Team: Buy Bitcoin

Growth Chart
• displayType: string • time: string
render()

The Growth Chart, which is a graphical representation of a customer’s balances (historical and projected) is at the highest level of classes in our system. It relies on several different classes for its information (as will be shown in the diagram below). It has 3 important values it takes, which are the customers asset values considering three different paths to be taken: nothing, save, invest.

Funding
• balances: int • time: string • type: string

calcRounding()
calcIncomePercent()
calcExpenditurePercent()
calcFlat()

The Funding Class, contains information on a user’s value and assets determined by different types of income accrual. It has different operators which calculate user balances depending on different saving methods, as well as selected dates.

Portfolio
• stockPercent: int • currentVallue: int
modifyPortfolio()

The Portfolio, is a collection of all the assets for a user. It shows the percentage of holdings in different investment buckets, and their current value. One of its operators allows for users to update the percentage breakdown of assets in that portfolio.

Team: Buy Bitcoin

Investment Bucket

- stock: int
- percentage: int

addStock()
deleteStock()
modifyStockComposition()

An investment bucket represents a combination of stocks to which a certain amount of money will be distributed according to a certain Stock Configuration. It supports operations to add, delete stocks, as well as to modify distribution configurations.

User

- name: string
- id: int
- google-id: string

authorize ()

A user is the key to the platform, and is the one who tests the platform with their personal data. Users must login through google in our platform, so we need their google account information. The operator is to verify that the user is logged on and active.

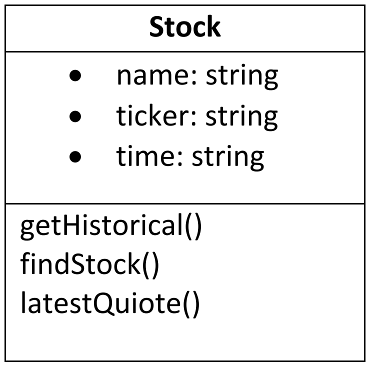
Bank

- name: string
- userId: string
- password: string

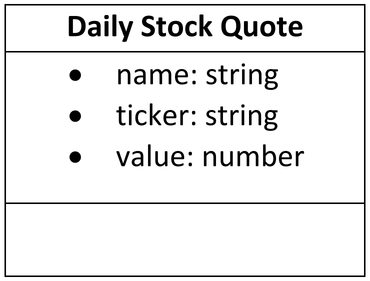
getTransactionData()
getBalances()

The bank class represents the available information collected from an external API that allows us to access a user's bank details, transactions, among other data. The operators allow us to request specific information requests from said bank.

