International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Quantitative Strategy Development and Portfolio Management Platform with Regime-Aware Multi-Stage Validation for Indian Equities

Shrey Pathre[1,*], Siddharth Roman[2], Om Singh[3], Raj Nagvekar[4], Neha Kudu[5]

[1–4]Students, Department of Information Technology, Vidyalankar Institute of Technology (University of Mumbai), Mumbai, India

[5]Project Mentor, Department of Information Technology, Vidyalankar Institute of Technology (University of Mumbai), Mumbai, India

*Corresponding author: shrey.pathre@vit.edu.in

Abstract—Trading strategies that look profitable in historical simulation often fail on unseen data because of overfitting, look-ahead bias, data leakage and changing market conditions. This paper presents S-RABLE (Systematic Regime-Aware Backtesting and Live-Simulation Environment), a web-based research platform for Indian equities that brings strategy development, market-regime classification, multi-stage validation and simulated portfolio management into one auditable workflow. Strategies are built from five families, a six-step no-code builder or sandboxed Python code, and are tested bar by bar with a simulated broker that charges slippage and brokerage. Regimes are described by a Gaussian mixture model selected by the Bayesian Information Criterion and by Random Forest and XGBoost classifiers trained on 12 causal features with purged, chronological splits. A five-stage pipeline applies explicit thresholds, and only validated strategies can receive simulated capital. On 125 out-of-sample NIFTY 50 days, the volatility and risk-sentiment classifiers reached 61.6% and 79.2% accuracy against majority baselines of 43.2% and 75.2%, whereas the trend classifier (27.2%) did not beat its baseline. The software passed 1,334 backend, 464 front-end and 330 end-to-end tests. No strategy is claimed to be profitable, and the platform never places real orders.

Keywords: Algorithmic trading, backtesting, market regime classification, walk-forward validation, Monte Carlo simulation, paper trading.

I. INTRODUCTION

Quantitative investing expresses an investment hypothesis as an explicit rule and evaluates it against historical market data. Running such an evaluation is cheap, but interpreting it is hard. A strategy that looks profitable in a simulation frequently fails on unseen data, and the usual causes are well documented: backtest overfitting [1], [2], data snooping [3], [4], information leakage between training and test periods [5], [6], selection among many trials [7], [8], and performance that depends on the market regime [9], [10], [11].

In practice, the research workflow is scattered across data downloads, notebooks, spreadsheets and separate charting or brokerage tools. Each hand-off between tools is a place where future prices can enter an indicator calculation, where parameters can be tuned on the same data used to judge them, or where a result can no longer be reproduced. These risks are higher for students, retail investors and small advisory teams, who typically lack institutional research infrastructure.

The Indian market adds its own requirements. A platform must follow the National Stock Exchange (NSE) session calendar and holidays [12], use the NIFTY 50 as a benchmark, and measure risk-adjusted return against an Indian risk-free rate rather than a US one. It must also serve users with different programming skills. Commercial visual platforms [13], [14] and international research platforms [15] exist, but the authors are not aware of one that links a leakage-aware regime model, threshold-based staged validation and validated-only portfolio allocation for NSE equities in a single stored and auditable workflow.

This paper presents S-RABLE (Systematic Regime-Aware Backtesting and Live-Simulation Environment), a web-based research environment for NSE-listed equities. The contribution is architectural and evidential, not a new trading algorithm. The main contributions are:

1) a layered architecture that combines five built-in strategy families, a six-step no-code rule builder and a three-layer sandbox for user-written Python strategies;

2) a regime engine that pairs an unsupervised Gaussian mixture model with supervised Random

Forest and XGBoost classifiers, trained on causal features with purged chronological splits;

3) a five-stage validation pipeline with explicit acceptance thresholds whose verdict gates capital allocation in simulated portfolios; and

4) measured evidence, including a negative result: automated test outcomes and out-of-sample regime-classification accuracy, reported exactly as computed.

The scope is deliberately limited. The platform is a simulator and never places a real order, and no strategy is claimed to be profitable. The rest of the paper is organised as follows. Section II reviews related work, Section III describes the methodology, Section IV reports results, Section V discusses limitations and threats to validity, and Section VI concludes.

II. LITERATURE REVIEW

A. Strategy families

Momentum strategies buy past winners and sell past losers [16], and the effect has been documented across asset classes in time-series form [17]. Mean reversion takes the opposite view, that stretched prices return toward an average [18]. Relative-value methods trade the divergence between related securities, as in distance-based pairs trading [19] and factor-residual statistical arbitrage [20]. Tactical asset allocation rotates exposure among assets according to simple trend or momentum signals [21], and volatility-oriented methods rely on the persistence of volatility captured by conditional-heteroskedasticity models [22], [23]. Practitioner texts describe how these ideas become tradable rules [24]. These studies motivate the five families in S-RABLE but do not by themselves provide a research workflow.

B. Backtest overfitting and validation

Bailey et al. [1] show that a modest number of trials can produce an impressive in-sample result by chance, and propose measuring the probability of backtest overfitting [2]. White's reality check [3] and the bootstrap study of technical trading rules [4] formalise data snooping, while the deflated Sharpe ratio [7] corrects for selection bias and non-normal returns. Harvey and Liu [25] and Harvey et al. [8] argue for higher evidence thresholds when many strategies are tried, and Lo [26] describes the sampling uncertainty of the Sharpe ratio [27]. For time-ordered data, ordinary cross-validation leaks information, so chronological, purged or walk-forward designs are preferred [28], [29], [6], [30]. Leakage itself is a general data-mining hazard [5]. Resampling methods such as the bootstrap [31] and its block variants [32] support significance testing of strategy returns.

C. Market regimes and machine learning

Regime-switching models date to Hamilton [9], and regime awareness has been shown to matter for asset allocation [10], [33] and dynamic strategies [11]. Unsupervised mixture models fitted by expectation–maximisation [34], [35] with model order chosen by the Bayesian Information Criterion [36] offer a label-free description of market states. Supervised learners, including random forests [37] and gradient-boosted trees [38], have been applied to return and signal prediction [39], [40], [41]. Reported predictive accuracy in this literature is sensitive to evaluation design, which is why S-RABLE reports each classifier against a majority-class baseline.

