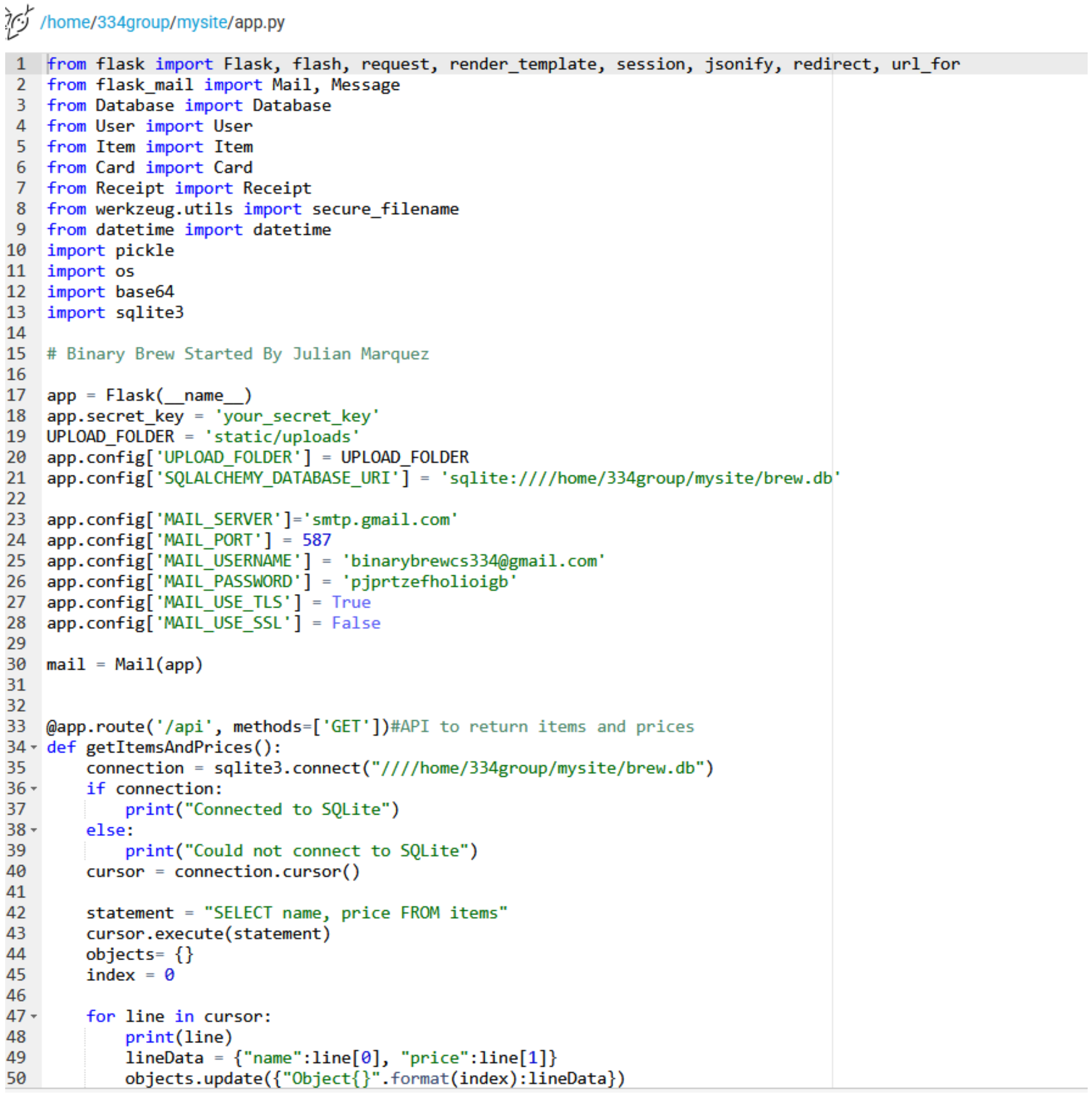
