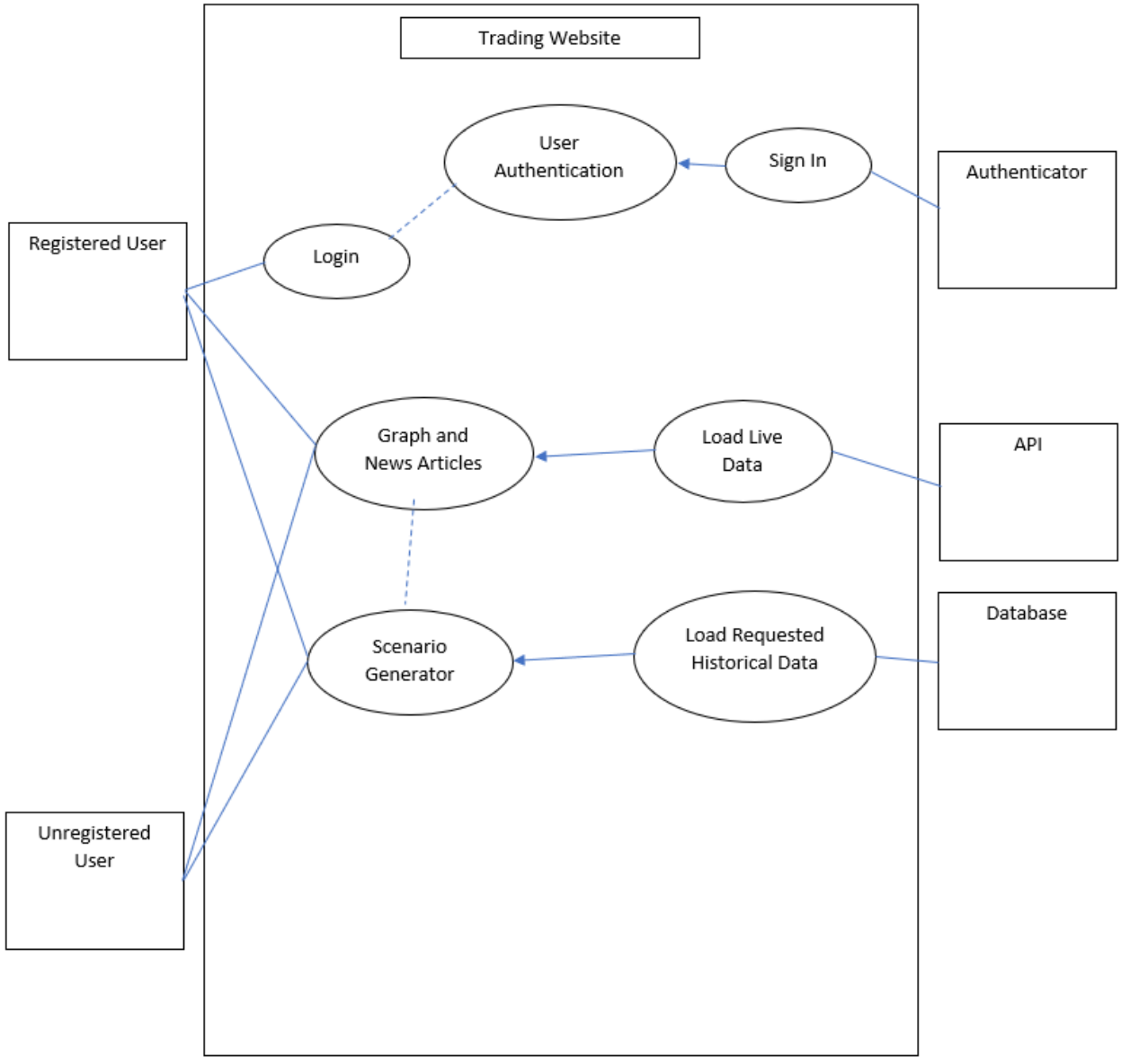
Pet Project – Live and Historical Data

Scenario - I want to view a point in time were bitcoin price was elevated for a large amount of time.

