White Paper: Automated Meme Coin Trading Bot on Solana (Pump.fun Strategy)

Executive Summary

This white paper outlines the development, strategy, implementation, and operational approach behind an automated trading bot targeting meme coin launches on the Solana blockchain via the Pump.fun platform. Through intelligent trade filtering, conservative capital deployment, and structured reinvestment, this bot aims to deliver sustainable daily profits while mitigating risk.

Project Goals

- Identify early-stage meme coin launches with high growth potential
- Automate trade execution with buy-in logic, exit strategy, and slippage management
- Simulate real-world constraints including transaction fees, price decay, and liquidity
- Generate consistent profit with minimal human oversight

System Architecture

A. Data Collection & Filtering

- Monitors token creation events using Solana RPC or Helius API
- Pulls buyer activity in first 10s, 30s, and 5m windows
- Simulates price trajectory using on-chain data or randomized modeling

B. Decision Engine

- **Buy Trigger**: Enters with 0.05–0.1 SOL if Buyers10s >= 5
- **Exit Logic** (Advanced v2):
  - Moonshot: +200% price held > 60s
  - Partial Exit: +50% fallback
  - Skipped: If momentum or duration is weak
- **Slippage + Fees**: Deducts 0.0001 SOL and applies 1–5% slippage based on trade type

C. Logging

- Each trade logs:
  - Timestamp
  - Buyers (10s/30s/5m)
  - Entry price, Exit price
  - Profit (SOL & USD)
  - Strategy decision (Skipped, Partial, Moonshot)

D. Trade Execution

- Uses Jupiter Aggregator API for swaps
- Wallet funded with secure Solana key
- Uses `@solana/web3.js` + `@jup-ag/core` for trade submission

Startup Strategy: Snowballing from Low Capital

To reduce initial risk, the bot begins with a minimal starting balance (e.g., 0.1–0.5 SOL). Profits are recycled into future trades until a working capital of 5 SOL is achieved. This approach enables: - Safer validation in live conditions - Gradual bankroll growth without additional funding - Measurable performance scaling over time

Rules: - Start with 0.05–0.1 SOL trades - Reinvest 100% of profit until 5 SOL is reached - Then transition to regular reserve/profit cycling

---

Continuous Operation

A. Runtime

The bot is designed for continuous operation, driven by real-time token creation events on Pump.fun. Rather than targeting a specific trade count per hour, it applies strict entry logic whenever a token launch is detected. The system operates via interval polling or WebSocket feeds, ensuring responsive and adaptive trade evaluation around the clock.

B. Profits

The current algorithm is structured for profit: - Simulations consistently return **$4,200–$5,100** after accounting for all overhead - Slippage, fees, and conservative exits are built into the model

C. Reinvestment and Reserve Logic

- **Snowball Phase:** Reinvest all profits until reaching 5 SOL
- **Post-Snowball:**
  - Withdraw 50% of net profit weekly
  - Recycle remaining into bankroll
  - Maintain buffer to survive streaks of low output or negative trades

---

Profit Expectations

Updated Realistic Throughput Estimate

- 🚀 **Moonshots/hour**: ~9.1
- ✅ **Partial Exits/hour**: ~8.9
- 💰 **Avg. Profit/Moonshot**: 0.20 SOL
- 💰 **Avg. Profit/Partial Exit**: 0.05 SOL
- 📋 **Total Profit/hour**: 2.27 SOL
- 💵 **Estimated USD/hour (@ $166/SOL)**: $377.09

This estimate is based on observed simulation results and realistic trade pacing (~18 accepted trades/hour).

Legacy Estimate (for comparison)

- Average trade frequency: 20–40 per hour
- Expected win rate: ~60%
- Average profit per trade: ~0.06 SOL ($10.00+)

Estimated Hourly Profit (Legacy): 1.2–2.4 SOL/hr (≈ $200–$400/hr)

Legacy model was based on theoretical assumptions. Simulated results now offer a more grounded profit projection.

Operational Philosophy

Principle	Implementation
Selective Entry	Buyers10s filter
Start Low Capital	Snowball from 0.1–0.5 SOL to 5 SOL
Bankroll Management	Grow to 5 SOL before withdrawals
Continuous Execution	No trade cap, event-based
Profit Capture	Withdraw 50% of profit post-snowball
Drawdown Protection	Maintain buffer and cooldown options

📈 Development Roadmap: From Research to Real Profits

To ensure safe and profitable bot deployment, we follow a phased development lifecycle:

Stage 1: Historical Trading (Backtesting)

- 🔍 **Purpose:** Validate trading logic against prior Pump.fun token launches.
- 💾 **Method:** Analyze past token data using stored blockchain snapshots.
- 📊 **Goal:** Identify patterns, refine triggers (e.g., Buyers10s ≥ 5), and simulate theoretical profits.
- 🛠️ **Tools:** JSON datasets, Solana transaction history, randomized price modeling.

Stage 2: Simulated Trading (Paper Trading)

- 🧪 **Purpose:** Test the bot with *live* token launch data — but without risking real funds.
- 📉 **Method:** Monitor Pump.fun in real-time and log simulated trades using entry/exit logic.
- 📄 **Outcome:** Update walletState.json with paper profits, moonshots, partial exits, and skips.
- 💼 **Features:** Buyers10s tracking, price spike simulation, adjustable trade thresholds.

Stage 3: Live Trading (Real Capital)

- 🚀 **Purpose:** Transition to active trading with real SOL in a secured wallet.
- 🔄 **Method:** Execute trades using Jupiter Aggregator and @solana/web3.js.
- 🔒 **Controls:** Fee/slippage protection, reinvestment rules, cold storage integration.
- 💰 **Payouts:** Withdraw 50% of weekly net profits, maintain operating reserve, auto-log trades.

