Credit Card Management System

Credit Card Management System

SWAFE 2023 – Hand in 1

Introduction

We have been contacted by a new client who runs a consultant business. Each of their consultants has a company credit card, and they want to keep track of expenses for each credit card. They have drafted with a requirement specification for a credit card management system.

The solution will provide the accounting department with an overview of use for each credit card. Accountants can add new credit cards and remove them when after they have expired or is lost. It will also be possible to check transaction details for each credit card, as well as searching in transactions across all registered cards.

Requirement specification

Functional requirements

`F0` Application skeleton

- `F0.1` Skeleton shall contain a navigation bar
- `F0.1.1` Navigation bar shall contain links for navigation to Add credit card screen
- `F0.1.2` Navigation bar shall contain a link for navigation to Transactions screen
- `F0.1.3` Navigation bar shall contain a link for navigation to Home screen

`F1` Home screen

- `F1.1`: Screen shall contain a list of credit cards

`F2` Credit card list

- `F2.1`: List shall contain an element for each credit card
- `F2.1.1`: List item shall contain properties `card_number`, `cardholder_name`, `issuer`
- `F2.1.2`: Navigate to a credit card details screen when an entry is clicked/pressed

`F3` Credit card details screen

- `F3.1` Screen shall contain elements displaying the following credit card properties: `card_number`, `cardholder_name`, `csc_code`, `expiration_date_month`, `expiration_date_year`, `issuer`

- `F3.2` Screen shall contain the option to remove the credit card
- `F3.3` Screen shall contain a list of transactions for the credit card

`F4` Add credit card screen

- `F4.1` Form that contains fields for `card_number`, `cardholder_name`, `csc_code`, `expiration_date_month`, `expiration_date_year`, `issuer`
- `F4.1.1` Field `card_number` only accepts numbers (integers)
- `F4.1.2` Field `card_number` length must be `7-16` digits
- `F4.1.3` Field `card_number` is required
- `F4.2.1` Field `csc_code` only accepts numbers (integers)
- `F4.2.2` Field `csc_code` length must be `3` digits (integers)
- `F4.2.4` Field `csc_code` is required
- `F4.3.1` Field `cardholder_name` is required
- `F4.4.1` Field `expiration_date_month` must be in range `1-12`
- `F4.4.2` Field `expiration_date_month` is required
- `F4.5.1` Field `expiration_date_year` is required

`F5` Transactions screen

- `F5.1` Screen shall show a list of all transactions registered in the system
- `F5.1.2` Screen shall present the option to add a transaction to the transaction list
- `F5.1.3` Screen shall present the option to filter transactions
- `F5.1.4` Screen shall provide filtering based on `card_number`

`F6` Transactions list

- `F6.1.1` Each transaction shall display properties `credit_card`, `amount`, `currency`, `comment`, `date`
- `F6.1.2` Field `credit_card` shall be selected from a list of credit cards
- `F6.1.3` Field `amount` must be a `number`
- `F6.1.4` Field `amount` is required
- `F6.1.5` Field `currency` is required
- `F6.1.6` Field `date` is required
- `F6.1.7` Each transaction shall present the option to remove itself

Design requirements

Futhermore, the solution must include the following:

- [ ] The solution shall be implemented using the lastest major release of the Angular development platform(GitHub)
- [ ] At least one module must be lazy-loaded(docs)
- [ ] The application must implement at least one custom pipe (docs) *Hint: Obvious candidates could be expiration date*
- [ ] At least one module must contain a routing module(docs)
- [ ] At least one component must be standalone(docs)
- [ ] The application must be seeded with data from the server found @ `hand-in/credit-card-server`

Credit card server documentation

Installation

1. Run `npm install` in `hand-in/credit-card-server`
2. Run `npm start` in `hand-in/credit-card-server`

The server is running @ http://localhost:3000

Available endpoint

- `GET /cards`—returns an array of credit cards
- `GET /cards/:card_number`—returns credit card with `card_number`
- `POST /cards`—creates a credit card
- `DELETE /cards/:card_number`–deletes a credit card
- `GET /transactions`—returns an array of transactions
- `POST /transactions`—creates a transaction
- `DELETE /transactions/:transaction_uid`–deletes a transaction

Internal notes

Our Senior Vice Principal Software Engineering Architect has chosen Angular to be used as the frontend framework. They have defined a proposal for an initial architecture. *Note: that some details is left out for the developer teams to decide. The list is not complete*

Angular Artifact Checklist

- Modules
  - [ ] `AppModule`

    - [ ] `HomeComponent`
    - [ ] `NavigationBarComponent`
  - [ ] `CreditCardModule`
    - [ ] `CreditCardListComponent`
  - [ ] `TransactionModule`
    - [ ] `TransactionOverviewComponent`
    - [ ] `TransactionAddComponent`
- Standalone components
  - [ ] `TransactionListComponent`
  - [ ] `CreditCardAddComponent`
- Services
  - [ ] `CreditCardService`
  - [ ] `TransactionService`

Formalia

- Group size: 1-4 people
- Deadline: 16th October 2023 (2022-10-16)

Submission

Before submitting your solution, do the following:

1. Delete the `node_modules` folder in the workspace root folder
2. Add a file `participants.txt` and insert a new line for each participant with the student number and name of each member separated by whitespace
3. Add `participants.txt` to the root folder of your application
4. Archive and compress you application using one the following formats: `zip`. All other formats (`rar`, `7z`, etc.) will result in a request for resubmission
5. The filename should be named `Group<no>.zip` *Example:* `Group01.zip`
6. And you are ready to upload it to Brightspace

Example `participants.txt` contents:

```
202101234 Alice Alison
202109876 Bob Bobson
```