ĐẠI HỌC XÂY DỰNG HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN

LÊ QUANG HUY – MSSV: 1334770

HÀ TRUNG HIẾU – MSSV: 1212070

VĂN ĐÌNH THÀNH TRUNG – MSSV: 1335370

LÊ ĐỨC TRƯỜNG – MSSV: 1234270

NGUYỄN QUANG KHÁNH – MSSV: 1007370

BÀI TẬP DỰ ÁN

XÂY DỰNG FORM XÉT HỌC BỔNG TRỰC TUYẾN

GIẢNG VIÊN HƯỚNG DẪN
TS. HOÀNG NAM THẮNG

TP. HÀ NỘI, 2025

Mục lục

TÓM TẮT

Trong bối cảnh số hóa quy trình quản lý và tuyển chọn sinh viên, các biểu mẫu trực tuyến đóng vai trò quan trọng trong việc thu thập, lưu trữ và xử lý dữ liệu. Đề tài *Xây dựng form xét học bổng trực tuyến* được thực hiện với mục tiêu thiết kế và triển khai một biểu mẫu web giúp thu thập thông tin phục vụ cho công tác xét học bổng sinh viên.

Dự án sử dụng các công nghệ nền tảng của lập trình web phía client bao gồm HTML, CSS và JavaScript. HTML được dùng để xây dựng cấu trúc biểu mẫu; CSS đảm nhiệm phần giao diện và bố cục, hướng tới trải nghiệm người dùng tương tự Google Form; JavaScript chịu trách nhiệm xử lý logic trên trình duyệt như kiểm tra dữ liệu đầu vào, hiển thị thông báo lỗi, tổng hợp thông tin và phản hồi cho người dùng.

Kết quả của dự án là một hệ thống form xét học bổng trực tuyến có giao diện trực quan, dễ sử dụng, hỗ trợ nhập liệu nhiều trường thông tin khác nhau, đồng thời thực hiện kiểm tra tính hợp lệ của dữ liệu ngay trên trình duyệt. Sản phẩm có thể được sử dụng như một công cụ hỗ trợ cho các đơn vị đào tạo trong việc thu thập dữ liệu đăng ký học bổng, hoặc làm nền tảng tham khảo cho các bài toán thu thập thông tin trực tuyến tương tự.

MỞ ĐẦU

Lý do chọn đề tài

Trong quá trình quản lý và xét học bổng cho sinh viên, việc thu thập thông tin thường được thực hiện thông qua các mẫu đơn giấy hoặc các biểu mẫu trực tuyến. Cách làm thủ công dễ dẫn đến sai sót, tốn thời gian xử lý và khó lưu trữ lâu dài. Trong khi đó, các nền tảng biểu mẫu trực tuyến như Google Form cho thấy hiệu quả rõ rệt trong việc thu thập và xử lý dữ liệu.

Xuất phát từ nhu cầu đó, nhóm quyết định thực hiện đề tài *Xây dựng form xét học bổng trực tuyến* với mục tiêu mang đến một biểu mẫu đơn giản và dễ sử dụng cho người, đồng thời giúp củng cố kiến thức cốt lõi khi xây dựng một trang web như HTML, CSS, JavaScript.

Mục tiêu và phạm vi

Mục tiêu chính của đề tài:

- Thiết kế và xây dựng một form xét học bổng trực tuyến với giao diện thân thiện, dễ sử dụng.
- Áp dụng các kiến thức về HTML, CSS và JavaScript để xử lý dữ liệu phía client, bao gồm kiểm tra tính hợp lệ và phản hồi cho người dùng.
- Tổ chức lại mã nguồn thành một dự án web rõ ràng, có thể dễ dàng mở rộng hoặc tái sử dụng.

Phạm vi của đề tài tập trung vào:

- Xử lý dữ liệu trên phía trình duyệt (client-side), chưa triển khai lưu trữ dữ liệu trên server thực tế.
- Mô phỏng quy trình đăng ký xét học bổng thông qua một biểu mẫu duy nhất.

Công nghệ sử dụng

Dự án sử dụng các công nghệ chính sau:

- **HTML**: Xây dựng cấu trúc form và các trường nhập liệu.
- **CSS**: Thiết kế giao diện, bố cục, màu sắc, định dạng form theo phong cách tương tự Google Form.
- **JavaScript**: Xử lý tương tác người dùng, kiểm tra dữ liệu đầu vào và hiển thị thông báo.

Thành viên thực hiện và phân công công việc

Nhóm thực hiện đề tài gồm 5 thành viên:

- Lê Quang Huy 1334770
- Văn Đình Thành Trung 1335370
- Hà Trung Hiếu 1212070
- Lê Đức Trường 1234270
- Nguyễn Quang Khánh 1007370

Phân công công việc (mang tính tương đối, có hỗ trợ lẫn nhau):

- Tìm hiểu yêu cầu bài toán, đề xuất các trường thông tin cần thu thập trong form xét học bổng.
- Thiết kế giao diện và bố cục form, lựa chọn màu sắc và phong cách hiển thị.
- Xây dựng cấu trúc HTML cho biểu mẫu.
- Cài đặt CSS để hoàn thiện phần giao diện.
- Lập trình JavaScript để kiểm tra dữ liệu và xử lý tương tác với người dùng.
- Viết báo cáo và chuẩn bị nội dung thuyết trình.

Với bố cục như trên, các chương tiếp theo của báo cáo sẽ trình bày lần lượt cơ sở lý thuyết liên quan, quá trình xây dựng và thực nghiệm dự án, cùng với các kết quả đạt được.

Chương 1

Cơ sở lý thuyết

1.1 Giới thiệu chung về lập trình web phía client

Lập trình web phía client (client-side) là quá trình xây dựng các thành phần giao diện và xử lý tương tác trực tiếp trên trình duyệt của người dùng. Ba công nghệ nền tảng đóng vai trò quan trọng trong mảng này là HTML, CSS và JavaScript. HTML chịu trách nhiệm định nghĩa cấu trúc nội dung, CSS dùng để định dạng và trình bày, trong khi JavaScript được sử dụng để xử lý logic, sự kiện và tương tác động.

Trong phạm vi đề tài *xây dựng form xét học bổng trực tuyến*, ba công nghệ này được kết hợp để tạo nên một biểu mẫu web có giao diện thân thiện, dễ sử dụng và có khả năng kiểm tra, xử lý dữ liệu trực tiếp trên trình duyệt mà không cần đến backend phức tạp.

