Faculty of Business and Technology: Programme -

NAME: Upendra Khanal

STUDENT NUMBER: 240704403

MODULE CODE: CET138

MODULE TITLE: Full Stack Development

MODULE LEADER: David Grey

SUBMISSION DATE AND TIME: 29 AUG 2025/23:59 GMT

ASSIGNMENT:

STATEMENT ON USE OF ARTIFICIAL INTELLIGENCE TOOLS:

• I have used artificial intelligence tools to generate an idea for my assignment:	**NO**
• I have used artificial intelligence tools to write my assignment for me:	**NO**
• I have used artificial intelligence tools to brainstorm ideas for my assignment:	**NO**
• I have used artificial intelligence tools to correct my original assignment:	**NO**

DECLARATION

ANY OTHER COMMENTS FROM STUDENT:

CSS (Cascading Style Sheets)

What is CSS?

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation and design of a document written in HTML or XML. While HTML defines the content and structure, CSS controls the visual aspects—such as layout, colors, fonts, spacing, and the overall look and feel of a webpage. It allows developers to create visually appealing and user-friendly designs by applying consistent styling across multiple pages. Beyond static styling, CSS can also add interactive elements through transitions and animations, such as fade-ins, hover effects, and sliding menus. It plays a key role in making websites responsive so they adapt seamlessly to different devices, including phones, tablets, and desktops, ensuring a smooth user experience regardless of screen size.

Without CSS, a website would appear as plain black text on a white background, with images stacked vertically and very little spacing or organization. There would be no consistent layout, no defined color scheme, and no visual hierarchy to guide the user's attention. By separating content from presentation, CSS keeps projects cleaner, more reusable, and easier to maintain, allowing for quick design updates without altering the HTML structure. This separation also improves accessibility, performance, and overall professionalism, making CSS an essential tool in modern web development.

Core Features of CSS

1. **Selectors & Properties —** Target elements and apply styles such as color, font-size, margin, and display.
2. **Typography Control —** Set fonts, sizes, line-height, letter/word spacing, alignment, and font-weight.
3. **Colors & Backgrounds —** Use solid colors, gradients, images, and blend modes for visual depth.
4. **Box Model —** Manage element size and spacing: content, padding, border, and margin.
5. **Layout Systems —** Build modern layouts with Flexbox and CSS Grid; use positioning where needed.
6. **Responsive Design —** Adapt designs with media queries and fluid units (%, vw/vh, rem).
7. **Transitions & Animations —** Animate properties smoothly and define keyframe animations for movement and emphasis.
8. **Pseudo-classes & Pseudo-elements —** Style element states (e.g., :hover, :focus) and virtual sub-parts (::before/::after).
9. **Custom Properties (CSS Variables) —** Store reusable values (colors, spacing) to keep styles consistent.
10. **Accessibility Enhancements —** Improve focus states, contrast, and motion preferences (prefers-reduced-motion).

Types of CSS

1. Inline CSS

Applied directly on an element with the `style` attribute. Affects only that element.

When to use: Use for very small, one-off tweaks or quick testing — not for production scale.

Pros: Fast to apply; overrides many other styles due to cascade specificity.

Cons: Mixes content and presentation; hard to maintain; not reusable; inflates HTML.

Example:
```
<p style="color: red; margin-top: 1rem;">This paragraph uses inline
CSS.</p>
```

2. Internal CSS

Placed inside a `<style>` block within the HTML `<head>`. Styles apply to the current page.

When to use: Use for single-page demos or when a page needs unique styles that won't be reused elsewhere.

Pros: Keeps styles in one place for a page; no extra HTTP request for a separate CSS file.

Cons: Not reusable across multiple pages; styles can grow messy as the page scales.

Example:
```
<style>
  h1 { color: #2b6cb0; }
  .intro { max-width: 60ch; line-height: 1.6; }
</style>
```

3. External CSS

Stored in a separate `.css` file and linked with `<link rel="stylesheet" href="styles.css">`.

When to use: Best for multi-page sites and production apps where reuse, caching, and maintainability matter.

Pros: Separation of concerns; reusable across many pages; browser caching; cleaner HTML.

Cons: Adds an extra HTTP request (minor; usually worth it); may require build/organization.