D. Existing platforms and research gaps

Platforms such as QuantConnect [15], Zerodha Streak [13] and Tradetron [14] provide research infrastructure, APIs or visual strategy building. The comparison here is qualitative, and no claim of quantitative superiority is made. Four gaps motivate this work. The geographic gap: advanced frameworks target US markets, so the NSE calendar, the NIFTY 50 and an Indian risk-free rate are rarely defaults. The accessibility gap: most quantitative research requires programming, so a no-code and a Python path to the same engine serve more users. The validation gap: a single backtest is weak evidence, and few tools enforce sequential, threshold-based validation. The integration gap: strategy configuration, execution records, orders, trades, positions and portfolio analytics are seldom held in one domain model, which limits auditability.

III. METHODOLOGY

A. System architecture

S-RABLE has five layers (Fig. 1). The presentation layer is a single-page application written with React 19 and TypeScript [42], using Lightweight Charts [43] for candlestick, equity and drawdown charts and TanStack Query [44] for server state. The API layer uses FastAPI [45] with Pydantic v2 validation [46] and exposes 124 REST endpoints in 25 routers. Authentication uses JSON Web Tokens signed with HS256 and a 30-minute lifetime, and passwords are hashed with bcrypt. The quantitative layer contains the strategy, backtest, regime, validation, scoring, portfolio and paper-trading engines, built on NumPy [47], pandas [48], SciPy [49], scikit-learn [50] and XGBoost [38]. A secure-execution layer isolates user code (Section III-H). Persistent data are held in PostgreSQL [51] through SQLAlchemy 2.0 [52], with schema changes managed by Alembic [53]; the database has 26 tables.

Resources are scoped to their owner. A user can read, change or delete only their own strategies, executions, orders and broker accounts. An execution record moves through PENDING, RUNNING, COMPLETED and FAILED states, and on start-up the service marks runs left stranded by a crash as failed. Orders follow a separate state machine, and every order, trade and position is stored so that a result can be regenerated from stored data and parameters.

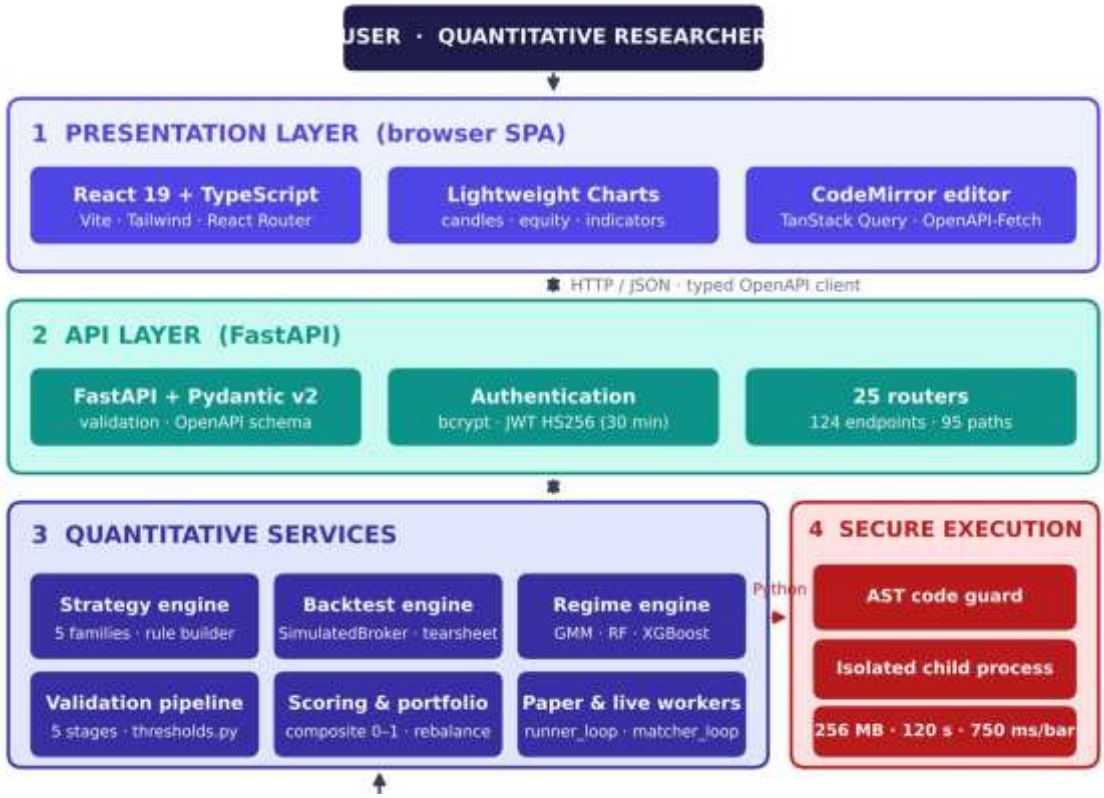


Fig. 1. System architecture of S-RABLE showing the presentation, API, quantitative-services and secure-execution layers.

B. Market data and chronological protocol

Daily open-high-low-close-volume bars are downloaded with yfinance [54] and aligned to the NSE trading calendar using exchange_calendars [55]. Stored bars pass integrity checks before use. All models and strategies see data in strict time order. Datasets are split chronologically into 70% training, 15% validation and 15% test, and a 20-trading-day purge gap is left before the validation and test segments so that forward-looking labels cannot overlap the next segment [6]. Within a backtest, the engine passes the strategy only the history up to and including the current bar, so signals cannot use future prices.

