Project Rationale

Tool Choice

For this project, we chose to build a mobile-friendly static web application using HTML, CSS (Bootstrap 5), and JavaScript, mainly eue to our team's familiarity with these underlying technologies. This tool-set enabled rapid development, maximum device compatibility, and avoided the complexity of setting up a back-end server. The site is deployed using GitHub Pages, which serves it securely over HTTPS with no infrastructure maintenance.

We integrated several open-source libraries that significantly enhanced the functionality:

• Bootstrap (https://getbootstrap.com) — for responsive layout and mobile-first design
• Flatpickr (https://flatpickr.js.org) — to provide intuitive start and end date pickers
• PapaParse (https://www.papaparse.com) — for exporting filtered data as CSV
• jsPDF + AutoTable (https://github.com/parallax/jsPDF) — for formatting and exporting purchase data as PDF

All other JavaScript logic was written manually, with key portions generated using the AI assistant ChatGPT (OpenAI, 2024), which accelerated development and aided in resolving layout and interactivity issues.

User Story Satisfaction

The application fully satisfies the user story by implementing the following features:

- A date-range picker to select a start and end date.
- A scrollable, responsive table that displays purchase data including date, merchant, amount, and category.
- Export buttons that allow for the download of filtered purchases in either CSV or PDF file formats.
- Graceful handling of empty results with a 'No purchases found' alert.
- Realistic mock data of 30 transactions spanning more than 3 months.

Limitations and Next Steps

At present, the application lacks additional filtering by category, keyword search, or sort capabilities, which are common in production retail applications. These could be added to enhance user experience and functionality.

Because the app is fully client-side, it cannot simulate database connectivity, query optimization, or performance under real-world server load. A future version could connect to a RESTful API for data retrieval and user authentication.

Finally, the user interface was designed based on best practices and heuristics, but would benefit from a formal usability test or product owner review to validate its effectiveness and accessibility.