An Intelligent Financial Advisor

- **Student Name:** CHEN Ziyang
- **Student ID No.** 21095751D
- **Programme-Stream Code:** 62435-FFT
- **Supervisor:** Dr. LEONG Hong Va
- **Co-Examiner:** Dr. XIA Xianjin
- **Working with** LEE, Ka Chun Caius, 20069386D

Table of contents

1. Introduction & Problem Statement

Introduction

- Aging Society Challenges
- Traditional Financial Planners: High fees and low efficiency
- FinTech & AI quickly development
- Greater efficiency and lower fees intelligent services are needed

Problem Statement

- Difficulty to predict future Scenario and ensure asset value preservation
- Lack of risk assessment tools to personalize user assets
- Lack of affordable professional financial planning services.

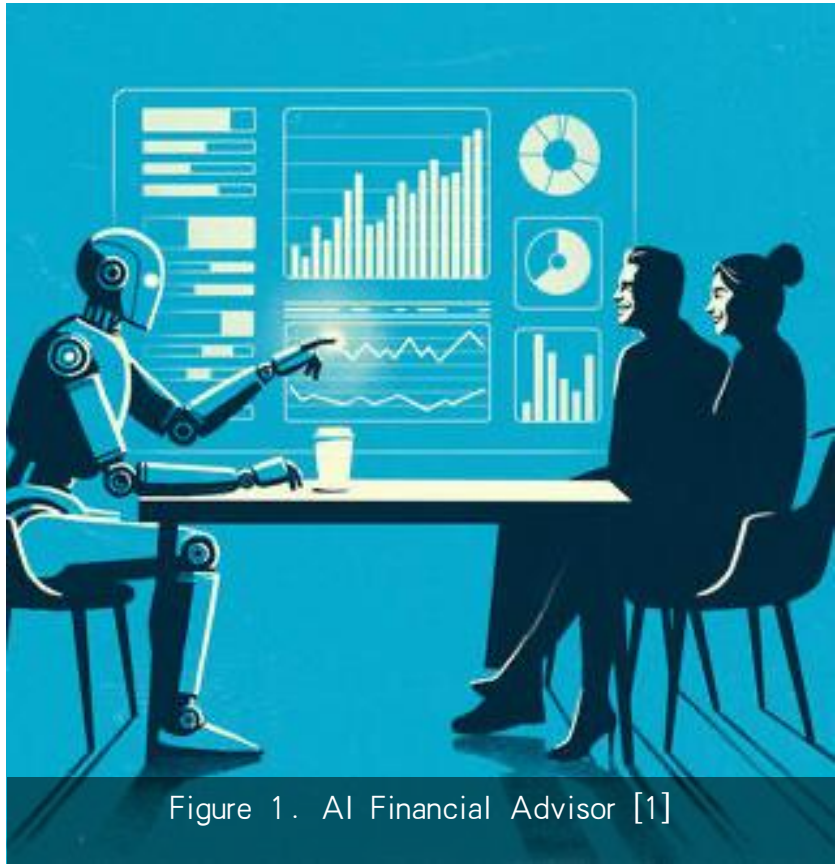


Figure 1. AI Financial Advisor [1]

2. Objectives and Outcome

1. **Portfolio Allocation Based on User Risk Tolerance**
   - Predict risk tolerance and Generate user portrait based on user info.
   - Allocate assets based on user risk tolerance and mean-variance optimization.
   - Help user min risk while max potential growth, make wise financial decisions.
2. **Portfolio Historical Performance**
   - Multi-portfolio backtesting with different asset allocations.
   - Assess portfolios performance with metrics (final balance, CAGR, max drawdown, and Sharpe ratio) over past periods, including withdrawals and rebalancing.
   - Help users to evaluate the past risk and return of the portfolios.
3. **Simulator Tool**
   - Simulate future financial scenarios using three types of models. (Historical, Statistical, Parametric simulation)
   - Test portfolios under optimistic, pessimistic, or baseline conditions.
   - Help users prepare for potential future economic outcomes.

3. Project Methodology

Figure 3. SCFP2022 (Household financial survey of the Federal Reserve) [2]

Project Methodology (Risk Tolerance Calculation)

- **Dataset:** SCFP2022 (Household financial survey of the Federal Reserve) [2]
- **Risk Tolerance Calculation:**

$$Risk\ Tolerance = \frac{Risky\ Assets}{Risky\ Assets + RiskFree\ Assets}$$

  - **Risky Assets:** Stocks, Bonds, Non-Money Market Funds (NMMF).
  - **RiskFree Assets:** Liquid assets, Time deposits, Savings bonds, Cash.

Project Methodology (Data Processing & Feature Engineering)

- **Handling Missing Values**
- **One-Hot Encoding:** Convert categorical variables into separate binary features.
  - Categorical variables: Education level, marital status, occupation.
- **Feature Selection:**
  - Key Features: Age, Income, Education, Occupation, Marital Status.
  - Method: Use ML model for initial feature selection.
  - Domain Knowledge: Combined with financial expertise to refine feature choices.

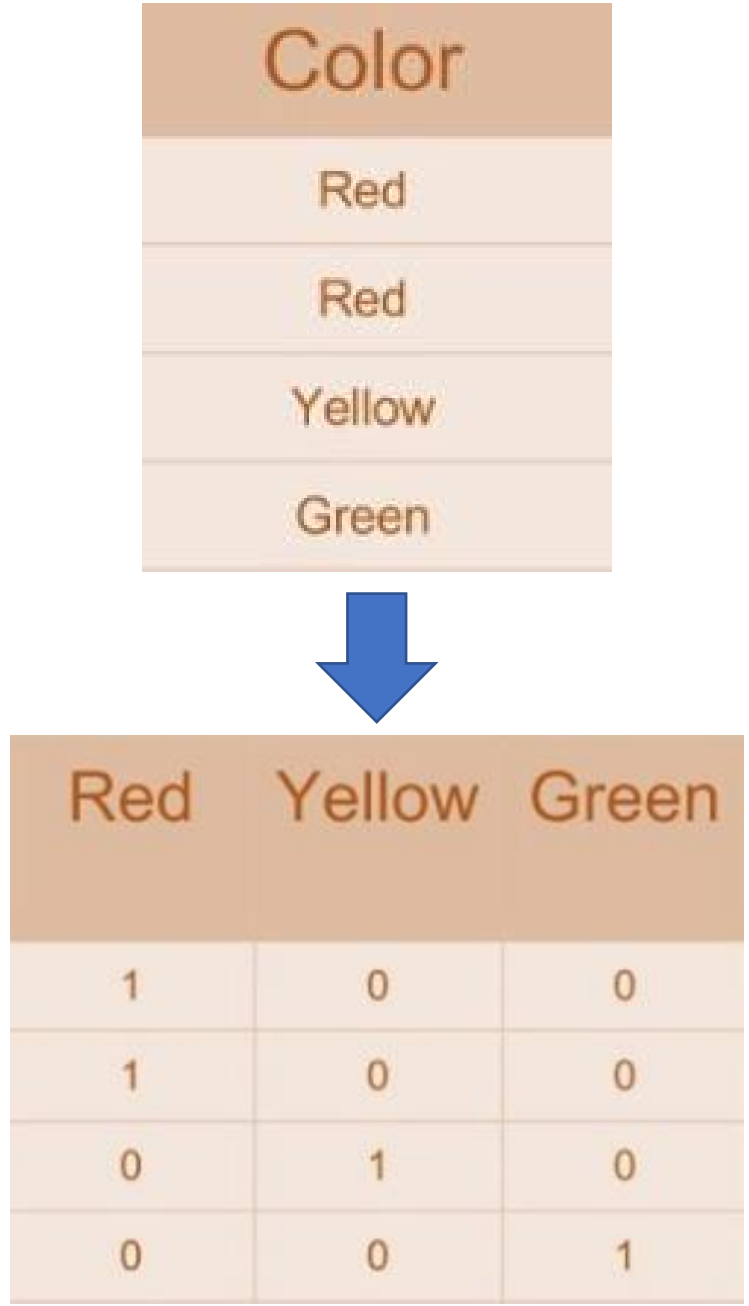


Figure 4. One-Hot Encoding Example [3]

Project Methodology (Portfolio Simulation)

Objective: Simulate future financial scenarios (Optimistic, Pessimistic, Baseline).

- Optimistic Scenario (↑μ, ↓σ): Increases mean returns and decreases volatility.
- Pessimistic Scenario (↓μ, ↑σ): Decreases mean returns and increases volatility.

Monte Carlo Simulation, with Historical, Statistical, and Parametric simulation.

- **Historical Simulation:** Uses actual past returns to simulate future portfolio performance.

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}} \times 100 \quad R_{portfolio} = \sum_{i=1}^{N} w_i \times r_i$$

- **Statistical Simulation (Normal Model):** assumes returns follow a normal distribution, generating asset returns based on mean and volatility.

$$r_{annual} \sim \mathcal{N}(\mu, \Sigma)$$

- **Statistical Simulation (GARCH Model):** models changing market volatility over time, helping simulate real-world ups and downs more accurately.

$$r_t = \mu + \epsilon_t, \epsilon_t \sim N(0, \sigma_t 2)$$

Project Methodology (Portfolio Simulation)

- **The Parameterized Simulation:** Defines parameters for return distribution (adjusts mean $\mu$and volatility σ) to simulate returns based on Lognormal distribution.
  - **Annualized Mean (μ):** A higher mean results in greater expected returns.
    - Adjusting μshifts the central value of asset returns.
    - Increasing μ increases expected returns
  - **Annualized Volatility (σ):** A higher volatility increases risk and return fluctuation.
    - Adjusting σalters the dispersion of returns, affecting risk.
    - increasing σ introduces more risk.
  - **Covariance & Correlation :** Affects portfolio risk by describing how assets move together.
    - High correlation leads to concentrated risk.

