U Mobile Cryptocurrency Prediction Software

Software Design and Research Report

Code Black

Name	Position	email	phone
Bernard Joshua	Team Leader/ Data Engineer	*103365867@student.swin.edu.au*	016-331 7910
Lionel Low	Backend Engineer	*103235180@student.swin.edu.au*	013-484 1402
Danial Imran	Frontend Engineer	*103701416@student.swin.edu.au*	012-644 5100
Ming Xuan	Frontend Engineer	*103701377@student.swin.edu.au*	012-360 9691

SWE40001-Software Engineering Project A, 4th Semester 20th June 2022

1   DOCUMENT CHANGE CONTROL

Version	Date	Authors	Summary of Changes
1	7/6/2022	Danial Imran	Introduction
1	8/6/2022	Ming Xuan	Problem Analysis
1	9/6/2022	Bernard Joshua	High-Level Architecture
1	10/6/2022	Lionel Low	Object-Oriented Design
2	15/6/2022	Lionel Low	Object-Oriented Design (Revised)
3	20/6/2022	Bernard Joshua	Final Proof-Reading and Editing

DOCUMENT SIGN OFF

Name	Position	Signature	Date
Bernard Joshua	*Team Leader/Data Engineer*	*Bernard*	**20th June 2022**
Lionel Low	*Backend Engineer*		**20th June 2022**
Danial Imran	*Frontend Engineer*	*Danial Imran*	**20th June 2022**
Ming Xuan	*Frontend Engineer*	*MingXuan*	**20th June 2022**

CLIENT SIGN OFF

Name	Position	Signature	Date
Chew Yew Choong	Head of Data Science & CVM		25/06/2022
Organisation			
U Mobile Sdn Bhd			

1. Introduction

This document is designed to provide a high-level insight to the design of a prediction software that uses a combination of historical data (structured) and real-time data (unstructured) to make predictions on various types of cryptocurrencies. It will aid the employer and the developers that work for the employer to better understand the system and make future improvements or for debugging purposes . It involves research into the type of system architecture that would be used. An important thing to note here is that this document will only cover the system that makes the prediction and not the interface. The interface just follows a generic structure and is not modelled after any architecture as it is only used to display the predictions.

1.1 Overview

This document will cover an analysis of the design problem by summarizing the high-level system goals and objectives, the assumptions made in designing the architecture, the high-level architecture that was chosen for the system, the alternative architectures that where explored and why they were rejected as well as an object-oriented design for the system. The document will clearly identify the components that need to be developed and how they relate to each other as well as the flow of the system on a high-level. It will provide diagrams and charts to visualize this and also a relevant description of each of these components.

1.2 Definitions, Acronyms and Abbreviations

SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service

2. Problem Analysis

This section provides a high-level analysis of the Software/System Requirements (SRS) of the target software system from the viewpoint of developing a design solution for it.

2.1. System Goals and Objectives

a. *Software is to be resilient.*
- *Doing so would increase the user base of the software as users know that the system is resilient and would not crash easily. This is because it deals with predicting cryptocurrencies which are time-based assets. If the system crashes at a crucial time the users may lose money and giving the existing user base, better access to crypto investment services.*

b. *Software is to be portable and scalable.*
- *This will help the employer reduce costs in maintaining and improving the software later.*

2.2. Assumptions

- *Software can be completed on time.*
    - o *The assumption is that the design is not too complex and since the team has spent an extensive amount of time researching Azure and how to develop a prediction software using it. The team assumes that the software can be completed on time.*

- *Software changes in one part will have little to no impact on the others.*
    - o *Since the software is not too complex and there is not a lot of dependencies among the components, the team believes that the future improvements of the software by the employer's team will not affect the overall system/software too much.*

- *The system design manages to encapsulate all the employer's needs.*
    - o *There are many different types of designs available for cloud-based applications. However due to time constraints only three were explored and only one was chosen. We believe that the chosen design is the best one that manages to meet the needs the employer has.*

2.3. Simplifications (if any)

The design of the software was already simple to begin with hence no further simplifications were made.

