Introduction and Concept:

Public transportation faces challenges in reducing emissions while encouraging more ridership. Current systems lack engaging incentives for passengers to actively support sustainability. Which is why, I introduced ‘PowerStride’ as a solution.

PowerStride, is an extension of the Leap Top Up App. The fundamental idea is that a user Sign In and goes to the Home page. From there, they can check if they are eligible to win the reward and if they do, then it reflects on their account.

It is a mobile first web application and has been developed using HTML, CSS and JavaScript. There has been extensive changes in the final design as compared to the initial proposed design in Design Practice.

The 4 pages I developed are:

Login Page: The user is required to input their Leap Card Number, email address, password and Date of Birth to access the homepage. The user inputs into the text fields are validated to process further.

Home Page: The PowerStride home page is fairly straightforward with minimal javascript and mainly HTML and CSS. It has a responsive hamburger menu icon which is toggled when the screen width reduces, and turned into a Drop Down Menu for the top nav tabs.

Transactions Page: I created this page to show the past transactions using the leap card by using Arrays and functions.

Leap Card Page: The leap card page has a dynamic circular slider which the user can interact with. When the user rotates the slider, the value changes in equal intervals. The page also has the ‘Check’ feature button for PowerStride which lets the user know that they have won a reward and it reflects on their account balance. If the user clicks the button again, they don’t get the reward but instead a prompt is displayed saying to logout and try again.

1. Inclusion of at least 4 different GUI controls

The provided JavaScript code demonstrates the use of the following GUI controls:

- **Text Box**:

  Used in the login form for inputs like Leap Card Number, Email, and Password.

- **Button**:

  Used for form submission and the "Powerstride" prize claim button on the Leap Card page.

- **Slider**:

  A circular slider implemented on the Leap Card page for selecting top-up amounts.

- **Table**:

  Dynamically populated transaction data is displayed in a table format on the Transactions page.

- **Drop Down**:

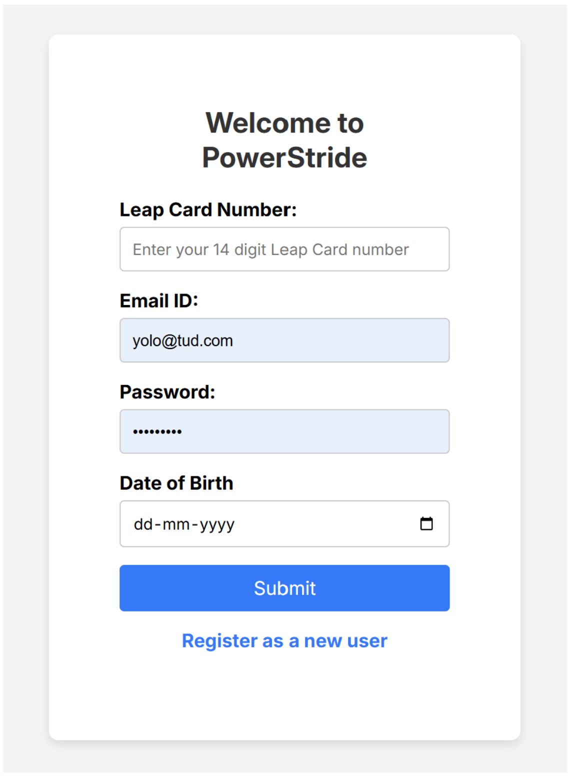
  Used in the navigation bar when the design is responsive as well as to show the Leap features

---

2. An HTML form containing at least 3 User Input elements

The login page includes a form (userForm) with the following input elements:

- Leap Card Number (Text Box)
- Email (Text Box)
- Password (Password Field)
- Date of Birth (Date Input Field)


Welcome to PowerStride

Leap Card Number:

Enter your 14 digit Leap Card number

Email ID:

yolo@tud.com

Password:

•••••••••

Date of Birth

dd-mm-yyyy

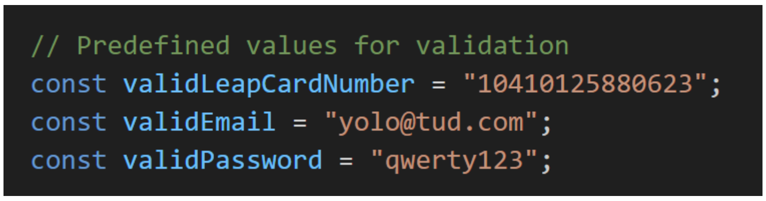
---

3. Proper validation of these input elements using JavaScript

The login form's JavaScript performs comprehensive validation:

