SE 3XA3: Software Requirements Specification CryptoMetrics

Team 15
Saif Fadhel, fadhels
Vanshaj Verma, vermav2
Himanshu Aggarwal, aggarwah

March 18, 2022

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
Mar. 14, 2022	1.0	Creation of MG Doc
Mar. 15, 2022	1.1	Add Introduction
Mar. 17, 2022	1.1	Complete Majority of sections
Mar. 18, 2022	1.1	All sections complete

1 Introduction

1.1 Overview

CryptoMetrics is a re-implementation of an open-source project called Cryptodash which analyzes historical trends related to various cryptocurrencies in order to help users make educated investment decisions.

1.2 Context

The context of this document is that it will build upon what the Software Requirements Specification Document started which is that in addition to identifying the functional and non-functional requirements proposed by the SRS, the Design document will elaborate on how these requirements can be achieved. Once completed, this document will be sent to different groups to get them familiar with the design of the project. These groups can be identified as follows:

- **New project members:** This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- **Maintainers:** The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- **Designers:** Once the module guide has been written, it can be used to check for consistency, feasibility and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

This document will assist in the creation of the Module Interface Specification (MIS) which is a detailed outline for all of the modules the are a part of CryptoMetrics. This includes all their state/environment variables, exported access programs, and their access program semantics.

1.3 Design Principles

The design principles that are used by the system includes:

- **Separation of Concerns:** Involves separating an application into distinct sections, so each section addresses a separate concern. Each module has its own assigned role that is distinct from every other module which is why it follows this design principle.

- **Low Coupling:** Coupling determines how closely related modules are to one another. Low Coupling is ideal since it allows for more modular code design. The implementation of modules follows low coupling since each of the modules are independent of one another by performing functionality required for a specific task only within a module assigned to that task and no other modules.
- **High Cohesion:** Cohesion determines how closely related elements inside a module are with one another. High cohesion is desirable since it allows the modules to be more easily understandable to anyone unfamiliar with the implementation. The implementation of modules follows this design principle of high cohesion since each component module contains methods and data closely related to one another.
- **Encapsulation:** Is a principle in object-oriented programming which refers to the packaging of closely related data and functions into classes or objects. It is followed since each component module can be viewed as an object of closely related data that can be reused as many times as needed.

1.4 Document Structure

The rest of the document is organized as follows. Section 2 lists the anticipated and unlikely changes of the software requirements. Section 3 summarizes the module decomposition that was constructed according to the likely changes. Section 4 specifies the connections between the software requirements and the modules. Section 5 gives a detailed description of the modules. Section 6 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 7 describes the use relation between modules.

2 Anticipated and Unlikely Changes

2.1 Anticipated Changes

AC1: The user able to compare multiple cryptocurrency's graphs stacked together.

AC2: Clicking on the cryptocurrency card to get more information about the coin.

AC3: Web application giving a pop up to the user when is unable to pull data through the API.

2.2 Unlikely Changes

UC1: The web application able to run on different screen sizes with different aspect ratios.

UC2: User able to shift and move cryptocurrency and its associated pricing information around the web application.

UC3: User able to have their preferences saved and loaded back up upon launching the web application.

3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 2. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

Level 1	**Level 2**
Hardware-Hiding Module	N/A
Behaviour-Hiding Module	Button Module (M5.2.1) ToggleButton Module (M5.2.2) FilterButton Module (M5.2.3) CryptoChartCard Module (M5.2.4) CompareChart Module (M5.2.5) CryptoRowLineChart Module (M5.2.6) Container Module (M5.2.7) Main Module (M5.2.8) Wrapper Module (M5.2.9) Dropdown Module (M5.2.10) DropdownItem Module (M5.2.11) FilterDropdown Module (M5.2.12) SecondaryFilterDropdown Module (M5.2.13) Filter Module (M5.2.14) Filters Module (M5.2.15) Input Module (M5.2.16) SearchInput Module (M5.2.17) RadioInputForm Module (M5.2.18) Radio Module (M5.2.19) PlaceholderSkeleton Module (M5.2.20) Sidebar Module (M5.2.21) SidebarItem Module (M5.2.22) Table Module (M5.2.23) TableRow Module (M5.2.24) TableHeader Module (M5.2.25) TableCell Module (M5.2.26) Tab Module (M5.2.27) Tabs Module (M5.2.28) BoldGradientHeading Module (M5.2.29) Home Module (M5.2.30) Compare Module(M5.2.31) Hooks Module(M5.2.32) Queries Module(M5.2.33)
Software Decision Module	N/A

Table 2: Module Hierarchy

4 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 3.

The system's requirements essentially require that the system allow users to view all cryptocurrency price data within a given time frame in multiple views, and enable the ability to compare them with one another. Each of the modules have met all of the requirements specified. Table Module along with its assistants TableHeader Module, TableRow Module, and TableCell Module all work together to provide a tabular view of all of the cryptocurrencies in the home page of the web application. ToggleButton Module and CompareChart Module are responsible for displaying historical data and allowing users to compare between two different cryptocurrencies price data.

