PLATFORM SYSTEM OJEK ONLINE MENGGUNAKAN PROGRAM PYTHON DAN DIJKSTRA'S UNTUK OPTIMALISASI

(Rata tengah, Tebal, Arial 14, Maksimum 15 Kata)

Dustin Pranata Wangsawidjaja, 5803024021
Geraldo Lauvanno, 5803024020

Abstrak (200 kata)

Platform sistem ojek online adalah aplikasi atau program yang dirancang khusus untuk mencocokkan penumpang dengan pengemudi berdasarkan efisiensi. Tujuan utama dari sistem ini adalah untuk mencocokkan setiap pesanan dengan pengemudi yang paling sesuai, meminimalkan jarak tempuh antara pengemudi dan penumpang, serta menghindari konflik penjadwalan di mana satu pengemudi hanya diperbolehkan untuk memiliki satu pesanan yang ditugaskan, dan bukan beberapa pesanan secara bersamaan. Kendala utama yang dipertimbangkan dalam sistem dan perhitungan adalah ketersediaan pengemudi yang terbatas, jam kerja mereka yang terbatas (setiap pengemudi hanya diizinkan untuk bekerja 8 jam sehari), dan kapasitas penumpang yang berbeda-beda (beberapa pengemudi dapat mengangkut satu penumpang sementara pengemudi lainnya dapat menangani banyak penumpang per perjalanan). Sistem ini akan digunakan untuk membantu menyederhanakan proses pemesanan pada sistem ojek online dan juga memberikan transparansi perhitungan, menunjukkan kepada pengguna bagaimana efisiensi diputuskan dan dihitung, memastikan bahwa pengemudi terbaik digunakan untuk perjalanan tersebut.

Kata kunci: *Sistem Ojek Online, Efisiensi, Minimisasi Jarak Tempuh, Penjadwalan Pengemudi, Kapasitas Penumpang*

Abstract (200 words)

The online motorcycle taxi system is an application or program that is specifically designed to match passengers with drivers based on efficiency. The primary objective of this system is to match each order with the most suitable driver, minimizing the travel distance between drivers and passengers, and avoiding any scheduling conflicts where a single driver is only allowed to have one assigned order rather than multiple simultaneous orders. The key constraints considered in the system and calculation is the limited availability of drivers, their restricted working hours (each driver is allowed only to work 8 hours a day), and varying passenger capacities (some drivers can carry one passenger while others can handle multiple passengers per trip). The system will be used to help simplify the process of ordering on an ojek online system while also providing the transparency of calculations, showing the user how efficiency is decided and calculated, ensuring that the best driver is used for that journey.

Keywords*: Online Motorcycle, Efficiency, Travel Distance Minimization, Scheduling Conflicts, Passenger Capacity*

1. Pendahuluan

Peningkatan permintaan untuk layanan transportasi on-demand telah berkembang sejak diperkenalkan ke pasar. Platform ojek online adalah salah satu layanan yang telah diperkenalkan juga, memberikan peran penting dalam layanan transportasi. Namun, masalah yang dapat terjadi adalah mencocokkan penumpang dengan pengemudi yang tersedia secara efisien, sehingga sulit untuk sepenuhnya mengoptimalkan dan menghemat waktu. Program ini bertujuan untuk menemukan solusi untuk program penjadwalan pengemudi-penumpang, sekaligus memberikan transparansi tentang bagaimana efisiensi dihitung, memastikan pengguna bahwa mereka memang mendapatkan perjalanan yang paling efisien.

Tantangan utamanya terletak pada keseimbangan berbagai faktor dinamis: pengemudi tersebar secara geografis, permintaan penumpang berfluktuasi tanpa bisa diprediksi, dan terdapat kendala yang ketat terkait ketersediaan pengemudi (shift kerja 8 jam), penugasan satu pesanan, dan kapasitas penumpang yang berbeda-beda per kendaraan. Pendekatan tradisional yang menggunakan sistem siapa cepat dia dapat sering kali menyebabkan pemanfaatan sumber daya yang kurang

optimal, sehingga menyoroti perlunya sistem penjadwalan yang cerdas yang dapat mengevaluasi variabel waktu nyata termasuk jarak, kondisi lalu lintas, dan beban kerja pengemudi.

Penelitian ini mengusulkan pendekatan sistematis untuk mengoptimalkan alokasi pengemudi melalui solusi algoritmik yang memprioritaskan:

1. Jarak tempuh minimal antara pengemudi dan penumpang
2. Distribusi pesanan yang adil di antara pengemudi yang tersedia
3. Proses pengambilan keputusan yang transparan untuk kepercayaan pengguna.

Dengan mengatasi dimensi-dimensi tersebut, platform dapat meningkatkan kualitas layanan, mengurangi waktu menganggur pengemudi, dan meningkatkan efisiensi sistem secara keseluruhan-faktor-faktor kunci dalam menjaga daya saing di industri pemesanan kendaraan yang berkembang pesat. Bagian selanjutnya akan menganalisis persyaratan teknis dari masalah tersebut, mengevaluasi metodologi potensial, dan mengusulkan kerangka kerja penjadwalan yang layak.

Selain itu, sistem ini juga dilengkapi dengan Graphical User Interface (GUI) yang intuitif yang dirancang untuk interaksi yang mulus. GUI ini menyediakan visualisasi real-time dari lokasi pengemudi, estimasi waktu kedatangan, dan penugasan pesanan, sehingga penumpang dapat melacak perjalanan mereka dengan mudah. Untuk pengemudi, antarmuka menampilkan pesanan yang tertunda dengan indikator prioritas berdasarkan kedekatan dan kapasitas penumpang, serta sisa jam kerja mereka untuk mencegah penjadwalan yang berlebihan. Peta interaktif, peringatan dengan kode warna untuk pesanan yang mendesak, dan fitur penerimaan/penolakan dengan sekali sentuh menyederhanakan pengambilan keputusan, sehingga mengurangi waktu respons. Selain itu, GUI mencakup fitur transparansi yang menunjukkan kepada pengguna bagaimana pengemudi dicocokkan dengan permintaan mereka, sehingga membangun kepercayaan pada keadilan platform. Dengan menjembatani penjadwalan algoritma yang rumit dengan desain yang ramah pengguna, GUI mengubah pengoptimalan teknis menjadi peningkatan layanan yang nyata.

2. Metode Pembuatan

Program ini dirancang untuk mengoptimalkan pencocokan pengemudi dan penumpang untuk platform ojek online dengan meminimalkan jarak tempuh. Proses pengembangan terdiri dari tiga tahap: Perancangan Logika Program, Perancangan GUI, dan Implementasi Algoritma.

2.1 Perancangan Logika Program:

Logika inti utama dari Sistem Penjadwalan Pengemudi Ojek adalah sebagai berikut:

- **Input Data:**
  - Pengemudi mendaftarkan lokasi mereka secara real-time, jenis kendaraan (kapasitas penumpang tunggal/banyak), dan jam kerja yang tersedia.
  - Penumpang mengajukan permintaan perjalanan dengan lokasi penjemputan/pengantaran dan jenis kendaraan yang diinginkan.
- **Perhitungan Kedekatan:**
  - Sistem menghitung jarak Euclidean antara pengemudi dan penumpang menggunakan data koordinat, dengan kondisi lalu lintas sebagai faktor pembobotan opsional.
- **Optimalisasi Pencocokan:**
  - Pengemudi dicocokkan dengan penumpang berdasarkan jarak terpendek, sisa jam kerja, dan kecocokan kapasitas penumpang.
- **Pencegahan Konflik:**
  - Sistem menerapkan batasan satu pesanan per pengemudi secara real-time dengan menandai pengemudi yang cocok sebagai “tidak tersedia” hingga pesanan selesai.
- **Presentasi Hasil:**
  - Menampilkan pasangan yang cocok untuk penumpang dan pengemudi melalui notifikasi, termasuk ETA dan detail rute.
- **Kalkulasi:**
  - Untuk menghitung efisiensi pengemudi agar sesuai dengan kebutuhan penumpang, kami akan membuat rumus arbitrer kami sendiri dengan parameter kami sendiri untuk mendapatkan skor yang akan menentukan efisiensi pengemudi
  - Contoh rumus adalah:

```python
# Worth calculation to find the most efficient driver
def worth_calculation(pickup_distance, total_distance, max_passengers, passenger_count):
    # Lower score is better
    # Factors:
    # 1. Pickup distance (weighted heavily as this is immediate customer wait time)
    # 2. Overall trip efficiency
    # 3. Vehicle utilization (how well the vehicle capacity is utilized)

    # Penalize long pickup distances
    pickup_score = pickup_distance * 3

    # Penalize overall inefficient routes
    trip_score = total_distance

    # Penalize underutilized vehicles (sending a 6-seater for 1 person)
    # but don't penalize if it's a good fit
    capacity_utilization = passenger_count / max_passengers
    if capacity_utilization < 0.5:
        capacity_score = (max_passengers - passenger_count) * 5
    else:
        capacity_score = 0

    # Lower is better
    total_score = pickup_score + trip_score + capacity_score
    return -total_score  # Return negative so higher is better in the selection algorithm
```

Gambar 1: Rumus untuk menghitung efisiensi

- Beginilah cara kerja rumusnya:

$$T = x + y + z$$
$$x = a * 3$$
$$y = b$$
$$c = (d / e) * 100$$
$$if\ c \leq 50\% \ --> \ z = (d - e) * 5$$
$$else \ --> \ z = 0$$

- Semakin rendah skor total, semakin efisien pengemudi ditentukan oleh algoritme
-

2.2 Perancangan GUI:

Antarmuka dibangun dengan matplotlib untuk visualisasi geografis dan Tkinter untuk interaksi pengguna:

- **Visualisasi Peta (matplotlib):**
  - Memetakan pengemudi dan penumpang sebagai titik-titik sebaran pada kisi 2D, dengan kode warna:
  - **Titik Hijau:** Pengemudi yang tersedia
  - **Titik Merah:** Pengemudi yang tidak tersedia
  - **Titik Pink:** Point destinasi
  - **Garis Biru:** Rute dari pengemudi ke penumpang untuk penjemputan
  - **Garis Hijau:** Rute pengemudi-penumpang yang cocok

  - Pembaruan secara real-time selama mode simulasi.
  - Tabel yang menyediakan informasi pengemudi seperti di mana pengemudi berada, berapa jam mereka telah bekerja, kapasitas penumpang mereka.

- **Panel Kontrol Tkinter:**
- **Bagian Masukan:**
  - Kolom teks untuk penjemputan di lokasi, penjemputan di tempat tujuan, dan kapasitas penumpang.
  - Menu tarik-turun untuk pemilihan jenis kendaraan dan jam kerja.

- **Bagian Keluaran:**
  - Menampilkan perhitungan berdasarkan jarak penjemputan, jarak tempuh, total jarak tempuh, perhitungan waktu dengan asumsi dan faktor seperti lalu lintas, Daftar pengemudi yang tersedia, dengan pengemudi yang paling efisien di bagian paling atas.
- **Kontrol Simulasi:**
  - "Show Locations" menampilkan semua lokasi dalam program
  - "Randomize Map" mengacak semua jarak lokasi bersama dengan posisi pengemudi
  - "Analyze Driver Efficiency" memunculkan jendela yang menunjukkan perhitungan untuk menunjukkan pengemudi yang paling efisien dalam situasi tersebut
  - "Refresh Driver Data" untuk memperbarui semua data di dalamnya.

2.3 Implementasi Algoritma:

- **Algoritma Dijkstra:**
  - Menghitung jalur terpendek sumber tunggal untuk pencocokan
  - Menggunakan antrian prioritas (min-heap) untuk pemilihan simpul yang efisien
  - Mengembalikan nilai jarak dan jalur lengkap

```python
# Dijikstra's Algorithm
def dijkstra(graph, start, end):
    queue = [(0, start, [])]  # (cost, node, path)
    visited = {}
    while queue:
        (cost, node, path) = heapq.heappop(queue)
        if node in visited:
            continue
        new_path = path + [node]
        visited[node] = (cost, new_path)
        if node == end:
            return cost, new_path
        for neighbor, weight in graph[node].items():
            if neighbor not in visited:
                heapq.heappush(queue, (cost + weight, neighbor, new_path))
    return float('inf'), []
```

Gambar 2: Implementasi Algoritma Dijkstra

- **Struktur Data Graf:**
  - Representasi daftar kedekatan dari jaringan jalan
  - Penyimpanan berbasis kamus untuk objek-objek simpul dan bobot-bobot sisi
- **Rekonstruksi Jalur:**
  - Melacak node pendahulu untuk membangun kembali jalur terpendek
  - Menyimpan informasi rute lengkap untuk visualisasi

3. Hasil dan Pembahasan

Program ojek ini dikembangkan dengan menggunakan bahasa pemrograman Python dan memanfaatkan TKinter untuk membuat antarmuka pengguna grafis untuk berinteraksi dengan pengguna. Beberapa fungsi dari program ini antara lain:

3.1. Tampilan Program

3.1.1 Tab 1 - Taxi Booking System:

Di tab pertama dari **"Taxi Booking System"**, terdapat banyak bagian penting di jendela. Bagian-bagian tersebut meliputi:

- Bagian input bernama **"booking details"** yang memungkinkan pengguna untuk memasukkan lokasi penjemputan, tujuan, dan jumlah penumpang yang mereka miliki.
  Sebuah tabel berjudul **"Driver Information"** yang mencakup nama pengemudi, kapasitas maksimum penumpang yang dapat mereka angkut, lokasi mereka, ketersediaan mereka, dan jumlah menit kerja mereka. Sebuah tombol berlabel **"Refresh Driver Data"** digunakan untuk menyegarkan tabel dan memperbarui isinya.
- Di bawah bagian input, terdapat tiga tombol yang diberi label **"Show Locations"**, **"Randomize Map"**, dan **"Analyze Driver Efficiency"**. Tujuan dari masing-masing tombol adalah:
  - Menampilkan semua lokasi di peta.

    - Mengacak posisi dan lokasi di peta, termasuk pengemudi.
    - Membuka tab lain yang digunakan setelah memasukkan semua parameter yang diperlukan untuk menentukan pengemudi yang paling sesuai.
- Bagian hasil yang menampilkan pengemudi terpilih beserta semua informasi yang diperlukan.