C. Event-driven backtesting and performance metrics

The backtest engine processes one bar at a time. A signal computed from data up to the close of bar t sets a target position q_t in fixed units, positive for long and negative for short. The simulated broker fills the order at the close of bar t, adjusted adversely by a configured slippage in basis points, and charges a brokerage fee proportional to traded value. With simple return

$$r_t = \frac{P_t}{P_{t-1}} - 1 \qquad (1)$$

the account is marked to market at every bar as

$$E_t = C_t + q_t P_t \qquad (2)$$

where C_t is cash and P_t the closing price. The compound annual growth rate (CAGR) uses a 365.2425-day year,

$$CAGR = (1 + R_{cum})^{\frac{1}{Years}} - 1 \qquad (3)$$

where R_{cum} is the cumulative return. The Sharpe ratio [27] is computed on excess returns over an annual risk-free rate R_f of 6.5%, the approximate yield of the Indian 10-year government security, converted to a per-period rate by

$$r_{f,period} = (1 + R_f)^{\frac{1}{A}} - 1 \qquad (4)$$

and then annualised with A = 252 trading periods per year:

$$SR = \sqrt{A} \cdot \frac{mean(r - r_{f,period})}{s} \qquad (5)$$

where s is the sample standard deviation of excess returns. The Sortino ratio [56] replaces s with the downside deviation. Drawdown measures the fall from the running equity peak M_t = max over u ≤ t of E_u [57]:

$$DD_t = \frac{M_t - E_t}{M_t}, \quad MDD = max(DD_t) \qquad (6)$$

The tearsheet also reports volatility, win rate, profit factor and the fraction of positive calendar years, and compares each run with the NIFTY 50 benchmark. Because the Sharpe ratio assumes one return per day, intraday equity curves should be resampled to daily returns before annualised ratios are interpreted.

D. Strategy families

Every strategy exposes a signal function returning BUY, SELL or HOLD, or target weights in the allocation case. Table I summarises the five built-in families, which implement the ideas reviewed in Section II-A in deliberately simple form. The rolling z-score used by mean reversion and statistical arbitrage is

$$z_t = \frac{P_t - \mu_t(n)}{\sigma_t(n)} \qquad (7)$$

with the mean and standard deviation taken over the last n bars (default 20).

TABLE I. BUILT-IN STRATEGY FAMILIES AND SIGNAL RULES

Family	Signal rule
Momentum	BUY if close > SMA(20), SELL if close < SMA(20), else HOLD
Mean reversion	BUY if z ≤ −entry; SELL if z ≥ +entry (20-bar z-score)
Volatility arbitrage	Spread = σ5 − σ20 of log returns; BUY if spread < 0, SELL if > 0
Statistical arbitrage	Spread of two assets; z ≤ −entry: buy A, sell B; z ≥ +entry: reverse; 1:1 legs
Tactical allocation	Rank assets by 20-bar relative momentum; hold top K at weight 1/K
Rule builder	User-defined indicator rules (SMA, EMA, RSI, ATR, MACD) saved as a JSON configuration
Python sandbox	User code run under the sandbox limits of Section III-H

The no-code builder is a six-step wizard (basics, parameters, indicators, entry conditions, exit conditions, review) that saves a configuration executed by the rule engine. The Python path opens an in-browser editor whose source is checked by the static guard before it can be saved.

E. Market-regime classification

Two complementary methods describe market state. The unsupervised method fits a Gaussian mixture model to three features (log return, realised volatility and trend strength) by expectation–maximisation [34], [35]. The number of components K is searched over 2, 3 and 4 and chosen by the Bayesian Information Criterion [36],

$$BIC = p \ln n - 2 \ln \hat{L} \qquad (8)$$

where p is the number of free parameters, n the number of observations and $\hat{L}$ the maximised likelihood. The posterior probability of each component is shown to the user as the probability of each regime.

The supervised method trains three classifiers on the same 12 causal features: log returns over 5, 20 and 60 days; rolling volatility over 10, 20 and 60 days; 20-day trend strength; distance from the 252-day high; a 60-day volume z-score; and three India VIX features (level, 20-day change and 252-day z-score). Labels look 20 trading days ahead. The volatility label is high when the forward annualised volatility exceeds the training-period median. The trend label is bull if the 20-day return exceeds +2%, bear if it is below −2% and flat otherwise. The risk-sentiment label is risk-off if the index falls more than 5% from the current close at any point in the window, or if average India VIX in the window exceeds its trailing 252-day median. In the default mode each target trains a Random Forest [37] and an XGBoost model [38] with class weights, selects hyperparameters by purged three-fold walk-forward cross-validation on the training split, keeps the model with the better validation balanced accuracy, and evaluates it exactly once on the test split. Accuracy is

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (9)$$

and, because classes are imbalanced, balanced accuracy, the majority-class baseline and a persistence baseline are stored with each model. The interface labels the output "Model classification, not a forecast".

F. Five-stage validation pipeline

Evidence for a strategy accumulates through five gates (Fig. 2, Table II). The thresholds are configured acceptance criteria, not results. Stage 1 runs a full-period backtest. Stage 2 divides history into at least four windows and measures performance retention, the ratio of out-of-sample to in-sample Sharpe ratio [30]:

$$Retention = \frac{Sharpe_{OOS}}{Sharpe_{IS}} \qquad (10)$$

Stage 3 slices the equity curve by regime, using the Gaussian-mixture and supervised labels, and computes the maximum drawdown within each regime. Stage 4 has two procedures. The bootstrap draws 1,000 resampled return paths, including block resampling that preserves serial dependence [31], [32], and derives a 95% value-at-risk of terminal return and drawdown. The sign-flip permutation

test randomly flips return signs to build a null distribution without directional edge, with p-value

$$p = \frac{1 + \#\{b: S*_b \geq S_{obs}\}}{B + 1} \qquad (11)$$

where B is the number of permutations. Stage 5 starts only from a validation that passed Stages 1 to 4. The strategy then processes new daily bars with simulated fills and no broker orders; the session skips missed days, never processes a day twice and respects exchange holidays. A verdict is tied to a version of the strategy definition, so editing the definition makes the verdict out of date, whereas renaming does not. Passing all gates shows only that a specified set of tests was passed. It does not guarantee profitability, and, because the number of trials is not yet recorded, multiple-testing corrections such as the deflated Sharpe ratio [7] are left as future work.



