

# Case Study 4: Creating Visualizations on the Web

## DAT-230 Data Visualization & Storytelling with AI

Instructor: Dr. Vikas Thammanna Gowda      Semester: Fall 2025  
Contact: vthammannagowda@champlain.edu      Office Location: West Hall 100  
Office Hours: TBD

**Case Study:** Used Car Sales Analysis on Web  
**Assigned Date:** 10/16/2025  
**Due Date:** 11/12/2025  
**Presentation:** 11/13/2025 Live in-class, 10–12 minutes

### 1 Module Learning Outcomes

This module focuses on developing both technical and design skills for creating data visualizations on the web for 4 weeks. By the end of Week 4, students will be able to:

- Install and configure a modern JavaScript development toolchain, and fetch & parse CSV data in a React application.
- Build basic visualizations (such as pie charts, bar graphs, and scatter plots) using the Recharts library, applying dynamic data transformations as needed.
- Manage application state and organize React components effectively for data-driven user interfaces.
- Apply user experience (UX) best practices, including creating responsive layouts and incorporating UI features like tooltips, legends, and appropriate loading or error messages.
- Document their development progress and reflect on design decisions through written progress reports.
- Deliver a polished, interactive data dashboard and confidently explain its design and implementation in a live demo.

### Summary

Overall, the learning outcomes capture the key competencies students are expected to develop: technical proficiency with modern web visualization tools, the ability to create interactive and user-friendly data displays, and strong communication skills to document and present their work.

### 2 Week 1: Environment & First Visualization

Week 1 introduces students to the development environment and guides them through building their first web visualization. This week is about setting up the necessary tools and creating a simple chart from a given dataset.

#### 2.1 Setup

**Details:** You begin by installing Node.js and a package manager on their machines. They then initialize a new React project using a template such as Create React App. Follow the steps below to set up the project:

- **Step 0:** Ensure you have Node.js installed on your machine.  
You can download it from <https://nodejs.org/>.

- **Step 1:** Open a terminal as an administrator. Verify Node.js by running `node -v` and `npm -v` in the terminal to check the versions.
- **Step 2:** Get into your project directory. Note that you can use the `cd "/path/to/your/folder"` command to change directories.
- **Step 3:** Create a React application using Create React App by running the command:

```
npx create-react-app used-car-dashboard
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Users\vthammannagowda\Desktop\Courses\Fall 2025\DAT_230\Module1> npx create-react-app used-car-dashboard

Creating a new React app in C:\Users\vthammannagowda\Desktop\Courses\Fall 2025\DAT_230\Module1\used-car-dashboard.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template
...
```

Figure 1: Creating a React App

This command sets up a new React project in a folder named `used-car-dashboard` as shown in Fig. 2.

