SE 3XA3: Test Plan
CryptoMetrics

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

April 12, 2022

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
Mar 10, 2022	1.0	Creation of the test plan document
Mar 11, 2022	1.1	Update and finalize the test plan
Apr 10, 2022	1.2	Update how tests would be performed

1 General Information

1.1 Purpose

The purpose of this document is to outline the development team's test plan for the proposed project, CryptoMetrics as per the verification and validation section of the software development cycle. The details pertaining to the test plan includes the automated testing approach, the tools that will be used to conduct tests, and the schedule prioritizing the tests that must take precedence over others.

1.2 Scope

The scope of testing includes the tests that will be conducted to verify that the functional and non-functional requirements have been met, the tests prescribed for the Proof of Concept, and the overall Unit-Testing Plan. Since the application being developed is primarily a web application that was created using NextJS, testing will be mostly focused on the ensuring that the features and functionalities respond appropriately to the user's interactions.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
TA	Teaching Assistant who is a client of the project.
UI	User Interface.
API	Application Programming Interface
JS	JavaScript. The language used to create the project.
URL	Uniform Resource Locator.
FR	Functional Requirements.
NFR	Non-functional Requirements.
HTTP	Hypertext Transfer Protocol

Table 3: **Table of Definitions**

Term	Definition
CryptoMiner	The name of the project in development.
Cryptocurrency	Digital currency secured by cryptography that uses a decentralized system to record transactions.
Browser	Software application utilized to access the World Wide Web.
React	JavaScript library that was used to create the reference project.
NextJS/Next.js	The specific React framework that will be utilized to develop and organize the web application.
Bug	Fault in the code that produces an incorrect/unexpected result.
Database	Organized collection of data.

1.4 Overview of Document

This document will summarize the testing plan for the web-based application CryptoMetrics. It will go over the various testing techniques which will be utilized, for example, manual testing, end-to-end testing and unit testing. This document will also highlight the tests made for the functional and non-functional requirements of the system and would also cover the tests for the PoC.

2 Plan

2.1 Software Description

The software will act as a resource to our targeted audience for looking up more information about their favourite cryptocurrencies. It is a web application where the user can access data on cryptocurrencies regarding its pricing, charts, its performance over a span of time and visuals. The implementation will be done in NextJS.

2.2 Test Team

The test team consists of the existing developers: Himanshu Aggarwal, Saif Fadhel, Vanshaj Verma.

2.3 Automated Testing Approach

The `cypress` library will be used to automate most of the frontend tests for this project. The approach is to do mostly automated integration and end-to-end testing to verify if the features are working as expected. The `cypress` library allows us to mimic "clicks" on a webpage and observe the resulting behavior and changes in the webpage. Using this, we can effectively automate most situations which would otherwise require manual testing.

Apart from this, `Jest` will be used to do unit testing for some utility methods.

2.4 Testing Tools

Majority of integration and end-to-end testing will be done either manually or using the `cypress` library. Unit testing will be done using `Jest`.

2.5 Testing Schedule

See the Gantt Chart here.

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Home Page

Front End Testing

1. FT-HP-1

   Type: Functional, Dynamic, Automated

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The user clicks on the 'Table' icon.

Output/Result: The system shall display every cryptocurrency's relevant information in a table-like manner.

How test will be performed: This test will be performed using the `cypress` library to verify if the Table component gets rendered to the DOM.

2. FT-HP-2

   Type: Functional, Dynamic, Automated

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The user uses the Search input to a valid name of a cryptocurrency.

   Output/Result: The cryptocurrencies being displayed shall be filtered based on user's input.

   How test will be performed: This test will be performed using the `cypress` library to verify if typing a cryptocurrency name filters the cryptocurrencies being displayed.

3. FT-HP-3

   Type: Functional, Dynamic, Automated

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The user uses the Search input to an invalid name of a cryptocurrency.

   Output/Result: The application shall display a message to indicate that the search query did not find any results.

   How test will be performed: This test will be performed using the `cypress` library to verify if the message is shown.

4. FT-HP-4

   Type: Functional, Dynamic, Automated

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The user uses filter options to add a filter.

Output/Result: The cryptocurrencies being displayed shall be updated based on the selected filters.

How test will be performed: This test will be performed using the `cypress` library to verify if adding a filter changes the data accordingly.

5. FT-HP-5

   Type: Functional, Dynamic, Automated

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The user removes a filter.

   Output/Result: The cryptocurrencies being displayed shall be updated based on the new filters.

   How test will be performed: This test will be performed using the `cypress` library to verify if removing a filter updates the data accordingly.

6. FT-HP-6

   Type: Functional, Dynamic, Manual

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The user uses custom sorting buttons to sort the currencies in order of how they shall appear in the list.

   Output/Result: The cryptocurrencies being displayed shall keep the selected order and maintain the order on page refreshes.

   How test will be performed: This test will be performed manually to see if the displayed charts can be rearranged in a custom order.

3.1.2 Details Page

Front End Testing

1. FT-DP-1

   Type: Functional, Dynamic, Automated

   Initial State: The server for the NextJS app is started and the web page is opened.

Input/Condition: The user opens the detail page for a cryptocurrency.

Output/Result: The cryptocurrency's hourly, weekly, monthly and yearly historical pricing shall be displayed using a chart.

How test will be performed: This test will be performed using the `cypress` library to verify if a chart is present along with options to change the time range of the data.

3.1.3 Compare Page

Front End Testing

1. FT-CP-1

   Type: Functional, Dynamic, Automated

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The user navigates to the compare page.

   Output/Result: A chart shall be displayed along with options to select different cryptocurrencies for comparison.

   How test will be performed: This test will be performed using the `cypress` library to verify if a chart is present along with options to select different cryptocurrencies.