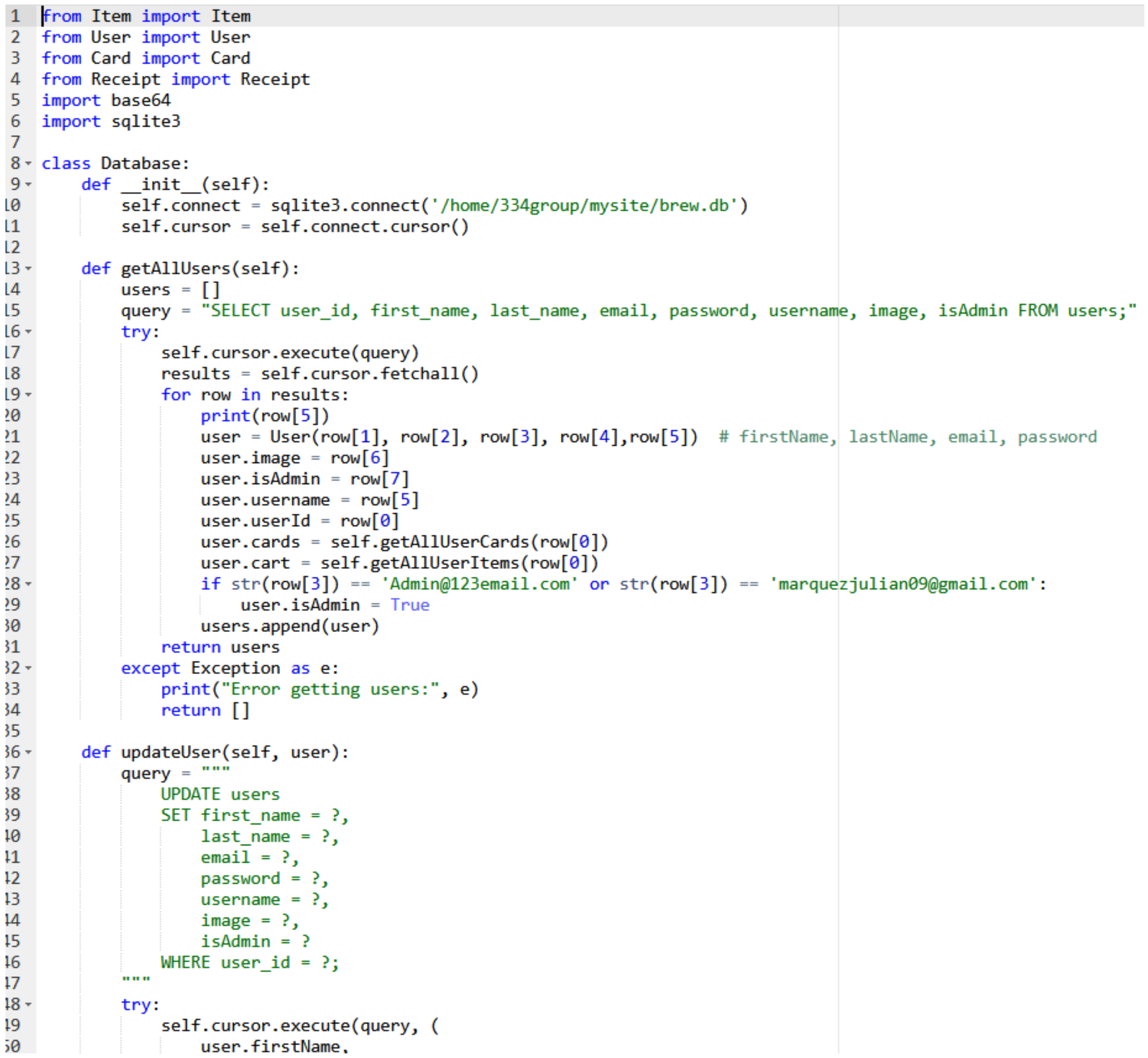
Stage 4 Execution



