University of Science and Technology
SW 302: User Interface Development

Assignment 2 Fall 2025

Assignment2 : Build an interactive Car Explorer Web App: 5-page website using HTML, JavaScript, and Tailwind CSS.

Car Explorer web app that must allows users to:

1. Browse and search cars fetched from a public API (e.g., from JSONPlaceholder or a custom JSON file).
2. View full car details (images, brand, price, model year, horsepower, etc.).
3. Favorite cars for later car comparisons functionality using localStorage.
4. Compare multiple cars side-by-side based on features.
5. Switch between custom themes using Tailwind (e.g., “Sport” and “Eco” modes).

This assignment requires dynamic data rendering, interactivity, storage, Tailwind customizations, and modular JS.

Required Pages

Page	Description
1. Home	Project intro, quick links, and theme toggle
2. All Cars	Fetch and display car list, with search/filter functionality
3. Car Details	View full details of a selected car using query parameters or ID
4. Favorites	Bookmark cars permanently using localStorage and view/edit list
5. Compare Cars	Select your favorite cars and compare their features side-by-side

JavaScript Logic Requirements

You must:

- Fetch data from a public API such as:
    - Fake data from JSONPlaceholder or
    - Custom JSON hosting car info via fetch().
- Implement at least five reusable functions, including two arrow functions.
- Use .map(), .filter(), .reduce(), etc. to transform data.
- Use localStorage to store favorite cars and theme settings.
- **Must** include the following interactive features:
    - Live search by model or brand
    - Filtering (e.g., by price or horsepower)
    - Adding/removing cars from favorites
    - Dynamic comparison system in "Compare" page
    - Sort cars by price or horsepower.
    - Filter cars by features (SUV, electric, hybrid, sports, luxury).

Tailwind CSS Requirements

You must:

- Create **two custom themes** (e.g., "Sport Mode" in red/black, and "Eco Mode" in green/white) in tailwind.config.js.
- Use at least **one Tailwind plugin**, like @tailwindcss/typography or @tailwindcss/forms.
- Create **2 reusable component classes** (e.g., .car-card, .button-action) using Tailwind @layer components.
- Make the site **responsive** across pages and devices using breakpoints.
- Must include animated loading skeletons using Tailwind UI.
- **Note:** The web UI must allow switching the theme from "Eco Mode" to "Sport Mode" → layout updates.

Usage Workflow Examples:

- Browse all cars → filter by price → open a card → view details.
- Mark 3 cars as favorites → open Compare → see side-by-side features.
- Switch theme from "Eco Mode" to "Sport Mode" → layout updates.
  Note: We cannot list all usage workflows, but we leave it to your creativity.

Suggested Folder Structure

```
/project
  /src
    /data
      cars.json
    /js
      api.js
      storage.js
      ui.js
      compare.js
      app.js
    /css
      styles.css
    /pages
      index.html
      cars.html
      details.html
      favorites.html
      compare.html
  tailwind.config.js
  package.json
  README.md
```

Grading Rubric (Updated)

Criteria	**Description**	**Points**
JavaScript Logic	Advanced use of ES6+, Fetch API, localStorage, DOM manipulation, and interactivity	35
Tailwind Customization	Themes, plugins, layers, and responsive styling	30
Functionality	All features work properly and interactively	25
Code Structure	Modular, readable, and well-commented code	10
Total		**100 pts**