Name: Dhruv Dakoria

Course: 4SA3

# Software Architecture Document - 4+1 View Model

## Name:

Trading Bot Application

## Project purpose and audience

The purpose of this software will be to use free public API (alpaca-trade-api) to fetch stock data and use that data to execute paper trades using certain trading strategies. I will be using a cloud database - Amazon DynamoDB to store the trade and transactional data. The application logs will be logged locally to a file as well as to Amazon CloudWatch Logs and a report containing all information will be generated based on database entries.

The target audience is stock market investing enthusiasts who would like to gather stock data and execute paper trades to test out an algorithmic trading strategy.

## Requirements

1. Users shall be able to provide a list of stocks they want to trade with. (FUNCTIONAL)

2. The application shall be able to fetch market data (stock prices) using public RESTful APIs. (FUNCTIONAL)

3. The application shall allow the creation or cancellation of trading orders. (FUNCTIONAL) 4. The application shall allow the execution of paper trades based on a defined strategy. (FUNCTIONAL)

5. The data for the executed/cancelled trades shall be written to a cloud database. (FUNCTIONAL)

1. Application logs shall be available to the users through a log file stored locally. (FUNCTIONAL)
2. Users shall be able to request a report of trades performed. (FUNCTIONAL)
3. API data should be live with the frequency of updates every minute. (NONFUNCTIONAL)

**Scenario Viewpoint**

**Concerns:** Understanding the central functionality of the system **Stakeholders:** All stakeholders, but particularly the end user **Modelling techniques:** UML Use Case Diagram

UML Use case diagram with User and System entities and set of use-cases that are involved.

**Diagram

Description automatically generated**

## Physical Viewpoint

**Concerns**: Mapping of software to hardware, communication protocols and modules related to  
communication  
**Stakeholders**: software architects, software developers  
**Modelling** **techniques**: UML Deployment Diagram

Deployment Diagram showing a user (client device) interacting with a Python application running on the client device which connects to some REST API services (Alpaca trade API), cloud database (DynamoDB) and sends logs to cloud watch logs.

Diagram

Description automatically generated

## Development Viewpoint

**Concerns**: Organization of software modules

**Stakeholders**: Software Developers, Manager

**Modelling** **techniques**: UML Class Diagram

|  |  |  |
| --- | --- | --- |
| Pattern | Role | Classes |
| Model-View-Controller  (MVC Architectural Pattern) | View for handling User Interaction, Model for database access and Controller for communication between model and view as well as executing business logic (executing trade strategy, generate reports etc.) | Model, View,  Controller |
| Singleton Pattern  (Creational Pattern) | App Class is a Singleton that manages the read-only state of the application like configuration data, Alpaca Trade API setup, logging, and the database connection | App |
| Chain of responsibility  (Behavioural Pattern) | Logging to terminal, file, and streaming logs to AWS CloudWatch logs. | Logger,  TerminalLogger,  FileLogger,  CloudWatchLogger |
| Template Method  (Behavioural Pattern) | Defines the skeleton of logging in the superclass but lets subclasses override specific steps of the logging (create\_log\_entry method) without changing its structure | Logger,  TerminalLogger,  FileLogger,  CloudWatchLogger |

Diagram

Description automatically generated

## Process Viewpoint

**Concerns**: Runtime communication

**Stakeholders**: Software Architect

**Modelling** **technique**: UML Sequence Diagram

The most important runtime communication that occurs is between the application and the Redis Labs cloud database, and between the application and the Alpaca Trade API. When generating a report, the report data is first requested from DynamoDB database. The application requests the stock related data to execute the trading strategy as provided by the user by making API calls to Alpaca API.

Diagram

Description automatically generated

The diagram below depicts Report Creation Communication Sequence Diagram using layered approach -

Diagram

Description automatically generated

## Logical Viewpoint

**Concerns:** Functional requirements

**Stakeholders**: End user, Software Architect

**Modelling** **technique**: UML Activity Diagram

The use cases are realized via a series of configurable inputs within a file that allows users to pick between printing and creating a report, selecting trading strategy and applicable stock inputs to trade with. The sequence of menus and options is captured below.

Diagram

Description automatically generated