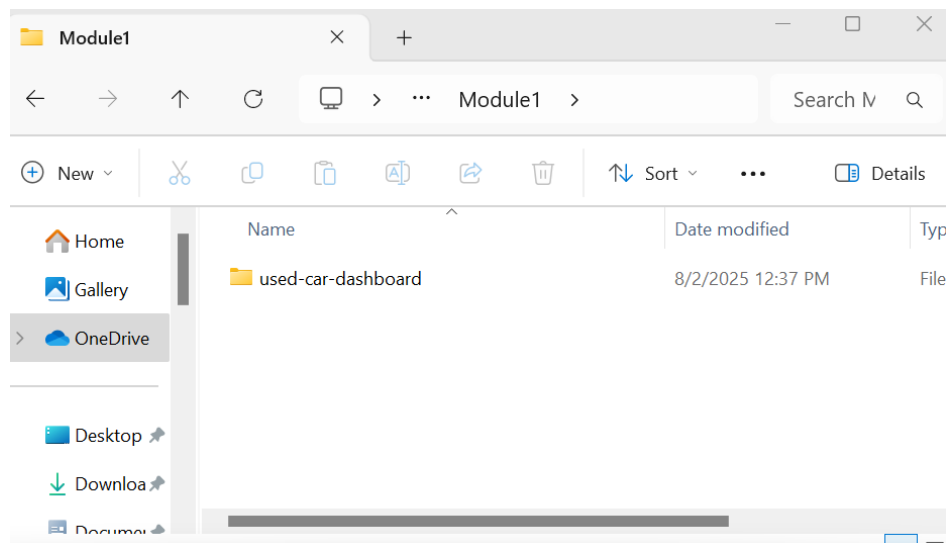
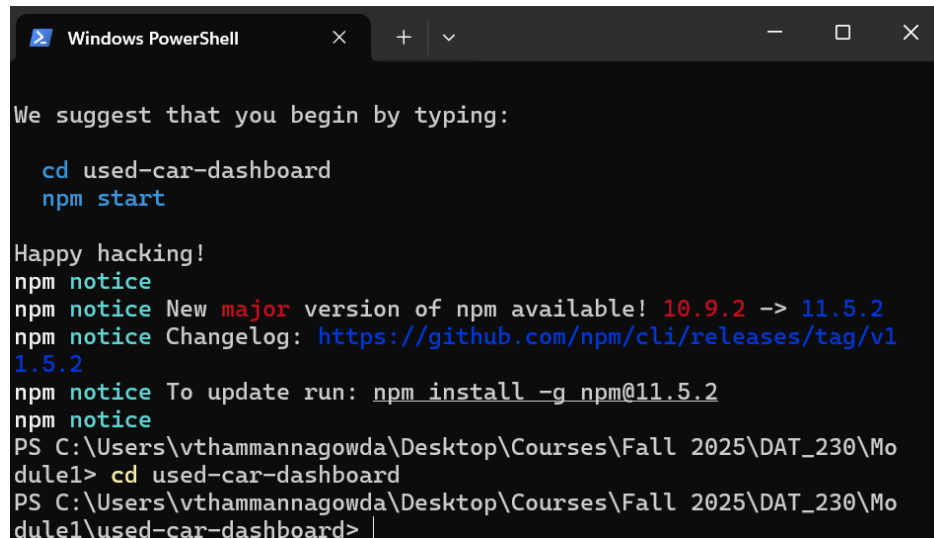


Figure 2: Folder set up by Create React App

- **Step 4:** Navigate into the newly created project directory by changing to the `used-car-dashboard` folder using the command:

```
cd used-car-dashboard
```



```
Windows PowerShell
We suggest that you begin by typing:

cd used-car-dashboard
npm start

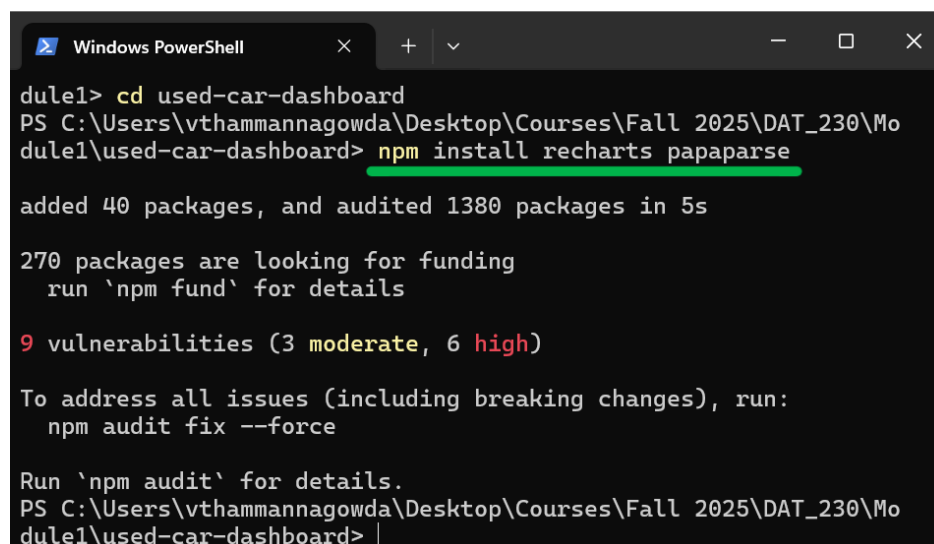
Happy hacking!
npm notice
npm notice New major version of npm available! 10.9.2 -> 11.5.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.5.2
npm notice To update run: npm install -g npm@11.5.2
PS C:\Users\vthammanagowda\Desktop\Courses\Fall 2025\DAT_230\Module1> cd used-car-dashboard
PS C:\Users\vthammanagowda\Desktop\Courses\Fall 2025\DAT_230\Module1\used-car-dashboard>
```

