

Software Dev Challenge

Purpose

The purpose of this challenge is to separate the strong applicants from the rest.

A strong applicant will:

- Follow instructions,
- Pay attention to details,
- Be able to explain how they have worked logically to solve a problem,
- Go above and beyond,
- Comment their code!

You are welcome to use Generative AI (GAI) to help you complete this challenge but please show how you have used it.

What Is Software Development?

This activity is designed to take 2-3 hours, but everyone is different, how long you take is not important. Our best applicants return this challenge in 5-7 days. Once completed this activity will be used to screen candidates and forwarded to employers who may use it for a discussion topic at interview.

To complete this activity, you don't need experience of writing code or developing solutions as everything can be searched for, Generative AI may be able to help, but remember you may be asked to explain your code at interview...

We are more interested in your ability to follow instructions, your attention to detail and desire to solve the problem than if you can cut and paste...

In this activity you will:

- Task 1: Review a series of [user stories](#)
- Task 2: Develop a simple web page using HTML, CSS, JavaScript.
- Task 3: Reflect on what you have learned,
- Task 4: Upload your challenge.

If you need them here are links to tutorials and reference guides for [HTML](#), [CSS](#) and [JavaScript](#)

This video [here](#) will show you how to create a web page with a text editor.

Task 1: Review User Stories:

Scenario:

As part of their online offer a training business wants to be able to validate a subscriber's name, email address and credit card number when they book a course.

The commercial director, marketing director and CTO were identified as stakeholders for this app and identified the User Stories in Appendix A:

Review the User Stories in Appendix A

If there are any areas that aren't clear you can make your own assumptions, just record these as part of your reflection.

Often, developers will identify the tests that their code needs to pass to be considered working.

This approach is called [Test Driven Development](#) we've simplified things for this example and have provided the Test Suite we will test your solution against in Appendix C.

You are to return the completed Test Suite.

Read these before you start to develop your solution.

Activity 2: Development time.

Using any IDE, you like (Visual Studio Code, Notepad++ etc...) or just a simple text editor and a browser, using only HTML, CSS, JavaScript create your solution to deliver the user stories.

There is no need to use JQuery, or any other libraries, no server is needed, this challenge can be completed with a text editor and a browser. Hint consider the mailto: tag.

Activity 3: Reflect

Create a 200+ word reflection on the activity.

How was working from User Stories?

What issues did you find with the User Stories?

What assumptions did you make about those issues?

How was it having the test cases before you started development?

How did you use Generative AI?

What are the risks of using Generative AI?

How did you implement the code?

What have you learned?

What would you do differently?

Activity 4: Upload your challenge.

Create a ZIP file with your name in the filename, put the following in it:

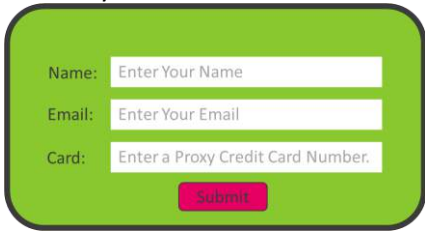
Add a copy of the .html .css and .js files you create.

Your complete Test Suite.

Your reflection

Upload your challenge as zip file to the questions at the bottom of the email containing this challenge. Or by providing a cloud storage link to where your challenge can be accessed to our apprenticeship training provider using the following details:

Appendix A – User Stories

ID	Stakeholder	User Story	
1	Commercial Director	As a user I need to be able to insert my name into the form i.e. At least a Given and Family name.	✓
2	Commercial Director	As a user I need to be able to insert my email into the form	✓
3	Commercial Director	As a user I need to be able to insert my Credit/Debit Card into the form. i.e. it conforms to the LUHN algorithm (see appendix B)	✗
4	Commercial Director	As a user I need to be able to send an email to challenge@dn-uk.com when the submit button is pressed and all fields are valid. i.e. use the mailto: HTML tag mailto: send an email to challenge@dn-uk.com with the entered data in the form.	✓
5	Marketing Director	<p>As the marketing director I need the page to be consistent with our other pages i.e., using the same Calibri font and text size 11, centred horizontally and vertically and to match the wireframe below:</p>  <p>Green RGB 137,200,46 Hex 89c82e Grey RGB 60,60,59 Hex 3c3c3b Pink RGB 231,0,100 Hex 700064</p>	✓
6	Marketing Director	As the marketing director I need the page to be consistent with our other pages any fields in error should be highlighted in Pink RGB 231,0,100 Hex 700064	✓
7	CTO	As CTO I need the application to reduce the load on the back-end server, validation should be done on page. i.e. JavaScript event onBlur() for Name and Credit Card number And HTML field type or Regex for Email.	✓
8	CTO	As CTO I need to ensure that any names entered are secure and there is no risk of database corruption through SQL injection. Only standard upper / lower case letters and permitted characters: !#\$%&'*+,-/=/?^_`{ }~ should be allowed in the name field.	✓

