Notes on development journey:

**List possible issues in Requirements: *(bracketed points to be researched)***

**1.** ok

**2.** ok

**3.** All details? Card type/card number/Security Code/Name/expiry.

**4.**define valid name. titles? full names/ names from predetermined list.

**5.**ensure format text@text.com, input type email

**6i.***(Note: There are 2 requirements numbered as 6, so will refer to the first as 6i and the second as 6ii.)* checks for validity - input type numbers, quantity of digits entered, expiry not exceeded, ping database?

**6ii.**branding guidelines - info?

**7.**text size =? (research suggests default text size is 16px)

**8.** (how to validate on page not back-end?)

**9.**(conform to W3C?)

**10.**UX guide required

**11.**(SQL injection?)

**12.**(LUHN algorithm?)

**13.**(perform compatibility check?)

**14.**(how to send info from submit?)

**Researched points:**

⦁ **How to reduce load on back-end server, validation done on page (8):**

Validation, ensuring that the data entered into input boxes is correct at the time of being entered by implementing strict rules for the input data. Allowing incomplete or incorrect information to be submitted into database will cause unnecessary load on back-end server.

⦁ **How to conform to W3C (9):**

To ensure the software is accessible to all users, full W3C list to be reviewed during development via https://www.w3.org/TR/WCAG10/full-checklist.html.

Some code editors may include a browser compatibility tool which will help to identify compliance with W3C.

⦁ **How to prevent SQL injection (11):**

SQL's are website attacks, allowing hackers to alter SQL queries. SQL is used in working with databases so SQL injection attacks have the potential to allow hackers to gain access to your data, which may be stolen or lost.

Prevention; simplest form of prevention is by filtering user input - validation methods for input boxes as such outlined for ID 8.

More complex attacks will require safeguarding processes outside of html,CSS or Javascript scope, to be passed to security team?

⦁ **How to conform to LUHN algorithm (12):**

LUHN algorithm is a checksum formula used to validate credit/debit card numbers, widely used by government and corporate organisations to protect against accidental errors when users are entering card numbers.

LUHN, also known as Modulus 10 or Mod 10 algorithm which can be implemented using C++, Java, Python3, C# and Javascript. (The latter 2 being most relevent for this exercise, algorithm for both languages saved to file under LUHNC# and LUHNJavascript respectively)

⦁ **How to run compatibility check for common browsers, devices and screen sizes (13).**

User testing - possible to test myself by installing various browsers, using personal mobile, tablet devices.

Code editors may include a browser compatibility tool to assist in identifying possible issues.

Responsive Web Design - <meta name="viewport" content="width=device-width, initial-scale=1.0">

this meta tag will assist with compatibility with varying screen sizes.

more info for reference found on https://www.w3schools.com/html/html\_responsive.asp

⦁ **How to email validated information upon submission (14):**

Explore mailto: tag

more sophisticated methods require PHP? - look into this.

**Development Test Suite:**

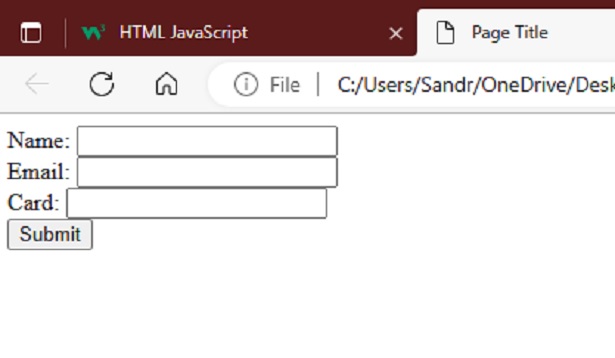
Add visual checks or purely functional?

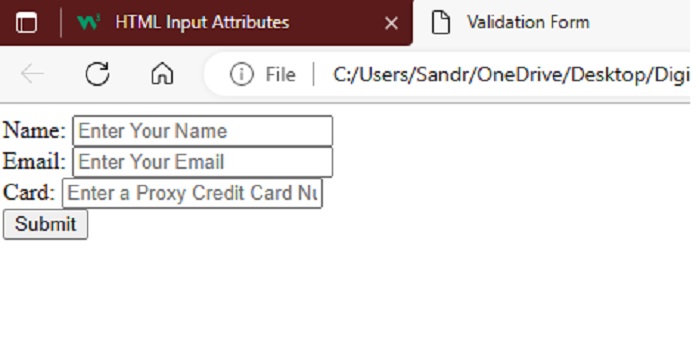
Findings: Both functional and non-functional elements should be included in TTD for best results. All requirements will be listed in Test Suite for this exercise.

