Digital Hub - Frontend Assignment

Assignment Purpose

The purpose of this test is to show off your level of Front-end development skills and to show your knowledge of modern front-end frameworks and practices.

Brief Description

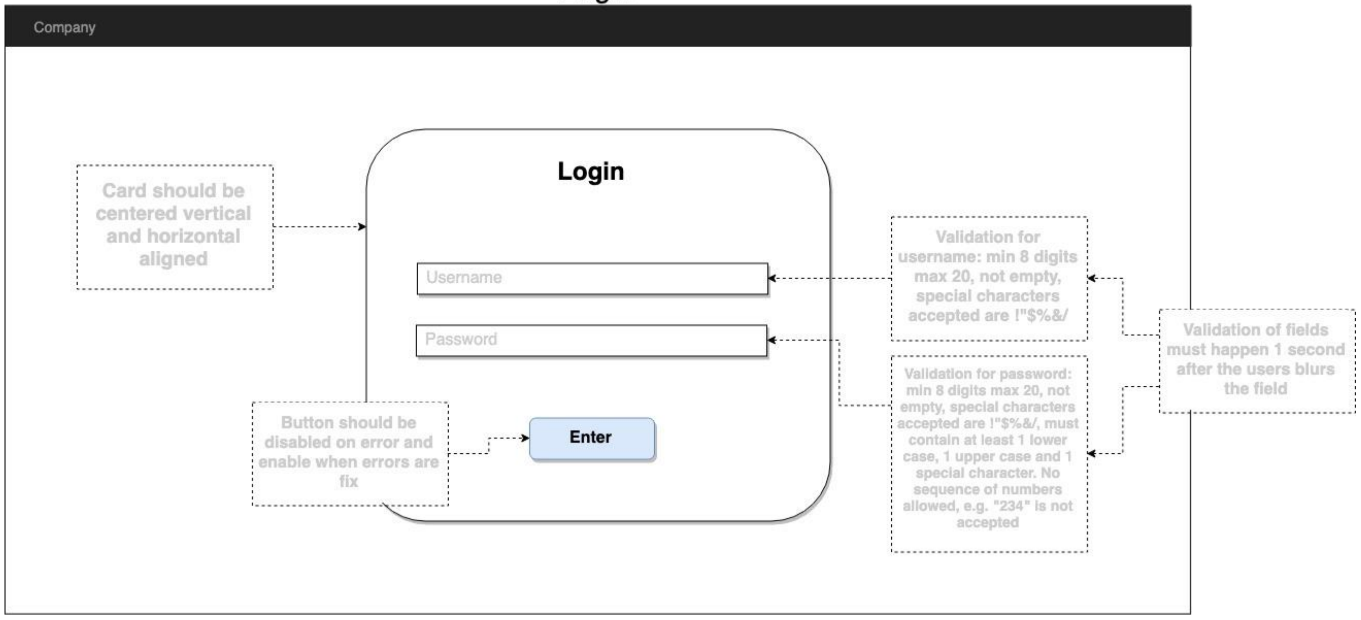
For this assignment you need to develop three views based on the given mock ups trying to follow them as close as possible and implement mock calls to an API (these can be plain JSON files faking the responses). Functionality for this assignment is focus only on the UI; meaning that databases, endpoint functionality and sessions are out of scope.

Functional Requirements

Login

As a user I want to land as first page in a login view, in this page as a user I'm able to introduce my credentials (username and password) and get access to the main page.

Login



Card should be centered vertical and horizontal aligned

Validation for username: min 8 digits max 20, not empty, special characters accepted are !"$%&/

Validation for password: min 8 digits max 20, not empty, special characters accepted are !"$%&/, must contain at least 1 lower case, 1 upper case and 1 special character. No sequence of numbers allowed, e.g. "234" is not accepted