3. High–Level System Architecture and Alternatives

This section is about the high-level architecture design which will be described, using appropriate models and notations, the chosen <u>high-level</u> architecture of the software system that will be developed. This section will also discuss two additional architecture alternatives that have been explored but not chosen

3.1. System Architecture

The chosen system architecture is a called Big-Data architecture. It can be used for any type of cloud-based application SaaS, PaaS, or IaaS. A Big-Data architecture is intended to handle data ingestion, processing, and analysis that is too large or complex for traditional database systems. This is a crucial point as prediction and analytics software's work with massive amounts of data. It involves all the following types of workloads to be done:

- *Batch processing of big data sources at rest.*
- *Real-time processing of big data in motion.*
- *Interactive exploration of big data.*
- *Predictive analytics and machine learning.*

All these things make Big-Data architecture superior for development over other architectural designs as Big-Data architecture was developed for the sole purpose of prediction and analytics type of software or systems. Hence, this is the main reason it was selected to be used for the architecture of our software. The other reasons it was selected is summarized below:

Reason	Description
Multiple Technology choices	*In HDInsight clusters **(Big-Data Component)**, you can mix and match Azure managed services and Apache technologies to capitalise on existing skills or technology investments. Using this will*

	enable integration of Microsoft AI tools which can aid in development of the software. In turn this will reduce development time.
High Performance	*Parallelism is used by big data solutions to enable high-performance solutions that scale to large volumes of data. High performing software/system would make it more reliable and highly accurate in predictions.*
Flexible Scalability and Manageable Costs	*All components of the big data architecture support scale-out provisioning, allowing you to tailor your solution to small or large workloads while only paying for the resources you use. This, helps the employer to save costs.*
High Interoperability	*Big data architecture components are also used in enterprise BI solutions, allowing you to create an integrated solution across data workloads. Make it more portable.*

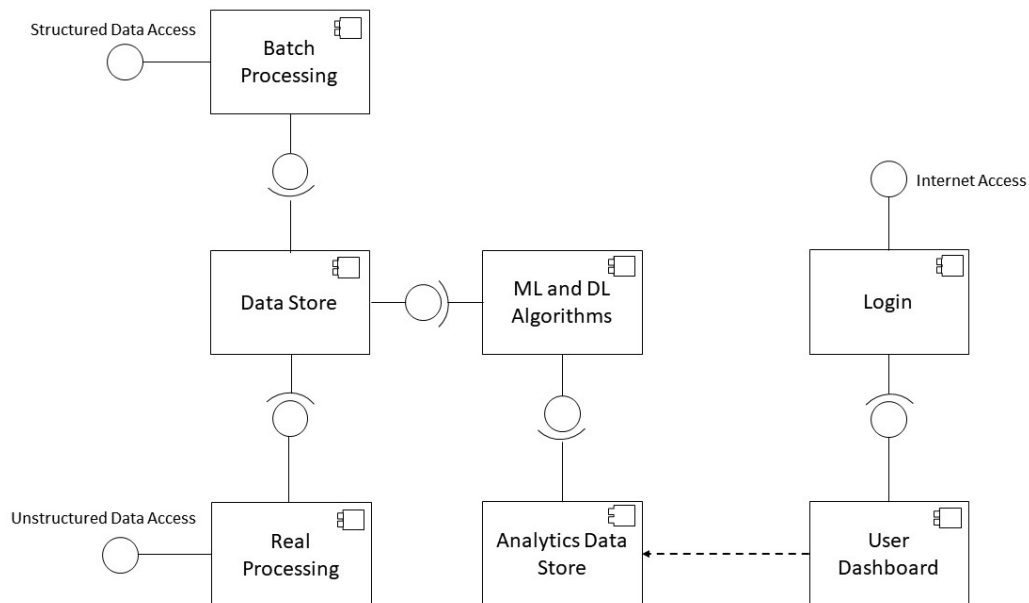*The diagram below shows how each component of the architecture relates to one another:*

Figure 1: High-Level Design Diagram: Showing relationships between components.

