**Crypto Alert System**

Implementation

**Prepared by**

Marcus Nardizzi 100327114  
Josiah Tolentino 100304547  
Avery Gibson 100330734  
Ceston Carino 100330308  
Anmol Singh Bains 100332252

**2nd April, 2018**

**Table of Contents**

[**Introduction**](#_gjdgxs) **4**

[Purpose](#_30j0zll) 4

[System Overview](#_1fob9te) 4

[System Description](#_3znysh7) 4

[Assumptions and Constraints](#_2et92p0) 4

[System Organization](#_tyjcwt) 5

[Glossary](#_3dy6vkm) 5

[**Management Overview**](#_1t3h5sf) **5**

[Description of Implementation](#_4d34og8) 5

[Major Tasks](#_sv4wturwbvla) 6

[Create a base layer for the UI](#_5s45e4jdt8xi) 6

[Create a user control for the Help tab](#_bopmuy7im8xw) 6

[Create the user control for the Dashboard tab](#_8rjeiuqhy7as) 6

[Fetch the raw JSON Data from the API](#_gzk9wabvp582) 6

[Deserialize the raw JSON Data](#_ntscro87bhpw) 6

[Create the graph for the Dashboard tab](#_2gp20vxaox8a) 6

[Create the Historical data graph](#_azp7zb6409m7) 6

[Create the User Control for the Cryptocurrency tab](#_bzlqo28i4ml5) 7

[Create three objects of Cryptocurrency User Control](#_87cduzsevkt2) 7

[Let the User pick a cryptocurrency](#_wed6kgdf65ot) 7

[Create a graph to display the real-time data retrieved for each cryptocurrency](#_195iichvxrfc) 7

[Allow the user to enter values for each trading rule](#_utz05qghseje) 7

[Generate Alerts](#_caudgyqw92bs) 7

[Implementation Schedule](#_17dp8vu) 8

[Security and Privacy](#_3rdcrjn) 9

[System Security Features](#_lnxbz9) 9

[Security Set Up During Implementation](#_35nkun2) 9

[**Implementation Support**](#_1ksv4uv) **9**

[Hardware, Software, Facilities, and Materials](#_44sinio) 9

[Hardware](#_2jxsxqh) 9

[Software](#_z337ya) 10

[Materials](#_1y810tw) 10

[Documentation](#_umfsrlfvhwm8) 10

[Outstanding Issues](#_2xcytpi) 11

[3.4.1 The Notifications tab](#_3cpk0a38xw7p) 11

[3.4.2 Clearing Cache](#_q8wyjqxh80n1) 11

[3.4.3 Refresh historical data](#_p8pq7a4af1fy) 11

[3.4.4 Additional trading rules](#_rkrjn7rj9aow) 11

[Implementation Impact](#_1ci93xb) 11

[Performance Monitoring](#_3whwml4) 11

[Configuration Management Interface](#_8845ig215bfz) 12

[**Implementation Requirements by Site**](#_qsh70q) **12**

[Acceptance Criteria](#_3as4poj) 12

# 

# Introduction

## Purpose

The purpose of the Crypto Alert System is to give detailed information about the trends of cryptocurrencies. This allows users of the application who are interested in cryptocurrencies to make planning their investments easier and to evaluate specific currencies.

## System Overview

The goal of our project was to create an application which can be used by any and all who are interested in investing money into the cryptocurrency market. The purpose of our application is to make it easier for those people to invest into cryptocurrencies. The application will provide an easy to use platform for keeping up to date on the users ups and downs in cryptocurrency trading. The user will pick any three cryptocurrencies of their choice from a group of five top performing cryptocurrencies. The user will get a real time view on the activity of these cryptocurrencies in the form of an auto-updating graph. Moreover, the user will be able to set rules on each of the cryptocurrencies they have chosen. These rules will trigger an alert for the user. C# will be used as the main programming language, allowing us to create a system which could run on all windows platforms. We will also be making use of an online API which will provide the system with all the necessary data about the cryptocurrencies.

