CHALLENGE - Online Banking
Same local bank hires you again to develop a prototype for their online banking system.

PHASE 1.0 - SETUP INSTALLATION /1
You'll notice that there are already files premade. In the first phase, your job is to set everything up so that you can start your online banking development.

You'll need express, express-session, body-parser, jquery, pg, socket.io, and webpack

PHASE 1.1 - SETUP WEBPACK /2
Complete your webpack, you need to build 2 files in the build folder.

1. main.js -> mainbundle.js
2. transfer.js -> transbundle.js

PHASE 1.2 - SETUP DATABASE /6
The database you'll be creating is called "online_banking". And you'll be creating 2 tables call "ob_accounts" and "ob_transfers".

In ob_accounts

1. id -> SERIAL PRIMARY KEY
2. password -> VARCHAR(4) UNIQUE
3. name -> VARCHAR(50)
4. secret -> VARCHAR(50)
5. question -> VARCHAR(500)

In ob_transfers

1. id -> SERIAL PRIMARY KEY
2. amount -> INTEGER DEFAULT 0
3. ob_accounts_id -> INTEGER REFERENCE ob_accounts (id)
4. transfter_type -> VARCHAR(50)

When you're done, type \d+ ob_accounts; and take a screenshot of the result.
Then type \d+ ob_transfers; and take a screenshot of the result.
Put both screenshots in the imgs folder.

PHASE 2 - SETUP SERVER /3
WITHOUT modifying the existing codes. Complete index.js.

The server should use port 12345 or whatever the online server gives
"/" leads to main.html
"/transfer" leads to transfer.html

PHASE 3 - MAIN.HTML AND MAIN.JS /7
Starting with main.html you'll need to use javascript to build the registration/login system.

1. Insert a new account by typing in your information and registering /1
2. Check whether BOTH the password and secret is between 5-10 characters long /1
3. Successful login leads to /transfer /1
4. Successful login store sessions properly /1
5. When login fails, but the name and password matches, show the question hint in the div /1
6. Hide and show the right elements /1
7. If a user tries to go to /transfer, but is not logged in, then redirect the user back to this main page /1

PHASE 4 - TRANSFER.HTML AND TRANSFER.JS /3

You'll have to make a system where you make payments/transfer money. Depending on which button you clicked on, you'll insert a different transfer_type to the table.

1. Clicking on MAKE PAYMENT or TRANSFER TO ACCOUNT makes the numbox visible /1
2. When OK is clicked on, insert into the database the amount, the account this amount is associated with, and the transfer_type "payment" or "account" /1
3. Logout destroys your sessions and redirects you back to the main page /1

PHASE 5 - STATISTICS.HTML /1

I didn't make a JS file for this, so you'll have to set everything up yourself. (Both js and the server) This page is simple. You just show all the transactions the account has done thus far.

1. Display all your transactions + the sum.
    a. The format is {amount} => {transfer_type}
2. Go back to /transfer when the back button is clicked on

PHASE 6 - OTHER TRANSACTIONS /1

In the statistics page, other users can connect to the page via socket. When a connection is made (including yourself), create a div for the connected user's name in the otherUsers div. If you click on a user, display all the transactions and sum of the clicked user instead.

BONUS

Make the transactions displayed sortable by transfer_type OR amount ascending OR amount descending.

Submit the following in a zip file

- build folder
- cli folder
- js folder
- img folder
- index.js
- webpack.config.js