4. Design & Development

Design & Development

The system adopts a front-end and back-end separation architecture.

- **Server:** Node.js + Express + Flask
- **Database:** Amazon Web Services RDS for MySQL (stores user data)
- **Front-end:** React framework + UI (Ant)
- Local csv files: store market data (daily update using Kaggle dataset) [4].

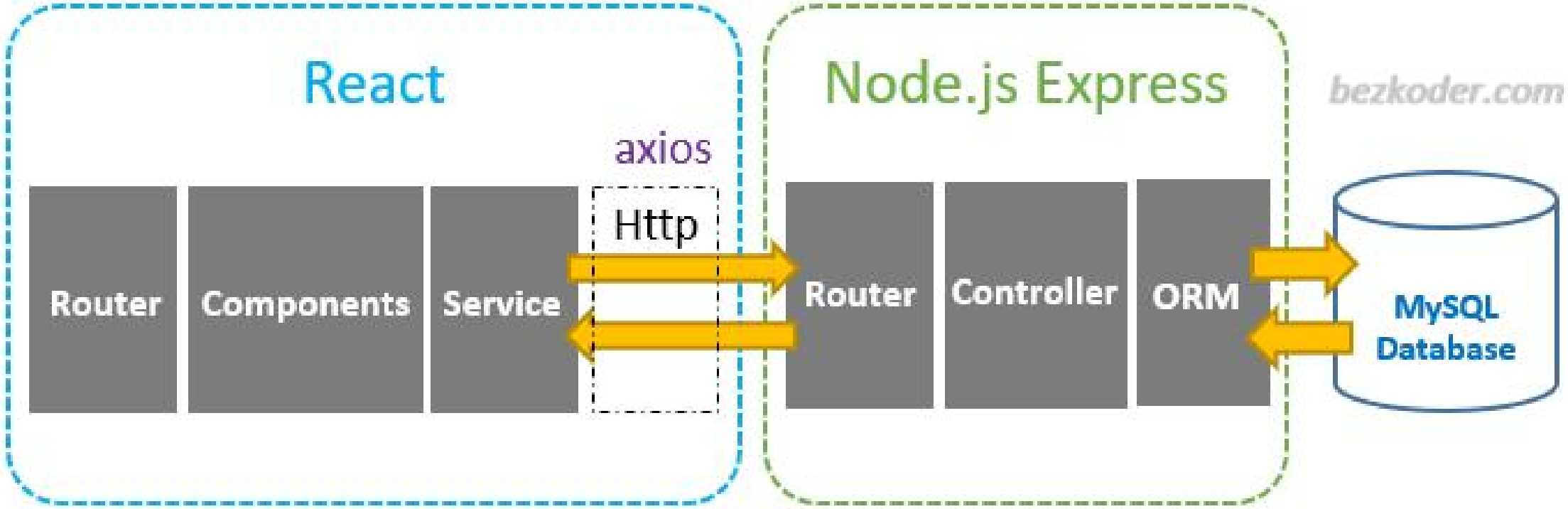


Figure 5. React + Node.js + Express + Amazon Web Services RDS Overview [5]

Design and Development - Components

Group project with my teammate LEE, Ka Chun Caius (20069386D). My personal contribution is detailed with attributes and methods. There are **three main components** in my contribution:

- **User Portrait**
- **Multi-Portfolio Back test**
- **Simulator**

- UserPortfolio one-to-many (1:0..*) relationship with Allocation.
- MultiPortfolioBacktest has zero-to-many (0..*) relationship with Asset.
- Simulator has a one-to-many (1:0..*) relationship with Asset.

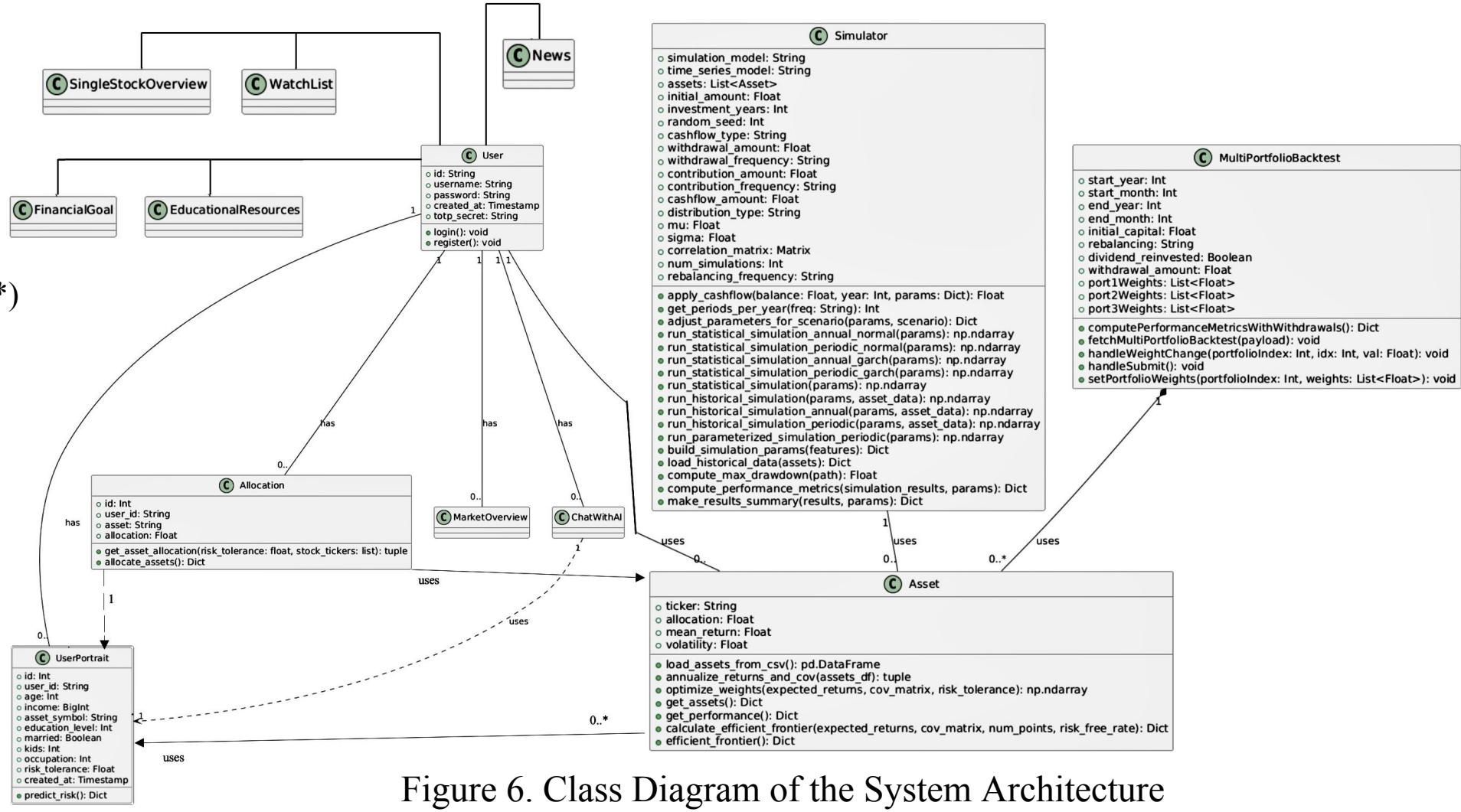


Figure 6. Class Diagram of the System Architecture

Design and Development - Database Design and Normalization

- **Goal:** Store and manage user data (personal information, investment portfolios, financial goals, consumption records, etc.)
- **Relational database:** MySQL
- **Cloud service platform:** Amazon RDS for MySQL
- **Normalized design:** Follows the 1NF (atomic), 2NF (no partial dependencies) and 3NF (no transitive dependencies)
- **user_portrait table:** records user's basic information and risk tolerance (generated by ML model);
- **allocations table:** records user's asset allocation;
- All tables are connected to the users table via the foreign key **user_id.**