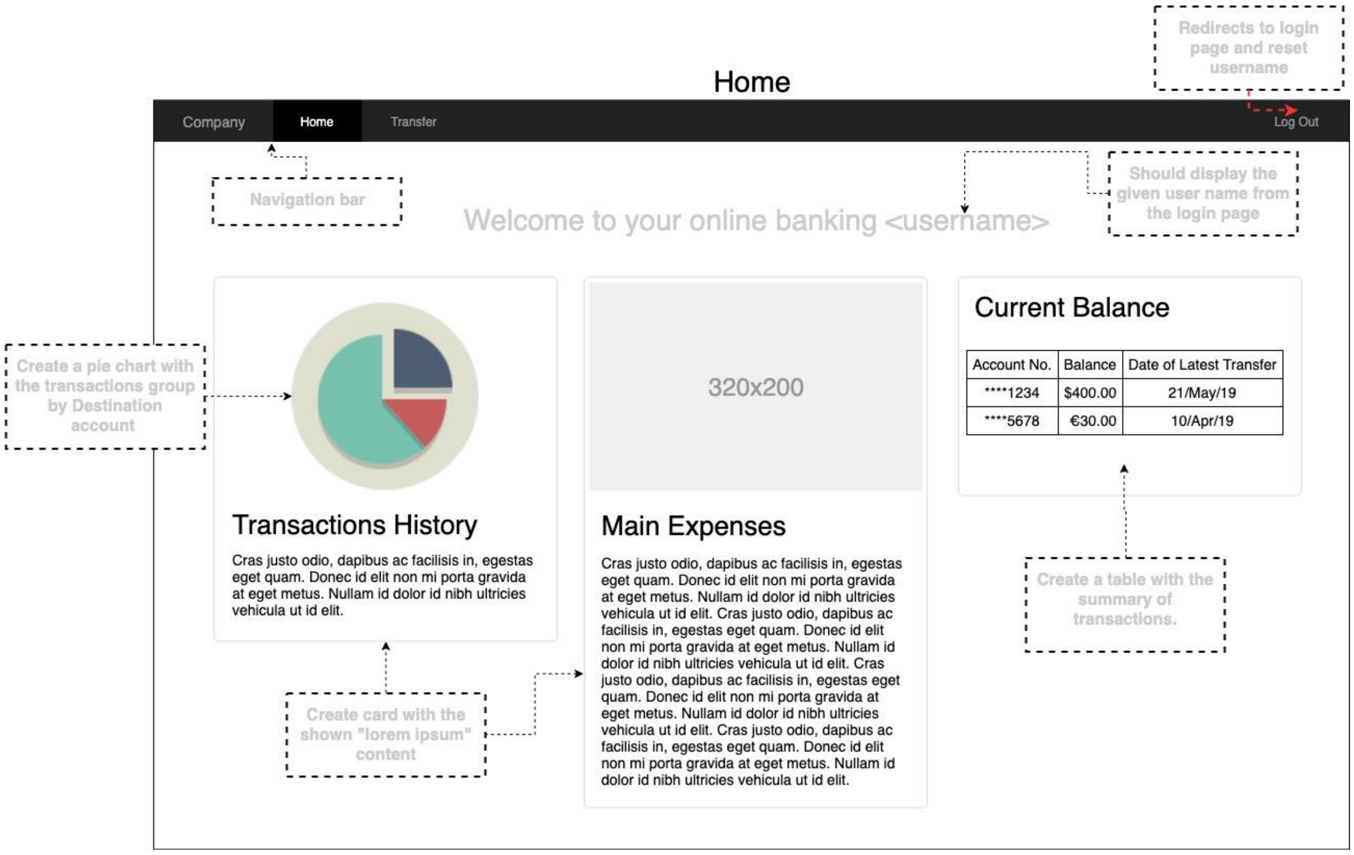
Validation of fields must happen 1 second after the users blurs the field

Button should be disabled on error and enable when errors are fix

Home Page

This is the main page of the application and it must show the described sections in the mock and three cards with the following information:

- Transaction History: only the pie chart functionality must be implemented (instructions on the mock)(consider that destination accounts will be the same currency; show the sum of amounts transferred) title and content text are static lorem ipsum.
- Main Expenses: Static lorem ipsum.
- Current Balance: implement a table with the summary of the user's accounts


Home
Company
Home
Transfer
Log Out
Redirects to login page and reset username
Navigation bar
Should display the given user name from the login page
Create a pie chart with the transactions group by Destination account
320x200
Create card with the shown "lorem ipsum" content
Create a table with the summary of transactions.