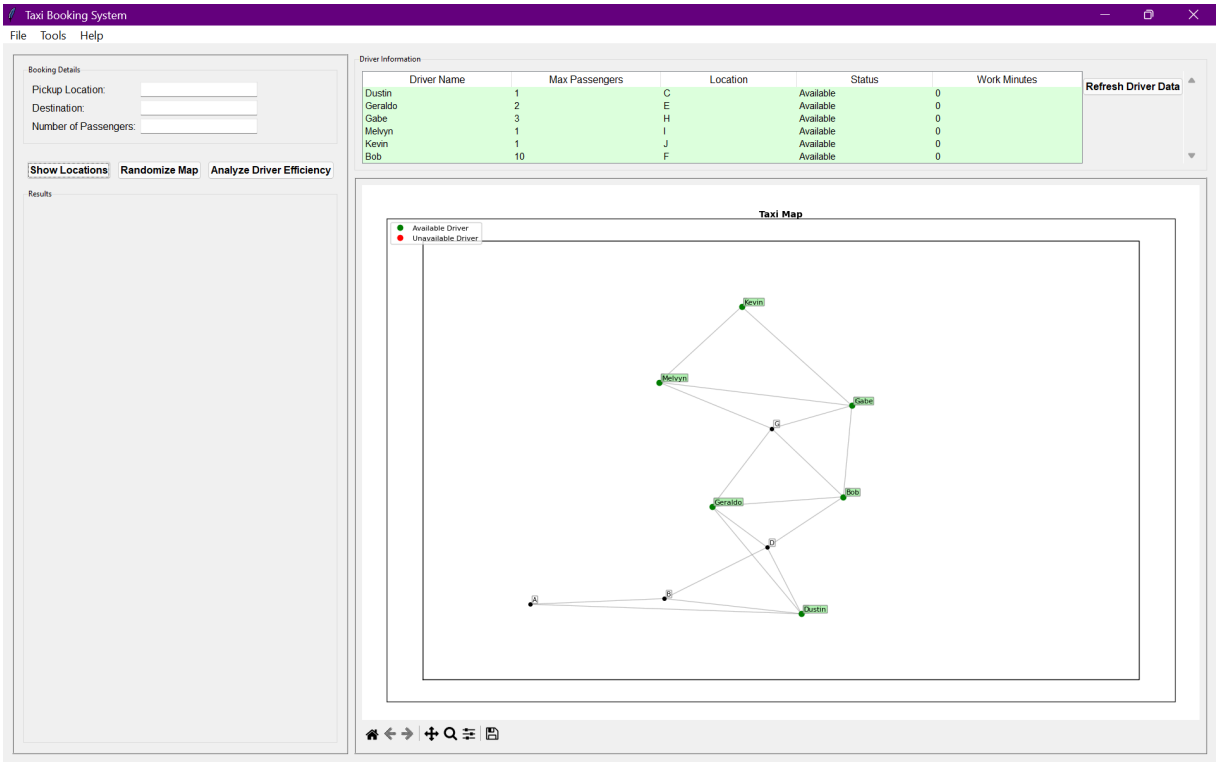
3.1.2 Tab 2 - Driver Efficiency Analysis:

Di tab kedua, terdapat output berdasarkan parameter yang dimasukkan. Tab ini menampilkan:

- **Trip Parameters** yang dimasukkan oleh pengguna.
- **Time Calculation Assumptions** yang digunakan untuk memberikan perkiraan waktu perjalanan.
- Berdasarkan parameter yang dimasukkan, sistem akan menampilkan semua hasil dan menunjukkan pengemudi yang paling efisien di antara yang tersedia. Dalam tabel, informasi yang ditampilkan mencakup:
    - **Current Location**
    - **Max Passengers**
    - **Distance to Pickup**
    - **Distance to Destination**
    - **Total Trip Distance**
    - **Estimated Travel Times** beserta perhitungannya
    - Rincian perhitungan efisiensi yang telah kami tentukan secara arbitrer
- Terdapat sebuah tombol berlabel **"Select Most Efficient Driver"**, yang akan secara otomatis memilih pengemudi yang paling efisien.

3.2. Hasil Program

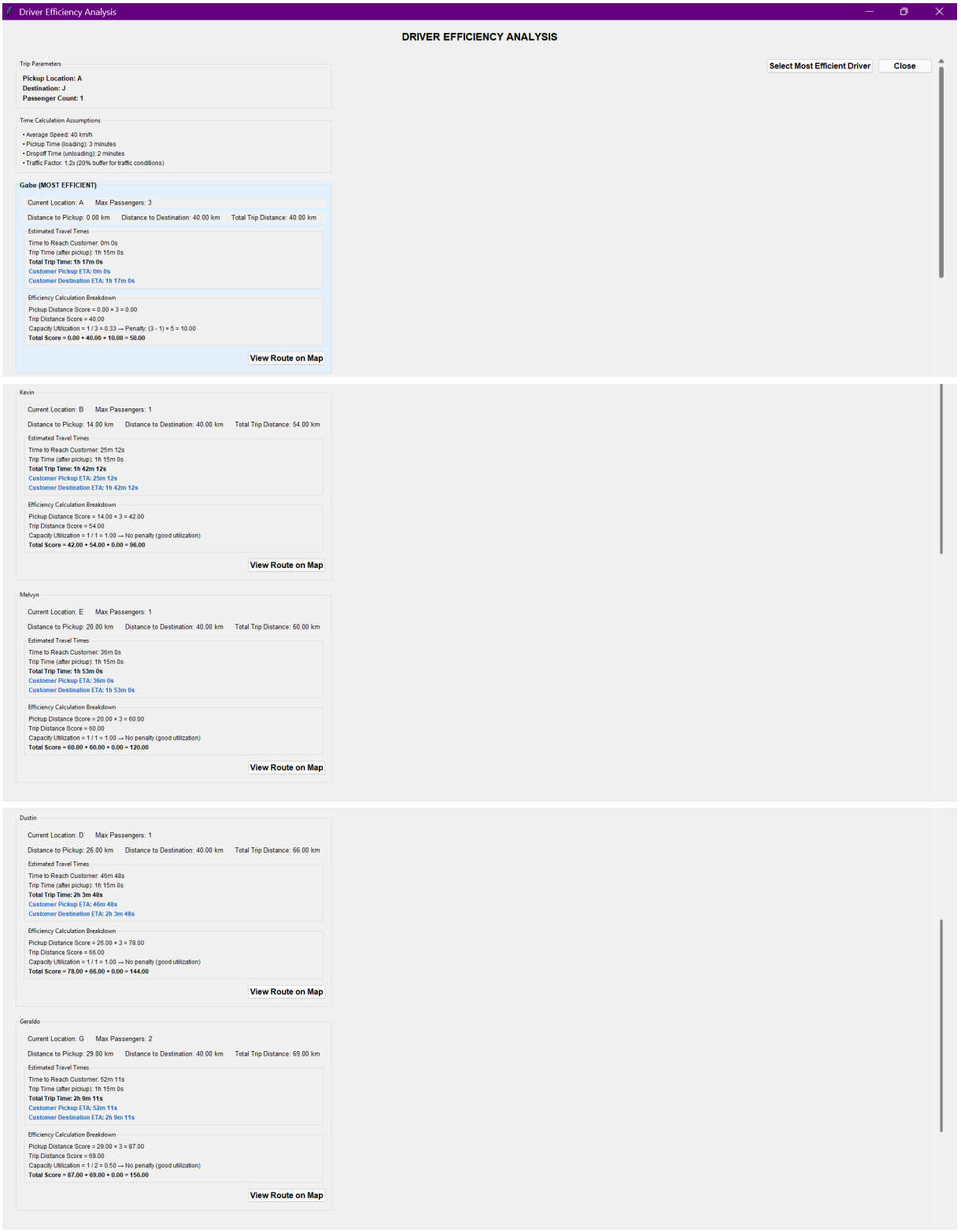
Setiap paragraf hendaknya terdiri dari satu kalimat inti dan beberapa kalimat penjelas. Pembahasan sebaiknya diberikan secara sistematis dan memberikan informasi tentang bagaimana penulis melakukan, baik berhubungan dengan data, metode dan tahapan didalam melakukan pembuatan aplikasi tersebut.

