Documentation

se-02-team-24 Aabishkar and Aoge

March 23 2021

For this weeks work in the frontend we have converted the html pages into REACT and did some more sytling to the webpages such that it looks more attractive. We also added the HomePage, an instruction page and a Gamepage which opens after a user successfully logins into the game with their credentials.

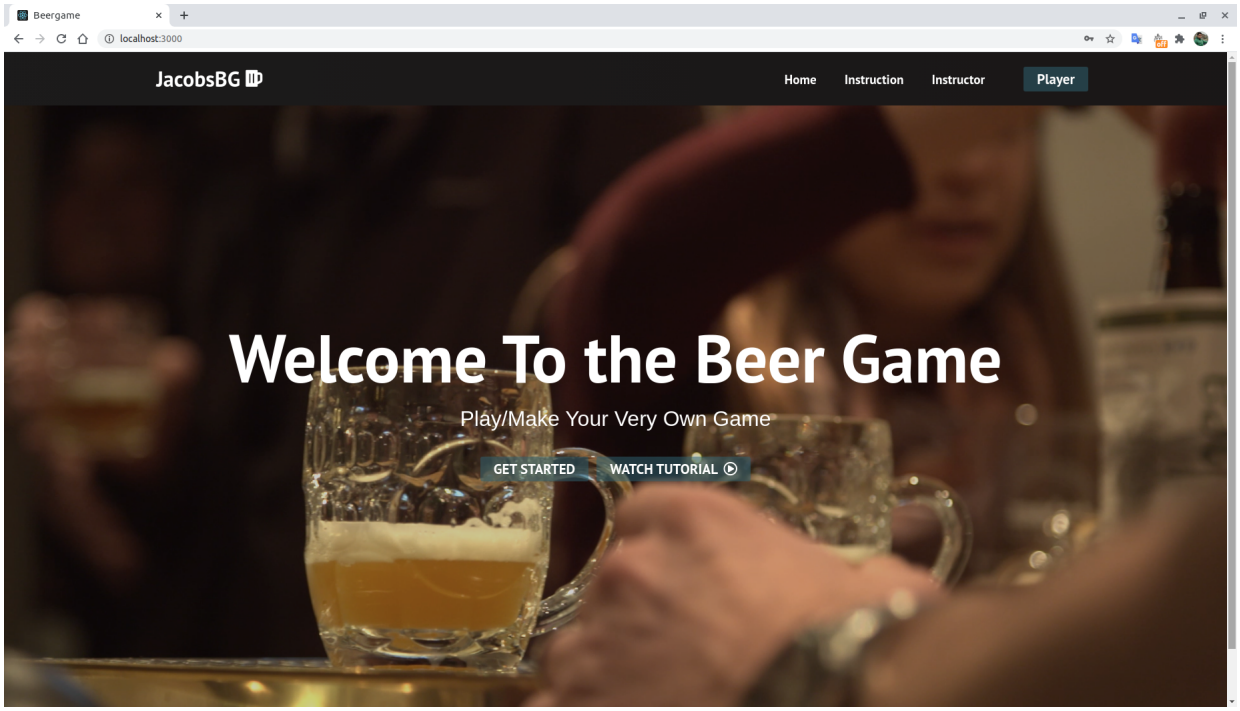

Figure 1: HomePage

Once you run the react app using ‘npm start‘ and goto ”localhost:3000” you come across this HomePage which contains a navigation bar and the game logo. Through this navigation bar a user can go to the instruction page or to the login and register page for Players and Instructors which have been created separately.

The instruction page contains the basic info of the game which can be seen in Figure 2. For the login and register pages we have created separated pages for both Player and Instructor. Players should click the Players tab in the navigation bar and do login/signup and Instructors should click the Instructors tab in the navigation bar and do login/setup or else they will face problems.

In Figure 3 you can see the layout of a login page where users must enter their email and password and press the login button. If the user is registered in the system in respective tables (Players and Instructors) then a new page will open displaying login successfull as seen in Figure 5. If the user enters a wrong password then an alert will pop up saying ”Wrong Password!!”. Similarly if the user is not registered in the system then a pop up will come saying ”the user is not registered!!!”. All of this is done in the backend side or the flask server.

