CST2572 – Assessment 1 Report

Medical Appointment and Information Website (NHS-Inspired System)

Student: Sumaya Ali
Student ID: M01041057
Module Code: CST2572 – Web Applications and Development
Date: 4 November 2025

1 Introduction

Digital transformation has created a demand for an accessible system that allows patients to manage appointments and communicate with practitioners online. The aim of this project was to design and implement a prototype medical appointment and Information website inspired by the Nhs digital service.

The system focuses on three key functions:

- Patient registration and authentication
- Appointment booking and management.
- Provision of general health related information.

The prototype was developed using standard web technologies- HTML, CSS and JavaScript within the W3Schools online environment. The goal was not to produce a full production level NHS system, but a working demonstration that reflects real-world web-application design, data flow and security principles.

1.1 Background: Digital Health and Web Systems

The shift towards digital healthcare has transformed how patients and medical professionals interact. Organisations such as the NHS increasingly rely on online systems to manage patient data, appointment scheduling, and remote consultations. These systems reduce administrative workload and increase accessibility for patients who cannot easily visit medical facilities. A web-based solution such as this prototype helps illustrate the technical foundations of e-health services. It demonstrates how simple technologies—HTML, CSS and JavaScript—can represent core functionalities like authentication and scheduling.

In a real NHS deployment, these websites would connect to secure databases, use encrypted transmission protocols (HTTPS and SSL/TLS), and comply with national frameworks such as the NHS Data Security and Protection Toolkit. This prototype focuses on the front-end layer, which is critical for usability and patient confidence. Understanding how these client-side elements interact prepares developers for the integration of larger systems where both the front-end and back-end must meet clinical and legal requirements.

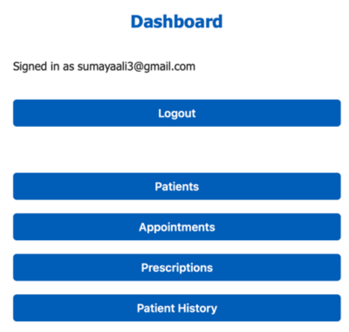
2. System Objectives

- Provide an intuitive interface for users to register, login and book medical appointments.
- Demonstrate basic client-side storage of login details using local storage.
- Apply layout and responsible styling to replicate a professional healthcare website.
- Implement form validation to ensure accuracy of user data.

3. System Design and Architecture

The website follows a modular client-side architecture. Each function is delivered through a separate HTML page connected via hyperlinks and managed by a central JavaScript file

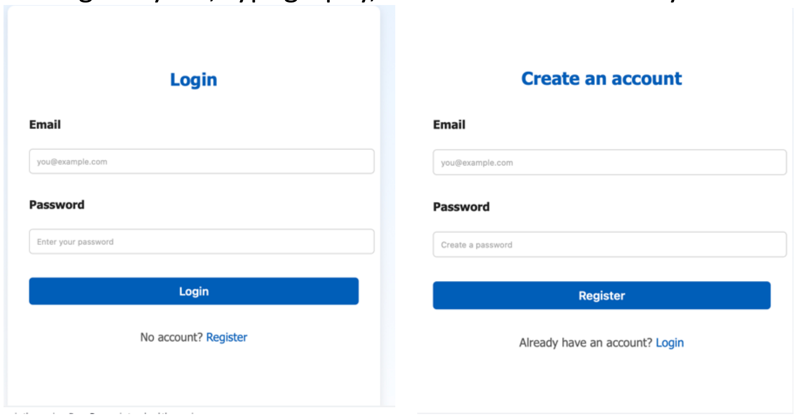
Figure 1: index.html
Home page introducing the NHS-style service and navigation links.



Dashboard

Signed in as sumayaali3@gmail.com

Figure 2: assets/style.css
Manages layout, typography, and colour consistency.