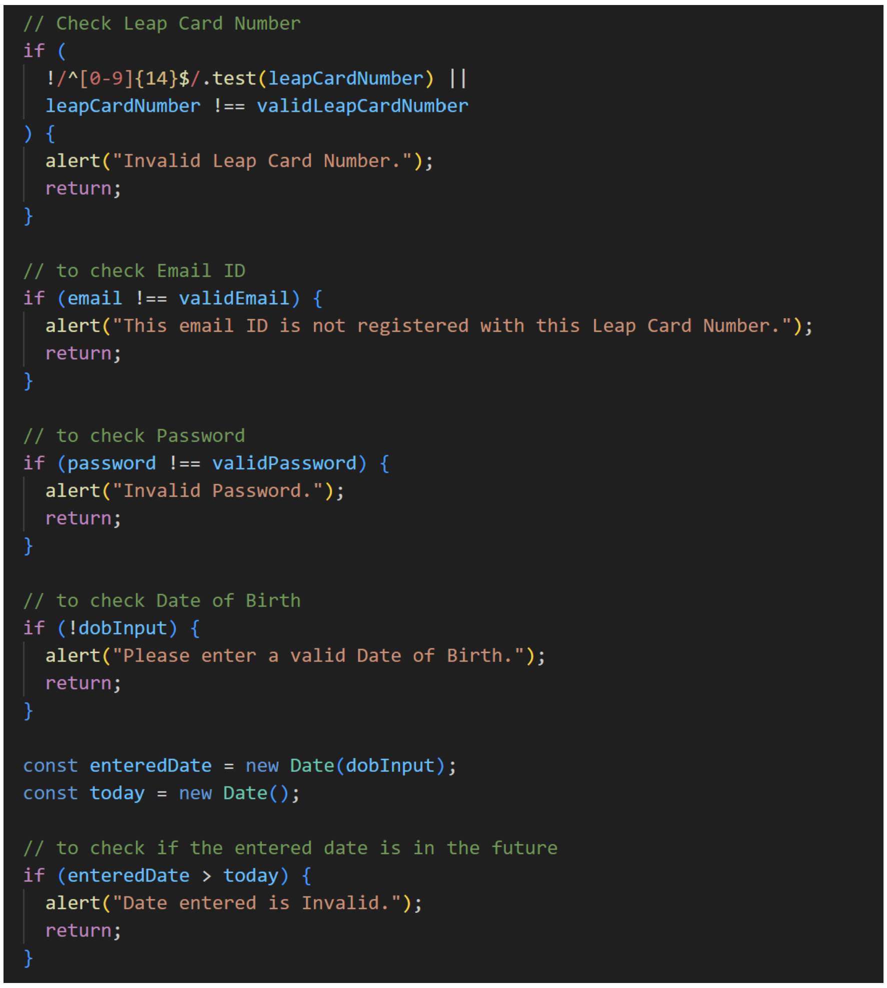
- **Leap Card Number**: Checks for a valid 14-digit number and matches it with a predefined valid number.
- **Email**: Validates the email against a predefined registered email ID.

- **Password**: Ensures the entered password matches the predefined valid password.



```
// Predefined values for validation
const validLeapCardNumber = "10410125880623";
const validEmail = "yolo@tud.com";
const validPassword = "qwerty123";
```

- **Date of Birth**:
  - Ensures the input is not empty.
  - Verifies that the entered date is not in the future.
  - Calculates the user's age to ensure they are above 18 years old.