Figure 3: Setting `used-car-dashboard` folder as the current directory

- **Step 5:** Install the charting & CSV-parsing libraries using the command:

```
npm install recharts papaparse
```

This command installs the Recharts library for creating charts and PapaParse for parsing CSV data, which are essential for the project.



```
Windows PowerShell
dule1> cd used-car-dashboard
PS C:\Users\vthammanagowda\Desktop\Courses\Fall 2025\DAT_230\Module1\used-car-dashboard> npm install recharts papaparse

added 40 packages, and audited 1380 packages in 5s

270 packages are looking for funding
  run `npm fund` for details

9 vulnerabilities (3 moderate, 6 high)

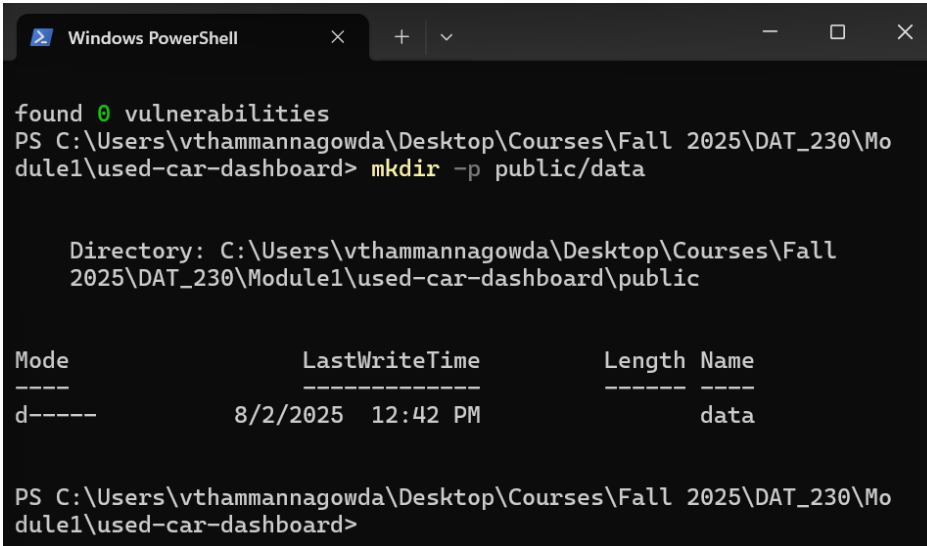
To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Users\vthammanagowda\Desktop\Courses\Fall 2025\DAT_230\Module1\used-car-dashboard>
```

Figure 4: Installing libraries

**Note:** You can fix the issues by running `npm audit fix --force` if you encounter any warnings during the installation.

- **Step 6:** Add your dataset to the project. Create a folder named `public/data` inside the project directory as shown Fig. 5,



```

found 0 vulnerabilities
PS C:\Users\vthammanagowda\Desktop\Courses\Fall 2025\DAT_230\Module1\used-car-dashboard> mkdir -p public/data

Directory: C:\Users\vthammanagowda\Desktop\Courses\Fall 2025\DAT_230\Module1\used-car-dashboard\public

Mode                LastWriteTime         Length Name
----                -
d-----            8/2/2025  12:42 PM              data

PS C:\Users\vthammanagowda\Desktop\Courses\Fall 2025\DAT_230\Module1\used-car-dashboard>

```

Figure 5: Creating a data folder

Place your CSV file (for example, `used_car_clean.csv`) in this folder as shown in Fig. 6. This structure allows the React app to access the data easily.