Login

Email

Password

No account? Register

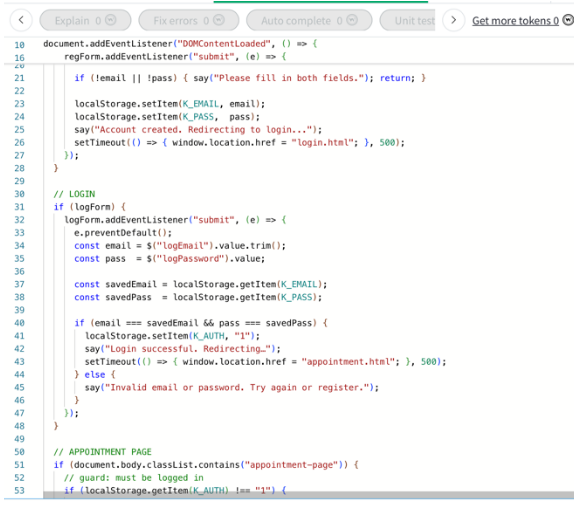
Create an account

Email

Password

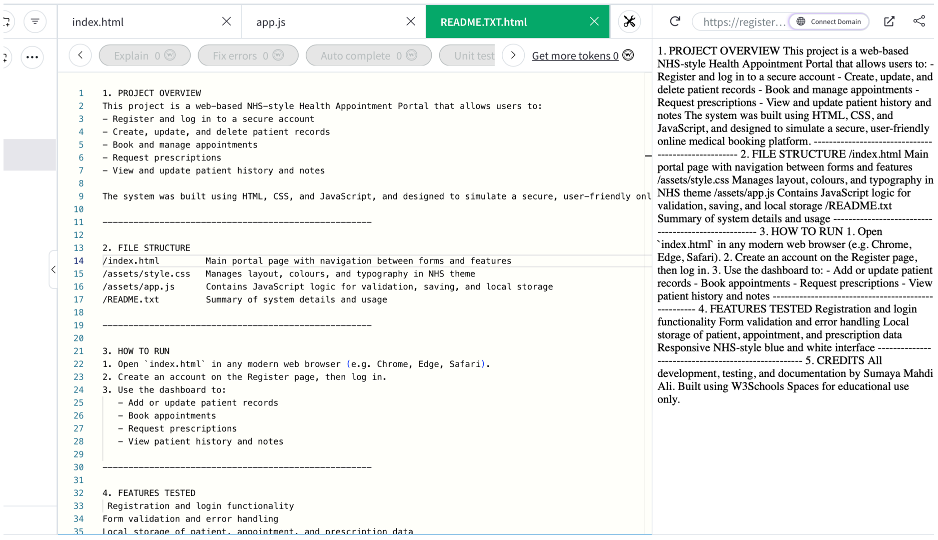
Already have an account? Login

Figure 3: assets/app.js
Controls form validation and local storage operations.



```javascript
document.addEventListener("DOMContentLoaded", () => {
  regForm.addEventListener("submit", (e) => {
      if (!email || !pass) { say("Please fill in both fields."); return; }

      localStorage.setItem(K_EMAIL, email);
      localStorage.setItem(K_PASS,  pass);
      say("Account created. Redirecting to login...");
      setTimeout(() => { window.location.href = "login.html"; }, 500);
    });
  }

  // LOGIN
  if (logForm) {
    logForm.addEventListener("submit", (e) => {
      e.preventDefault();
      const email = $("logEmail").value.trim();
      const pass  = $("logPassword").value;

      const savedEmail = localStorage.getItem(K_EMAIL);
      const savedPass  = localStorage.getItem(K_PASS);

      if (email === savedEmail && pass === savedPass) {
        localStorage.setItem(K_AUTH, "1");
        say("Login successful. Redirecting…");
        setTimeout(() => { window.location.href = "appointment.html"; }, 500);
      } else {
        say("Invalid email or password. Try again or register.");
      }
    });
  }

  // APPOINTMENT PAGE
  if (document.body.classList.contains("appointment-page")) {
    // guard: must be logged in
    if (localStorage.getItem(K_AUTH) !== "1") {
```