1.2 Ngôn ngữ đánh dấu siêu văn bản HTML

1.2.1 Khái niệm và vai trò của HTML

HTML (HyperText Markup Language) là ngôn ngữ đánh dấu chuẩn dùng để xây dựng cấu trúc của trang web. Một tài liệu HTML được tổ chức thông qua các thẻ (tag) lồng nhau, mô tả các thành phần như tiêu đề, đoạn văn, hình ảnh, liên kết và đặc biệt là các biểu mẫu (form) dùng để nhập liệu.

Đối với các hệ thống thu thập dữ liệu trực tuyến, HTML cung cấp tập hợp phong phú các thẻ và thuộc tính cho phép thiết kế các form nhập liệu phù hợp với nhiều loại thông tin khác nhau: văn bản, số, ngày tháng, lựa chọn một hoặc nhiều giá trị, v.v.

1.2.2 Cấu trúc cơ bản của một form HTML

Biểu mẫu HTML được định nghĩa bởi thẻ `<form>` và chứa các phần tử con như `<input>`, `<select>`, `<textarea>`, `<button>`, ... Một form cơ bản thường bao gồm các thuộc tính quan trọng:

- `action`: địa chỉ URL sẽ nhận dữ liệu khi form được gửi đi.
- `method`: phương thức gửi dữ liệu, thường là `GET` hoặc `POST`.
- `name`, `id`: dùng để định danh form và phục vụ cho việc thao tác bằng JavaScript.
- `autocomplete`: cho phép hoặc vô hiệu hóa việc trình duyệt tự động gợi ý dữ liệu.

Trong đề tài này, do quá trình xử lý được thực hiện chủ yếu ở phía client, thuộc tính `action` có thể để trống hoặc dùng dấu `#` và việc xử lý logic được đảm nhiệm bởi JavaScript thông qua sự kiện `submit`.

1.2.3 Các trường nhập liệu thường dùng trong form xét học bổng

Để xây dựng form xét học bổng, có thể cần thu thập nhiều loại thông tin khác nhau: thông tin cá nhân, kết quả học tập, hoàn cảnh gia đình, thành tích ngoại khoá, v.v. Một số loại trường nhập liệu HTML quan trọng gồm:

- **Trường văn bản** (`<input type="text">`): dùng cho họ tên, lớp, ngành, trường, địa chỉ, ...
- **Trường email** (`<input type="email">`): dùng cho địa chỉ thư điện tử của sinh viên.
- **Trường số** (`<input type="number">`): dùng cho điểm trung bình, số tín chỉ, thu nhập bình quân, ...
- **Trường ngày tháng** (`<input type="date">`): dùng cho ngày sinh hoặc các mốc thời gian quan trọng.
- **Nút chọn một** (`<input type="radio">`): dùng cho các lựa chọn chỉ được phép chọn một, ví dụ: giới tính, hệ đào tạo.
- **Ô chọn nhiều** (`<input type="checkbox">`): dùng cho các lựa chọn có thể chọn đồng thời, ví dụ: các loại thành tích, các cam kết.
- **Danh sách lựa chọn** (`<select>`, `<option>`): dùng cho các trường như ngành học, khóa, loại học bổng.

- **Vùng nhập nhiều dòng** (`<textarea>`): dùng cho phần trình bày hoàn cảnh, lý do xin học bổng, kế hoạch học tập, ...

Việc lựa chọn đúng loại trường nhập liệu giúp người dùng thao tác thuận tiện hơn, đồng thời hỗ trợ trình duyệt và JavaScript trong việc kiểm tra tính hợp lệ của dữ liệu.

1.2.4 Nhóm thông tin và khả năng truy cập

Để form trở nên rõ ràng và dễ sử dụng, các trường liên quan thường được nhóm lại bằng các thẻ như `<fieldset>` và `<legend>`. Các nhóm phổ biến trong form xét học bổng có thể là:

- Thông tin cá nhân.
- Thông tin học tập.
- Thông tin về hoàn cảnh gia đình.
- Thành tích, hoạt động ngoại khoá.

Bên cạnh đó, việc sử dụng thẻ `<label>` gắn với mỗi `<input>` bằng thuộc tính `for` (trùng với `id` của trường nhập liệu) giúp tăng khả năng truy cập và cải thiện trải nghiệm người dùng, đặc biệt đối với những người sử dụng công cụ hỗ trợ như screen reader.

1.2.5 Các thuộc tính hỗ trợ kiểm tra dữ liệu ngay trong HTML

HTML5 cung cấp nhiều thuộc tính giúp kiểm tra dữ liệu (validation) ngay trên trình duyệt mà không cần viết thêm JavaScript, chẳng hạn:

- `required`: bắt buộc trường phải được nhập.
- `min`, `max`: giới hạn giá trị nhỏ nhất và lớn nhất cho các trường số.
- `maxlength`: giới hạn độ dài tối đa của chuỗi ký tự.
- `pattern`: dùng biểu thức chính quy để ràng buộc định dạng dữ liệu (ví dụ định dạng mã sinh viên).
- `step`: bước nhảy cho các trường số hoặc ngày.

Mặc dù khả năng kiểm tra dữ liệu sẵn có trong HTML rất hữu ích, tuy nhiên trong nhiều trường hợp, JavaScript vẫn cần thiết để thực hiện các kiểm tra phức tạp hơn (ví dụ: kết hợp nhiều trường, tính toán điểm trung bình, đối chiếu giữa các giá trị, ...).

1.3 Ngôn ngữ định kiểu CSS

1.3.1 Khái niệm và vai trò của CSS

CSS (Cascading Style Sheets) là ngôn ngữ dùng để mô tả cách trình bày của tài liệu HTML. Thông qua CSS, lập trình viên có thể điều chỉnh màu sắc, font chữ, khoảng cách, bố cục, hiệu ứng tương tác, ... mà không cần thay đổi cấu trúc HTML.

Đối với form xét học bổng trực tuyến, CSS đóng vai trò then chốt trong việc mang lại trải nghiệm người dùng trực quan, thống nhất và hiện đại. CSS cho phép tổ chức bố cục form hợp lý, nhấn mạnh các tiêu đề, làm nổi bật các trường nhập liệu quan trọng, đồng thời cung cấp tín hiệu trực quan khi có lỗi dữ liệu.

1.3.2 Liên kết CSS với tài liệu HTML

Có ba cách phổ biến để áp dụng CSS cho tài liệu HTML:

- **Inline style**: sử dụng thuộc tính `style` trực tiếp trên từng phần tử HTML.
- **Internal stylesheet**: đặt mã CSS trong thẻ `<style>` bên trong phần `<head>` của tài liệu.
- **External stylesheet**: lưu CSS trong một file riêng (ví dụ: `style.css`) và liên kết bằng thẻ `<link>`.

