

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 styling 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 logs in into the game with their credentials.

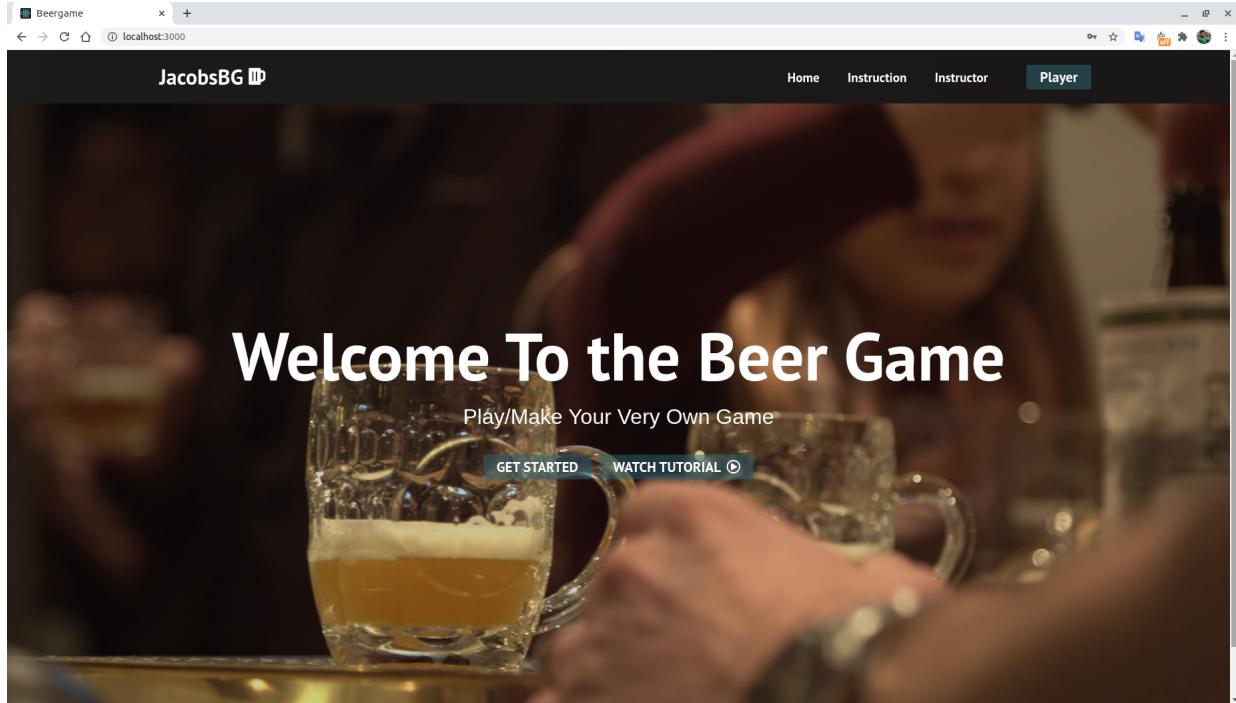


Figure 1: HomePage

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.

