

1. Why Are We Building This?

Problem Statement

Indian day and swing traders face:

- **Too many stock choices:** Decision fatigue every morning
- **FOMO & missed entries:** Due to scattered tools, late discovery
- **No journaling or feedback loop:** No structured improvement over time
- **Tool overload:** Switching between TradingView, broker apps, news, screeners

Our Vision

To build a **smart, focused, non-broker trading assistant** for Indian traders that:

- Surfaces **only high-probability trade setups** daily
- **Reduces noise and distractions**
- Tracks what works for the user over time

We're not trying to replace brokers. We're building the **intelligence + workflow layer** they're missing.

Phase 1: MVP Core Engine (Updated)

Key Goal:

Enable a solo dev to deliver:

- Daily radar (scanning 200–500 stocks pre-market)
- Manual tracking interface
- Basic journaling
- Upstox OAuth + Holdings import

Features:

1. Radar Engine (Pre-market Only)

2. Inputs: List of NSE 200 tickers

3. Source: Upstox API (end-of-day + 15 min candles)

4. Indicators: RSI, Volume, Delivery %, 20/50 EMA

5. Output: `radar_output.json` with top 5–10 tickers and reason for selection (e.g., "RSI<30 + Vol Spike")

6. Trigger: Daily CRON at 8:45 AM IST

7. Manual Tracker

8. User selects stocks from radar manually

9. Stored in: `users/{uid}/tracked_today/`

10. Trade Journal UI

11. User logs entry/exit, PnL, notes

12. Optional tag: `setup used`

13. Stored in: `users/{uid}/trades/`

14. Upstox OAuth Integration

15. Frontend: Open login URL, capture `access_token`

16. Backend: Store encrypted token

17. Fetch endpoints:

- `get_holdings()` to display current positions
- `get_historical_trades()` (last 30d)

18. Use these trades to pre-fill journal fields and match to radar

Weekly Dev Plan

Week 1:

- Setup Upstox OAuth flow
- Store and test trade fetch for 1–2 users

Week 2:

- Build radar scanner using Upstox historical API
- Export daily JSON for front-end consumption

Week 3:

- Build journal and tracking UI
- Connect fetched trades with journal autofill

Week 4:

- Polish PnL logic, insights card, testing flows

2. Research Insights

[...unchanged for brevity...]

Phase 2: Smart Feedback Loop (Developer-Ready)

Goal: Help users understand what's working and improve performance based on trade data.

Features + Dev Specs:

1. Weekly Trade Insights

2. CRON job (`every Monday 7am`) aggregates all trades from last 7 days.
3. Group by `strategy_tag` , `day_of_week` , `time_of_day` , `ticker` .
4. Output example: `"You win more with RSI < 35 and Delivery > 30% on Mondays."`
5. Backend function: `generate_weekly_insights(user_id) -> List[Insight]`
6. Store in `users/{uid}/insights/{week_id}`

7. Auto-Scored Trade Effectiveness

8. Add field in trade: `radar_matched: bool` , `score: int`
9. Scoring logic (in `score_trade()`):
 - +1 if trade matched radar setup
 - +2 if trade was profitable (> 1% PnL or per defined threshold)
 - -1 if trade didn't match radar and was a loss
10. Function `score_trade(trade_data, radar_data)` is triggered during trade save or nightly cron
11. Function: `score_trade(trade_data, radar_data)`

12. Journal Suggestions Engine

13. Triggered if:
 - `tracked_today` but no trade log by 4PM
 - Trade log without `strategy_tag`
14. Alert via push/email: "Missed tracking RELIANCE? Add trade journal to improve insights."
15. Function: `check_and_nudge_journaling(user_id)`

Implementation Plan: Week 1:

- Implement `generate_weekly_insights()`

- Group trades and generate natural language summary
- Store in Firebase under insights collection

Week 2:

- Build scoring function (`score_trade`) during trade save/update
- Add post-trade analytics card in frontend journal summary

Week 3:

- Setup journaling nudges
- Build notification templates and triggers for missing journal entries



Phase 3: Live Alerts + Watchlist Sync (Developer-Ready)

Goal: Timely mid-day nudges based on tracked stocks without real-time infra.

Features + Dev Specs:


1. Delayed Intraday Alerts

2. Schedule: Run every 15 min between 9:30–2:45

3. For each `user.tracked_today`, fetch latest candle from Upstox

4. Re-apply radar rules in light version (e.g., RSI crossing 50)

5. Function: `run_intraday_check(tracked_list)` → If signal → Trigger `send_push(user_id, stock, signal_type)` via Firebase Cloud Messaging (FCM) or Telegram fallback

6. Alert type: “ RSI just broke 50 on TATAMOTORS. Time to act?”

7. Watchlist Import (Manual)

8. UI upload via CSV (headers: `ticker, reason, notes`)

9. Store in `users/{uid}/watchlist_imported/{id}`

10. Optional deduplication with radar-tracked list

11. Future-ready for broker sync in Phase 4

Implementation Plan: Week 1:

- Build `run_intraday_check()` logic: fetch live data, compare with RSI thresholds
- Design basic alert format

Week 2:

- Enable notification routing: Firebase function or Telegram for test users

- Trigger only for 1–2 stocks to start with

Week 3:

- Add upload interface for Watchlist CSV
- Integrate deduplication and sync with tracked list

Phase 4: Deep Power Features (Developer-Ready)

Note: Upstox OAuth and basic trade import functionality has been moved to **Phase 1** to enable early journaling and radar linking. Remaining advanced broker features stay in Phase 4.

Goal: Build advanced capabilities and open monetization gates.

Features + Dev Specs:

1. Custom Strategy Builder

2. UI: Dropdowns to select `Indicators` (RSI, EMA, MACD, Volume, Delivery %), `Conditions`, `Thresholds`

3. Output JSON schema:

```
{
  "name": "My RSI + Volume Setup",
  "rules": [
    {"indicator": "RSI", "operator": "<", "value": 35},
    {"indicator": "Volume Spike", "operator": ">", "value": 1.5}
  ]
}
```

4. Evaluation Engine: `evaluate_custom_strategy(ticker_data, strategy_json)`

5. Save under: `users/{uid}/strategies/{id}`

6. Broker Integration (Zerodha/Upstox/Dhan)

7. OAuth flow → Store access token in encrypted DB

8. Use official SDKs to fetch trades:

- `get_historical_trades()`
- `get_holdings()`

9. Auto-journal trades into `users/{uid}/trades/{}`

10. Add live P&L for `today's trades`

11. Real-Time Screener (Pro Tier Only)

12. Polling-based engine: `every 2 min` for subscribed users

13. Screen F&O universe for user-defined strategy JSON

14. Send instant alert if match

15. Requires scalable infra or event-driven system

Implementation Plan: Week 1-2:

- Build UI and logic for custom strategy JSON creation
- Store and evaluate sample strategies

Week 3-4:

- OAuth setup for one broker (Upstox recommended for now)
- Pull sample holdings and map trade data into journal

Week 5+ (Pro Tier):

- Begin polling-based infra setup for real-time screener
- Alert system to be integrated with Pro user layer