StrategyFlow

Visual Low-Code Strategy Builder for Kraken

Kraken Forge Hackathon 2025

Track #3: Strategy Builder

Author: Maurice Boendermaker

Live demo: https://strategyflow.dev

1. Summary **3**
2. Problem statement **4**
Solution approach 4
3. Key features **5**
4. Technical architecture **6**
Technology stack 6
Design decisions 6
5. Block reference **7**
Order actions 7
Other block categories 7
6. Installation & Setup **7**
Prerequisites 7
Installation steps 8
Environment variables 8
7. User guide **9**
How to get started 9
8. API integration **9**
Authentication flow 9
Demo note 9
9. Screenshots **10**
10. Future enhancements **14**
Scalability considerations 14
11. License **15**

1. Summary

StrategyFlow is a production-quality, low-code trading strategy builder designed for Kraken. It enables traders to visually compose, simulate, and execute strategies using drag-and-drop blocks. No coding required. Built with React, Next.js, and TypeScript, it integrates directly with Kraken's authenticated API endpoints.

Table 1.1 - Overview of important details about the project

Item	Details
Track	#3 - Strategy Builder
Tech stack	React 19.2, Next.js 16, TypeScript 5, React Flow, Tailwind CSS 4
Live demo	https://strategyflow.dev
License	MIT License
Author	Maurice Boendermaker

In the table below is a list of all available page URLs (routes) within the application. They also work on localhost when the app is running. If running locally is preferred, replace the domain name with http://localhost:3000/

Table 1.2 - Overview of all available URL routes in the application

Page	Route
Landing page	https://strategyflow.dev/
About	https://strategyflow.dev/about
Docs	https://strategyflow.dev/docs
Editor	https://strategyflow.dev/editor
Export	https://strategyflow.dev/export
Settings	https://strategyflow.dev/settings

Simulator	https://strategyflow.dev/simulator
Strategies	https://strategyflow.dev/strategies
Templates	https://strategyflow.dev/templates
Validation	https://strategyflow.dev/validation
Version history	https://strategyflow.dev/version-history

2. Problem statement

Building automated trading strategies on crypto exchanges presents significant challenges that prevent many traders from effectively implementing their ideas:

- **Technical barriers**: Traditional strategy development requires extensive programming knowledge in languages like Python or JavaScript, excluding non-technical traders.
- **API complexity**: Kraken's API authentication requires proper HMAC-SHA512 signature generation, nonce handling, and secure credential management.
- **No visual feedback**: Code-based strategies lack visual representation, making it difficult to understand, debug, and communicate strategy logic.
- **Testing difficulties**: Safely testing strategies before live deployment typically requires custom simulation infrastructure.

Solution approach

StrategyFlow addresses these challenges by providing a visual interface where traders can drag and drop pre-built blocks that abstract away API complexity, handle authentication automatically, and provide simulation capabilities before live execution.

3. Key features

Table 3.1 - Core features and their descriptions

Feature	Description
Visual editor	Drag-and-drop node editor powered by React Flow with intuitive block connections and real-time visual feedback
Kraken API integration	Full integration with AddOrder, CancelOrder, AmendOrder, and batch operations using secure HMAC-SHA512 authentication
Execution simulator	Step-by-step dry-run mode with block highlighting, input/output inspection, and error visualization
Template gallery	Pre-built strategies: DCA, Stop-Loss, Grid Trading, RSI Momentum, and more for quick start
Inspector panel	Configure block parameters with validation, real-time preview, and contextual help
Flow debugger	Inspect inputs, outputs, errors, and execution timing for each node during runs
Export system	Export strategies as JSON definitions or TypeScript SDK code skeletons for integration

4. Technical architecture

Technology stack

- **Front-end**: React 19.2 with Next.js 16 for server-side rendering and API routes
- **Type safety**: TypeScript 5 with strict mode enabled for compile-time guarantees
- **Visual editor**: React Flow for node-based canvas with virtualization support
- **Styling**: Tailwind CSS 4 for utility-first styling, Radix UI for accessible components
- **Authentication**: crypto-js for HMAC-SHA512 signature generation per Kraken API specs

