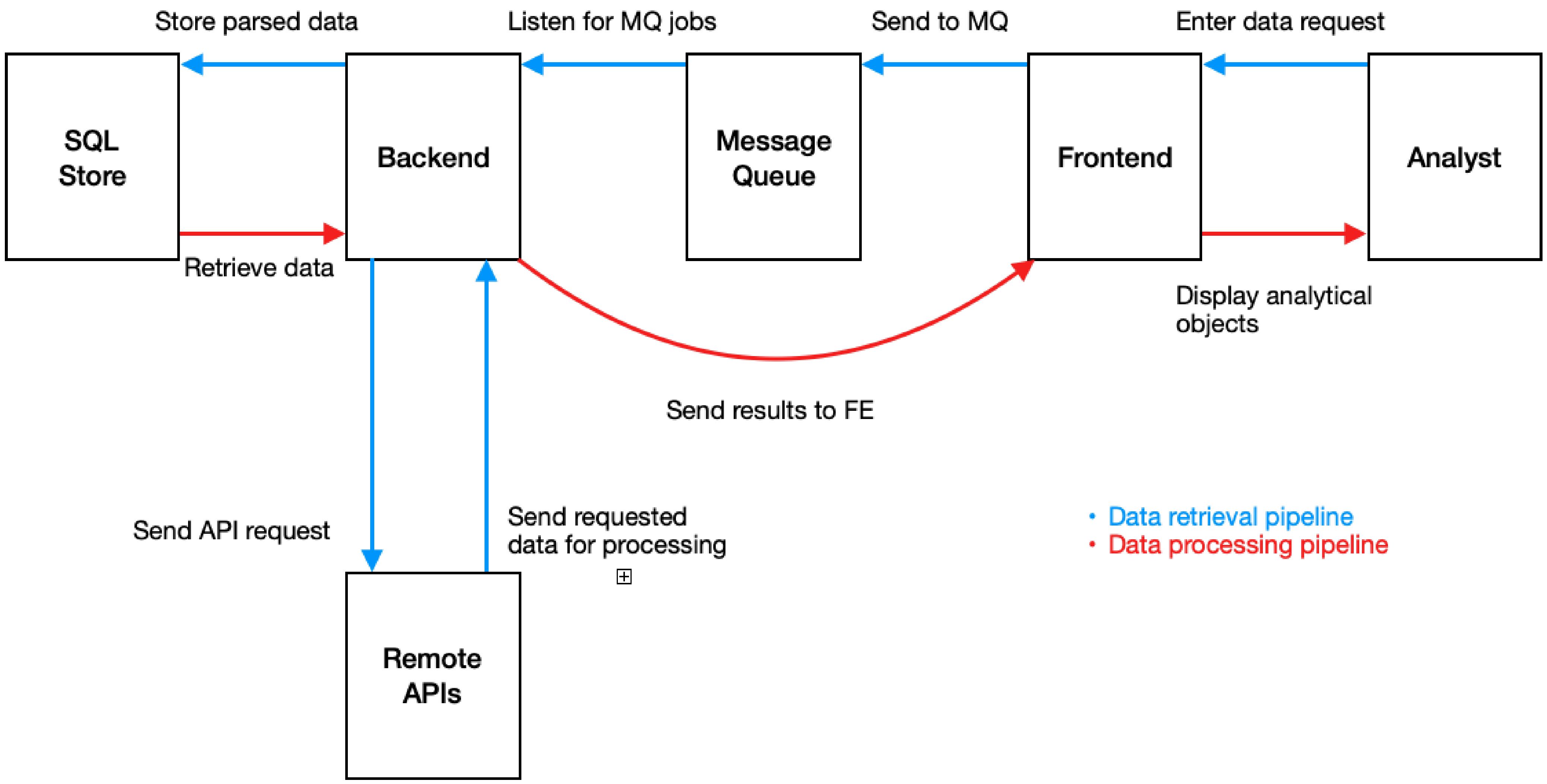
COSC880 Documentation Slides

By: Devere Anthony Weaver

Application Overview

- TODO: Add application goals and functionality this slide

Conceptual Architecture - Overview



- Data retrieval pipeline
- Data processing pipeline

Conceptual Architecture - Components

- **SQL Store** – stores parsed API data from the backend in a structured format
- **Backend** – listens to message queue for jobs submitted by the analyst on the frontend, crafts and sends API requests, parses API response object and sends to SQL store for persistence, performs risk computations based on desired risk model to send to frontend
- **Message Queue** – message queue to store jobs from the frontend initiated by the analyst

Conceptual Architecture - Components

- **Frontend –** presents UI for analyst to enter requests, presents results of computation and analytical objects from backend to analyst
- **Remote APIs –** APIs used to retrieve financial data related to the analyst's request