Appendix B – The LUHN Algorithm

The Luhn algorithm, also known as the **modulus 10** or **mod 10** algorithm, is a simple checksum formula used to validate a variety of identification numbers, such as credit card numbers. The LUHN formula was created in the late 1960s by a group of mathematicians. Shortly thereafter, credit card companies adopted it. Because the algorithm is in the public domain, it can be used by anyone. Most credit cards and many government identification numbers use the algorithm as a simple method of distinguishing valid numbers from mistyped or otherwise incorrect numbers.

It was designed to protect against accidental errors, not malicious attacks.

The Steps below show a simplified example for a 10 digit number, this can be used as a guide to help your understanding but for the case of a credit card the number would be 16 digits long.

You will need to write JavaScript code to implement each of these steps below to check the credit card number.

Note: we are looking for the steps below to be implemented, not just a cut and paste of an example from a web page...

Consider the example of an account number “79927398713”.

Step 0 Check that the length of the number is 11-19 characters in length.

Note: you should research the valid lengths for a credit card.

Step 1 Check the number only contains numeric characters.

i.e. 0123456789

Step 2 Check that the sum of the number is not 0.

i.e. is $7+9+9+2+7+3+9+8+7+1+3 = 0$?

Step 3 – Starting from the rightmost digit, double the value of every second digit,

7	9	9	2	7	3	9	8	7	1	3
	x2		x2		x2		x2		x2	
	18		4		6		16		2	

Step 4 – If doubling the number results in a two-digit number greater than 9, then add the digits of the product.

i.e. from the above $9 \times 2 = 18$ is greater than 9 so we use $1 + 8 = 9$ instead.

Same for $8 \times 2 = 16$ so we use, $1 + 6 = 7$, to get a single digit number.

7	9	9	2	7	3	9	8	7	1	3
	x2		x2		x2		x2		x2	
	18		4		6		16		2	
	9		4		6		7		2	

Step 5 – Now take the sum of all the single digits.

7	9	9	2	7	3	9	8	7	1	3
	x2		x2		x2		x2		x2	
	18		4		6		16		2	
7	9	9	4	7	6	9	7	7	2	3

Step 6 – If the total modulo 10 is equal to 0 (i.e., if the total ends in zero 10,20,30 etc) then the number is valid according to the Luhn formula; else it is not valid.

7	9	9	2	7	3	9	8	7	1	3
	x2		x2		x2		x2		x2	
	18		4		6		16		2	
7	9	9	4	7	6	9	7	7	2	3

$$7 + 9 + 9 + 4 + 7 + 6 + 9 + 7 + 7 + 2 + 3 = 70$$

Since the sum is 70 which is a multiple of 10, the account number is valid.

Below is an example in C++ code, you will need to produce code in JavaScript.

```
// C++ program to implement Luhn algorithm
#include <bits/stdc++.h>
using namespace std;

// Returns true if given card number is valid
bool checkLuhn(const string& cardNo)
{
    int nDigits = cardNo.length();

    int nSum = 0, isSecond = false;
    for (int i = nDigits - 1; i >= 0; i--) {

        int d = cardNo[i] - '0';

        if (isSecond == true)
            d = d * 2;

        // We add two digits to handle
        // cases that make two digits after
        // doubling
        nSum += d / 10;
        nSum += d % 10;

        isSecond = !isSecond;
    }
    return (nSum % 10 == 0);
}

// Driver code
int main()
{
    string cardNo = "79927398713";
    if (checkLuhn(cardNo))
        printf("This is a valid card");
    else
        printf("This is not a valid card");
    return 0;
}
```

Appendix C Simplified Test Suite

Below is a simplified Test Suite Template this contains the test cases that we will use to test your submitted challenge.

