INTER IIT
TECH MEET 15.0



HIGH PREP

imc

Autonomous Dealer Bot: Strategize on market microstructure

INTRODUCTION

IMC is a research-driven trading firm where quantitative modeling, machine learning, and engineering shape how modern markets are traded. A stabilizing force in markets since 1989, we provide liquidity across trading venues, delivering the best outcome in value and risk management to investors. Using our own technology and capital, we build proprietary systems and algorithms that operate across global markets. Our researchers, traders, and engineers work as a collective, combining rapid experimentation, advanced infrastructure, and real-time feedback to turn insight into execution and execution into advantage.

PROBLEM STATEMENT

Build an autonomous dealer bot that intelligently prices customer requests, competes against other dealers to win trades, and manages inventory risk to maximise net profit over repeated trading sessions. Customers enter the market with requests to buy or sell an asset at a given time, and every participating dealer receives the same request simultaneously. Your bot must independently evaluate each opportunity, decide whether to participate, and determine the price at which it is willing to trade. Since the best eligible price wins the trade, the bot must strike a balance between quoting competitively enough to secure transactions and maintaining sufficient margins to remain profitable.

Every successful trade changes the bot's inventory position, introducing exposure to future price movements. The bot must continuously monitor its holdings, assess the associated market risk, and decide whether to hedge its position through subsequent trades or retain it in anticipation of favourable price movements.

The bot must operate autonomously across multiple trading sessions, processing incoming customer requests, generating quotes, executing trades, and adjusting its inventory strategy without manual intervention once a session begins. Its decision-making framework should adapt to evolving market conditions and balance trade execution, profit margins, inventory exposure, and risk management.

THE CHALLENGE

You run a dealer: a market maker that customers come to when they want to trade.

1. Customers broadcast requests for quote (RFQs). Each RFQ names a product, a direction (buy or sell), a quantity and a response deadline. Every dealer receives the same RFQ and may quote a price or decline.
2. The customer picks the best eligible price. For a customer buy, the lowest ask wins; for a sell, the highest bid wins. Ties are broken at random.
    - "Eligible" means the price is within that customer's hidden limit, which is unique to each order and never published.
    - Each customer's limit is a maximum distance from the exchange price at the moment of matching, unique to each order and never published.
    - Example: a customer wants to buy 10 lots, and the dealers quote 10, 11 and 12. The customer takes 10, but only if 10 is within their limit (say, at most 0.158*X above the exchange price{X}). If not, nobody trades.
3. Winning a trade gives you a position, and with it risk. You can hedge on the exchange (which costs fees) or hold the position, hoping later customer flow offsets it.
4. At the end of each session, every position is valued at its settlement price. Your result is your net PnL: customer trades, hedging costs, fees, penalties and the value of your positions at settlement.

What separates good bots:

- Competitive pricing: tight enough to win, wide enough to earn.
- Avoiding adverse selection: trades that look good but aren't.
- Smart hedging: what to hedge, when, and at what cost.
- Staying inside the risk limits.

DATES OF THE ROUNDS

There are 9 rounds in three difficulty tiers. You get time to improve your bot between rounds. Dates are subject to change; any change will be announced on the discord.

Round	Date	Days from Previous Round	Tier	Round Points
Round 1	20 Oct 2026	-	Beginner	100
Round 2	25 Oct 2026	5 days	Beginner	100
Round 3	29 Oct 2026	4 days	Moderate	1000
Round 4	02 Nov 2026	4 days	Moderate	1000
Round 5	06 Nov 2026	4 days	Moderate	1000
Round 6	13 Nov 2026	7 days	Difficult	2500
Round 7	20 Nov 2026	7 days	Difficult	2500
Round 8	27 Nov 2026	7 days	Difficult	2500
Round 9	04 Dec 2026	7 days	Difficult	2500

Milestone	Date
Kick-off: competition repository, dev kit, and Round 1 data released	With PS
Submission deadline for each round	EoD (11:59 pm) of the day on which round occurs
Mid Evaluation	9th November, 2026
Final Submission deadline of report	5th December, 2026

Tier	Rounds	Products	Main Character
Beginner	1-2	2-40	Published fixed relationships (e.g. B settles at 2 × A), several independent groups, baskets, inventory management
Moderate	3-5	60-300	Probabilistic relationships to be estimated, independent products, mean-reverting spreads, lead–lag effects, changing correlations
Difficult	6-9	400-1000	Nonlinear relationships, regime changes, relationship breaks, market impact, stressed liquidity

- All products are cash securities; there are no options.
- Early rounds: prices move independently of what dealers do.
- Later rounds: your trading can move prices (market impact). Hedging may move from simple execution at the exchange price to a shared order book with finite liquidity and transaction costs.
- Customers: behaviour becomes more varied, including liquidity-driven and information-driven customers. The latter trade when they know something you don't.

