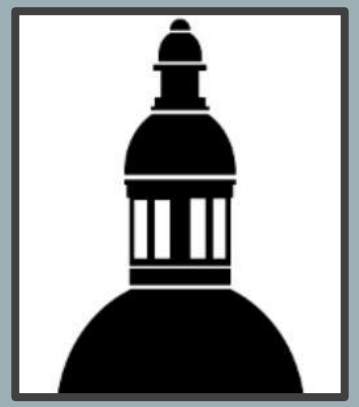


QUEEN'S TOWER CAPITAL

Engineering Division

Development Plans

Thomas

OVERVIEW

- QT Engineering is aimed for developing skills for students to prepare for their career in quantitative trading and finance
- The project is a prototype of a trading system that could be used by banks or hedge funds written in Python, Matlab and later in Java
- We aim to provide learning opportunities for people with different technical skills and experience, so that people can choose to be working more independently and with more supervision
- We encourage our members to develop their own ideas and be an independent software developer / data scientist

Software development

Code comments

Enhancement of functions

- *Use of optional parameters, keyword parameters*
- *Inherit a function with underscore*
- *Debug and error log*

Parallelization and scalability concerns

Collaboration and communication

Technical resources: Check Github page for technical guides

Workflow design and solution design

Good Software Development practice

Come prepared for project meetings: read the slides before meeting so that each meeting can be shortened to 30 minutes

Breakdown large problem into smallers

Write standalone and reusable functions, generalize design so that others can build on your work

Documentation and communication

Github Commit Message

Ask your friends for help: Slack

Software Development life cycle

Design: Identify business needs and design a technical solution

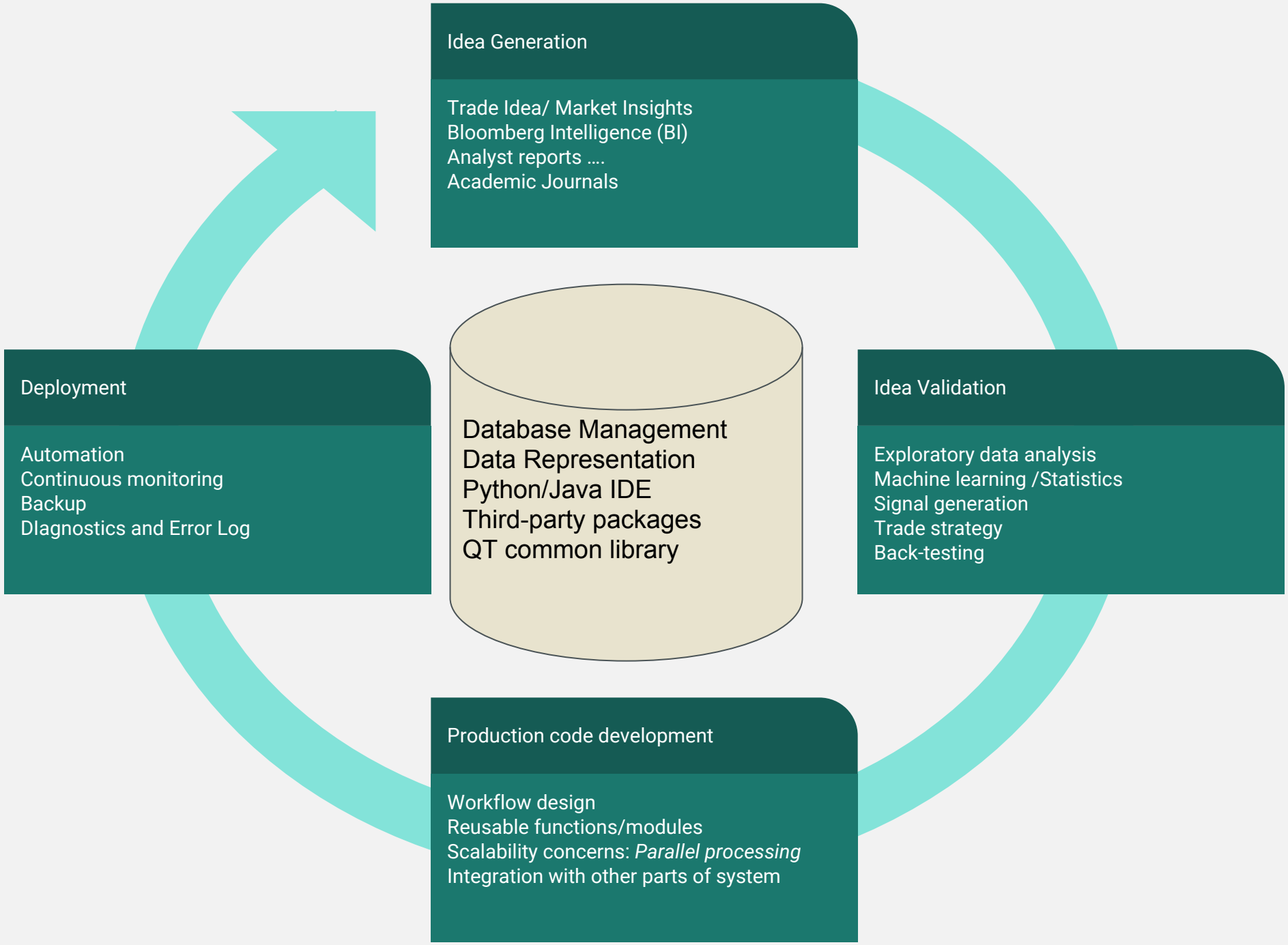
Prototype: Make a standalone prototype to verify the design is feasible

