# Credit Card Processing

Write a small full stack application for a credit card provider. It should allow you to add new credit card accounts and view them as a list. The backend must be a RESTful API.

**NodeJS Engineer:** Please only work on the back end part of the assignment. You can ignore the UI component, instead please provide a swagger UI or postman collection to call the APIs if you are having issues with the front-end part.

## Requirements

Two REST Endpoints must be implemented

* "**Add**" will create a new credit card for a given name, card number, and limit
  + Card numbers should be validated using Luhn 10
  + New cards start with a £0 balance
  + for cards not compatible with Luhn 10, return an error
* "**Get all**" returns all cards in the system

The endpoints should be given appropriate URLs and HTTP methods, according to RESTful design principles. There is no right and wrong answer here, but you may be asked to explain and justify your design, so give it some thought.

## Validation

* All input and output will be JSON
* Credit card numbers may vary in length, up to 19 characters
* Credit card numbers will always be numeric

## User Interface

* Build a UI according to the provided wireframe
* The UI should be in React
* You can design it how you wish, and feel free to use a design system
* The UI should utilise the RESTful services you have built
* Show error messages for invalid form data

## Technical requirements

* Create the RESTful API using Spring Boot and Use Maven/Gradle for dependency management
* Create the endpoints use appropriate HTTP Methods and define the payloads, return codes and response structures
* Write unit test cases, though please note that we’re looking for quality over quantity – coverage does not need to be high
* Use an in-memory DB to store the information while the API is running, so that it can store the credit card information
* Hand code the Luhn 10 check, don’t use a library

## Evaluation Criteria

We will assess your work on the following basis:

* Clear documentation
* Code structure, quality and consistency
* Technology choices
* Performance
* Cross browser compatibility
* Responsiveness
* Accessibility
* Attention to detail
* Dependency management
* Test quality and coverage
* Git commit history