# Full-stack developer Test Paper

**Instructions:**

- You are required to complete the 3 parts of the exercise:
    - Part 1 (Front-end);
    - Part 2 (Back-end); and
    - Part 3 (Spring Security Limit Login Attempts).

- Create a repository in your GitHub account, commit the code into that repository for each part, and send us the link to the repository.

# Part 1: Front-end

## 1.1 General

1. Create a login component to allow users to login
   a. Once login, there must be a main page showing “New Transaction” and “View Submitted Transactions”.
   b. There must be a logout button to allow the users to logout.
   c. There must be an Admin user and a Normal User

2. Create responsive **Material Design Navigation Bar** consisting of Logo, Customer Form, Table, User Profile and Log Out

3. Create a component to perform a new bank transaction
   a. Fields:
      Radio Button: [New, Existing],
      Reference, Customer number, Customer name, Customer address, Customer Phone number, transfer amount, Transfer currency, Beneficiary Bank, Beneficiary Account Number, Payment details, Credit/Debit Card Details,
      Region: [Drop Down Values: Port Louis, Curepipe, Vacoas, Port Mathurin]
      All fields are required.
   b. Once the user input the customer number, and then the field’s customer name, address and phone number must be populated using the customer JSON response file provided.

4. Create a component to view the list of submitted transactions
   a. The labels should consist of Customer name, Transfer amount, Transfer Currency and Reference
   b. Table should be in Material Design with Sorting and Pagination
   c. You should populate data before submitting for submitting purpose. You may use a runnable Node Express Server or a JSON File.
   d. New Submitted data should appear in table above.

5. Use generic model(s) to store the field values (separate customer information *[customer number, customer name, customer address, customer phone number]* with the others)

*6.* Add validations on customer phone number and transfer amount to check if they have numbers only. *[use above Angular concepts to implement generic functions]*

7. Add validations on beneficiary bank and payment details to check if they have characters only. *[use above Angular concepts to implement generic functions]*

8. Currency should be a drop down list with the following values: AED, EUR, CHF, MUR and USD

9. The reference must have a prefix "CUS" and followed by an auto-generated number (consisting of YYYYMMDD and a sequence number. Max length of reference should be 15 characters)

10. Use any online rest API to get and post data into the Angular app. You can use end-to-end testing script to test this rest API. You will need to mention which rest API you have used when submitting your assignment.

11. Create end to end testing scripts

## 1.2: Multi Value Form

*Multi Value form with the following fields: Reference, Customer number, Customer name, Customer address, Customer Phone number, transfer amount, Transfer currency, Beneficiary Bank, Beneficiary Account Number, Payment details, Credit/Debit Card Details, Region*

- Form should be incremented by Add Button
- Form should be removed by Remove Button
- Multi Form to be reset by Clear Button
- Consume a web service that will auto-populate all fields when called by Customer Number, with appropriate error response in UI. The Call should only populate form fields in which the customer number is queried.
- If region **Port Mathurin** is selected, Customer Address is not required (can be hidden). Required if region is not **Port Mathurin**
- If New selected, only Customer Name to be made required and form is valid. Customer number should be disabled.
- If Existing selected, User must fill all required fields.

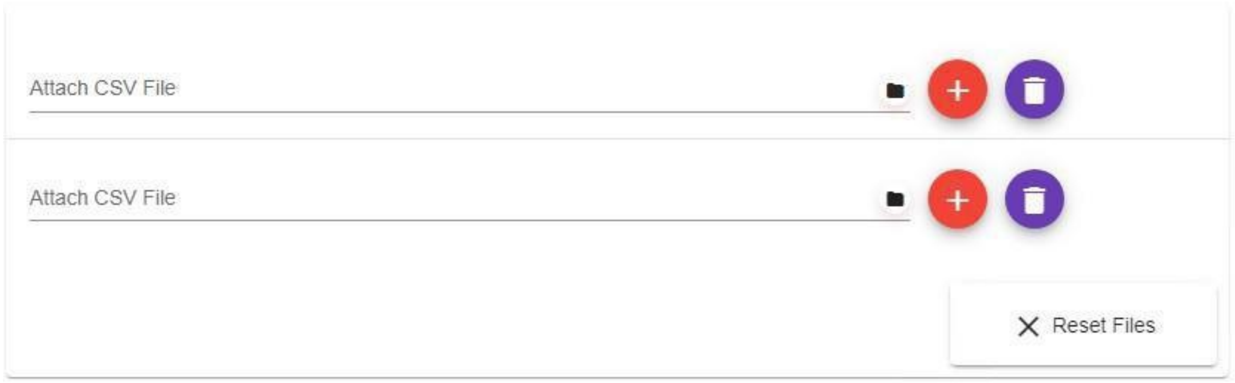


*Figure 1: Example of Multivalue form*

You may use any CSS like bootstrap to make the pages look nice.

## Assessment Criteria

- Application of Angular concepts (lifecycle hooks, modules, directives, components, pipes, event binding, property binding, routing, generic API call)
- Code quality of the Angular Code.
- Code quality of the HTML and CSS if any.
- Look and feel of the form and list pages.