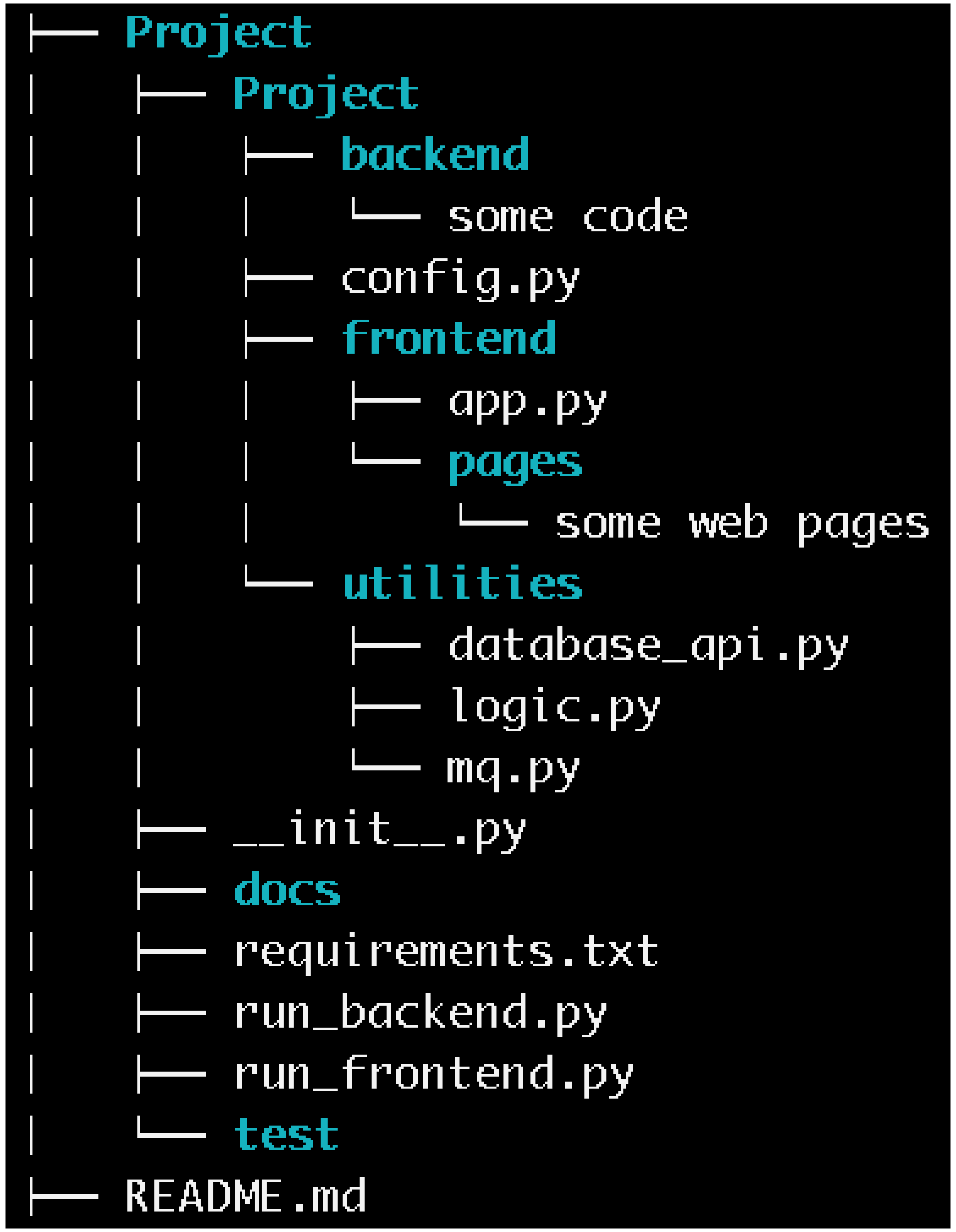
Implementation – Tech Stack

- **SQL Store** – PostgreSQL
  - https://www.postgresql.org/
- **Backend** – Dash (built-on Flask for backend), Pandas, Pytorch (depending on models implemented)
  - https://dash.plotly.com/
  - https://pytorch.org/
- **Message Queue** – RabbitMQ
  - https://www.rabbitmq.com/
- **Frontend –** Dash
- **Remote APIs –** Yahoo Finance, St. Louis Fed (FRED), TD Ameritrade, FastAPI
  - https://developer.yahoo.com/api/
  - https://fred.stlouisfed.org/docs/api/fred/
  - https://tda-api.readthedocs.io/en/latest/
  - https://fastapi.tiangolo.com/

Application Structure

- Application will be structured such that the outermost directory will be the be a Python package with two modules that will execute the application code inside of the Project subfolder and its subdirectories which contain all the application logic

- Using code organized as packages ensures Python interpreter will enforce absolute imports which will prevent any issues with dependencies that typically arise



```
├── Project
│   ├── Project
│   │   ├── backend
│   │   │   └── some code
│   │   ├── config.py
│   │   ├── frontend
│   │   │   ├── app.py
│   │   │   └── pages
│   │   │       └── some web pages
│   │   └── utilities
│   │       ├── database_api.py
│   │       ├── logic.py
│   │       └── mq.py
│   ├── __init__.py
│   ├── docs
│   ├── requirements.txt
│   ├── run_backend.py
│   ├── run_frontend.py
│   └── test
├── README.md
```

TODO

- Add remaining slides that include functional and non-functional dependencies
- Add risk model specs
- Add slides that include code files/snippets/etc. with explanations