5 Module Decomposition

5.1 Hardware Hiding Module

Secrets: The data structure and algorithm used to implement the virtual hardware.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

Implemented By: OS

5.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours.

Services: Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: ReactJS

5.2.1 Button Module

Secrets: Button styles and features.

Services: Adds a styled HTML button to the page with `onClick` events.

Implemented By: ReactJS

5.2.2 ToggleButton Module

Secrets: Toggle Button styles and features.

Services: Wraps the Button component such that it can be toggled on/off by clicking on it.

Implemented By: ReactJS

5.2.3 FilterButton Module

Secrets: Filter Button styles.

Services: Wraps the Button component such that it can be used to open the FiltersDropdown component.

Implemented By: ReactJS

5.2.4 CryptoChartCard Module

Secrets: The format and styling of crypto charts.

Services: Renders card with cryptocurrency information and an embedded chart.

Implemented By: ReactJS

5.2.5 CompareChart Module

Secrets: Data and logic for displaying multiple charts.

Services: Renders multiple charts for comparison and displays options to configure specific settings for the charts.

Implemented By: ReactJS

5.2.6 CryptoRowLineChart Module

Secrets: The format and styling of crypto charts.

Services: Renders a chart inline with the other components

Implemented By: ReactJS.

5.2.7 Container Module

Secrets: Properties for enclosing other components.

Services: Wraps a set of components with certain properties.

Implemented By: ReactJS

5.2.8 Main Module

Secrets: Properties for main portion of the page.

Services: Wraps the main portion of the page with certain properties (this portion excludes the sidebar)

Implemented By: ReactJS

5.2.9 Wrapper Module

Secrets: Properties for wrapping components.

Services: Wraps the entire page and provides them with certain properties.

Implemented By: ReactJS

5.2.10 Dropdown Module

Secrets: Properties for displaying a list of options.

Services: Renders a dropdown with a list of options.

Implemented By: ReactJS

5.2.11 DropdownItem Module

Secrets: Styles for displaying a dropdown item.

Services: Renders an item inside a dropdown.

Implemented By: ReactJS

5.2.12 FilterDropdown Module

Secrets: Properties for displaying a set of filters.

Services: Renders a multi-level dropdown with a list of filters.

Implemented By: ReactJS

5.2.13 SecondaryFilterDropdown Module

Secrets: Advanced properties for selecting a set of filters.

Services: Renders a secondary-level dropdown view with advanced options for a filter.

Implemented By: ReactJS

5.2.14 Filter Module

Secrets: Filter data and properties.

Services: Renders a tag with filter data and a button to remove the filter.

Implemented By: ReactJS

5.2.15 Filters Module

Secrets: Properties for encapsulating multiple Filter Modules.

Services: Takes as input several Filter components and renders them in a horizontal fashion.

Implemented By: ReactJS

5.2.16 Input Module

Secrets: Styles and properties for input fields.

Services: Renders an HTML input field and provides advanced customization.

Implemented By: ReactJS

5.2.17 SearchInput Module

Secrets: Styles and properties for search input field.

Services: Wraps the Input component to add a search icon.

Implemented By: ReactJS

5.2.18 RadioInputForm Module

Secrets: Form properties.

Services: Renders a form with radio elements, each with an input field.

Implemented By: ReactJS

5.2.19 Radio Module

Secrets: Properties and styles for radio.

Services: Renders radio elements.

Implemented By: ReactJS

5.2.20 PlaceholderSkeleton Module

Secrets: Styles for skeleton.

Services: Renders a skeleton component when data is loading.

Implemented By: ReactJS

5.2.21 Sidebar Module

Secrets: Sidebar options and properties.

Services: Renders a sidebar to the page with navigation links to other pages.

Implemented By: ReactJS

5.2.22 SidebarItem Module

Secrets: Styles for the sidebar item.

Services: Renders a sidebar item with an icon.

Implemented By: ReactJS

5.2.23 Table Module

Secrets: Properties for enclosing multiple table rows.

Services: Wraps a set of table rows together with specific properties.

Implemented By: ReactJS

5.2.24 TableRow Module

Secrets: Properties for enclosing multiple table cells.

Services: Wraps a set of table cells together with specific properties.

Implemented By: ReactJS

5.2.25 TableHeader Module

Secrets: Properties for enclosing multiple table cells for the header.

Services: Wraps a set of table cells together with specific properties for the header.

Implemented By: ReactJS

5.2.26 TableCell Module

Secrets: Styles and properties of a table cell.

Services: Renders a table cell with specific properties.

Implemented By: ReactJS

5.2.27 Tab Module

Secrets: Styles and properties.