Implementation: Develop functions and workflows to fulfill the business needs

Integration: Test the software developed and integrate into the whole system

Enhancement : Based on user feedback, add features to meet the changing demands of end users

Maintenance : Based on hardware and software upgrades,


Idea Generation
Trade Idea/ Market Insights
Bloomberg Intelligence (BI)
Analyst reports ….
Academic Journals
Idea Validation
Exploratory data analysis
Machine learning /Statistics
Signal generation
Trade strategy
Back-testing
Production code development
Workflow design
Reusable functions/modules
Scalability concerns: *Parallel processing*
Integration with other parts of system
Deployment
Automation
Continuous monitoring
Backup
Dlagnostics and Error Log
Database Management
Data Representation
Python/Java IDE
Third-party packages
QT common library

Development vs Production

	Development	Production
Database	Local Arctic DB Csv files	Online MongoDB Google sheets
Code	Stand-alone Discovery/proof of concept Python/R/Matlab	Integrated Efficient and fault-proof Python/Java/C++
Philosophy	Creativity Complexity	Reliability Efficiency Simplicity
People	Data Scientists	Infrastructure Engineers

Basic packages

Python packages:

Standard library: re,os,datetime.logging,subprocess,json,

Pandas: https://github.com/pandas-dev/pandas

Data Structure: Array, Dictionary, DataFrames, json string,

Database design: https://www.tutorialspoint.com/mongodb/index.htm

Preparation

Create a Github student account / Overleaf student account

Install Selenium IDE https://www.seleniumhq.org/projects/ide/

Install Git Bash

Install MS VS Code / MS Visual Studio Community (For development)

https://www.imperial.ac.uk/admin-services/ict/self-service/computers-printing/software-hub/

Install Anaconda (For Production / Deployment)

Preparation

Python version: please keep both 3.5 and 3.6 compiler as separate virtual environment on conda and VS code

Python version at library: 2.7, Anaconda 2

MongoDB: Currently 3.6, Moving to 4.0

Preparation

Install MongoDB:

https://docs.mongodb.com/manual/installation/
https://docs.mongodb.com/compass/current/#compass-index
https://mlab.com/

Windows: create a .bat file to start loading the local database
"C:\Program Files\MongoDB\Server\3.6\bin\mongod.exe" --dbpath d:\qtdb\mongodb\data