This structured approach ensures we validate and optimize before capital is risked, minimizing exposure and accelerating learning.

Testing Summary

Most Recent Run:

- **Timestamp**: 2025:05:20 15-28-06-375
- **Total Trades**: 500
- **Moonshots**: 85
- **Partial Exits**: 168
- **Net Profit (SOL)**: 25.4
- **Net Profit (USD @ $166)**: $4,216.40

Simulation used Advanced Exit Strategy v2 with fees and slippage.

Sample Moonshot Lifecycle Graphic

Sample Moonshot Lifecycle

Additional Considerations

A. Risks

- Honeypot tokens or scam contracts
- Liquidity locks and instant rugs
- RPC downtime or API limits
- Wallet blacklisting (anti-bot measures)

B. Future Enhancements

- Use real-time token charts via Jupiter or Helius
- Add wallet blacklists for known rug creators
- Portfolio heatmap and drawdown tracker
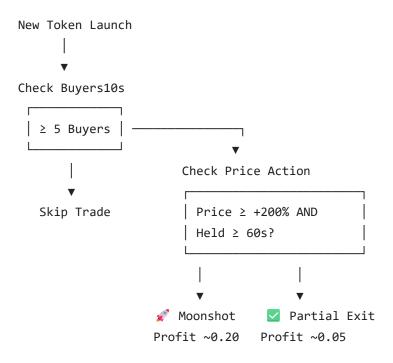- Dynamic position sizing based on bankroll health

C. Compliance

- All operations occur on decentralized exchanges
- No custodial control of user funds unless authorized
- System designed for personal/private deployment

📈 Lifecycle of a Typical Trade

Trade Categories:

Category	Trigger Condition	Action Taken	Expected Outcome
❌ Skipped	Buyers in 10s < 5	No trade; move on	0 SOL profit
✅ Partial Exit	Buyers10s ≥ 5 AND either: • Price < +200% • Peak held < 60s	Early exit at +50% gain	~0.05 SOL profit
🚀 Moonshot	Buyers10s ≥ 5 AND: • Price ≥ +200% • Held ≥ 60s	Full ride to +200% gain	~0.20 SOL profit

🔄 Trade Flow Diagram (Textual)

```
New Token Launch
      │
      ▼
Check Buyers10s
   ┌─────────┐
   │ ≥ 5 Buyers │ ─────────────┐
   └─────────┘               │
        │                    ▼
        ▼            Check Price Action
   Skip Trade        ┌──────────────────┐
                     │ Price ≥ +200% AND │
                     │ Held ≥ 60s?       │
                     └──────────────────┘
                        │            │
                        ▼            ▼
                  🚀 Moonshot   ✅ Partial Exit
                  Profit ~0.20   Profit ~0.05
```

📊 Sample Trade Distribution (Based on 8 Runs)

Outcome	Avg % of Accepted Trades	Notes
❌ Skipped	~52%	Most tokens filtered early
✅ Partial Exit	~26%	Moderate price movement
🚀 Moonshot	~23%	High buyer + stable pump

📊 Sample Trade Distribution (Based on 8 Runs)

🔐 Wallet Architecture & Profit Management

1. Initial Capital Setup (Snowball Protocol)

- **Startup Capital:** Begin with **0.5 SOL** to initiate the snowball protocol.
- **Source of Funds:** Acquire SOL from an exchange like *Crypto.com* or via a pre-funded Phantom wallet.
- **Initial Buy-In:** Trades will enter with **0.05–0.1 SOL** during the snowball phase.
- **Compounding Goal:** Grow to **5 SOL** before initiating withdrawals.

2. Wallet Roles & Flow

Wallet	Purpose	Details
🎯 Hot Wallet	Active trading via bot	Receives 50% of weekly profits; automated via hourly cron job with email confirmation
🏦 Payout Wallet	Profit extraction	Automated long-term transfer; includes hourly confirmation via email logging
🧊 Cold Storage	Backup & profit preservation	Manual or automated transfers for long-term safekeeping

3. Fund Transfer Logic

- Trading bot uses the **Hot Wallet** exclusively for operations.
- Once bankroll surpasses 5 SOL:
  - Withdraw 50% of weekly net profit to the **Payout Wallet**
  - Optional: Allocate a portion to **Cold Storage**
- If **Hot Wallet** drops below 2 SOL, manual or conditional refill is permitted from Cold Storage.

4. Phantom Wallet Usage

Using a **Phantom Wallet** is highly recommended for this system:

- Native to the Solana blockchain
- Seamless integration with `@solana/web3.js` and Jupiter Aggregator APIs
- Browser and mobile support for on-the-go management

Alternatively, users may implement **Solflare**, **Ledger**, or similar Solana-compatible wallets for secure deployment.

---

5. Automation & Monitoring

- A **cron job** runs hourly to:
    - Transfer profits from Hot Wallet to Payout Wallet (50% of gains)
    - Transfer a portion to Cold Storage (if balance exceeds thresholds)
- Each transfer event sends a **confirmation email** to the operator for monitoring and recordkeeping.
- Failures or skipped cycles should trigger alerts to ensure reliability.

Conclusion

This project demonstrates a viable, automated trading bot capable of generating consistent profits on meme coin launches. Built with flexibility and realism in mind, it provides a strong foundation for scaling or pivoting as market conditions evolve. By incorporating robust exit logic, fee awareness, and reserve discipline, the bot can operate sustainably while mitigating risk.

Appendix

- Key Libraries: `@solana/web3.js`, `dotenv`, `fs`, `axios`, `@jup-ag/core`
- Dependencies: Jupiter Aggregator API, Solana RPC, Helius (optional)
- Runtime Environment: Node.js 18+, secure VPS or dedicated server