/home/334group/mysite/app.py

```python
from flask import Flask, flash, request, render_template, session, jsonify, redirect, url_for
from flask_mail import Mail, Message
from Database import Database
from User import User
from Item import Item
from Card import Card
from Receipt import Receipt
from werkzeug.utils import secure_filename
from datetime import datetime
import pickle
import os
import base64
import sqlite3

# Binary Brew Started By Julian Marquez

app = Flask(__name__)
app.secret_key = 'your_secret_key'
UPLOAD_FOLDER = 'static/uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:////home/334group/mysite/brew.db'

app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USERNAME'] = 'binarybrewcs334@gmail.com'
app.config['MAIL_PASSWORD'] = 'pjprtzefholioigb'
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USE_SSL'] = False

mail = Mail(app)


@app.route('/api', methods=['GET'])#API to return items and prices
def getItemsAndPrices():
    connection = sqlite3.connect("////home/334group/mysite/brew.db")
    if connection:
        print("Connected to SQLite")
    else:
        print("Could not connect to SQLite")
    cursor = connection.cursor()

    statement = "SELECT name, price FROM items"
    cursor.execute(statement)
    objects= {}
    index = 0

    for line in cursor:
        print(line)
        lineData = {"name":line[0], "price":line[1]}
        objects.update({"Object{}".format(index):lineData})
```