Trong các dự án có quy mô lớn hơn, việc sử dụng file CSS ngoài được khuyến khích vì giúp mã nguồn dễ quản lý, dễ bảo trì và tái sử dụng. Dự án form xét học bổng cũng áp dụng cách tiếp cận này.

1.3.3 Mô hình hộp và bố cục form

Mỗi phần tử HTML trên trang web có thể được xem như một *hộp* (box) với các thành phần: nội dung (content), phần đệm (padding), đường viền (border) và lề (margin). Hiểu rõ mô hình hộp (box model) là điều kiện tiên quyết để thiết kế bố cục form một cách chính xác.

Các thuộc tính CSS thường sử dụng khi dàn trang form bao gồm:

- **Kích thước**: `width`, `height`, `max-width`, ...
- **Lề và đệm**: `margin`, `padding`.
- **Đường viền**: `border`, `border-radius`.
- **Hiển thị**: `display`, `flex`, `inline-block`, ...

- **Căn chỉnh**: `text-align`, `justify-content`, `align-items`.

Bằng cách kết hợp các thuộc tính trên, form xét học bổng có thể được căn giữa trên trang, các nhóm câu hỏi được sắp xếp gọn gàng, tạo cảm giác dễ theo dõi cho người dùng.

1.3.4 Thiết kế giao diện theo phong cách Google Form

Để thiết kế giao diện một cách trực quan, có thể áp dụng một số nguyên tắc thiết kế sau:

- Sử dụng tông màu chủ đạo nhẹ nhàng (thường là các tông màu trung tính), kết hợp với nền trắng cho các khối câu hỏi.
- Chọn font chữ gọn gàng, đơn giản và dễ đọc.
- Tạo khoảng trắng (white space) hợp lý giữa các nhóm câu hỏi, giữa tiêu đề và phần nội dung để form không bị rối mắt.
- Làm nổi bật tiêu đề form, mô tả form, và các heading của từng nhóm câu hỏi.

CSS cũng được dùng để tạo các hiệu ứng tương tác như thay đổi màu viền của trường nhập liệu khi người dùng focus, đổi màu nút gửi khi hover, qua đó tăng tính thân thiện và hiện đại cho giao diện.

1.3.5 Biểu diễn trạng thái lỗi và thông báo cho người dùng

Trong các form nhập liệu, việc hiển thị trạng thái lỗi một cách rõ ràng là rất quan trọng. Một số kỹ thuật thường dùng trong CSS bao gồm:

- Gán các lớp (class) như `.error`, `.valid` cho trường nhập liệu tuỳ theo kết quả kiểm tra dữ liệu.
- Đổi màu viền, nền hoặc chữ của các trường có lỗi (ví dụ: viền đỏ đối với trường chưa nhập hoặc nhập sai định dạng).
- Hiển thị thông báo lỗi ngay bên dưới hoặc bên cạnh trường nhập liệu, với kích thước chữ nhỏ hơn và màu sắc nổi bật (thường là đỏ).

Các lớp CSS này thường được kết hợp với JavaScript: khi JavaScript phát hiện lỗi, nó sẽ thêm lớp `error` vào phần tử tương ứng, và CSS sẽ đảm nhiệm phần hiển thị trực quan cho người dùng.

1.4 Ngôn ngữ lập trình JavaScript

1.4.1 Tổng quan về JavaScript phía client

JavaScript là ngôn ngữ lập trình được hỗ trợ bởi hầu hết các trình duyệt hiện đại, cho phép trang web trở nên tương tác, linh hoạt và có khả năng xử lý logic phức tạp ngay trên máy người dùng. Trong bối cảnh form xét học bổng, JavaScript được sử dụng để:

- Lắng nghe và xử lý các sự kiện do người dùng tạo ra (nhập dữ liệu, bấm nút gửi, ...).
- Kiểm tra tính hợp lệ của dữ liệu trước khi chấp nhận.
- Hiển thị hoặc ẩn các thông báo lỗi, hướng dẫn.
- Tóm tắt và phản hồi lại thông tin cho người dùng (ví dụ: thông báo gửi form thành công).

JavaScript được nhúng vào trang thông qua thẻ `<script>` hoặc file `.js` bên ngoài, thường được đặt ở cuối thân trang để tránh chặn quá trình tải giao diện.

1.4.2 Truy cập và thao tác với DOM

DOM (Document Object Model) là mô hình đối tượng biểu diễn cấu trúc tài liệu HTML. JavaScript thao tác với DOM để đọc và thay đổi nội dung, thuộc tính và kiểu dáng của các phần tử trên trang.

Một số thao tác thường gặp trong dự án form xét học bổng:

- Tìm phần tử theo `id` hoặc selector:
  - `document.getElementById('...')`
  - `document.querySelector('...')`
  - `document.querySelectorAll('...')`
- Lấy và gán giá trị của trường nhập liệu: `element.value`.
- Thay đổi lớp CSS: `element.classList.add()`, `element.classList.remove()`.
- Cập nhật nội dung thông báo: `element.textContent` hoặc `element.innerText`.

Nhờ các thao tác này, JavaScript có thể kiểm soát giao diện form một cách linh hoạt, ví dụ: làm nổi bật trường có lỗi, hiển thị thông báo bên dưới trường chưa được nhập, hoặc ẩn hiện các phần của form tuỳ theo lựa chọn của người dùng.

1.4.3 Xử lý sự kiện trong form

Sự kiện (event) là các hành động xảy ra trên trang web, ví dụ: người dùng nhấn nút, thay đổi giá trị một trường, di chuyển chuột, ... Đối với form, các sự kiện quan trọng bao gồm:

- **Sự kiện** `submit` của thẻ `<form>`: được kích hoạt khi người dùng gửi form.
- **Sự kiện** `click` trên nút gửi.
- **Sự kiện** `input` **hoặc** `change` trên các trường nhập liệu: kích hoạt khi người dùng nhập hoặc thay đổi giá trị.

Thông qua hàm xử lý sự kiện, JavaScript có thể:

- Gọi `event.preventDefault()` để ngăn trình duyệt gửi form khi dữ liệu chưa hợp lệ.
- Thực hiện kiểm tra dữ liệu cho từng trường hoặc toàn bộ form.
- Cập nhật giao diện (thêm lớp lỗi, hiện thông báo, ...).