Any change to the API or the trading rules for a round is announced with that round's data release.

ROUND CONDUCT

How a round runs:

Round x−1 ends ⟶ Round x data released (data_round_x.csv, settlement_rules_round_x.csv, demo exchange) ⟶ development window (build, test on the product specifications) ⟶ submission deadline ⟶ Round x evaluated (repeated sessions, reset portfolios) ⟶ results

NOTE: demo_data.csv is the dataset for Round 1

First round dataset

1. Data release. When round x − 1 is complete, you receive the data for round x. Round 1's data comes with the kick-off.
2. Development window. Improve your bot and test it locally with the development simulator.
3. Submission. Upload <IIT_code>_<round_name>.py before the deadline.
4. Evaluation. The organisers run your bot, with no manual intervention, across repeated sessions:
   - each with configurations you haven't seen, drawn from the same model as the released data
   - each starting from a reset portfolio
5. Results. A public leaderboard per round (each team's net PnL, customer orders and customer notional), plus, privately to each team, its own trades and results for every session. Other teams' trades and quotes, the full audit log and the session seeds are not shared.

Data you receive for each round

File	Contents
data_round_x.csv	Historical prices of every product in round x: a simulated history from the same price model as the live sessions, at the live resolution. Columns: time_s, round, price_version, <one column per product>. Use it to estimate volatility (realised volatility information), correlations, spreads and lead–lag.
settlement_rules_round_x.csv	How each product settles: LAST_PRICE, or an exact formula of other products' settlement prices (e.g. B = 2*A, BASKET = 0.5*A + 0.5*C). Exact relationships are published where they exist; statistical relationships must be estimated from the data.
Product specifications	Tick size, lot size, position limits, breach penalties and exchange fees per product. Also available live to your bot in market["products"].

The live sessions follow different random paths from the released history. The data tells you how products behave, not what the prices will be.

TRADING FORMAT

A session is a sequence of RFQ cycles. These are the defaults; each round's specifications confirm the actual values.

Item	Default
RFQ cycle ("interval")	5 seconds
Exchange price updates	10 per interval (every 0.5 s)
RFQs per interval	Grows with the round (3 in small rounds, 50+ in large ones)
Final hedge window	2 seconds after the last interval: no new RFQs, exchange trading only
A session	A sequence of intervals; each round is evaluated over many sessions
Starting capital	Equal for all teams (100,000 in the demo; confirmed per round)
Session start	Interval 1 begins when every team's bot has connected (or after a grace period)

In every interval

1. New RFQs are published to all dealers with current exchange prices and the deadline (closes_at).
2. Until the deadline, you can quote and re-quote as prices update. Only your latest quote counts.
3. At the deadline, each RFQ goes to the best eligible quote. Fills are all-or-nothing, at your quoted price.
4. You can trade on the exchange at any time, at the latest price plus fees, to hedge or reposition.

At the end of the session

- Penalties: positions beyond their limits are charged.
- Settlement: every position is valued at its settlement price, following the published rules.
- Score: net PnL = cash (customer trades, exchange trades, fees, penalties) + positions × settlement prices × multiplier − starting capital. The multiplier is 1 unless a product's specification says otherwise.

Rule	Effect
Position limits (long and short, per product)	Exchange trades that would take you further beyond a limit are rejected. Customer fills can breach a limit, and are then penalised
Breach penalty	charged per unit beyond the limit, at every interval boundary and at settlement, until you reduce the position
Exchange fees	per unit and/or a percentage of notional, on exchange trades only. Customer trades are free
Compute and API budget	equal for all teams: each call must return within 1 second, plus message-rate caps (see PARTICIPANTS.md)

What your bot can and cannot see

Rule	Effect
Public market data: exchange prices and every RFQ	Customers' limits and motives
Your own quotes, fills, inventory, cash, fees and penalties	Other teams' quotes, inventory or cash during the session
Each RFQ's outcome: the winning price and whether it filled	Future prices or upcoming market changes

API GUIDELINES

You write one Python class. The organiser's runner handles all networking, timing and reconnection. Full reference: PARTICIPANTS.md.