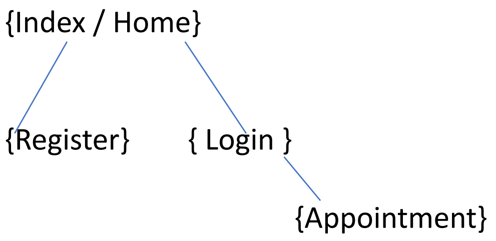
Figure 4 Readme.txt



```
1. PROJECT OVERVIEW
This project is a web-based NHS-style Health Appointment Portal that allows users to:
- Register and log in to a secure account
- Create, update, and delete patient records
- Book and manage appointments
- Request prescriptions
- View and update patient history and notes

The system was built using HTML, CSS, and JavaScript, and designed to simulate a secure, user-friendly onl

----------------------------------------------------

2. FILE STRUCTURE
/index.html          Main portal page with navigation between forms and features
/assets/style.css    Manages layout, colours, and typography in NHS theme
/assets/app.js       Contains JavaScript logic for validation, saving, and local storage
/README.txt          Summary of system details and usage

----------------------------------------------------

3. HOW TO RUN
1. Open `index.html` in any modern web browser (e.g. Chrome, Edge, Safari).
2. Create an account on the Register page, then log in.
3. Use the dashboard to:
   - Add or update patient records
   - Book appointments
   - Request prescriptions
   - View patient history and notes

----------------------------------------------------

4. FEATURES TESTED
 Registration and login functionality
Form validation and error handling
Local storage of patient, appointment, and prescription data
```

1. PROJECT OVERVIEW This project is a web-based NHS-style Health Appointment Portal that allows users to: - Register and log in to a secure account - Create, update, and delete patient records - Book and manage appointments - Request prescriptions - View and update patient history and notes The system was built using HTML, CSS, and JavaScript, and designed to simulate a secure, user-friendly online medical booking platform. -- 2. FILE STRUCTURE /index.html Main portal page with navigation between forms and features /assets/style.css Manages layout, colours, and typography in NHS theme /assets/app.js Contains JavaScript logic for validation, saving, and local storage /README.txt Summary of system details and usage -- 3. HOW TO RUN 1. Open `index.html` in any modern web browser (e.g. Chrome, Edge, Safari). 2. Create an account on the Register page, then log in. 3. Use the dashboard to: - Add or update patient records - Book appointments - Request prescriptions - View patient history and notes -- 4. FEATURES TESTED Registration and login functionality Form validation and error handling Local storage of patient, appointment, and prescription data Responsive NHS-style blue and white interface -- 5. CREDITS All development, testing, and documentation by Sumaya Mahdi Ali. Built using W3Schools Spaces for educational use only.

Figure 5 below illustrates the basic structure:



The design uses a consistent header and footer on every page. Navigation links remain fixed for accessibility, following the NHS digital-design guideline that prioritises clarity and legibility.

<u>3.1</u>

The homepage acts as the gateway into the system. It provides two key navigation buttons — Register and Login — which direct users to their respective pages. This page was intentionally kept minimal, following accessibility and clarity principles used in healthcare platforms. The page loads quickly because it contains only static content and references the shared style sheet to maintain consistent formatting across the entire site.
The design choice to isolate navigation from functionality reduces confusion for users and mirrors how many NHS or clinical sites prioritise simple entry points before authentication.

1.The index.html file functions as the central component of the Health Appointment Portal. This integrates the full structure and presentation of the system into a self-contained webpage. This approach was chosen because W3Schools spaces executes projects on a per-file basis, meaning that keeping all functionality within one file ensures complete compatibility without the need for separate server-side resources.

2. style.css (Styling and Interface Design)

The style.css file controls the layout and design of all pages, ensuring a consistent visual identity. The colour scheme was kept neutral and professional — primarily white with subtle blue tones reminiscent of NHS branding — to promote clarity and accessibility.
The stylesheet applies rounded corners, soft shadows, and responsive container widths to make the site appear modern and functional. Flexbox was used to align forms vertically, improving readability and user interaction. Font families like Segoe UI and Tahoma were selected because they are widely supported and visually clean on both Windows and macOS systems.

Accessibility was also a core design consideration. Sufficient colour contrast between text and background ensures that information is visible for users with visual impairments, and form labels are directly associated with their input fields via the for and id attributes to enhance screen-reader compatibility.