Fig. 2. Five-stage validation pipeline and configured acceptance criteria. A strategy that fails a stage is rejected; only validated strategies are eligible for capital.

TABLE II. VALIDATION STAGES AND CONFIGURED THRESHOLDS

Stage	Criterion
1 Historical backtest	Sharpe ≥ 1.0; max drawdown ≤ 20%; positive-return years ≥ 60%
2 Walk-forward	At least 4 windows; performance retention ≥ 60%
3 Regime stress	Max drawdown ≤ 25% in every regime
4 Monte Carlo	1,000 simulations; p-value ≤ 0.05; 95% VaR ≤ 25%
5 Paper trading	Retention ≥ 70%; zero loss-limit breaches

G. Strategy scoring and portfolio construction

A transparent score ranks strategies on the instruments of their most recent backtest. Three measures from stored daily closes (trend-friendliness, volatility and stability) are normalised and combined with weights that sum to one,

$$S = \Sigma_j w_j f_j, \ \Sigma_j w_j = 1 \qquad (12)$$

where f_j is the normalised measure and w_j its weight. The contribution of each measure is shown to the user. A rule table maps each family to the regimes in which it is expected to work (for example, momentum in a trending market and mean reversion in a sideways market) and flags each strategy as COMPATIBLE, NOT_COMPATIBLE or UNKNOWN; this is a documented heuristic, not a prediction.

The portfolio orchestrator selects scored candidates, filters them by regime compatibility, allocates capital in proportion to score and applies risk limits: a cash reserve, a maximum weight per strategy, a maximum weight per asset (exposures to the same asset in different strategies are summed) and a maximum invested fraction. By default only validated strategies can receive capital. A rebalance plan compares current holdings with targets at the latest stored closes and lists BUY, SELL or HOLD per asset; it is a calculation only, and nothing is saved or ordered until the user confirms. A portfolio backtest runs several strategies as sleeves, rebalances weekly or monthly and compares the result with buy-and-hold on the NIFTY 50, and reports turnover.

H. Sandbox for custom Python strategies

Running user code on a server carries remote-execution and resource-exhaustion risks, so execution is layered (Fig. 3). Layer 1 is a static guard that parses the source into an abstract syntax tree [58] without running it and rejects imports such as os, sys, socket and subprocess, calls to __import__, eval and exec, and file-write operations. Layer 2 runs the code in a separate child process with runtime audit hooks that report file or network access attempts, and enforces 256 MB of memory, 120 s of total run time and 750 ms per bar. Layer 3 applies operating-system containment: Windows Job Objects, Linux setrlimit [59] or a Docker container with all capabilities

dropped, a read-only file system and no network. The sandbox is a risk-mitigation measure, not a guarantee; static analysis of a dynamic language can be evaded, and resource limits reduce but do not remove denial-of-service risk. Code that passes the guard is then validated like any other strategy.

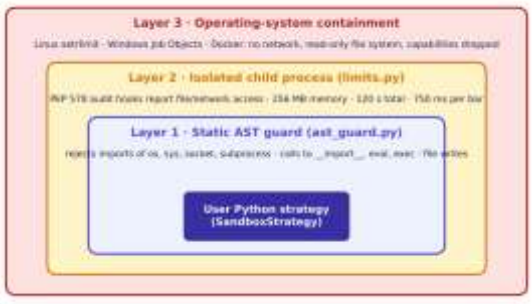


Fig. 3. Three-layer sandbox for custom Python strategies.

I. Delayed live paper trading and replay

The paper-trading module runs strategies on virtual money with intraday bars of 1, 5 or 15 minutes. Bars come from Yahoo Finance with a 15-minute delay, or from a replay of a recorded day at 1×, 5× or 10× speed when the market is closed. A background runner executes each active strategy as a new bar arrives, orders are written to a paper-order table, and a 60-second matcher fills them and updates holdings, cash and equity snapshots. Intraday positions are squared off at 15:20 IST, ten minutes before the 15:30 close. Delayed quotes are not an exchange feed, so latency, queue position and market impact are not modelled.

J. Bias taxonomy and controls

Table III maps each source of error discussed in Section I to the control S-RABLE applies and to the risk that remains. It makes explicit that no control is complete.

TABLE III. SOURCES OF ERROR, CONTROLS AND RESIDUAL RISK

Source	Control	Residual risk
Look-ahead bias	Signals use data up to the current bar; causal features	Errors in user-written code or indicators
Data leakage	Chronological split; 20-day purge; purged walk-forward CV	Label windows longer than the purge
Overfitting, snooping	Out-of-sample windows; Monte Carlo and permutation tests	Number of trials not recorded
Regime dependence	Regime labels; regime stress stage; regime-fit rules	Unstable labels; weak trend classifier
Survivorship	User picks from an instrument catalogue	Catalogue holds liquid, surviving stocks
Costs, slippage	Slippage (bps) and brokerage in simulated broker	No transaction tax, borrowing or margin
Execution realism	Delayed paper trading; calendar; 15:20 square-off	Fills at bar close; no order book
Faulty user code	Three-layer sandbox	Static analysis can be bypassed

IV. RESULTS

This section reports the evidence available for S-RABLE: the software-verification results, the implementation inventory and the measured out-of-sample accuracy of the regime models. The platform is evaluated as an engineered research system. Strategy-level return statistics are outputs that the platform computes for whatever instrument, period and parameters a user selects; they are not claims of this paper.

A. Experimental setup

The repository (branch develop, commit 0deee7a6) was audited on 21 October 2026 on a Windows 11 machine with Python 3.13.7, Node.js v22.19.0 and npm 10.9.3. The application database was PostgreSQL 15 hosted on Supabase, and backend tests ran against a local PostgreSQL instance only. Four NSE equities were present in the demonstration workspace (JSWSTEEL, RELIANCE, SBIN and TCS), and the stored SBIN reference series contained 6,682 daily bars from 3 January 2000 to 19 October 2026. Table IV lists the evaluation parameters.