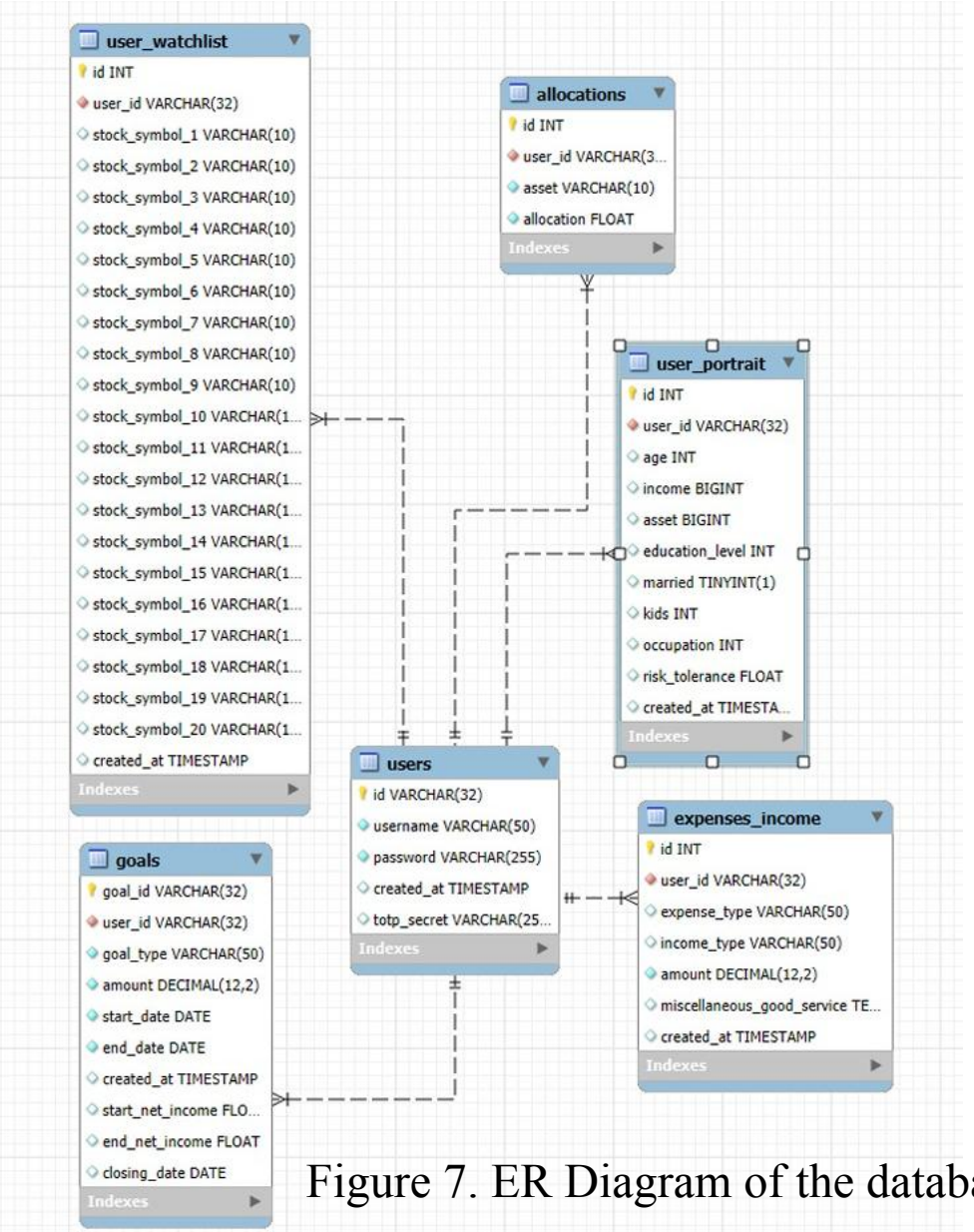


Figure 7. ER Diagram of the database

5. Implementation

5.1 User Portrait Module – Implementation

Save, update and manage the user's basic information. Machine Learning model predicts risk tolerance. Allocate personalized investment portfolio.

Backend (Flask + Node.js)

- /predict_risk
- /allocate
- /get_performance
- /efficient_frontier

Database (MySQL on AWS RDS)

- user_portrait, allocations

Frontend (React + Recharts):

- InvestorForm Collects input for prediction.
- AllocationChart, PerformanceChar EfficientFrontierChart: Visualize AI-generated allocation and return.

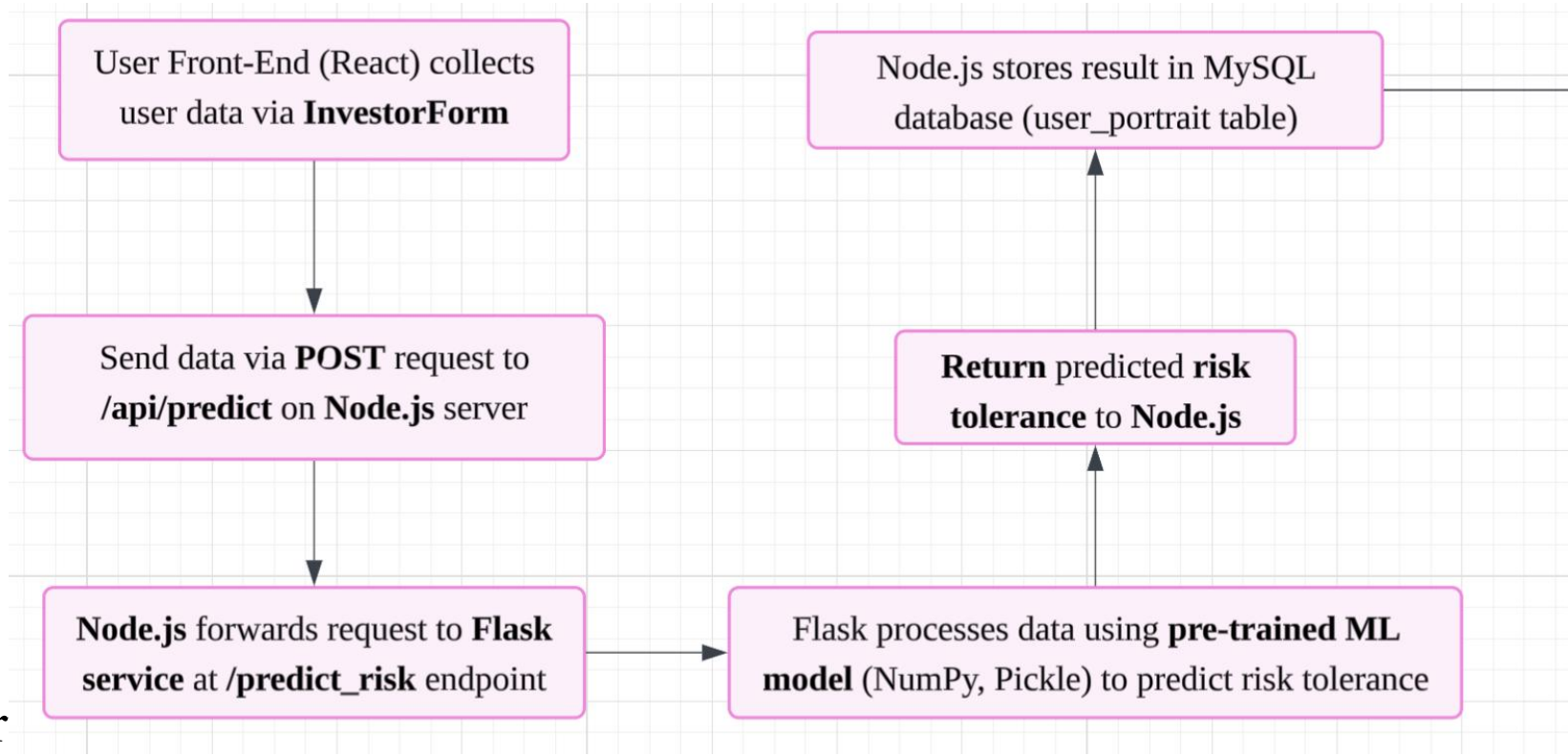


Figure 8. User Risk Tolerance Prediction Workflow

5.1 User Portrait Module – Implementation

Save, update and manage the user's basic information. Machine Learning model predicts risk tolerance. Allocate personalized investment portfolio.

Backend (Flask + Node.js)

- /predict_risk
- /allocate
- /get_performance
- /efficient_frontier

Database (MySQL on AWS RDS)

- user_portrait, allocations

Frontend (React + Recharts):

- InvestorForm Collects input for prediction.
- AllocationChart, PerformanceChar EfficientFrontierChart: Visualize AI-generated allocation and return.