Team: Buy Bitcoin

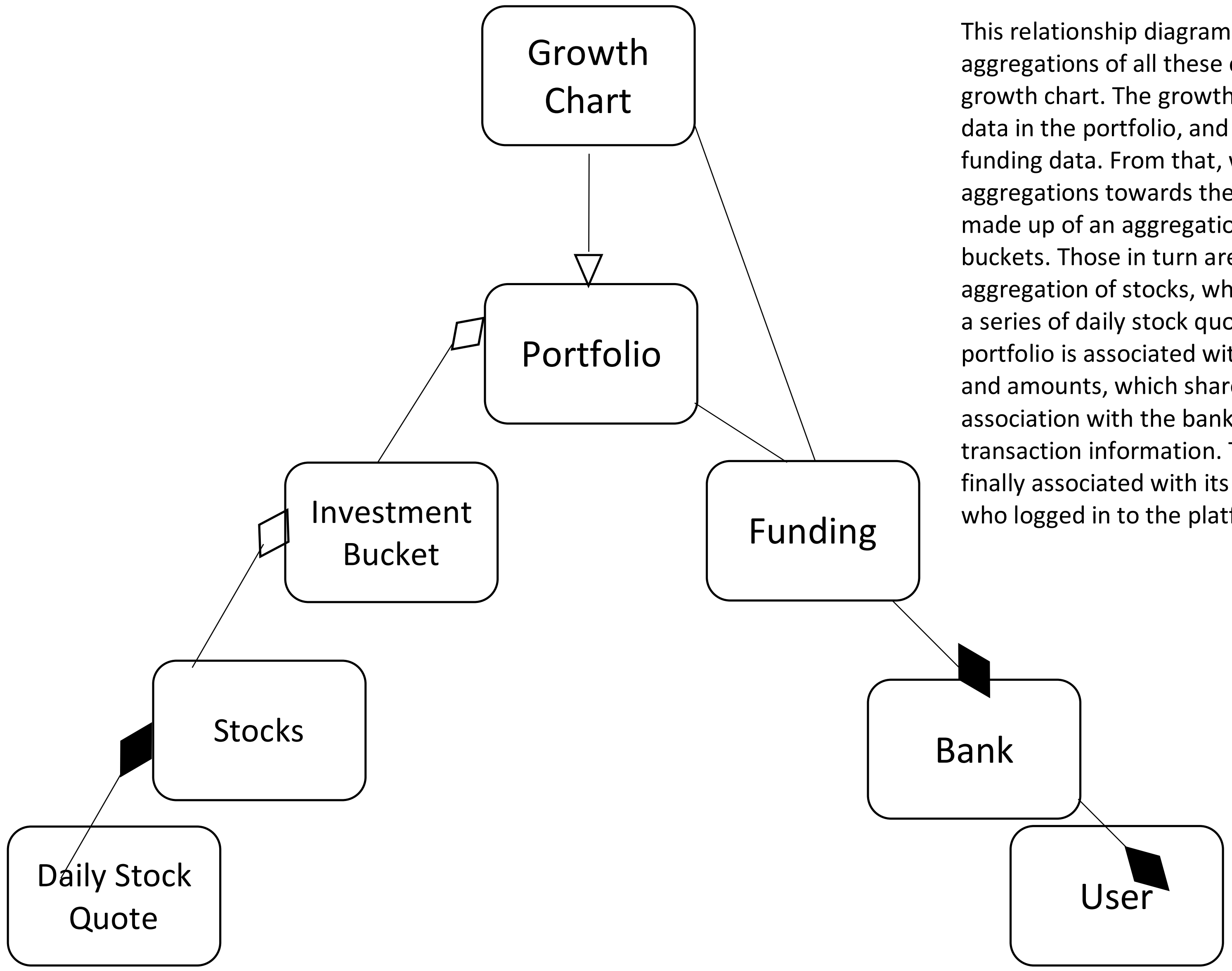


The Stock Class represents a specific stock and ticker with its historical price data. Its operator allows access to stock data available for 10 years prior, if that stock was listed.



This class simply represents a daily quote calculated for a specific stock, which is taken from the stock price over a continuous amount time, and calculated to simplify to a daily value.

Team: Buy Bitcoin



This relationship diagram represents how the aggregations of all these classes leads to a growth chart. The growth chart inherits the data in the portfolio, and is associated with funding data. From that, we see a series of aggregations towards the portfolio, which is made up of an aggregation of investment buckets. Those in turn are made up of an aggregation of stocks, which are comprised of a series of daily stock quotes. Finally, the portfolio is associated with funding methods and amounts, which shares an important association with the bank class and its transaction information. The bank class is finally associated with its corresponding user, who logged in to the platform.

Team: Buy Bitcoin

ADDITIONAL CLASSES:

graphQL
• graphQlTypes: string
makeResponse()

This class exposes the Graph QL endpoint to Django HTTP requests. Its operator makes a response from said endpoint so Django can process it.

Stock market
• name: string • ticker: string

validateTicker()
fetch()

The stock market class opens up endpoints for yahoo finance data to access both historical and current stock information. The operators validateTicker guarantees that a certain stock exists, and fetch will retrieve the information.

Pair info: We all worked on this assignment, with Jordan and Julian dealing with CRC cards and class descriptions, and Nigel and Christophe handling class diagrams and operator, type definition.

Team: Buy Bitcoin

Second Iteration of class diagrams begins here:

TradingAccount
• name: string • profile: string

Totalvalue()
tradingBalance()
availableBuckets()
availableStocks()
hasEnough()

The Trading Account Class has important methods to check if an account is capable of performing a trade, and in reporting balances and distributions of stock, buckets, and other important assets associated with our platform.

TradeStock
• account: string • timestamp: string • quantity • stock
currentValue()

The Trade Stock class is able to track trades relative to specific accounts, and performs an assessment of current value of that trade while factoring in other details. It measures the amount and price of stock that is raded.

TradeBucket
• account: string • timestamp: string • quantity • stock
currentValue()

The Trade bucket class is similar to the TradeStock class but with a more general level of abstraction. It assesses a bucket value, and report its value at a designated point in time.