Cách tiếp cận này cho phép việc kiểm tra dữ liệu diễn ra ngay trên trình duyệt, mang lại phản hồi nhanh cho người dùng và giảm thiểu các yêu cầu gửi dữ liệu không hợp lệ.

1.4.4 Kiểm tra và xử lý dữ liệu người dùng

Một trong những nhiệm vụ quan trọng của JavaScript trong dự án là kiểm tra tính hợp lệ của dữ liệu (validation). Các bước kiểm tra điển hình có thể bao gồm:

- Kiểm tra các trường bắt buộc đã được nhập hay chưa.
- Kiểm tra định dạng email.
- Kiểm tra khoảng giá trị cho các trường số (ví dụ: điểm trung bình nằm trong khoảng cho phép).
- Kiểm tra độ dài tối đa của một số trường văn bản.
- Kết hợp nhiều trường để đưa ra cảnh báo phù hợp (ví dụ: nếu điểm thấp hơn một ngưỡng nhất định thì hiển thị thông báo không đủ điều kiện).

Thông thường, các hàm kiểm tra được tổ chức theo hướng chia nhỏ logic: mỗi hàm phụ trách một nhóm điều kiện, sau đó được gọi trong hàm xử lý chung khi form được gửi. Kết quả kiểm tra (hợp lệ hoặc không) được phản ánh lại trên giao diện thông qua việc thêm hoặc xoá lớp CSS, cũng như cập nhật nội dung thông báo lỗi.

1.4.5 Tóm tắt và phản hồi cho người dùng

Sau khi dữ liệu vượt qua các bước kiểm tra, JavaScript có thể thực hiện một số hành động như:

- Hiển thị thông báo gửi form thành công.
- Tóm tắt lại một số thông tin quan trọng mà người dùng đã nhập (ví dụ: họ tên, lớp, loại học bổng đăng ký).
- Làm sạch form (reset) nếu muốn cho phép nhập lại từ đầu.

Trong trường hợp đề tài chỉ xử lý dữ liệu phía client, thông tin có thể được in ra màn hình hoặc hiển thị trong một khu vực riêng trên trang. Nếu mở rộng trong tương lai, dữ liệu này có thể được gửi tới server để lưu trữ lâu dài hoặc xử lý tiếp.

1.5 Tổng kết chương

Trong chương này, báo cáo đã trình bày các cơ sở lý thuyết chính phục vụ cho việc xây dựng form xét học bổng trực tuyến, bao gồm: cấu trúc và vai trò của HTML trong việc định nghĩa form và các trường nhập liệu; vai trò của CSS trong việc tạo bố cục, giao diện và hiển thị trạng thái lỗi; cùng với cách dùng JavaScript để thao tác với DOM, xử lý sự kiện và kiểm tra dữ liệu người dùng.

Các kiến thức này là nền tảng để triển khai dự án ở chương tiếp theo, nơi form xét học bổng sẽ được thiết kế, cài đặt và kiểm thử dựa trên những nguyên tắc đã được trình bày.

Chương 2

Thực nghiệm và xây dựng hệ thống

2.1 Giới thiệu chương

Chương này trình bày chi tiết quá trình xây dựng form đăng ký xét học bổng trực tuyến cho Trường Đại học Xây Dựng Hà Nội. Hệ thống được phát triển dựa trên ba công nghệ web cơ bản: HTML xây dựng khung sườn trang web, CSS trang trí giao diện đẹp mắt, và JavaScript xử lý các tương tác với người dùng. Form được thiết kế với giao diện hiện đại và thân thiện.

2.2 Xây dựng cấu trúc form bằng HTML

2.2.1 Khung trang cơ bản

Trang web bắt đầu với phần khai báo và thiết lập ban đầu. Đây là nền tảng để trình duyệt hiểu cách hiển thị trang đúng cách.

```
<!DOCTYPE html>
<html lang="vi">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
       scale=1.0">
    <title>Đơn Xin Học Bong - Trường ĐH Xây Dựng Hà Nội</title>
    <link rel="stylesheet" href="style.css">
</head>
```

Listing 2.1: Phần đầu trang HTML

Dòng đầu tiên khai báo đây là tài liệu HTML5 phiên bản mới nhất. Thuộc tính `lang="vi"` cho biết trang viết bằng tiếng Việt, giúp công cụ tìm kiếm và trình đọc

màn hình hoạt động tốt hơn. Thẻ `meta charset` đảm bảo các chữ cái có dấu hiển thị đúng. Thẻ `viewport` đặc biệt quan trọng - nó bảo điện thoại hiển thị trang với kích thước vừa màn hình, không phóng to thu nhỏ tự động.

```
<body>
    <div class="container">
        <div class="form-header">
            <img src="pexels-pixabay-267885.jpg" alt="Banner"
                 class="header-image">
            <div class="header-content">
                <h1>Đơn Xin Học Bong</h1>
                <p class="form-description">
                    Vui lòng điền đầy đủ và chính xác các thông
                        tin
                </p>
            </div>
        </div>

        <form id="scholarshipForm" class="scholarship-form">
            <!-- Dữ liệu form -->
        </form>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

Listing 2.2: Phần thân trang và header

Phần thân trang được bọc trong khối `container` để dễ căn giữa. Phần đầu form có ảnh banner đại diện cho trường và tiêu đề chính. Form với mã số `scholarshipForm` sẽ chứa tất cả các ô để người dùng điền thông tin. File JavaScript được gọi ở cuối để đảm bảo trang đã tải xong mới chạy các đoạn mã xử lý.

2.2.2 Thu thập thông tin cá nhân

Phần đầu tiên của form thu thập các thông tin cơ bản nhất về sinh viên.

```
<section class="form-section">
    <h2 class="section-title">Thông tin cá nhân</h2>

    <div class="form-question">
        <label class="question-label">
            Họ và tên <span class="required">*</span>
```

```
        </label>
        <input type="text" name="fullName" class="form-input"
               placeholder="Nhập họ và tên đầy đủ" required>
        <span class="error-msg"></span>
    </div>

    <div class="form-question">
        <label class="question-label">Số CCCD *</label>
        <input type="text" name="idCard" class="form-input"
               placeholder="12 chữ số" maxlength="12" required>
        <span class="error-msg"></span>
    </div>
</section>
```

Listing 2.3: Ô nhập họ tên và CCCD

Mỗi câu hỏi được bọc trong khung `form-question` gồm ba thành phần: nhãn hỏi, ô nhập liệu, và dòng trống để hiện thông báo lỗi. Dấu sao màu đỏ đánh dấu trường bắt buộc phải điền. Với số CCCD, thuộc tính `maxlength` giới hạn chỉ được gõ tối đa 12 ký tự đúng quy định của Việt Nam.