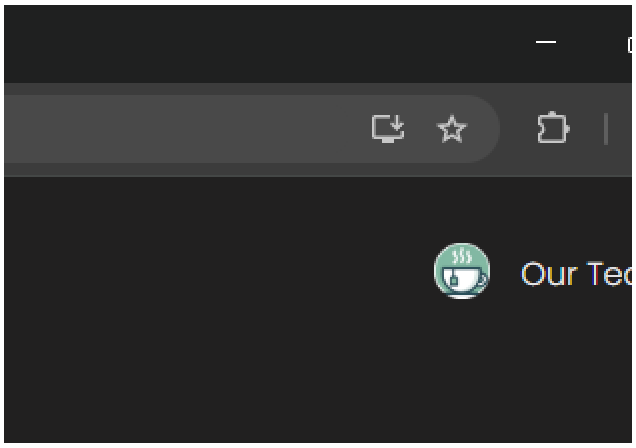
Web app is being implemented using Flask and rendering our html docs as templates.



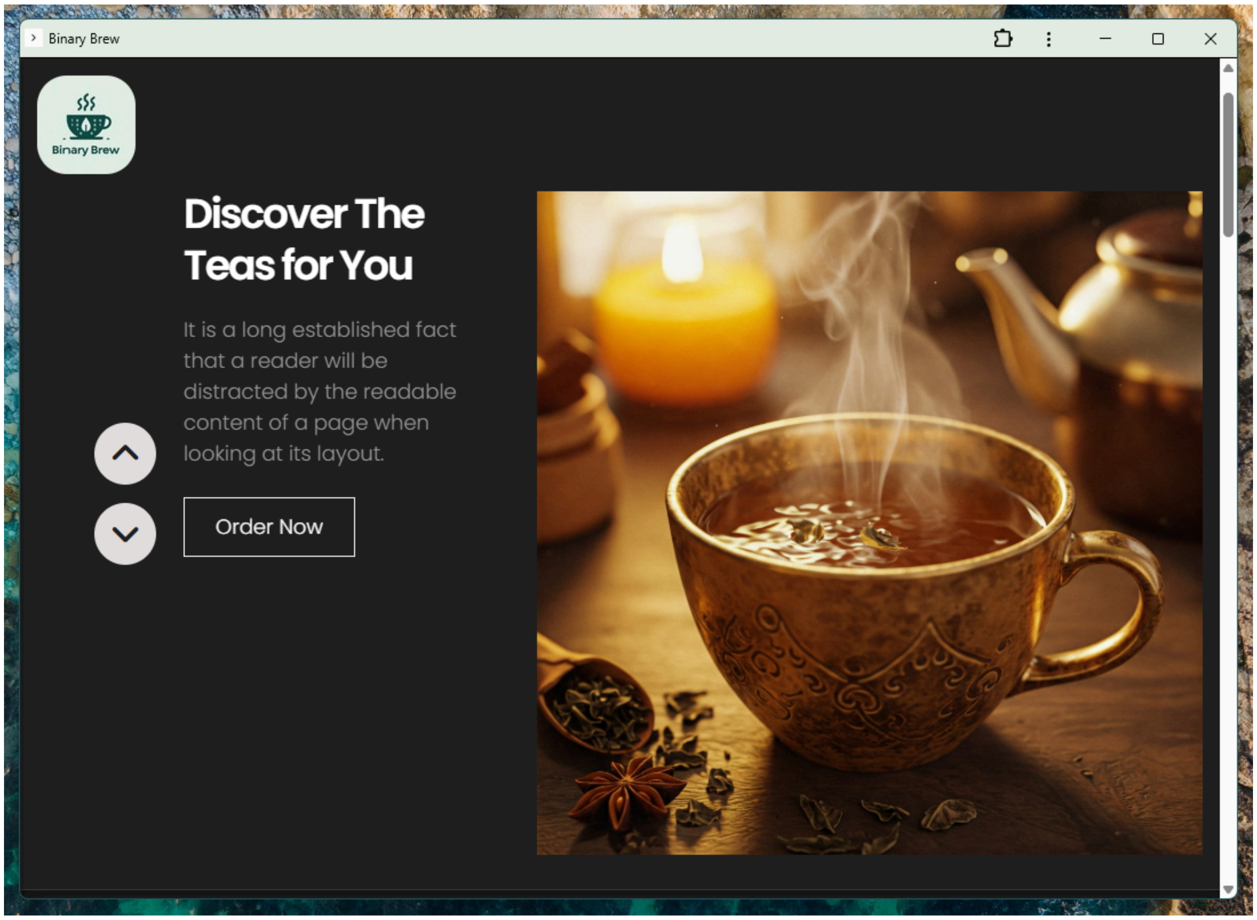
```python
from Item import Item
from User import User
from Card import Card
from Receipt import Receipt
import base64
import sqlite3

class Database:
    def __init__(self):
        self.connect = sqlite3.connect('/home/334group/mysite/brew.db')
        self.cursor = self.connect.cursor()

    def getAllUsers(self):
        users = []
        query = "SELECT user_id, first_name, last_name, email, password, username, image, isAdmin FROM users;"
        try:
            self.cursor.execute(query)
            results = self.cursor.fetchall()
            for row in results:
                print(row[5])
                user = User(row[1], row[2], row[3], row[4],row[5])  # firstName, lastName, email, password
                user.image = row[6]
                user.isAdmin = row[7]
                user.username = row[5]
                user.userId = row[0]
                user.cards = self.getAllUserCards(row[0])
                user.cart = self.getAllUserItems(row[0])
                if str(row[3]) == 'Admin@123email.com' or str(row[3]) == 'marquezjulian09@gmail.com':
                    user.isAdmin = True
                users.append(user)
            return users
        except Exception as e:
            print("Error getting users:", e)
            return []

    def updateUser(self, user):
        query = """
            UPDATE users
            SET first_name = ?,
                last_name = ?,
                email = ?,
                password = ?,
                username = ?,
                image = ?,
                isAdmin = ?
            WHERE user_id = ?;
        """
        try:
            self.cursor.execute(query, (
                user.firstName,
```

The SQLite database handles CRUD operations and stores information about the items sold, user orders, and account data.

