Team: Buy Bitcoin

Already examined user stories:
Login: User logs into our platform. (**WIREFRAME A)**

Signup / Connect bank account: User registers and signs into their respective bank account allowing us access to bank data. (**WIREFRAME B)**

Select funding method: User chooses out of three options to determine what funding method they will choose (rounding, income percent, expenditure percent, fixed rate). (**WIREFRAME C)**

Select risk portfolio: User decides what risk they want to take with their assets, determining shape and assets in their portfolio. (**WIREFRAME D)**

Learn more - portfolio: User wants to see what one of the portfolios is made up of, to analyze the specific industries and asset types in the portfolio. (**WIREFRAME E)**

User Stories to add:

Create Investment Bucket: User wants to create a bucket so that he may invest or so other users may invest in the same assets the user is investing in.

Edit Investment Bucket: User wants to edit the bucket so that he may change stocks or configuration of a specific bucket.

Add Stock to Investment Bucket: A user wants to add a stock to an investment bucket, so that the bucket becomes more diversified and varied.

Delete Investment Bucket: A user wants to delete an investment bucket so they do not longer see it or are invested in it.

Select investment Bucket: A user selects an investment bucket from the list of available buckets so that they can explore the composition of the bucket and decide whether to invest in the bucket.

Invest in Bucket: A user decides to fund an investment bucket and invest in it so that they may see returns on investment.

Team: Buy Bitcoin

Adjust Investment Levels: A user decides to modify the levels of funding (ratios for each stock) so that they may see different levels and returns on their investment.

Logout: A user decides to logout so that there information is secure and unavailable to others whom may user their computer.

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

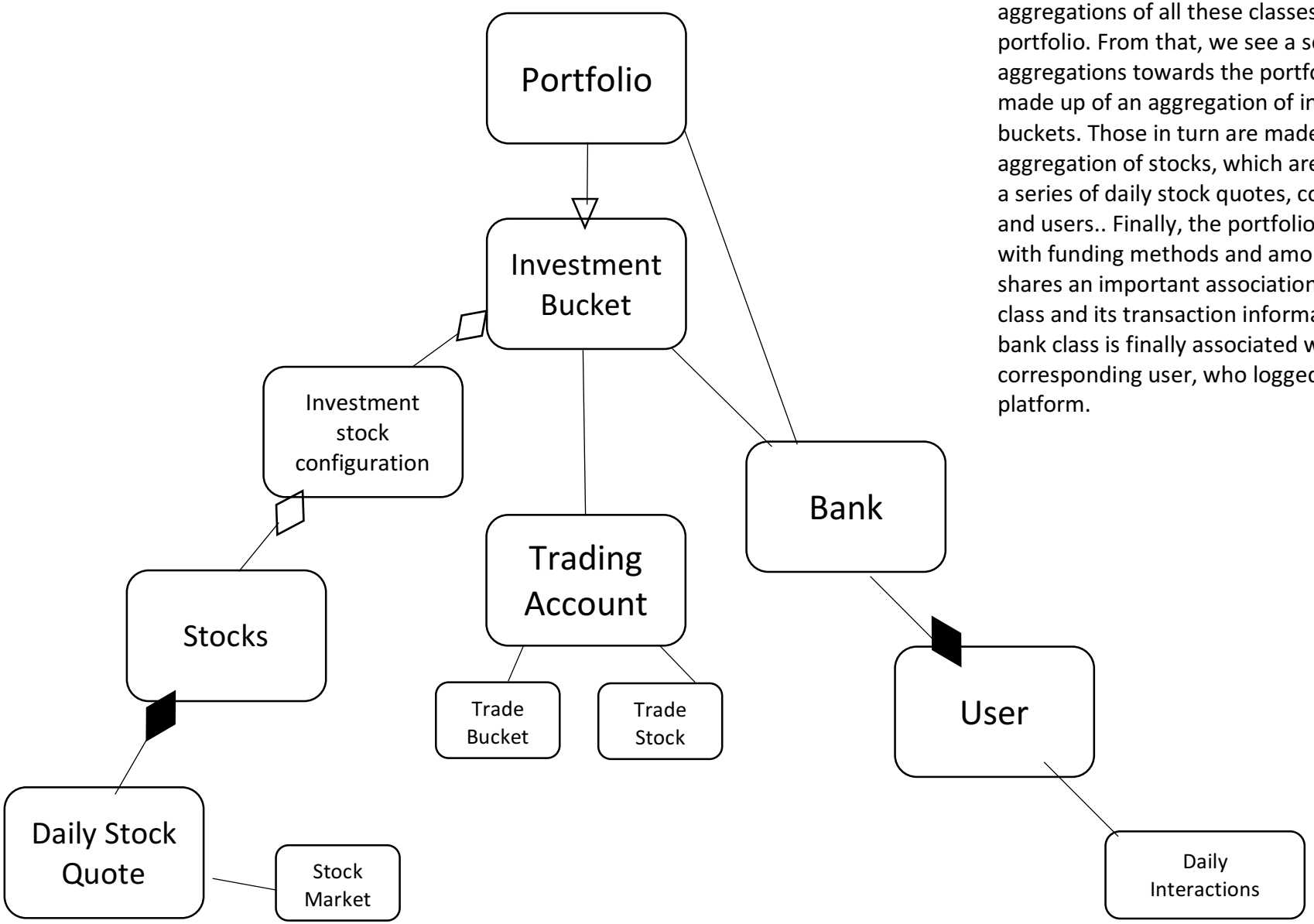
Stock
• name: string • ticker: string • time: string
getHistorical() findStock() latestQuiote()