```
<div class="form-question">
    <label class="question-label">Ngày sinh *</label>
    <input type="date" name="birthDate" class="form-input"
       required>
    <span class="error-msg"></span>
</div>

<div class="form-question">
    <label class="question-label">Giới tính *</label>
    <div class="radio-group">
        <label class="radio-option">
            <input type="radio" name="gender" value="male"
               required>
            <span class="radio-label">Nam</span>
        </label>
        <label class="radio-option">
            <input type="radio" name="gender" value="female">
            <span class="radio-label">Nữ</span>
        </label>
    </div>
    <span class="error-msg"></span>
```

```
</div>
```

Listing 2.4: Chọn ngày sinh và giới tính

Ô chọn ngày hiển thị lịch để người dùng click chọn thay vì gõ tay, tránh nhầm lẫn định dạng. Phần giới tính dùng nút tròn (radio button) - loại nút chỉ chọn được một trong hai. Hai nút có cùng tên `gender` nên trình duyệt tự hiểu chúng là một nhóm và chỉ cho chọn một.

2.2.3 Các ô nhập đặc biệt

HTML5 cung cấp nhiều loại ô nhập thông minh, tự động kiểm tra và hiện bàn phím phù hợp trên điện thoại.

```
<div class="form-question">
    <label class="question-label">Email *</label>
    <input type="email" name="email" class="form-input"
           pattern="[a-z0-9._%+-]+@st\.huce\.edu\.vn$"
           placeholder="example@st.huce.edu.vn" required>
    <span class="error-msg"></span>
</div>

<div class="form-question">
    <label class="question-label">Số điện thoại *</label>
    <input type="tel" name="phone" class="form-input"
           placeholder="0912345678" required>
    <span class="error-msg"></span>
</div>
```

Listing 2.5: Ô nhập email và số điện thoại

Ô email tự kiểm tra xem có dấu @ và tên miền hay không. Khi mở trên điện thoại, bàn phím tự động hiện phím @ và dấu chấm để gõ dễ hơn. Thuộc tính `pattern` kiểm tra chặt hơn - chỉ chấp nhận email có đuôi @st.huce.edu.vn của sinh viên trường. Ô số điện thoại khi mở trên mobile sẽ hiện bàn phím số thay vì chữ cái.

```
<div class="form-question">
    <label class="question-label">GPA tích lũy *</label>
    <input type="number" name="gpa" class="form-input"
           step="0.01" min="0" max="4"
           placeholder="VD: 3.50" required>
    <span class="error-msg"></span>
</div>

```

```
<div class="form-question">
    <label class="question-label">Khoa *</label>
    <select name="faculty" class="form-select" required>
        <option value="">-- Chọn khoa --</option>
        <option value="cntt">Công nghệ Thông tin</option>
        <option value="ktqlxd">Kinh tế & Quãn lý Xây dựng</
           option>
        <option value="ck">Cơ khí</option>
    </select>
    <span class="error-msg"></span>
</div>
```

Listing 2.6: Ô nhập điểm và danh sách chọn khoa

Ô nhập điểm GPA có hai nút mũi tên nhỏ để tăng giảm số. Thuộc tính `step="0.01"` cho phép nhập số thập phân như 3.50 hoặc 3.75. Thuộc tính `min` và `max` giới hạn điểm trong khoảng 0 đến 4 theo thang điểm Việt Nam. Danh sách thả xuống cho người dùng chọn khoa, dòng đầu tiên là gợi ý chưa chọn gì - vì ô này bắt buộc nên phải chọn một khoa cụ thể.

2.2.4 Hướng dẫn và tải file

Một số thông tin cần giải thích thêm để người dùng tự đánh giá đúng.

```
<div class="form-question">
    <label class="question-label">Xếp loại học lực *</label>

    <div class="helper-text">
        <p><strong>Hướng dan xếp loại theo GPA:</strong></p>
        <ul>
            <li><strong>Xuất sắc:</strong> GPA từ 3.6 đến 4.0</
               li>
            <li><strong>Giỏi:</strong> GPA từ 3.2 đến dưới 3.6<
               /li>
            <li><strong>Khá:</strong> GPA từ 2.5 đến dưới 3.2</
               li>
        </ul>
    </div>

    <select name="academicRank" class="form-select" required>
        <option value="">-- Chọn xếp loại --</option>
        <option value="excellent">Xuất sắc</option>
```

```
        <option value="good">Giỏi</option>
        <option value="fair">Khá</option>
    </select>
</div>
```

Listing 2.7: Hộp hướng dẫn xếp loại

Hộp màu xám nhạt hiện tiêu chí từng mức xếp loại giúp sinh viên tự so sánh điểm của mình và chọn đúng. Ký tự "dưới" được viết bằng mã `<` vì dấu < viết trực tiếp sẽ bị nhầm với thẻ HTML.

```
<div class="form-question">
    <label class="question-label">Ảnh minh chứng</label>
    <input type="file" name="studyProof" class="form-file"
           accept="image/*" multiple>
    <div class="file-hint">JPG, PNG (tối đa 5MB)</div>
</div>

<div class="form-question">
    <label class="checkbox-container">
        <input type="checkbox" name="agreement" required>
        <span class="checkbox-label">
            Tôi cam đoan thông tin chính xác *
        </span>
    </label>
    <span class="error-msg"></span>
</div>
```

Listing 2.8: Tải ảnh và checkbox đồng ý

Nút chọn file khi click sẽ mở hộp thoại để chọn ảnh từ máy tính hoặc điện thoại. Thuộc tính `accept="image/*"` lọc chỉ hiện các file ảnh, dấu sao nghĩa là tất cả loại ảnh (JPG, PNG, GIF...). Thuộc tính `multiple` cho phép chọn nhiều ảnh cùng lúc. Ô vuông (checkbox) để người dùng đánh dấu đồng ý - vì có thuộc tính bắt buộc nên phải tích vào mới gửi được form.

2.2.5 Hộp thông báo thành công

Sau khi gửi form, một hộp sẽ hiện lên giữa màn hình để thông báo.