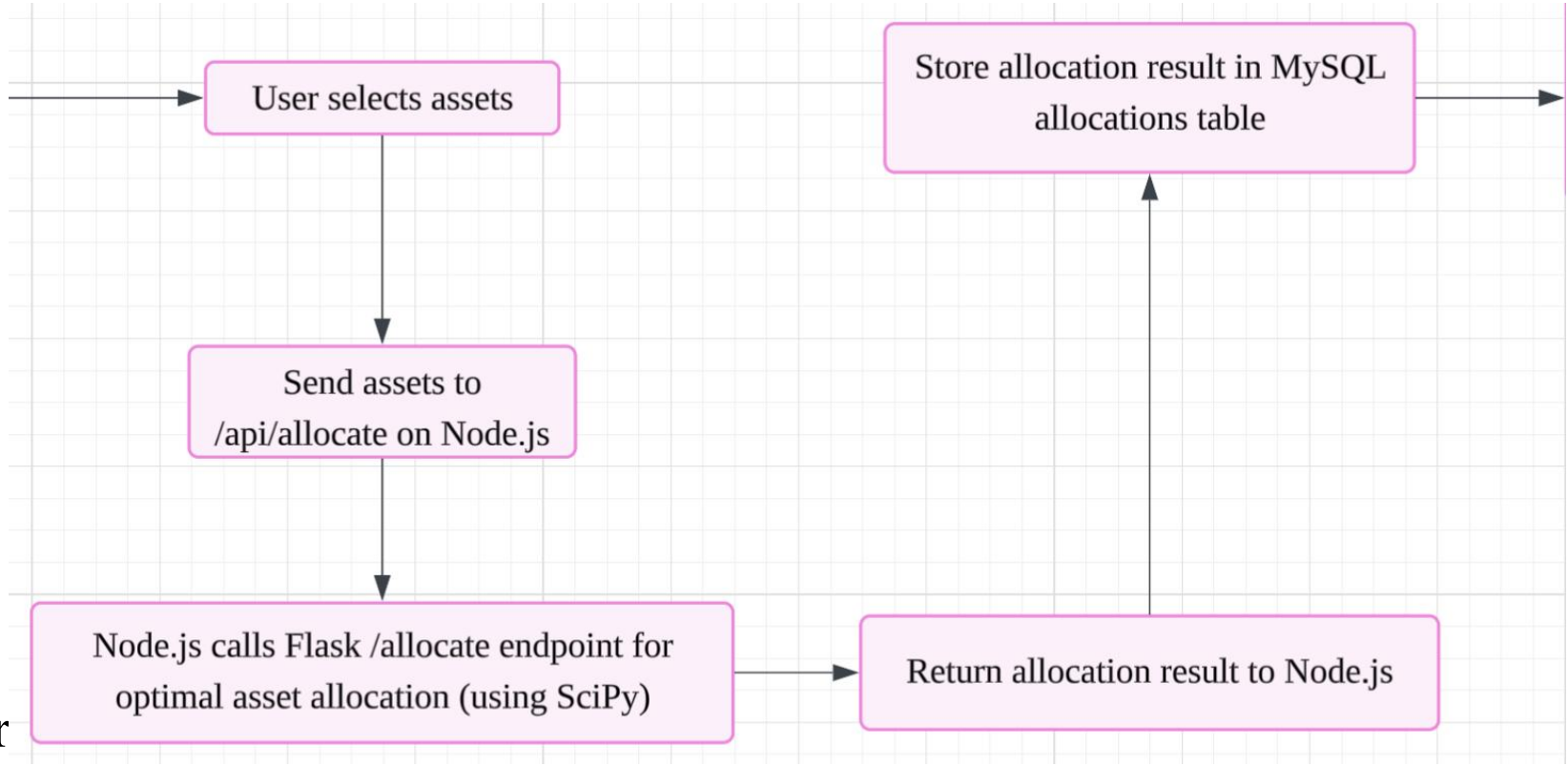


Figure 9. Asset Allocation Process Flow

5.1 User Portrait Module – Implementation

Save, update and manage the user's basic information. Machine Learning model predicts risk tolerance. Allocate personalized investment portfolio.

Backend (Flask + Node.js)

- /predict_risk
- /allocate
- /get_performance
- /efficient_frontier

Database (MySQL on AWS RDS)

- user_portrait, allocations

Frontend (React + Recharts):

- InvestorForm Collects input for prediction.
- AllocationChart, PerformanceChart, EfficientFrontierChart: Visualize AI-generated allocation and return.

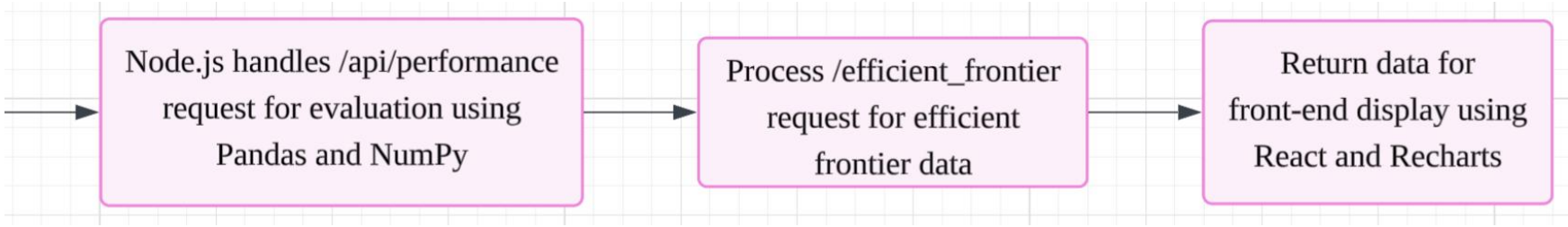


Figure 10. Performance Evaluation and Efficient Frontier Calculation Process

5.2 Risk Tolerance predicting Implementation

1. **Data Preprocessing**: Remove rows with missing values (NaN), infinity (inf), and negative infinity (-inf).
2. **One-Hot Encoding**: Categorical variables (e.g., Education Level, Marital Status) were transformed into binary features. Avoids introducing bias that could arise from treating categories as ordinal.
3. **Feature Selection**: Used RandomForest to assess feature importance. Combined with domain knowledge and project requirement to select features like age, education level, income.
4. **Model Training**: Tested multiple models. **ExtraTreesRegressor** perform the best, and tuned ExtraTreesRegressor with **RandomizedSearchCV** for optimal performance.
5. **Risk Tolerance Prediction**: User submits financial data through the front-end. Back-end processes the data using the trained model and outputs a risk tolerance value (1 to 5).

5.3 Dynamic Asset Allocation Implementation

1. Data preparation and annualization:
   - Load monthly closing prices from CSV: assets_monthly = load_assets_from_csv()
   - Calculate monthly returns: pct_change().dropna()
   - Annualized returns μ and covariance matrix Σ: annualize_returns_and_cov()
2. Minimize portfolio volatility:
   - Initial equal weight $w_0 = 1/N$
   - Define the objective function: portfolio volatility $\sigma(w) = \sqrt{(w^T \Sigma w)}$, constraint $\sum w_i = 1$, $0 \leq w_i \leq 1$
   - Call scipy.optimize.minimize to find w* that minimizes portfolio volatility σ(w).
3. Optimize weights with risk tolerance:
   - Optimize weight w* combined with risk tolerance (τ): $w_{adjusted} = \tau \times w^* + (1 - \tau) \times w_{equal}$
   - Normalization: $w_{final} = w_{adjusted} / \sum w_{adjusted}$
4. Output and backtesting
   - Return w_{final}, calculate the monthly return and cumulative performance of the portfolio based on w_{final}: get_asset_allocation()

5.4 Simulation Module – Implementation

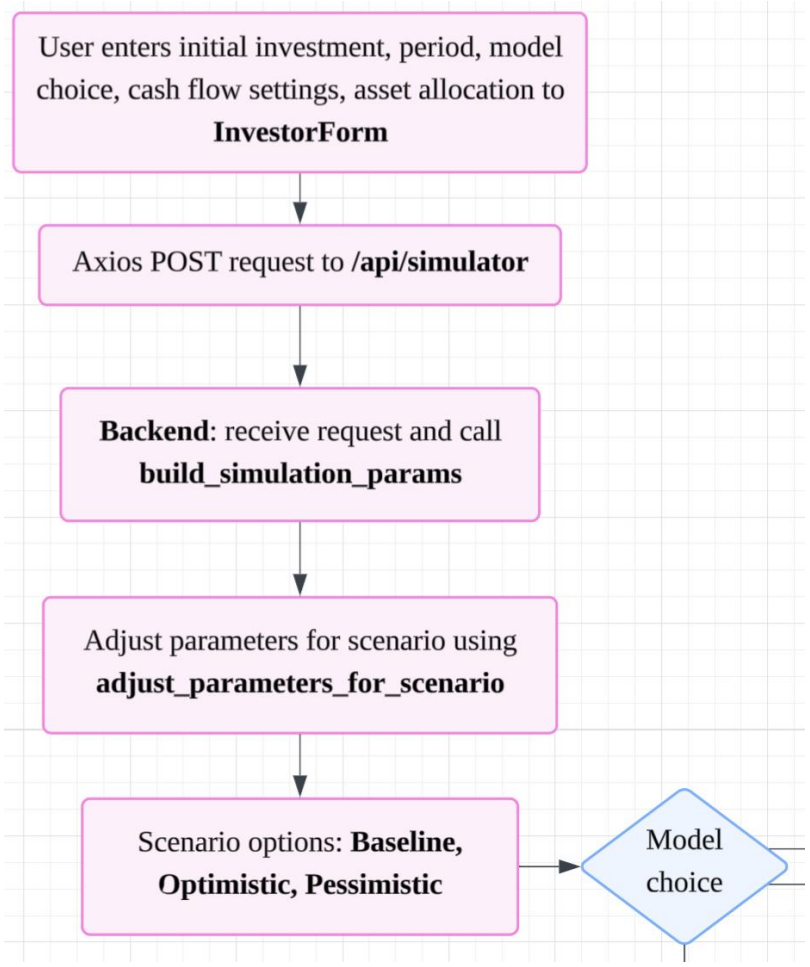


Figure 11. Simulation Module Flowchart

Purpose: simulate portfolio performance under different market scenarios (Optimistic, Pessimistic, Baseline).