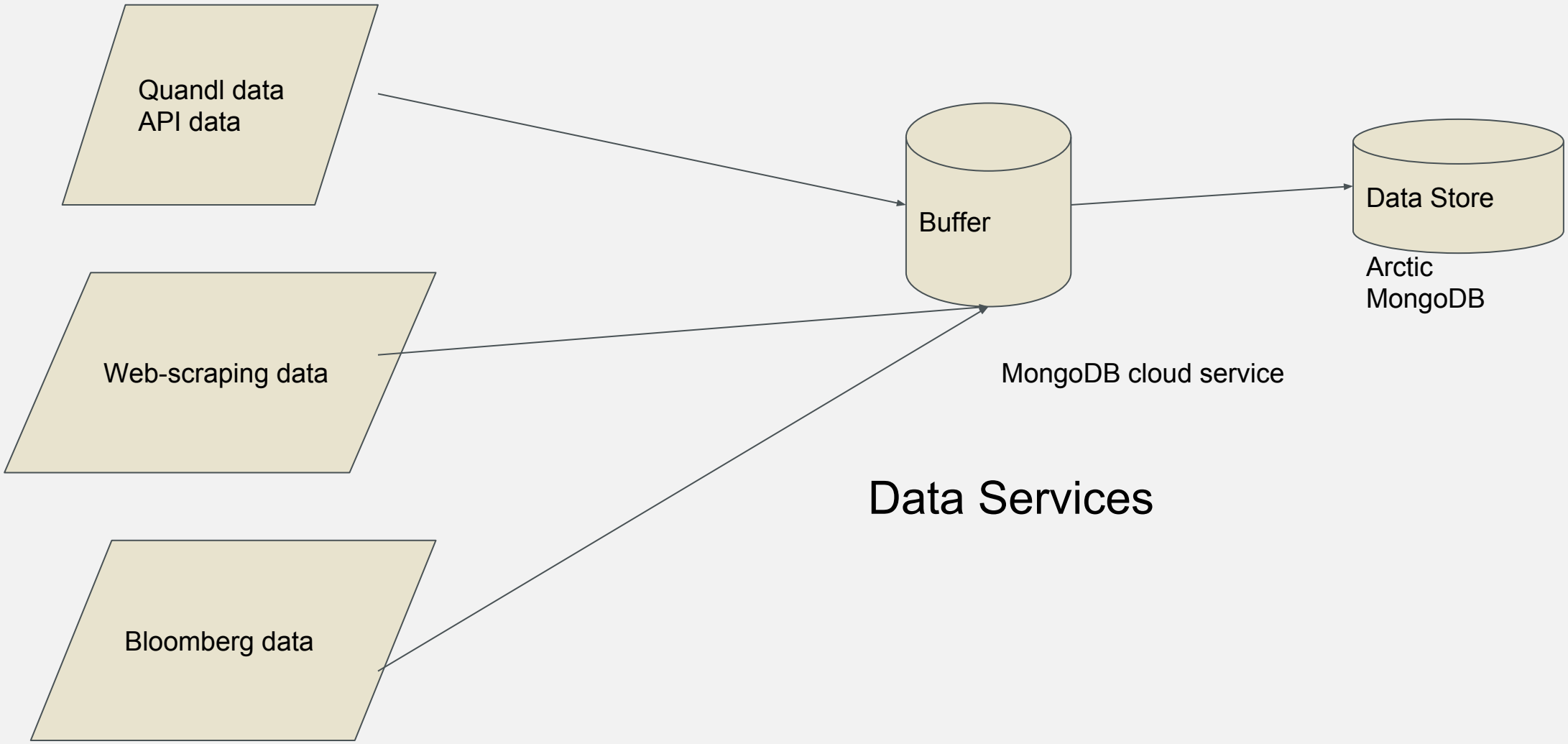
Preparation

Install python packages:

Conda: conda install …. Use pip install if not found

MS Visual Studio: use the package manager provided

https://docs.microsoft.com/en-us/visualstudio/python/tutorial-working-with-python-in-visual-studio-step-05-installing-packages?view=vs-2017