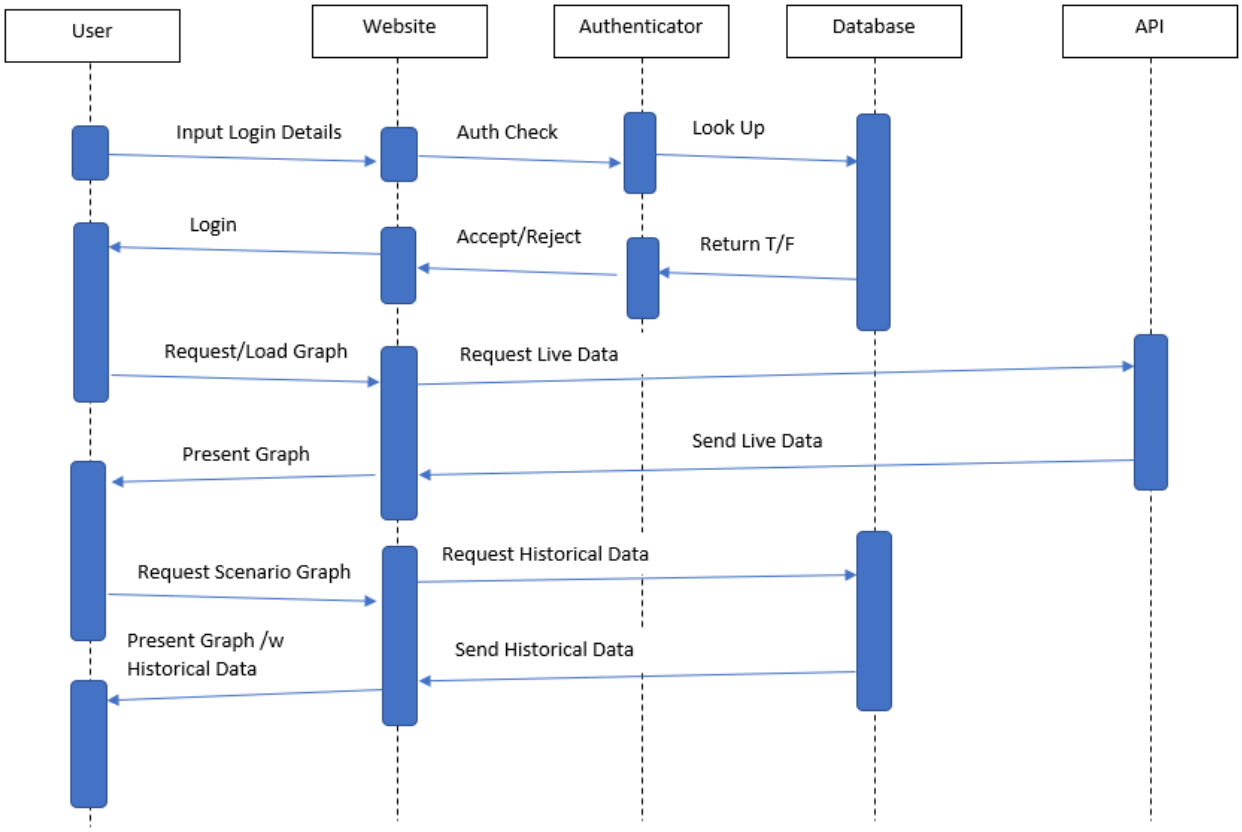
User Story - User navigates to website. User then finds the bitcoin chart/graph. User toggles through time groups (1 month, 3 months, 1 year) using corresponding buttons under graph. User navigates to the scenario section. User inputs the range of time desired for more focussed data in the text boxes and selects 'input' button. User views intended data.

Concept – A user will be able to produce results on a user determined set of variables (Start Date, End Date, Initial Investment, Increment Time, Increment Price). From this a graph with the corresponding data will be shown, along with tertiary data listed below.

Use Case Diagram –



Sequence Diagram –



Graph libraries which can present historical data:

Google charts supported with ORM

Graphos good but it's an app, need library ideally.

Pycha is a very simple Python package for drawing charts using the great Cairo library.

Chart.js seems the best at the moment for Django and graph data from api's can use json requests, all compatible.