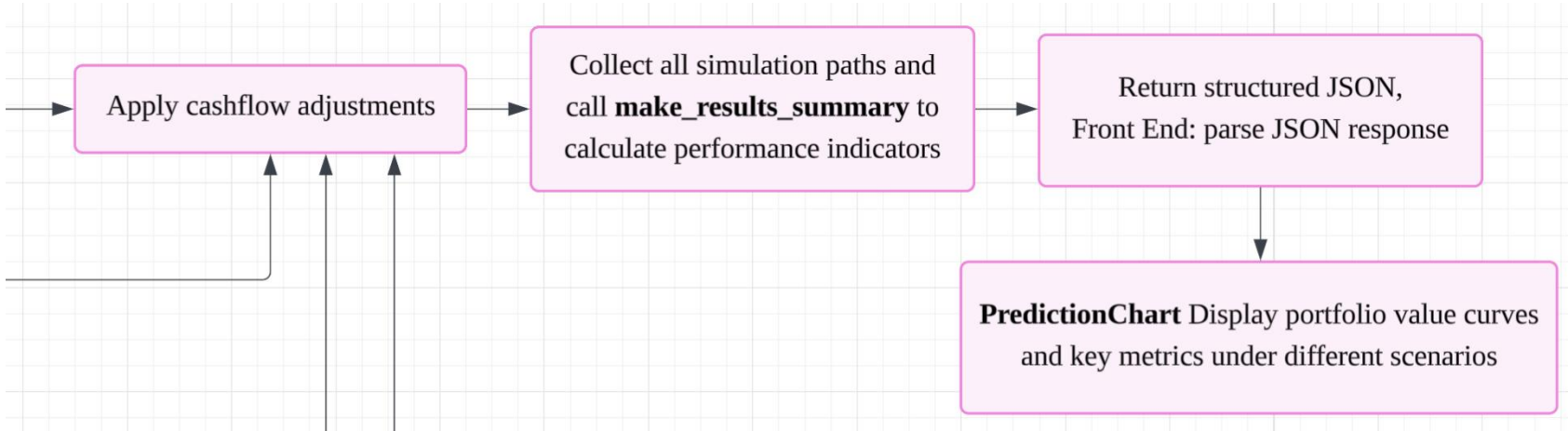
- Built-in **scenario adjustment & cash-flow functions.**
- **Three Monte Carlo simulation:** historical, statistical, parametric.

Backend (Flask):

1. /api/simulator accepts JSON, builds params via build_simulation_params()
2. Calls adjust, then selected run function, collects results via make_results_summary()
3. Returns structured JSON per scenario

Frontend (React):

1. InvestorForm gathers inputs, validates weights = 100%
2. Axios POST to /api/simulator with JWT
3. Displays PredictionChart & metrics table under tabs



5.4 Simulation Module – Implementation

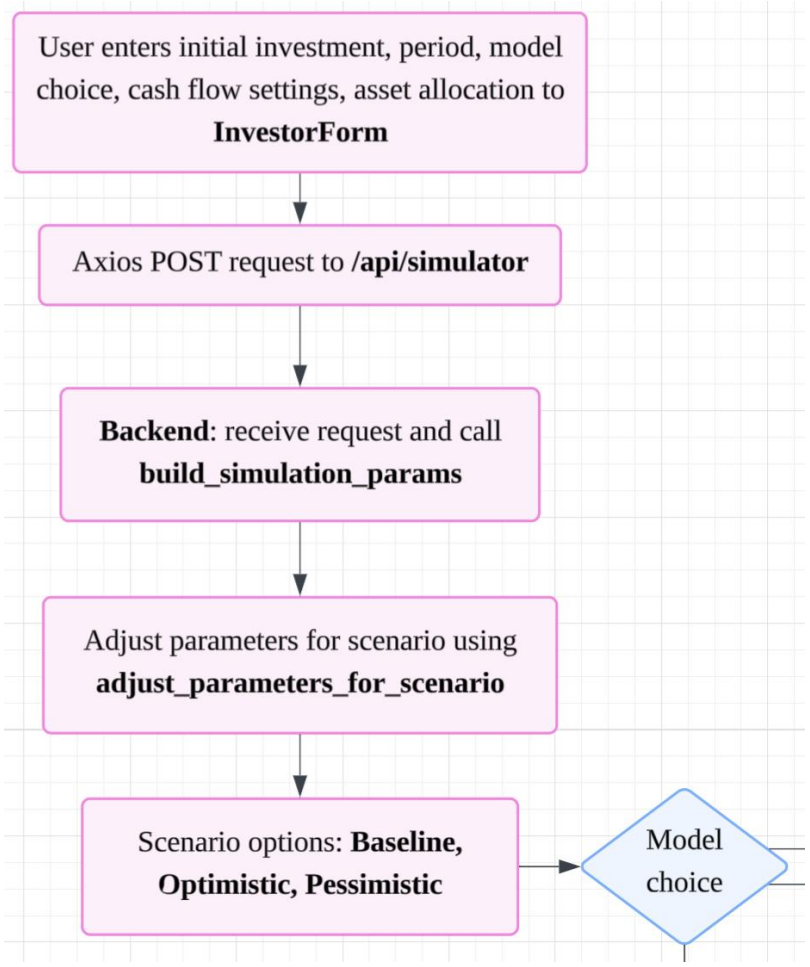


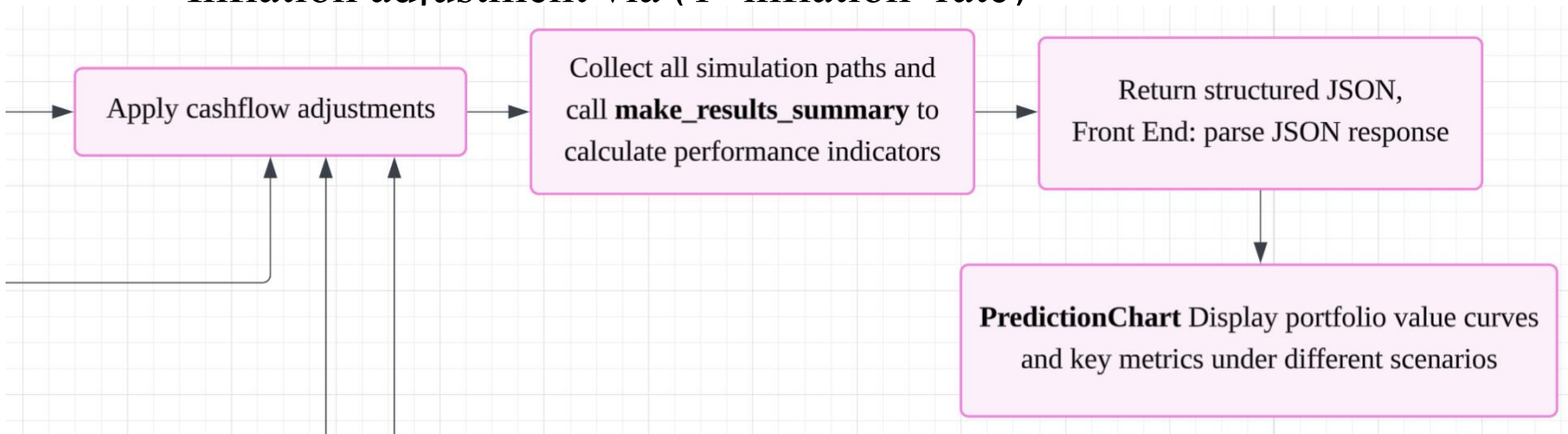
Figure 11. Simulation Module Flowchart

Scenario Adjustment (adjust_parameters_for_scenario):

- Optimistic: $\mu_{new} = \mu_{old} + 0.01, \quad \sigma_{new} = \sigma_{old} \times 0.9$
- Pessimistic: $\mu_{new} = \mu_{old} - 0.01, \quad \sigma_{new} = \sigma_{old} \times 1.1$
- Baseline: no change

Cash Flow (apply_cashflow):

- Fixed withdrawal/contribution & percentage withdrawal
- Inflation adjustment via $(1+\text{inflation rate})^{\text{year}}$



5.4 Simulation Module (Historical) – Implementation

- Repeatedly **sample past monthly returns** (eg. randomly extracts 12 returns), **compound them into annual** portfolio returns based on asset weights
- Calls **apply_cashflow** to update the account balance.
- **Iterate** across years and **Monte Carlo paths** to **build a full distribution** of potential portfolio values.
- The main functions: run_historical_simulation_annual.

