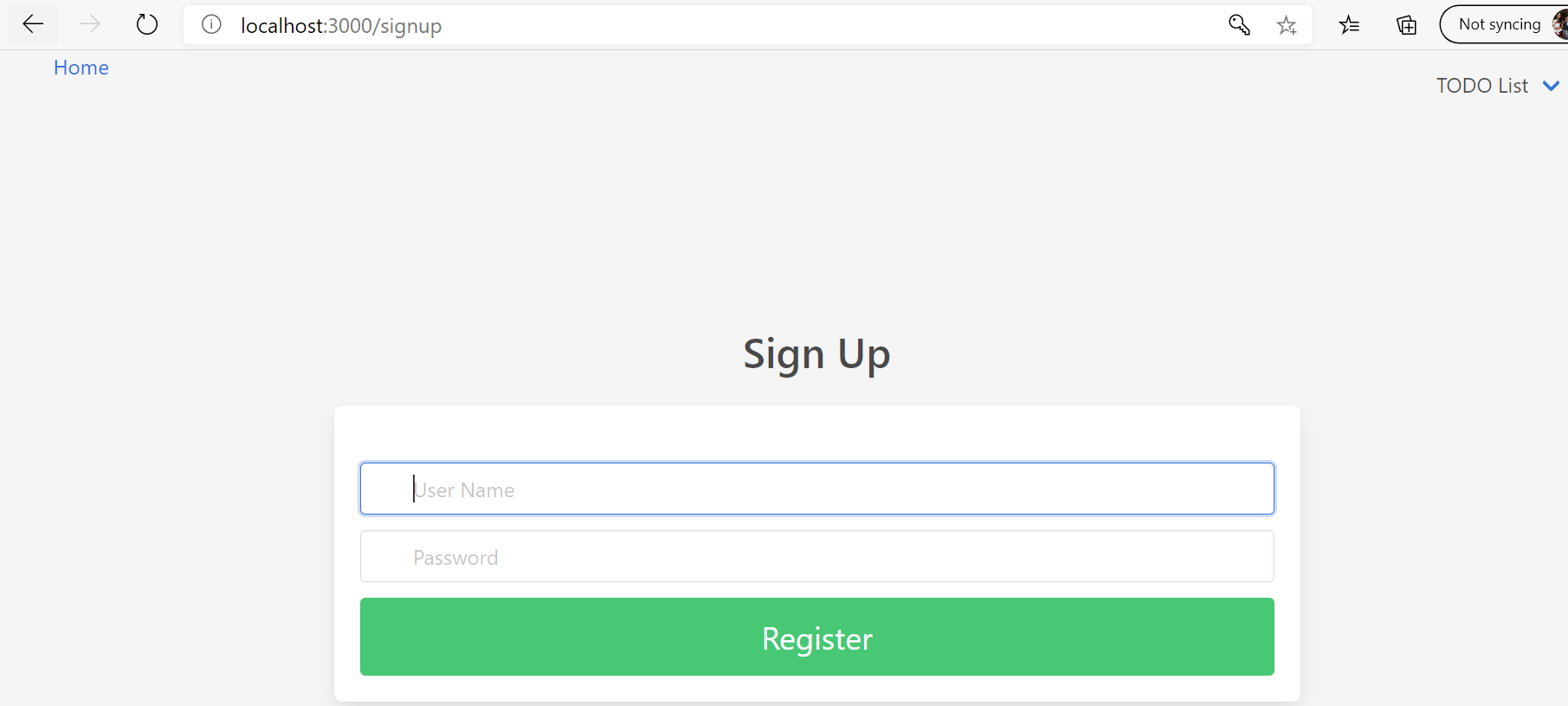
**Database :- deloitte\_todo\_list**