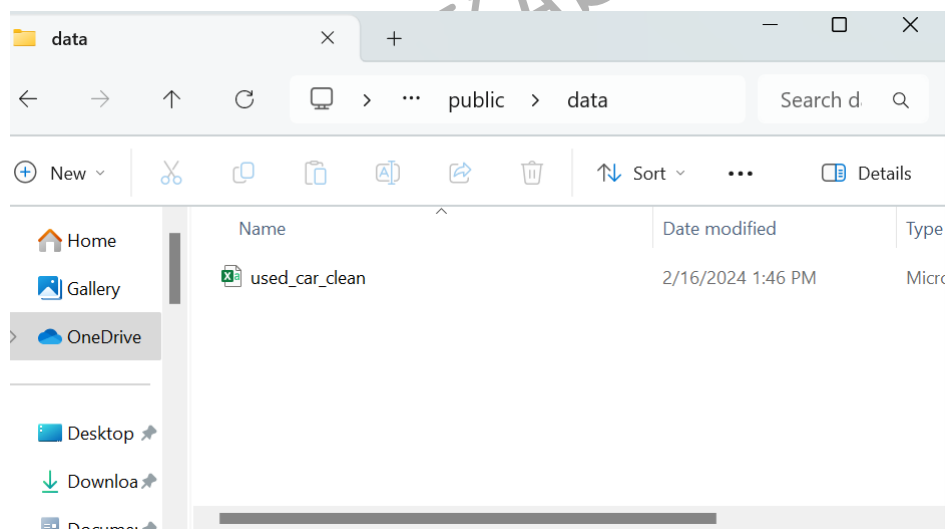


Figure 6: Storing the dataset in the data folder

- **Step 7:** Replace `src/App.js` with the given starter code:

```

// src/App.js
import React, { useEffect, useState } from 'react';
import Papa from 'papaparse';
import {
  PieChart,
  Pie,
  Cell,

```

```

    Tooltip,
    ResponsiveContainer
  } from 'recharts';

function App() {
  const [chartData, setChartData] = useState([]);

  useEffect(() => {
    // 1. Parse the CSV from the public folder
    Papa.parse('/data/used_cars.csv', {
      header: true,
      download: true,
      dynamicTyping: true,
      complete: ({ data }) => {
        // 2. Count records by body_type
        const counts = {};
        data.forEach(row => {
          if (row.body_type) {
            counts[row.body_type] = (counts[row.body_type] || 0) + 1;
          }
        });
        // 3. Convert to [{ name, value }] for Recharts
        const pieData = Object.entries(counts).map(
          ([name, value]) => ({ name, value })
        );
        setChartData(pieData);
      }
    });
  }, []);

  // 4. Colors for each slice
  const COLORS = ['#0088FE', '#00C49F', '#FFBB28', '#FF8042'];

  return (
    <div style={{ width: '100%', height: 400, textAlign: 'center' }}>
      <h1>Cars by Body Type</h1>
      <ResponsiveContainer>
        <PieChart>
          <Pie
            data={chartData}
            dataKey="value"
            nameKey="name"
            cx="50%"
            cy="50%"
            outerRadius={100}
            label
          >
            {chartData.map((entry, idx) => (
              <Cell key={idx} fill={COLORS[idx % COLORS.length]} />
            ))}
          </Pie>
          <Tooltip />
        </PieChart>
      </ResponsiveContainer>
    </div>
  );
}

```

```
    </div>  
  );  
}  
  
export default App;
```

VIKAS

- **Step 8:** Start the development server by running:

```
npm start
```

Your browser should open <http://localhost:3000> showing the pie chart of “Cars by Body Type.”