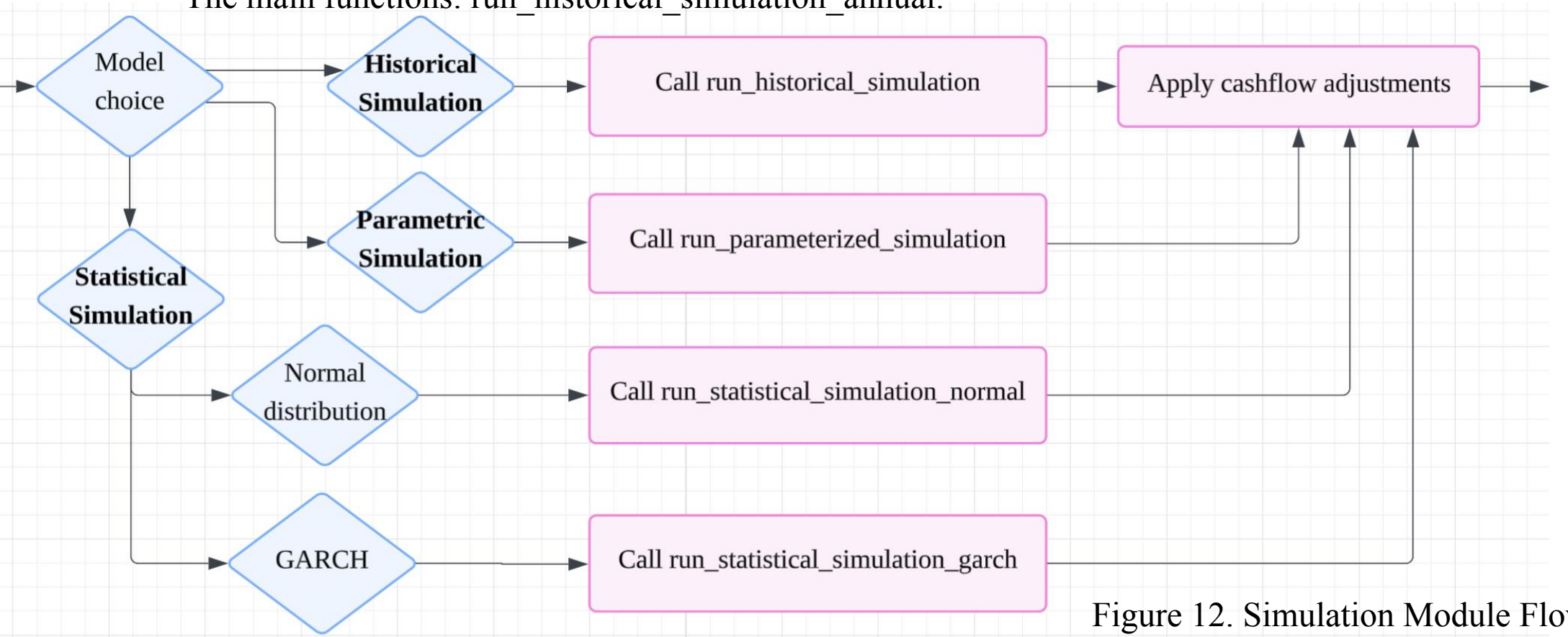


Figure 12. Simulation Module Flowchart (model part)

5.4 Simulation Module (Statistical) – Implementation

- **The normal model** assumes the asset return rate follows a **normal distribution** with **a fixed mean** and **fixed volatility.**
- **The GARCH model** treats **"volatility"** as a **variable** that **changes over time**. Continuously update the current conditional variance based on the historical forecast errors (residuals) and past volatility levels.
- Related functions: run_statistical_simulation_normal, run_statistical_simulation_garch

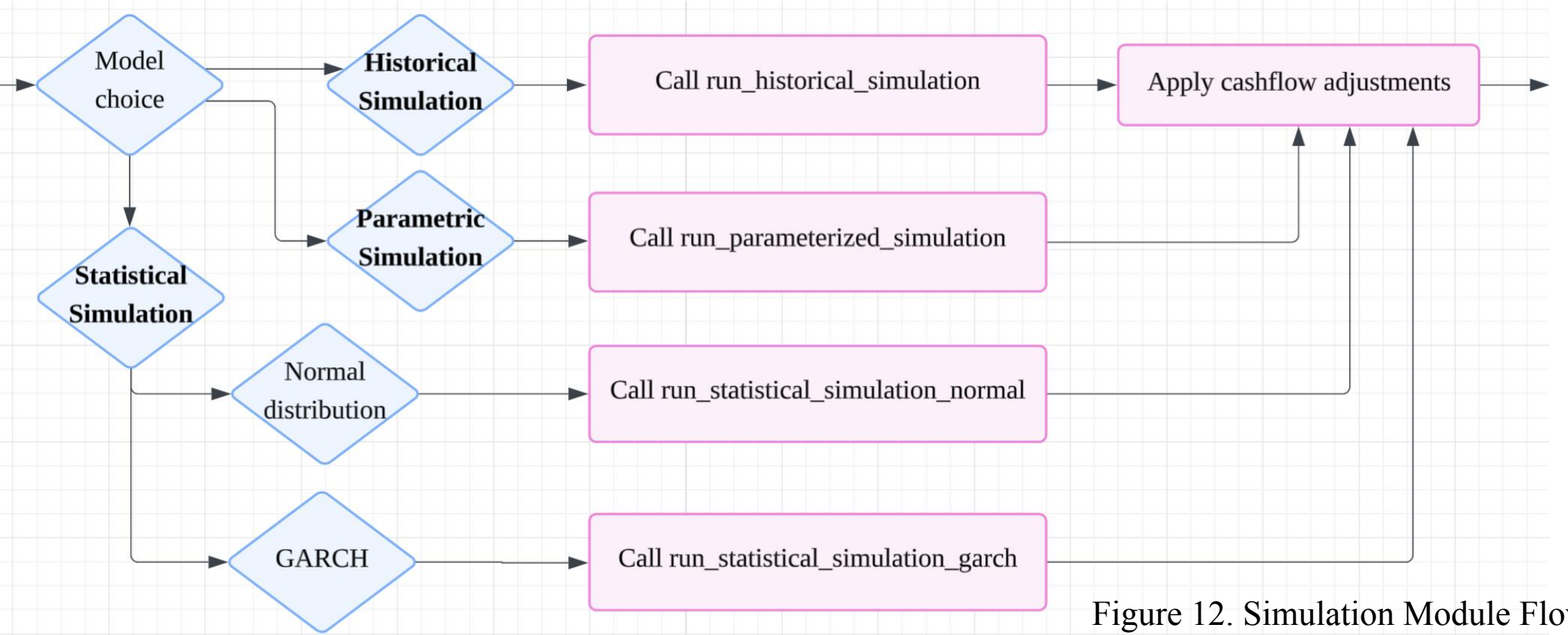


Figure 12. Simulation Module Flowchart (model part)

5.4 Simulation Module (Parametric) – Implementation

- **Does not rely on any historical data**.
- Directly samples according to the Lognormal distribution based on **the μ and σ input by the user**
- Calls apply_cashflow to complete the cash flow processing.
- The main functions: run_parameterized_simulation

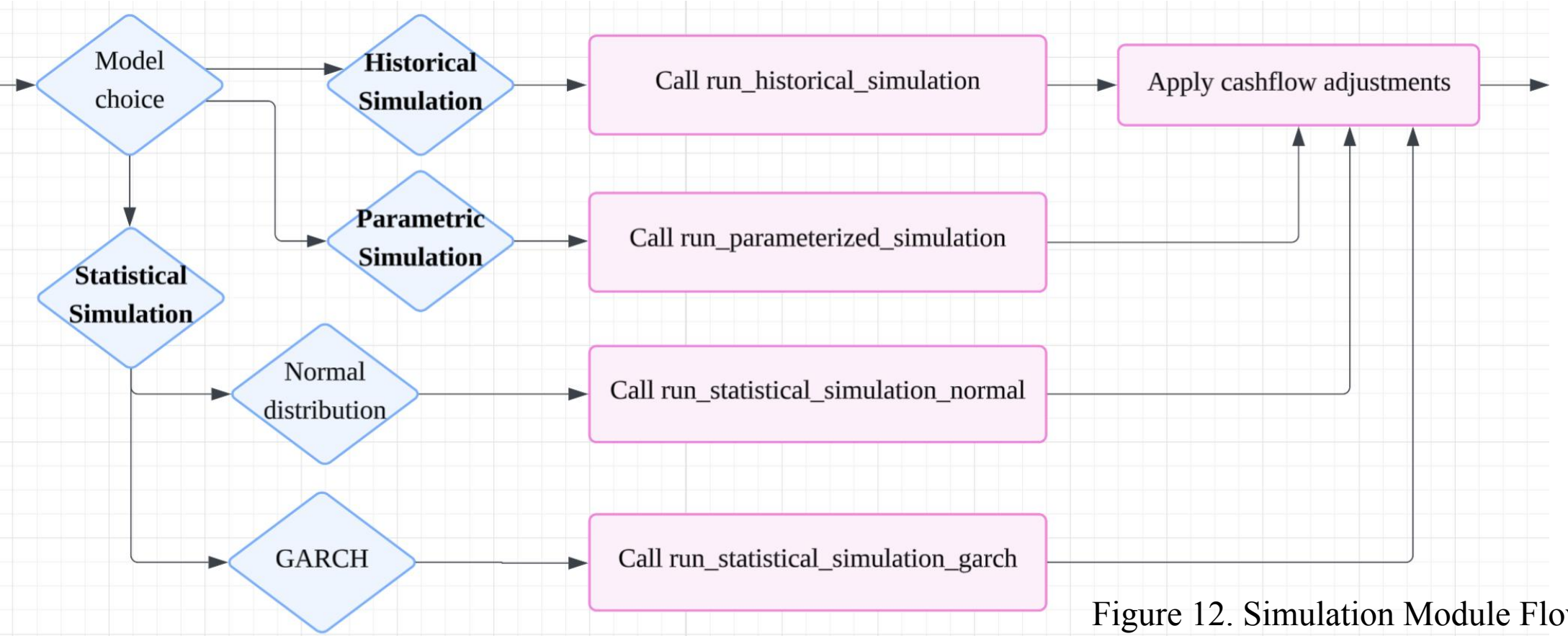


Figure 12. Simulation Module Flowchart (model part)

6. Result analysis and Reflection

6.1 Dataset

Dataset: **SCFP2022**

Published by the **Federal Reserve Board (FRB)** of the US [2].