Team: Buy Bitcoin

DailyInteractions

- id: string
- Date: string
- Description: string

The Daily Interactions class keeps a record of user’s interactions and descriptions of said interactions, these interactions may be funding a checking account, an expense of some sort, etc.

AddTrade

- id: string
- quantity: int
- account: string

The Add Trade class allows a user to add a trade, specifying the quantity, value, stock, and direction of the trade.

InvestBucket

- quanityt: int
- account: string
- bucket:string

The Invest Bucket class invests into a specific bucket with a specific amount of available balance, identifying the bucket by its id, and thus following all of the buckets investment configurations.

Team: Buy Bitcoin

Add Atribute

- Bucket: string
- Desc: string

The Add Attribute Class adds a description to an investment bucket meant for users to identify buckets, and their potential value, allowing them to decide whether to invest in said investment bucket.

Delete Attribute

- Bucket: string
- Attribute: string

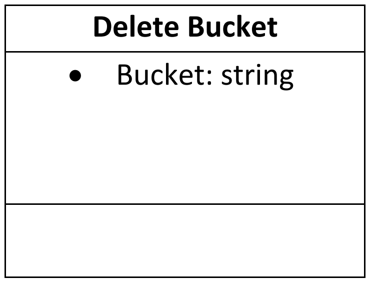
Deletes the Attribute mentioned above. Important for attributes that have become obsolete in reference to a specific investment bucket.

Edit Attribute

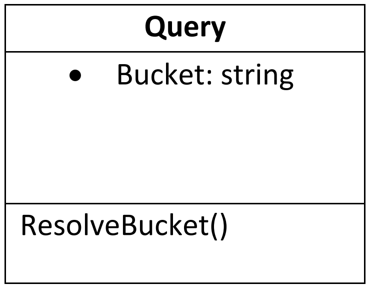
- Bucket: string
- Attribute: string
- Desc: string

Allows for a user to edit a pre-existing attribute, to adapt the description of an investment bucket, and update potential investors on the formation and composition of the bucket.

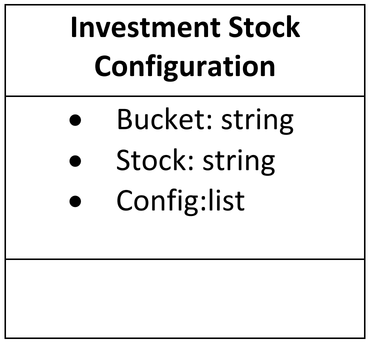
Team: Buy Bitcoin



This class completely deletes a bucket, eliminating it from public and private records, and no longer allowing it to be used for an investment by a any user.

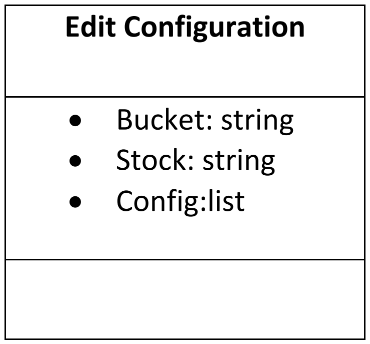


The Query Class queries a specific bucket or set of buckets so that they may be accessed, edited, and updated.



The investment stock configuration class determines the quantity percentage and distribution of stock and investment in a specific bucket, allowing for users and bucket creators to determine the correct, safe, or risky configuration to select in an investment.

Team: Buy Bitcoin



This class calls for the adjustment of the all-important configuration of a bucket, leading to an update of percentages and ratios of investment selection in given buckets, and thus, updating the diversification and vulnerability of a portfolio.

For a detailed explanation of the equivalence partitions and boundary conditions covered, visit the following text file in our Github repository:
https://github.com/Neitsch/ASE4156/blob/master/documentation/EquivalencePartitions.txt

The text above lists Equivalence Partitions with classes divided into attributes and functions, showing detailed pass and fail scenarios for each, including sample inputs that would fail and or pass depending on situation.

The tests for these partitions and classes and functions can be found at https://github.com/Neitsch/ASE4156/tree/master/tests, and they are all invoked using pytest.

Pair info: Jordan and Julian dealt with completing class diagrams and part of test writing, and Nigel and Christophe handled equivalence partitions and further test writing.