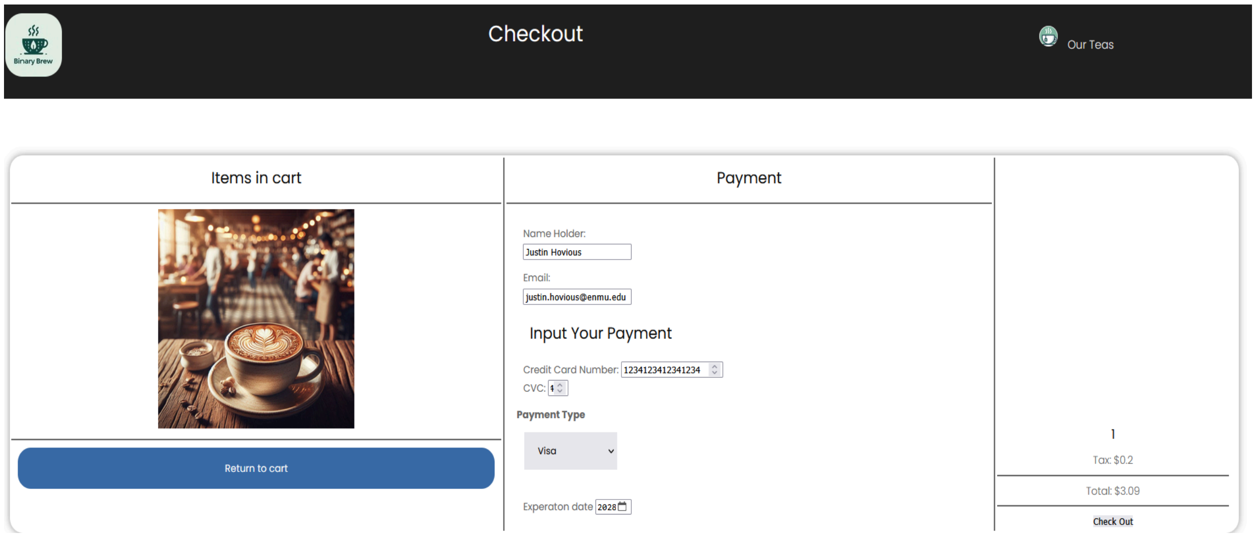
In supported browsers (Chrome & Edge), we have the ability to download our site as a Progressive Web App.



The PWA functions identically to the web app when run in a browser and will function even when offline.


Binary Brew
Binary Brew
Discover The Teas for You
It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout.
Order Now

We have payment and order processing simulated using the database objects



Checkout

Items in cart

Return to cart

Payment

Name Holder: Justin Hovious

Email: justin.hovious@enmu.edu

Input Your Payment

Credit Card Number: 1234123412341234

CVC: 1

Payment Type

Visa

Experaton date 2028

1

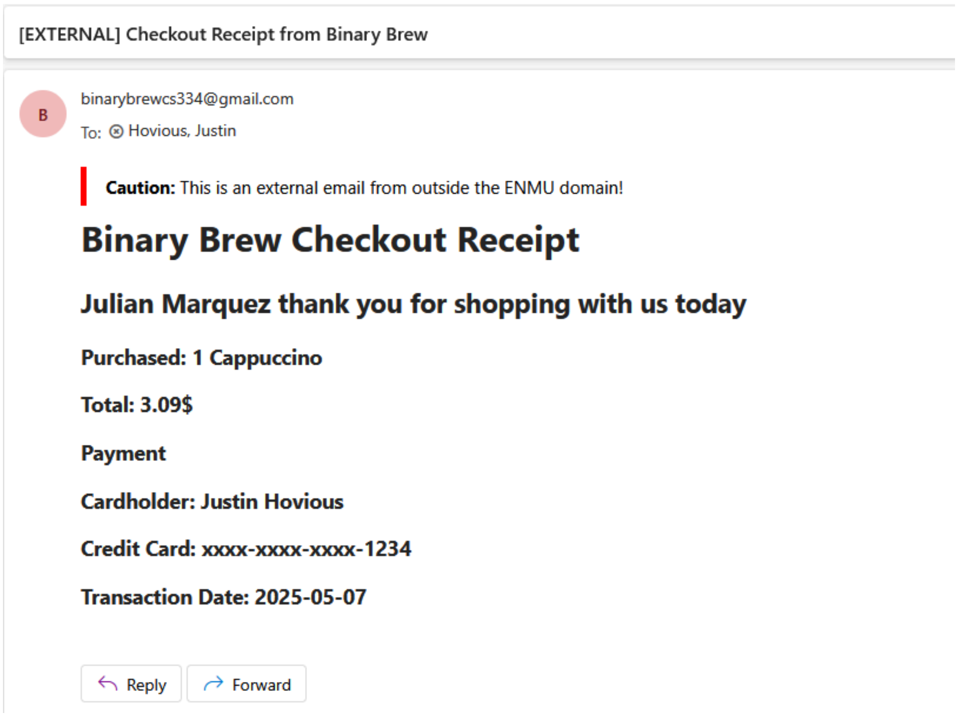
Tax: $0.2

Total: $3.09

Check Out

To speed up the process, users can add and store a payment method from their profile page. Similarly, if the user has payment methods stored/managed by their browser, the payment info can auto-fill.