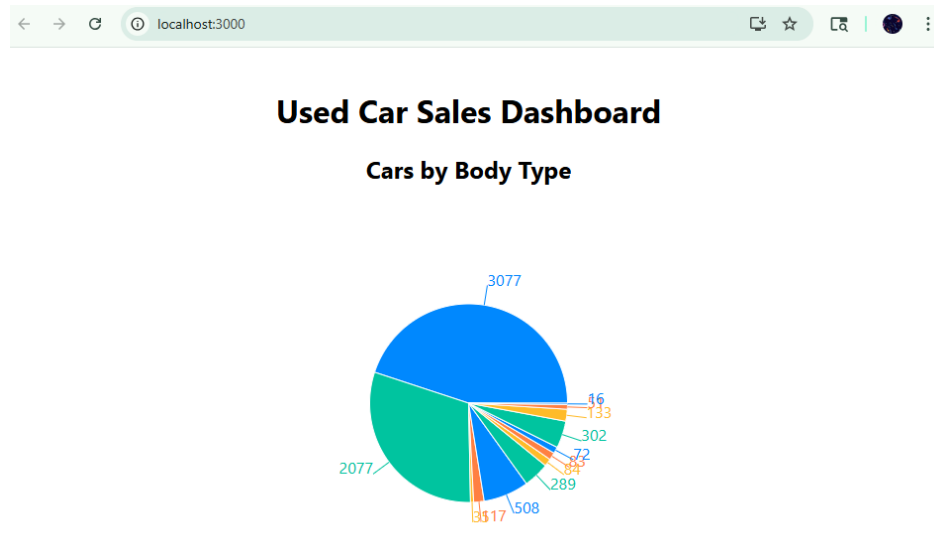


Figure 7: Browser displaying the pie chart

## 2.2 Starter Code Walk-through

**Details:** After setup, the instructor provides a walk-through of the starter code. The codebase includes a specific folder structure and a sample dataset (for example, a CSV file of used car listings located at `public/data/used_car_clean.csv`). You will learn how the application can fetch this data using a call like `fetch('/data/used_cars.csv')` and then parse it with PapaParse. This demonstration shows how raw CSV data is loaded and converted into a format suitable for visualization.

## Summary of Week 1

In summary, Week 1 lays the foundation by equipping students with a proper development setup and guiding them through their first data visualization. By completing a simple pie chart based on real data, students gain confidence in the workflow of loading data and visualizing it in React. This sets the stage for more complex visualizations and interactive features in the following weeks.

### 3 Weeks 2-3: Independent Development & Progress Report

Weeks 2 and 3 are devoted to independent development, where you will extend the dashboards with new features and gradually improve the user experience. During this period, students also document their progress in two written reports.

#### 3.1 Extending the Dashboard

Week 2 and 3 focuses on adding new visualizations to the initial dashboard and optimizing data handling.

**Objectives:** By the end of Week 3, each student or team will have implemented at least two new visualizations (for example, a bar chart of car makes and a scatter plot of price vs. mileage). Students should also use techniques like React's `useMemo` (or similar memoization) to optimize any expensive data transformations for performance.

**Activities:** During this week, students work individually or in pairs to extend the starter code with additional charts. They can attend office hours to get help on issues such as managing React state or customizing chart configurations. Peer collaboration is encouraged to share tips and troubleshoot problems using LLMs.

**Deliverable (Due: By start of Week 3 Class 3):** Progress Report, a roughly 500-word document. In this report, each group or individuals will describe which charts they built and the rationale behind choosing those visualizations. They should include screenshots of **any 2 new** chart and discuss any blockers or design decisions encountered.

**Note: Use of LLM to generate report is strictly prohibited. Any violation of this rule will result in academic dishonesty.**

**Assessment:** The progress report is evaluated based on a rubric focusing on clarity, completeness, and reflection. Clarity and completeness entail providing a thorough description of the work done and including all required elements (such as meaningful screenshots). Reflection involves thoughtful discussion of why particular charts or techniques were chosen, and how challenges were addressed. The instructor will also verify that the submitted code runs without errors and that the new charts render correctly in the application (a basic code functionality check).

#### Summary of Weeks 2–3

By the end of Week 3, students have significantly expanded their data visualization dashboards beyond the initial pie chart. They have incorporated multiple types of charts and added interactive features to create a more engaging user experience. Through the progress report, students practiced articulating their design decisions and technical solutions, while also receiving feedback to guide further improvements. These weeks of independent work prepare the students for showcasing a polished, fully functional dashboard in the final week.



## 4 Week 4: Final Demo & Reflection

Week 4 is dedicated to finalizing the project and presenting it. Students fix any remaining issues, ensure the application is polished and responsive, and then demonstrate their work to the class.