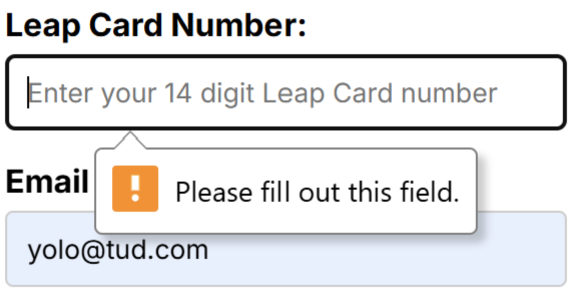


```
// Check Leap Card Number
if (
  !/^[0-9]{14}$/.test(leapCardNumber) ||
  leapCardNumber !== validLeapCardNumber
) {
  alert("Invalid Leap Card Number.");
  return;
}

// to check Email ID
if (email !== validEmail) {
  alert("This email ID is not registered with this Leap Card Number.");
  return;
}

// to check Password
if (password !== validPassword) {
  alert("Invalid Password.");
  return;
}

// to check Date of Birth
if (!dobInput) {
  alert("Please enter a valid Date of Birth.");
  return;
}

const enteredDate = new Date(dobInput);
const today = new Date();

// to check if the entered date is in the future
if (enteredDate > today) {
  alert("Date entered is Invalid.");
  return;
}
```

4. Event Handlers for at least 4 events on the website

The JavaScript includes the following event handlers:

- **Form Submission**: Prevents default submission behavior and validates user inputs.



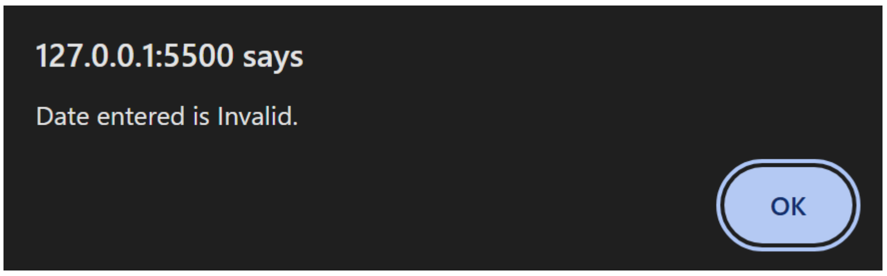
Leap Card Number:

Enter your 14 digit Leap Card number

Please fill out this field.

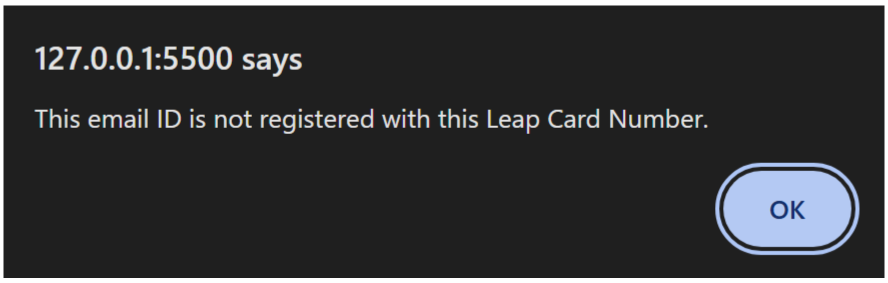
Email

yolo@tud.com



127.0.0.1:5500 says

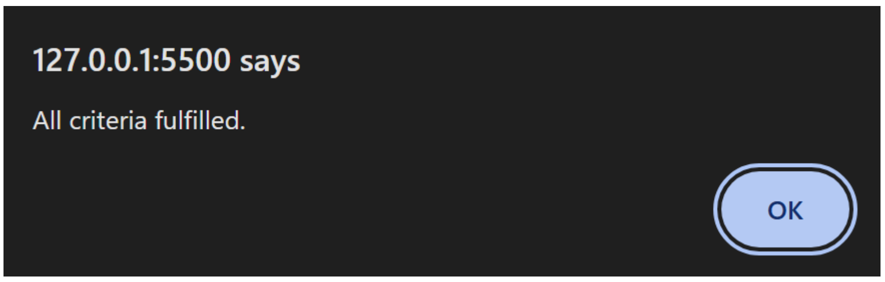
Date entered is Invalid.



127.0.0.1:5500 says

This email ID is not registered with this Leap Card Number.

Displaying errors after incorrect validation



127.0.0.1:5500 says

All criteria fulfilled.