Quandl data
API data
Web-scraping data
Bloomberg data
Buffer
Data Store
Arctic
MongoDB
MongoDB cloud service
Data Services

Data Services

Overview:

Data Selection/Config files

Data acquisition/ API management/Web scraping and Automated browsing

Data Processing

Database design

Data Services

Data import/export function:

Moving data from MongoDB to another

Moving data from MongoDB to Arctic

Batch upload of csv files to MongoDB

Moving data from MongoDB to Google Spreadsheets

Sending text/csv attachments and dataframes through email

Data Services

Data preprocessing

Datetime conversion/ Trading hours and calendar

Numbers conversion/ Currency formats

Remove duplicate/ forward fill/fillna

Set index/multiindex

Data Services

Bloomberg API

Get data from bloomberg terminal: historical data, time series data, current data, bulk data

Search for securities in bloomberg terminal

Quandl API / Web scraping

Web scraping of news websites/ social media feeds

Quandl API

Tasks to do

Bloomberg API

Create conda virtual environment and install blpapi

Develop functions to get data from blpapi (blpapiwrapper.py)

Develop configurations for data we need

Quandl / IEX / Alpha Vantage /

https://www.quandl.com/ , https://www.alphavantage.co/

https://iextrading.com/developer/docs/, https://api.tiingo.com/

https://www.barchart.com/ondemand/free-market-data-api/faq#data_included

Data Services

blpapi https://www.bloomberg.com/professional/support/api-library/

(note: we have our own way to install on library computers)

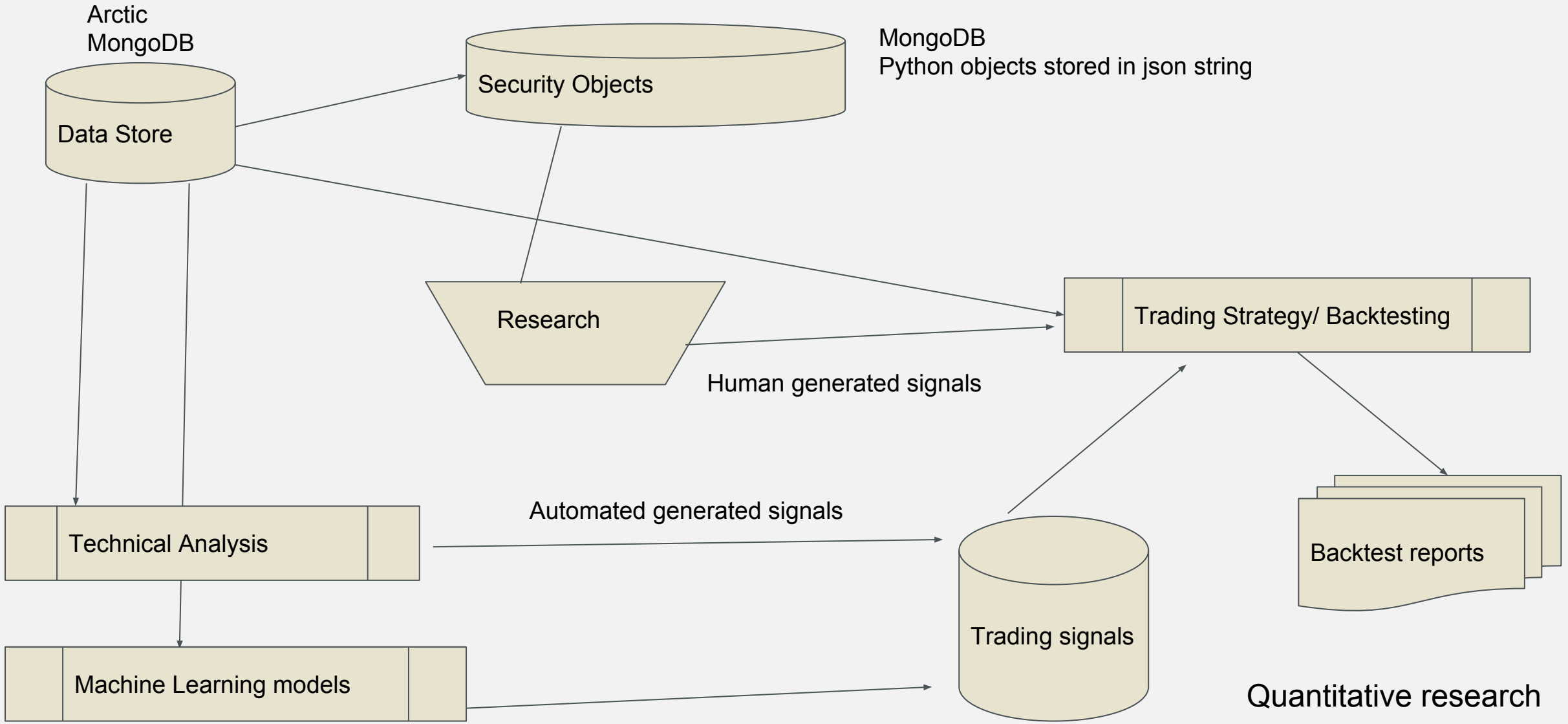
pymongo https://github.com/mongodb/mongo-python-driver

