

CAB COMPANY WEB APPLICATION

About the Deployment

This document provides a snapshot of the production processes involved in creating an interactive Cab platform utilizing Flask, a robust web framework for Python. It serves as a continuation of the previously offered insights into the performance of Pink Cab and Yellow Cab companies. The initial analysis delved into cab transactions, holidays, and customer demographics, revealing key metrics such as the number of transactions, transactions on holidays, number of customers, cost per kilometer, price charged per kilometer, and average profit per kilometer. Additionally, it included a one-year profit forecast for both Pink Cab and Yellow Cab, initially implemented with Facebook Prophet and later with Scikit-Learn due to additional features that needed consideration.

Key Features:

Calculate Price and Profit: Users can input the distance of a trip (with plans for automation improvement) and select the cab company (Pink Cab or Yellow Cab) to calculate the trip's price and profit.

How to Use:

Visit the home page to access the "Calculate Price and Profit" feature.

Input the distance of your trip in kilometers and select the cab company (Pink Cab or Yellow Cab).

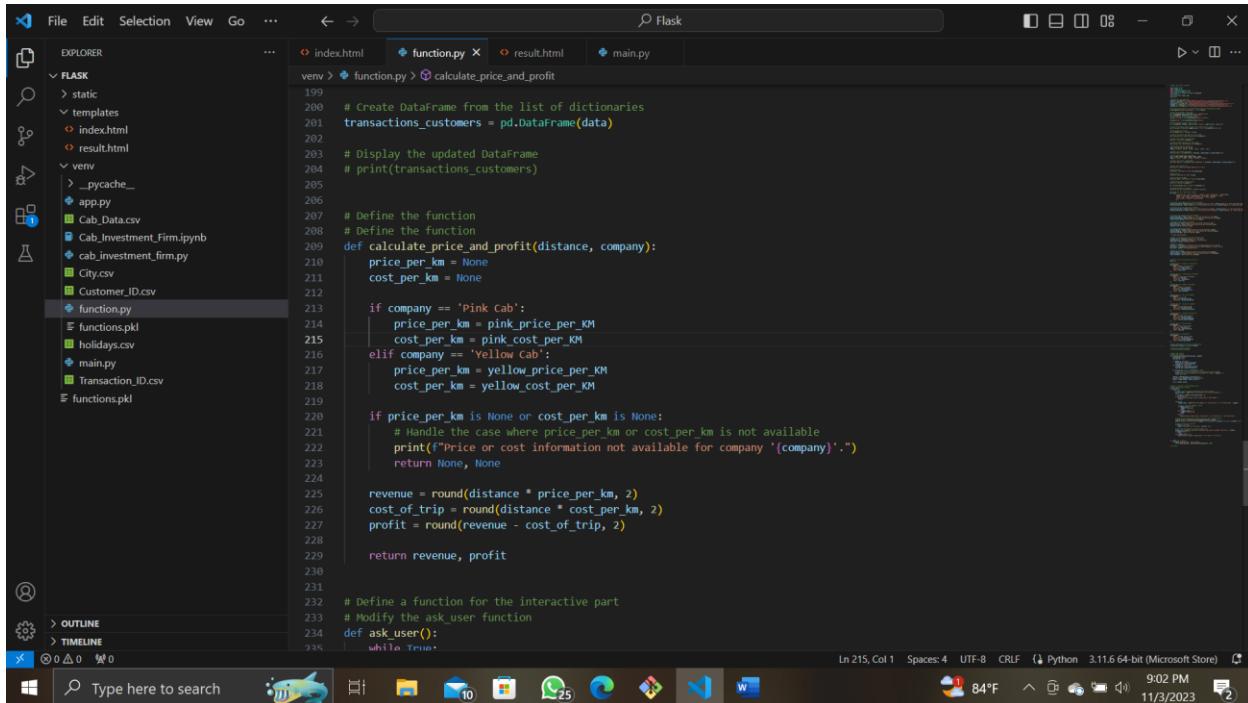
Click "Calculate" to view the price and profit for the specified trip.

Important Note:

The application is designed for educational and analytical purposes, forming part of the tasks undertaken during a data science internship at Data Glacier. It utilizes a dataset containing information on cab transactions, customer details, and holiday data. The data has undergone preprocessing and analysis to provide meaningful insights into the operations of the cab companies.