Welcome to your online banking <username>

Transactions History

Cras justo odio, dapibus ac facilisis in, egestas eget quam. Donec id elit non mi porta gravida at eget metus. Nullam id dolor id nibh ultricies vehicula ut id elit.

Main Expenses

Cras justo odio, dapibus ac facilisis in, egestas eget quam. Donec id elit non mi porta gravida at eget metus. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Donec id elit non mi porta gravida at eget metus. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Donec id elit non mi porta gravida at eget metus. Nullam id dolor id nibh ultricies vehicula ut id elit. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Donec id elit non mi porta gravida at eget metus. Nullam id dolor id nibh ultricies vehicula ut id elit.

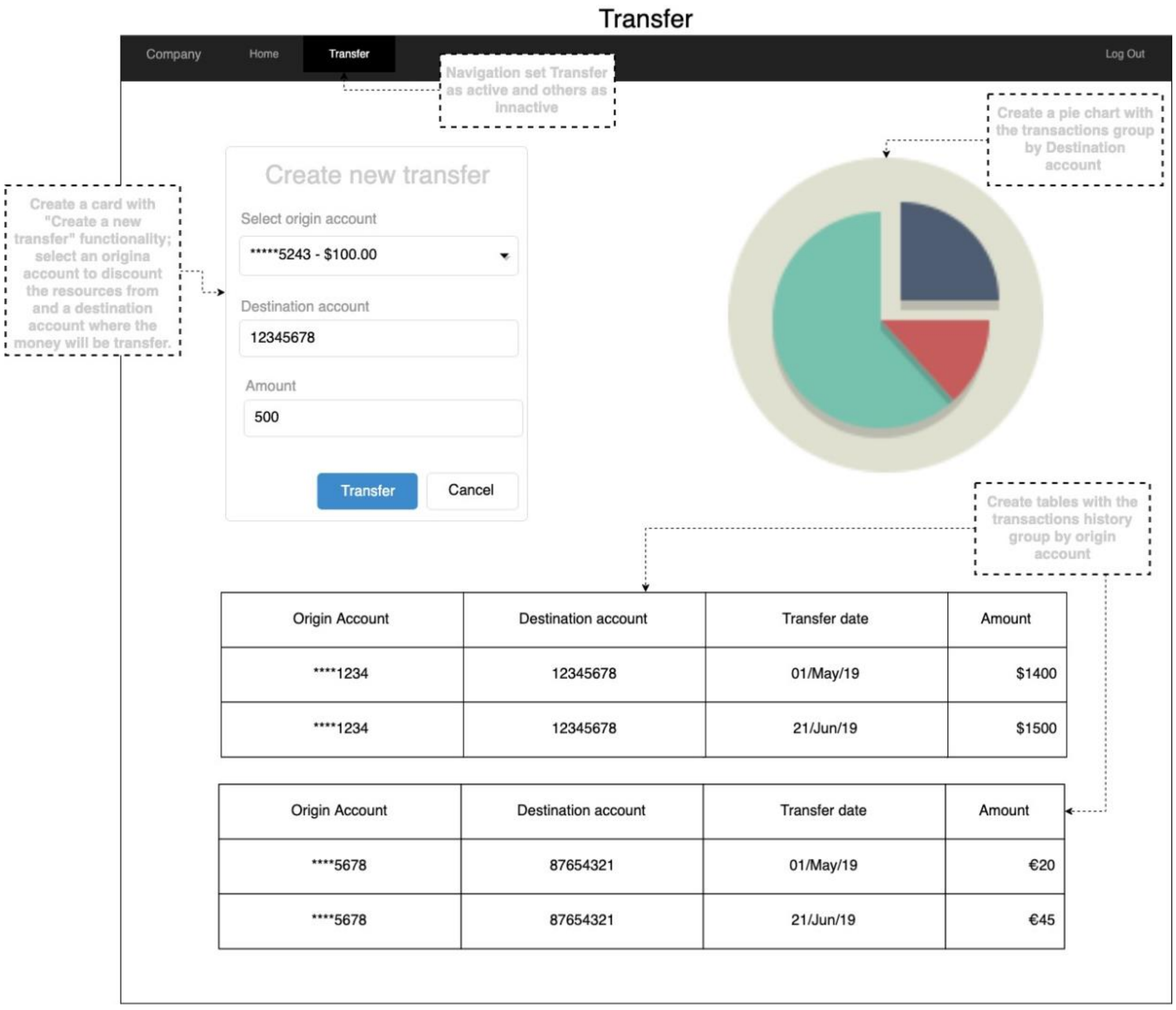
Current Balance

Account No.	Balance	Date of Latest Transfer
****1234	$400.00	21/May/19
****5678	€30.00	10/Apr/19

Transfers

In this section you must create a basic functionality to simulate a money transfer between accounts.

1. User selects origin account
2. User types destination account (8 chars length)
3. User types the amount to be transfer
4. User clicks “Cancel” must clean the form
5. User clicks “Transfer” must do the following validations
    a. Origin account has enough money to perform the transaction
    b. Amount to be transfer is less than 100000
6. In case the transfer can be perform; update the correspondent account table and pie chart with the new transaction. Note that transactions are not mandatory to be store or tracked.



Navigation set Transfer as active and others as innactive

Create a pie chart with the transactions group by Destination account

Create a card with "Create a new transfer" functionality; select an origina account to discount the resources from and a destination account where the money will be transfer.

Create new transfer

Select origin account: *****5243 - $100.00

Destination account: 12345678

Amount: 500

Create tables with the transactions history group by origin account

Origin Account	Destination account	Transfer date	Amount
****1234	12345678	01/May/19	$1400
****1234	12345678	21/Jun/19	$1500

Origin Account	Destination account	Transfer date	Amount
****5678	87654321	01/May/19	€20
****5678	87654321	21/Jun/19	€45

Mock Endpoints

For this application you must consume the fake endpoints with the given responses. Feel free to add as many examples as you consider but keep the format of the responses for all endpoints.

Transfer Money

As a user should be able to transfer money using the Transfer POST method with the payload as follows:

```
{
"fromAccount": 987654321,
"toAccount": 123456789,
"amount": 99.54
}
```

Transaction History

The expected response object is as follows:

```
{
