Technical Summary – 6CCS3PRJ Individual Project

Student: Jaemin An (King's College London)

Module: 6CCS3PRJ – Individual Project (Final Year)

Deliverables: Working codebase + 55-page dissertation (design, experiments, results)

Project Title: *Predicting Future Stock Prices Using Genetic Algorithms*

Project Aim

I designed and implemented an end-to-end system that uses a **Genetic Algorithm (GA)** to optimise parameters for a **rule-based trading strategy** evaluated via full historical backtests, selecting for **risk-adjusted performance** (Sharpe ratio).

Note: In this project, "predicting" refers to **parameter optimisation via backtesting**, rather than training a standalone price-forecasting model.

What I Built (End-to-End)

Data pipeline: Loaded raw market CSV datasets (e.g., Binance/index data), normalised timestamps, and standardised inputs into **OHLC** format suitable for indicator computation and backtesting.

Strategy implementation: Implemented a custom long/short strategy integrated with the `backtesting` framework, generating signals using **RSI**, **candlestick pattern triggers**, and **Williams %R**, with realistic simulation settings (e.g., commission and execution assumptions).

GA optimiser: Implemented a complete GA workflow (population initialisation, selection, crossover, mutation, elitism, generation loop) for parameter search over a configurable strategy parameter space.

Chromosome design: Represented candidates as **configurable numeric gene vectors** with explicit parameter ranges, making the search space extensible.

Mutation design: Implemented a **two-tier mutation** scheme (coarse resampling + bounded fine-tuning) to balance exploration and exploitation.

Parallel evaluation: Parallelised candidate fitness evaluation using Python multiprocessing (e.g., `ProcessPoolExecutor`) to reduce runtime across populations and make the search computationally tractable.

Fitness function: Defined fitness as **Sharpe ratio derived from full backtest statistics**, optimising risk-adjusted results rather than raw returns alone.

Reproducibility: Maintained a clear project entry point and documented run steps/dependencies to support repeatable experiments.

Core CS Skills Demonstrated

Algorithm design & implementation: Custom GA operators and optimisation loop; structured representation of candidate solutions.

Systems & performance: Multiprocessing-based parallelism; practical runtime/performance trade-offs in iterative evaluation.

Software engineering: End-to-end pipeline ownership; clean interfaces between data, strategy, optimiser, and evaluation.

Empirical evaluation: Backtest-driven metrics and risk-aware objectives; experimental framing documented in the dissertation.