```
<div id="successModal" class="modal">
    <div class="modal-content">
        <div class="modal-icon">   </div>
        <h2>Đã gửi đơn thành công!</h2>
```

```
        <p>Đơn xin học bong đã được ghi nhận.</p>
        <p class="modal-detail">
            Chúng tôi sẽ phãn hồi trong 7-10 ngày làm việc.
        </p>
        <button class="btn-close-modal" onclick="closeModal()">
            Đóng
        </button>
    </div>
</div>
```

Listing 2.9: Cấu trúc hộp thông báo

Hộp thông báo hiển thị nổi bật trên lớp nền mờ phủ toàn trang. Ban đầu bị ẩn, hệ thống sẽ kích hoạt hiển thị qua JavaScript khi cần thiết. Bên trong chứa dấu tích xanh báo hiệu nộp đơn thành công, kèm nút Đóng để người dùng tắt thông báo.

2.3 Thiết kế giao diện bằng CSS

2.3.1 Màu sắc và thiết lập chung

CSS quyết định màu sắc, kích thước, vị trí của mọi thứ trên trang. Ta dùng biến để dễ quản lý.

```
:root {
    --primary: #673ab7;
    --primary-dark: #5e35b1;
    --text: #202124;
    --text-light: #5f6368;
    --border: #dadce0;
    --error: #d93025;
    --success: #1e8e3e;
    --radius: 8px;
    --shadow: 0 1px 2px 0 rgba(60,64,67,.3);
}
```

Listing 2.10: Khai báo màu sắc chung

Đây là cách đặt tên cho các màu hay dùng. Màu chủ đạo là tím `-primary`, màu chữ là xám đen `-text`, màu lỗi là đỏ `-error`. Khi cần dùng chỉ cần gọi tên, ví dụ `var(-primary)`. Lợi ích lớn nhất: muốn đổi màu toàn bộ trang chỉ sửa ở đây thay vì tìm sửa hàng trăm chỗ.

```
* {
```

```
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Roboto', 'Segoe UI', Arial, sans-serif;
    background: #f0ebf8;
    color: var(--text);
    line-height: 1.6;
}
```

Listing 2.11: Thiết lập ban đầu

Dấu sao (*) chọn tất cả phần tử, đặt lại khoảng cách trong và ngoài về 0 để loại bỏ sự khác biệt giữa các trình duyệt. Thuộc tính `box-sizing` thay đổi cách tính kích thước - giờ chiều rộng đã bao gồm viền và đệm, giúp tính toán dễ hơn. Font chữ được ưu tiên: dùng Roboto nếu có, không có thì dùng Segoe UI, cuối cùng là Arial. Khoảng cách dòng 1.6 lần giúp chữ thoáng hơn, dễ đọc.

2.3.2 Căn giữa và phần đầu

Container căn giữa trang, header có ảnh và viền màu đặc trưng.

```
.container {
    max-width: 760px;
    margin: 0 auto;
}

.form-header {
    background: white;
    border-radius: 8px 8px 0 0;
    overflow: hidden;
    box-shadow: var(--shadow);
}

.header-image {
    width: 100%;
    height: 200px;
    object-fit: cover;
}

```

```
.header-content {
    padding: 24px;
    border-top: 10px solid var(--primary);
}
```

Listing 2.12: Khung chứa và header

Khung chứa rộng tối đa 760 pixel, trên màn nhỏ hơn sẽ tự thu lại. Kết hợp với lề tự động (`margin: 0 auto`) nó sẽ tự căn giữa. Header bo tròn hai góc trên (8 pixel), hai góc dưới để vuông. Thuộc tính `overflow: hidden` cắt ảnh theo bo góc của khung. Ảnh chiều rộng 100% để luôn vừa khung, chiều cao cố định 200 pixel. Thuộc tính `object-fit: cover` giúp ảnh tự cắt để lấp đầy mà không méo. Viền trên dày 10 pixel màu tím là đặc trưng thiết kế Google Forms.

2.3.3 Tạo kiểu cho ô nhập

Các ô nhập cần có nhiều trạng thái: bình thường, di chuột vào, đang gõ, và có lỗi.

```
.form-input, .form-select, .form-textarea {
    width: 100%;
    padding: 12px 14px;
    border: 1px solid var(--border);
    border-radius: 4px;
    font-size: 0.875rem;
    transition: all 0.2s ease;
}
```

Listing 2.13: Ô nhập trạng thái bình thường

Tất cả ô nhập rộng 100% để tự động vừa khung chứa, giúp trang co giãn tốt. Khoảng đệm bên trong 12 pixel trên/dưới và 14 pixel trái/phải tạo không gian thoải mái. Viền mỏng 1 pixel màu xám nhạt, bo góc 4 pixel cho mềm mại. Cỡ chữ dùng đơn vị rem - ưu điểm là tự điều chỉnh theo cài đặt của người dùng. Thuộc tính chuyển đổi (transition) tạo hiệu ứng mượt trong 0.2 giây khi thay đổi màu.

```
.form-input:hover {
    border-color: var(--text);
}

.form-input:focus {
    outline: none;
    border-color: var(--primary);
    box-shadow: 0 0 0 1px var(--primary);
}
```

```
10
11 .form-input.error {
12     border-color: var(--error);
13 }
14
15 .error-msg {
16     display: block;
17     color: var(--error);
18     font-size: 0.75rem;
19     margin-top: 4px;
20 }
```

Listing 2.14: Các trạng thái của ô nhập

Khi di chuột vào (hover), viền đổi sang màu xám đậm hơn để người dùng biết đây là vùng có thể nhấn. Khi click vào để gõ (focus), viền chuyển màu tím và có thêm vòng sáng bao quanh - rất quan trọng cho người dùng bàn phím biết đang gõ ở đâu. Đường viền xanh mặc định của trình duyệt bị xóa nhưng được thay bằng viền tím đẹp hơn. Đổ bóng với giá trị đặc biệt tạo vòng viền thứ hai mà không làm thay đổi kích thước ô. Khi có lỗi, JavaScript sẽ thêm class "error" làm viền chuyển đỏ. Thông báo lỗi hiện bên dưới với màu đỏ, cỡ chữ nhỏ hơn.

2.3.4 Tùy chỉnh danh sách thả

Danh sách thả xuống mặc định xấu và khác nhau giữa các trình duyệt. Ta tự vẽ lại cho đẹp.

