Data Engineering Capstone Project Plan

Name: Richard Lo

# Data Source(s)

Intended Primary Data Source: Oldschool Runescape Wiki Real-time Grand Exchange Prices

Documentation: <https://oldschool.runescape.wiki/w/RuneScape:Real-time_Prices>

Hourly API: <https://prices.runescape.wiki/api/v1/osrs/1h>

Static Data: <https://prices.runescape.wiki/api/v1/osrs/mapping>

Secondary Data Source: Oldschool Runescape Official Website

Documentation: <https://www.reddit.com/r/2007scape/comments/3g06rq/guide_using_the_old_school_ge_page_api/?rdt=63004>

API: <https://secure.runescape.com/m=itemdb_oldschool/api/catalogue/detail.json?item=20997>

Notes: The theme of capstone has deviated from the original scope of collecting data about real stocks and options. This decision came about due to the depreciation of older reliable APIs such as Yahoo Finance, and the difficulty in obtaining the necessary permissions and API keys for current data sources such as Alphavintage, Polygon, or FinnHub. Further difficulty arose from the legality of using such keys for a Streamlit dashboard. As such, these APIs were discarded. When the scope of the theme was opened to a more wider staging area, the videogame Oldschool Runescape was considered. It can be utilised as a real-time data source for information on in-game items, not too dissimilar to stocks in the world of finance. Hopefully, the project will yield skills that are transferrable to other API sources in the future.

# Objective

Capstone project will consist of a two-part system: a real-time data pipeline, and a Streamlit dashboard. The pipeline will extract data from the API, containing time series data of economic information like item prices and volume traded. Data extracted from the API will be cleaned and transformed before being loaded into the targeted database. Some initial ideas on data cleaning and transformation might include data type handling, or feature engineering. So far, only item prices and volume are available from the API, but feature engineering can yield information such as the mid and micro price. Data stored in the database can then be accessed by a Streamlit application to display economic fluctuations, and enable stakeholders to query items they would be interested in.

# Technical Plan

The data pipeline will manifest as a python script running on a CRON job hosted on the Digital Futures server, collecting data from the data source hourly at xx:05. The python script will extract data from (possibly multiple) API(s) in JSON format. The API request must also satisfy the wishes that the API owner has set, that is, the API ping must contain information about how their data will be used. The API documentation states clearly that requests that do not satisfy wishes will be rejected. The data is then subject to cleaning, processing, and transforming with the help of the Pandas library. The data is then sent into a PostgreSQL database, where sensitive credentials are stored in an external environment file. On the front end, the user will interact with a dashboard application, hosted on Streamlit community cloud, which will access the PostgreSQL/Pagila database to display the statistics and visualisation of the tracked assets.

# Potential Risks

Working with Python libraries like Pandas, Requests, Psycopg2 are new to me, so I will need to research and work through this part. Streamlit is a tool where I have very limited knowledge and I have not had experience connecting it a SQL database in this way. Other risks may include, but not be limited to the following:

* CRON job failure. I will need to deliver the pipeline script in a timely manner, and keep communications open with the team working on the CRON job.
* API failure. Should the primary API fail, back up data may be taken from the secondary API.
* Database failure. Should the database fail, it may be worth investigating backup methods for storing data.
* Credential sensitivity. Pipeline source code will be freely available on GitHub, so an environment file has been created to house sensitive data.
* Pipeline operational failure. Written code must be clean, double checked, and tested multiple times before deployment.
* Streamlit library limitation. The requirements file must be kept up to date with all necessary dependencies that keep the front-end application running.

# Timeline & Goals

* By End of Wednesday 26th of June – Project plan and scope outlined, and Data Pipeline code started.
* By End of Thursday 27th of June – Minimum viable product for Data Pipeline must be delivered to ensure data collection can start as soon as possible.
* 28th of June – Digital Futures field trip. Capstone project will be absent for one day.
* By End of Tuesday 2nd of July – Minimum viable product for Streamlit application.
* By End of Wednesday 3rd of July – Diagram and presentation completed.
* 4th or 5th July – Capstone deadline.

Extra time will either be spent troubleshooting or extending goals… Speaking of goals:

* Data pipeline must be delivered to the CRON job team
* Streamlit application must show data and visualisation based on user queries
* Data pipeline output must be clean (e.g. correct data type), and be in at least first normal form (primary key must exist, data must not be duplicated, data must be atomic)

# Additional Targets

Assuming minimum viable product targets are met for both the data pipeline and Streamlit application, I can extend my goals by considering:

* Clean code and documentation: I would like to ensure that the code is easily understood by interested parties wishing for a deeper understanding of back-end logic.
* Feature engineering and data manipulation: The data currently only provides very basic metrics such as bid price, ask price, and volume. Additional features such as mid-price, micro price, moving average, and volatility can be calculated from the given data.
* Pipeline efficiency: It is worth considering that redundant data may already exist in the database. Is there a way of querying only the new data so that computational power can be saved?
* Pipeline robustness: Failover handling would be an impressive feature to have, especially due to the time sensitivity of the capstone deadline of July 5th. Perhaps pipeline operation failure can be addressed by an email notification. An API failure of the primary data source could trigger data mining of the secondary data source. A database failure could dump data into an emergency csv file.
* Expand Streamlit scope: Although Streamlit will display data and visualisation, further considerations can be made into exactly how data can be presented in a meaningful way for the stakeholder(s).

# Appendix: Data Diagram Proposal

