Computer Science Fundamentals and Career Pathways

Assignment 1 - Foundations of Computer Science & Computational Thinking

By - Daksh 2501730141 (AI&ML Sec – A)

Assignment Title – Design and Stimulate a Real-World Process Using Flowcharts and Pseudocode

Problem Selected

Ride-hailing service (like Rapido/Uber)

INTRODUCTION:

Ride-hailing services such as Uber, Ola, and Rapido have become an integral part of modern transportation, offering convenient and reliable mobility through digital platforms. These systems are not only user-friendly but also rely heavily on computational processes such as booking management, payment verification, driver allocation, and real-time tracking.

This assignment focuses on designing and simulating the working of a ride-hailing service using computational thinking principles. By applying abstraction, decomposition, and pattern recognition, the problem is analyzed and structured into manageable components. A flowchart and pseudocode are then developed to represent the complete process logically. Finally, a sample implementation in Python demonstrates how a part of the system can be translated into working code.

The objective of this exercise is to bridge the gap between real-world problem-solving and computational design, while also improving algorithmic thinking and coding skills relevant to computer science.

Problem Analysis

Abstraction:

1. The user books a ride and makes the required online payment though the app.
2. The system checks the booking and processes the payment.
3. A driver is assigned for the ride their details are shared with the user through the app.
4. The user tracks the driver in real-time through the app.

Decomposition:

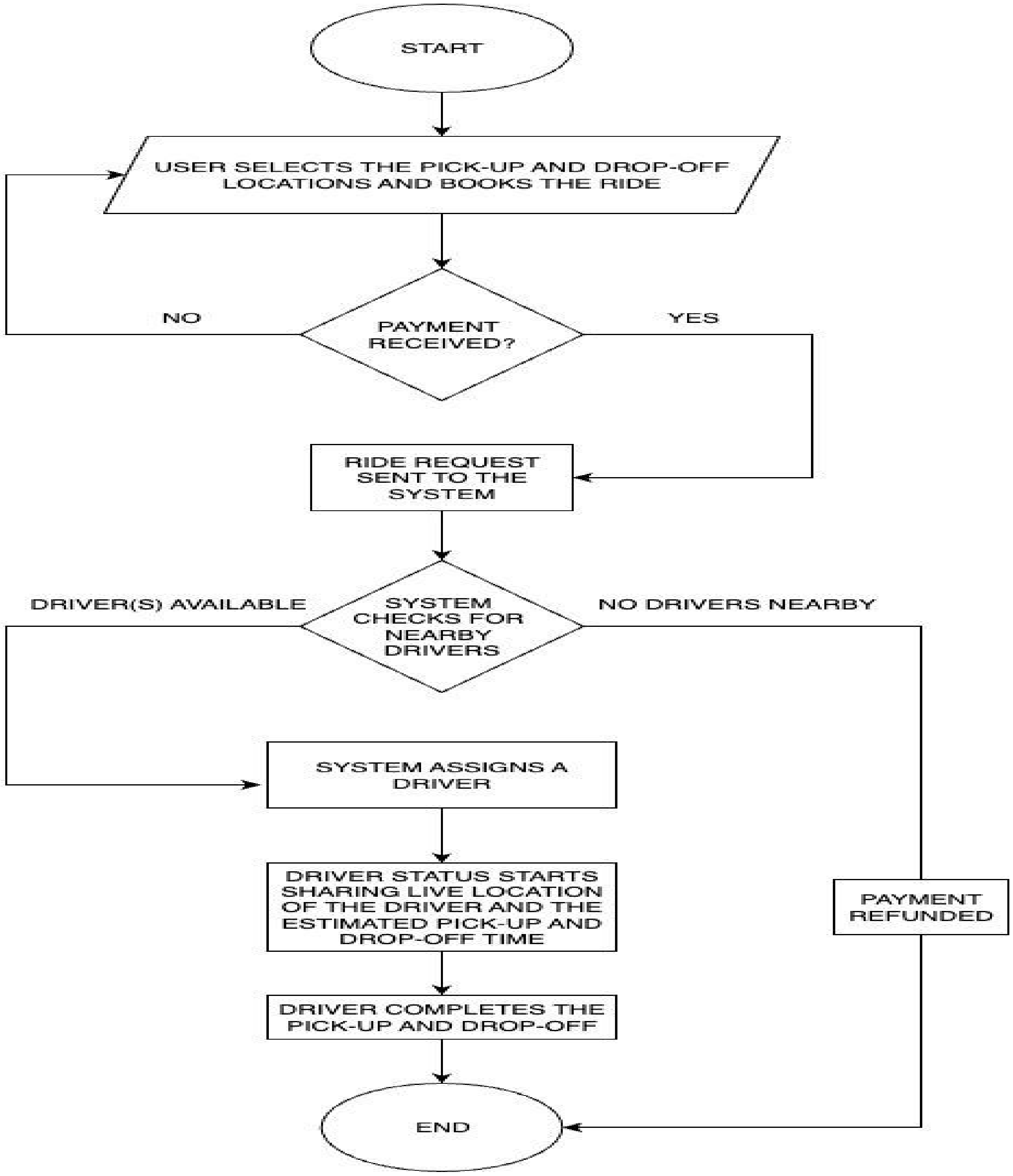
1. User - Selects the time and locations (pick-up and drop-off) and makes the required payment.
2. System - Checks the booking made and processes the payment.
3. Driver Assigning - Match the booking with the nearest available driver.
4. Driver status - The driver's real-time location and the estimated arrival time is made available to the user through the app.
5. Notify - The user gets notified for any important changes or reminders throughout the course of the ride through the app.