```
1 .form-select {
2     appearance: none;
3     background-image: url("data:image/svg+xml,%3Csvg xmlns='
          http://www.w3.org/2000/svg' width='12' height='12'%3E%3
          Cpath fill='%235f6368' d='M6 9L1 4h10z'/%3E%3C/svg%3E");
4     background-repeat: no-repeat;
5     background-position: right 12px center;
6     padding-right: 36px;
7     cursor: pointer;
8 }
```

Listing 2.15: Vẽ lại danh sách thả

Thuộc tính đầu tiên xóa bỏ giao diện mặc định của trình duyệt. Sau đó ta vẽ mũi tên tam giác nhỏ bằng hình SVG nhúng trực tiếp vào CSS, tránh phải tải file riêng làm chậm trang. Mũi tên được đặt bên phải cách lề 12 pixel, căn giữa theo chiều dọc.

Khoảng đệm bên phải 36 pixel tạo chỗ trống cho mũi tên, tránh chữ chạy ra che mất. Con trỏ chuột đổi thành bàn tay khi di vào để biết đây là nơi có thể click.

2.3.5 Hộp thông báo trượt lên

Hộp thông báo phủ toàn màn hình, căn giữa và có hiệu ứng trượt từ dưới lên.

```
.modal {
    display: none;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.5);
    z-index: 1000;
    align-items: center;
    justify-content: center;
}

.modal.show {
    display: flex;
}
```

Listing 2.16: Màn che và căn giữa

Vị trí cố định (fixed) giữ hộp thoại ở một chỗ trên màn hình, không cuộn theo. Kết hợp vị trí 0 từ trên và trái, rộng và cao 100% tạo lớp phủ toàn màn hình. Nền đen trong suốt 50% làm mờ nội dung phía sau, giúp người dùng tập trung vào thông báo. Số z-index lớn đảm bảo luôn hiện trên cùng. Ban đầu hộp bị ẩn hoàn toàn, JavaScript thêm class "show" để hiện lên và kích hoạt flexbox căn giữa.

```
.modal-content {
    background: white;
    border-radius: 8px;
    padding: 32px;
    max-width: 500px;
    animation: slideUp 0.3s ease;
}

@keyframes slideUp {
    from {
        opacity: 0;
```

```
12          transform: translateY(30px);
13      }
14      to {
15          opacity: 1;
16          transform: translateY(0);
17      }
18  }
```

Listing 2.17: Nội dung và hiệu ứng

Phần nội dung có nền trắng, bo góc 8 pixel. Rộng tối đa 500 pixel nhưng trên màn nhỏ sẽ thu lại. Phần định nghĩa chuyển động (keyframes) mô tả hiệu ứng: bắt đầu từ trong suốt hoàn toàn và ở dưới 30 pixel so với vị trí cuối, kết thúc ở hiện rõ hoàn toàn và đúng vị trí. Điều này tạo hiệu ứng mờ dần và trượt lên mượt mà trong 0.3 giây.

2.3.6 Điều chỉnh cho điện thoại

Trên màn hình nhỏ cần thay đổi một số thứ để dễ dùng hơn.

```
1  @media (max-width: 600px) {
2      body {
3          padding: 12px 8px;
4      }
5
6      .header-content h1 {
7          font-size: 1.5rem;
8      }
9
10     .form-actions {
11         flex-direction: column;
12     }
13
14     .btn-submit, .btn-clear {
15         width: 100%;
16     }
17 }
```

Listing 2.18: Quy tắc cho màn hình nhỏ

Dòng đầu nghĩa là "áp dụng các quy tắc bên trong khi màn hình rộng tối đa 600 pixel" - kích thước điển hình của điện thoại. Giảm khoảng đệm để tận dụng không gian. Giảm cỡ chữ tiêu đề tiết kiệm chiều cao vì trên điện thoại phải cuộn nhiều. Các nút xếp dọc thay vì ngang, mỗi nút rộng 100% để dễ nhấn bằng ngón tay.

2.4 Xử lý tương tác bằng JavaScript

2.4.1 Lắng nghe hành động người dùng

JavaScript làm cho trang web phản ứng lại khi người dùng làm gì đó.

```
let formData = {};

document.addEventListener('DOMContentLoaded', () => {
    const form = document.getElementById('scholarshipForm');

    form.querySelectorAll('input, select, textarea').forEach(
       field => {
        field.addEventListener('blur', () => {
            validateField(field);
        });
    });

    form.addEventListener('submit', handleSubmit);
});
```

Listing 2.19: Khởi tạo ban đầu

Dòng đầu tạo một cái hộp rỗng để sau này chứa dữ liệu form. Sự kiện "DOMContentLoaded" kích hoạt khi trang đã sẵn sàng, không chờ ảnh tải xong - thời điểm lý tưởng để bắt đầu chạy JavaScript. Tìm form theo mã số, sau đó tìm tất cả các ô nhập bên trong. Với mỗi ô, ta gắn một "người lắng nghe" - khi người dùng rời khỏi ô (blur) thì chạy hàm kiểm tra. Cuối cùng lắng nghe sự kiện gửi form (submit).

```
field.addEventListener('input', () => {
    if (field.classList.contains('error')) {
        validateField(field);
    }
});
```

Listing 2.20: Kiểm tra khi đang gõ

Sự kiện "input" kích hoạt mỗi khi người dùng gõ một ký tự. Nhưng ta chỉ kiểm tra lại khi ô đang có lỗi (có class "error") - khi người dùng sửa thì thông báo lỗi tự động biến mất, tạo trải nghiệm tốt mà không làm phiền khi đang gõ lần đầu.

2.4.2 Kiểm tra dữ liệu nhập vào

Mỗi loại thông tin có cách kiểm tra riêng.

```
function validateField(field) {
    const errorMsg = field.closest('.form-question')
                          ?.querySelector('.error-msg');
    let isValid = true;
    let message = '';

    if (field.hasAttribute('required') && !field.value.trim())
       {
        isValid = false;
        message = 'Trường này bắt buộc phãi điền';
    }
```

Listing 2.21: Hàm kiểm tra cơ bản

Hàm này nhận vào một ô cần kiểm tra. Đầu tiên tìm vùng hiện lỗi bằng cách đi lên tìm khung cha, rồi đi xuống tìm dòng thông báo. Dấu chấm hỏi là cách an toàn - nếu không tìm thấy thì không báo lỗi mà chỉ trả về rỗng. Biến "isValid" mặc định là đúng, sẽ chuyển sang sai nếu phát hiện lỗi. Kiểm tra đầu tiên: nếu ô bắt buộc thì phải có giá trị. Hàm xóa khoảng trắng đầu cuối - ngăn người dùng gõ toàn dấu cách để qua mặt.