Framework:

Django:
Django had been selected as the base framework for this project due to its compatibility with other software and required components for a website such as libraries for graphs, API usage for functionality with outside sources and growing popularity means increased support. Django's framework is a model-view-template framework using Python as its main programming language which is new to the development team as a programming language, however its appeal is that Python has a large array of libraries and content which can be ported to our website to increase functionality and implement it at a more efficient rate.

Angular:
Angular is another choice in framework for the website. It is one of the most popular frameworks that is developed by Google and is a front-end framework and is marketed at building 'rich single-page applications' using Typecast. It is also a particularly popular framework, meaning there should be (and appears to be from limited research) many resources for assistance in using Angular to make web pages and applications.

Graph libraries:

Chart.js – popular with many frameworks means there should be a lot of support available. It also works well with other software like bootstrap and supposedly can acquire data from API's efficiently. It also has a nice selection of graphs and has both line and candlestick available which are the main two graphs trading apps use.

Pycha - is a very simple Python package for drawing charts using the Cairo library, its simplicity helps in the integration with the rest of the system and seems to be the easier option for quick implementation. However, its simplicity is also its downfall with the package does not have a large amount of content, meaning that there are larger limitations to start with such as graph visuals and functionality.

Graphos – popular with Django framework which will help with implementation as there appears to be more content available to help create and develop the graphs. However, it's an app and the website ideally needs a library to import the content.

Google charts - supported with ORM (object relation mapping) which Django uses for retrieving data from a database. Google charts also has a large amount of graph types to use which may be beneficial in the latter stages of development if more sophisticated graphs are required or more functionality.

Other Graphs for trading:

Freetrade – Uses line graph, live and historical data all of which presented in one graph. The '1D' button presents live and historical data for that day, which appear to be updated on a 5-minute bases. The graph then has four more buttons '7D' showing the last seven days' worth of data, '1M' which shows the last month of data, '1Y' showing the last years' worth of data and 'MAX' which shows all the available historical data.

https://www.google.com/url?sa=i&source=images&cd=&ved=2ahUKEwity9mZhdPlAhXD6eAKHaqnCCAQjRx6BAgBEAQ&url=https%3A%2F%2Fwww.altfi.com%2Farticle%2F5392_freetrade-review&psig=AOvVaw1ocTb3S3yU0Kl-18hkWsLX&ust=1573042135957902

Additional material and libraries for Angular usage:

Typecast – is a sub version of Javascript and has most of the potential and abilities of regular Javascript.

Node.js – a Javascript runtime environment used for making Javascript programs and code usable outside of a typical browser. It will be required when using chart.js in order to test and run the Angular applications.

Bootstrap – is a front-end CSS framework containing Javascript based design elements. It will be utilised by the project to implement basic page design for graph testing and could be used for lesser aspects of the graphs such as buttons for timespan.

Charts.js + ng2-charts – is a library containing the possibility of creating graphs and charts by implementing said library instead of creating your own from scratch, utilising HTML5 canvas. It will be the main library that will dictate the look and functionality of the graphs which for trading purposes will likely be line and candlestick, both of which are included.

API's for live and historical bitcoin data:

Crytocompare – uses URL call for data, which may be able to put in a Json request. From preliminary investigations, Crytocompare appears to have a wider variety of functionality for its API meaning more data of different varieties and more specific calls for the data can be made. However, due to this it would seem that implementation will be harder to achieve for the Angular framework but can yield higher returns once implemented due to its larger array of functionality. This API will likely be the main/base API for getting live and historical data.

Alphavantage – uses Json requests to get the live and historical data from their API. From preliminary investigations, Alphavantage appears to be the easier API to use for the Angular framework and general implementations, though this is mainly due to this API having a smaller range of usability and therefore requires less effort to implement. This API will likely be a secondary/backup API for when the primary API isn't functioning correctly or is offline.

Data Storage:

Azure Table Storage – With Microsoft Azure being the platform that the website will be hosted, it is advantages to use its storage capabilities. Azure table storage is useful in that it can store sorted data, giving a very reliable and structure to data itself. Furthermore, this specified storage system will also enable a higher level of usage for required data in API calls and charts. However, the downside to this is that the mandatory rowKey table element must be unique and must be a string, meaning only data with a unique rowKey values can be stored and in addition, with the rowKey possibly being the only reliably unique element of data, using it in an API will be a little more problematic as it can only be a string which limits the range of functionality.