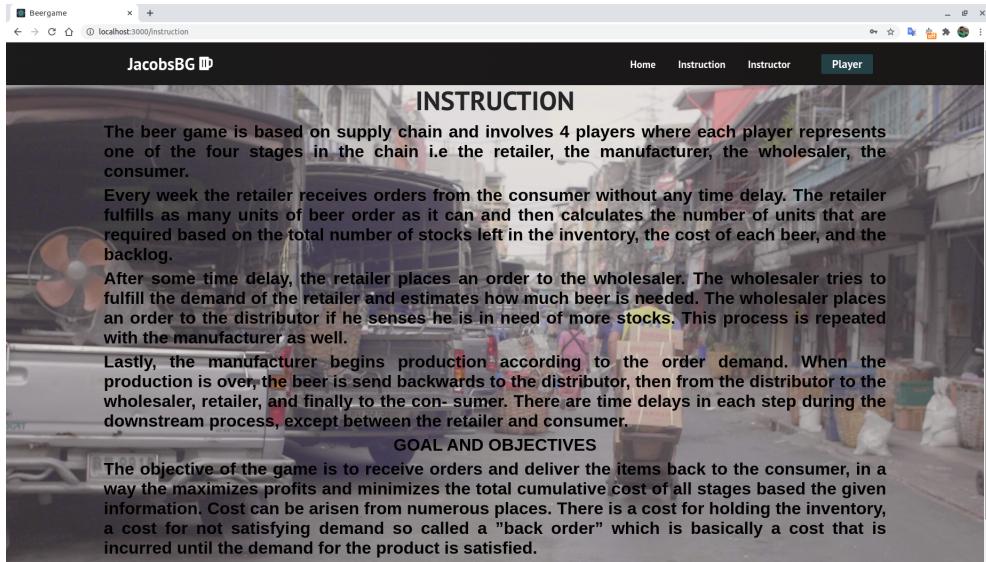


Figure 2: Instruction Page

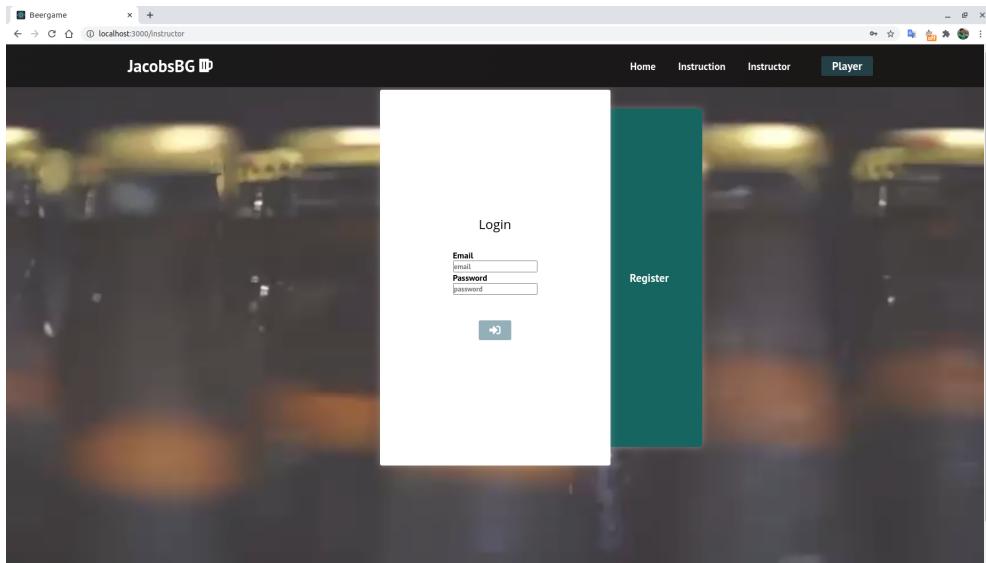


Figure 3: Login Page

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.