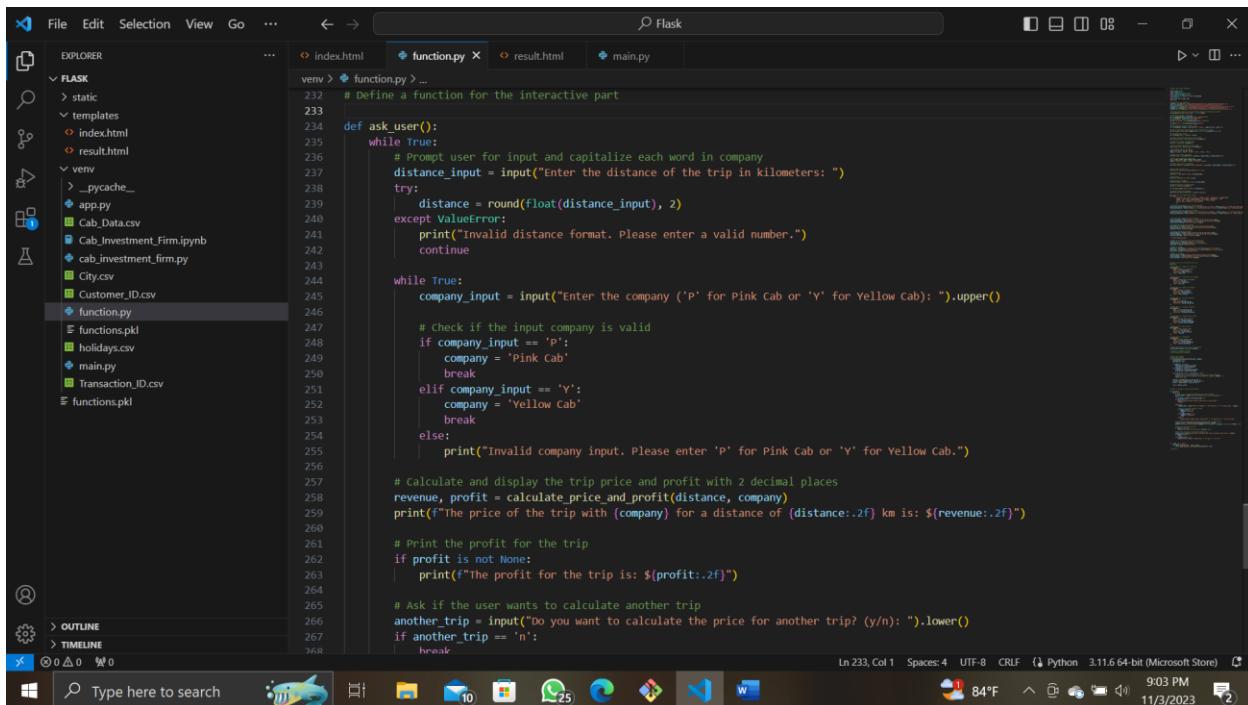
CAB COMPANY WEB APPLICATION

Function Definition:



The screenshot shows the Microsoft Visual Studio Code interface with the Python file 'function.py' open. The code defines a function 'calculate_price_and_profit' that takes 'distance' and 'company' as parameters. It handles two companies: 'Pink Cab' and 'Yellow Cab', setting their respective price and cost per kilometer. If no company is specified, it prints a warning and returns None. The function then calculates revenue and profit based on the distance and company. A comment indicates a modification to the 'ask_user' function.

```
199
200     # Create DataFrame from the list of dictionaries
201     transactions_customers = pd.DataFrame(data)
202
203     # Display the updated DataFrame
204     # print(transactions_customers)
205
206
207     # Define the function
208     # Define the function
209     def calculate_price_and_profit(distance, company):
210         price_per_km = None
211         cost_per_km = None
212
213         if company == 'Pink Cab':
214             price_per_km = pink_price_per_KM
215             cost_per_km = pink_cost_per_KM
216         elif company == 'Yellow Cab':
217             price_per_km = yellow_price_per_KM
218             cost_per_km = yellow_cost_per_KM
219
220         if price_per_km is None or cost_per_km is None:
221             # Handle the case where price_per_km or cost_per_km is not available
222             print(f"Price or cost information not available for company '{company}'")
223             return None, None
224
225         revenue = round(distance * price_per_km, 2)
226         cost_of_trip = round(distance * cost_per_km, 2)
227         profit = round(revenue - cost_of_trip, 2)
228
229         return revenue, profit
230
231
232     # Define a function for the interactive part
233     # Modify the ask_user function
234     def ask_user():
235         while True:
```