```
if (field.type === 'email' && field.value) {
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!emailRegex.test(field.value)) {
        isValid = false;
        message = 'Email không đúng định dạng';
    }
}

if (field.name === 'phone' && field.value) {
    const phoneRegex = /^0\d{9}$/;
    if (!phoneRegex.test(field.value)) {
        isValid = false;
        message = 'SĐT phãi 10 số, bắt đầu bằng 0';
    }
}
```

Listing 2.22: Kiểm tra email và số điện thoại

Kiểm tra email dùng mẫu (pattern): phải có phần trước @, dấu @, tên miền, dấu chấm, và phần đuôi. Hàm "test" kiểm tra chuỗi có khớp mẫu không. Kiểm tra số điện thoại Việt Nam: bắt đầu bằng số 0, theo sau là đúng 9 chữ số nữa, tổng cộng 10 số.

```
if (field.name === 'gpa' && field.value) {
    const gpa = parseFloat(field.value);
    if (gpa < 0 || gpa > 4) {
        isValid = false;
        message = 'GPA phãi từ 0 đến 4';
    }
}

if (isValid) {
    field.classList.remove('error');
    if (errorMsg) errorMsg.textContent = '';
} else {
    field.classList.add('error');
    if (errorMsg) errorMsg.textContent = message;
}

return isValid;
```

Listing 2.23: Kiểm tra khoảng giá trị và cập nhật giao diện

Với điểm GPA, chuyển chuỗi sang số thập phân rồi kiểm tra có nằm trong khoảng 0-4 không. Cuối cùng cập nhật giao diện: nếu hợp lệ thì xóa class "error" và xóa thông báo lỗi. Nếu không hợp lệ thì thêm class "error" (làm viền đỏ) và hiện thông báo. Hàm trả về đúng/sai để các hàm khác biết kết quả.

2.4.3 Xử lý khi gửi form

Khi người dùng nhấn nút Gửi, ta kiểm tra toàn bộ rồi xử lý.

```
function handleSubmit(e) {
    e.preventDefault();

    const form = e.target;
    let isValid = true;

    form.querySelectorAll('input, select, textarea').forEach(
       field => {
        if (!validateField(field)) {
            isValid = false;
        }
    });

```

```
    if (!isValid) {
        form.querySelector('.error')?.scrollIntoView({
            behavior: 'smooth', block: 'center'
        });
        return;
    }
```

Listing 2.24: Kiểm tra toàn bộ form

Dòng đầu ngăn form gửi đi theo cách thông thường (tải lại trang). Lặp qua tất cả các ô và kiểm tra từng cái. Nếu có lỗi, tự động cuộn trang đến ô lỗi đầu tiên với hiệu ứng mượt mà, đặt ô đó ở giữa màn hình. Sau đó thoát khỏi hàm, không làm gì thêm.

```
    collectFormData(form);
    formData.submittedAt = new Date().toISOString();

    saveToLocalStorage(formData);
    showSuccessModal();
    form.reset();
```

Listing 2.25: Thu thập và lưu dữ liệu

Nếu tất cả hợp lệ, gọi hàm thu thập dữ liệu vào biến đã chuẩn bị. Thêm thời gian gửi dạng chuỗi chuẩn quốc tế. Lưu vào bộ nhớ trình duyệt - dữ liệu sẽ còn lại ngay cả khi tắt trình duyệt. Hiện hộp thông báo thành công. Cuối cùng xóa sạch form.

```
function collectFormData(form) {
    formData = {};

    form.querySelectorAll('input, select, textarea').forEach(
       field => {
        const name = field.name;
        if (!name) return;

        if (field.type === 'checkbox') {
            formData[name] = field.checked;
        } else if (field.type === 'radio') {
            if (field.checked) formData[name] = field.value;
        } else if (field.type === 'file') {
            if (field.files.length > 0) {
                formData[name] = Array.from(field.files).map(f
                   => ({
                    name: f.name, size: f.size, type: f.type
                }));
```

```
            }
        } else {
            formData[name] = field.value;
        }
    });
}
```

Listing 2.26: Thu thập dữ liệu từng loại

Lặp qua tất cả ô nhập, với mỗi loại xử lý khác nhau. Ô đánh dấu (checkbox) lưu giá trị đúng/sai của việc có tick hay không. Nút tròn (radio) chỉ lưu nếu nút đó được chọn. File lưu thông tin (tên, kích thước, loại) chứ không lưu nội dung ảnh. Các ô khác lưu giá trị trực tiếp.

```
function saveToLocalStorage(data) {
    let submissions = JSON.parse(
        localStorage.getItem('scholarships')
    ) || [];
    submissions.push(data);
    localStorage.setItem('scholarships', JSON.stringify(
       submissions));
}

function showSuccessModal() {
    document.getElementById('successModal').classList.add('show
       ');
}

window.closeModal = function() {
    document.getElementById('successModal').classList.remove('
       show');
}
```

Listing 2.27: Lưu trữ và hiển thị

Lấy danh sách các lần gửi trước (nếu có) từ bộ nhớ trình duyệt, chuyển từ chuỗi sang dữ liệu thật. Thêm dữ liệu mới vào danh sách. Chuyển lại thành chuỗi và lưu vào bộ nhớ. Hàm hiện modal tìm hộp thông báo theo mã số và thêm class "show" để nó xuất hiện. Hàm đóng modal xóa class "show" để ẩn đi.

2.5 Tổng kết chương

Chương này đã trình bày chi tiết cách xây dựng một form xét học bổng hoàn chỉnh từ đầu đến cuối. HTML tạo khung sườn với đầy đủ các loại ô nhập phù hợp từng loại thông tin. CSS trang trí giao diện đẹp mắt với màu sắc nhất quán, bo góc mềm mại, và tự động co giãn trên mọi kích thước màn hình. JavaScript xử lý mọi tương tác: khi người dùng gõ hoặc click, kiểm tra dữ liệu đúng định dạng, thu thập và lưu trữ thông tin, hiện thông báo phản hồi. Kết quả là một ứng dụng web hoàn chỉnh, dễ sử dụng và chuyên nghiệp, đáp ứng tốt nhu cầu thực tế.

Tài liệu tham khảo

[1] Mozilla Developer Network, “Html: Hypertext markup language,” 2025.

[2] Mozilla Developer Network, “Css: Cascading style sheets,” 2025.

[3] Mozilla Developer Network, “Javascript guide,” 2025.

[4] Mozilla Developer Network, “Introduction to the dom,” 2025.

[5] Mozilla Developer Network, “Client-side form validation,” 2025.

[6] Mozilla Developer Network, “Your first form,” 2025.