Write-Up

This is a write-up that covers

- the overall architecture and design of this application
- key decisions made in the implementation, including any trade offs or areas that could be improved upon with more time; and
- any additional features or functionality that could be added to the application given more time.

Overall Architecture & Design

- backend: Python with Flask and Stripe Checkout APIs
- frontend: React.js with TypeScript
- automated linting: Flake8 for backend, ESLint for frontend
- automated testing: pytest for backend
- CI/CD: GitHub Actions
- containerization: Docker

Key Decisions

1. Downloaded the sample app from Stripe Checkout Quickstart docs here to save time.
2. Switched from using an array to using a hash table to hold the items in the shopping cart, because we can buy two of the same item! So it's nice to have a hash table that looks like this: `{"egg": 2}`, which represents that have two of the same egg in the shopping cart.
3. Converted the React frontend from JavaScript to TypeScript, which was a life-saver in debugging.
4. Automated linting and testing in CI/CD.
5. Moved the `STRIPE_API_KEY` secret from being hard-coded in the Python script (dangerous!) to environment variables both locally and in GitHub Actions secrets.
6. Downgraded React and TypeScript versions in `package.json` so that `npm install` is compatible across all required node packages.
7. Moved all Python backend code into the `/backend` folder and all React frontend code into the `/frontend` folder; this allowed the folders to operate as separate Docker contexts.
8. Made the buttons prettier because the default buttons were just ugly.

Trade-Offs

- Instead of using Tailwind CSS, which allows CSS styling in HTML elements (thus eliminating all the `.css` files), I just used vanilla CSS for this project. Because I tried getting Tailwind CSS to work with the Stripe sample app for 2+ hours and decided it's not worth it due to the time constraint.

Areas to be Improved

- Probably the user interface (UI) design. If I have more time, I would definitely try to make the UI prettier and more user-friendly.

Next Steps (Additional Features & Functionalities)

- Deploy the web app to AWS using infrastructure as code (IaC); i.e. AWS Cloud Development Kit (CDK).
- Monitor the web app using Datadog, enabling observability.
- Create the `Product` prop instances from a database instead of hard-coding them.
- Use Tailwind CSS so we can eliminate all the `.css` files!
- Make the web app compatible with tablets and smartphones.
- Decide to have the `ShoppingCart.tsx` component be a child component of whether the `App.tsx` component or the `Header.tsx` component. Not sure which approach improves debuggability at the moment.
- Test the React components, props, and states by writing unit tests using the React Testing Library.