        "transactions": [
                {
                        "fromAccount":123456789,
                        "toAccount":192837465,
                        "amount": {
                                "currency": "€",
                                "value": 876.88
                        },
                        "sentAt": "2012-04-23T18:25:43.511Z"
                },
                {
                        "fromAccount":123456789,
                        "toAccount":192837465,
                        "amount": {
                                "currency": "€",
                                "value": 654.88
                        },
                        "sentAt": "2012-04-21T18:25:43.511Z"
                },
                {
                        "fromAccount":987654321,
                        "toAccount":543216789,
                        "amount": {
                                "currency": "$",
                                "value": 543
                        },
                        "sentAt": "2012-04-23T18:25:43.511Z"
                },
                {
                        "fromAccount":987654321,
                        "toAccount":543216789,
                        "amount": {
                                "currency": "$",
                                "value": 987.54
                        },
                        "sentAt": "2012-04-23T18:25:43.511Z"
                }

        ]
}
```

Account Balance

The expected response object is as follows:

```
{
        "balance": [
                {
                        "account":123456789,
                        "balance": {
                                "currency": "€",
                                "value": 765095.54
                        },
                        "owner": 7612333392,
                        "createdAt": "2012-04-23T18:25:43.511Z"
                },
                {
                        "account":987654321,
                        "balance": {
                                "currency": "$",
                                "value": 524323.54
                        },
                        "owner": 7612333392,
                        "createdAt": "2012-04-23T18:25:43.511Z"
                }

        ]
}
```

Technology Restrictions

Design your application any way you want but please focus on writing, clean and reusable code following front-end development best practices and components.

React is the allowed technology for this assignment, if you include any framework please elaborate the reasons.
Create a Readme file explaining the approach you followed and how to run the application.
There is no need to persist all transactions in any specific way.
Scaffolding or code generators are forbidden.

Good Luck!