Overview

Solution files are available in the \ProjectAssets folder. To get the full-stack solution running, follow the steps below:

1. Setup and Check the Database
2. Setup and Check the REST API
3. Setup and Check the Front-End
4. Check out the Solution Application

The above steps shown above and detailed below assume the use of the previously described virtual machine which has software such as, Java, NodeJS, SQL database etc. already installed.

Note that the solutions are partial. They provide just enough structure and code to help the class understand how they might go about completing the project on their own. Implementations created during the capstone are encouraged to take advantage of ideas and skills brought forth by the class members and are not expected to conform to any specific solution. The true test will be whether and how well the created application succeeds at solving the initial problem statement.

01 Setup and Check the Database

The REST API gets data from a SQL database. The database software should already be installed on your machine. The tables and example data though still need to be created.

If you don't plan on changing the tables name or structures, then you can re-use the ones you create here with your own application implementation.

Follow the steps below:

1. Create a "dbuser" user
2. Run the schema and data creation scripts
3. Create a user-id that the REST API can connect with

Create a "dbuser" User

1. Open a terminal
2. Check that MySQL is up and running:

```
sudo systemctl status mysql
```

3. Navigate to the scripts directory under ProjectAssets

```
cd ~/ProjectAssets/sql-scripts
```

4. Check for the sql-schema.sql file:

```
ls
```

5. Check that MySQL is up and running:

```
sudo systemctl status mysql
```

6. Connect to database from command line client

```
sudo mysql
```

7. Check databases:

```
show databases;
```

8. Create a database named "talent":

```
create database talent;
```

9. Make "talent" the current database:

```
use talent;
```

10. Check tables (should be empty):

```
show tables;
```

Run the schema and data creation scripts

11. Run the sql_schema.sql script

```
source ./sql_schema.sql;
```

12. Check tables (should have 5 tables now):

```
show tables;
```

```
mysql> show tables;
+-------------------+
| Tables_in_talent3 |
+-------------------+
| application       |
| candidate         |
| job               |
| manager           |
| user              |
+-------------------+
5 rows in set (0.00 sec)
```

13. Check the structure of the 'user' table:

```
describe user;
```

```
mysql> describe user;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int         | NO   | PRI | NULL    | auto_increment |
| username | varchar(20) | YES  |     | NULL    |                |
| password | varchar(20) | YES  |     | NULL    |                |
| type     | varchar(20) | YES  |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

14. Run the talent_user.sql script:

```
source ./talent_user.sql;
```

15. Check the 'user' table (should have 4 rows now):

```
select * from user;
```

```
mysql> select * from user;
+----+----------+----------+----------------+
| id | username | password | type           |
+----+----------+----------+----------------+
|  1 | cindyloo | cindyloo | Candidate      |
|  2 | edsmith  | edsmith  | Hiring_Manager |
|  3 | admin    | admin    | Administrator  |
|  6 | jackson  | jackson  | Candidate      |
+----+----------+----------+----------------+
4 rows in set (0.00 sec)
```

Create a user-id that the REST API can connect with

16. Create a new ID with the username "dbuser" and the password "passpass"

```
create user 'dbuser'@'localhost' identified by 'passpass';
```

17. Grant permissions to the new user

```
grant all privileges on talent.* to  'dbuser'@'localhost';
```

18. Exit the mysql cli shell:

```
quit;
```

19. Connect with the new username and password:

```
mysql -u dbuser -p
```

20. When prompted enter the password "passpass" and hit enter. The "mysql" shell prompt appears

21. Try these commands (one at a time):

```
show databases;
use talent;
select * from user;
insert into user values(100,'abc','pass','test');
select * from user;
delete from user where id=100;
select * from user;
```

22. Close the mysql shell

```
quit;
```

The tables are now set up for use with the partial solution projects.

02 Setup and Check the REST API

The REST API gets data from a SQL database and makes it available over HTTP. In this step you'll setup and run the solution project for the REST API:

1. Copy the **~/ProjectAssets/back-end/solution-talent-api.zip** file to your ~/ProjectWork directory
2. Right click on the copied file and choose: "**Extract Here**". This will create a **solution-talent-api** directory under "**ProjectWork**"
3. Open a terminal and navigate to the "**solution-talent-api**" directory
4. Run the following to initialize the project:

```
./gradlew
```

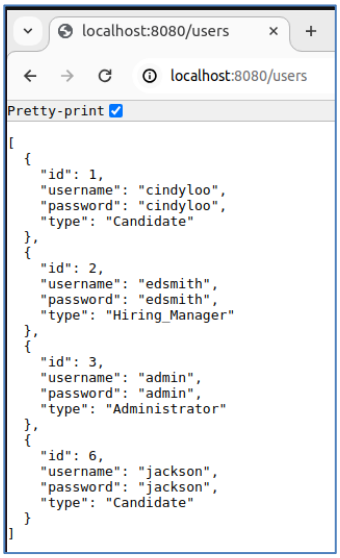
5. Execute the following to compile and run the project:

```
./gradlew bootRun
```

6. Open a browser to the following URL:

```
http://localhost:8080/users
```

7. You should see the following data:



8. If you see the data, then the REST API is working.
9. Leave the application running in the terminal.

03 Setup and Check the Front-End

The front-end of the solution is a React application. Follow the instructions below to get it running:

1. Copy the **solution-front-end.zip** file from **~/ProjectAssets/front-end** to your **~/ProjectWork** directory.
2. Right-click on the copied zip file and choose "**Extract here**"
3. Open a terminal and navigate to "**solution-talent-client**" directory that was just created.
4. Install project dependencies with the following command:

```
npm install
```

5. Wait for the command to complete.
6. Run the application using this command:

```
npm run dev
```

The application is now running.

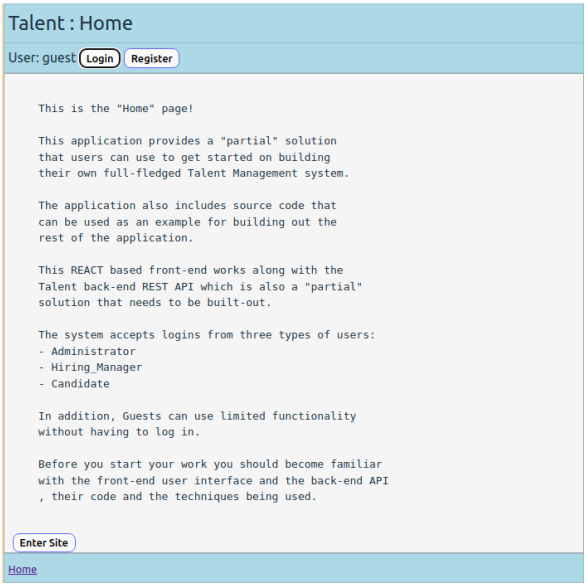
04 Check Out the Solution Application

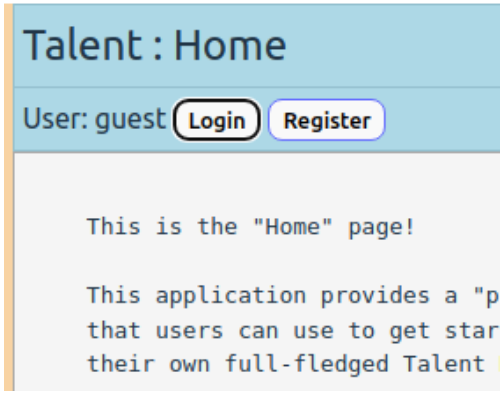
The REST API gets

1. Open a browser to the following URL:

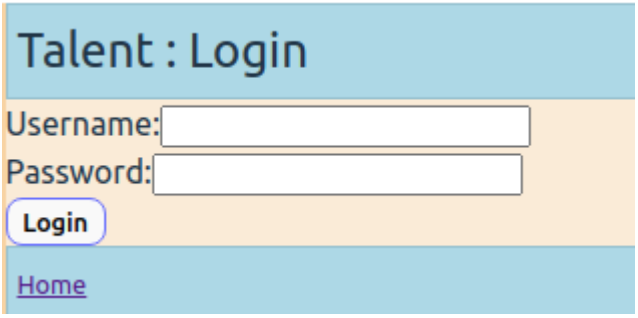
   ```
   http://localhost:5173/
   ```

2. The Home page should open, and you will be logged in as "guest":





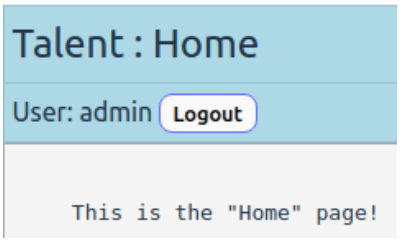
3. Click the "Login" button, the login page will open:



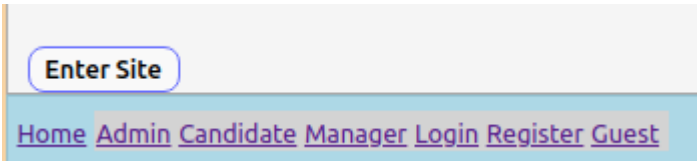
4. Login with the following credentials:

   ```
   username: admin
   password: admin
   ```

5. You will be logged in as "admin" and brought back to the home page:

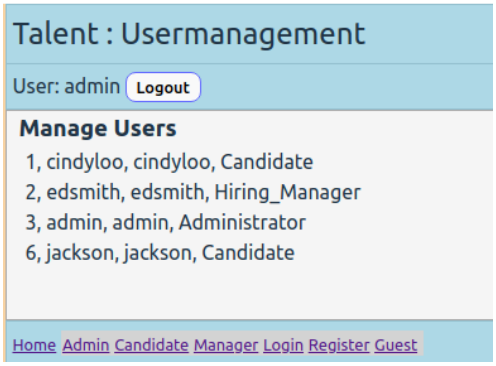


6. You should also notice a difference in the footer. It now shows links to various pages.



7. The "admin" user is of type "Administrator". Clicking on "Enter Site" will take you to the Administrator dashboard page:

8. From here, clicking on the "Manage Users" link will open the "User Management Page" which is only accessible by "Administrator" type users.



9. The User Management page displays the same users that you inserted into the "user" table during the SQL database setup earlier, and which were shown when you opened a browser to the "/users" endpoint.
10. Click the "candidate" link in the application's footer to go to the Candidate Dashboard page.
11. Like most pages in the solution application, the Candidate Dashboard page includes information on what still needs to be done to build out the page:

```
This "Candidate Dashboard" page needs to be developed!

This page should include:
- a list of jobs the logged-in user selected
- jobs in the list are ones the user is planning to or has applied for
- the job list should include application status
- user can click on an applied-for job and go to a detail page with more info
- a "Search" button that navigates to an "open jobs search page"
  - reject data when username already exists
  - reject passwords less than 8 characters
  - save valid registrations in the user table
```

12. The links in the lower part of the page show some of the functionality that is supposed to be available on the Dashboard. That functionality is not yet implemented. If you click on them now, they just pop up an alert.
13. Try logging in as the other user types: Manager and Candidate and see what's available with each login.

The code that runs the back-end server and front-end client is fully available. Feel free to open it up and take a look as you start working on your own implementation.