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.

Daily Stock Quote
• name: string • ticker: string • value: number

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 portfolio. 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, configurations, and users.. 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

Delete Bucket
• Bucket: string

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.

Query
• Bucket: string
ResolveBucket()

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

Investment Stock Configuration
• Bucket: string • Stock: string • Config:list

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

Edit Configuration
• Bucket: string • Stock: string • Config:list

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.

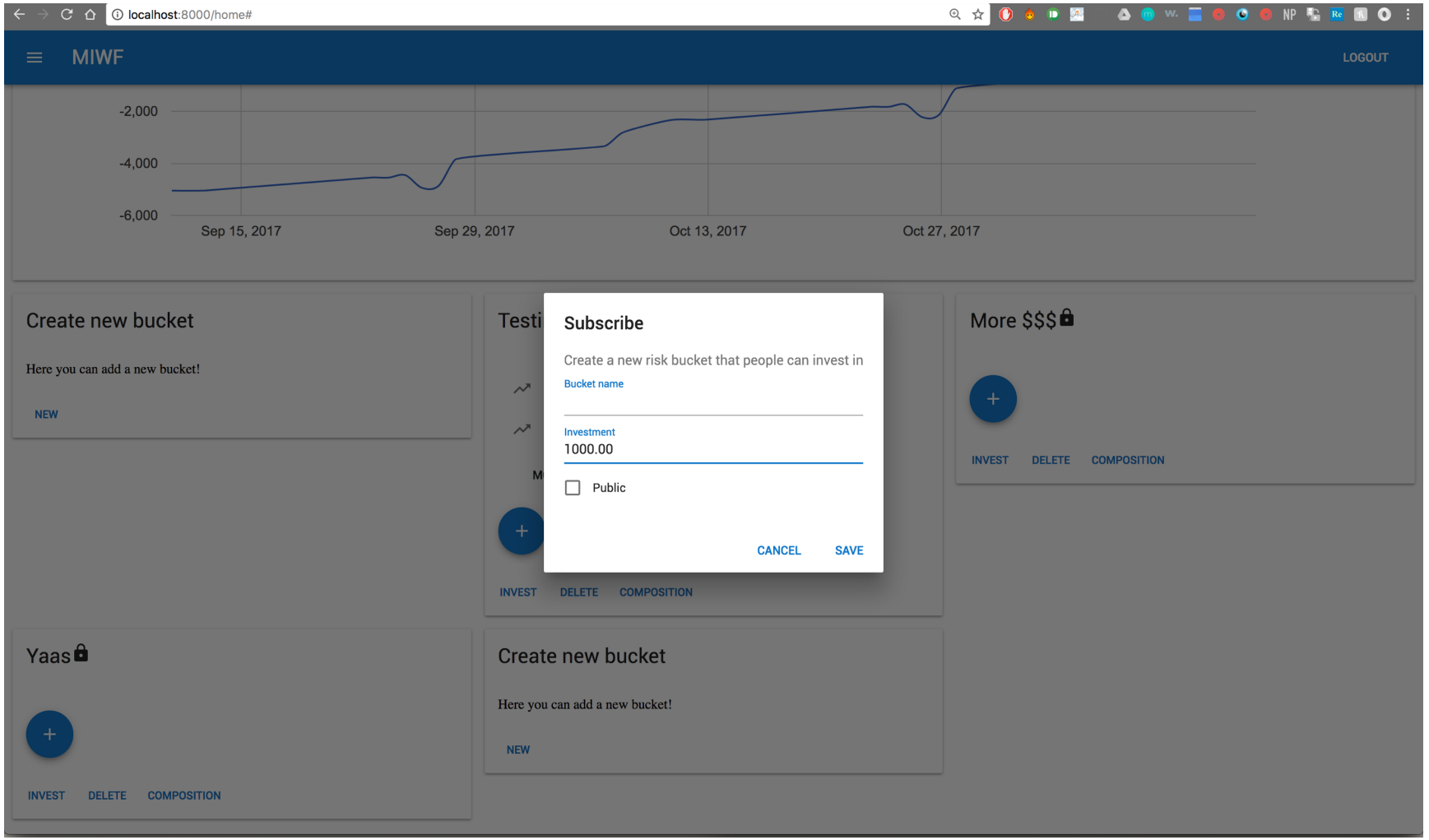
Note: The classes: Edit Configuration, Delete Attribute, Delete Bucket, AddAtribute, AddTrade, are DB writes, which are mutations, but qualify as classes.