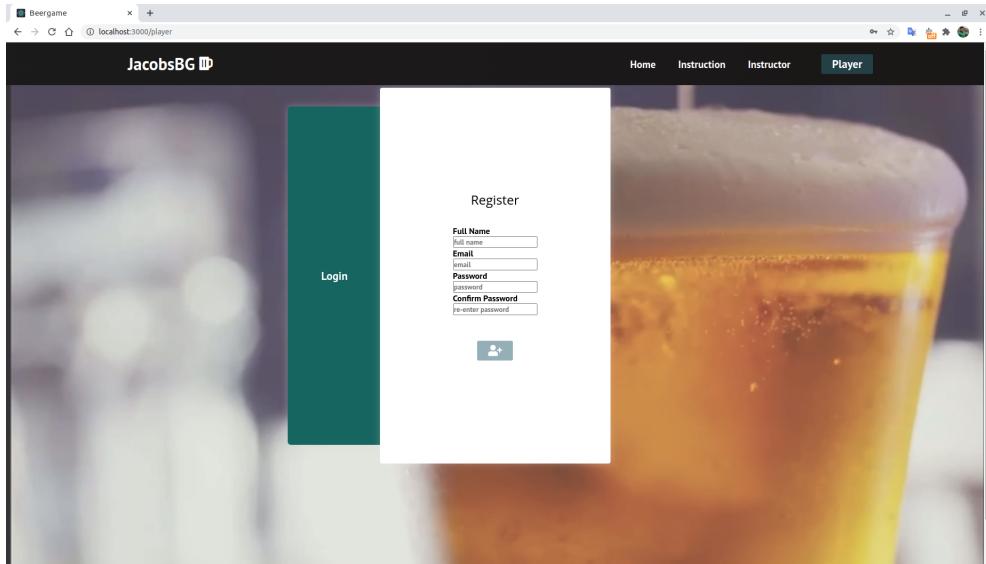


Figure 4: Register Page

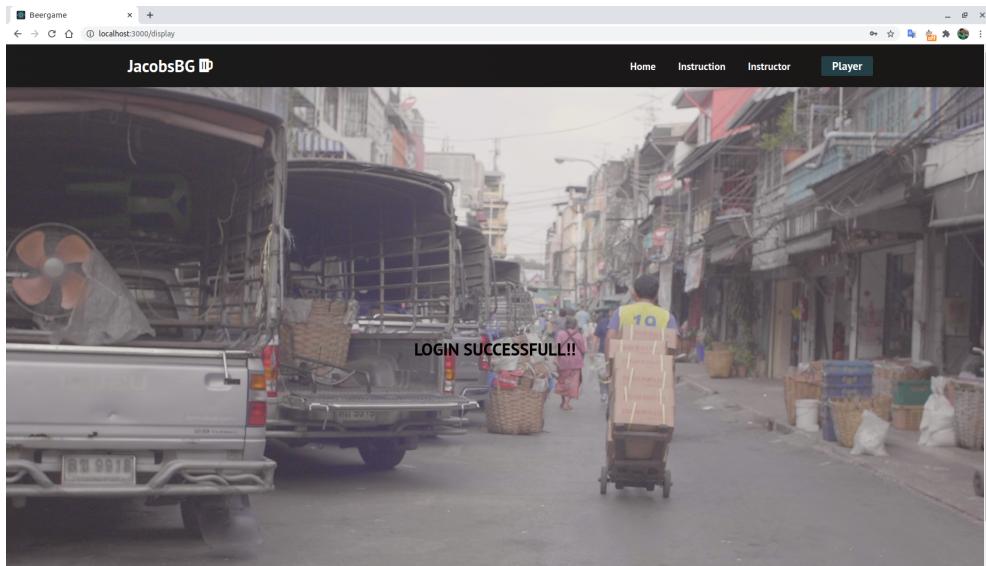


Figure 5: Success page after instructor Login

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.