Services: Acts as a button to trigger switching components.

Implemented By: ReactJS

5.2.28 Tabs Module

Secrets: Data and logic for switching components.

Services: Wraps multiple Tab components and conditionally renders different components.

Implemented By: ReactJS

5.2.29 BoldGradientHeading Module

Secrets: Styles for the heading.

Services: Renders a gradient header.

Implemented By: ReactJS

5.2.30 Home Module

Secrets: Data and logic to control flow between different components.

Services: Renders the home page with all components.

Implemented By: ReactJS

5.2.31 Compare Module

Secrets: Properties for handling different components.

Services: Renders the compare page with the required components.

Implemented By: ReactJS

5.2.32 Hooks Module

Secrets: Logic and data.

Services: Help to hook-into state and lifecycle methods of different components.

Implemented By: ReactJS

5.2.33 Queries Module

Secrets: APIs and their properties.

Services: Provides Hooks to fetch data from APIs.

Implemented By: ReactJS

5.3 Software Decision Module

N/A

6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	**Modules**
FR-1	M5.2.23, M5.2.25, M5.2.24, M5.2.26
FR-2	M5.2.17, M5.2.16
FR-3	M5.2.5, M5.2.2
FR-4	M5.2.5, M5.2.10, M5.2.11
FR-5	M5.2.3, M5.2.12, M5.2.13 M5.2.14, M5.2.15, M5.2.18, M5.2.19, M5.2.1
FR-6	Pending
FR-7	M5.2.33
FR-8	M5.2.33
FR-9	M5.2.33
FR-10	M5.2.20
FR-11	M5.2.33
FR-12	M5.2.21, M5.2.22
FR-13	Pending
NFR-1	M5.2.9, M5.2.7, M5.2.8, M5.2.21, M5.2.22
NFR-2	M5.2.30, M5.2.31
NFR-3	M5.2.27, M5.2.28
NFR-4	M5.2.29

Table 3: Trace Between Requirements and Modules

AC	**Modules**
AC1	M5.2.31, M5.2.5
AC2	M5.2.30
AC3	M5.2.30, M5.2.31, M5.2.33

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. **?**) said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

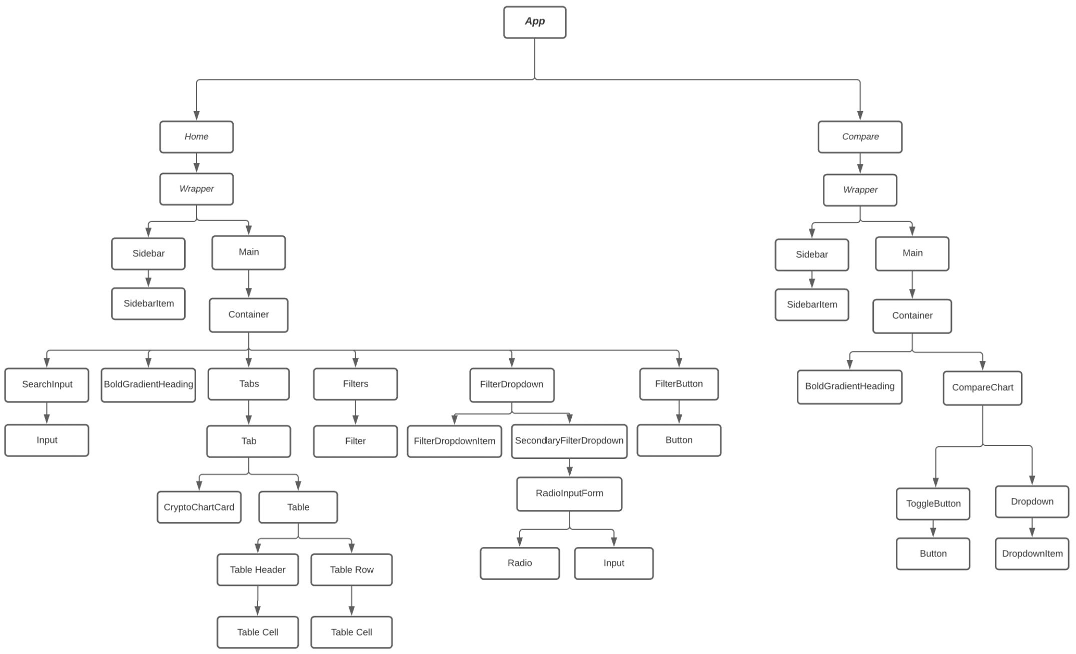

Figure 1: Use hierarchy among modules

References

[1] David L. Parnas and P.C. Clements. A rational design process: How and why to fake it. *IEEE Transactions on Software Engineering*, 12(2):251–257, February 1986.

[2] James Robertson and Suzanne Robertson. *Volere Requirements Specification Template.* Atlantic Systems Guild Limited, 16 edition, 2012.