The screenshot shows the Microsoft Visual Studio Code interface with the Python file 'function.py' open. The code has been modified to include an 'ask_user' function. This function prompts the user for the distance of the trip in kilometers, validates the input, and then asks for the company ('P' for Pink Cab or 'Y' for Yellow Cab). It then calculates and prints the trip price and profit, and asks if the user wants to calculate another trip.

```
232     # Define a function for the interactive part
233
234     def ask_user():
235         while True:
236             # Prompt user for input and capitalize each word in company
237             distance_input = input("Enter the distance of the trip in kilometers: ")
238             try:
239                 distance = round(float(distance_input), 2)
240             except ValueError:
241                 print("Invalid distance format. Please enter a valid number.")
242             continue
243
244             while True:
245                 company_input = input("Enter the company ('P' for Pink Cab or 'Y' for Yellow Cab): ").upper()
246
247                 # Check if the input company is valid
248                 if company_input == 'P':
249                     company = 'Pink Cab'
250                     break
251                 elif company_input == 'Y':
252                     company = 'Yellow Cab'
253                     break
254                 else:
255                     print("Invalid company input. Please enter 'P' for Pink Cab or 'Y' for Yellow Cab.")
256
257                 # Calculate and display the trip price and profit with 2 decimal places
258                 revenue, profit = calculate_price_and_profit(distance, company)
259                 print(f"The price of the trip with {company} for a distance of {distance:.2f} km is: ${revenue:.2f}")
260
261                 # Print the profit for the trip
262                 if profit is not None:
263                     print(f"The profit for the trip is: ${profit:.2f}")
264
265                 # Ask if the user wants to calculate another trip
266                 another_trip = input("Do you want to calculate the price for another trip? (y/n): ").lower()
267                 if another_trip == 'n':
268                     break
```

CAB COMPANY WEB APPLICATION

This screenshot shows a code editor interface with two tabs open: `function.py` and `main.py`. The `function.py` tab contains Python code for a command-line application that prompts the user for a company ('P' for Pink Cab or 'Y' for Yellow Cab), calculates trip price and profit, and asks if they want to calculate another trip. The `main.py` tab contains the Flask application code, defining routes for the home page and a calculation endpoint, and setting up the template folder.

```
function.py
venv > function.py > ...
245     company_input = input("Enter the company ('P' for Pink Cab or 'Y' for Yellow Cab): ").upper()
246
247     # Check if the input company is valid
248     if company_input == 'P':
249         company = 'Pink Cab'
250         break
251     elif company_input == 'Y':
252         company = 'Yellow Cab'
253         break
254     else:
255         print("Invalid company input. Please enter 'P' for Pink Cab or 'Y' for Yellow Cab.")
256
257     # Calculate and display the trip price and profit with 2 decimal places
258     revenue, profit = calculate_price_and_profit(distance, company)
259     print(f"The price of the trip with {company} for a distance of {distance:.2f} km is: ${revenue:.2f}")
260
261     # Print the profit for the trip
262     if profit is not None:
263         print(f"The profit for the trip is: ${profit:.2f}")
264
265     # Ask if the user wants to calculate another trip
266     another_trip = input("Do you want to calculate the price for another trip? (y/n): ").lower()
267     if another_trip == 'n':
268         break
269     elif another_trip != 'y':
270         print("Invalid response. Please enter 'y' for yes or 'n' for no.")
271
272
273     if __name__ == "__main__":
274         with open('functions.pkl', 'wb') as file:
275             pickle.dump((ask_user, calculate_price_and_profit), file)
276
277     # ask_user()
278
279
main.py
venv > main.py > ...
1  # main.py
2  from flask import Flask, render_template, request
3  import pickle
4  from function import ask_user, calculate_price_and_profit
5
6  app = Flask(__name__, template_folder='C:/Users/HP/Desktop/dataGlaciers_week2/Flask/templates')
7
8  @app.route('/')
9  def index():
10     return render_template('index.html')
11
12  @app.route('/calculate', methods=['POST'])
13  def calculate():
14     distance = float(request.form['distance'])
15     company = request.form['company']
16     revenue, profit = calculate_price_and_profit(distance, company)
17     return render_template('result.html', revenue=revenue, profit=profit)
18
19 if __name__ == '__main__':
20     app.run(debug=True)
```

Main App Snapshot:

This screenshot shows a code editor interface with the `main.py` tab selected. It displays the same code as the previous screenshot, which defines a Flask application with routes for the home page and a calculation endpoint, and sets up the template folder.

```
main.py
venv > main.py > ...
1  # main.py
2  from flask import Flask, render_template, request
3  import pickle
4  from function import ask_user, calculate_price_and_profit
5
6  app = Flask(__name__, template_folder='C:/Users/HP/Desktop/dataGlaciers_week2/Flask/templates')
7
8  @app.route('/')
9  def index():
10     return render_template('index.html')
11
12  @app.route('/calculate', methods=['POST'])
13  def calculate():
14     distance = float(request.form['distance'])
15     company = request.form['company']
16     revenue, profit = calculate_price_and_profit(distance, company)
17     return render_template('result.html', revenue=revenue, profit=profit)
18
19 if __name__ == '__main__':
20     app.run(debug=True)
```