Pair info: Jordan and Julian worked on diagrams while Christophe and Nigel worked on determining relationship diagrams and new user stories.

Team: Buy Bitcoin

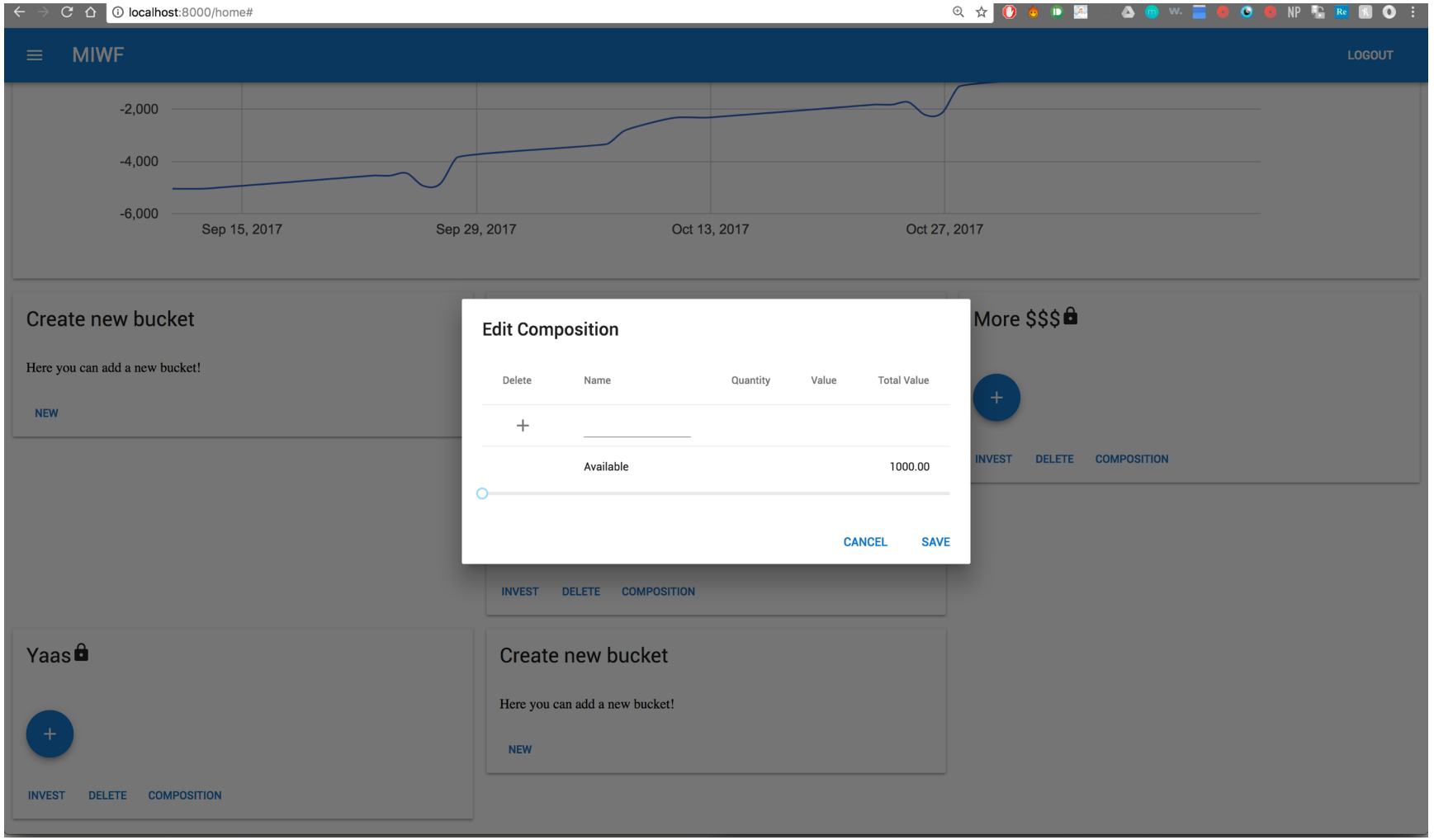
Wireframe user stories:

Create Investment Bucket:



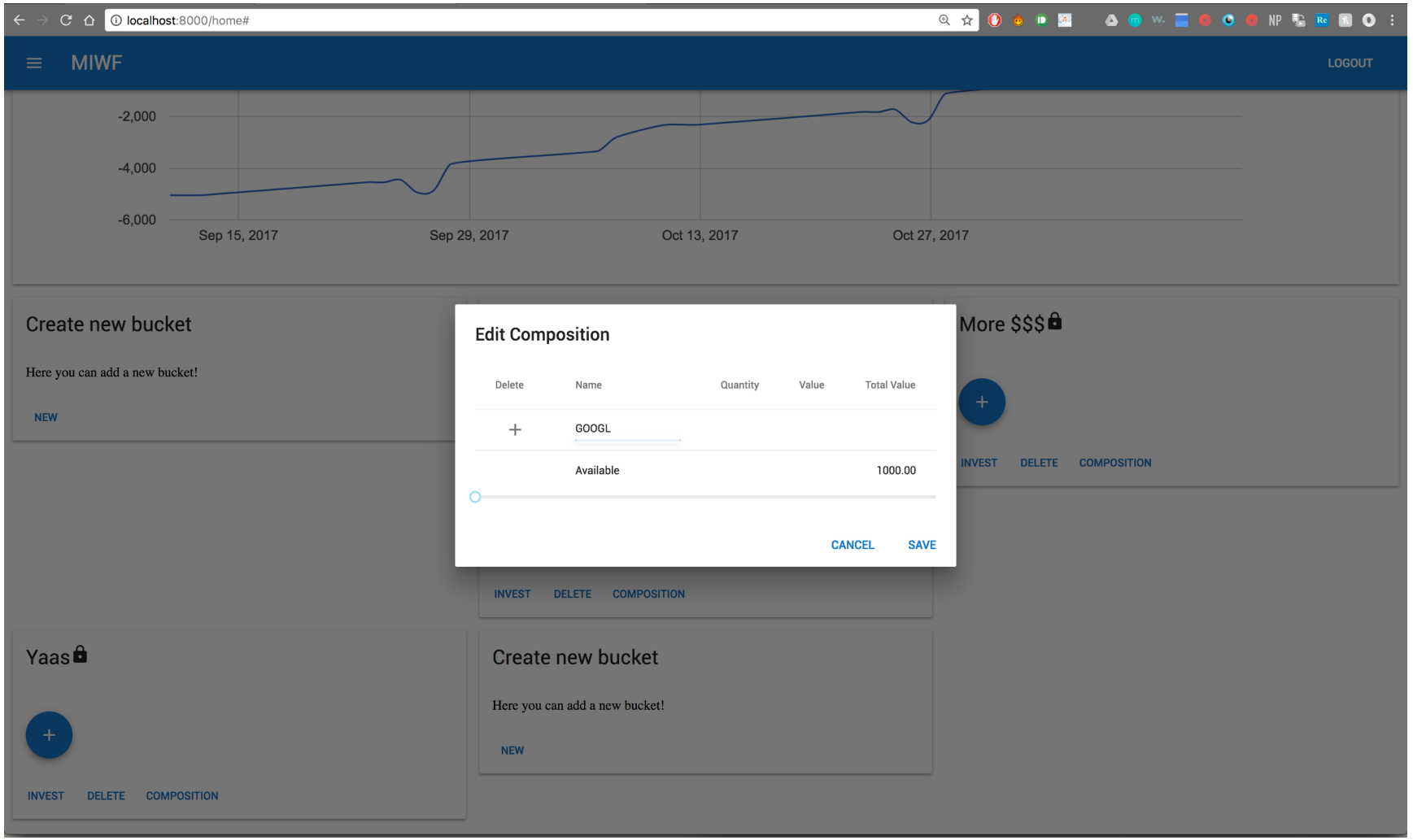
Team: Buy Bitcoin

Edit Investment Bucket:



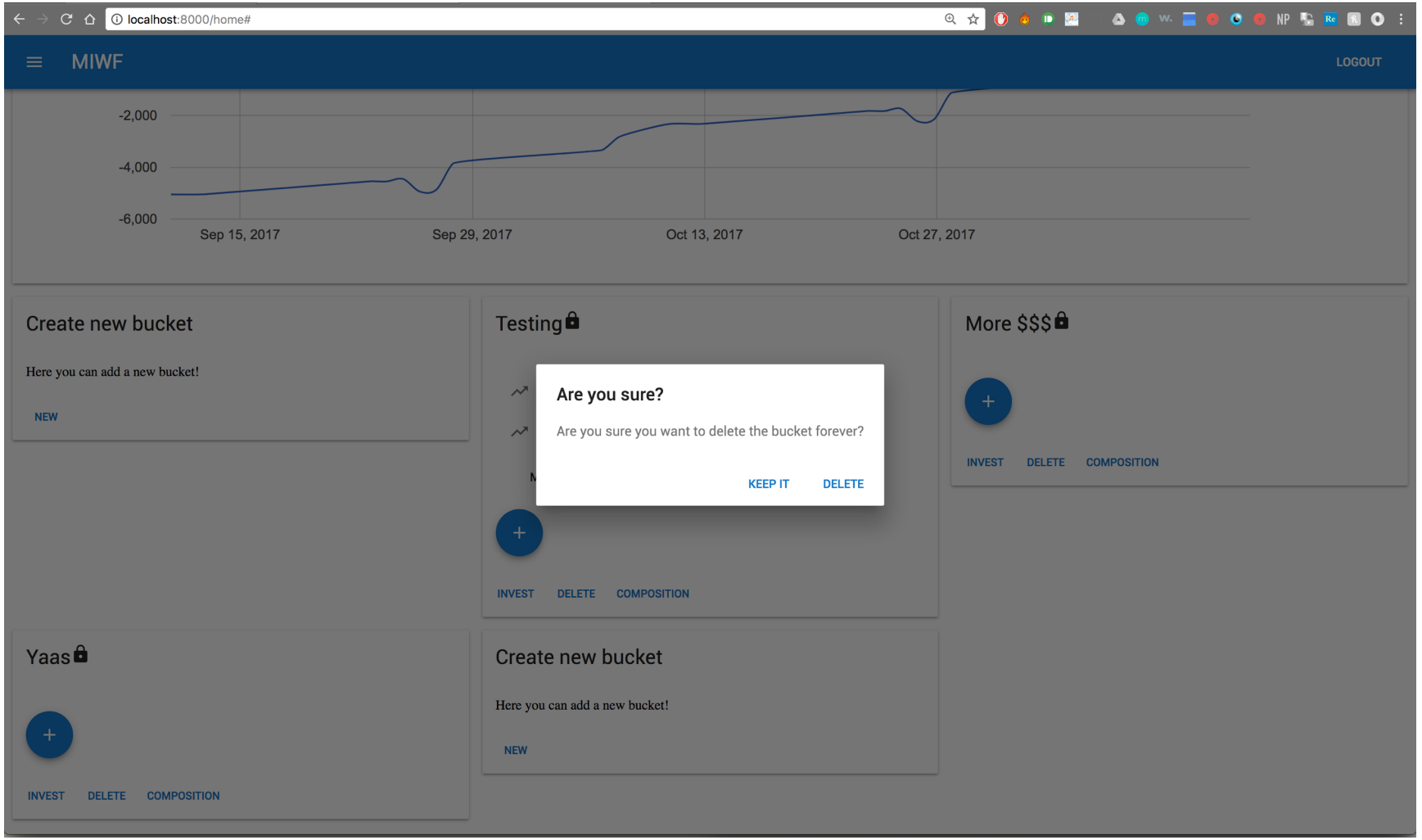
Team: Buy Bitcoin

Add Stock to Investment Bucket:



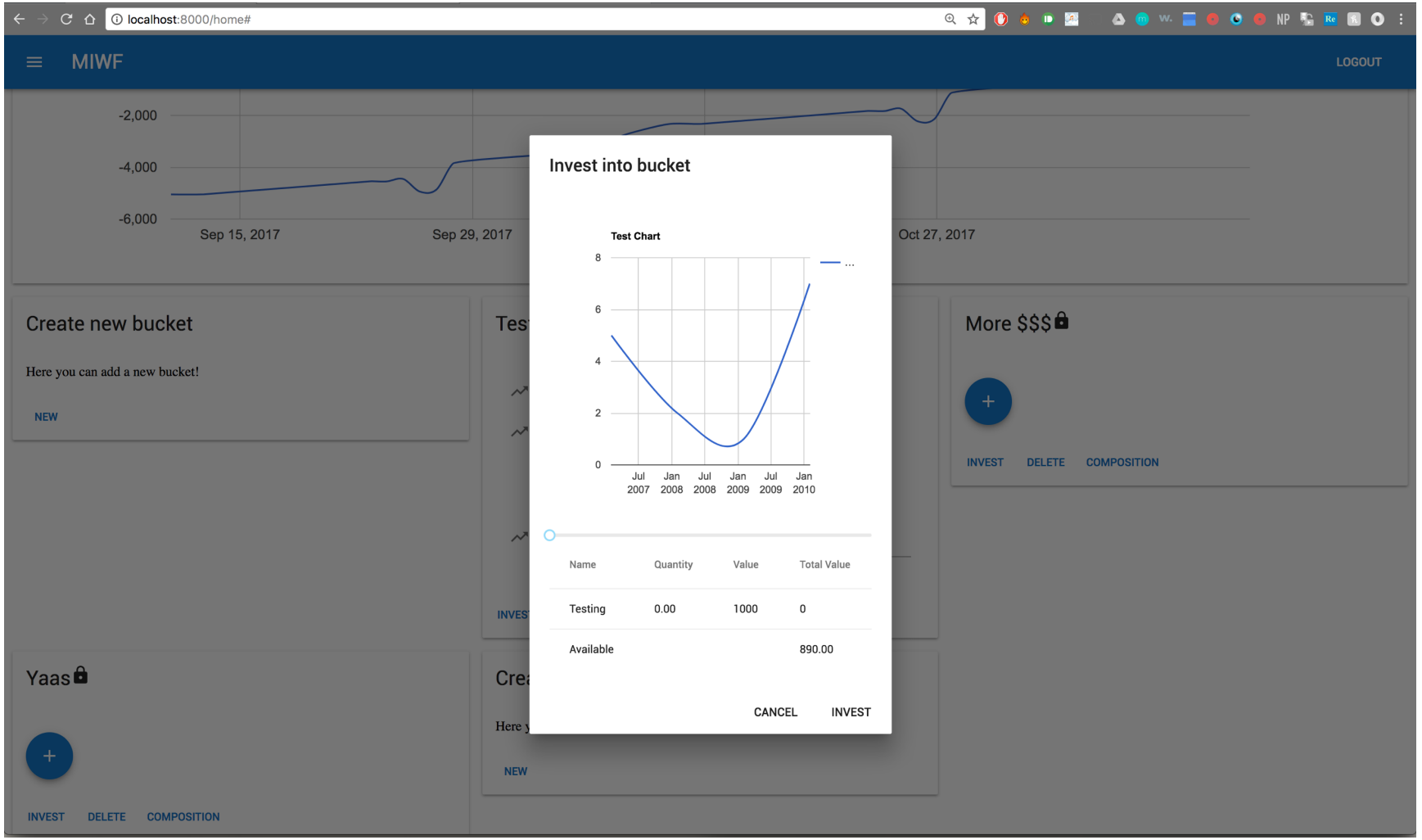
Team: Buy Bitcoin

Delete Investment Bucket:



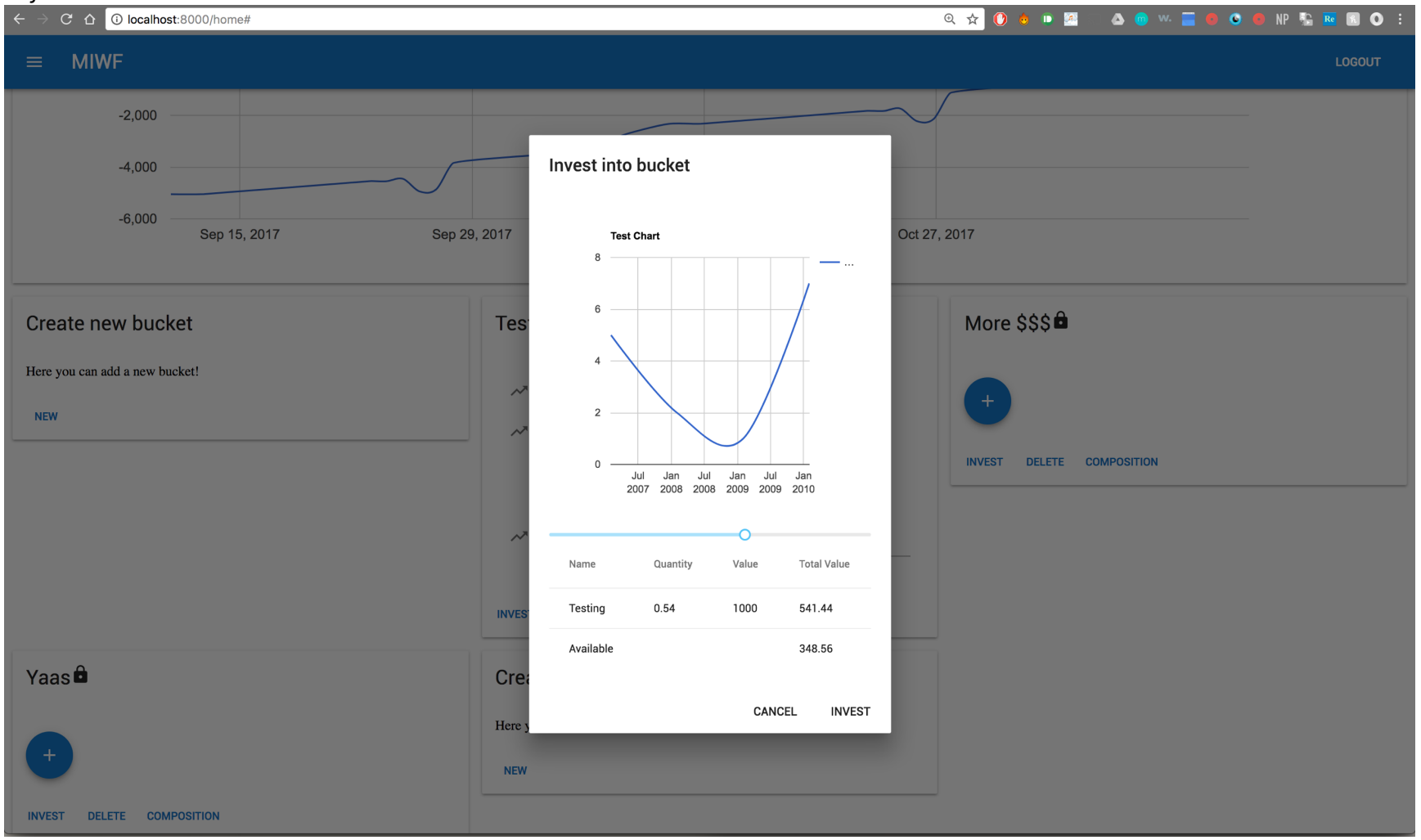
Team: Buy Bitcoin

Invest in Bucket:



