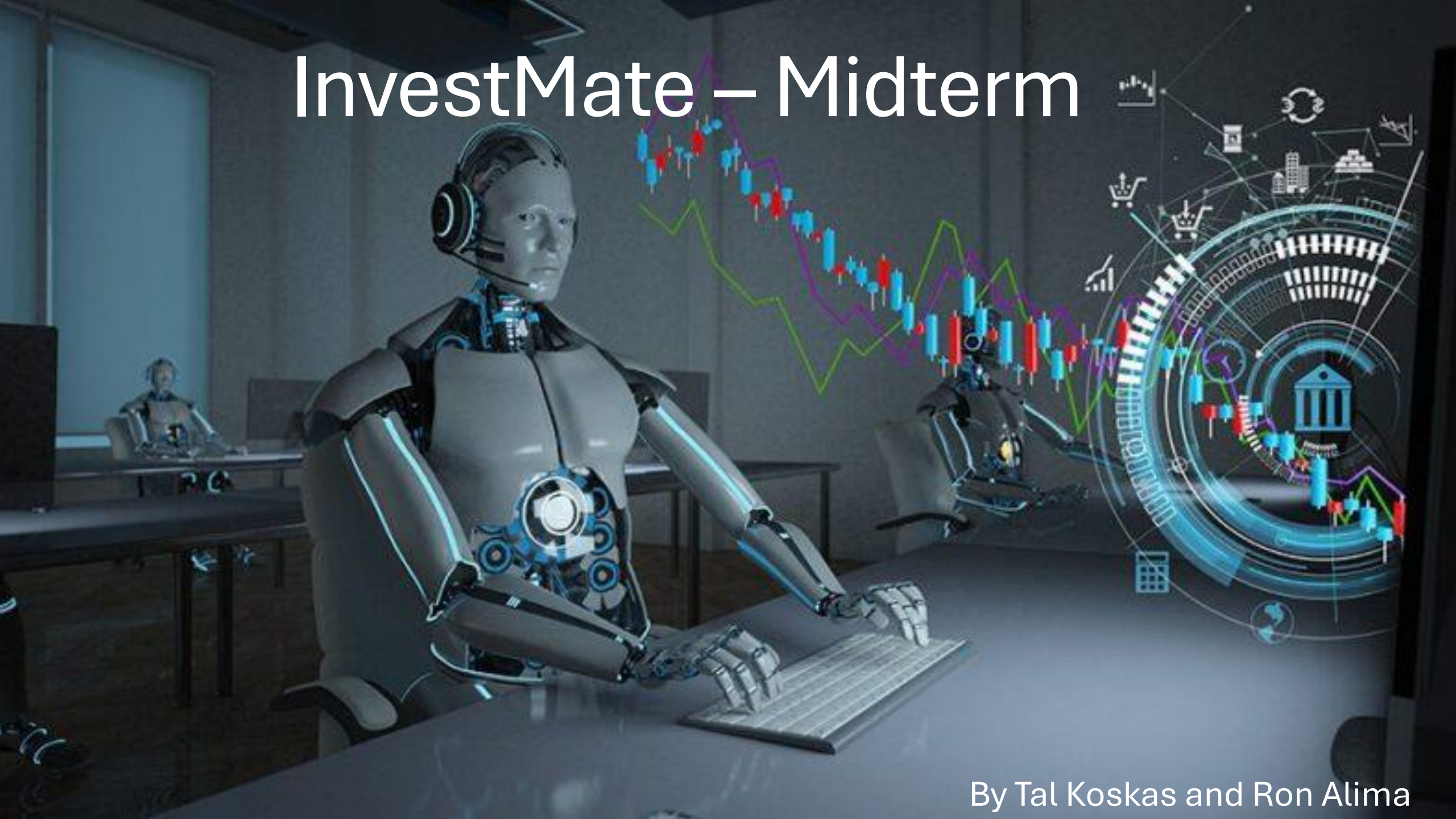


InvestMate – Midterm



By Tal Koskas and Ron Alima

Project Overview

- **Project Name:** Robo Advisor Portfolio Recommender
- **Goal:** Build a web-based robo advisor that recommends personalized investment portfolios using modern portfolio theory (Markowitz Model).
- **Key Feature:** Dynamic portfolio recommendations based on the user's risk profile (Conservative / Balanced / Aggressive).

Problem Statement

- **Problem:** Retail beginner-advanced investors struggle to build efficient portfolios aligned with their risk preferences.
- **Challenges Faced by Users:**
 1. Lack of financial knowledge
 2. Overwhelmed by stock market options
 3. No personalized financial guidance
- **Need:** A system that simplifies investment and provides trustable smart recommendations.

Target Users & Value Proposition

- **Target Audience:**

- Beginner to intermediate investors
- Young professionals with limited financial experience

- **Value Provided:**

- Data-driven portfolio suggestions
- Simple, intuitive interface
- Clear risk-adjusted trade-offs

High-Level Architecture

Frontend (Flask Web App):

- Built with Python + Flask, served over HTTP API.
- Allows users to select risk profile and view personalized portfolio
- Additional tabs: market statistics, investment view, leaderboard

Backend (Python):

- Fetches financial data using yfinance (Yahoo Finance API)
- Core logic based on Markowitz Efficient Frontier model
- Portfolio simulation, optimization, and recommendation

Backend Implementation (Tal Koskas & Ron Alima)

MarkowitzModel Class:

- Calculates expected returns and covariance matrix from historical prices
- Simulates 100,000 portfolios
- Computes Sharpe Ratio, Returns and Volatility