Design decisions

- **Modular components**: Each node type (Order, Condition, Logic, Utility) is a separate file for maintainability and reusability
- **Server-side API routes**: Kraken credentials handled in Next.js API routes, never exposed to client-side code
- **Type-safe blocks**: TypeScript discriminated unions for exhaustive pattern matching across block types
- **Removable landing page**: Demo landing page can be easily removed for production integration

5. Block reference

Order actions

Table 5.1 - Available order blocks and their corresponding Kraken API endpoints

Block	Description	Kraken endpoint
Place order	Submit buy/sell order	`/0/private/AddOrder`
Cancel order	Cancel existing order by ID	`/0/private/CancelOrder`
Amend order	Modify existing order params	`/0/private/AmendOrder`
Batch add	Submit multiple orders	`/0/private/AddOrderBatch`
Batch cancel	Cancel multiple orders	`/0/private/CancelOrderBatch`

Other block categories

- **Conditions**: Price thresholds, balance checks, time-based triggers, portfolio percentage conditions
- **Utilities**: Delay, log output, calculate values, format data, variable storage
- **System**: Start/End markers, loop constructs, error handlers, webhooks

6. Installation & Setup

Prerequisites

- Node.js 18+ and npm
- Kraken Pro account with API credentials
- Git for cloning the repository

Installation steps

```
# Clone the repository
git clone

https://github.com/MauriceBoendermaker/kraken_forge_hackathon_2025.git

cd kraken_forge_hackathon_2025/front-end

# Install dependencies
npm install

# Configure environment
cp .env.example .env

# Start development server
npm run dev
```

Environment variables

Table 6.1 - Required environment variables for .env

Variable	Description
KRAKEN_API_KEY	Your Kraken API public key
KRAKEN_PRIVATE_KEY	Your Kraken API private key (base64 encoded)
API_URL	https://api.kraken.com

7. User guide

How to get started

1. **Open the editor**: Navigate to `/editor` to access the visual strategy builder
2. **Add blocks**: Drag blocks from the sidebar categories (Order Actions, Conditions, Utilities, System)
3. **Connect nodes**: Click and drag between node handles to define execution flow
4. **Configure parameters**: Select a node and use the Inspector panel on the right to set values
5. **Test strategy**: Use the Simulator at `/simulator` to dry-run before live execution
6. **Execute**: Click "*Run Strategy*" to execute against Kraken's live API

8. API integration

Authentication flow

StrategyFlow implements Kraken's authentication protocol using HMAC-SHA512 signatures. The flow is handled server-side via Next.js API routes, ensuring credentials are never exposed to the browser. Each request includes a unique nonce (timestamp), API key, and computed signature.

Demo note

Important: The live demo at strategyflow.dev connects to a Kraken Pro account *without* a funded balance for safety. When executing orders, it will receive "Insufficient funds" errors. This is expected and confirms the API integration is working correctly. For production use, add funds to your Kraken account.

9. Screenshots

The following screenshots demonstrate StrategyFlow's key interfaces and features.