TABLE IV. EVALUATION PARAMETERS

Parameter	Value
Momentum, mean-reversion lookback	20 bars
Default position size	10 units (fixed)
Risk-free rate (Sharpe, Sortino)	6.5% per year, compounded

Parameter	Value
Annualisation; CAGR year	252 periods; 365.2425 days
Data split; purge	70/15/15 chronological; 20 days
Gaussian mixture components	K = 2, 3, 4 chosen by BIC
Supervised models	Random Forest (100 trees); XGBoost (depth 3)
Live data delay; bar sizes	15 min; 1, 5 or 15 min
Replay speeds; square-off	1×, 5×, 10×; 15:20 IST
Sandbox limits	256 MB; 120 s total; 750 ms per bar
Access-token lifetime	30 min (HS256), no refresh token

B. Software verification

Three automated layers were used: Pytest for backend logic, Vitest for front-end components and hooks, and Playwright for complete user journeys in a real browser on desktop and mobile viewports. The Playwright suite has mocked journeys that run against recorded API fixtures, and real-backend gates, one per headline feature, that backtest template strategies on stored data and check that numbers on screen equal those returned by the API. The final full run on 21 October 2026 gave the results in Table V and Fig. 4.

TABLE V. AUTOMATED-TEST RESULTS OF THE FINAL FULL RUN

Layer	Tool; files	Result
Backend	Pytest; 113	1,334 passed, 1 skipped (Docker half of sandbox suite)
Front end	Vitest 3.2.4; 65	464 passed; type check and lint clean
End to end	Playwright 1.52.0; 27	330 passed, 1 failed (known timing flake), 33 skipped (mobile real-API runs)

The Playwright run took 35.6 minutes at four workers. Two serial tests that did not run after the failure were rerun on their own and passed. Two lessons are worth recording. First, a green run can hide skipped tests: three consecutive green runs were needed to reveal that real-backend tests for four features had been skipped because a three-second readiness check timed out while FastAPI built its OpenAPI schema. The check now allows 30 seconds, and skipped tests are counted on every run. Second, the real-backend gates found a genuine defect, in which idle connections to the Supabase pool were dropped; it was fixed with pool_pre_ping and pool_recycle. The test harness refuses to run unless the configured database host is local, so tests that create and delete data cannot touch the hosted database. These counts reflect verification effort only; they say nothing about strategy performance.

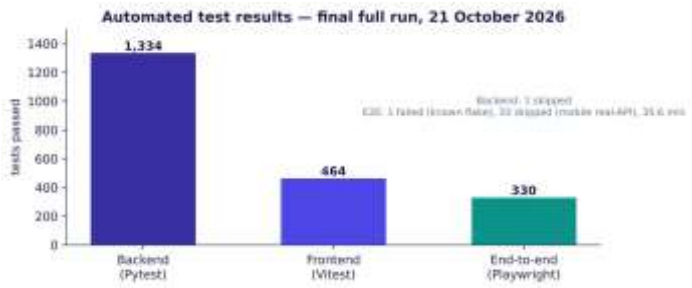


Fig. 4. Automated test results of the final full run, 21 October 2026.

C. Implementation inventory

The audit established the scale of the system (Table VI): 124 endpoints over 95 paths in 25 routers, a database of 26 tables with 342 columns, 45 foreign keys and 28 indexes, 21 front-end screens, five strategy families plus a rule engine and a sandbox adapter, and a regime layer with one Gaussian mixture model and three supervised classifiers.

TABLE VI. IMPLEMENTATION INVENTORY

Item	Count
REST routers / paths / endpoints	25 / 95 / 124
Tables / columns / foreign keys / indexes	26 / 342 / 45 / 28
Built-in strategy families	5 (plus rule builder, sandbox adapter)

Item	Count
Regime features (supervised / mixture)	12 / 3
Validation stages	5
Front-end screens	21
Backend / front-end / E2E tests passed	1,334 / 464 / 330

D. Regime-model evaluation

The supervised models were evaluated once on 125 out-of-sample NIFTY 50 test days, using the chronological split and purge of Section III-B. Table VII and Fig. 5 give test accuracy, balanced accuracy and the accuracy of always predicting the majority class.

TABLE VII. OUT-OF-SAMPLE PERFORMANCE OF THE SUPERVISED REGIME MODELS

Model	Test acc.	Balanced acc.	Majority baseline
Volatility (high/low), Random Forest	61.60%	58.22%	43.20%
Trend (bull/bear/flat), XGBoost	27.20%	25.68%	51.20%
Risk sentiment (on/off), XGBoost	79.20%	75.36%	75.20%
Mean of the three	56.00%	53.09%	56.53%

The results are mixed and are reported as measured. The volatility classifier exceeds its majority baseline by 18.4 percentage points, which is consistent with the well-known persistence of volatility [22], [23]. The risk-sentiment classifier reaches 79.2%, the only result above the 70% target set in the model card, but it is only 4.0 points above its majority baseline, so much of this accuracy reflects class imbalance. The trend classifier scores 27.2%, below both the 33% expected from guessing among three classes and its 51.2% baseline; it cannot be used to predict the market trend 20 days ahead. The mean accuracy of the three models (56.00%) is slightly below the mean of their baselines (56.53%).

With only 125 test days, the estimates are imprecise: at 60% accuracy the standard error is 4.4 percentage points, giving a 95% confidence interval of about ±8.6 points. The figures should therefore be read as indicative. For this reason the regime layer is presented as a descriptive classification and not as a forecast, and the unsupervised mixture model does not depend on prediction at all.