After completing the checkout successfully, an email is sent to the user via Flask-Email showing their transaction information



[EXTERNAL] Checkout Receipt from Binary Brew

binarybrewcs334@gmail.com

To: Hovious, Justin

Caution: This is an external email from outside the ENMU domain!

Binary Brew Checkout Receipt

Julian Marquez thank you for shopping with us today

Purchased: 1 Cappuccino

Total: 3.09$

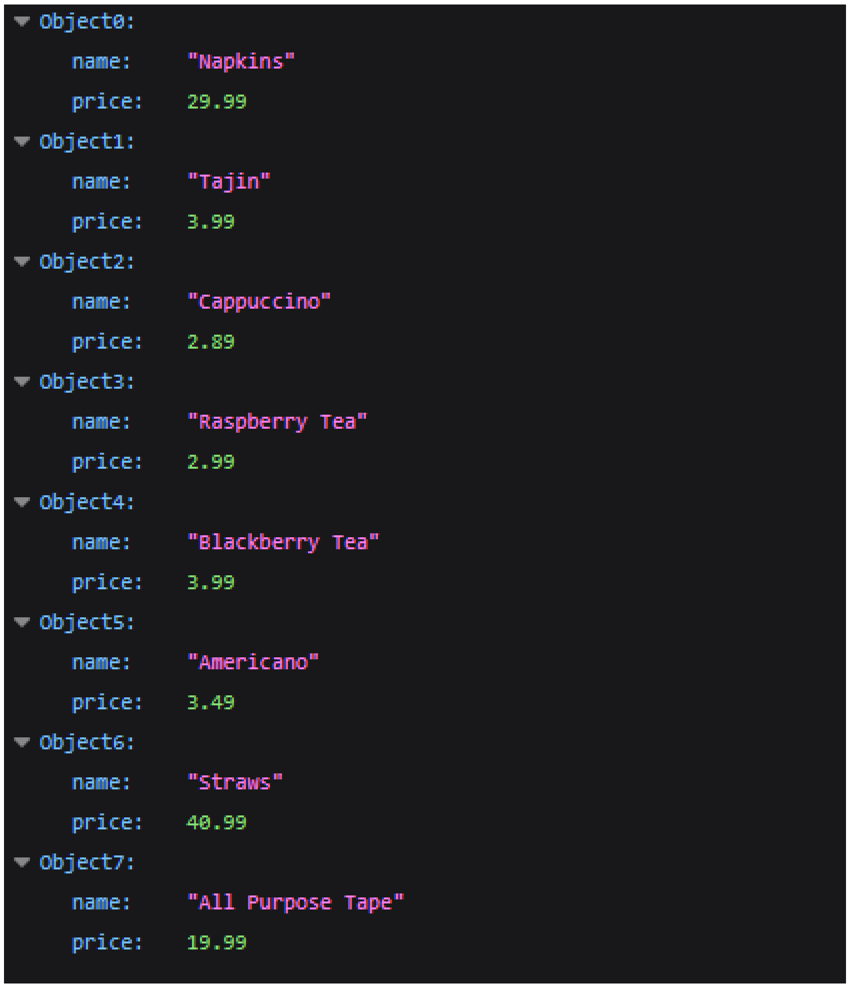
Payment

Cardholder: Justin Hovious

Credit Card: xxxx-xxxx-xxxx-1234

Transaction Date: 2025-05-07

Typing our website followed by /api will allow you to access our public API, which returns a JSON file with items and prices.
https://334group.pythonanywhere.com/api



```
▼ Object0:
    name:    "Napkins"
    price:   29.99
▼ Object1:
    name:    "Tajin"
    price:   3.99
▼ Object2:
    name:    "Cappuccino"
    price:   2.89
▼ Object3:
    name:    "Raspberry Tea"
    price:   2.99
▼ Object4:
    name:    "Blackberry Tea"
    price:   3.99
▼ Object5:
    name:    "Americano"
    price:   3.49
▼ Object6:
    name:    "Straws"
    price:   40.99
▼ Object7:
    name:    "All Purpose Tape"
    price:   19.99
```