Senior Design Final Report
Team Pachira
Becky Shanley, Tuhina Dasgupta, Lily Schwarz

Section 1

Tuhina: primarily Chrome Extension, some parts of Django views (sust, snippets, nlp)

Lily: Getting alternative product suggestions, Personal Sustainability Quiz, Data Visualization, Sustainability Scoring

Becky: primarily Django web app, infrastructure and deployment, login/signup/authentication

Section 2

Online shopping is an essential part of many people's daily lives. Whether the reason for online shopping is out of necessity or convenience, the reality is that the modern consumer is more likely to purchase online than they are to go to a department store. The modern consumer is also becoming more environmentally conscious through their shopping by being more selective about which brands they buy from. These two sentiments don't always align, however. Online shopping incurs a carbon footprint that shopping in person at department stores doesn't: the transportation of their order. This, coupled with the regular carbon footprint produced by products, like fast fashion overproduction and waste, can lead consumers feeling cognitive dissonance when purchasing their goods online.

This is where our product, Pachira, comes in. Pachira is a browser extension that enables online shoppers to scan their shopping carts and receive a "sustainability score" on their shopping cart. It will evaluate factors such as whether or not it is fast fashion, internationally shipped, and others to see how sustainable the production and shipping of your entire shopping cart is. Then, if their score doesn't meet a score that a user chooses for their sustainability goal, Pachira will find alternatives, be it a local store selling something similar, or another online merchant selling the same product but shipping more sustainably. This way, Pachira helps consumers make educated and sustainable decisions while enjoying the convenience of online shopping.

Section 3

https://rebeccc.github.io/pachira/

Senior Design Final Report
Team Pachira
Becky Shanley, Tuhina Dasgupta, Lily Schwarz

Section 4

APIs

-Rapid Product Categorization

-Yelp

-Open Business

-Trip to Carbon

Packages (Pip)

django-cors-headers

django

djangorestframework

pygments

gunicorn

psycopg2

google-search-results-serpwow

simplejson

monkeylearn

djangorestframework-simplejwt

geopy

Senior Design Final Report
Team Pachira
Becky Shanley, Tuhina Dasgupta, Lily Schwarz

Section 5

Pachira allows users to transition to sustainable shopping through an easy-to-use and convenient web extension. Pachira's innovative nature stems from its cohesive toolset, as different aspects of Pachira are found in other tools, but Pachira offers them all in one convenient location. Pachira users will be prompted to make an account using the web app interface, and their sustainable shopping history, location, and shopping preferences will be stored. The natural language processing component of Pachira will then use that data to make personalized, sustainable shopping recommendations based on the products parsed from the user's online shopping cart. Pachira is technically novel due to the fact that it combines many different technologies in order to create the most seamless user experience.

Pachira is composed of three major components: Chrome extension, web application, and machine learning model. The Chrome extension, consisting of HTML and vanilla Javascript, uses message passing to be able to parse a user's shopping cart into JSON to send to the web app. The web application is deployed on AWS Elastic Beanstalk and contains a load balancer, compute instance, and relational database. The web application is a Django server and computes the sustainability score of the user's shopping cart contents. This score is then sent and displayed in the Chrome extension. Upon seeing the sustainability score of their cart, users can choose to see alternatives. These alternative choices are calculated by a machine learning collaborative filtering algorithm, which outputs recommendations with a lower sustainability score.

Section 6

Becky:

I would have liked to incorporate a better UI for the web app. Additional user functionality on the web app, such as choosing what kind of statistics are being displayed. Additional login functionality such as the ability to reset a user's password would have been great. I learned overall about the importance of planning ahead and developing iteratively when developing a product like this (we had to redo a lot of things whenever something went wrong).

Tuhina:

I would likely have used a front end framework for the Chrome Extension instead of writing a UI from scratch and implementing all of the REST calls by hand. A few popular options include React.js, Vue.js, and Angular.js. Additionally, I would have liked to have been able to dedicate time to making the extension cross-browser compatible, like browsers such as Firefox, Safari, and Opera. If I had time, I would have also liked to dedicate a few weeks to making Django more secure by implementing the Let's Encrypt package.

Senior Design Final Report
Team Pachira
Becky Shanley, Tuhina Dasgupta, Lily Schwarz

Lily:

I would have liked to have looked into a few more backup options for the API we used to categorize products, as towards the end of the project the API often was down. Additionally, I would have liked to look into designing a better dashboard for the data visualization, and I wish I had talked to Elyse about making improvements on it.

Section 7

1. How to download and get the project working. If there's equipment, a description of where to purchase (what model # etc).

   Both the Chrome Extension and Django Web App are available for download from github and their respective README's contain instructions on how to get them running. The Chrome Extension needs to be loaded onto the Chrome Extension Manager. The Web App needs several packages to be installed (detailed earlier in this report), a running Postgresql database, and then can be run locally.

2. What works, what doesn't, what to be aware of (pitfalls, issues).

   The product suggestion engine relies on an API that has been down recently, so checking that the API is up is a consideration to make. Chrome Extension localStorage can be inconsistent so reloading is important.

3. Ideas for next steps.
   - Cart contents sustainability scoring API implementation (CarbonScope, CarbonKit)
   - Increased Django Security
   - Chrome Extension UI framework
   - Additional APIs added to Corporate Sustainability (Open Business is primarily for UK companies)
   - Browser Agnostic