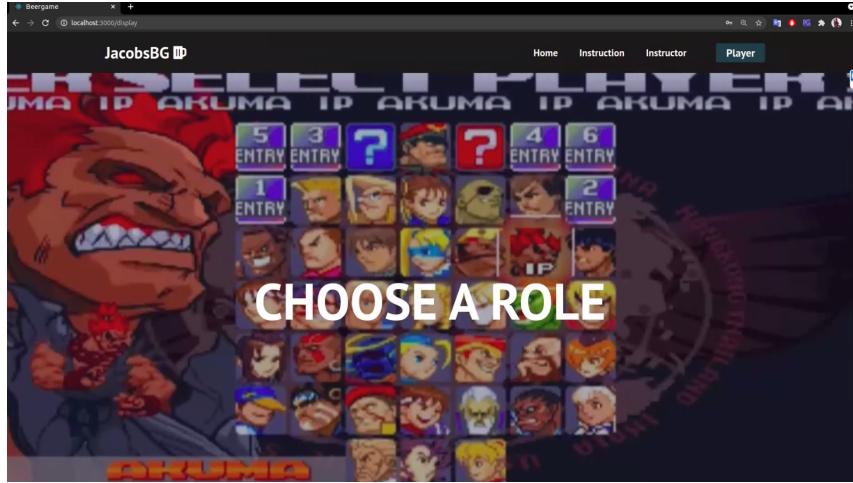


Figure 6: role choosing page after player successfully Login

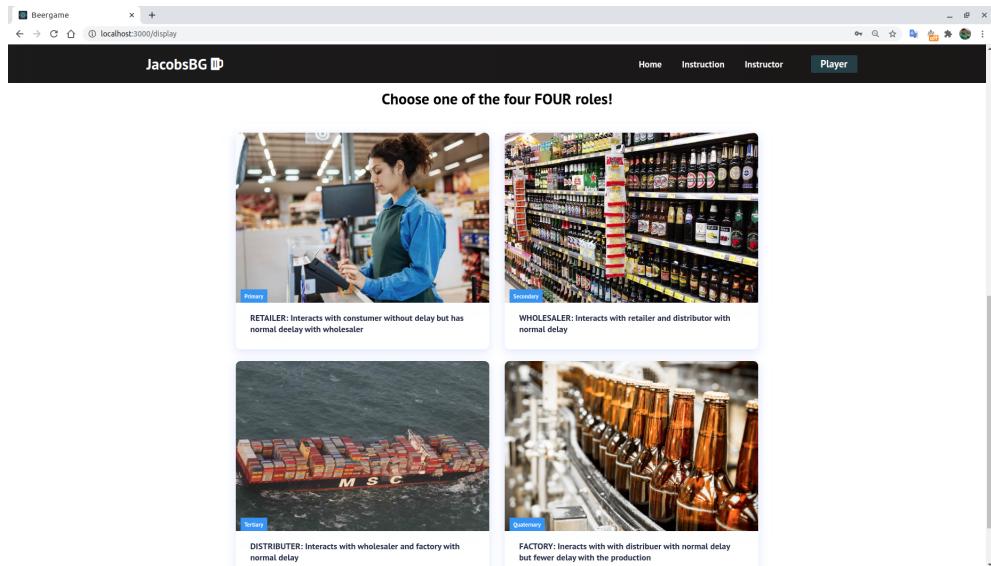


Figure 7: roles in the role choosing page

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 'yo yo 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.

The screenshot shows a Visual Studio Code interface. The left sidebar has a tree view with 'README.md' selected. The main editor area shows the content of README.md. A terminal tab is open with the command 'yojo apply' being run. The output shows migrations being applied to the database.

```

README.md - Sprint3A - Visual Studio Code
File Edit Selection View Go Run Terminal Help
README.md > # se-01-team-24
1 # se-01-team-24
2 SE Sprint 01, Team 24
3
4 We chose a clean, minimal look for the design of the game. Keeping things simple and
5 easy to find. In this first sprint we made a working homepage with a navigation menu
6 that takes you to login and register pages. From the login page you can select your role
7 as an Instructor or as Player. We made separate login pages for each role.
8 All pages were linked using Django, created using HTML and styled with CSS and JavaScript.
9
10 Forms Folder - HTML forms
11 Static Folder - images and Css file
12

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[venv] lezende@Lezend777:~/Desktop/WORKSPACE FOURTH SEMESTER/Software Engineering/Project/SPRINT3/Sprint3A/backend$ yojo apply
[createtables]
Shall I apply this migration? [Ynvdqajk]: Y
Selected 1 migration:
[createtables]
Apply this migration to sqlite:///beer_game.db [Yn]: Y
[venv] lezende@Lezend777:~/Desktop/WORKSPACE FOURTH SEMESTER/Software Engineering/Project/SPRINT3/Sprint3A/backend$ python3 main.py
* Serving Flask app "main" (lazy loading)
* Environment: development
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
127.0.0.1 - - [23/Mar/2021 19:48:35] "OPTIONS /auth HTTP/1.1" 200 -
127.0.0.1 - - [23/Mar/2021 19:48:35] "POST /auth HTTP/1.1" 200 -
127.0.0.1 - - [23/Mar/2021 20:56:48] "OPTIONS /register HTTP/1.1" 200 -
127.0.0.1 - - [23/Mar/2021 20:56:48] "POST /register HTTP/1.1" 200 -
127.0.0.1 - - [23/Mar/2021 20:57:01] "OPTIONS /auth HTTP/1.1" 200 -
127.0.0.1 - - [23/Mar/2021 20:57:01] "POST /auth HTTP/1.1" 200 -
127.0.0.1 - - [23/Mar/2021 21:22:14] "OPTIONS /auth HTTP/1.1" 200 -
127.0.0.1 - - [23/Mar/2021 21:22:14] "POST /auth HTTP/1.1" 200 -
127.0.0.1 - - [23/Mar/2021 21:22:25] "OPTIONS /auth HTTP/1.1" 200 -
127.0.0.1 - - [23/Mar/2021 21:22:25] "POST /auth HTTP/1.1" 200 -