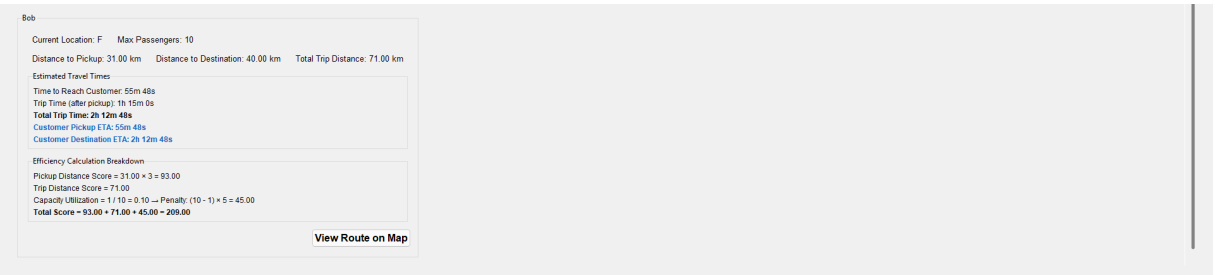


Gambar 3 : Tampilan Awal Aplikasi Ojek

Berikut adalah tampilan awal pada saat membuka aplikasi Ojek, terdapat **Booking Details** untuk melakukan pesanan / order, **Driver Information** untuk menampilkan status dan informasi

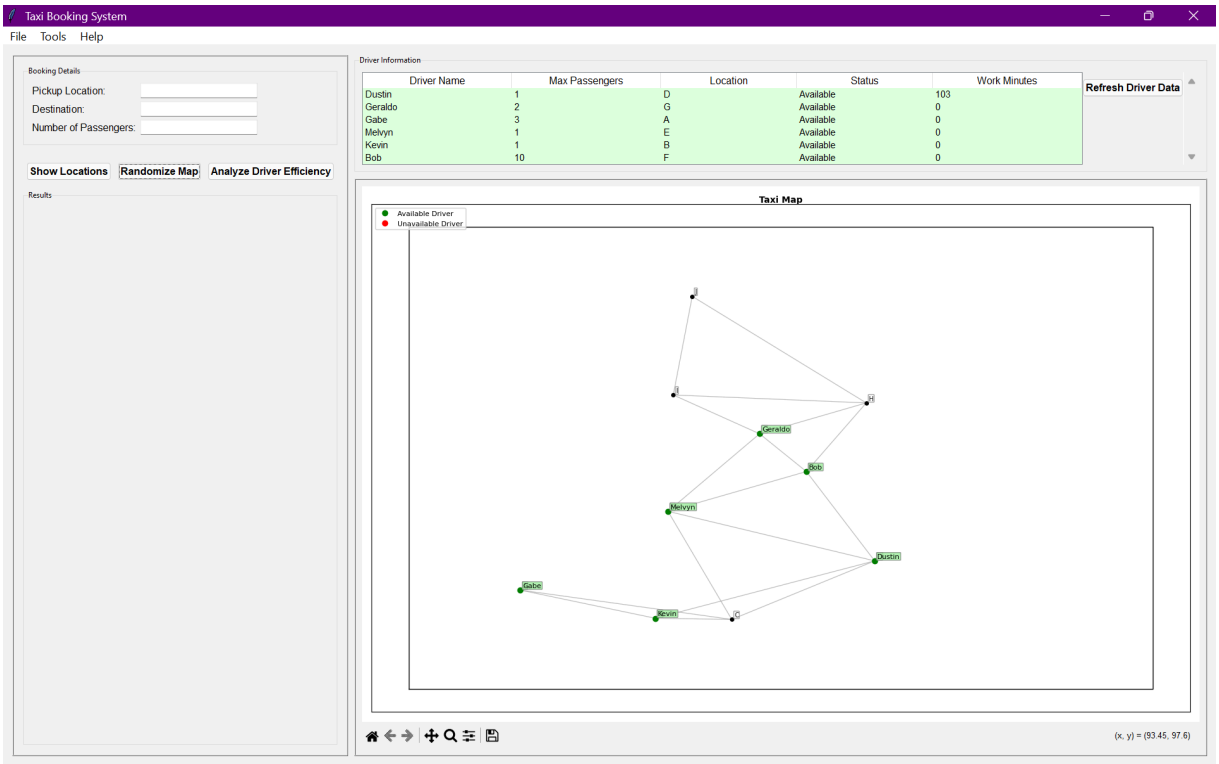
mengenai driver, **Taxi Map** untuk menampilkan weighted graph / peta / map, dan **Result** untuk menampilkan hasil setelah melakukan booking / pesan / order.





Gambar 4 : Tampilan Analisis Aplikasi Ojek

Berikut adalah tampilan analisis untuk menampilkan penentuan dalam mencari driver yang paling tepat untuk lokasi yang ditentukan oleh pengguna. Berisikan **Nama**, **Trip Parameter**, dan **Time Parameter Calculation**



Gambar 5 : Tampilan Setelah Melakukan Randomize

Ini adalah output dari kode yang mengacak peta taksi, untuk memberikan masalah lain di mana kita dapat sepenuhnya menguji apakah program yang kita tulis dapat menemukan pengemudi yang paling efisien.

4. Kesimpulan

Penelitian ini menyajikan sistem pencocokan pengemudi-penumpang yang dioptimalkan untuk platform ojek online dengan menggunakan algoritma Dijkstra untuk menghitung jarak tempuh terpendek antara pengemudi dan penumpang. Sistem ini dirancang dengan pendekatan berbasis graf, di mana jaringan jalan dimodelkan sebagai graf berbobot, sehingga memungkinkan pencarian jalur yang efisien dan alokasi pengemudi yang adil. Implementasinya mencakup GUI berbasis Tkinter dengan visualisasi matplotlib untuk menampilkan hasil pencocokan waktu nyata, lokasi pengemudi, dan rute yang dihitung, untuk memastikan transparansi dan kemudahan penggunaan.

Kesimpulannya, sistem ini menawarkan solusi yang praktis, efisien, dan transparan untuk platform ojek online, meningkatkan utilisasi pengemudi dan kepuasan penumpang, sekaligus menjadikannya ramah pengguna bagi mereka yang tidak terlalu paham teknologi. Dengan terus menyempurnakan algoritma dan antarmuka, pendekatan ini dapat diadaptasi untuk berbagai layanan transportasi dan logistik di seluruh dunia.

5. Daftar Notasi

Contoh penulisan notasi dapat diuraikan dengan keterangan sebagai berikut:

T = total_score
x = pickup_score
y = trip_score
z = capacity_score
a = pickup_distance
b = total_distance
c = capacity_utilization
d = passenger_count
e = max_passengers

Referensi

Referensi / acuan utama yang digunakan ialah jurnal nasional / internasional dan prosiding. Semua referensi sebaiknya *up-to-date* dengan perkembangan keilmuan dan ditulis dengan menggunakan *IEEE style*. Silahkan menggunakan format – format yang telah disediakan dalam panduan penulisan makalah ini:

Jurnal:

[1] M. Ghallab, D. Nau, and P. Traverso, Automated Planning and Acting. Cambridge University Press, pp 486-497, 2016. *(Discusses AI planning techniques for resource allocation, applicable to driver scheduling.)*

[2] H. N. Psaraftis, "Dynamic Vehicle Routing: Status and Prospects," Transportation Research Part C: Emerging Technologies, vol. 19, no. 6, pp. 1027–1045, 2011. *(Discusses real-time matching algorithms for dynamic transportation systems like ojek platforms.)*

Prosiding:

[3] L. Zhao et al., "A Graph-Based Approach for Real-Time Ride-Sharing Optimization," in Proc. IEEE Intl. Conf. on Data Mining (ICDM), 2018, pp. 1073–1078. *(Proposes Dijkstra's-based graph methods for ride-sharing, directly applicable to your case.)*

[4] A. F. Abdelghany et al., "A Simulation Framework for Modeling Large-Scale Ride-Hailing Systems," in Proc. Winter Simulation Conf., 2019, pp. 724–735. *(Includes GUI design principles for ride-hailing simulations, supporting your matplotlib/Tkinter approach.)*