The table below summarizes this in text form as well as provide descriptions for the components:

Component	Description	Dependant (Relationship)
Batch Processing	*Big data solutions will use long-running batch jobs to filter, aggregate, and otherwise prepare data files (structured data) for analysis. These jobs typically entail reading source files, processing them, and writing the output to new files. Running U-SQL jobs in Azure Data Lake Analytics is one option to implement this.*	*Structured Data*
Real Processing	*After ingesting real-time data which is usually unstructured, the solution will filter, aggregate, and otherwise prepare the data for analysis. After that, the processed stream data is written to an output sink.*	*Unstructured Data*
Data Store	*Data for batch and real-time processing operations is typically stored in a distributed file store capable of storing large volumes of large files in a variety of formats. This type of storage is commonly referred to as a data lake.* **Azure Data Lake Store** *is an option for implementing this storage.*	*Batch and Real Processing Components*
Machine and Deep Learning Algorithms	*Machine and Deep Learning algorithms will be used to analyse the data to get insights and make predictions. This can be coded manually or can use Apache Spark to do so.*	*Data Store Component*
Analytics Data Store	*Store and serve processed data in a structured format that can be queried using analytical tools.*	*Machine and Deep Learning Algorithms Component*
User Dashboard	*Presents the Analytics and Reporting.*	*Login*
Login	*Provides Access to the User's dashboard.*	*Internet Access*

The High-Level diagram below visualizes the software's dataflow following the Ingest, Store, Prep and Model method, which is the Microsoft recommended method to carry out predictions:
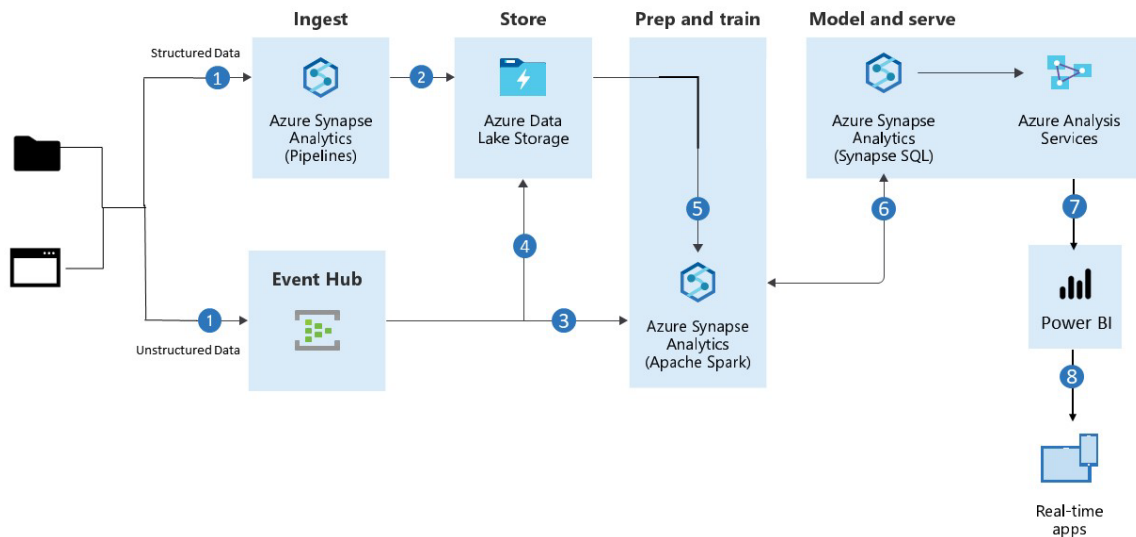

Figure 2: High-Level Design Diagram: Showing Flow of Software/System.

Data Flow

Reference	Description
1	*Ingests structured data through Azure Synapse Analytics (Pipelines) and unstructured data through Azure Events Hub. Azure Synapse Analytics (Pipelines) also carries out batch processing.*
2	*Store the structured data in the Azure Data Lake.*
3	*Carry out real-processing through Apache Spark pools to clean and transform the real-time data.*
4	*Combine the processed real-time data with the batch data in the Azure Data Lake.*
5	*Retrieve the combined data from the Azure Data Lake and analyse the data to get insights from it using ML/DL algorithms in Apache Spark notebook.*
6	*Send the processed data (insights) to the Synapse SQL to store. This can also be retrieved by Apache Spark later to use as data of previous insights for more accurate predictions and be sent back to the Synapse SQL to store as a new prediction entry. Azure Analytics service uses this data and builds models.*
7	*Azure Analytics Services uses Power BI to build visualizations and reports that can be deployed.*

| 8	The report generated will then be embedded into the Web Application hence the user is able to see it. Each time the software makes a prediction it will be reflected on the user's dashboard.