```

Figure 8: Starting Flask Server

The screenshot shows a Visual Studio Code interface. The left sidebar has a tree view with 'connection_test.py' selected. The main editor area shows the content of connection_test.py. A terminal tab is open with the command 'python3 connection_test.py' being run. The output shows the test script running and adding instructors and players to the database.

```

connection_test.py - Sprint3A - Visual Studio Code
File Edit Selection View Go Run Terminal Help
connection_test.py > DatabaseOperationsTests > test_instructor.signup
51     conn = Connector()
52     random.seed(time.perf_counter())
53
54     print("\nAdding Instructors:")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[venv] lezende@Lezend777:~/Desktop/WORKSPACE FOURTH SEMESTER/Software Engineering/Project/SPRINT3/Sprint3A/backend$ python3 connection_test.py
Adding Instructors:
1test178804 istest178804@test.de MR7gjVAXxe
1test198084 istest198084@test.de mR7gjVAXxe
1test86025 istest86025@test.de mfqfIUYZry
1test7277 istest7277@test.de gOKhncB8sDI
1test41349 istest41349@test.de CVgPPFyJ
1test128901 istest128901@test.de PBePXwKQR
1test128901 istest128901@test.de PBePXwKQR
1test44098 istest44098@test.de ucvnXzqM9M
1test74620 istest74620@test.de hDoeREZRAc
1test51810 istest51810@test.de mmljKvWNa

Adding Players:
1test12322 istest12322@test.de JDxQxJ0Mv
1test8047 istest8047@test.de NuLOPfxOs
1test94152 istest94152@test.de jtWLmOpPH
1test59106 istest59106@test.de cosHpmP1
1test13145 istest13145@test.de jzbUvWdL
1test13145 istest13145@test.de jzbUvWdL
1test76648 istest76648@test.de D0VjxHsA
1test15297 istest15297@test.de YqagmheM4p
1test15297 istest15297@test.de ODrLqsj3K
1test40246 istest40246@test.de OpvzeomkC
-----
Ran 2 tests in 0.096s
OK
[venv] lezende@Lezend777:~/Desktop/WORKSPACE FOURTH SEMESTER/Software Engineering/Project/SPRINT3/Sprint3A/backend$ 