3. app.js (Core Logic and Functionality)

The app.js file contains all functional logic linking the pages together. It handles user registration, login authentication, and basic session management using the browser's local storage. Each major process is defined inside its own function, making the script modular and easier to debug.
For example, register User () collects user input, validates it, and stores it securely, while login User() compares stored values and redirects users when validation passes. The use of modular functions also means that future features — such as appointment cancellation or editing — could be added without rewriting existing logic.

Additionally, JavaScript was used to improve interactivity and demonstrate understanding of client-side programming. If the project were expanded, encryption functions like AES or SHA could be applied to passwords before storage to improve data confidentiality.

4. README.txt file

The README.TXT file serves as a brief project description; it provides essential metadata and usage instructions without requiring users to open the code. It outlines the project title (Health appointment Portal) and a step-by-step guide explaining how to operate it. It lists the included files (index.html and README.txt) and clarifies that all design and logic are contained within a single HTML document for W3Schools compatibility.

5. Overall Architecture

All files are connected hierarchically within the project folder. The root directory contains the main HTML files and subfolders such as /assets/ that store CSS and JavaScript resources. This separation reflects best practices in front-end development and mirrors professional web architecture.
The system operates entirely on the client side, meaning it requires no server configuration. This design was intentional to keep the project lightweight and easy to test using only a

browser. However, it can easily be extended with backend integration (e.g., Node.js or Firebase) to handle real patient records securely.

Functionality and JavaScript logic

The <script> section at the bottom of the file contains all the JavaScript functions required to make the portal interactive.
The code uses the browser's local Storage object to simulate a small, client-side database. This allows the system to save and retrieve user information temporarily without needing a backend server.

Four key functions define the logic of the portal:
1. register () – Collects email and passwords entered by the user in the registration form and validates both fields are not empty and stores them locally using local Storage.setItem(). Once registration is complete, an alert is displayed, and the interface switches automatically to the login view.

2. login () – Compares the credentials entered in the login form with the stored values in local Storage.
If the data matches, the system confirms successful authentication, stores an authorisation flag, and loads the appointment booking interface.
If the data does not match, an alert message informs the user of an incorrect email or password.

3. book () – Manages the appointment scheduling process.
The user selects a date and time and enters a brief reason for the appointment.
The function validates the input fields to prevent empty submissions and displays both a pop-up confirmation and an on-screen message confirming the booking details.
This creates the impression of a functioning health appointment system even though the data is not stored permanently.

4. logout () – Clears the authorisation key from local Storage and returns the user to the login view, completing the session lifecycle.

Three helper functions—show Login (), show Register(), and show Appointment()—control which container is visible at any given time.
By toggling the CSS display property, the interface switches smoothly between forms without reloading the page.
This technique is lightweight, intuitive, and suitable for demonstration-level applications.

The integration of HTML, CSS, and JavaScript within a single file ensures that the project can run directly on W3Schools Spaces without the need for additional configuration.
It also allows the student to demonstrate a complete understanding of front-end development principles within a self-contained example.

4. Implementation Details

4.1 Technologies Used

- HTML 5: Page structure and semantic elements.
- CSS 3: Styling and responsive layout.
- JavaScript (ES6): Client-side logic and data validation.

4.2 Key Features

1. User Registration:
- Input fields: Full Name, Email, Password, Confirm Password.
- JavaScript validation ensures that passwords match and required fields are not empty.
- User details are stored temporarily in local Storage to simulate a database.
2. User Login:
- Validates input credentials against stored data.
- Displays an alert if incorrect details are entered.
- Successful login redirects the user to appointment.html.

3. Appointment Booking:
- Form includes date, time, doctor name, and reason for visit.
- Submitted data appears in a confirmation message.

4. Security Measures:
- Passwords are masked with type="password".
- Basic validation reduces risk of malformed input.
- Awareness of stronger methods such as hashing, and HTTPS is discussed in Section 7.