1. The landing page introduces StrategyFlow with a clean, dark-themed design featuring animated elements.

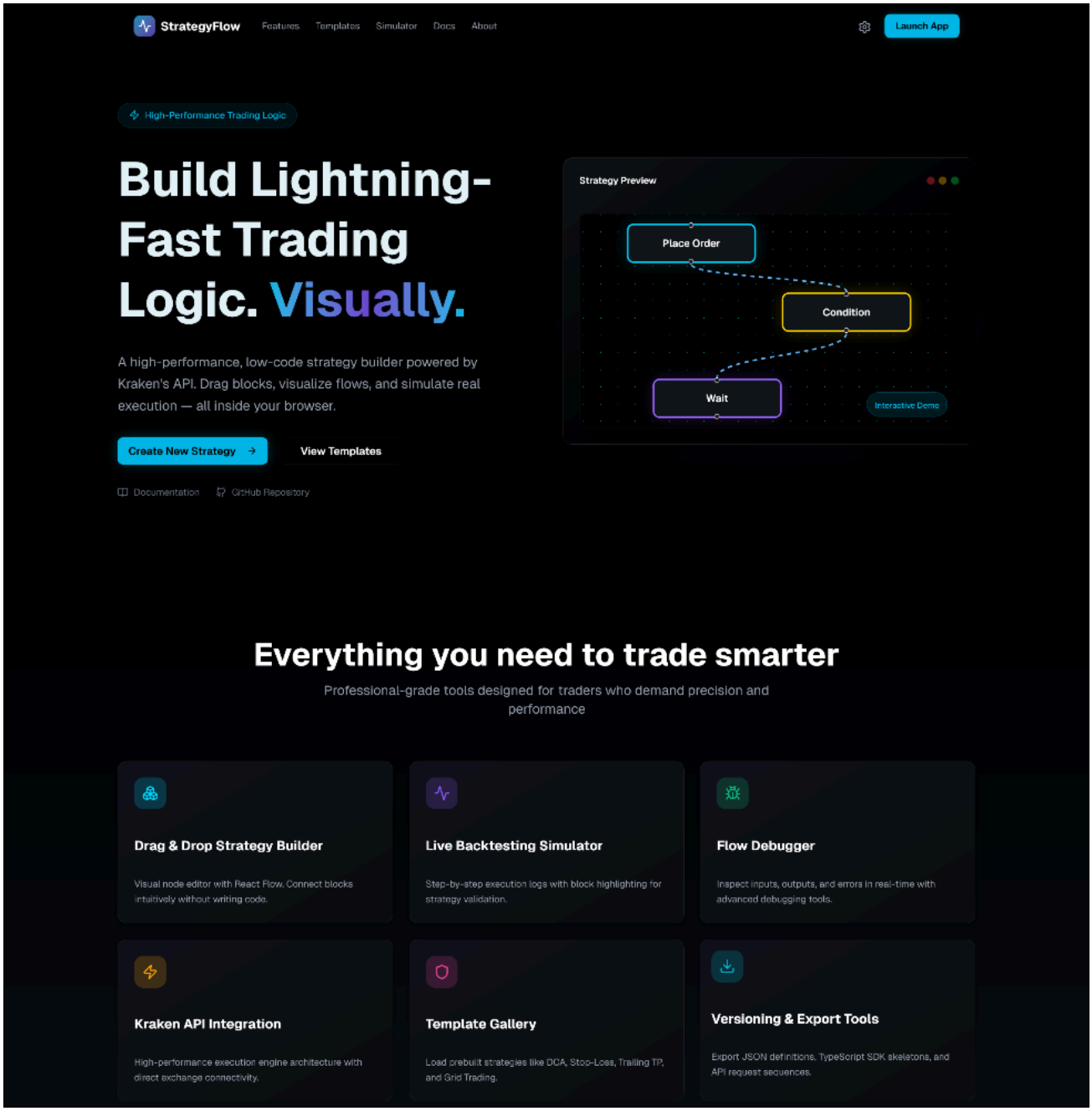

Figure 9.1 - Landing page

2. The main editor interface shows the node canvas with sidebar, inspector panel, and toolbar.

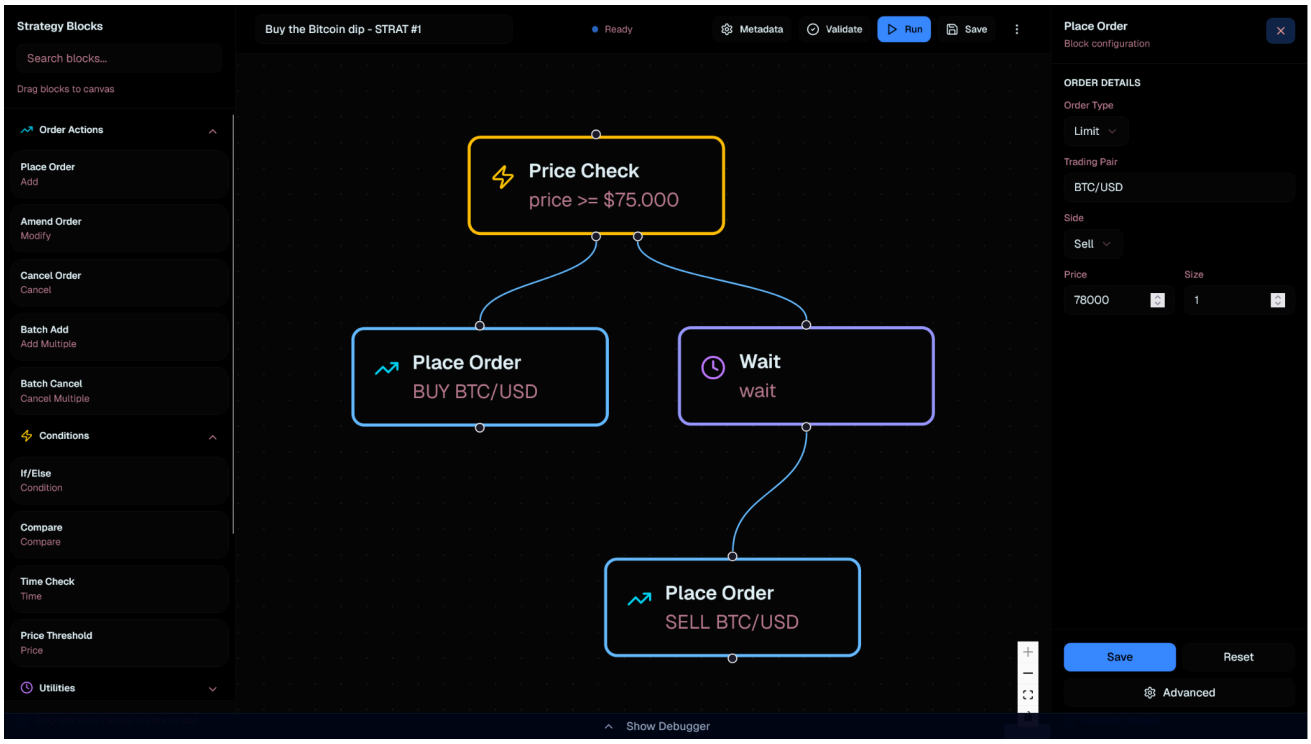

Figure 9.2 - Strategy editor

2.1 The main editor interface with the inspector panel opened.

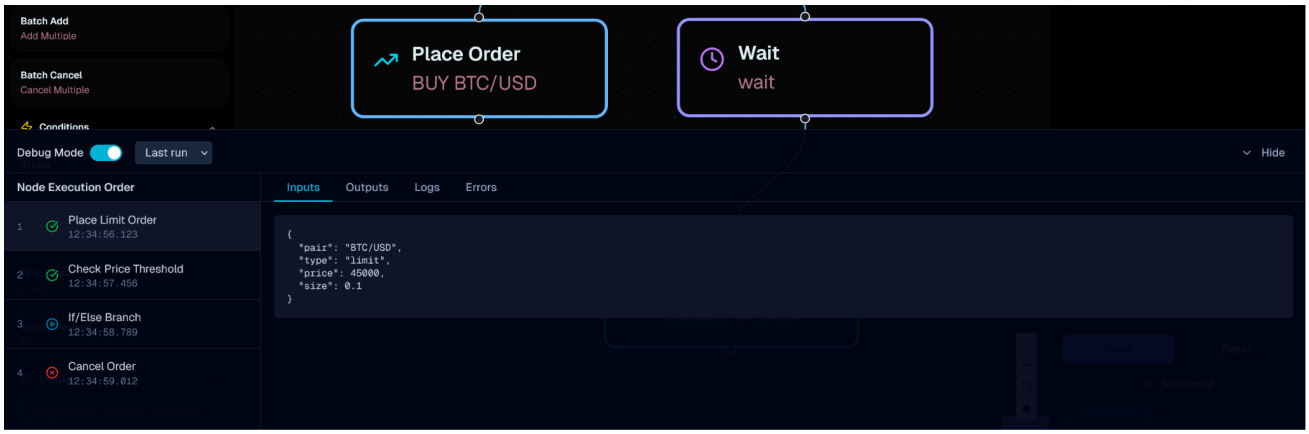

Figure 9.3 - Strategy editor with inspector panel