- **Category:** EDCL, MARRIED, OCCAT1, YESFINRISK
- **Discrete numeric:** KIDS
- **Continuous numeric:** AGE, INCOME, ASSET, RT

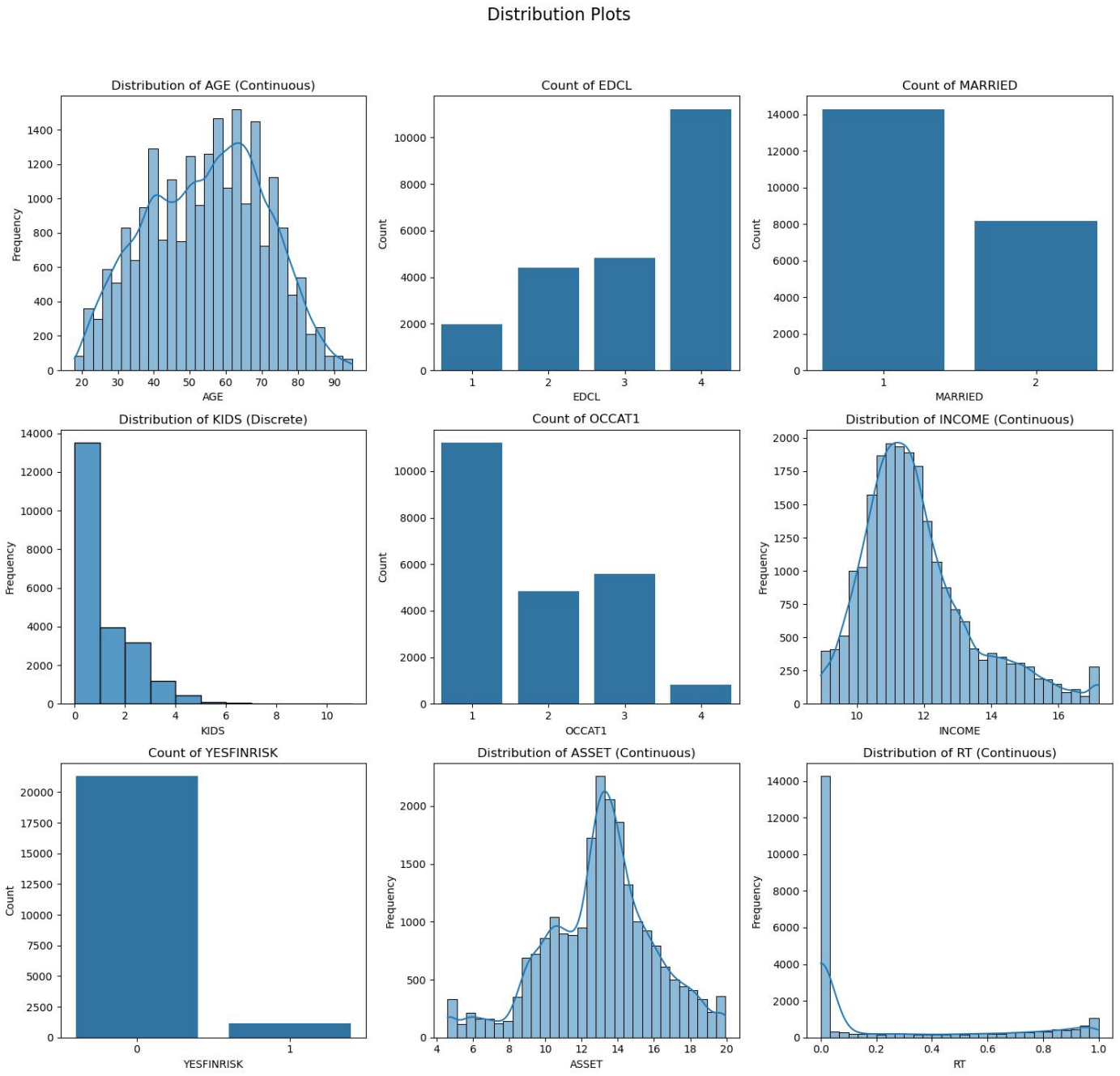


Figure 13. Feature Distribution

6.2 Model Training

- **10-fold** cross-validation
- **R^2** as the performance score
- **Model:** LR, decision trees, SVR, MLP, RandomForest Regressor, ExtraTreesRegressor
- **GridSearchCV** to tuned hyperparameters
- **ExtraTreesRegressor** performs **the best!**

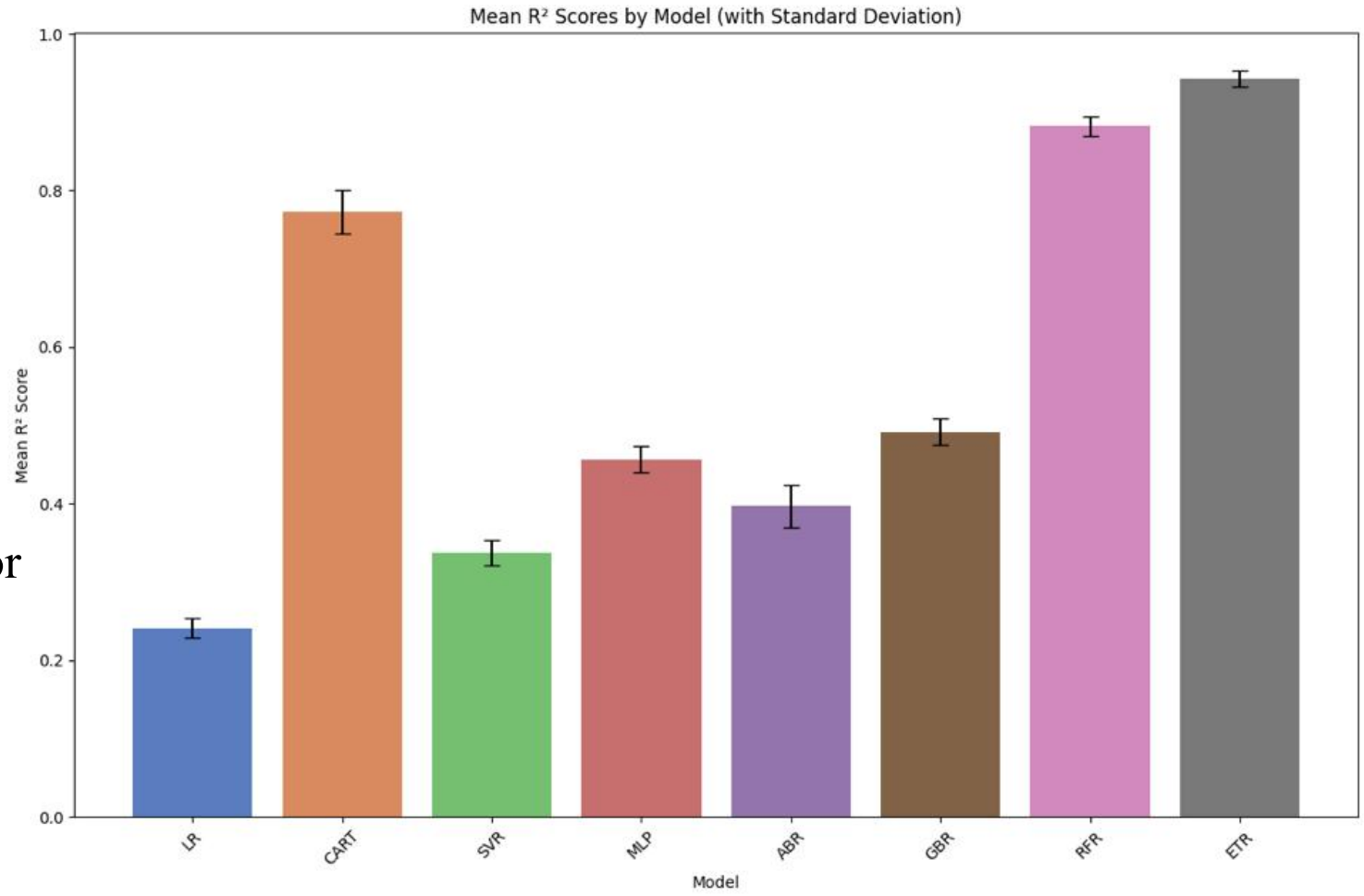


Figure 14. Overall Model Performance

6.3 ExtraTreesRegressor Model evaluation results

- R^2, **MSE, and RMSE** are performed well

To calculate the **F1 score:**

- **Risk tolerance** divides into **five levels** (classification task).
- **F1 validation value** is **0.74.**
- May be the regression task will **lose some details** in the classification task.
- Risk tolerance score is **noisy.**
- Need more model **generalization**

Metric	Training Value	Validation Value
R^2	0.9873	0.9275
MSE	0.0016	0.0090
RMSE	0.0399	0.0950
MAE	0.0179	0.0439
F1 Score	0.8907	0.7394