### System Description

Since our application works in real time by fetching data from an API, we will not host the data on the users device or need a database as the application will create an array keeping all the data that is has fetched from the API to create the graph. When the users enter the trading rules, the rules are only applied to incoming data. If one of the trading rules applies to the incoming data, it will notify the user on their device.

### Assumptions and Constraints

The constraints we encountered during developing our system was the programming language. Due to C#, the application will only be able to run on Windows. The system also only available on desktop, so the application always needs to be running in the background in order for it to perform its tasks.

It will be important to consider a decent budget for running this application. This application will mainly be used for long periods of time, requiring a high amount of resources and you will need a machine that is energy efficient while constantly producing power. You will want a large RAM space or a USB to record all of the data from your session as the application can only hold a small amount of the data and deletes the data after the program is shut down.

It will be extremely important to have a powerful and consistent internet connection as the application uses the internet connected to gather data from the API. The application can only run on a the latest .NET framework and only one user can be on the app at a time.

### System Organization

**Application and API**

The most important part of the program is the application and the API. The app will be run on a windows platform. The app will make recursive calls for the current data on each cryptocurrency chosen by the user. The application does not know the location, type or amount of data stored by the API. The application only makes request for the API for a specific data, where then the API will forward that specific data piece.

**User Interface**

The User interface is essential to the application as we would not be able to display the graphs for the user. The interface is also where all the trading rules are entered by the user. The user interface has five tabs. One representing the historical data graphs of the currency, where the user can select the cryptocurrencies of their choice. The others will represent the specific cryptocurrency graphs chosen by the user. Each graph page will have updates on the trading rules.

## Glossary

API Application programming interface

User Someone who uses the ‘Crypto Alert System’

Cryptocurrency A digital currency

RAM Random Access Memory

# Management Overview

## Description of Implementation

The program will be downloadable and will require no installation. The program utilizes an instant-on approach. All the user has to do is:

1. Download the CryptoAlertSystem.zip file
2. Extract both the files in the .zip into a new folder
3. Run the .exe file

## Major Tasks

### Create a base layer for the UI

Form1.cs acts as the base layer for the application. It is the container for the different tabs of the application which represent different UserControls.

### Create a user control for the Help tab

HelpControl.cs is the very first tab the users will see once they start the application. This tab will display a short summary of the purpose of the application, and will act as a quick how-to guide for the user to use the application

### Create the user control for the Dashboard tab

Dashboard.cs is the tab that will act as the home screen for the application. It contains a graphical representation of how all 5 cryptocurrencies have been doing over the past 30 days.

### Fetch the raw JSON Data from the API

The application uses a WebClient object to fetch the raw JSON data from a particular URL.

### Deserialize the raw JSON Data

JSONStructure.cs consists of all the classes required to gather the deserialized data. The Newtonsoft.Json framework consists of classes that can convert raw JSON data, that you get from the API, into more usable data types. This process is called deserialization. The deserialized data is ‘put’ into classes that represent how the data was organized in it’s raw JSON form. These classes allow the application to access the data.

### Create the graph for the Dashboard tab

The graph showing the historical data for all 5 cryptocurrencies is be loaded as soon as the application is started so that the user does not have to wait until after clicking on the dashboard tab. In order to make this happen, all the functions required to fetch and display the data are performed inside the constructor for the Dashboard.cs user control.

### Create the Historical data graph

The data that has been deserialized and ‘put’ into classes, can now be accessed, through a loop, and turned into points on a graph. These points are then connected to form a line graph. This process is done for each cryptocurrency.

### Create the User Control for the Cryptocurrency tab

CryptoCurrencyControl.cs is the user control for each cryptocurrency. It consists of a graph showing the real time data for each cryptocurrency selected by the user. It also allows the user to select which cryptocurrency they want to view.