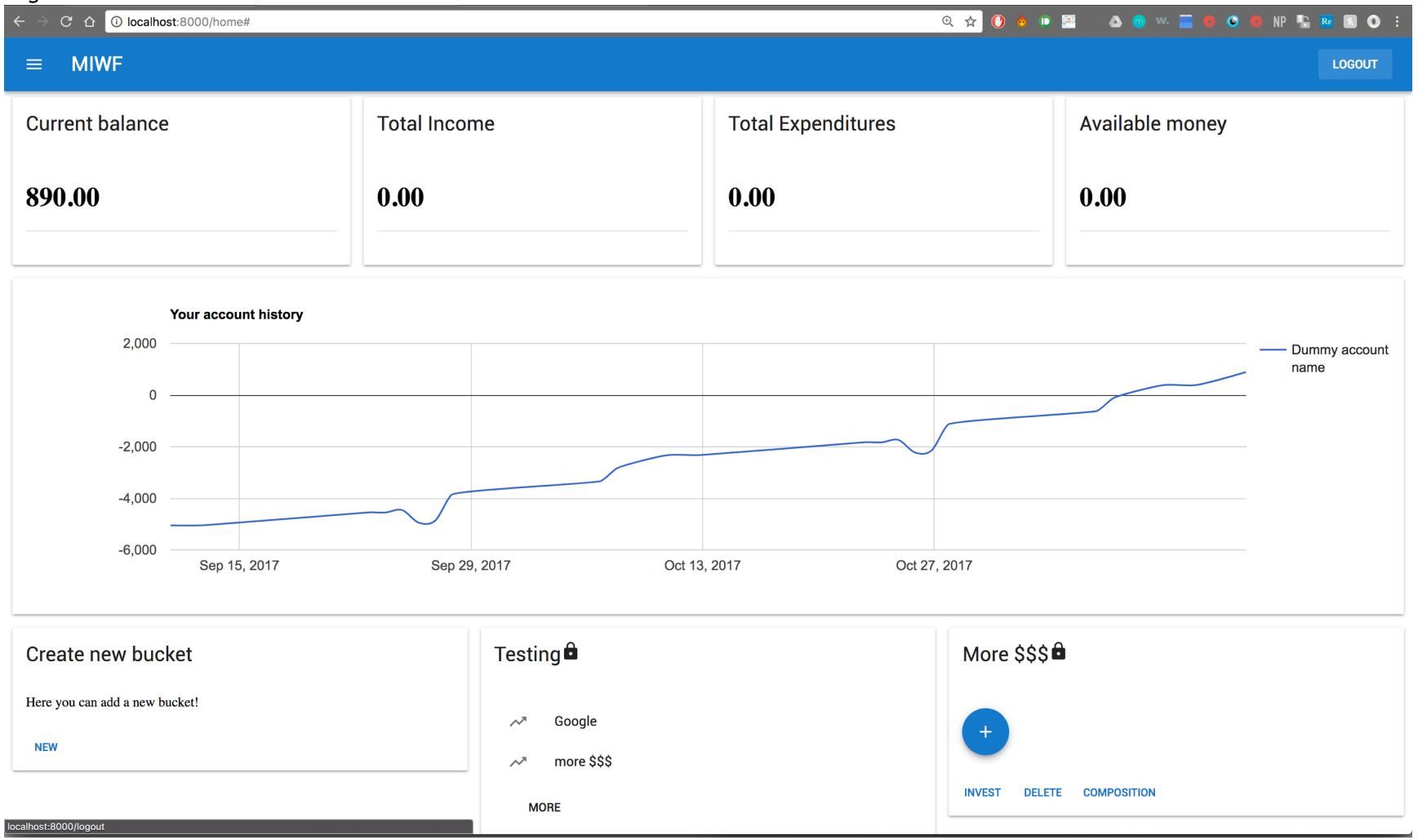
Team: Buy Bitcoin

Adjust Investment Levels:



Team: Buy Bitcoin

Logout:



Team: Buy Bitcoin

Mobile Frames (bonus):