Global Full Stack Development Course

Front-end	Back-end
HTML 5, CSS 3, JS, React	TDD, Java, Jenkins, spring, git GitHub

Project(s)

Gerry and David

March 2021

Global Full Stack Development Course

FRONT END PROJECT

HTML, CSS, Styling Framework, JavaScript, React

Gerry and David

March 2021

Front End form one – driver details input

HTML, CSS, Bootstrap, React, JavaScript ……………….

Design a front-end user input form to collect driver data required to build an insurance quote. The form should be designed using a range of the 'technologies' taught during the front end section of the course. On completion the form will be used to connect to the back end, once it has been developed, so careful consideration should be given to the input component names. Attention should be given to the use of clean code and project structure to ensure that maintainability has been considered.

The information required from the user through the front end form will be:

1. What is the vehicle owners name?
    - **Prefix**
    - **First Name**
    - **Last Name**

2. What is the vehicle owners **telephone number**?

3. What is the vehicle owners Address?
    - **Address line 1**
    - **Address line 2**
    - **City**
    - **Postcode/Zip Code**

4. What is the **vehicle type**?
    - Cabriolet
    - Coupe
    - Estate
    - Hatchback
    - Other

5. What is the **engine size**
    - 1000
    - 1600
    - 2000
    - 2500
    - 3000
    - Other

6. How many **additional drivers** will use the vehicle (give the options 1, 2, 3, 4)?

7. Will the vehicle be **used for commercial purposes**? (Yes/No)?

8. Will the vehicle be **used outside the registered state**? (Yes/No)?

9. What is the **current value** of the vehicle (range 0 - 50000)?

10. **Date vehicle was first registered**?

11. A submit button which will be used to send the data from the input components to a Restful API which will return the car quote value.

12. Add JavaScript code to the form to validate relevant user inputs, rather than depending on the standard HTML validation.

13. Create an administrator web page i.e. a second web page that will have the following components and will be connected to the back end once it has been created:

    - an input box to accept a driver ID and fetches from the database the driver details and display the details (GET)

    - an input box to accept a driver ID and deletes the driver record from the database (DELETE)

    - an input box to accept a driver ID, an input box to accept a new driver telephone number and update the driver record in the database (UPDATE)

14. Add JavaScript code to the form to validate relevant user inputs, rather than depending on the standard HTML validation.

Global Full Stack Development Course

BACK END PROJECT

**Test Driven Development, Java, Spring,
Continuous Integration, Continuous Development**

Gerry and David

March 2021

Back End application to consume front end user inputs, store data and apply business logic to give an insurance quotation to the front-end user.

Data Types, Selection, Iteration, Arrays, Methods, File Handling, Classes

1. Develop a Maven application with a project name that includes your name.

2. Using a **Test Driven Development** approach create methods to calculate a series of factor values which will then be used in a calculation to find the quote amount. The methods should be for the following:

   - The vehicle type factor which is calculated as:

Cabriolet	factor is	1.3
Coupe	factor is	1.4
Estate	factor is	1.5
Hatchback	factor is	1.6
Other	factor is	1.7

   - The engine size factor is calculated as:

1000	factor is	1.0
1600	factor is	1.6
2000	factor is	2.0
2500	factor is	2.5
3000	factor is	3.0
Other	factor is	3.5

   - The vehicle value factor is calculated as:

<=5000	factor is	1.0
else	factor is	1.2

   - The additional drivers factor which is calculated as:

<2	factor is	1.1
else	factor is	1.2

   - The commercial use factor which is calculated as:

Yes	factor is	1.1
No	factor is	1.0

   - The outside state use factor which is calculated as:

• Yes	factor is	1.1
• No	factor is	1.0

3. Create a method to calculate the insurance quotation for the vehicle based on the factor values and the formula:

   100 * typeFactor * enginesizeFactor * driversFactor *commercialFactor * outsideStateFactor * vehicleValueFactor

4. Store all the user entered details in an Array or ArrayList i.e. prefix, first name, last name, telephone number, address line 1, address line 2, City, Postcodezip code, engine size, additional drivers, commercial use, outside state use, current value, date registered and the final quote amount.

5. Display the details from the Array or ArrayList using formatted output to the console.

6. Carefully consider the project structure including having separate classes for the methods that calculate the quote, the methods that store the quote details and the main method class.

Testing

7. Create a test class called **TestFactors** and write appropriate tests to validate that the methods calculate the different factors correctly e.g.

8. Create a test class called **TestQuoteAmount** and write appropriate tests for the method that calculates the quote amount. Use at least the following 2 tests e.g.

   - A hatchback with an engine size of 1600, 3 drivers, used commercially, outside the state and current market value of 5000 will receive a quote for $ 371.712 (100 * 1.6 * 1.6 * 1.2 * 1.1 * 1.1 * 1.0).
   - A Cabriolet with an engine size of 3000, 1 driver, not used commercially, not outside the state and current market value of 15000 will receive a quote for $ 514.80 (100 * 1.3 * 3.0 * 1.1 * 1.0 * 1.0 * 1.2).

9. Create a test suite class called **TestSuiteClass** which will be used to run the test classes you have just created. Example code is shown below:

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({
                TestFactors.class,
                TestQuoteAmount.class
              })

public class TestSuiteClass
{

}
```

10. Execute the tests using a Maven build configuration such as:

    test -Dtest=*.java

Continuous Integration and Continuous Development

11. Create a GitHub repository with your name included using your GitHub account.

12. Push the project you have just created to your newly created repository, using either the command line or from with the Integrated Development Environment.

13. Create a new Maven job in Jenkins with your name included, point it to your newly created GitHub repository.

14. Now build the job in Jenkins and check all the tests have passed.