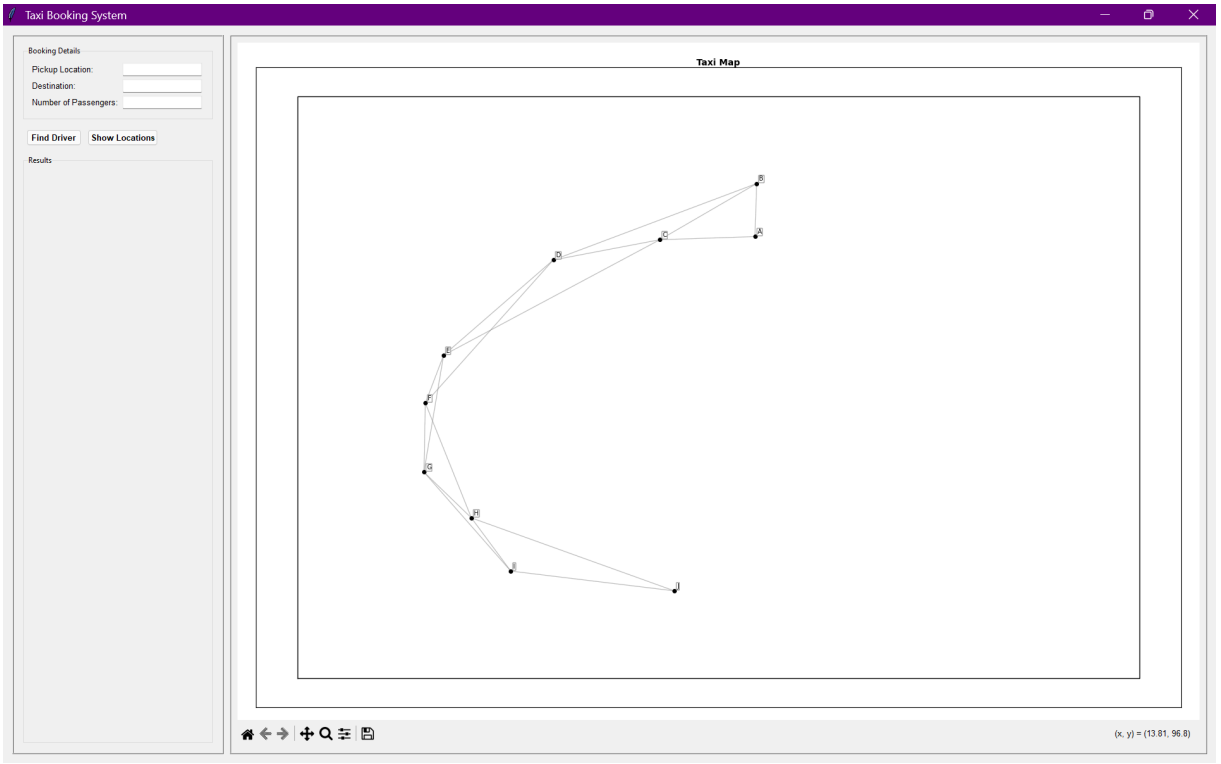
Buku:

[5] G. Berbeglia, J.-F. Cordeau, and G. Laporte, Dynamic Pickup and Delivery Problems. Now Publishers, 2010. *(Addresses real-time vehicle-passenger matching in transportation networks.)*

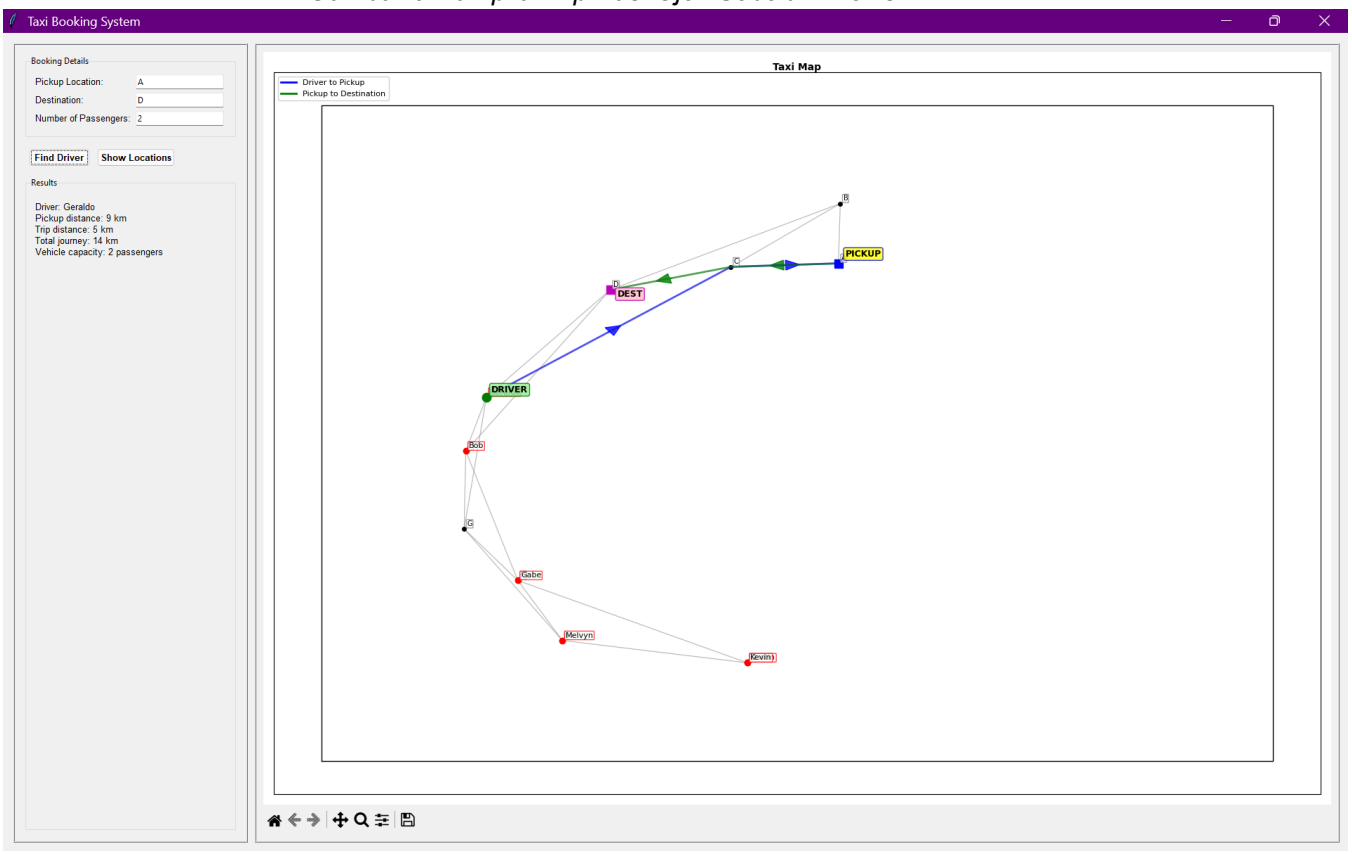
[6] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Network Flows: Theory, Algorithms, and Applications. Prentice Hall, 1993. *(Covers graph-based algorithms like Dijkstra's, applicable to route optimization.)*

6. Lampiran

- Tampilan awal ojek aplikasi sebelum revisi:

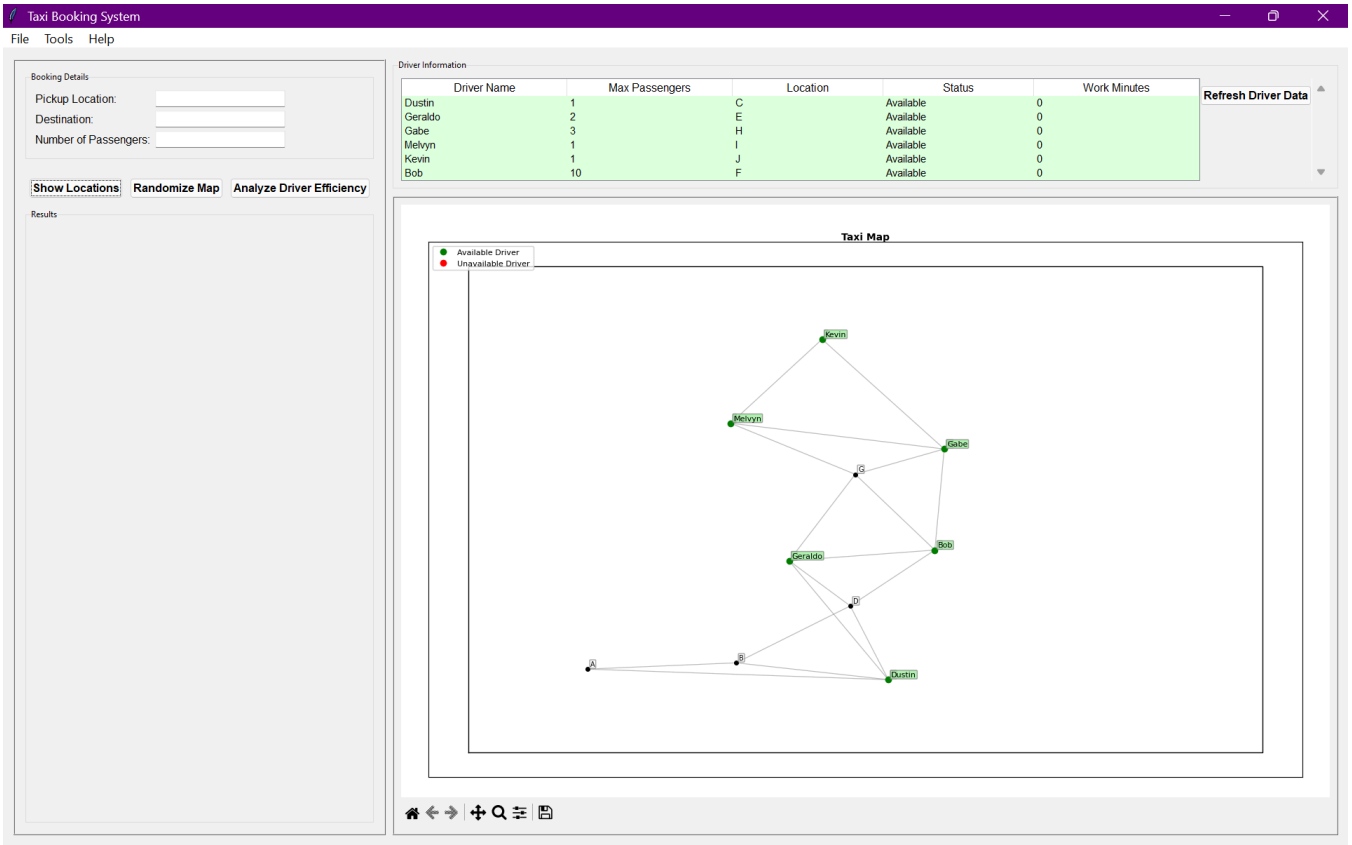


Gambar 6: Tampilan Aplikasi Ojek Sebelum Revisi



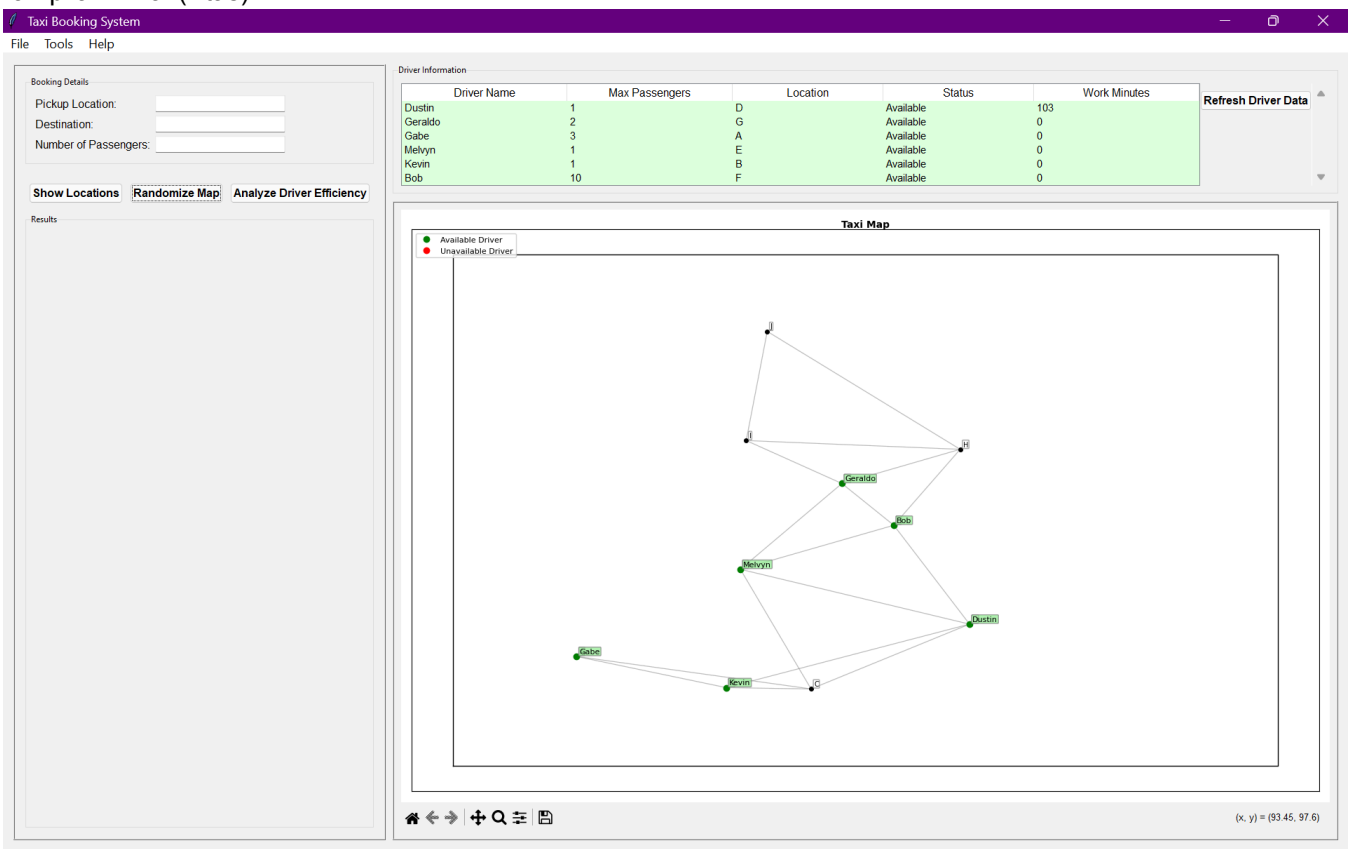
Gambar 7: Tampilan Hasil Aplikasi Ojek Sebelum Revisi

- Tampilan akhir aplikasi ojek setelah revisi:

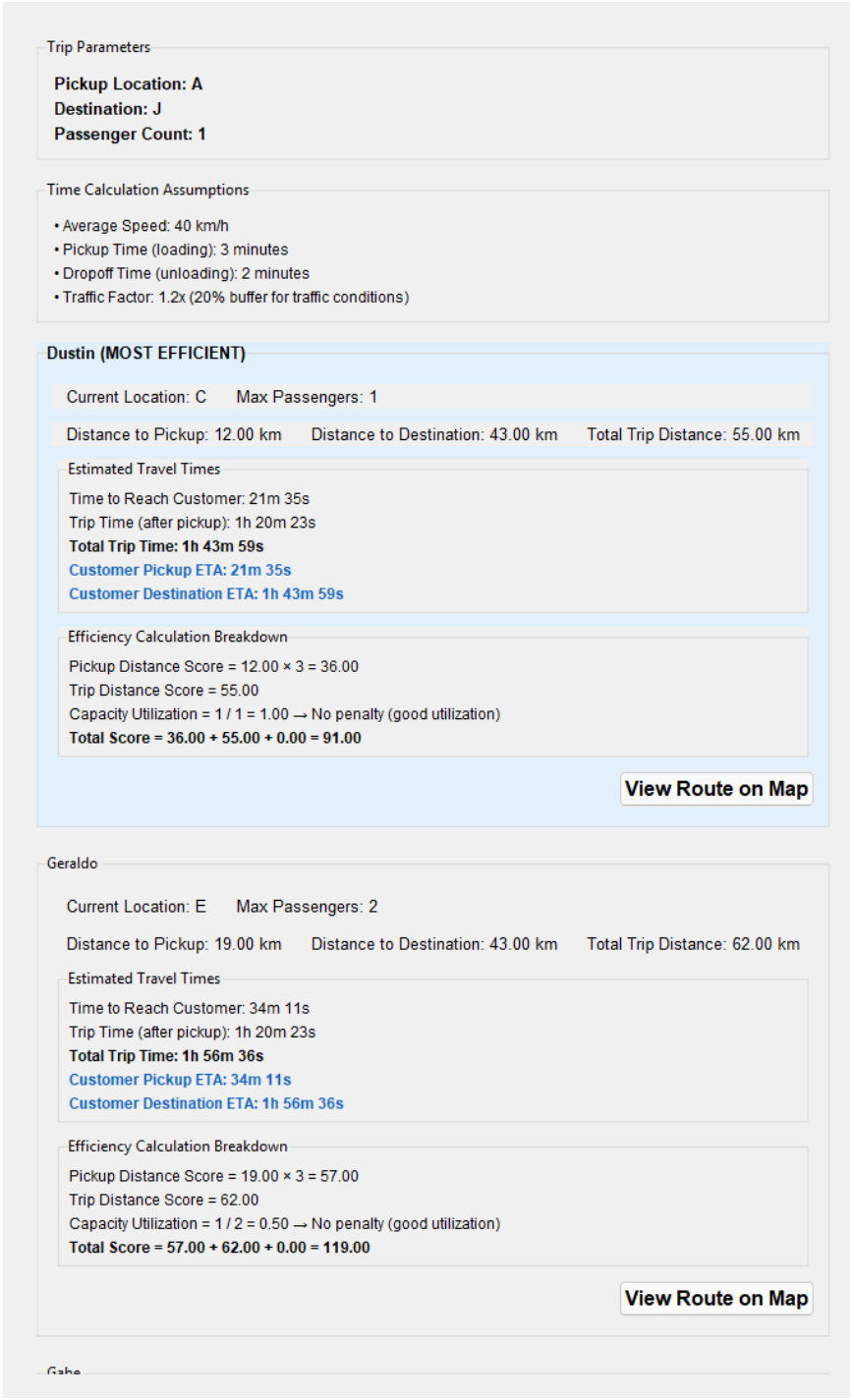


Gambar 8: Tampilan Hasil Aplikasi Ojek Setelah Revisi

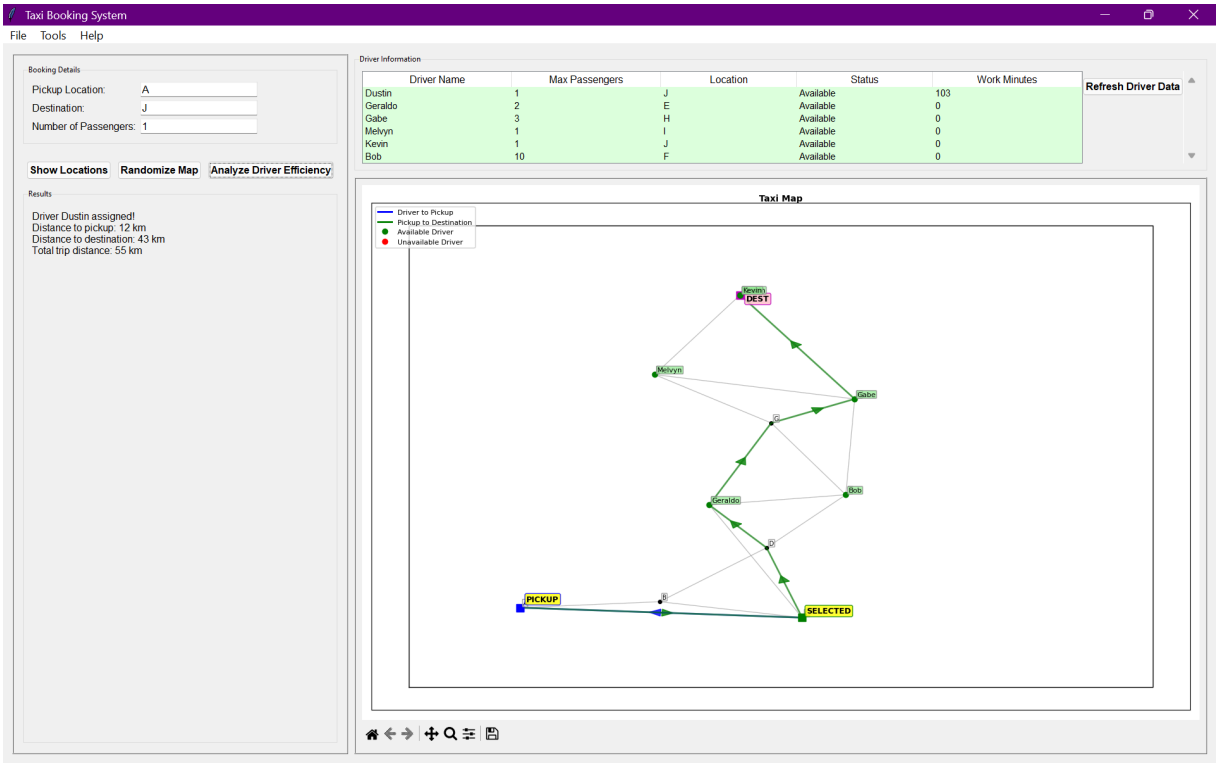
Tampilan Awal (Atas)



