# Derivative Instrument Collections Manager – Ben Sandham

My collections manager will be based on financial derivatives, which are financial products that base their value on an underlying security.

## Requirements

### Core Functionality – Data Management:

* Ability to read data from CSV and JSON files
* Ability to write data to CSV and JSON files, e.g writing Options that match a certain criteria to a CSV file.
* Ability to read and write to SQL Databases, e.g using SQLite
* Ability to rewrite specific data references
* Ability to “tag” specific securities based on fundamental and quantitative factors e.g Underlying Ticker, Contract Type (Call/Put), Strike, Expiry, Price, IV, Moneyness
* Ability to store multiple entries for each security to store changes in data, e.g ITM/OTM

### Performance Tracking:

* For any given derivative, the user should be able to:
  + Calculate the contract value
  + Find the change in contract value over 1H, 1D, 1W, 1Month time periods
  + Track Greeks (delta, rho, gamma, vega, volGa, theta).
  + Track the risk of the option based on the Greeks.
* Ability to create your own portfolio of “favourite” securities, stored on a separate page. This is to allow for easier tracking.

### Visualisation and UX/UI:

* Matplotlib will be used to provide graphical interpretations of price, IV, value.
* Seaborn will be used to generate a volatility surface model for the IV.
* We will use a website to present the UI. The link between the website and backend will be a Flask API, and the actual UI will be done with Streamlit.
* Integration with yfinance API to get real-time market news insights. This will be displayed on its own page with references for each relevant option.
* Visualisation of “winners” and “losers” of the day. E.g the top 5 securities with the highest daily gain, and the top 5 securities with the biggest daily loss.

## User Defined Use Cases

### User 1: Junior Quantitative Researcher at a Hedge Fund random-picture

* Profile: Quantitative Researcher working on US Equity Derivatives strategy research. Strong educational background.
* Desired Functionality: Full access to the Greeks and IV, detailed history of the Greeks/IV for strategy backtesting, notional risk calculation and visualisation, performance tracking. Export a selection to CSV.

### User 2: Owner of a non-financial business, part-time trader



* Profile: Risk conscious, limited time to trade and perform analysis, limited background in financial mathematics.
* Desired Functionality: Simple overview of Options – potentially hiding some more complex data like the Greeks from view, basic view of risk, spacious and non-crowded interface, easy to navigate design.

### User 3: Professor of Finance at a University



* Profile: Highly educated researcher / lecturer
* Desired Functionality: Access to historical data about a given option, ability to export a set of data to CSV or JSON to further analyse with Python for demonstration in their lectures, plenty of visualisation options for IV, Moneyness and Vol surfaces to help improve understanding.

### User 4: Independent Retail Trader



* Profile: Strong contextual understanding of Options, limited mathematical theory.
* Desired Functionality: Integration with real-time news platforms to better inform decision making, view of all Options ranked by IV to identify potential opportunities, performance visualisation for securities in the portfolio / “favourited” securities.

## Application Design and Architecture

*flowchart LR*

*User[User] -->|Interacts| StreamlitInterface[Streamlit Interface]*

*subgraph StreamlitInterface [Streamlit Frontend]*

*MyPortfolio[MyPortfolio Page]*

*OptionViewer[OptionViewer Page]*

*DayMovers[DayMovers Page]*

*end*

*StreamlitInterface -->|Flask API Requests| FlaskAPI[Flask Backend API]*

*subgraph FlaskAPI [Backend Components]*

*PrepareOptionData[PrepareOptionData]*

*displayOptionData[Display Data]*

*subgraph DataSources*

*CSVSource[CSV Files]*

*JSONSource[JSON Files]*

*SQLDatabase[SQLite Database]*

*YFinanceAPI[yfinance API]*

*end*

*OptionAnalysis[Option Analysis - Contract Valuation - Risk Calculation]*

*PerformanceTracking[Performance Tracking- Time Series Analysis - Comparative Metrics - Vol Surfaces]*

*Visualization[Matplotlib/Seaborn\nVisualization]*

*end*

*DataSources -->|Raw Data| PrepareOptionData*

*PrepareOptionData --> displayOptionData[Display Data]*

*displayOptionData[Display Data] --> |Processed Data| OptionAnalysis*

*OptionAnalysis -->|Metrics| PerformanceTracking*

*PerformanceTracking -->|Results| Visualization*

A screenshot of a computer

Description automatically generated