Example:
```
<link rel="stylesheet" href="styles.css">
```

CSS Example — Hosted on Github

Live Example Link:
https://upendra11223.github.io/CET138-A1-ePortfolio/css-demo/css-demo.html

This example is a pure HTML+CSS page featuring a gradient background, a responsive CSS Grid card layout, hover interactions, and keyframe animations. It demonstrates how CSS alone can transform plain HTML into a professional, interactive experience.

Detailed Breakdown of CSS Components in the Example

1. Global Styles & Typography

```
body {
  font-family: Arial, sans-serif;
  background: linear-gradient(120deg, #74ABE2, #5563DE);
  margin: 0;
  padding: 0;
  display: flex;
  flex-direction: column;
  align-items: center;
  min-height: 100vh;
  color: #fff;
}
```

font-family: Arial, sans-serif; Sets a clean, universally available font that ensures readability across devices.

background: linear-gradient(120deg, #74ABE2, #5563DE); Applies a modern gradient background that enhances visual appeal without needing images.

margin: 0; padding: 0; Removes default browser spacing, allowing full control over layout.

display: flex; flex-direction: column; align-items: center; Uses Flexbox to align and stack content neatly in the center of the page.

min-height: 100vh; Ensures the page fills the full height of the browser viewport.

color: #fff; Sets white text for maximum contrast and readability against the gradient background.

2. **Header Styling**

```
header {
  text-align: center;
  padding: 2rem 1rem;
  animation: fadeInDown 1s ease-in-out;
}
```

text-align: center; Centers the title and introductory content for better symmetry.

padding: 2rem 1rem; Adds space around the header, preventing elements from feeling cramped.

animation: fadeInDown 1s ease-in-out; Applies a smooth drop-down animation when the header appears, improving user experience.

3. **Responsive Card Layout (CSS Grid)**

```
.card-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 1.5rem;
  width: 90%;
  max-width: 1000px;
  padding: 2rem 0;
  animation: fadeInUp 1.2s ease-in-out;
}
```

display: grid; Enables CSS Grid layout to organize items in rows and columns.

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr)); Creates a responsive layout that adjusts the number of columns based on available space.

gap: 1.5rem; Adds consistent spacing between cards for visual separation.

width: 90%; max-width: 1000px; Restricts layout width for better readability on large screens.

padding: 2rem 0; Adds vertical spacing around the grid.

animation: fadeInUp 1.2s ease-in-out; Introduces the cards with a smooth upward fade animation.

4. Card Styling & Hover Effects

```
.card {
  background: rgba(255, 255, 255, 0.1);
  backdrop-filter: blur(10px);
  border-radius: 12px;
  padding: 1.5rem;
  text-align: center;
  transition: transform 0.3s ease, box-shadow 0.3s ease;
}
.card:hover {
  transform: translateY(-10px);
  box-shadow: 0 10px 20px rgba(0, 0, 0, 0.3);
}
```

background: rgba(255, 255, 255, 0.1); Creates a semi-transparent card background for a glassmorphism effect.

backdrop-filter: blur(10px); Blurs the background behind the card for a polished look.

border-radius: 12px; Rounds the card corners for a modern, soft appearance.

padding: 1.5rem; Adds internal spacing for readability.

text-align: center; Centers card content for balanced presentation.

transition: transform 0.3s ease, box-shadow 0.3s ease; Ensures smooth animations when the card state changes.

transform: translateY(-10px); Lifts the card slightly when hovered to indicate interactivity.

box-shadow: 0 10px 20px rgba(0, 0, 0, 0.3); Adds a subtle shadow for depth when hovered.

4. Animations

```
@keyframes fadeInDown {
  from { opacity: 0; transform: translateY(-20px); }
  to { opacity: 1; transform: translateY(0); }
}

@keyframes fadeInUp {
  from { opacity: 0; transform: translateY(20px); }
  to { opacity: 1; transform: translateY(0); }
}
```

@keyframes fadeInDown Animates elements by moving them from slightly above while increasing opacity from 0 to 1.

@keyframes fadeInUp Animates elements by moving them from slightly below while increasing opacity from 0 to 1.

5. **Footer Styling**

```
footer {
  text-align: center;
  padding: 1rem;
  font-size: 0.9rem;
  opacity: 0.8;
}
```

text-align: center; Centers the footer content for symmetry.

padding: 1rem; Adds vertical spacing so the footer does not touch the edges of the viewport.

font-size: 0.9rem; Uses slightly smaller text to indicate secondary content.

opacity: 0.8; Softens the text so it's visible but not distracting.