selenium https://github.com/seleniumhq/selenium

requests https://github.com/requests/requests

gspread https://github.com/burnash/gspread

arctic https://github.com/manahl/arctic


Arctic
MongoDB
Data Store
Security Objects
MongoDB
Python objects stored in json string
Research
Trading Strategy/ Backtesting
Human generated signals
Technical Analysis
Automated generated signals
Backtest reports
Trading signals
Machine Learning models
Quantitative research

Quantitative Research

Overview

Configuration and custom data feed for backtesting

Object-oriented design for security objects

Technical analysis and machine learning models

Trading strategy

Data visualization

Quantitative Research

Configuration and custom data feed for backtesting

Build flask application for accessing data from local and cloud database

Build custom data feed class on backtester

https://github.com/backtrader/backtrader/blob/master/backtrader/feeds/pandafeed.py

Build task-scheduler for running backtests and report generation

Data Visualization

Generate historical price charts and save as jpg files

Quantitative Research

Object-oriented design for security objects

Design Class and methods for representing tradable objects

- Represent our understanding of the security and constructing our pricing models
- Reference to database for pricing data, fundamentals data and other data sources
- Design: How to represent security by regions and asset class portfolio by construction methods
- Serialise https://stackoverflow.com/questions/10252010/serializing-class-instance-to-json/10252138
- Properties https://docs.python.org/3/library/functions.html#property
- Abstract class https://www.python-course.eu/python3_abstract_classes.php
-
- Example: USDEUR is an instance from the base class of currency which is inherited from the abstract class
- The currency class should have properties to get most current price data, get the recent historical prices, compute technical indicators, get the recent news insights, generate theoretical prices from different models

