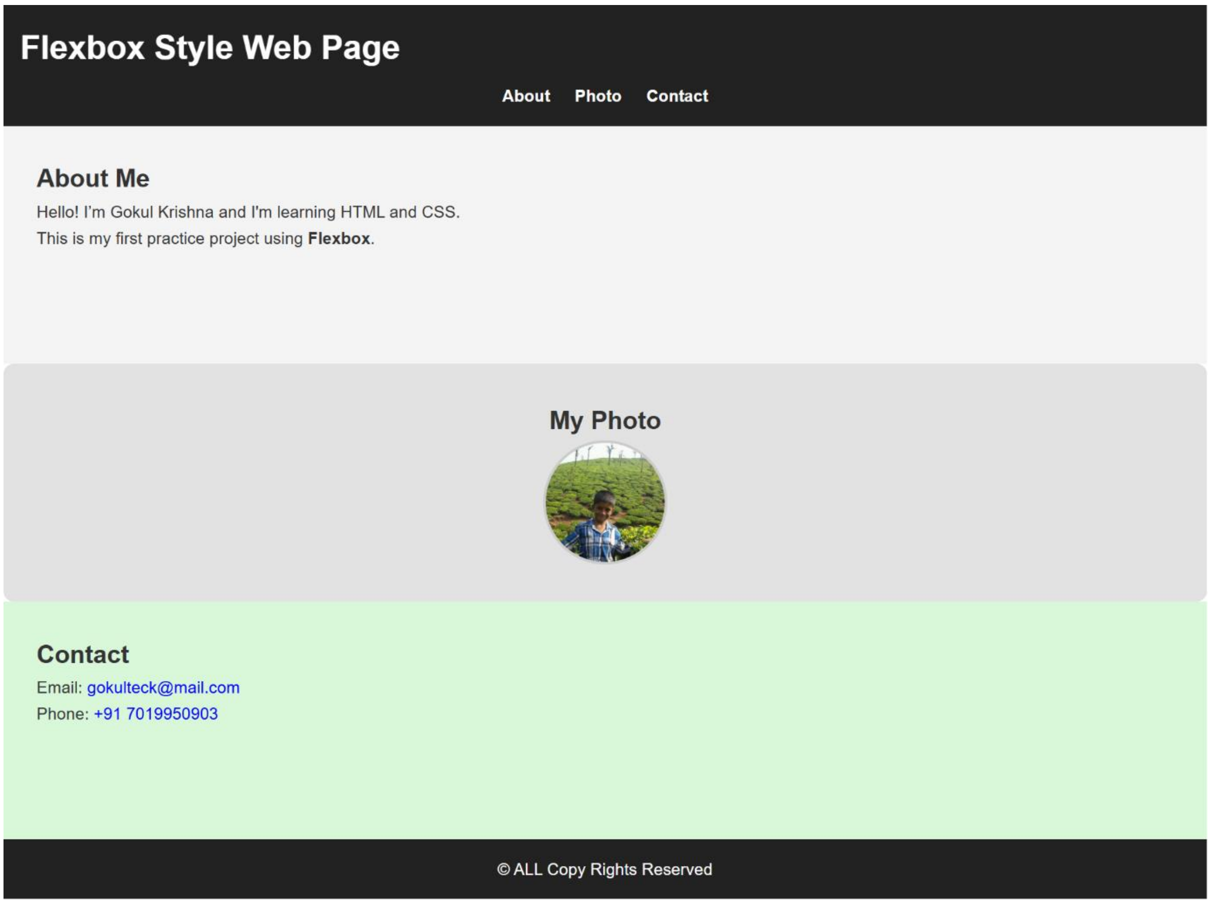
AeroAspire-SDE Intern
Gokul Krishna S

Week 1 – Day 2 (Sep23)

Task/Assignment :
Build basic HTML page: About / Photo / Contact sections, Style the sections;
header/nav/footer;
layout using Flexbox or Grid;

Flexbox Style



Flexbox Style Web Page

About Photo Contact

About Me

Hello! I'm Gokul Krishna and I'm learning HTML and CSS.
This is my first practice project using **Flexbox**.

My Photo

Contact

Email: gokulteck@mail.com
Phone: +91 7019950903

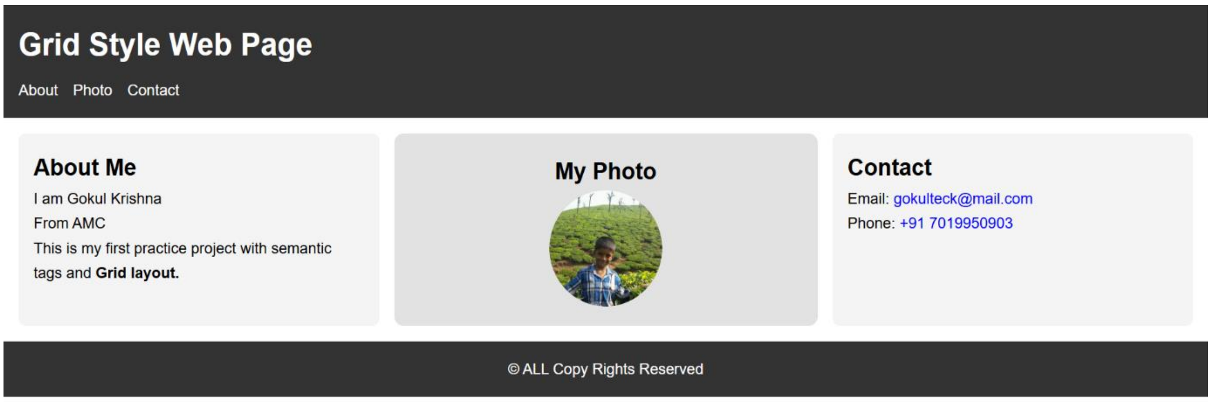
© ALL Copy Rights Reserved

Layout Direction

- **Flexbox:**
  - Works in one dimension at a time — column (vertical) or row (horizontal).
  - Great for arranging items in a line or stacking elements.