3. Pre-built strategy templates available for quick start, including DCA, Stop-Loss, and Grid Trading.

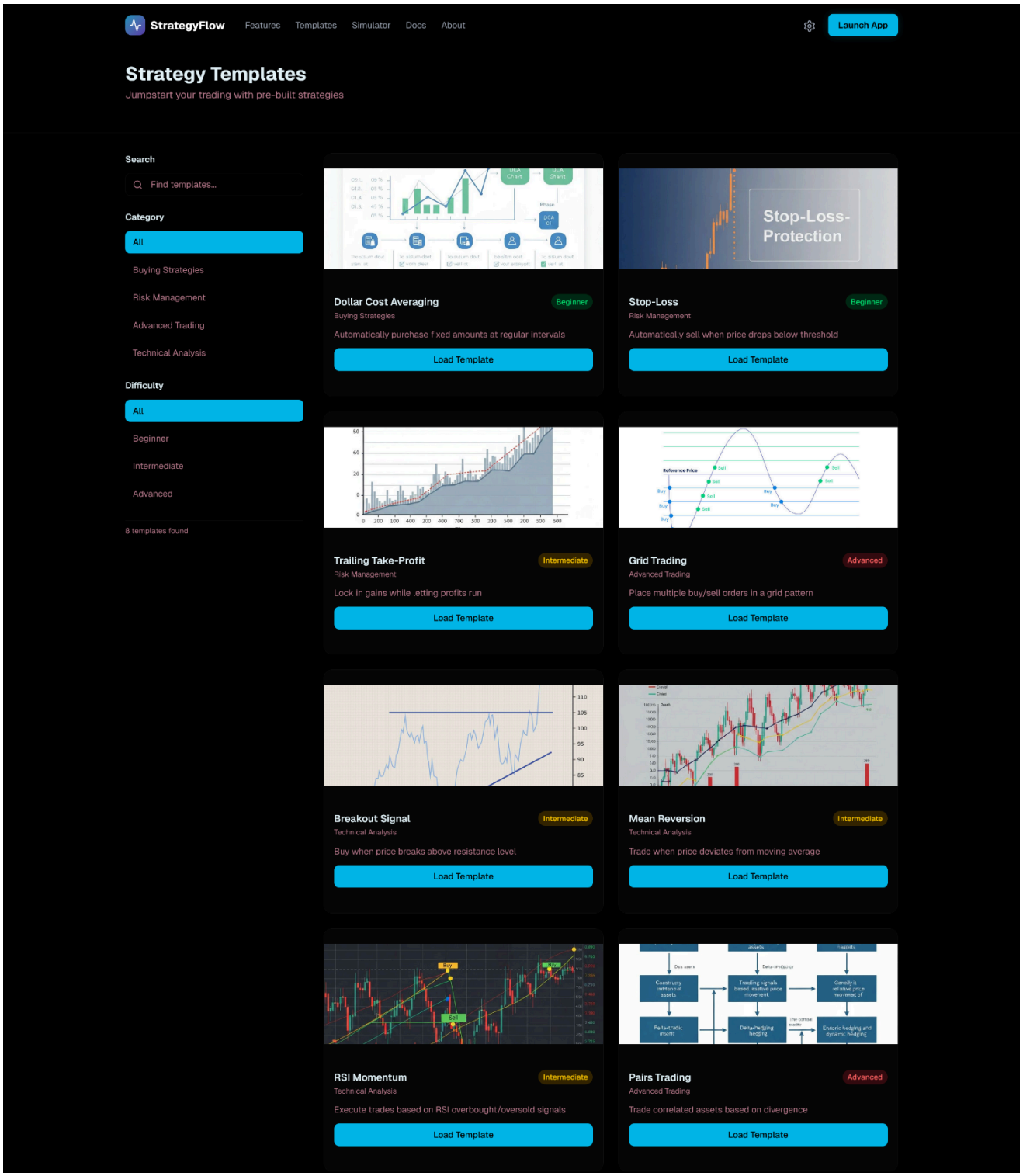

Figure 9.4 - Template gallery

4. Step-by-step simulation mode with block highlighting and I/O inspection for debugging.

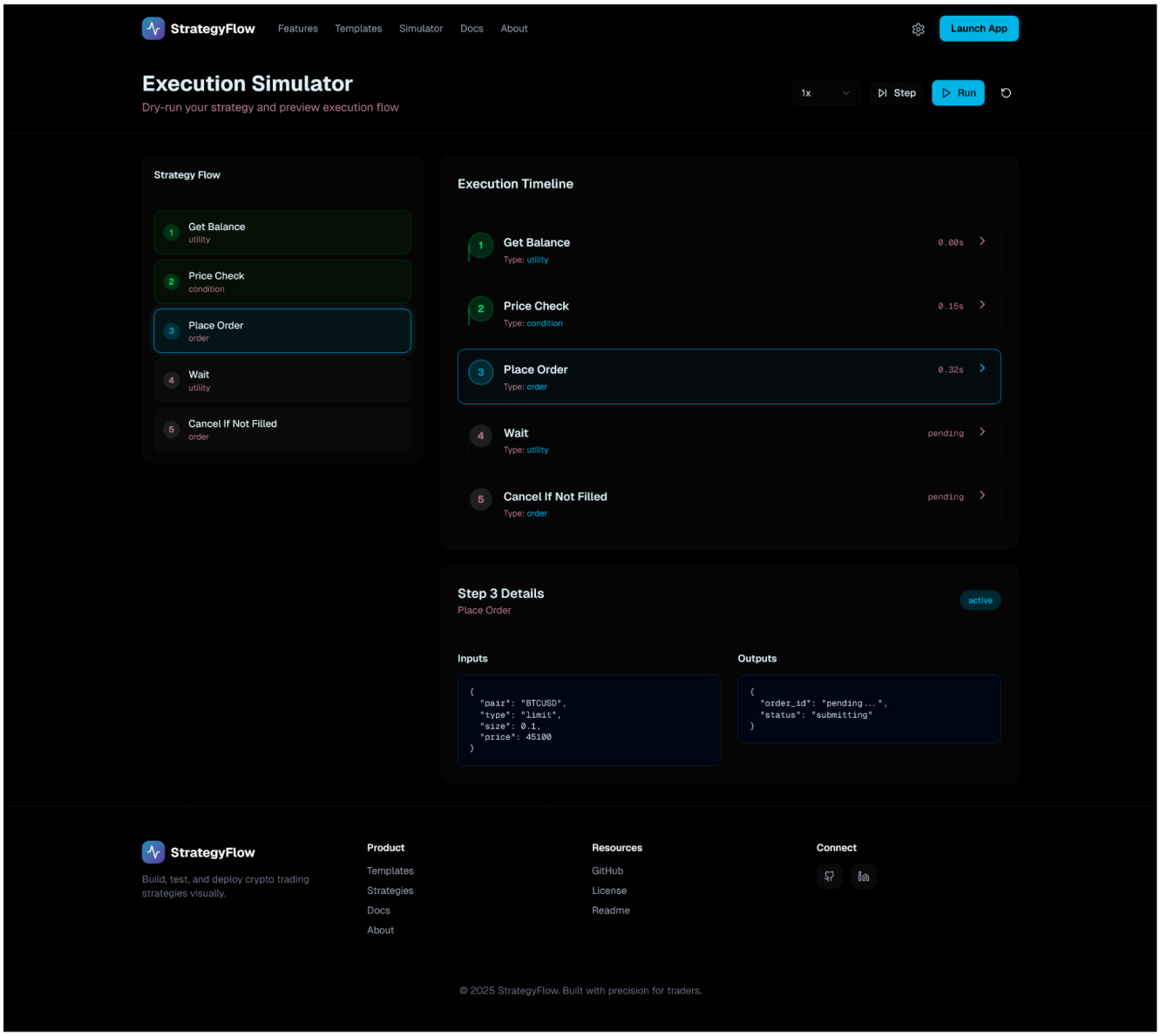

Figure 9.5 - Execution simulator

10. Future enhancements

The following features are planned for future development:

- **OpenAI integration**: Natural language strategy creation where users can type requests like "Create a DCA strategy for BTC every Monday" and have blocks automatically generated and placed on the canvas.
- **WebSocket Real-time data**: Live orderbook and ticker feeds displayed directly within the editor for market-aware strategy building.
- **Backtesting engine**: Historical data simulation with performance metrics, win rate analysis, and risk assessment.
- **Strategy marketplace**: Community platform to share, discover, and remix trading strategies built by other users.

Scalability considerations

The modular architecture supports scaling through microservice extraction of the strategy engine, database-backed strategy persistence, and queue-based execution for long-running strategies.

11. License

This project is released under the MIT License as required by the Kraken Forge Hackathon rules.

Thank you for reviewing StrategyFlow

Built with care for Kraken Forge Hackathon 2025

Track #3: Strategy Builder

https://strategyflow.dev