Quantitative Research

Technical analysis library

https://github.com/mrjbq7/ta-lib

https://github.com/bukosabino/ta

Technical indicators

Volume, Volatility, Trend, Momentum

Style factors

momentum, market cap, value, mean reversion, and volatility

Quantitative Research

scipy https://github.com/scipy/scipy

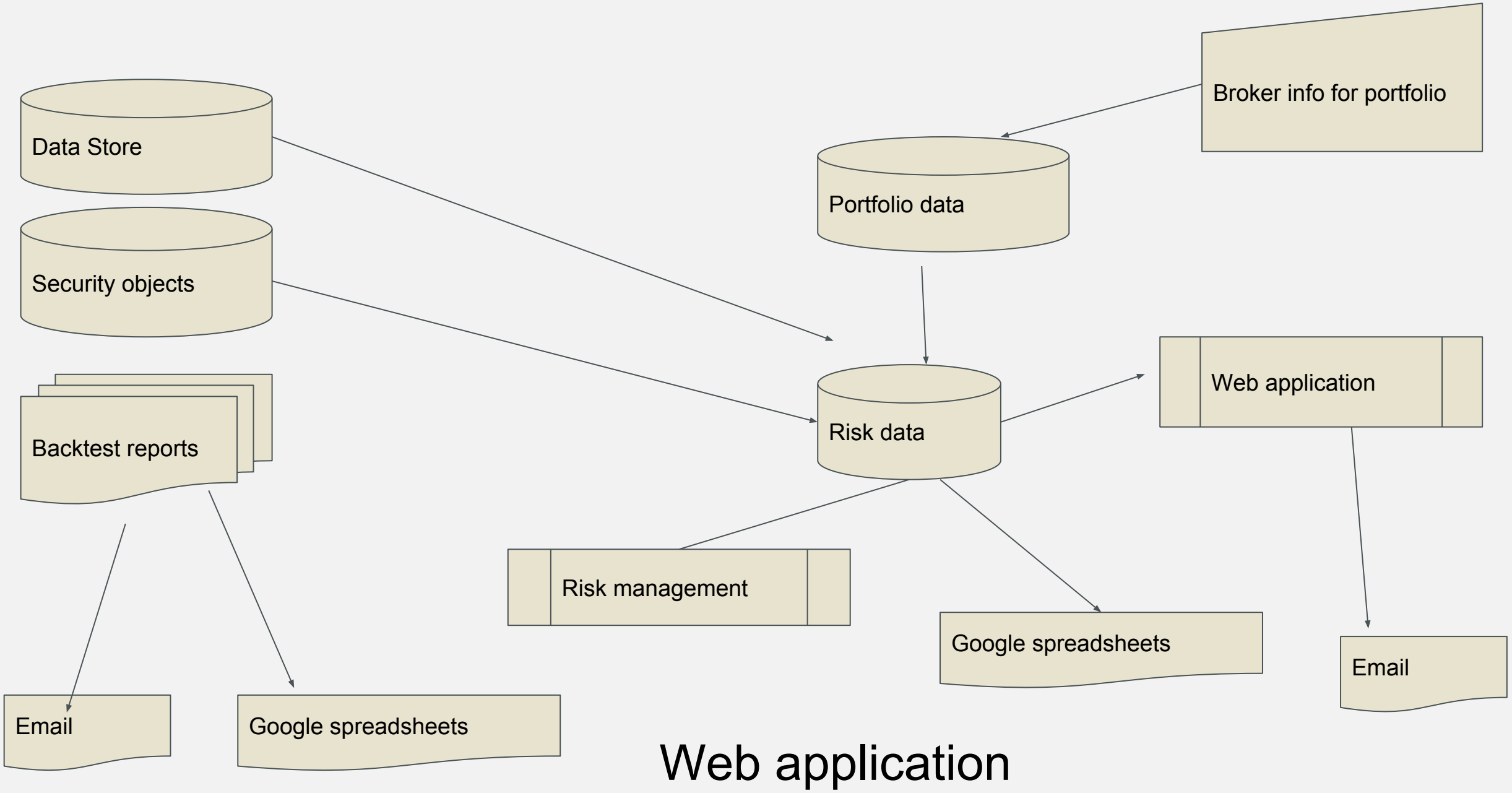
tensorflow https://github.com/tensorflow/tensorflow

scikit learn https://github.com/scikit-learn/scikit-learn

statsmodels https://github.com/statsmodels/statsmodels

pykalman https://github.com/pykalman/pykalman

convex optimization https://github.com/cvxopt/cvxopt


Data Store
Security objects
Backtest reports
Broker info for portfolio
Portfolio data
Risk data
Web application
Risk management
Google spreadsheets
Email
Email
Google spreadsheets
Web application

