**Testing Strategy**

**Challenge 1:-**

Create an application that allows the uploading of a file. Once the file is loaded, the application must validate and parse each line of file information (plaintext) and then upload the information to the database. The file structure is as follows:

x, y, z

* x is the username id of a user. Its length is 10 characters. They are just numbers, and can have zeros on the left.
* y the amount of coins associated with the user x. must accept only natural numbers.
* z is the user name x.

The three fields are mandatory and can not be blank. If you find a wrong value when making the validation of each line, stop the process and if the database is modified, returning it to its initial state. Warn the user which line had an error.

### The purpose of the test strategy of Challenge 1 is as follows :-

### The Challenge is validated to accept a .txt/.csv file with containing 3 values/data in the file, which are comma separated.

### The first value is the username id and length of the username id should be not be more than 10 characters and can have zeros on the left.(Ex. 00001)

### The username id should not be null or empty while entering.

### The second value are the coins entered by the user and should only be natural numbers.(Ex. 12)

### The third value is the Username of the User.(Ex. Username)

### The file cannot contain a null/empty value or wrong value.

### Error is thrown, if wrong value is inserted and the database is returned to the initial stage.

### Challenge 2 :-

Using this webservice (<https://www.q88.com/WS/Q88WSInternal.asmx?op=ConvertTemperature>), make a web calculator to do conversions of temperature. Preferably the results should be displayed asynchronously.

### The purpose of the test strategy of Challenge 2 is as follows :-

1. The Challenge is to consume the third party WS API given above and convert the Temperature from Celsius to Fahrenheit and vice versa.
2. The user is asked to select the option and enter the desired temperature value in the text box given.
3. The results are displayed asynchronously as per the inputs provided.

**Challenge 3 :-**

Lets assume we are an online marketing company. We got some new requirment to capture phone numbers of our customers every customer can have many phone numbers (1 customer : N phone numbers) and we need following 3 APIs from backend team:

1. Get all phone numbers
2. Get all phone numbers of a single customer
3. Activate a phone number

Note, provide documentation for frontend team so that they can implement implement your api in frontend.

### The purpose of the test strategy of Challenge 3 is as follows :-

### The Challenge is to capture the phone numbers of all the Customers and provide the API’s to the backend team.

### The first task is get all the phone numbers of the customers.

### The second task is to get all the phone numbers of a Customer by Customer id.

### The third is to activate a single phone number of the Customer among several others provided.