Patter Recognition:

1. The user interface is similar to a food delivery service where instead of the food item(s) they choose location(s).
2. The payment process is also similar since its generally online, whether through UPI/cards or through cash on delivery.
3. The matching of drivers is similar to matching food delivery men (swiggy/zomato).
4. Status updates completely resemble the tracking system in food delivery apps.

Solution Design

(A) Flowchart


START
USER SELECTS THE PICK-UP AND DROP-OFF LOCATIONS AND BOOKS THE RIDE
PAYMENT RECEIVED?
NO
YES
RIDE REQUEST SENT TO THE SYSTEM
SYSTEM CHECKS FOR NEARBY DRIVERS
DRIVER(S) AVAILABLE
NO DRIVERS NEARBY
SYSTEM ASSIGNS A DRIVER
DRIVER STATUS STARTS SHARING LIVE LOCATION OF THE DRIVER AND THE ESTIMATED PICK-UP AND DROP-OFF TIME
DRIVER COMPLETES THE PICK-UP AND DROP-OFF
PAYMENT REFUNDED
END

(B) Pseudocode

```
BEGIN
PRINT "Welcome to Ride-Hailing Service"

// Step 1: User requests a ride

INPUT user_location
INPUT destination

// Step 2: Estimate fare and time
SET estimated_fare = CALCULATE_FARE(user_location, destination)
PRINT estimated_time
SET estimated_time = CALCULATE_ETA(driver, user_location)
PRINT estimated_fare

// Step 3: Payment
INPUT payment_method
PROCESS_PAYMENT(payment_method, estimated_fare)

// Step 4: Search for available drivers nearby.
SET driver_list = FIND_DRIVERS_NEAR(user_location)
IF driver_list is EMPTY THEN PRINT "No drivers available. Please try again later."
EXIT
```

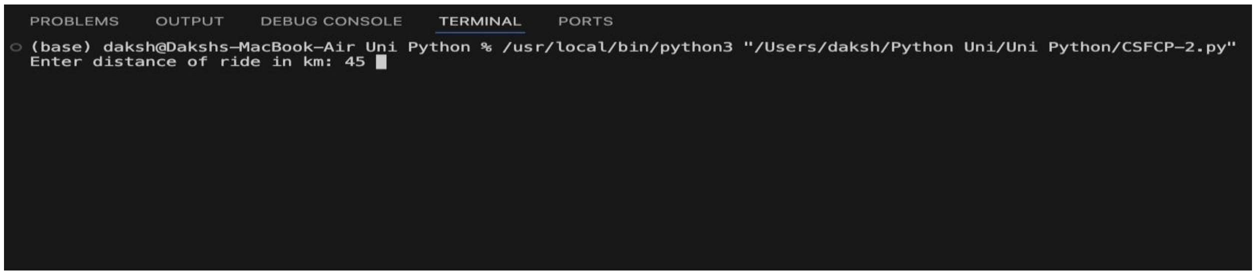
```
// Step 5: Match ride with the closest driver

SELECT driver = FIND_NEAREST_DRIVER(driver_list, user_location)


// Step 6: Notify driver

SEND_REQUEST_TO_DRIVER(driver, user_location, destination)

IF driver_accepts == FALSE THEN PRINT "Driver declined. Searching for another driver..."
GO TO Step 5 ENDIF


// Step 7: Track ride

PRINT "Driver on the way..."

UPDATE driver_position

PRINT "Driver is ", distance_to_user, " km away" ENDWHILE

IF distance_to_user <= 50m

PRINT "Driver has arrived. Ride started."


// Step 8: End of ride

PRINT "You have arrived at your destination."

PRINT "Thank you for using our Ride-Hailing Service!"

END
```