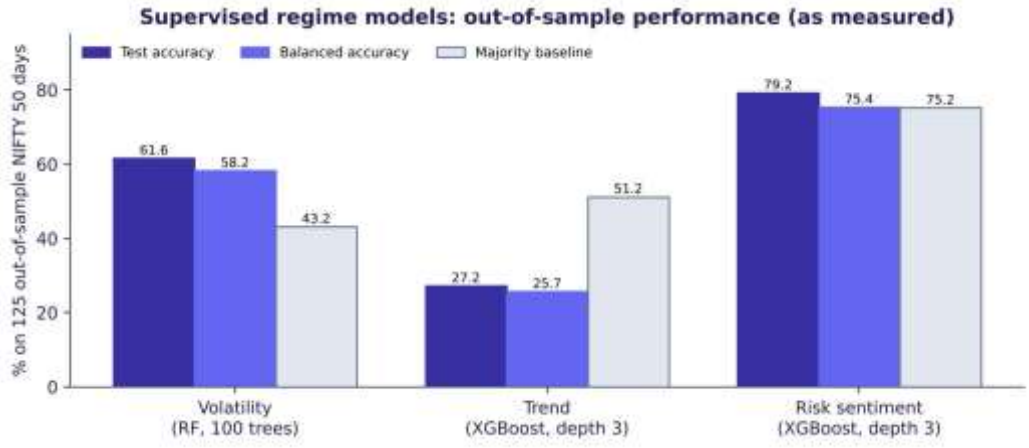


Fig. 5. Test accuracy, balanced accuracy and majority baseline of the supervised regime models on 125 out-of-sample NIFTY 50 days.

E. Application outputs

A completed backtest produces a run record, a tearsheet (cumulative return, CAGR, Sharpe, Sortino, maximum drawdown, win rate and profit factor), equity and drawdown charts, a comparison with the NIFTY 50, and a filterable, CSV-exportable chain of simulated executions, orders, trades and positions. The tearsheet of any run can be regenerated from stored data and parameters. As an example of how regime context qualifies a ranking without making a performance claim, the Momentum template on JSWSTEEL ranked first with a composite score of 0.537 (components 0.162, 0.154 and 0.220), yet it was flagged

as not compatible with the active regime, "R2 High volatility, sideways" (72.95%).

The Market Regime page (Fig. 6) shows the current classification, the probability of each regime and a regime history, labelled "Model classification, not a forecast". In the freshly created workspace used for the screenshots, SBIN was classified as "R1 Low volatility, sideways" with 92.57% confidence as of 1 January 2020. Strategy returns are intentionally not tabulated here, because they depend on user choices and no strategy is asserted to be profitable.

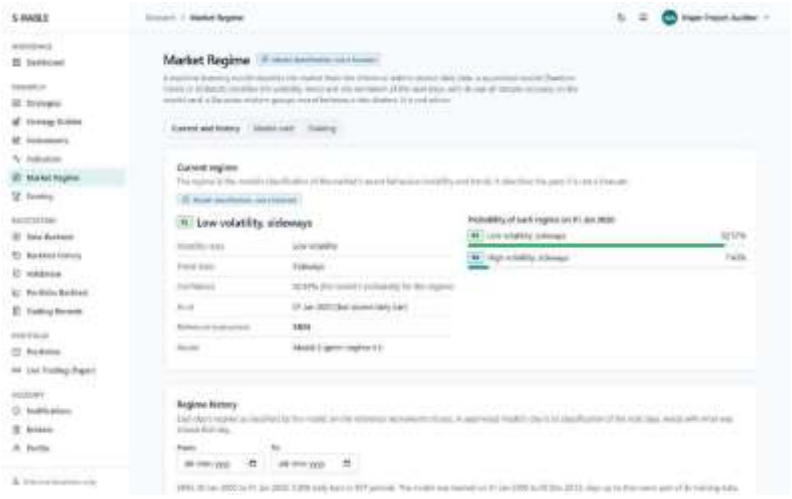


Fig. 6. Market Regime page of the S-RABLE interface: current classification, regime probabilities and regime history (fresh workspace, 20 October 2026).

V. DISCUSSION

A. Interpretation

The central finding is about process more than about performance. S-RABLE ties the usual research steps into one stored workflow with explicit thresholds, so a strategy cannot receive simulated capital without passing a sequence of out-of-sample, regime-conditioned and significance tests. Compared with ad-hoc notebooks, open-source backtesting libraries and commercial retail platforms, the platform's distinguishing feature is not any single analytical tool, which are common, but the integration of walk-forward windows, regime-linked stress testing, Monte Carlo and permutation tests, purged splits, sandboxed user code and validated-only allocation behind one interface. A numerical comparison with other platforms would require the same data, costs and strategies, which are not available, so the comparison is qualitative [15], [13], [14]. Whether this integration improves research decisions is an empirical question that needs a controlled user study.

The regime results show both the usefulness and the limits of the approach. Volatility is partly predictable from its own recent history, so the volatility classifier adds information over the baseline. The trend classifier does not, which agrees with the general difficulty of forecasting direction. Publishing this negative result is a deliberate choice: it shows where the regime layer is informative and where it is not, and it supports the decision to treat regimes as a descriptive context for evaluation.

B. Limitations

The audit identified ten limitations that bound the results. (1) Signals are filled at the close of the bar on which they are computed, with no next-bar delay. (2) Short positions are funded immediately, with no margin, borrowing cost or locate, and securities transaction tax and other statutory charges are not modelled. (3) Two Sharpe ratio implementations exist: the engine's step-level version uses a zero risk-free rate without annualisation, while the tearsheet uses $R_f = 6.5\%$ and $\sqrt{252}$. (4) The mean-reversion exit-threshold parameter is accepted but not used. (5) Statistical arbitrage assumes a 1:1 hedge ratio with no cointegration test [19]. (6) Tactical allocation lacks the 200-day trend filter of classical timing rules [21]. (7) The trend classifier underperforms its baseline. (8) Access tokens expire after 30 minutes with no refresh or revocation, and no CORS middleware is registered. (9) The XGBoost version is not pinned. (10) Delayed market data limits the realism of paper trading. Timing and throughput were not measured, so no claim is made about latency or concurrency.