Here, I have used both **Flexbox** and **Grid** layouts to organize the content efficiently.
Flexbox is used for aligning elements in a single direction, such as arranging items in a row or column, which makes centering and spacing very easy.
On the other hand, **Grid** is used for creating a structured, two-dimensional layout, allowing me to manage rows and columns simultaneously, which is perfect for the overall page design.

Grid Style



Grid Style Web Page

About Photo Contact

About Me

I am Gokul Krishna

From AMC

This is my first practice project with semantic tags and **Grid layout.**

My Photo

Contact

Email: gokulteck@mail.com

Phone: +91 7019950903

© ALL Copy Rights Reserved

Layout Direction :

- **Grid :**
    - Works in **two dimensions** — rows and columns simultaneously.
    - Perfect for creating complex layouts like entire web pages or cards in rows and columns.

Questions/Reflection :

1. What is **<section>** vs **<div>**?

- **<div>** - A generic container with no meaning.
  - Used only for grouping or styling.
- **<section>** - A sematic element that represents a meaningful section of content. Ex : About, Services, Contact.

2. Why semantics matter?

- Semantic tags add meaning to HTML. Ex : <header>, <footer>, <article>, <nav>, etc.
- Benefits:
  - **Accessibility**: Screen readers can understand page structure better.
  - **SEO**: Search engines index pages more effectively.
  - **Readability**: Developers instantly understand content purpose.

3. What is the flow from writing HTML → rendering by browser?

1. Browser downloads HTML.
2. Builds the DOM (Document Object Model) tree.
3. Downloads and parses CSS → builds CSSOM (CSS Object Model)**.**
4. Combines DOM + CSSOM → Render Tree**.**
5. Browser calculates layout (size, position).
6. Paints pixels on the screen.

4. How does semantic HTML improve accessibility and SEO?

- **Clear Content Structure** – Tags like <header>, <main>, <footer> define sections, helping both screen readers and search engines understand the layout.
- **Better Navigation** – <nav> and headings (<h1>–<h6>) allow users and bots to easily find important content.
- **Improved Content Hierarchy** – Semantic headings and sections indicate importance, aiding accessibility tools and search indexing.
- **Enhanced Context for Content** – Tags like <article> and <section> provide meaning, helping users with assistive tech and improving SEO relevance.

- **Optimized User & Search Experience** – Semantic forms, buttons, and descriptive elements make the site easier to use and more likely to rank well in search results.

5. Describe how the browser parses HTML + CSS to render layout.

- The browser reads HTML and builds the DOM tree, representing the structure of the page.
- CSS files and <style> blocks are parsed into the CSSOM, which stores styling rules for elements.
- The browser determines the final computed styles for each render tree node (color, font, size, etc.).
- The browser calculates the exact size and position of each element based on the box model and parent-child relationships.
- The browser paints pixels for text, colors, borders, backgrounds, and images onto layers on the screen.

6. How Flexbox handles alignment when container resizes?

- Flexbox is responsive by design**.**
- Items grow/shrink according to available space using:
  - **flex-grow** → how much items expand.
  - **flex-shrink** → how much items shrink.
  - **justify-content & align-items** → control alignment.
    Example: If screen width shrinks, Flexbox adjusts spacing and wrapping automatically.

7. Describe the CSS box model and how margin/padding/border/content interact.

- Every element is a box:
  - **Content** → text/image.
  - **Padding** → space inside border, around content.
  - **Border** → around padding.
  - **Margin** → outside border, space between elements.
  - **Total width** = content + padding + border + margin.

8. What is the flow of CSS specificity?

CSS Specificity Flow

CSS specificity determines **which style rules take precedence** when multiple rules target the same element. The browser follows a **ranking system** to decide which style to apply.

1. Specificity Hierarchy (from lowest to highest)

- **Inline styles** – e.g., <p style="color:red;"> → **highest priority**.
- **IDs** – e.g., #header → overrides classes and elements.
- **Classes, attributes, pseudo-classes** – e.g., .menu, [type="text"], :hover.
- **Elements and pseudo-elements** – e.g., div, p, ::before → lowest priority.

9. How would you approach making a layout responsive?

- **Use Fluid/Relative Units** – Employ %, em, rem, vw, vh instead of fixed pixels.
- **Flexible Images & Media** – Scale images and videos with max-width: 100%; height: auto;
- **CSS Media Queries** – Apply different styles at breakpoints for mobile, tablet, and desktop.
- **Mobile-First Design** – Design for small screens first, then enhance for larger screens.
- **Use Flexbox/Grid Layouts** – Create adaptable, flexible layouts that adjust naturally to screen sizes.