CS3404: GUI Programming - Mini Project
The Modern Single Page Application (SPA)

Computer Engineering

January 31, 2026

1. Project Overview

Students are required to develop a **Data-Driven Single Page Application** (SPA) using Vue 3, TypeScript, and Tailwind CSS.

The application must consume data from the public API DummyJSON.com. You have the freedom to choose the domain of your application based on the available API endpoints (e.g., E-Commerce Store, Recipe Book, User Dashboard, Blog).

Learning Outcomes

- Implementing strictly typed Front-End Architecture (TypeScript Interfaces).
- Consuming Asynchronous REST APIs using `fetch` or `axios`.
- Managing Component State and Props flow.
- Implementing modern styling using a Utility-First Framework (Tailwind).
- **Prompt Engineering:** Effectively using AI to accelerate development.
- **Version Control:** Managing project history using Git.

2. Technical Requirements (Mandatory)

1. **Tech Stack:** Vue 3 (Composition API), TypeScript, Vite, Tailwind CSS.
2. **No "Any" Types:** You must define strict Interfaces for all API responses.
3. **Component Architecture:** The app must not be a single App`.`vue. You must break the UI into logical, reusable components (e.g., `ProductCard.`vue, `NavBar.`vue, `FilterBar.`vue).
4. **Responsiveness:** The layout must be fully responsive (Mobile, Tablet, Desktop) using Tailwind Grid/Flex classes.
5. **Data Interaction:**
   - Fetch data from at least one endpoint (e.g., `/products`).
   - Implement **Search** or **Filtering** (e.g., Filter by Category).
   - Clicking an item must show a **Detail View** (Modal or separate route).

3. Bonus Features (For A/A+ Grades)

To achieve a Distinction level, implement **at least one** of the following:

- **Authentication Simulation:** Use the `/auth/login` endpoint from DummyJSON. Store the received JWT token in `localStorage` and manage a ”Log In / Log Out” state.
- **Shopping Cart / Bookmarks:** Implement a global state (using Pinia or Composables) to add items to a cart/list that persists across page reloads.
- **Dynamic Routing:** Use Vue `Router` to handle navigation (e.g., `/product/:id`).
- **Dark Mode:** Toggle between Light/Dark themes using Tailwind’s `dark:` modifier.

4. GenAI Policy & Deliverables

You are **encouraged** to use AI (Gemini, ChatGPT) to generate CSS classes, debug errors, or generate TypeScript interfaces. However, you must prove you understand the code.

Submission Requirements

You must submit two items: a **GitHub Repository Link** and a **ZIP File**.

A. GitHub Repository

Provide a link to your public repository. This will be used to assess your commit history and development progress.

B. ZIP File Content

The ZIP file must contain:

1. **Source Code:** The complete Vue project folder (excluding node_`modules`).
2. **README.md:** Instructions on how to install dependencies and run the project (e.g., `npm install`).
3. **Report.pdf:** A PDF document containing:
   - List of features implemented.
   - A brief architectural explanation of your Component hierarchy (diagrams are welcome).
4. **`prompts.txt`:** A log of your AI usage. You must strictly follow this format:

```
The aim 01: [What you wanted to achieve]
Actual prompt: [The exact text you pasted into AI]

The aim 02: [Next goal]
Actual prompt: [Next prompt]
```

5. Assessment Rubric

Criteria	C: Pass (40-54)	B: Credit (55-69)	A: Distinction (70-84)	A+: Outstanding (85+)
Functionality	App runs. Fetches data. Basic display works. Minor bugs present.	Search/Filter works correctly. UI updates reliably. No console errors.	Smooth interactions. Detail view implemented. Bonus feature attempted.	Complete simulation (Auth, Cart, etc.). Feels like a production app.
Architecture & TS	Heavy logic in one file. Heavy use of any.	Basic component split. Interfaces defined but loose.	Clean component separation. Strict Types. Reusable Props.	Advanced patterns (Composables/Pinia). 100% Type Safety.
UI / UX (Tailwind)	Basic layout. Elements misaligned on mobile.	Responsive grid. Clean spacing. Consistent colors.	Polished UI. Hover states. Transitions. Accessible colors.	Professional design system. Dark mode support. Custom animations.
Git Activity	Single ”Upload files” commit or very few large commits.	Regular commits but vague messages (”Update”).	Meaningful commit history. Messages describe changes (”Added Filter”).	Professional history. Atomic commits. Use of feature branches.
GenAI Usage	No log or code looks auto-generated without understanding.	Prompts included but generic (”Write code”).	Used AI for tedious tasks (CSS/Types) but controlled logic manually.	Used AI to audit code or optimize architecture. Insightful, specific prompts.

Instructor Tips for Students

- **Git Early, Git Often:** Do not wait until the end to commit. We want to see your progress in the GitHub history.
- **Start Simple:** Fetch the data and show it in a list first. Don't style until the logic works.
- **Use the Docs:** Read the DummyJSON Documentation carefully to see what fields are available.