Web applications

Overview

Risk Management models

Design of real-time cloud database for recent historical data

Design of web application for showing up-to-date changes in portfolio

Report services in email and google spreadsheet for data

Web applications

Design of real-time cloud database for recent historical data and portofolio

Capped collection https://docs.mongodb.com/manual/core/capped-collections/

Change streams: https://docs.mongodb.com/manual/changeStreams/

Automate download of statements from brokers and prasing of csv files

Recent news and analysis from web and internal resources

User input form for adding analysis insights from QT

Web applications

Design of web application for showing up-to-date changes in portfolio

Interactive flask application for displaying portfolio information, searching recent prices and information about a security

Historical price from bloomberg (with delay of 15 minutes)

Recent and historical holdings data (sync with database)

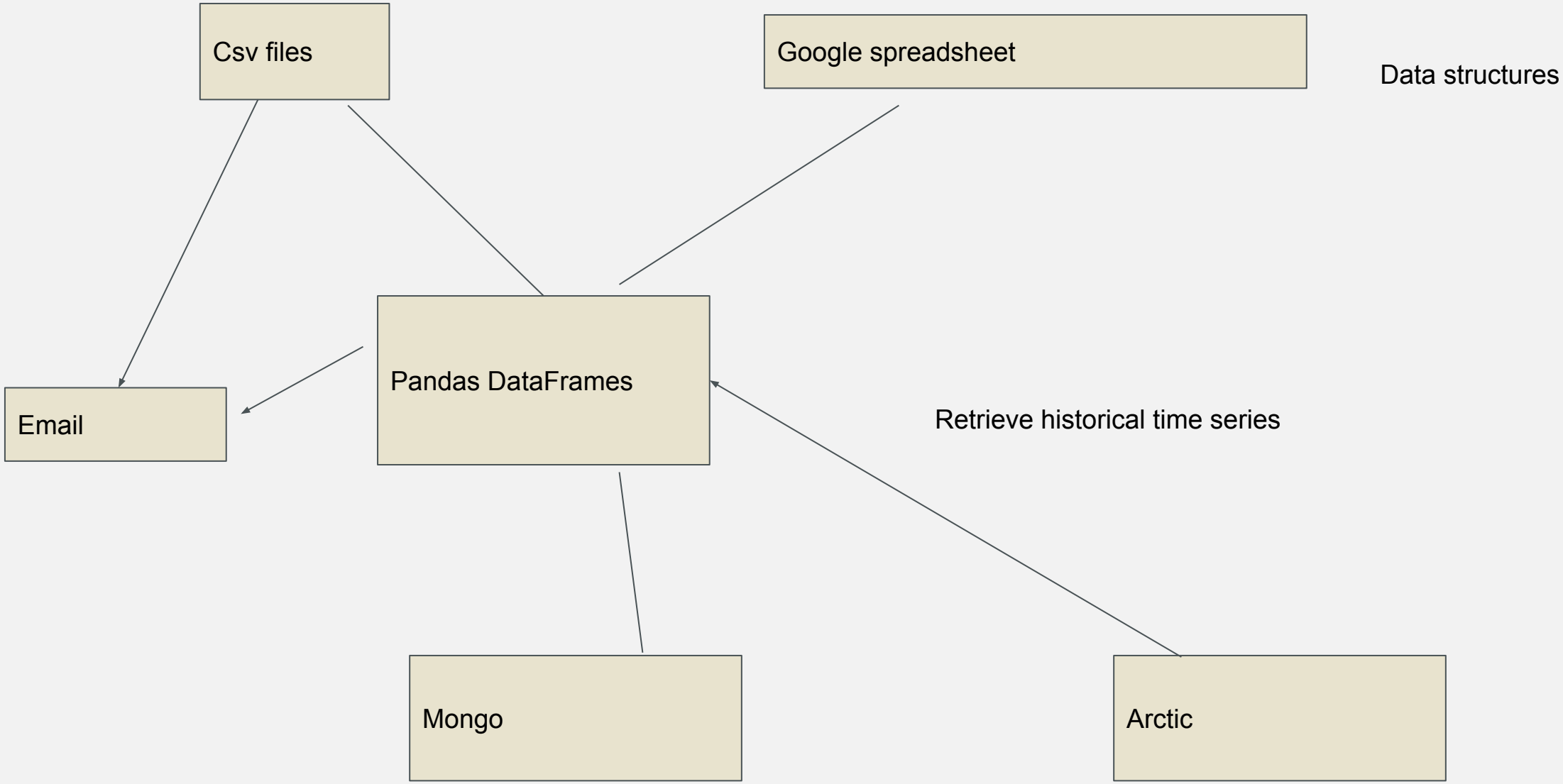
News report and analysis view (web scraper and internal resources)