Gambar 9: Tampilan Hasil Aplikasi Ojek Setelah Revisi dan Setelah di Randomize

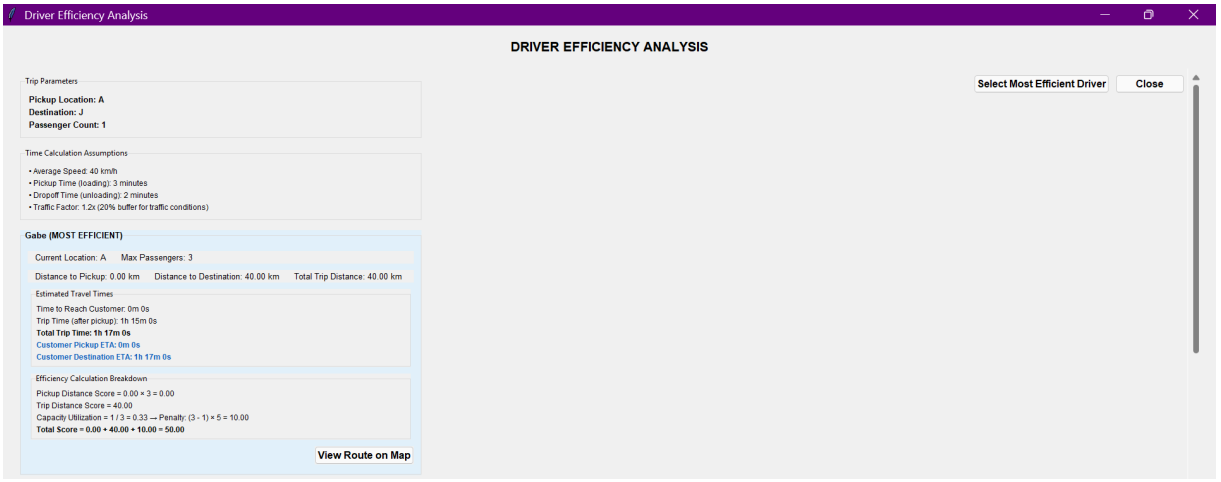


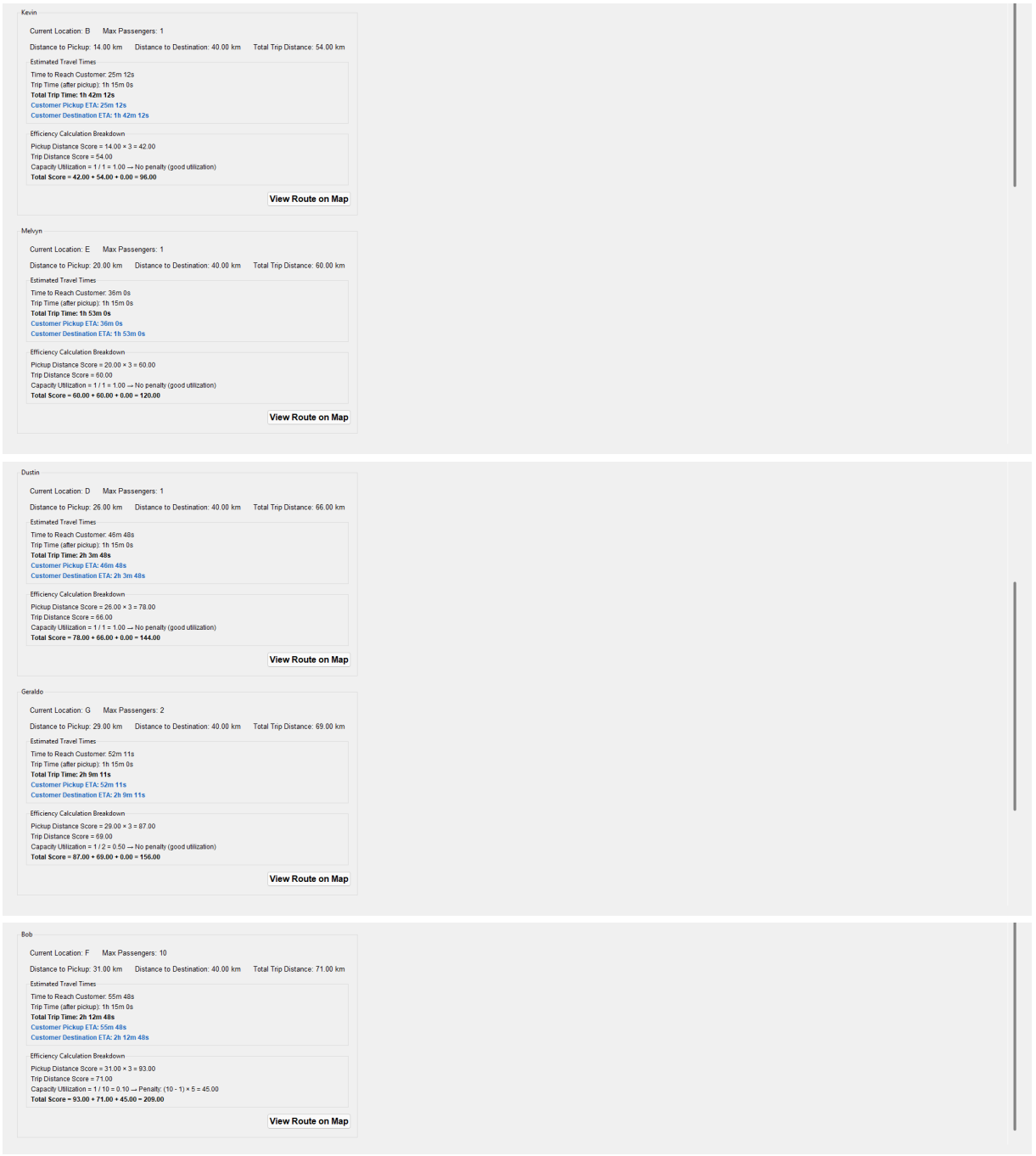
Gambar 10: Tampilan Hasil Driver yang paling efficient



Tampilan After Analisis (Atas)

Tampilan Analisis B (Bawah)




Kevin
Current Location: B Max Passengers: 1
Distance to Pickup: 14.00 km Distance to Destination: 40.00 km Total Trip Distance: 54.00 km
Estimated Travel Times
Time to Reach Customer: 25m 12s
Trip Time (after pickup): 1h 15m 0s
Total Trip Time: 1h 42m 12s
Customer Pickup ETA: 25m 12s
Customer Destination ETA: 1h 42m 12s
Efficiency Calculation Breakdown
Pickup Distance Score = 14.00 × 3 = 42.00
Trip Distance Score = 54.00
Capacity Utilization = 1 / 1 = 1.00 → No penalty (good utilization)
Total Score = 42.00 + 54.00 + 0.00 = 96.00
View Route on Map
Melvyn
Current Location: E Max Passengers: 1
Distance to Pickup: 20.00 km Distance to Destination: 40.00 km Total Trip Distance: 60.00 km
Estimated Travel Times
Time to Reach Customer: 36m 0s
Trip Time (after pickup): 1h 15m 0s
Total Trip Time: 1h 53m 0s
Customer Pickup ETA: 36m 0s
Customer Destination ETA: 1h 53m 0s
Efficiency Calculation Breakdown
Pickup Distance Score = 20.00 × 3 = 60.00
Trip Distance Score = 60.00
Capacity Utilization = 1 / 1 = 1.00 → No penalty (good utilization)
Total Score = 60.00 + 60.00 + 0.00 = 120.00
View Route on Map
Dustin
Current Location: D Max Passengers: 1
Distance to Pickup: 26.00 km Distance to Destination: 40.00 km Total Trip Distance: 66.00 km
Estimated Travel Times
Time to Reach Customer: 46m 48s
Trip Time (after pickup): 1h 15m 0s
Total Trip Time: 2h 3m 48s
Customer Pickup ETA: 46m 48s
Customer Destination ETA: 2h 3m 48s
Efficiency Calculation Breakdown
Pickup Distance Score = 26.00 × 3 = 78.00
Trip Distance Score = 66.00
Capacity Utilization = 1 / 1 = 1.00 → No penalty (good utilization)
Total Score = 78.00 + 66.00 + 0.00 = 144.00
View Route on Map
Geraldo
Current Location: G Max Passengers: 2
Distance to Pickup: 29.00 km Distance to Destination: 40.00 km Total Trip Distance: 69.00 km
Estimated Travel Times
Time to Reach Customer: 52m 11s
Trip Time (after pickup): 1h 15m 0s
Total Trip Time: 2h 9m 11s
Customer Pickup ETA: 52m 11s
Customer Destination ETA: 2h 9m 11s
Efficiency Calculation Breakdown
Pickup Distance Score = 29.00 × 3 = 87.00
Trip Distance Score = 69.00
Capacity Utilization = 1 / 2 = 0.50 → No penalty (good utilization)
Total Score = 87.00 + 69.00 + 0.00 = 156.00
View Route on Map
Bob
Current Location: F Max Passengers: 10
Distance to Pickup: 31.00 km Distance to Destination: 40.00 km Total Trip Distance: 71.00 km
Estimated Travel Times
Time to Reach Customer: 55m 48s
Trip Time (after pickup): 1h 15m 0s
Total Trip Time: 2h 12m 48s
Customer Pickup ETA: 55m 48s
Customer Destination ETA: 2h 12m 48s
Efficiency Calculation Breakdown
Pickup Distance Score = 31.00 × 3 = 93.00
Trip Distance Score = 71.00
Capacity Utilization = 1 / 10 = 0.10 → Penalty: (10 - 1) × 5 = 45.00
Total Score = 93.00 + 71.00 + 45.00 = 209.00
View Route on Map