Python

```
from rfq.sdk import round_lot, round_quote

class Dealer:
    def __init__(self, edge=0.10):          # every parameter needs a default
        self.edge = edge

    def quote(self, orders, market, portfolio):
        """Return {order_id: price or None}. Called each interval and after every price update."""
        quotes = {}
        for o in orders:
            fair = market["prices"][o["product_id"]]
            tick = market["products"][o["product_id"]]["tick_size"]
            price = fair + self.edge if o["side"] == "BUY" else fair - self.edge   # side = CUSTOMER's side
            quotes[o["order_id"]] = round_quote(price, tick, o["side"])
        return quotes

    def hedge(self, market, portfolio):
        """Return [(product_id, "BUY" | "SELL", quantity), ...] to trade on the exchange now."""
        return [(k, "SELL" if q > 0 else "BUY", round_lot(abs(q), market["products"][k]["lot_size"]))
                for k, q in portfolio["positions"].items() if abs(q) >= market["products"][k]["lot_size"]]
```

Rules for your code

Rule	Detail
Language	Python 3.11
Allowed imports	Python standard library, numpy 2.3.5, rfq.sdk. Nothing else unless approved: if you need another module (e.g. scipy), ask the organisers on the official channel (Discord). Approved modules are added to the environment for all teams and announced, with pinned versions. Plan for a request to take effect from the next round
No I/O	No files, network, threads or subprocesses. Estimate parameters offline and put them in your code
Time	Each call must return in under 1 second
Robustness	An exception, a timeout or a wrong return type disconnects your bot for the rest of the session. Your open positions stay open and are settled, with penalties if they're beyond limits
Fair play	No attempts to interfere with the exchange, other bots or the evaluation infrastructure; exploiting bugs in it is grounds for disqualification. Report bugs to the organisers

Development Kit

Item	Use
demo/demo_participant.py	Baseline bot and starting template
demo/run_demo.py --bot myteam_round1.py	Development simulator: checks your file, then runs a full session against example opponents on demo data. The demo replays the same data each run, so vary --seed and --start-row before trusting a result. --rfqs 100 lets you practise the RFQ load of later rounds
examples/dealer.py	Baseline opponent bot used by the simulator
demo/demo_data.csv, demo/demo_settlement_rules.csv	Sample data in the same format as each round's release
demo/demo_data_holdout.csv	A second sample history from the same model, for testing on data you didn't tune on
Output of each run	Results.json, trades.csv (every trade), audit.sqlite (every message), for analysis
README.md	setup and the command for a first local session
PARTICIPANTS.md	the technical reference: session flow, the quote / hedge interface, data formats, exchange rules, and how to read results
demo/demo_exchange.py	the local exchange the simulator uses. It replays the sample prices and settles with the sample rules
rfq/sdk.py	the only helpers your bot may import: round_quote and round_lot. The rest of rfq/ is the local runner and exchange; your bot does not import it
requirements.txt environment.json	the pinned packages (Python 3.11, numpy 2.3.5) and the submission limits

Link to Development Kit

SUBMISSION FORMAT

Item	Requirement
File name	<IIT_code>_<round_name>.py, e.g. iitcode_round1.py, (IIT's name to not be revealed, penalties to be imposed otherwise)
team_code	your registered team name (IIT code as given): lowercase letters and digits only, no spaces, the same every round
round_name	Round 1 through 9
Contents	one self-contained file with a class named Dealer implementing quote() and hedge()
Constructor	Dealer() must work with no arguments; set your parameters as defaults
Where	To be notified before the first round starts
Resubmissions	Allowed until the deadline; the latest file counts
Missed deadline	No points awarded (Before mid-eval, post mid eval best 3 on 4 to be considered, refer scoring matrix)

Before you submit, check that python demo/run_demo.py --bot <IIT_code>_<round_name>.py passes the pre-flight check and completes a session. A file that fails to load scores nothing for the round.

SCORING MECHANISM

Points within a round

Each round has a points value P: 100 for Rounds 1–2, 1,000 for Rounds 3–5 and 2500 for Rounds 6–9. Points come from three rankings:

Ranking	Who is ranked	Score P = basepoints N = number of submissions by IITs r = rank *In that particular round
A. Net PnL	teams with a positive average net PnL, by that average	$((N - r) / N)^2 \times P$ (1st position gets full points, Dense Ranking to be followed in case of tie)
B. Customer volume	teams with a positive average net PnL and at least one customer trade, by executed customer notional (Σ quantity × price × multiplier), summed over the round's sessions	$((N - r) / N)^2 \times P/2$ (1st position gets full points, Dense Ranking to be followed in case of tie)
C. Customer orders	the same teams as B, by the number of customer RFQs filled, summed over the round's sessions	$((N - r) / N)^2 \times P/2$ (1st position gets full points, Dense Ranking to be followed in case of tie)

How the rankings are computed:

- Average PnL is over all of the round's sessions. A session where your bot didn't connect counts as 0. A session where it crashed counts at its actual PnL, settled with any open positions.
- Rounding: PnL and notional are rounded to 0.01 before ranking.
- Ties: the tied teams shall share equally the total points allocated to the ranks they occupy, calculated in accordance with the applicable scoring formula.
- Zero PnL: a team with zero or negative average PnL gets no points from any ranking in that round.