Test Case ID	User Story ID	Acceptance Condition	Preconditions	Test Data	Steps / Instructions	Expected Results	Actual Results	Pass / Fail	Notes
1	1	'John Doe' is a valid name	Web page showing validation screen is loaded. Web page contains input capable 'Name' field.	John Doe	Enter test data in name field	Validation passes, no action.	✓		
2	1	John is an invalid name (no family name)	Web page showing validation screen is loaded. Web page contains input capable 'Name' field.	John	Enter test data in name field	Name field highlighted in Pink	✓		
3	1	J Doe is an invalid name, initial and Family name	Web page showing validation screen is loaded. Web page contains input capable 'Name' field.	J Doe	Enter test data in name field	Name field highlighted in Pink	✓		
4	1	John Doe-Deer is a valid name, contains Given, Family and printable characters	Web page showing validation screen is loaded. Web page contains input capable 'Name' field.	John Doe-Deer	Enter test data in name field	Name field highlighted in Pink	✓		
5	1	John D is an invalid name, no Family name	Web page showing validation screen is loaded. Web page contains input capable 'Name' field.	John D	Enter test data in name field	Name field highlighted in Pink	✓		
6	1 / 8	John@Doe is invalid contains non permitted character.	Web page showing validation screen is loaded. Web page contains input capable 'Name' field.	John@Doe	Enter test data in name field	Name field highlighted in Pink	✓		
7	2	test@abc.com is a valid email	Web page showing validation screen is loaded. Web page contains input capable 'Email' field.	test@abc.com	Enter test data in email field	Validation passes, no action	✓		
8	2	@abc.com is an invalid email	Web page showing validation screen is loaded. Web page contains input capable 'Email' field.	@abc.com	Enter test data in email field	Email field highlighted in Pink	✓		
9	2	test@.com is an invalid email	Web page showing validation screen is loaded. Web page contains input capable 'Email' field.	test@.com	Enter test data in email field	Email field highlighted in Pink	✓		

10	2	test@abc is an invalid email	Web page showing validation screen is loaded. Web page contains input capable 'Email' field.	test@abc	Enter test data in email field	Email field highlighted in Pink	✓		
11	3	41111111111111111111 is a	Web page showing validation screen is loaded. Web page contains input capable 'Credit Card I' field	41111111111111111111	Enter test data in email field	Validation passes, no action	✓		
12	3	41111111111111111111 is an invalid entry – too long	Web page showing validation screen is loaded. Web page contains input capable 'Credit Card I' field	41111111111111111111	Enter test data in Credit Card field	Credit Card field highlighted in Pink			
13	3	41111111111111111111 is an invalid entry – too short	Web page showing validation screen is loaded. Web page contains input capable 'Credit Card I' field	41111111111111111111	Enter test data in Credit Card field	Credit Card field highlighted in Pink			
14	3	00000000000000000000 is an invalid entry – 0 Value	Web page showing validation screen is loaded. Web page contains input capable 'Credit Card I' field	00000000000000000000	Enter test data in Credit Card field	Credit Card field highlighted in Pink			
15	4	Submit button triggers email when all fields are valid.	Web page showing validation screen is loaded. Web page contains submit button. Name has valid name, Email has valid email, Credit Card has Valid Credit Card.	John Doe test@abc.com 41111111111111111111	Select Submit.	Desktop email Client is triggered, email created with the contents of the test data in in.			
16	4	Submit button doesn't send email when validation error	Web page showing validation screen is loaded. Web page contains submit button. Name has Invalid name, Email has invalid email, Credit Card has invalid credit card	John D @abc.com 41111111111111111111	Select Submit	No action			
17	4	Submit button doesn't send email when validation error	Web page showing validation screen is loaded. Web page contains submit button. Name has valid name, Email has invalid email, Credit Card has invalid credit card	John Doe @abc.com 41111111111111111111	Select Submit	No action 2 fields highlighted Pink Email and Credit Card.			
18	4	Submit button doesn't send email when validation error	Web page showing validation screen is loaded. Web page contains submit button. Name has valid name, Email has valid email, Credit Card has invalid credit card	John Doe test@abc.com 41111111111111111111	Select Submit	No action Credit Card field highlighted Pink			

19	5	Branding applied correctly	Web page showing validation screen is loaded.		Install colour picker and check on the CSS for colours displayed.	As per User Story.			
20	7	Correct OnBlur() Event Used	Web page showing validation screen is loaded.		View Page Source	OnBlur Event is used to call JavaScript validation functions for Name and Credit Card.			