C. Threats to validity

Construct validity: thresholds such as a Sharpe ratio of 1.0 are conventions, not proof of quality, and measure robustness only roughly. Internal validity: if any indicator, feature or label contains look-ahead or leakage, the results are biased; causal construction and purging reduce but do not prove the absence of such errors. External validity: the demonstration universe is four large NSE stocks and the regime models were tested on one index over 125 days, so results may not carry over to other markets, periods or frequencies. Conclusion validity: with few strategies and short test windows, statistical power is low, and because the number of trials is not recorded, multiple testing is uncontrolled. The software-test counts address none of these four threats.

VI. CONCLUSION

This paper presented S-RABLE, a web platform that unifies strategy development, regime classification, five-stage validation and simulated portfolio management for NSE equities. Its implementation comprises a React 19 front end, a FastAPI backend with 124 endpoints, a 26-table PostgreSQL database, five strategy families, a no-code builder, a three-layer sandbox, an event-driven backtester with slippage and brokerage, a regime engine combining a Gaussian mixture model with Random Forest and XGBoost classifiers, and delayed paper trading with replay. The software was verified by 1,334 backend, 464 front-end and 330 end-to-end test runs. On 125 out-of-sample NIFTY 50 days, the volatility and risk-sentiment classifiers exceeded their majority baselines (61.6% vs 43.2%, and 79.2% vs 75.2%), while the trend classifier did not (27.2% vs 51.2%).

The main limitations are bar-close fills, no transaction taxes, margin or borrowing costs, simplified statistical-arbitrage and allocation rules, a weak trend classifier, short-lived access tokens and delayed data. A passed validation is evidence that a defined set of tests was met, not a guarantee of profit, and the platform is a research and educational tool, not investment advice.

Future work will (i) add securities transaction tax, exchange charges, stamp duty, GST, margin and borrowing costs and report results gross and net of costs; (ii) offer next-bar-open execution and unify the two Sharpe implementations; (iii) add cointegration tests and hedge-ratio estimation to statistical arbitrage and a trend filter to tactical allocation; (iv) improve the trend classifier with longer histories, more features and probability calibration, and publish confusion matrices; (v) record the number of trials per user so that the deflated Sharpe ratio and the probability of backtest overfitting can be computed [7], [2]; (vi) compare the mixture model with hidden Markov models; (vii) add refresh tokens, token revocation and CORS configuration; and (viii) run extended out-of-sample and multi-month paper-trading experiments with confidence intervals. Real broker order routing would be considered only after compliance review, with paper trading remaining the default.

ACKNOWLEDGEMENT

The authors thank Dr. Neha Kudu, Department of Information Technology, Vidyalankar Institute of Technology, Mumbai, for guidance throughout the project, and the maintainers of the open-source tools on which the platform is built.

REFERENCES

[1] D. H. Bailey, J. M. Borwein, M. López de Prado, and Q. J. Zhu, “Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance,” Notices of the American Mathematical Society, vol. 61, no. 5, pp. 458–471, 2014.

[2] D. H. Bailey, J. M. Borwein, M. López de Prado, and Q. J. Zhu, “The probability of backtest overfitting,” Journal of Computational Finance, vol. 20, no. 4, pp. 39–69, 2017.

[3] H. White, “A reality check for data snooping,” Econometrica, vol. 68, no. 5, pp. 1097–1126, 2000.

[4] R. Sullivan, A. Timmermann, and H. White, “Data-snooping, technical trading rule performance, and the bootstrap,” The Journal of Finance, vol. 54, no. 5, pp. 1647–1691, 1999.

[5] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman, “Leakage in data mining: Formulation, detection, and avoidance,” ACM Transactions on Knowledge Discovery from Data, vol. 6, no. 4, pp. 1–21, 2012.

[6] M. López de Prado, Advances in Financial Machine Learning. Hoboken, NJ, USA: Wiley, 2018.

[7] D. H. Bailey and M. López de Prado, “The deflated Sharpe ratio: Correcting for selection bias, backtest overfitting, and non-normality,” The Journal of Portfolio Management, vol. 40, no. 5, pp. 94–107, 2014.

[8] C. R. Harvey, Y. Liu, and H. Zhu, “… and the cross-section of expected returns,” The Review of Financial Studies, vol. 29, no. 1, pp. 5–68, 2016.

[9] J. D. Hamilton, “A new approach to the economic analysis of nonstationary time series and the business

cycle," Econometrica, vol. 57, no. 2, pp. 357–384, 1989.

[10] A. Ang and G. Bekaert, "International asset allocation with regime shifts," The Review of Financial Studies, vol. 15, no. 4, pp. 1137–1187, 2002.

[11] M. Kritzman, S. Page, and D. Turkington, "Regime shifts: Implications for dynamic strategies," Financial Analysts Journal, vol. 68, no. 3, pp. 22–39, 2012.

[12] National Stock Exchange of India Ltd., "Trading hours, holidays and charges — equity segment." [Online]. Available: https://www.nseindia.com (accessed Oct. 20, 2026).

[13] Zerodha Streak, "Visual strategy building and backtesting." [Online]. Available: https://streak.tech (accessed Oct. 21, 2026).

[14] Tradetron, "Visual algorithmic trading platform." [Online]. Available: https://tradetron.tech (accessed Oct. 21, 2026).

[15] QuantConnect, "Algorithmic trading and quantitative research platform." [Online]. Available: https://www.quantconnect.com (accessed Oct. 21, 2026).

[16] N. Jegadeesh and S. Titman, "Returns to buying winners and selling losers: Implications for stock market efficiency," The Journal of Finance, vol. 48, no. 1, pp. 65–91, 1993.

[17] T. J. Moskowitz, Y. H. Ooi, and L. H. Pedersen, "Time series momentum," Journal of Financial Economics, vol. 104, no. 2, pp. 228–250, 2012.