**Development Time:**

Html:

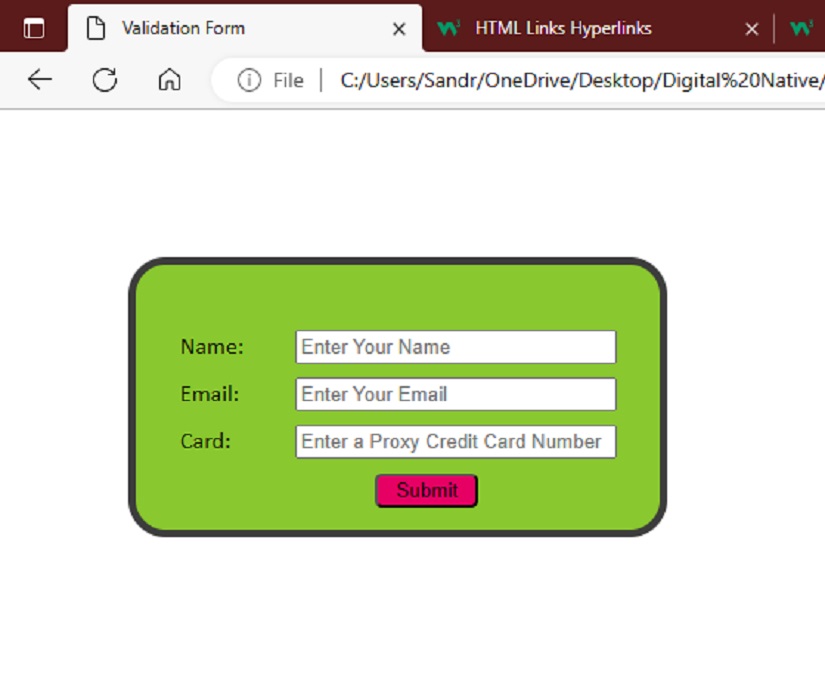
Basics found in link provided to create initial content:





Styling now to be added using CSS, validation scripts to be added using Javascript.

**CSS:**



Found hexidecimal colour for Pink shows incorrectly, rgb code is correct so will swap other colours into rgb for more consistent syntax.

# Javascript:

My first time looking at Javascript, however some of the rules and keywords appear similar to that of Python.

Bit of trouble with red/green on keyup- EventListeners good for this.

LUHN big sticking point.

Email upon submit, another sticking point. mailto is easy but not user friendly at all. Looks to be that due to information security regulations I need to set up a server/or pay for a hosted one, and use PHP to be able to send an email with encrypted data.

Try to host local server - set up IIS for web hosting - didn't get as far as I was expecting. disabled IIS.

WAMP - (windows, Apache, MySQL, PHP) block web requests on firewall. forward port 80 on router. ok.

No joy for product testing, altering settings further that I don't fully understand would be too risky for my system security. Getting a cheap host may be worth it for the time being.

www.sandraacko.co.uk . FTP (Filezilla). not needed - inbuilt in host.

php.ini - SMPT details - still no joy

PHPMailer via Composer - no joy

PHPMailer - manually installed, file by file, into webhost - still no joy (with many tweeks and versions of php files/code) :(

Tried converting index.html to index.php - no fix

Need SSL certificate for website for email request to be accepted? a little costly and still may not be the sloution.

**Conclusion:** after many hours of trying, I'll just have to provide my only ‘working’ solution until I can gain further experience, although it is certainly not what I would like. regulations and restrictions surrounding the sending of emails can be incredibly difficult to navigate. I have to admit that I require help with this.

# Testing Suite officially commenced:

Time to begin working through the Testing Suite, testing and re-testing has been done frequently throughout development as I am still learning the languages but I am ready to do the formal testing process now that I am preparing to submit my files.

Testing Complete

Change Email recipient on Mailto: tag to test@dn-uk.com.

Submitted project still viewable on www.sandraacko.co.uk.