3.2. Other Alternative Architectures Explored

Among the architecture designs that have been explored together with the Big Data architecture is the N-Tier and Web-Queue worker architectural designs. Both designs are cloud-architectural style designs. N-Tie is used for IaaS applications while Web-Queue worker is used for PaaS applications. Which each meet a specific criterion our team was looking for in developing our software. Following is a summary of the reasons why the team explored these architectural types:

N-Tier	Web-Queue Worker
Portability between cloud and on-premises.	*Relatively simple architecture that is easy to understand.*
Portability between cloud platforms.	*Easy to deploy and manage.*
Less learning curve for most developers.	*Clear separation of concerns.*
Openness to heterogeneous environment (Windows/Linux)	*Decoupled frontend from the worker using asynchronous messaging and the frontend and the worker can be scaled independently.*

<u>Justification of Rejection of Alternative Architectures</u>

These were the reasons they were rejected. Firstly, both do not support real-time data processing. This is a crucial part of the software as; cryptocurrency is time-based and hence needs to be monitored and reported on constantly. The Big-Data architecture does not have this problem as it supports both static and real-time data processing. The next pitfall is that N-Tier and Web-Queue worker has inferior performance when compared to Big-Data architecture. As the N-Tier has multiple tiers it increases the latency of the software which in turn would reduce performance while the Web-Queue worker uses a Queue-based architecture which works consecutively not concurrently which leads to the same latency-based issue as the N-Tier architecture. The Big-Data architecture does not have this problem as it takes advantage of parallelism, enabling high-performance through scaling large volumes of data concurrently. Another key issue with N-Tier and Web-Queue architecture is that they both have security issues. N-Tier has network security issues in a large system and our software would be expected to work in such a system as there is a lot of data to be processed as well as a large user base since the employer has a large user base too, while the Web-Queue worker may have a lot of hidden dependencies if the front end and worker components share data schemas or code modules which may make data abstraction difficult leading to it being vulnerable to XSS attacks or any other web-based attacks to steal or manipulate data. Big Data Architecture though has its data security problems, it is minor when compared to the alternative architectures and can be easily contained through good coding practices. Finally, N-Tier and Web-Queue worker does not support the integration of other technologies into their architecture. These technologies are namely Microsoft Azure's inbuilt AI frameworks which would be needed in developing the prediction software. Big-Data architecture does not have this problem as it was designed to work with other AI tools/frameworks as this architecture is mainly used for data science and AI-related software.