Implementation

SIMPLE PYTHON CODE FOR FARE CALCULATION AND TIME ESTIMATION:

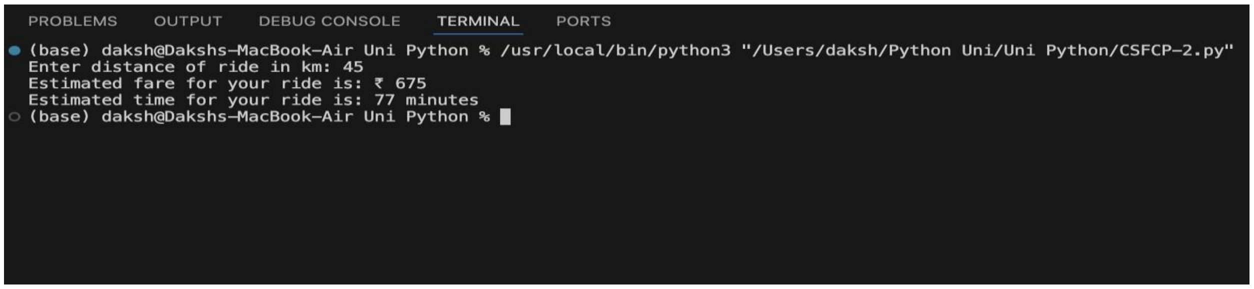
```python
# Simple Ride Fare and Time Estimation

# Take input from the user
distance = int(input("Enter distance of ride in km: "))
rate = 15 # fixed rate per km in rupees(₹)
speed = 35 # average speed in km/h

# Calculate fare
fare = distance * rate

# Calculate estimated time (in hours, then convert to minutes)
time_hours = distance / speed
time_minutes = round(time_hours * 60)

# Show results
print("Estimated fare for your ride is: ₹", fare)
print("Estimated time for your ride is:", time_minutes, "minutes")
```

INPUT:



```
(base) daksh@Dakshs-MacBook-Air Uni Python % /usr/local/bin/python3 "/Users/daksh/Python Uni/Uni Python/CSFCP-2.py"
Enter distance of ride in km: 45
```

OUTPUT:



```
(base) daksh@Dakshs-MacBook-Air Uni Python % /usr/local/bin/python3 "/Users/daksh/Python Uni/Uni Python/CSFCP-2.py"
Enter distance of ride in km: 45
Estimated fare for your ride is: ₹ 675
Estimated time for your ride is: 77 minutes
(base) daksh@Dakshs-MacBook-Air Uni Python %
```

Reflection

INSIGHTS GAINED:

Working on this assignment enhanced my understanding of breaking down complex, real-world systems like a ride-hailing service into smaller, manageable components. Creating the flowchart and pseudocode was especially helpful in organizing the system's logic, enabling clear visualization of how each part interacts. This structured approach made it easier to focus on the essential functions, such as booking, payment processing, driver allocation, and tracking, while filtering out unnecessary details. It reinforced the importance of computational thinking principles like abstraction, decomposition, and pattern recognition in problem-solving.

CHALLENGES FACED AND POTENTIAL IMPROVEMENTS:

One of the main challenges was accurately translating dynamic real-world activities, such as driver assignment and real-time updates, into computational steps that are logical and efficient. Implementing the fare and time calculation in Python helped solidify my algorithmic skills and the importance of clear, precise coding. I also realized the value of anticipating potential issues like error handling, which would make the system more robust. In future projects, adding such features or simulating more complex scenarios could deepen the learning experience and improve the solution's practicality. Overall, this assignment strengthened both my technical and analytical skills in developing computational solutions.

GitHub Profile: https://github.com/daksh-dua30

GitHub CSFCP Repository: https://github.com/daksh-dua30/CSFCP-Assignments/tree/e559de9a8cb76c38c4ae8f482d1ebfa6153719e8