[18] J. M. Poterba and L. H. Summers, "Mean reversion in stock prices: Evidence and implications," Journal of Financial Economics, vol. 22, no. 1, pp. 27–59, 1988.

[19] E. Gatev, W. N. Goetzmann, and K. G. Rouwenhorst, "Pairs trading: Performance of a relative-value arbitrage rule," The Review of Financial Studies, vol. 19, no. 3, pp. 797–827, 2006.

[20] M. Avellaneda and J.-H. Lee, "Statistical arbitrage in the U.S. equities market," Quantitative Finance, vol. 10, no. 7, pp. 761–782, 2010.

[21] M. T. Faber, "A quantitative approach to tactical asset allocation," The Journal of Wealth Management, vol. 9, no. 4, pp. 69–79, 2007.

[22] R. F. Engle, "Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation," Econometrica, vol. 50, no. 4, pp. 987–1007, 1982.

[23] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," Journal of Econometrics, vol. 31, no. 3, pp. 307–327, 1986.

[24] E. P. Chan, Algorithmic Trading: Winning Strategies and Their Rationale. Hoboken, NJ, USA: Wiley, 2013.

[25] C. R. Harvey and Y. Liu, "Backtesting," The Journal of Portfolio Management, vol. 42, no. 1, pp. 13–28, 2015.

[26] A. W. Lo, "The statistics of Sharpe ratios," Financial Analysts Journal, vol. 58, no. 4, pp. 36–52, 2002.

[27] W. F. Sharpe, "The Sharpe ratio," The Journal of Portfolio Management, vol. 21, no. 1, pp. 49–58, 1994.

[28] C. Bergmeir and J. M. Benítez, "On the use of cross-validation for time series predictor evaluation," Information Sciences, vol. 191, pp. 192–213, 2012.

[29] V. Cerqueira, L. Torgo, and I. Mozetič, "Evaluating time series forecasting models: An empirical study on performance estimation methods," Machine Learning, vol. 109, no. 11, pp. 1997–2028, 2020.

[30] R. Pardo, The Evaluation and Optimization of Trading Strategies, 2nd ed. Hoboken, NJ, USA: Wiley, 2008.

[31] B. Efron, "Bootstrap methods: Another look at the jackknife," The Annals of Statistics, vol. 7, no. 1, pp. 1–26, 1979.

[32] D. N. Politis and J. P. Romano, "The stationary bootstrap," Journal of the American Statistical Association, vol. 89, no. 428, pp. 1303–1313, 1994.

[33] M. Guidolin and A. Timmermann, "Asset allocation under multivariate regime switching," Journal of Economic Dynamics and Control, vol. 31, no. 11, pp. 3503–3544, 2007.

[34] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," Journal of the Royal Statistical Society: Series B, vol. 39, no. 1, pp. 1–38, 1977.

[35] C. M. Bishop, Pattern Recognition and Machine Learning. New York, NY, USA: Springer, 2006.

[36] G. Schwarz, "Estimating the dimension of a model," The Annals of Statistics, vol. 6, no. 2, pp. 461–464, 1978.

[37] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[38] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[39] C. Krauss, X. A. Do, and N. Huck, "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500," European Journal of Operational Research, vol. 259, no. 2, pp. 689–702, 2017.

[40] S. Gu, B. Kelly, and D. Xiu, "Empirical asset pricing via machine learning," The Review of Financial Studies, vol. 33, no. 5, pp. 2223–2273, 2020.

[41] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," European Journal of Operational Research, vol. 270, no. 2, pp. 654–669, 2018.

[42] "React documentation." [Online]. Available: https://react.dev (accessed Oct. 20, 2026).

[43] "Lightweight Charts documentation." [Online]. Available: https://tradingview.github.io/lightweight-charts/ (accessed Oct. 20, 2026).

[44] "TanStack Query documentation." [Online]. Available: https://tanstack.com/query (accessed Oct. 20, 2026).

[45] S. Ramírez, "FastAPI documentation." [Online]. Available: https://fastapi.tiangolo.com (accessed Oct. 20, 2026).

[46] "Pydantic documentation." [Online]. Available: https://docs.pydantic.dev (accessed Oct. 20, 2026).

[47] C. R. Harris et al., "Array programming with NumPy," Nature, vol. 585, pp. 357–362, 2020.

[48] W. McKinney, "Data structures for statistical computing in Python," in Proc. 9th Python in Science Conf., 2010, pp. 56–61.

[49] P. Virtanen et al., "SciPy 1.0: Fundamental algorithms for scientific computing in Python," Nature Methods, vol. 17, pp. 261–272, 2020.

[50] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[51] "PostgreSQL documentation." [Online]. Available: https://www.postgresql.org/docs/ (accessed Oct. 20, 2026).

[52] "SQLAlchemy documentation." [Online]. Available: https://docs.sqlalchemy.org (accessed Oct. 20, 2026).

[53] "Alembic documentation." [Online]. Available: https://alembic.sqlalchemy.org (accessed Oct. 20, 2026).

[54] R. Aroussi, "yfinance: Download market data from Yahoo! Finance's API." [Online]. Available: https://github.com/ranaroussi/yfinance (accessed Oct. 20, 2026).

[55] "exchange_calendars: Trading calendars for financial exchanges." [Online]. Available: https://github.com/gerrymanoim/exchange_calendars (accessed Oct. 20, 2026).

[56] F. A. Sortino and L. N. Price, "Performance measurement in a downside risk framework," The Journal of Investing, vol. 3, no. 3, pp. 59–64, 1994.

[57] M. Magdon-Ismail and A. F. Atiya, "Maximum drawdown," Risk, vol. 17, no. 10, pp. 99–102, 2004.

[58] Python Software Foundation, "ast — Abstract syntax trees," Python 3 documentation. [Online]. Available: https://docs.python.org/3/library/ast.html (accessed Oct. 20, 2026).

[59] Python Software Foundation, "resource — Resource usage information," Python 3 documentation. [Online]. Available: https://docs.python.org/3/library/resource.html (accessed Oct. 20, 2026).