4. Object Oriented Design (or alternative)
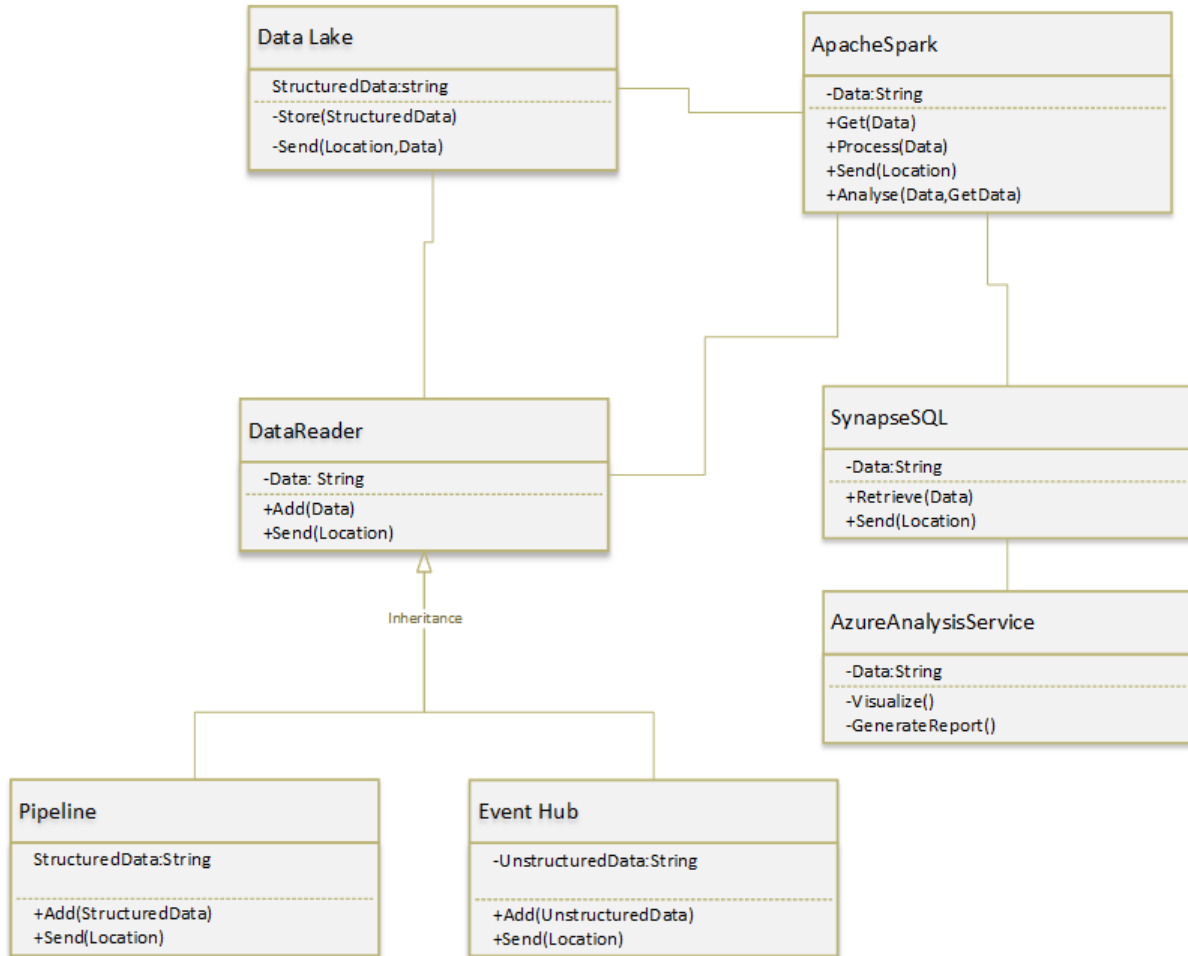
4.1. Object Oriented Design and Justification

Component	Justification
DataReader	For reading and sending data, parent class for pipeline and event hub
Pipeline	For reading and sending Structured Data
Event Hub	For reading and sending Unstructured Data
Data Lake	Stores Data from Pipeline or Event Hub and sends data to where its needed
ApacheSpark	Can process Data from Pipeline or EventHub, Analyse data from Pipeline or EventHub and Data Lake, and Send to Synapse SQL
SynapseSQL	Get Data from ApacheSpark and Send to Aure Analytic Service
Azure Analytic Service	Visualize Data and generate Report using

	data

4.2. Design Verification

Analyse Data and Generate Visual and Graph
- *Data will come in from either Pipeline or Event Hub depending on the type of data (structured/unstructured)*
- *Pipeline/Event Hub sends real-time the data to ApacheSpark for processing then sends the cleaned and transformed data to the Data Lake.*
- *Data Lake will send the data to ApacheSpark so it can perform analysis on processed data.*
- *ApacheSpark sends the analysed data to SynapseSQL to store*
- *SynapseSQL sends data back to ApacheSpark if needed or Azure Analytic Service if no more processing is required.*
- *Azure Analytic Service uses the data to create a visual graph through Power BI and generate a report.*

5. References

Architecture styles - Azure Application Architecture Guide, Microsoft.com, Microsoft, viewed 10 June 2022, < https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/>.

Big data architecture style - Azure Application Architecture Guide, docs.microsoft.com, Microsoft, viewed 10 June 2022, < https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/big-data>.

N-tier architecture style - Azure Application Architecture Guide 2022, docs.microsoft.com, Microsoft, viewed 10 June 2022, < https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier>.

Real-time analytics on big data architecture - Azure Solution Ideas 2022, docs.microsoft.com, Microsoft, viewed 10 June 2022, <https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/real-time-analytics>.

Web-Queue-Worker architecture style - Azure Application Architecture Guide 2022, docs.microsoft.com, Microsoft, viewed 10 June 2022, < https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/web-queue-worker>.

What is Azure Analysis Services 2022, docs.microsoft.com, Microsoft viewed 10 June 2022, < https://docs.microsoft.com/en-us/azure/analysis-services/analysis-services-overview>.