```

Figure 9: Running Backend Testing

The screenshot shows a Visual Studio Code window with two tabs open: `connection_test.py` and `connection_test.py`. The `connection_test.py` tab contains Python code for connecting to a database and performing operations. The `connection_test.py` tab shows the output of running the script, displaying data from two tables: `Player` and `Instructor`.

```

connection_test.py - Sprint3A - Visual Studio Code
File Edit Selection View Go Run Terminal Help
connection_test.py % DatabaseOperationsTests > test_instructor_signup
backend: connection_test.py
31 | conn = Connector()
32 | random.seed(time.perf_counter())
33 |
34 | OK
35 | lexend@Lexend777:~/Desktop/WORKSPACE FOURTH SEMESTER/Software Engineering/Project/SPMINTS/sprint3A/backends python3 view_table_data.py
36 |
37 Data of Player Table:
38 | (None, ptest13602, 'ptest13602@test.de', 'XuiyiduASZ', None, None, None, None, None)
39 | (None, ptest79956, 'ptest79956@test.de', 'POWnqKIP', None, None, None, None, None)
40 | (None, ptest19568, 'ptest19568@test.de', 'uAKvVf0x3t', None, None, None, None, None)
41 | (None, ptest19569, 'ptest19569@test.de', 'uAKvVf0x3t', None, None, None, None, None)
42 | (None, ptest19568, 'ptest19568@test.de', 'uAKvVf0x3t', None, None, None, None, None)
43 | (None, ptest19569, 'ptest19569@test.de', 'uAKvVf0x3t', None, None, None, None, None)
44 | (None, ptest14429, 'ptest14429@test.de', 'OKPhYJPF', None, None, None, None, None)
45 | (None, ptest14429, 'ptest14429@test.de', 'OKPhYJPF', None, None, None, None, None)
46 | (None, ptest13346, 'ptest13346@test.de', 'nielwpxR0', None, None, None, None, None)
47 | (None, ptest13346, 'ptest13346@test.de', 'nielwpxR0', None, None, None, None, None)
48 | (None, ptest147453, 'ptest147453@test.de', 'nehrEdTc', None, None, None, None, None)
49 | (None, ptest147453, 'ptest147453@test.de', 'nehrEdTc', None, None, None, None, None)
50 | (None, ptest148627, 'ptest148627@test.de', 'bsruruAjy', None, None, None, None, None)
51 | (None, ptest148627, 'ptest148627@test.de', 'bsruruAjy', None, None, None, None, None)
52 | (None, ptest193366, 'ptest193366@test.de', 'V1Zrmw1j1', None, None, None, None, None)
53 | (None, ptest193366, 'ptest193366@test.de', 'V1Zrmw1j1', None, None, None, None, None)
54 | (None, ptest144791, 'ptest144791@test.de', 'QoCqkPzP', None, None, None, None, None)
55 | (None, ptest144791, 'ptest144791@test.de', 'QoCqkPzP', None, None, None, None, None)
56 | (None, ptest10606, 'ptest10606@test.de', 'SuK0X0Dg', None, None, None, None, None)
57 | (None, ptest122351, 'ptest122351@test.de', 'uGmDwQmH', None, None, None, None, None)
58 | (None, ptest122359, 'ptest122359@test.de', 'bfUf0PwA', None, None, None, None, None)
59 | (None, ptest122359, 'ptest122359@test.de', 'bfUf0PwA', None, None, None, None, None)
60 | (None, ptest126089, 'ptest126089@test.de', 'BDUImxzL', None, None, None, None, None)
61 | (None, ptest126089, 'ptest126089@test.de', 'BDUImxzL', None, None, None, None, None)
62 | (None, ptest165799, 'ptest165799@test.de', 'wodT00L0', None, None, None, None, None)
63 | (None, ptest165799, 'ptest165799@test.de', 'wodT00L0', None, None, None, None, None)
64 | (None, ptest123222, 'ptest123222@test.de', 'jD8XjI0W', None, None, None, None, None)
65 | (None, ptest123222, 'ptest123222@test.de', 'jD8XjI0W', None, None, None, None, None)
66 | (None, ptest194152, 'ptest194152@test.de', 't1uLMqPH', None, None, None, None, None)
67 | (None, ptest194152, 'ptest194152@test.de', 't1uLMqPH', None, None, None, None, None)
68 | (None, ptest158187, 'ptest158187@test.de', 'jcsdHsmp1', None, None, None, None, None)
69 | (None, ptest158187, 'ptest158187@test.de', 'jcsdHsmp1', None, None, None, None, None)
70 | (None, ptest13145, 'ptest13145@test.de', 'jj109WvArN', None, None, None, None, None)
71 | (None, ptest13145, 'ptest13145@test.de', 'jj109WvArN', None, None, None, None, None)
72 | (None, ptest15297, 'ptest15297@test.de', 'YopdHqHg', None, None, None, None, None)
73 | (None, ptest15297, 'ptest15297@test.de', 'YopdHqHg', None, None, None, None, None)
74 | (None, ptest166881, 'ptest166881@test.de', '0fr105u5K', None, None, None, None, None)
75 | (None, ptest166881, 'ptest166881@test.de', '0fr105u5K', None, None, None, None, None)
76 | (None, ptest44246, 'ptest44246@test.de', '4pvzeouAJC', None, None, None, None, None)

Data of Instructor Table:
77 | (None, itest146654, 'itest146654@test.de', 'kuCC0qz00T', None)
78 | (None, itest146654, 'itest146654@test.de', 'kuCC0qz00T', None)
79 | (None, itest19569, 'itest19569@test.de', 'jWVCIjTv0', None)
80 | (None, itest19569, 'itest19569@test.de', 'jWVCIjTv0', None)
81 | (None, itest19567, 'itest19567@test.de', 'SLu1tWp0', None)
82 | (None, itest19567, 'itest19567@test.de', 'SLu1tWp0', None)
83 | (None, itest128131, 'itest128131@test.de', 'g0QloqsoC9t', None)
84 | (None, itest128131, 'itest128131@test.de', 'g0QloqsoC9t', None)
85 | (None, itest113806, 'itest113806@test.de', 'ptP5p0MFn', None)
86 | (None, itest113806, 'itest113806@test.de', 'ptP5p0MFn', None)
87 | (None, itest15329, 'itest15329@test.de', 'xw0lBqJFrv', None)
88 | (None, itest15329, 'itest15329@test.de', 'xw0lBqJFrv', None)
89 | (None, itest177253, 'itest177253@test.de', 'jDeekGZFOx', None)
90 | (None, itest177253, 'itest177253@test.de', 'jDeekGZFOx', None)
91 | (None, itest199645, 'itest199645@test.de', '2BFh0WzHn', None)
92 | (None, itest199645, 'itest199645@test.de', '2BFh0WzHn', None)
93 | (None, itest15237, 'itest15237@test.de', 'DfRa5iyYde', None)

```

Figure 10: Viewing database datas