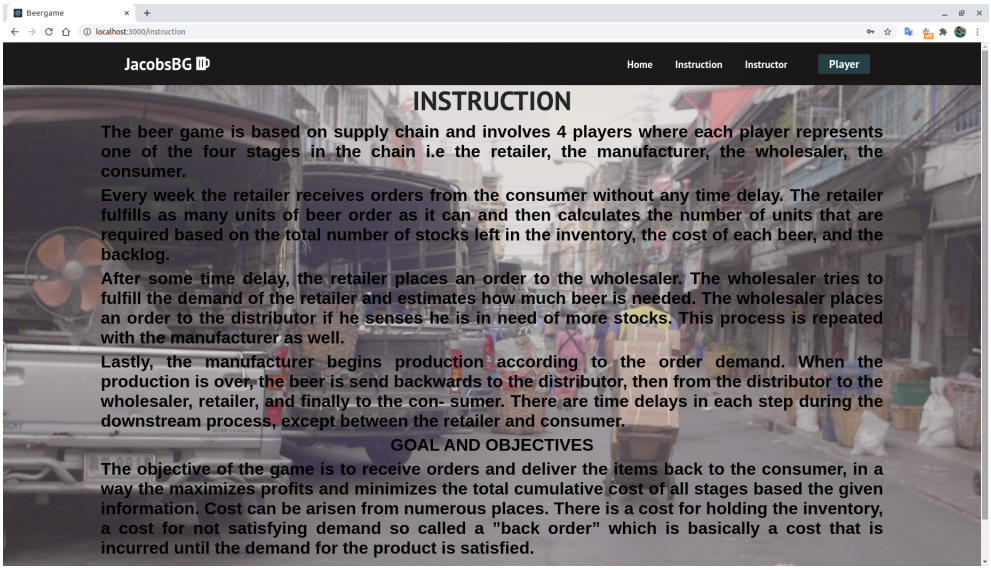

Figure 2: Instruction Page

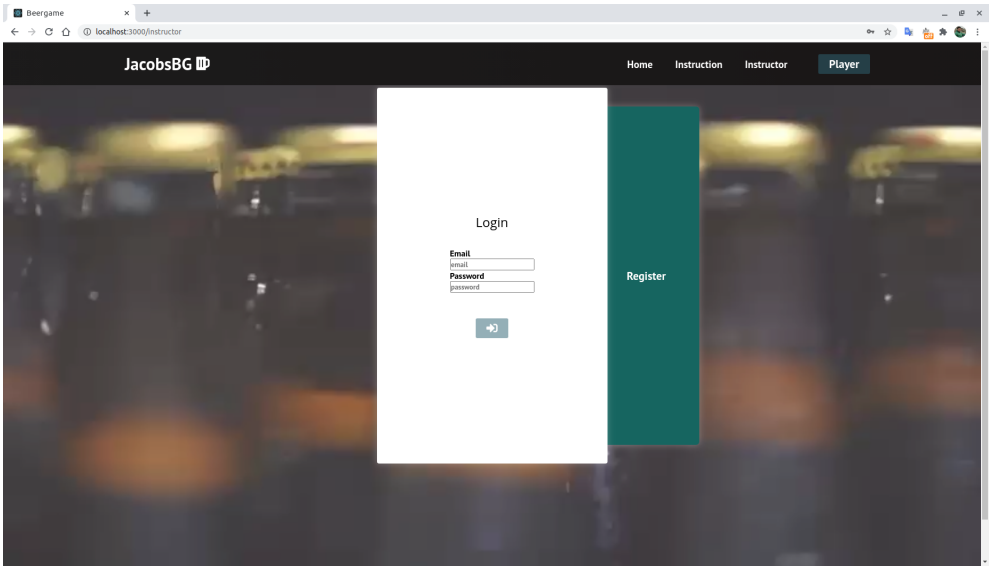

Figure 3: Login Page

In Figure 4 we can see the layout of the Registration page where a user can enter fullname, email, password and confirm password and then click on submit button. At first if password and confirm password are not same then an alert will be popped up saying the passwords are not same. Secondly if the email is already registered in the database then also a error will pop up saying "Failure" and similarly for other errors as well a "Failure" alert will pop up. On the other hand, if the instructor is present in the database then it says success and will be redirected to the page in figure 5 which later should contain the player info, lobby, game monitoring that are mentioned in the requirement manual which we are following.