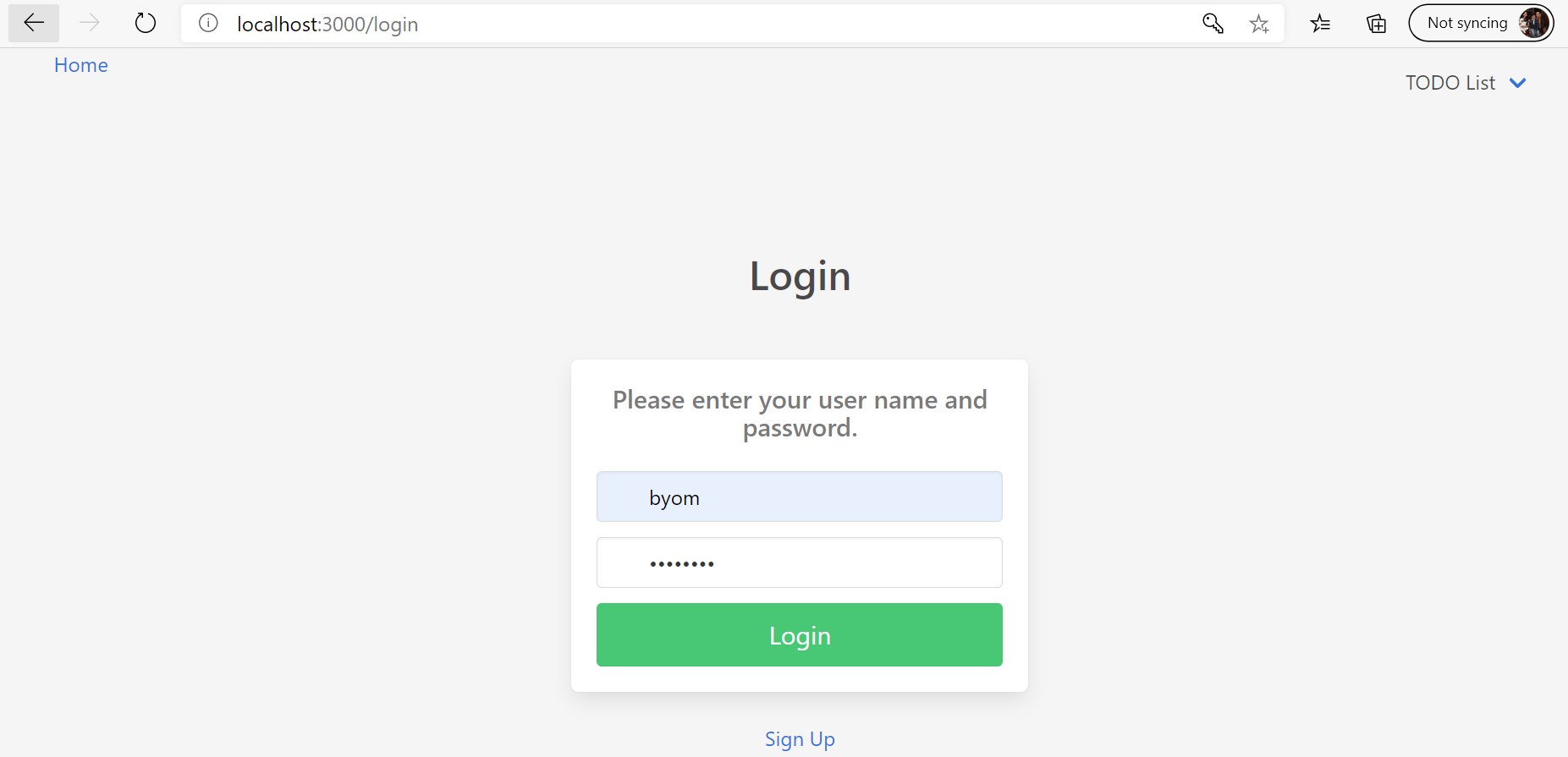
Just to keep it simple, I have created two tables as given below.

