Cargo Hitchhiking Python Simulation Programming Specification

# 1. Context & Objective

This simulation models a cargo-hitchhiking logistics platform operating in Islamabad, integrating ad-hoc drivers, dedicated fleets, and potential transit hitchhiking options. It will follow a rolling-horizon, event-based matching approach where assignments are updated whenever new orders/drivers arrive or deadlines approach. Pricing and wage policies will be dynamic, and orders can expire, be cancelled, or be delivered. A dedicated fleet is used as a backup.

# 2. Core Entities & Data Fields

## Orders

- order\_id: str  
- created\_at: datetime  
- pickup\_lat, pickup\_lng: float  
- drop\_lat, drop\_lng: float  
- time\_window\_start, time\_window\_end: datetime (hard time windows)  
- latest\_departure: datetime (dispatch deadline)  
- parcel\_volume\_l: float (liters)  
- parcel\_weight\_kg: float  
- parcel\_size\_class: {XS, S, M, L, XL}  
- service\_level: {same\_day, next\_day, flex}  
- base\_price: float  
- status: {published, accepted, delivered, expired, cancelled}

## Drivers (Ad-hoc)

- driver\_id: str  
- home\_base\_lat, home\_base\_lng: float  
- current\_lat, current\_lng: float  
- available\_from, available\_to: datetime  
- vehicle\_type: {bike, motorbike, car, van}  
- capacity\_volume\_l: float  
- max\_weight\_kg: float  
- max\_detour\_km: float  
- speed\_kmph: float  
- acceptance\_rate\_7d: float  
- rating: float  
- wage\_expectation\_per\_km: float

## Dedicated Fleet (Backup)

- fleet\_id: str  
- capacity\_volume\_l: float  
- max\_weight\_kg: float  
- cost\_per\_km: float  
- cost\_per\_min: float  
- dispatch\_cutoff: datetime

## Pricing/Wage Policy

- surge\_multiplier\_grid: dict[(cell\_id, timeslot)] -> float  
- commission\_rate: float  
- wage\_model: {"fixed", "dynamic"}  
- price\_model: {"fixed", "dynamic"}  
- time\_slot\_discounts: {next\_day: %, flex: %}

## Network

- travel\_time\_matrix: callable((lat1, lng1), (lat2, lng2), time) -> minutes  
- distance\_matrix: callable(...) -> km

# 3. Events in Simulation

- OrderArrival(order\_id)  
- DriverArrival(driver\_id)  
- Tick(t) – triggers expiry checks and dispatch  
- Cancellation(entity\_id, reason)  
- DeliveryComplete(order\_id, driver\_id)

# 4. Matching & Optimization

At each decision point:  
1. Collect current orders (not yet assigned and not expired) and available drivers.  
2. Pre-filter feasible matches based on:  
 - Capacity (volume & weight limits)  
 - Time windows  
 - Max detour & driver availability  
3. Solve assignment problem:  
 - ILP for small/medium instances  
 - Greedy/local search for larger/real-time constraints  
4. Objective:  
 - Maximize profit (revenue - payouts - penalties) OR minimize operational cost  
5. Dispatch dedicated fleet for unmatched orders at expiry.

# 5. Pricing & Wages

- Price: base\_price(distance, size) × surge\_multiplier(area, timeslot) × service\_level\_factor  
- Wage: α \* distance + β \* time + γ(size)  
- Support commission, fixed, dynamic, optimal models

# 6. Constraints

- Parcel volume and weight limits per vehicle  
- Hard pickup and delivery time windows  
- Latest departure enforcement  
- Max detour distance  
- Bundle size limit per driver  
- Order expiry and cancellation handling

# 7. KPIs to Track

- Match rate (% assigned orders)  
- On-time delivery rate  
- Average delivery cost  
- Platform profit  
- Driver earnings/hour  
- Dedicated fleet usage rate  
- Average detour distance  
- Cancellation/expiry rates  
- Emissions per order

# 8. Suggested Python Module Layout

/sim  
 config.py  
 entities.py  
 network.py  
 events.py  
 matcher/  
 filters.py  
 milp.py  
 greedy.py  
 policies/pricing.py  
 engine.py  
 kpi.py  
 scenarios.py  
 run\_experiment.py

# 9. Event Loop Pseudocode

while event\_queue.not\_empty():  
 t, ev = event\_queue.pop()  
 apply(ev)  
 C = collect\_current\_orders\_not\_fixed()  
 D = collect\_available\_drivers()  
 E = build\_feasible\_edges(C, D, network, constraints)  
 if E:  
 sol = solve\_assignment(E, objective="profit", time\_limit=2s)  
 freeze\_assignments\_with\_latest\_departure\_leq(t)  
 dispatch\_dedicated\_for\_orders\_crossing\_deadline()  
 update\_kpis()