Web applications

flask https://github.com/pallets/flask

matplotlib https://github.com/matplotlib/matplotlib

plotly https://github.com/plotly/plotly.py


Csv files
Google spreadsheet
Data structures
Pandas DataFrames
Email
Retrieve historical time series
Mongo
Arctic

Datafeed

Database Schema Design

MongoDB Buffer: Data obtained from bloomberg terminal, online sources

Arctic DB: historical time series of security, sorted by asset class and time frequency

MongoDB production: Real-time cache for recent historical data

Data Service Workflow design:

1. Design individual functions that perform the task on a single security
2. Design the config file, the python file and the bat file and task-scheduler to run the process
3. Advanced: Multi-threads and concurrent tasks on python
4. Alerts for task completed and failure, rollback operations for the last 5 days

Datafeed

Example: Download data every day/week from library

The MongoDB buffer is set up as capped collections with different size limit of documents for each collection

Prepare a config file which contains the list of security, list of fields

Write our own functions to provide the same excel functions in python, two is already done in the github link

We need to build the rest similarly according to blpapi documents

http://bloomberg.github.io/blpapi-docs/ https://github.com/ThomasWongMingHei/blpapiwrapper

Run the scripts so that data are downloaded from bloomberg terminal and uploaded to the drive

Design a workflow and scheduled running the task on library

Datafeed

Example: Download data every day/week from library

Download data from buffer to arctic database

Design a workflow and scheduled running the task on your computers

Set up task alerts for failures and roll-back features (retry download data within the last 5 trading days)

Datafeed

Example: Download data from bloomberg library

Prepare a config file which contains the list of security, list of fields

Write our own functions to provide the same excel functions in python, two is already done in the github link

We need to build the rest similarly according to blpapi documents

http://bloomberg.github.io/blpapi-docs/ https://github.com/ThomasWongMingHei/blpapiwrapper

Run the scripts so that data are downloaded from bloomberg terminal and save as csv

Push data from csv to arctic database

Datafeed

Example: Subscribe to delayed price data from bloomberg terminal

The MongoDB Production is set up as capped collections with different size limit of documents for each collection

Prepare a config file which contains the list of security, list of fields

Write our own functions to get recent data, need to create our own update function for the observer

https://github.com/ThomasWongMingHei/blpapiwrapper

Run the scripts so that data are downloaded from bloomberg terminal and uploaded to MongoDB

We will have “real-time’ risk management system developed in future will depends on this ‘real-time’ data

Datafeed

Example: Database maintenance

Trading calendar: build calendars which allows us to extract

Arctic DB: check that data are unique, forward fill data for intra-day data if needed, check existence of data using trading calendar,remove columns that are all zero

In each transaction, save the original df as a csv file in temp and then delete this copy after the transaction is successful

Idea generation

How trade ideas are generated

Economic theory background: Academic research papers, books...

Discussion forum: Seeking Alpha https://seekingalpha.com/investing-strategy

Your peers: We are bring in experience from our internships

Ideas generation

How to implement a trade idea

Formulate investment thesis

Prepare data source

Universe selection

Build price,volume,volatility,correlation,drawdown,turning point,trend models

Build technical indicators, factors,

Construct trading strategy

Portfolio optimization and risk management

Task Automation

Build tools from datafeed, analytics and back-testing engines

Workflow design

Resources management and progress report

Example: write standalone tasks

Example: (Windows): bat files to run multiple .py files at the same time for multi-threading

Example: Notify status of process on Slack