### Create three objects of Cryptocurrency User Control

Three different instances are created for the CryptoCurrencyControl.cs User Control. Each instance represents one of the 3 different tabs for the Cryptocurrencies. This allows the user to simultaneously run three different cryptocurrencies in three different tabs. Each cryptocurrency having its own unique graph, and unique values for trading rules.

### Let the User pick a cryptocurrency

The user is allowed to pick one cryptocurrency (out of 5) per Cryptocurrency tab. Each currency has its own unique URL that is used to retrieve data from the API.

### Create a graph to display the real-time data retrieved for each cryptocurrency

The *‘getRLTData()’*  method makes recurring calls to the API, using the URL of the currency picked by the user. This data is then plotted (in real-time), on a graph in each Cryptocurrency tab.

### Allow the user to enter values for each trading rule

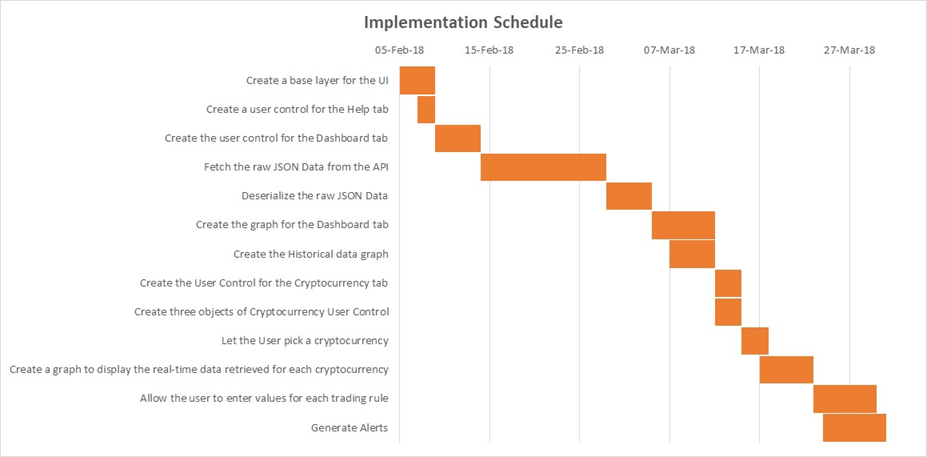
There are two predefined trading rules for each cryptocurrency, which accept a user’s input. This input is used as a trading rule, and for generating alerts.

### Generate Alerts

Alerts pop-up on the user’s screen whenever a trading rule is met, for any cryptocurrency.

## Implementation Schedule

|  |  |  |
| --- | --- | --- |
| **Task** | **Start Date** | **Due Date** |
| Create a base layer for the UI | 05-Feb-18 | 09-Feb-18 |
| Create a user control for the Help tab | 07-Feb-18 | 09-Feb-18 |
| Create the user control for the Dashboard tab | 09-Feb-18 | 14-Feb-18 |
| Fetch the raw JSON Data from the API | 14-Feb-18 | 28-Feb-18 |
| Deserialize the raw JSON Data | 28-Feb-18 | 05-Mar-18 |
| Create the graph for the Dashboard tab | 05-Mar-18 | 12-Mar-18 |
| Create the Historical data graph | 07-Mar-18 | 12-Mar-18 |
| Create the User Control for the Cryptocurrency tab | 12-Mar-18 | 15-Mar-18 |
| Create three objects of Cryptocurrency User Control | 12-Mar-18 | 15-Mar-18 |
| Let the User pick a cryptocurrency | 15-Mar-18 | 18-Mar-18 |
| Create a graph to display the real-time data retrieved for each cryptocurrency | 17-Mar-18 | 23-Mar-18 |
| Allow the user to enter values for each trading rule | 23-Mar-18 | 30-Mar-18 |
| Generate Alerts | 24-Mar-18 | 31-Mar-18 |



## Security and Privacy

### System Security Features

Since this application does not require any personal user data, there are not any major security impacts due to it. The application generates requests for an API, every 10 seconds, to retrieve raw JSON data. Even though the requests are made to a public server, they are secure, as the data retrieved using the http-s protocol. Also, since the data is only handled by the application itself, and not stored in its raw form, i.e. it’s deserialized first, it is not as much of a security hazard.