ModelManager Class:

- Trains and stores models (e.g. “markowitz”)
- Returns optimal portfolio per risk profile

Backend Implementation (Continued)

Risk Profiles Implemented:

- Conservative – lowest volatility
- Balanced – max Sharpe ratio
- Aggressive – highest return

REST API Endpoint:

- /api/portfolio: Accepts risk profile, returns recommended portfolio in JSON

Smart Logic – Markowitz Model

- **Inputs:** Historical prices of selected tickers (2020–2025)
- **Simulation:**
 - Portfolio weights sampled randomly
 - Return, volatility, Sharpe calculated per portfolio
 - Efficient frontier plotted
- **Selection:** Best-fit portfolio for each risk profile strategy

Frontend Implementation (Ron Alima)

- **Built using Flask + HTML/CSS/JS**
- **UI Elements:**
 - Dropdown for selecting risk profile
 - **Portfolio display:** ticker + allocation + return
 - **Additional tabs:** Statistics, Investments, Leaderboard, Profile
- **Profile Tab:**
 - User can input budget and view share profit
- **Statistics Tab:**
 - Shows user budget
 - Graph of SPY share performance over time (day/week/month)

Frontend Implementation (Continued)

- **Investments Tab:**

- Displays budget, shares profit, achievements
- "My Positions" table: symbol, shares, entry price, type, % and \$ difference, stop limit, add new position option

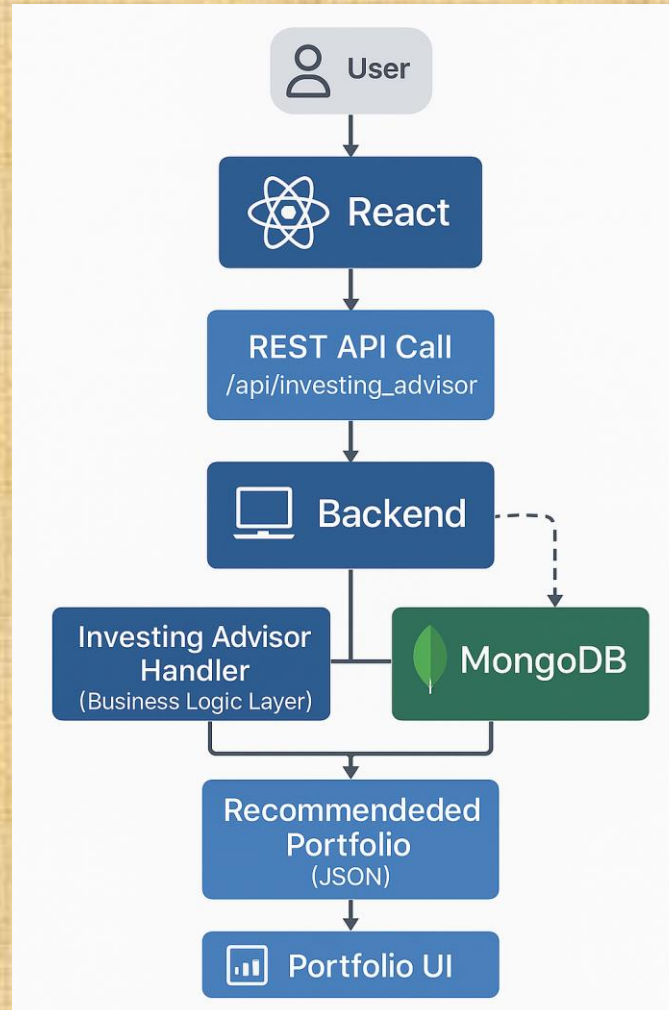
- **Leaderboard Tab:**

- Displays achievements (points), rank, shares profit
- Ranks table: rank, name, achievements, profit

- **Uses AJAX to call Flask API**

- **Responsive design for easy use on desktop/mobile**

Architecture Flow – Portfolio Recommendation Flow



User Interface Features

- - Login/Register Dialog (MUI Dialog + Validation)
- – Portfolio Page: Dynamic allocation table per user
- – Statistics Tab: SPY chart + budget display
- – Investments Tab: Positions, achievements, gain/loss
- – Leaderboard Tab: AJAX-based real-time updates

Backend Architecture Highlights

- - Built with ASP.NET Core 8 (C#)
- – Controllers (e.g., InvestingAdvisorController)
- – Business Logic Layer: Service handlers per domain
- – MongoDB integration (Users, Positions, Risk Profiles)
- – REST API: returns recommended portfolio integrated with tal's part.
- – Coordination with Python ML model via internal service

Demo Time – Live Use Case

- **Input:** User selects risk profile (via dropdown)
- **Output:**
 - Portfolio recommendation: e.g., SPY: 20%, AAPL: 15%, etc.
 - Visualization of efficient frontier
 - Interactive tab interface

Challenges & Future Plans

- **Challenges:**

- Financial data accuracy and availability
- High computational load for simulations
- User-friendly interface for non-technical users

- **Next Steps:**

- Add LSTM-based return forecasting module (deep learning)
- Integrate database (user sessions, portfolio history)
- Deploy full system to cloud (e.g., AWS/Heroku)
- Improve interactivity with real-time refresh and analytics

Summary & Q&A

- **Progress:**

- End-to-end MVP: backend logic, API, and frontend interface
- Real data, real logic, real-time demo

- **Learning Outcomes:**

- Full-stack system implementation
- Practical use of finance + ML concepts
- Collaboration across backend and frontend

- **Any questions or feedback?**