Displaying correct validation before proceeding to next screen

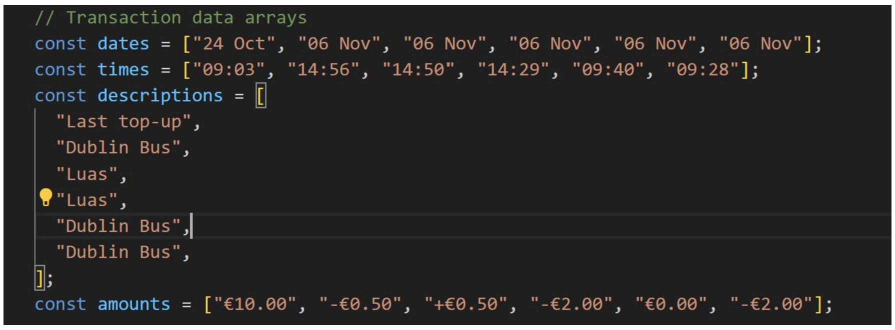
- **Mouse Down and Mouse Move**: Handles dragging of the circular slider handle.
- **Mouse Up**: Stops slider rotation after the user releases the mouse.
- **Button Click**: Handles the "Powerstride" prize claim functionality, displaying alerts based on conditions.
- **Page Load**: Initializes the handle's position and transaction data population using the DOMContentLoaded event.

---

5. At least 4 Arrays demonstrating creating, updating, and displaying data

Arrays are used extensively to store and display transaction data dynamically:

- **Arrays Used**: dates, times, descriptions, and amounts.
- **Usage**:
    - Arrays are created to store transaction data.
    - The populateTransactions() function loops through these arrays to dynamically generate table rows.



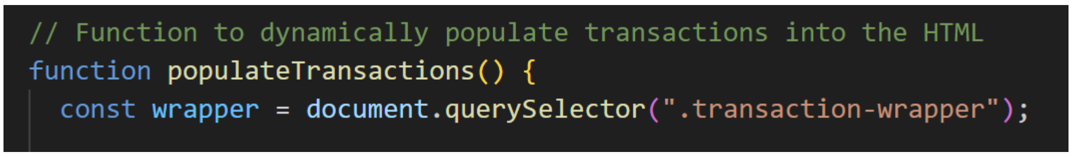
```
// Transaction data arrays
const dates = ["24 Oct", "06 Nov", "06 Nov", "06 Nov", "06 Nov", "06 Nov"];
const times = ["09:03", "14:56", "14:50", "14:29", "09:40", "09:28"];
const descriptions = [
  "Last top-up",
  "Dublin Bus",
  "Luas",
  "Luas",
  "Dublin Bus",
  "Dublin Bus",
];
const amounts = ["€10.00", "-€0.50", "+€0.50", "-€2.00", "€0.00", "-€2.00"];
```

---

6. At least 4 Functions for the reuse of functionality

The provided JavaScript code includes multiple reusable functions:

- updateTopUpValue(): Calculates and updates the top-up value based on the slider's angle.
- updateHandlePosition(): Adjusts the position of the slider handle dynamically.
- rotateSlider(event): Calculates the slider's rotation angle and updates the handle's position.
- populateTransactions(): Dynamically populates the transaction table using data from arrays.



```
// Function to dynamically populate transactions into the HTML
function populateTransactions() {
  const wrapper = document.querySelector(".transaction-wrapper");
```

- Additional functions are used for form validation (e.g., age calculation).

---

7. The use of one or more external JavaScript files

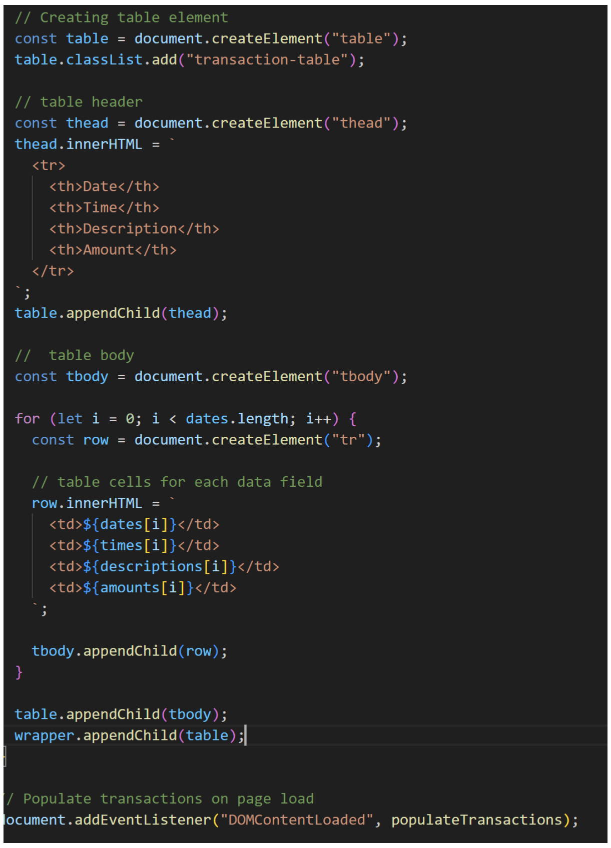
- The JavaScript for each page is modularized into separate files (login.js, home.js, transactions.js, and leapcard.js).
- This demonstrates a clear separation of concerns, making the code reusable and maintainable.