### Security Set Up During Implementation

Since the application makes constant requests to the API, the user’s firewall must be configured to allow these requests to go through. Also, if the user is using an antivirus, it may block the program, and mark it as malicious, as it does not have a certification.

# Implementation Support

## Hardware, Software, Facilities, and Materials

### Hardware

Hardware we used for testing were windows computers with the capability to run the software needed for the application.

### Software

This application is supported only on machines running Windows 7 and later or Windows Server 2012 and later and have the .NET runtime framework version 3.5 or later installed.

**Windows** is a group of several graphical operating system families, all of which are developed, marketed, and sold by Microsoft. This application can be run on Windows NT and Windows Embedded families. The operating systems are licensed and paid. They also have certain hardware requirements.

The .NET framework is a software framework developed by Microsoft, primarily for Windows OS. It provides a controlled environment for the execution, installation and development on Windows OS. The .NET framework is licensed, but is available for free online, and comes pre-installed with Windows OS.

The application in particular makes use of Json.Net framework. It is an open-source, third-party framework for the serialization and deserialization of Json data. Being an open-source framework, it is free, and can be packaged as a .dll along with the application deployment.

### Materials

No other materials are required to run this application

## Documentation

The program itself is fairly simple to use, with a minimal UI, and descriptive text. The application does contain a help menu that will provide the user with basic instructions on how to use the program, and the basic workings of the program.

In order to start the application:

1. Download the CryptoAlertSystem.zip file
2. Extract both the files in the .zip into a new folder
3. Run the .exe file

Inside the application:

1. Dashboard: Displays a graph of all crypto currencies
2. Crypto Currency 1,2,3: You can set alerts up to 3 crypto currencies

Step 1: Select a Crypto Currency tab

Step 2: Click Start to go on with the default Cryptocurrency

OR

Step 2: Pick a currency from the choices provided

Step 3: Enter a dollar value for the alert fields

Click Refresh to reload the current cryptocurrency

1. Alerts:

Rises above: When the $ value rises above the amount entered in the field, an alert will display

Falls below: When the $ value falls below the amount entered in the field, an alert will display

## Outstanding Issues

### 3.4.1 The Notifications tab

We are planning on implementing the Notifications tab in future builds of the application. The notifications tab will show a list of all the alerts that have already been generated. The user can use this list as a reference if needed.

### 3.4.2 Clearing Cache

We would like to create a provision for the application to automatically empty it’s cache every 12 hours. This way the program will not be using too many resources on the user’s machine.

### 3.4.3 Refresh historical data

Would also like to refresh the historical data graph every 24 hours. The way the program works right now is that the historical data graph in the Dashboard tab is generated when the user runs the application. If the application is left running for a long time, the graph will not update itself.

### 3.4.4 Additional trading rules

We would also like to add more trading rules to the next version of the application.

## Implementation Impact

The application itself does not take too much space on storage or use too much bandwidth from the internet. Although it may impact the RAM, if it is left running for more than 12 hours.

## Performance Monitoring

We can monitor the CPU usage, RAM usage, and bandwidth usage via Task Manager on Windows.

# Implementation Requirements by Site

The implementation for the application is not site specific. As long as the user meets the software and hardware requirements, the program can be run as a .exe file.

## Acceptance Criteria

The user can select up to 3 different crypto-currencies and set alerts for each selection. The alerts the user can set are when the amount is lesser or greater than the set amount. The application will show a visual graph for the selected crypto-currency, which will update frequently. On every update, an alert should should if it has met the criteria set by the user.