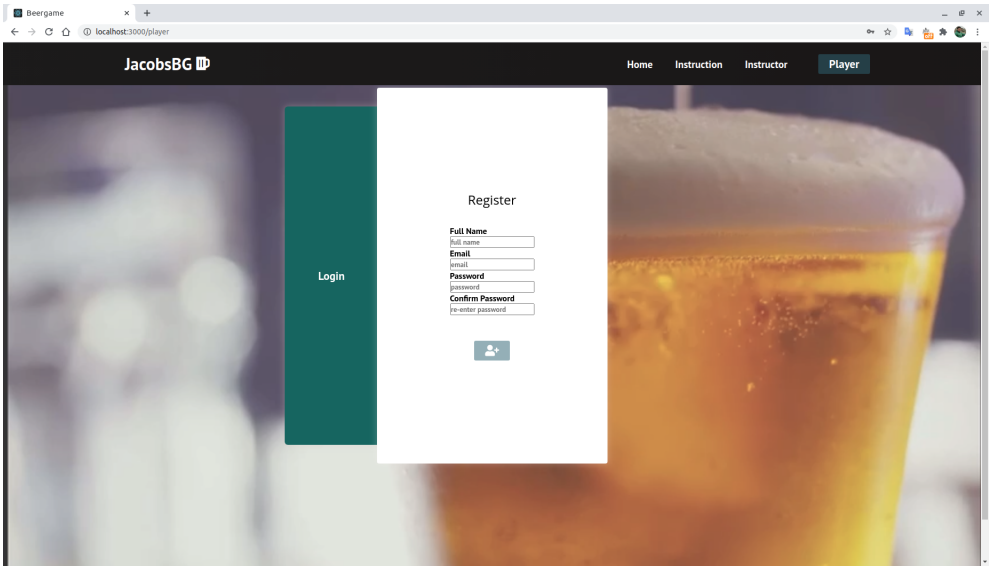

Figure 4: Register Page

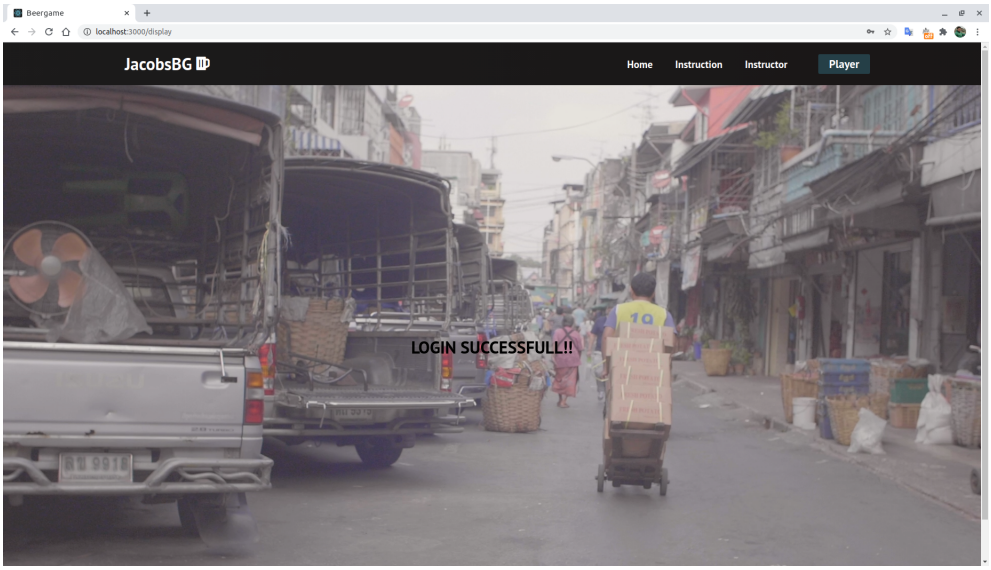

Figure 5: Success page after instructor Login

For the player, after successful logging in they will be redirected to the role choosing page (in figure 6) which player should scroll down to choose a role shown in figure 7. the variety of roles for player to choose are the retailer, wholesaler, distributor, and the factory. The info about each role are written right behind its name. After player choosing the role, they should be redirected to game page that is dependent on which role they have chosen.