### 4.1 Final Touches

**Details:** At the start of Week 4, students focus on clean-up and final improvements. You are required to generate 5 distinct visualizations, ensuring that the dashboard is fully functional and visually appealing. They resolve any remaining bugs identified in earlier weeks and verify that the dashboard works smoothly on different devices (ensuring mobile responsiveness). Students also prepare project documentation that provides setup instructions and a summary of the dashboard's features. These final touches ensure that the project is ready for deployment and presentation.

**Deliverable:** There is no new code deliverable at this stage beyond what has already been built; rather, this step is about preparing for the presentation. Students are expected to submit an up-to-date codebase and clear documentation ready (500 - 750 words) along with a reflective paragraph on integrating AI tools into their learning process. for submission along with their final presentation.

**Assessment:** You will be evaluated similar to the previous progress report and carries 20%.

### 4.2 Peer Interviews

**Details:** Find two people to interview about the generated plots. They should be people who have not studied data visualization and don't typically use it in their day-to-day lives. Show each person the chart, and do a brief interview answering the following questions like:

- What does this chart say? Notice what order they go in, and what elements they talk about first.
- What is their takeaway from this chart? (Follow-ups: Did they learn anything new? Did anything in the chart surprise them?)

For each person, summarize the feedback you received. Then answer the following:

- What elements seemed to be the most/least effective in helping the person understand your chart?
- Did this interview change the way you looked at your chart?
- What were the similarities between how people processed and understood your chart? Were there things that one person got that the other didn't?
- Will you change anything about your charts as a result of these interviews? If so, what?

### 4.3 In-Class Demos

**Details:** During the class meeting in Week 4, each student or team gives a live demo of their data visualization dashboard along with the results/discussions of the interviews. Each presentation lasts about 10-12 minutes.

**Presentation Length Penalty:** The presentation must be at least **10 minutes** long. If the presentation is shorter than 10 minutes, **10% of the presentation score will be deducted for each full minute below 10.**

*Example:* A presentation lasting 8 minutes is 2 minutes under the minimum and will incur a 20% deduction on the presentation component.

**Note:** You must be present during all live presentation of your peers to receive credit.

**Deliverable:** The primary deliverable is the live demonstration itself, accompanied by the submission of supporting materials. By the demo day, students must submit their complete project code and a brief written reflection. This written report (approximately 500-700 words) should describe how AI tools or resources were used to help learn concepts or solve problems during the project. The inclusion of this “AI reflection” encourages students to think about their learning process and how modern tools can support their work.

VIKAS

**Assessment:** The final demo is evaluated using a detailed rubric.

Criterion	Weight (%)
Functionality of the dashboard	20%
Summary of the interviews	20%
Design and user experience quality	10%
Presentation clarity	15%
Q&A performance	15%
Report	20%

Table 1: Evaluation rubric for the final demo

### Summary of Week 4

In summary, Week 4 brings everything together: students put the finishing touches on their projects and then demonstrate a polished, interactive dashboard to the class. This final week emphasizes not only the importance of a well-functioning and well-designed application, but also the ability to communicate one's work and reflect on the learning journey (including the use of AI in a classroom setting).

VIKAS

## 5 Grading Breakdown

The course grading is concentrated on two progress update reports and the final project presentation. The weights and evaluation criteria are as follows:

- **Progress Report #1 (due end of Week 2) – 30%:** This update report (around 500 words) should detail the charts added in Week 2 with explanations and screenshots. The grading rubric for this report values clarity in describing what was done, completeness in covering all required elements, and thoughtful reflection on any challenges or design decisions. In addition, the instructor will verify that the submitted code runs properly and that the new visualizations are correctly implemented (a code check for functionality).
- **Final Presentation and Submission (Week 4) – 70%:** This capstone assessment consists of the in-class demo, the final code submission, and the AI reflection report. The demo portion is scored using a rubric: approximately 40% of the points for demonstrating a fully functional dashboard that meets all requirements, 30% for the design and user experience quality, and 30% for the effectiveness of the presentation (including the Q&A session). The submitted code is reviewed to ensure it is complete, well-organized, and consistent with what was shown in the demo. Additionally, each student must turn in a brief report ( 500 words) on how AI tools or resources aided their learning and development process in this course. While this AI reflection is a required part of the final submission, its insights can enrich the overall evaluation by highlighting the student's learning strategies.

VIKAS