CAB COMPANY WEB APPLICATION

Homepage Snapshot:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Calculate Price and Profit</title>
</head>
<body>
    <h1>Calculate Price and Profit</h1>
    <form action="/calculate" method="post">
        <label for="distance">Enter distance:</label>
        <input type="number" id="distance" name="distance" required>
        <br>
        <label for="company">Enter company ('P' for Pink Cab or 'Y' for Yellow Cab):</label>
        <input type="text" id="company" name="company" required>
        <br>
        <input type="submit" value="Calculate">
    </form>
</body>
</html>
```

Result Page Snapshot:

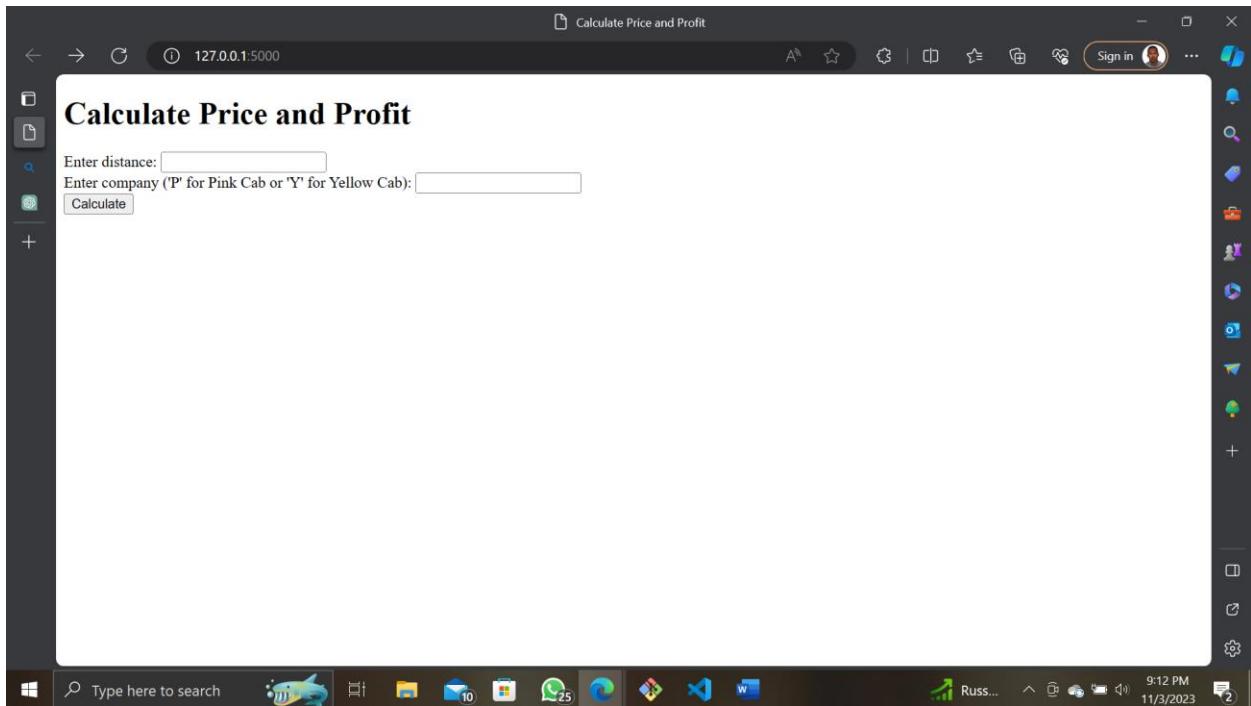
The screenshot shows a code editor interface with the following details:

- File Explorer:** On the left, it lists project files including `index.html`, `function.py`, `result.html` (which is currently selected), and `main.py`. It also shows a `templates` folder containing `index.html` and `result.html`.
- Code Editor:** The main area displays the content of `result.html`. The code is an HTML template with embedded Python logic using Jinja syntax. It includes a `<h1>Result</h1>` heading, a conditional block to calculate price and profit, and a fallback message if input is invalid.
- Bottom Status Bar:** Shows "Ln 1, Col 1" and "Spaces:4" along with other standard status icons.
- Taskbar:** At the bottom, there are icons for File Explorer, Task View, Mail, File Explorer, and a browser window.

```
index.html function.py result.html main.py
templates > result.html ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Result</title>
7  </head>
8  <body>
9      <h1>Result</h1>
10     {% if revenue is defined and profit is defined %}
11         <p>The price of the trip is: ${{ revenue }}</p>
12         <p>The profit for the trip is: ${{ profit }}</p>
13     {% else %}
14         <p>Unable to calculate the price and profit. Please check your input.</p>
15     {% endif %}
16
17  </body>
18  </html>
19
```

CAB COMPANY WEB APPLICATION

Homepage View:



Name: Tomisin Adeniyi

Batch code: LISUM26

Submission date: November 3rd, 2023