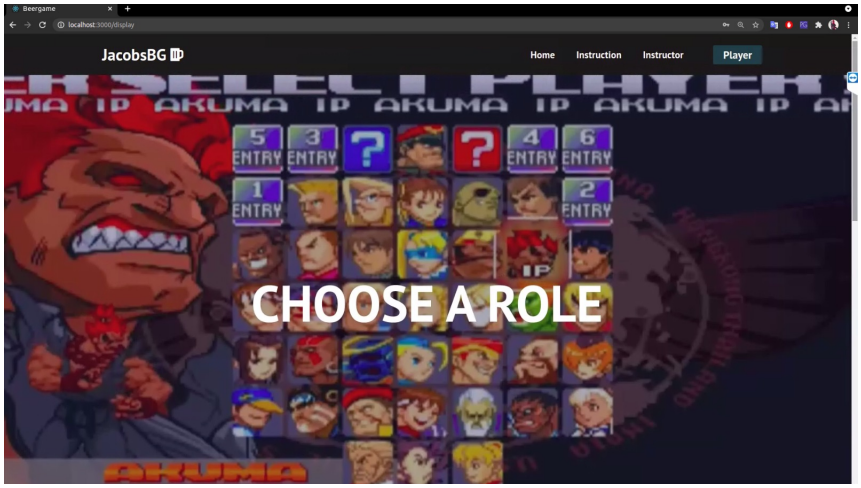

Figure 6: role choosing page after player successfully Login

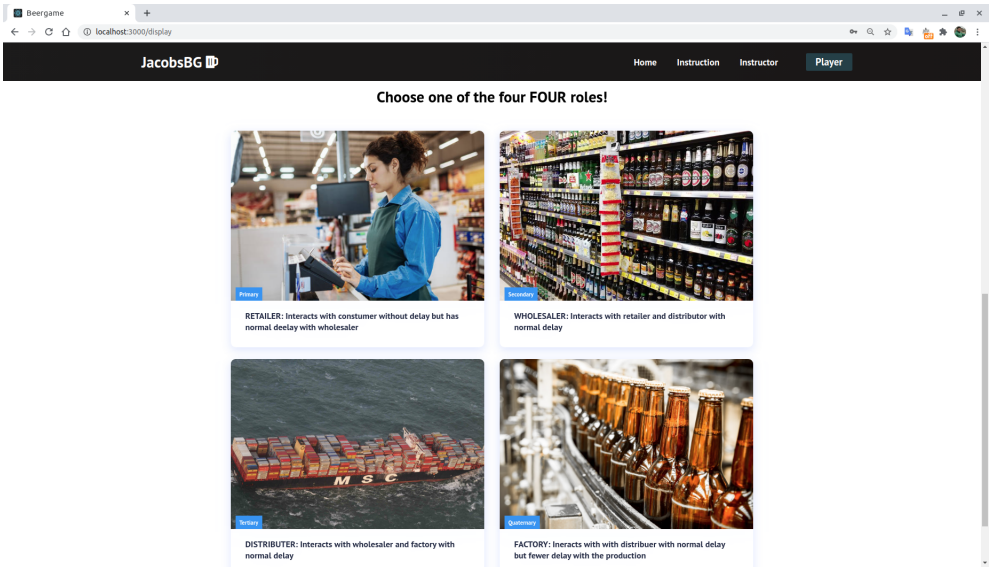

Figure 7: roles in the role choosing page

For the backend side of the code we have created a flask server and connected it to our test database which has been created using sqlite.The test database is stored in the file called "beer_game.db" which is created when we run 'yoyo apply' in the beginning. We can look at the datas of the database by going through the steps done in Figure 10. For testing the backend/database one can look through the steps of Figure 9. The backend testing has been created using unittest framework and it supplies random data (name,email,password) for Player and Instructor registration. Also the passwords which are received from the frontend are hashed and stored in the database for safety purposes so we can see random characters in the password column. While logging in the hashed password is received from the database, then it is unhashed and compared with the user entered password to check for validity.

