**Credit Card Provider Application**

**Problem Statement:**

Develop 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. Requirements Two REST Endpoints must be implemented

• "Add" will create a new credit card for a given name, card number, and limit o Card numbers should be validated using Luhn 10 o new cards start with a £0 balance o 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

**Tech Stack:**

|  |  |  |
| --- | --- | --- |
| Sr. | Tech Stack | Version |
| 1 | Java | 1.8 |
| 2 | Maven | 3.8.4 |
| 3 | Git | 2.35 |
| 4 | IDE: IntelliJ | 2021.3.2 |
| 5 | Node | 16.14.0 |
| 6 | NPM | 8.3.1 |
| 7 | Spring Boot | 2.6.3 |
| 8 | ReactJs | 17.0.2 |
| 9 | JPA | 2.6.1 |
| 10 | MySQL | 8.0.27 |

**Solution:**

1. **Create a new Spring Boot application:**

* Go to <https://start.spring.io> and fill the project details. We are using project name as ‘credit-card-provider’. Add ‘Spring-Web’ as dependency and click on generate.
* It will create zip - credit-card-provider.zip
* Extract it into git checkout project - D:\publicis\_sapient\workspace\credit-card-provider
* Commit and push code to bitbucket

Graphical user interface, text, application

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

1. **TDD (Test driven Development) approach:**
2. **Start with Writing Tests**,

* Create a package com.bank.cib.cc.web.controller’ in the ‘src/test/java’ directory.
* Create a Test class ‘UserRestControllerTest’ in the com.bank.cib.cc.web.controller package of ‘src/test/java’ directory.
* We will now add below annotations on the class:

@WebMvcTest(controllers = UserRestController.class)

@ExtendWith(SpringExtension.class)

* We are using ‘@ExtendWith’ annotation to register it with JUnit 5 test, and ‘@WebMvcTest’ is going to scan the controller we have defined and the MVC infrastructure.
* Now once we add these codes in test file, we will see compilation error since ‘UserRestController.class’ does not exist yet. It will look something like this:

A screenshot of a computer

Description automatically generated with medium confidence

1. **Start writing Implementation Class**

TDD rules state that we should stop writing tests as soon as we get any error and start writing production code to fix that error, and we should not write any more production code than needed to fix the error.

We got compilation error for file name ‘UserRestController.class’, so now we need to write code and create this file. We can simply do this by hovering mouse over the error and using IDE features to create this file should be in ‘src/main/java’ directory. Once we make these changes it should look something like this:

A screenshot of a computer

Description automatically generated

1. After error is resolved, and now we should move to test file and start writing tests.

* We will autowire ‘MockMvc’ in our test class and use this to make requests to our controller.

@Autowired

MockMvc mockMvc;

* We will write our first test method ‘getAllUser ()’ and annotate it with ‘@Test’. We will write an assert statement where make a ‘GET’ request to our controller for API path ‘/user/all’ and check the status for 200.
* We will execute our test, and it will fail as it should since we have not written any logic in our controller class. It should look something like this:

A screenshot of a computer

Description automatically generated with medium confidence

Test failed with 404 – mapping not found.

A screenshot of a computer

Description automatically generated with medium confidence

We start with creating ‘getAllUser()’ method and adding annotations ‘@RestController’, ‘@RequestMapping(“/user”)’, and ‘@GetMapping(“/all”)’. We will add return statement.

Now when we execute the test, it should pass and we get green signal.

It should look similar to this:

Text

Description automatically generated with medium confidence

We will now repeat this process for further api, service, dao layer implementations.

**Task Implementation**

1. addCreditCard

2. getAllCreditCards

first to write down test case and then rest controller. when we execute the test, it should pass and we get green signal. It should look like this:

Graphical user interface

Description automatically generated

Build and Deploy application:

Click on right hand side maven tab

It will open maven build project

Click on Execute Maven Goal button and run **mvn clean install**

It will compile all classes and creates **credit-card-provider-0.0.1-SNAPSHOT.jar** deployable jar file.

Graphical user interface

Description automatically generated

Now start the Credit Card Provider application from IntelliJ from Run Configuration

It will start to default 8080 tomcat port

Graphical user interface

Description automatically generated

I have refactored some of UserRestController url to keep same naming convention.

Open Browser: and hit below url to get response.

**Add Cards:**

<http://localhost:8080/cred/cards/add>

<http://localhost:8080/cred/cards/get/all>

**Add Users Detail:**

<http://localhost:8080/cred/users/get/all>

<http://localhost:8080/cred/users/add>

Now all Test cases and implementation is ready for controller, service, repository layer.

**We can test add and get api with postman.**

<http://localhost:8080/cred/users/add>

Graphical user interface, text, application, email

Description automatically generated

{

    "firstName":"Gopal",

    "middleName":"V",

    "lastName":"Patil",

    "mobileNumber":"1234567890",

    "dob": "14-11-1983",

    "creditCards": [

        {

            "cardNumber": "4556-0690-9685-2293",

            "cardName":"Gopal",

            "balance": 0,

            "cardLimit": 2000

        }

    ]

    }

<http://localhost:8080/cred/cards/add>

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

   {

        "cardNumber": "4556-0690-9685-2293",

        "cardName":"Gopal",

        "balance": 0,

        "cardLimit": 2000

    }

**Database:**