Table 1. ExtraTreesRegressor model evaluation results

6.4 the Reflection of Feature Importance

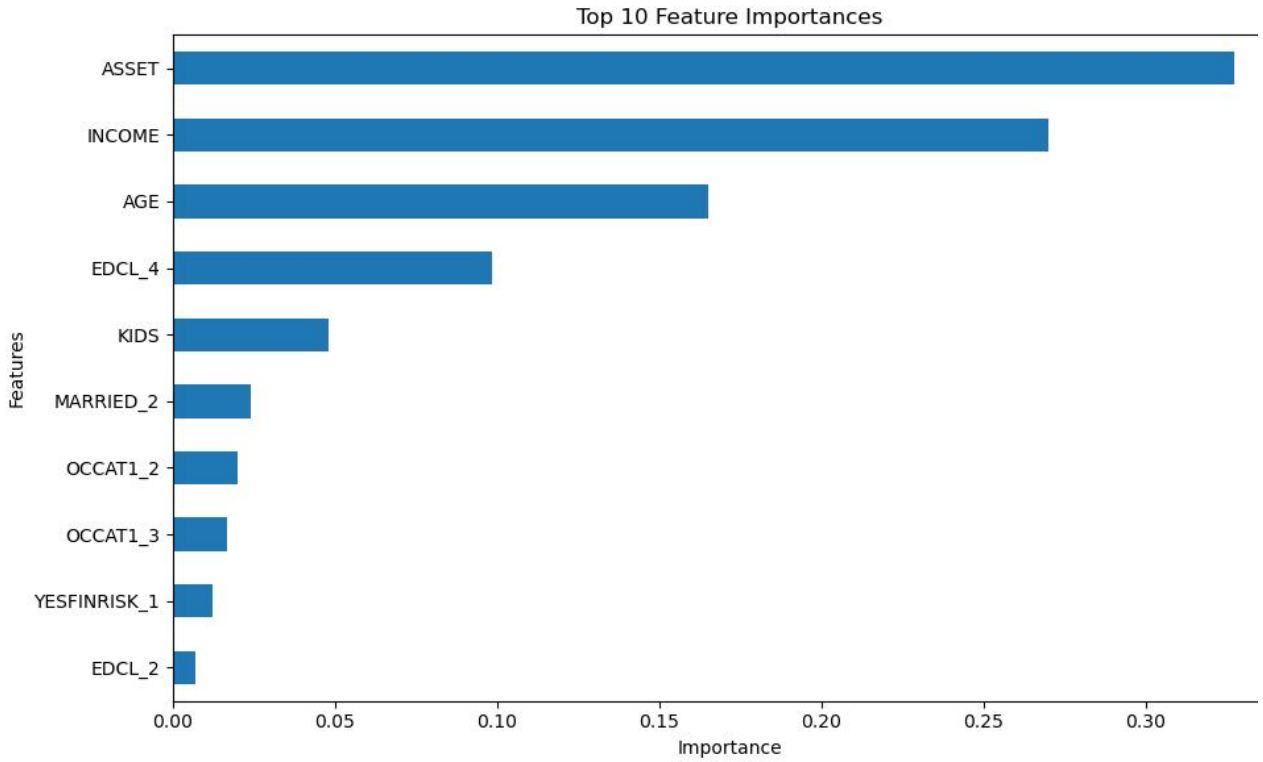


Figure 15. Feature Importance

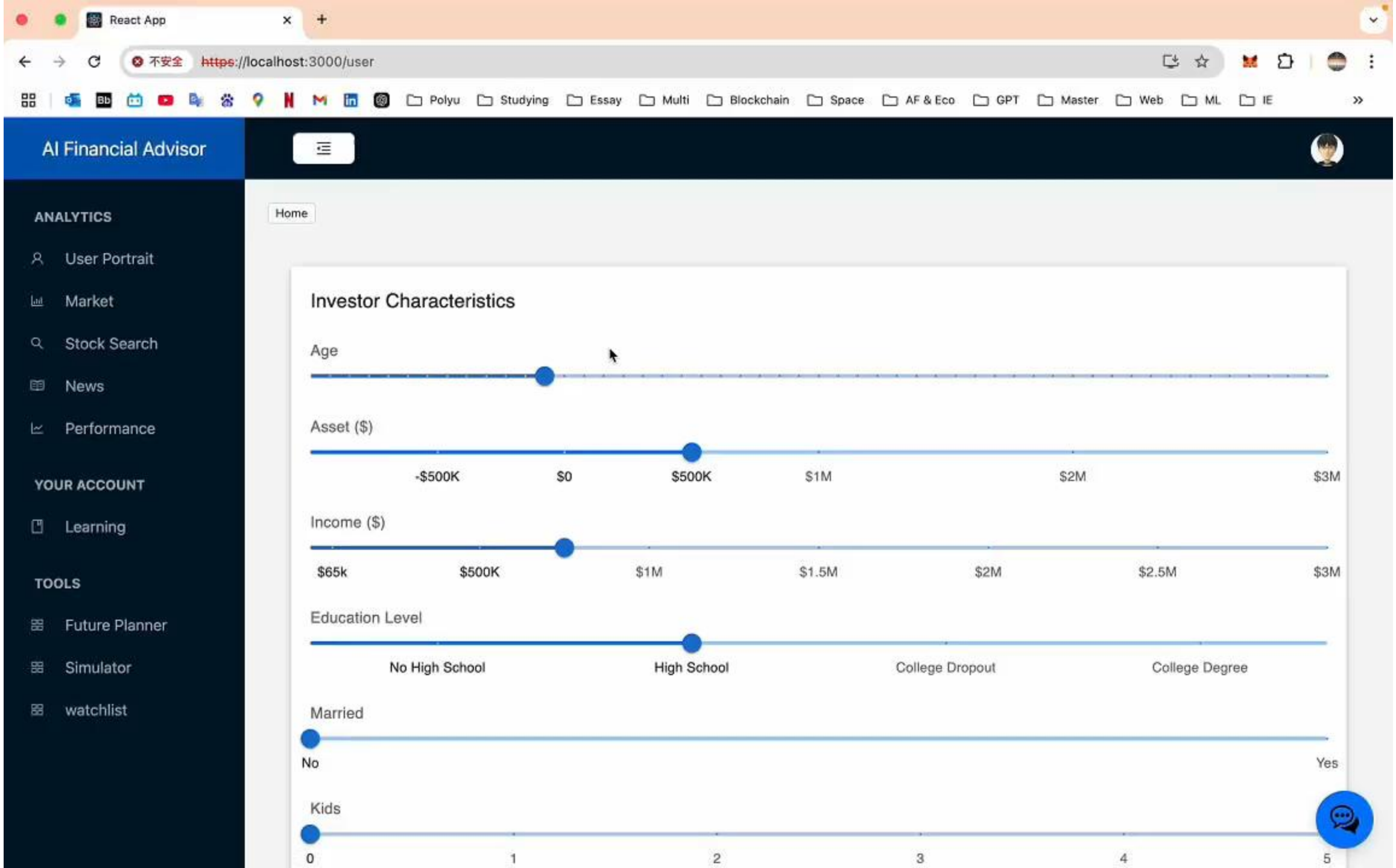
Most important features: **ASSET, INCOME**

- Financial resources is the core role of the wealth safety net.

EDCL_4 vs EDCL_2: Education NOT ONLY improves knowledge, BUT ALSO affects decision-making by improving economic status and cognitive ability

- High education (EDCL_4): Master more information and resources → Stronger risk management ability
- Low education (EDCL_2): Face uncertainty and pressure → Limited risk tolerance

7. Demonstration


React App
不安全
https://localhost:3000/user
Polyu
Studying
Essay
Multi
Blockchain
Space
AF & Eco
GPT
Master
Web
ML
IE
AI Financial Advisor
ANALYTICS
User Portrait
Market
Stock Search
News
Performance
YOUR ACCOUNT
Learning
TOOLS
Future Planner
Simulator
watchlist
Home
Investor Characteristics
Age
Asset ($)
-$500K
$0
$500K
$1M
$2M
$3M
Income ($)
$65k
$500K
$1M
$1.5M
$2M
$2.5M
$3M
Education Level
No High School
High School
College Dropout
College Degree
Married
No
Yes
Kids
0
1
2
3
4
5

8. Conclusion

- Successfully built risk preference prediction model
- ASSET and INCOME are key factors in risk tolerance
- Achieved Dynamic asset allocation
- Simulation tool supports multi-scenario testing
- **Inspiration:**
  Increasing asset and education support for low-income and low-education groups.
  This system can be used as an inclusive financial tool to help users make smarter financial decisions
- **Future work:**
  Optimize model generalization, expand data sources

9. Reference

1. B. Vereckey, “Can generative AI provide trusted financial advice?,” *MIT Sloan*. [Online]. Available: https://mitsloan.mit.edu/ideas-made-to-matter/can-generative-ai-provide-trusted-financial-advice

2. Survey of Consumer Finances (SCF), “Changes in U.S. family finances from 2019 to 2022: Evidence from the survey of consumer finances,” *Federal Reserve Bulletin*, Oct. 2023. doi:10.17016/8799

3. DigitalSreeni, “137 - What is one hot encoding in machine learning?,” *YouTube*, [Online]. Available: https://www.youtube.com/watch?v=i2JSH5tn2qc.

4. Yash, “S&P500 Daily Update dataset,” *Kaggle*. [Online]. Available: https://www.kaggle.com/datasets/yash16jr/s-and-p500-daily-update-dataset.

5. Bezkoder, “React + node.js + express + mysql example: Build a crud app,” *BezKoder*. [Online]. Available: https://www.bezkoder.com/react-node-express-mysql/.