---

8. Demonstration of dynamic manipulation of the DOM

Dynamic DOM manipulation is demonstrated in multiple ways:

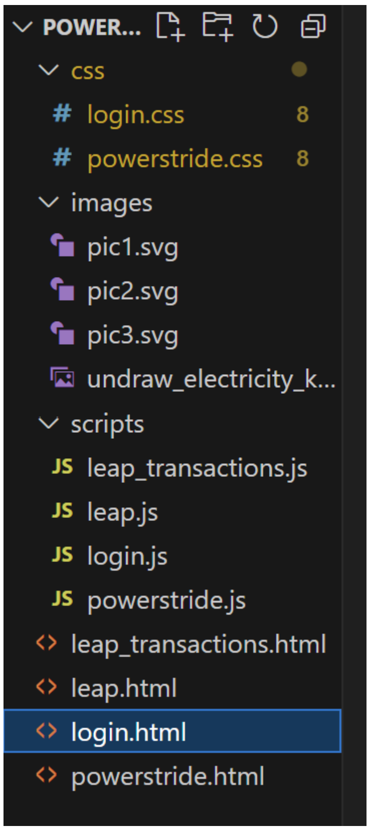
- **Transaction Table**: Rows are dynamically generated and appended to the table on the Transactions page.



```
// Creating table element
const table = document.createElement("table");
table.classList.add("transaction-table");

// table header
const thead = document.createElement("thead");
thead.innerHTML = `
  <tr>
    <th>Date</th>
    <th>Time</th>
    <th>Description</th>
    <th>Amount</th>
  </tr>
`;
table.appendChild(thead);

//  table body
const tbody = document.createElement("tbody");

for (let i = 0; i < dates.length; i++) {
  const row = document.createElement("tr");

  // table cells for each data field
  row.innerHTML = `
    <td>${dates[i]}</td>
    <td>${times[i]}</td>
    <td>${descriptions[i]}</td>
    <td>${amounts[i]}</td>
  `;

  tbody.appendChild(row);
}

table.appendChild(tbody);
wrapper.appendChild(table);


/ Populate transactions on page load
ocument.addEventListener("DOMContentLoaded", populateTransactions);
```

- **Circular Slider**: The slider handle's position is dynamically updated based on user interaction.
- **Form Validation Alerts**: Alerts are displayed dynamically based on user input validation

9. Appropriate directory structure and use of relative URLs



```
POWER...
  css
    login.css            8
    powerstride.css      8
  images
    pic1.svg
    pic2.svg
    pic3.svg
    undraw_electricity_k...
  scripts
    leap_transactions.js
    leap.js
    login.js
    powerstride.js
  leap_transactions.html
  leap.html
  login.html
  powerstride.html
```

All files are well segregated and organised in folders and have the HTML, CSS and JS files have been linked using URL`s

Conclusion:

The provided JavaScript code satisfies all the specified criteria for Authoring Principles. It demonstrates a well-structured and functional implementation of a dynamic web application with appropriate validations user interactions, and modularized scripts.

To create the DOB, I used code from Codepen.io (https://codepen.io/stevenhanley0/pen/VjRKGq) and the slider has been derived from ChatGPT after giving appropriate logic.
For the slider, for every 30 degrees the user rotates it, an increment of +5 should take place and be reflected.

Additionally, I have used ChatGPT to refine and restructure my code in certain places and get clarity on code that I don't know about. Even though I am aware of the logic on how to move forward , I am still unaware of the correct terminology, the syntaxes, and the coding structure and language which is to be used. At these places, using help from ChatGPT and codepen.io was quite beneficial and crucial for my work.