Hedge turnover (trading on the exchange) earns no volume or order points. Only trades with customers count.

Worked example (Round 4, P = 1,000):

Assume there are (N) participating IITs. Points are awarded based on rank using the applicable scoring formula.

- Team X finishes 2nd in PnL, 1st in customer volume, and 3rd in customer orders. Its points for each category are calculated using the rank-based scoring formula, with the relevant base points for each category. The total score is the sum of the points earned across all categories.
- Team Y records the highest customer volume but has a negative average PnL. It is ineligible for points in the round, and the customer volume points are awarded to the highest-ranked eligible team.
- Team Z is profitable but does not trade with customers (exchange trading only). It can earn PnL points based on its rank but receives no points for customer volume or customer orders.

Scoring Matrix

Component	Weight	What is assessed
Mid Evaluation	20% (75% idea+ 25% Documentation)	Interim report on strategy made, results obtained on the previous rounds, tweak made.
Results and Rubrics	40%	Your total round points, normalised against the top score and ranked.
Final Evaluation	30% (75% idea+ 25% Documentation)	Submission of report, results in all rounds, changes and approach
Presentation	10%	Clarity, Delivery, Explanation

***Considering the Endsems of the IITs post Mid Evaluation, best 3 out of 4 submissions will be used to calculate final scores in the Difficult rounds.**

DELIVERABLES

Per Round Submission:

1. A deployable dealer bot, compatible with the competition API: your final <IIT_code>_<round_name>.py (no names to be mentioned revealing the IIT otherwise strict penalty to be imposed)

Mid and End Eval Submission:

1. For both Mid Evaluation and Final Submission:
   A concise report explaining your strategy idea (7 Pages Mid eval and 15 Pages final term submission). How you came up with it? What did you use to test it? What conditions does it underperform in? Performance in each round, changes made to improve bot performance? **(.PDF format only, page limit is inclusive of cover and appendix.)**
2. Include the scripts, settings and seeds needed to reproduce them, using the development simulator.
3. Reproducible evidence:
   - baseline comparisons: your bot against the provided baseline bots
   - ablation studies: what each component of your bot contributes
   - failure analysis: where and why your bot lost money

No additions submissions of any kind to be made unless specified otherwise.

RULES AND FAIR PLAY

- Team composition, and one submission per team per round.
- Your bot must be your own team's work. Sharing code between teams is not allowed.
- Equal conditions: all teams get equal starting capital, common risk limits, and equal compute and API budgets.
- No manual intervention: bots run unattended during evaluation.
- Scoring rules are published in advance of each round, and the organisers' decision on scoring is final.
- Questions and clarifications: Through a Discord Channel.
- Submissions should not reveal the IIT name (strict penalty to be imposed).

FAQ

- **Do I have to hedge?**
  No. Holding positions is allowed, but they carry price risk, limit penalties and settlement exposure.

- **Can I quote only some RFQs?**
  Yes. Return None for an order to withdraw your quote. Leaving it out keeps any quote you already sent.

- **Will the live prices match the released data?**
  No. Same model, new random path. Learn the behaviour, not the numbers.

- **Can I use a library other than numpy?**
  Not until specified separately by the OC.

- **Can my bot read the CSV files during a session?**
  No. Do the analysis offline and put the parameters you estimate in your code.

- **What if my bot crashes mid-session?**
  It's disconnected for that session, and its open positions are settled with penalties if they're beyond limits. Other sessions in the round start fresh.

- **Does trading volume with the exchange count?**
  No. Only customer volume and customer orders count, and only for teams with positive average PnL.

- **Will the API change between rounds?**
  The interface (quote/hedge) stays the same. New features (e.g. an order book, market impact) are announced with the round's data release.

- **How do I know I'm ready to submit?**
  python demo/run_demo.py --bot <your file> passes its pre-flight check and ends with "✓ Your bot connected, quoted, traded, and stayed up for the whole session." If it lists things that need a look, read them first.

- **How do best 3 out of 4 is considered in the final submissions**

  Considering the Endsems of the IITs post Mid Evaluation, best 3 out of 4 submissions will be used to calculate final scores in the Difficult rounds. Percentage PnL to be considered for comparisons in case all 4 submissions are made. Any team not participating in the round would reduce N, refer **Scoring Mechanism**.