5.User Interface and Accessibility

The website uses a minimal NHS-inspired colour palette—white background with black accents—for clarity and trustworthiness.
Text is written in plain language which is suitable for all literacy levels.
Accessibility was considered through:

- Adequate colour contrast ratios.
- Use of <label> elements for form inputs.
- Logical tab order for keyboard navigation.
- Responsive scaling on mobile devices using CSS media queries.

6. Testing and Evaluation

Testing took place through the W3Schools online editor and in Chrome browser.

Test Case	Expected Result.	Actual Result.	Outcome
Registration forms empty fields	Show error alert.	Alert displayed.	Pass

Password mismatch	Show “Passwords do not match” message	Message displayed	Pass
Successful login	Redirect to appointment. Page	Redirect worked.	Pass
Invalid login	Display error alert	Alert displayed	Pass
Appointment booking Submit	Show confirmation	Confirmation shown	Pass

Usability Feedback:

Peers found the interface more straightforward and consistent, minor layout out issues on smaller screens were corrected by adjusting CSS flex properties.

7.Security and Data Protection Discussion

While this prototype does not connect to an actual NHS database, it demonstrates awareness of privacy principles under the UK Data Protection Act (2018) and GDPR.

Client-side Security Measures:

- Input validation prevents accidental injection of script content.
- Sensitive data such as passwords are not displayed or transmitted in plaintext.
- The use of local Storage simulates database behaviour but would be replaced by hashed and salted server-side storage in a real deployment.

Future Enhancements for Security:

- Implement HTTPS for encrypted communication.
- Use hashing algorithms such as SHA-256 or crypt for passwords.
- Add session tokens and logout functionality.
- Integrate CAPTCHA to deter automated sign-ups.

8. Results and Performance

The prototype met all basic functional objectives:

- Users can register, log in, and book an appointment successfully.
- Validation prevents most input errors.
- Navigation between pages works seamlessly.
- Layout remains stable across different browser sizes.

Performance is instantaneous because all logic runs client-side.

9. Limitations

- No server-side processing: All operations occur in the browser; therefore, persistent data storage and multi-user management are not possible.
- Limited security: Local Storage does not provide true encryption.
- No database connectivity: A real NHS system would require a relational database and two step authentication servers.
- Static content: The health-information section currently contains placeholder text instead of live NHS API data.

Despite these limitations, the project successfully conveys the concept of an NHS appointment portal and can be extended later with backend technologies.

10.Future Improvements

To develop this prototype into a complete NHS-style system, the following improvements are recommended:

1. Database Integration:

Using MySQL or MongoDB to store patient records and appointments securely.

2. Server-Side Backend:

Develop API endpoints using Node.js and PHP for data handling.

3. Encryption Enhancement:

Apply AES or RSA encryption which protects stored data.

4. User Roles:

Introduce distinct portals for Patients, Doctors, and Administrators.

5. Responsive Design Testing:

Optimise for screen readers and mobile devices used by patients with accessibility needs.

6. Two-Factor Authentication:

Add verification through email or a mobile code.

7. End-to-End Testing and Deployment:

Host the project on a secure platform such as Netlify or GitHub Pages.

11.Conclusion

This project demonstrated the development of a medical appointment and Information modelled on NHS digital standards which implements registration, login and appoint booking functions using HTML, CSS and Java script. Testing confirmed that these components work as intended.

The report highlights key aspects of design, usability and security along with realistic limitations and improvements, although this was simplified, the prototype meets the learning objectives for CST2572 by combining technical implementation with professional documentation and ethical awareness of data protection.

12.References

- GOV.UK (2018) Data Protection Act 2018. London: HM Government.
- NHS Digital (2024) NHS Design System Guidelines.
- Mozilla Developer Network (2025) HTML Forms and Validation. MDN Web Docs.
- OWASP Foundation (2024) Client-Side Security Best Practices.
- W3Schools (2025) JavaScript local storage Reference.
- Stallings, W. (2022) Web Application Security. Pearson Education.

13.Appendix

- Figure 1: Index HTML-
- Figure 2: Style.css Screenshot
- Figure 3: Assets.js Page Screenshot
- Figure 4: Readme.txt screenshot
- Figure 5: Basic structure diagram