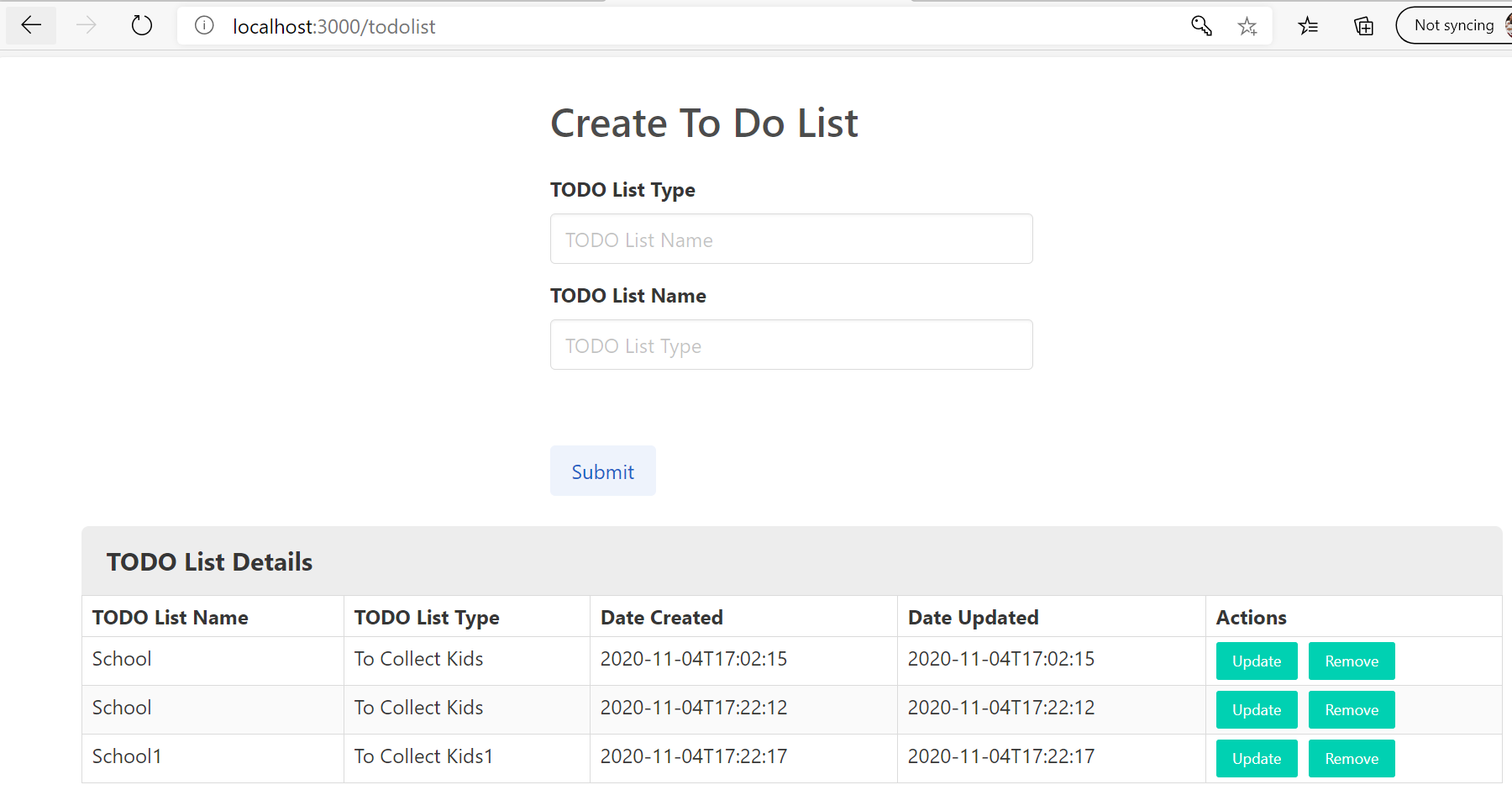


**For Front-End**



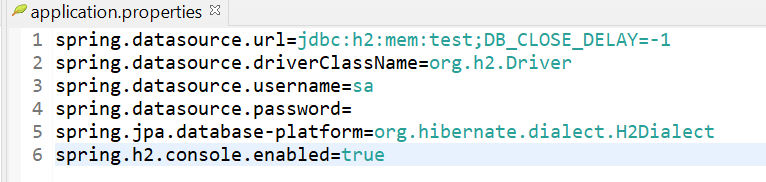






**For Junit Test:-**

I have used in memory db i.e H2 for the testing which gives a db call. So for that I have added application.properties file in test/resources folder.



**Why certain approaches were used, why others were not selected**

As per the assignment we need to create a war file and that to be deployed into Tomcat(7/8). I have added the maven plugin that would generate the war file which will include the reactjs files. However, I would strongly recommend, we should to let the components (front-end & back-end) separate to make it more loosely coupled and let the Spring Boot REST API call to happen to the frontend. In that way it would be quite easier to maintenance.

**Any design patterns used**

There are various patterns I have included here with this implementation. Singleton is one of them and part of the software design principle i.e. SOLID also been implemented wherever required.

**Anything extra you would have done give more time**

I could have finished all the Junit test cases and the front-end bit including the test cases for all the components I have written for the frontend. Also the date and time could have formatted.

1. User can sign in using unique login and password securely (this can be hard coded to a default user list, at list one user e.g. with username: test, password: pwd123) **– I have used Spring Security for securely log in to the db.**
2. User can view her/his task list **– Yes**
3. User can add/remove task **– Yes**
4. All changes can be persistent to allow view them in next sign in by the same user **– Yes.**
5. Each task should display the date of last updates and description **– Yes**
6. User can check/uncheck any task on their list **– Yes**
7. Consider performance **– Yes**

**How To Run**

Create the database (MySQL & schema provided.) with two tables.

DB Name: deloitte\_todo\_list

Two Tables : tbl\_user , tbl\_todolist

I have already created two users, however on frontend, user can also be created, passwords are hashed before storing to the db.

|  |  |
| --- | --- |
| User | Password |
| test | test123 |
| byom | password |

Rather going by war file option, I would suggest running the demo software separately. As we have written the software with Spring Boot, Tomcat is embedded with it, so don’t need to deploy the jar/war separately. Currently the packaging of the java classes are done for .jar and that can be changed by modifying the <packaging>jar</packaging> to war.

**For FrontEnd - <workspace on your drive>\DeloitteOnlineToDo\src\main\frontend>**

**<workspace on your drive>\DeloitteOnlineToDo\src\main\frontend> npm i**

**After the installation done, run the command:-**

**<workspace on your drive>\DeloitteOnlineToDo\src\main\frontend> npm start**

**Note:** I would assume npm is set up in your computer. If it’s not then please install node.js and npm will be installing automatically. Here is a quick one to follow. It’s very easy. Actually I am learning reactjs myself, so thought to use as a front end here. Hope that’s ok. <https://www.codecademy.com/articles/react-setup-i>

Frontend will run on port 3000

Spring boot application will run on 8080.

Once both the server up, open a browser and paste <http://localhost:3000/> to start the application and click on TODO list Menu and follow accordingly to browse through the TODO LIST functionalities.