## Deliverable for Front-end

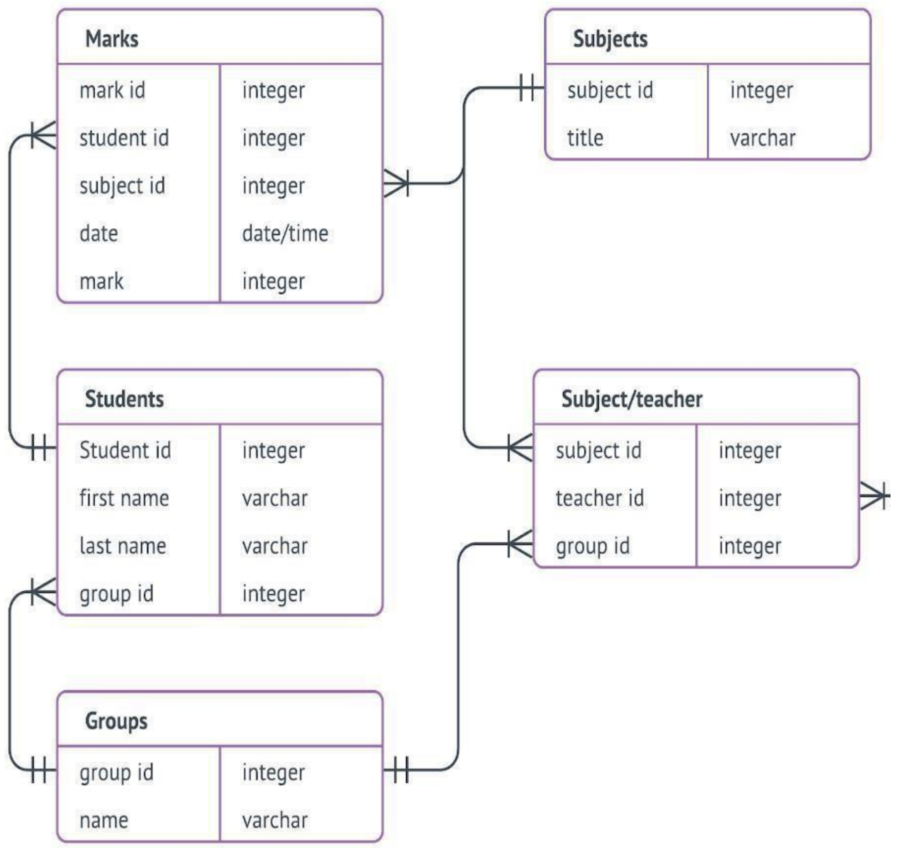
- Project should be developed in Angular 9.
- Project should be runnable out of the box by ***ng serve/npm start***.
- Create a repository in your GitHub account and commit the code into that repository. Send us the link to the repository

# Part 2: Back-end

## 2.1 : Create CRUD REST API/Service with Spring Boot, JPA and Hibernate

1. Create a Spring Boot project (Use maven build)

2. Create the tables below in h2 (When running your project, it must create the tables automatically in h2)



3. Create rest API (Add, Update, Delete, Find All and Find by Id)

4. Create rest API:
    a. to get the mark for a particular student id
    b. to get the number of students for a particular teacher id
    c. to get the list of marks in each subject for a particular student id

5. Use maven build

6. Implement JWT to secure the rest API

7. Add Swagger in your project
8. Implement JUnit for part 4

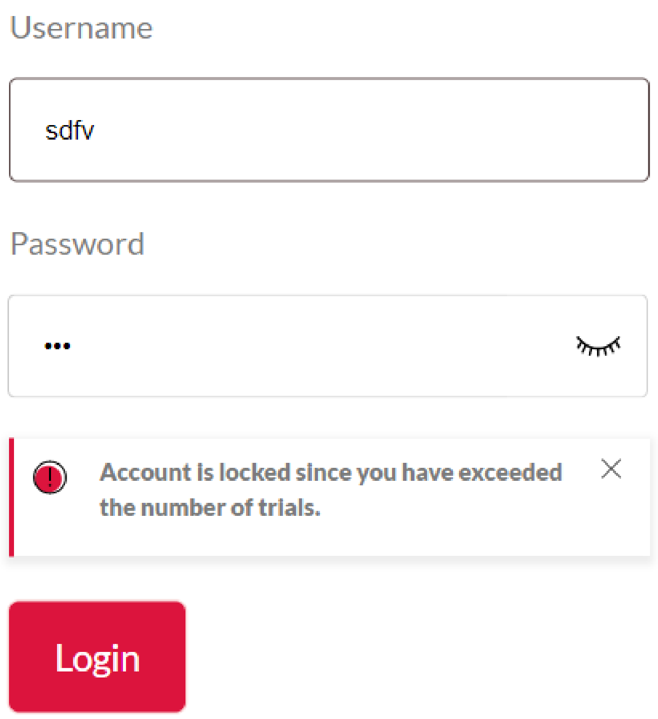
## Assessment Criteria

- Use of Spring Boot concepts such as controller, service, DAO, entity, etc.
- Code quality
- Rest API must work via Swagger
- It must be a runnable project

## Deliverable

- Create a repository in your GitHub account and commit the code into that repository. Send us the link to the repository.

## Part 3: Spring Security Limit Login Attempts



Account is locked since you have exceeded the number of trials.

For the Authentication of Part 1, you are requested to use Spring Security and any database (Oracle/MySQL/H2) to store user information to limit attempt to login of three times, such that:

- the account shall be locked on the last failed attempt with an error message shown on screen. These details shall be persisted to keep a log of same; and
- the user account shall be locked during 24 hours. To be able to unlock it will require a CRON job (Spring Boot) to be run in the background for updating of this flag from db.

You are can use standard Spring Data JPA and Hibernate in the data access layer and unit test should be used to be able to test different scenarios.