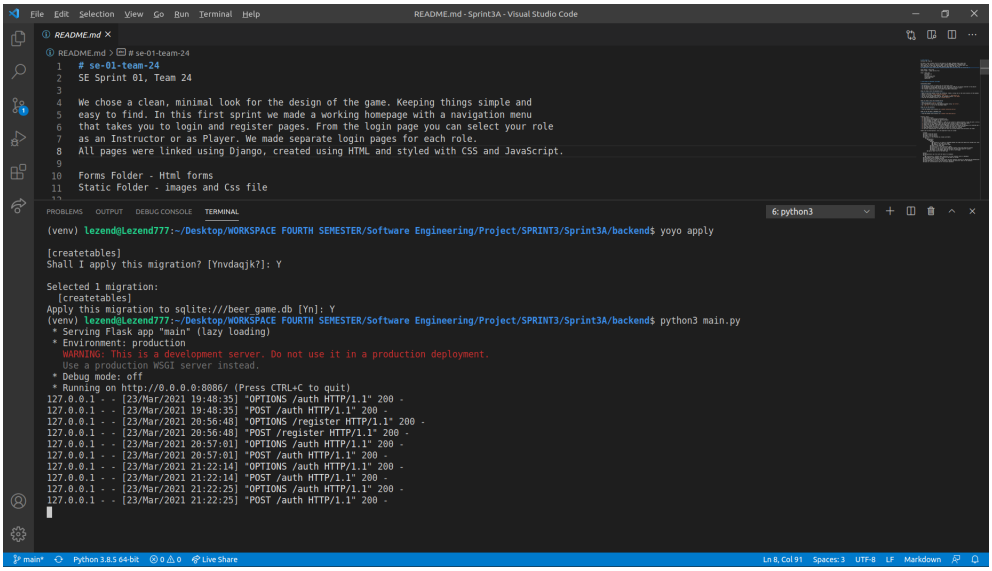

Figure 8: Starting Flask Server

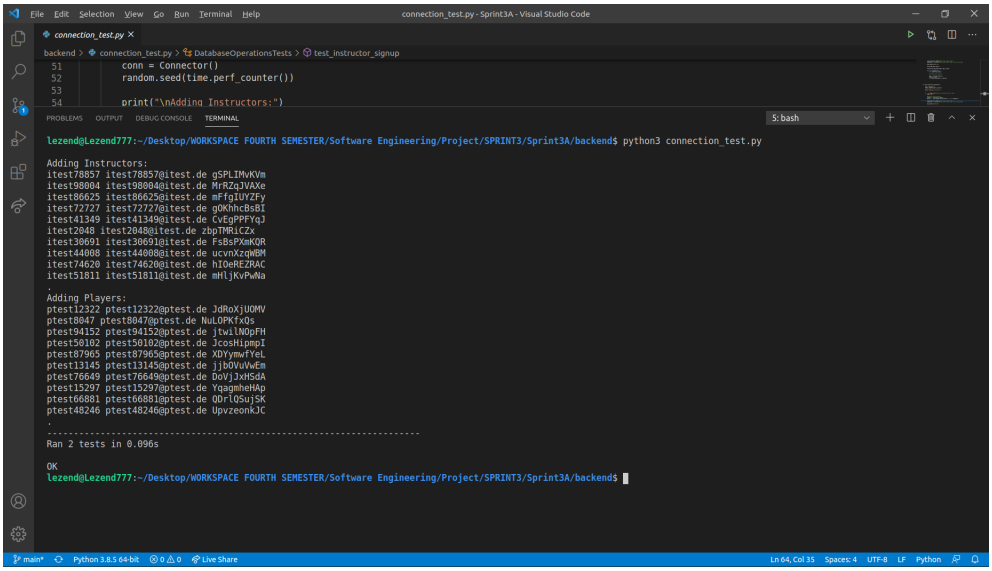

Figure 9: Running Backend Testing

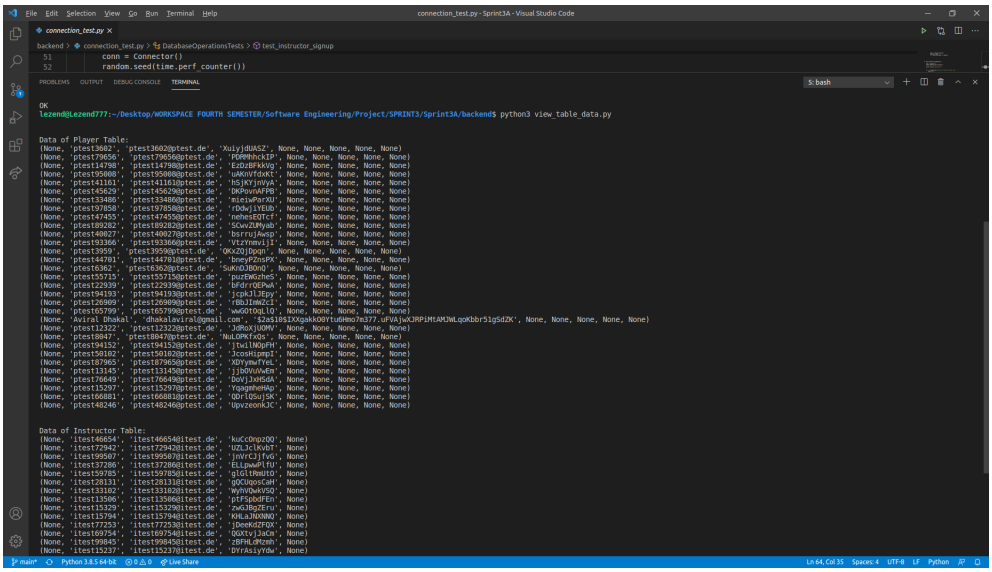

Figure 10: Viewing database datas