2. FT-CP-2

   Type: Functional, Dynamic, Automated

   Initial State: The server for the NextJS app is started and the compare page is opened in the web app.

   Input/Condition: The user selects the first cryptocurrency for comparison.

   Output/Result: After the first cryptocurrency is selected, the user shall not be able to select the same cryptocurrency a second time.

   How test will be performed: The `cypress` library will be used to verify that the option to add the same cryptocurrency is removed.

3.1.4 Fetching Data

Front End Testing

1. FT-FD-1

   Type: Functional, Dynamic, Manual

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The data is fetched using HTTP protocol.

   Output/Result: The data fetched should be cached for ~~1 minute~~ 5 minutes and refreshed after ~~1 minute~~ 5 minutes.

   How test will be performed: This test will be performed manually to see if the data displayed refreshes on page change after ~~1 minute~~ 5 minutes.

2. FT-FD-2

   Type: Functional, Dynamic, Manual

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The API call to fetch data results in an error.

   Output/Result: An error dialog/toast shall be displayed to alert the user of the error.

   How test will be performed: This test will be performed by manually creating API endpoints that result in an error to see if the error message is shown.

3. FT-FD-3

   Type: Functional, Dynamic, Manual

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The API call to fetch data results in an error.

   Output/Result: The system shall wait 10 seconds and retry the API call.

How test will be performed: This test will be performed by manually creating API endpoints that result in an error to see if the call is made again after 10 seconds.

4. FT-FD-4

   Type: Functional, Dynamic, Manual

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The API call to fetch data takes longer than 10 seconds.

   Output/Result: The system shall cancel the request and treat this as an error.

   How test will be performed: This test will be performed by manually creating API endpoints that result in a long turnaround times to see if the API call is timed-out.

3.1.5 User Interface

Front End Testing

1. FT-UI-1

   Type: Functional, Dynamic, Automated

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The user wants to change the theme of the web app and clicks a theme toggle in the sidebar.

   Output/Result: The system shall change the theme of all the elements being displayed and remember these preferences for page refreshes.

   How test will be performed: The `cypress` library will be used to verify that toggling the theme option changes the theme from dark to light and vice versa.

2. FT-UI-2

   Type: Functional, Dynamic, Automated

Initial State: The server for the NextJS app is started.

Input/Condition: The user opens the web page or navigates to another page.

Output/Result: The system shall display skeleton placeholders while waiting for an API response for the data.

How test will be performed: ~~The `cypress` library~~ Manual testing will be used to verify that skeleton placeholders are being displayed while API is being loaded.

3. FT-UI-3

   Type: Functional, Dynamic, Manual

   Initial State: The server for the NextJS app is started.

   Input/Condition: The user adds a filter or changes other options.

   Output/Result: The system shall automatically save the changes to local storage and load them automatically upon starting the product in the future.

   How test will be performed: ~~Manual testing~~ The `cypress` library will be used to observe if preferences are loaded back on page refreshes.

3.2 Tests for Nonfunctional Requirements

3.2.1 Look and Feel

1. NFT-LF-1

   Type: Structural, Dynamic, Manual

   Initial State: The server for the NextJS app is started.

   Input/Condition: Users are asked to launch the web page.

   Output/Result: At least 80% of users shall agree that the product feels professional, trustworthy, and informative.

   How test will be performed: A test group of people (that are representative of the target audience) will be shown the application for the first time and asked whether they feel the application is professional, trustworthy, and informative. If more than 80% of people agree, it should be considered a success.

3.2.2 Usability and Humanity

1. NFT-UH-1

   Type: Structural, Dynamic, Manual

   Initial State: The server for the NextJS app is started and the web page should be launched.

   Input/Condition: Users are asked to navigate between different pages and use the functionality.

   Output/Result: At least 80% of users shall be able to navigate through the different pages without any training.

   How test will be performed: A test group of people with a basic understanding of the English language and not prior training or experience with cryptocurrencies will be asked to navigate through the different pages and components. If more than 80% of people are able to perform the tasks, it should be considered a success.

2. NFT-UH-2

   Type: Structural, Dynamic, Manual

   Initial State: The server for the NextJS app is started and the web page should be launched.

   Input/Condition: Users are asked the meaning of the icons they see on the web page.

   Output/Result: 95% of users shall be able to understand what the icons represent.

   How test will be performed: A diverse group of people will be shown the web page and asked the meaning of the various icons on the page. If more than 95% of people are able to correctly explain what the icons represent, it should be considered a success.

3. NFT-UH-3

   Type: Structural, Dynamic, Manual

   Initial State: The server for the NextJS app is started.

   Input/Condition: Testers will be asked to open the web page on different devices with varying screen sizes.

Output/Result: The web page shall automatically adjust to the size of the screen.

How test will be performed: A team of testers will be asked to open the web page on different devices with varying screen sizes (desktop, tablet and mobile devices). If the web page adjusts to the size of the screen at least 95% of the times, it should be considered a success.

3.2.3 Performance

1. NFT-P-1

   Type: Structural, Dynamic, Manual

   Initial State: The server for the NextJS app is started.

   Input/Condition: Testers are asked to launch the web page on different types of devices.

   Output/Result: The product shall initialize and setup in less than 10 seconds.

   How test will be performed: A team of testers will be asked to launch the web page on devices varying from mobile to desktop, and from slow to fast. Regardless of the device, if the web page opens up in less than 10 seconds, it should be considered a success.

2. NFT-P-2

   Type: Structural, Dynamic, Manual

   Initial State: The server for the NextJS app is started and the web page is launched.

   Input/Condition: Testers are asked to navigate to different pages and observe the time.

   Output/Result: The product shall navigate to different pages within 1 second for 95% of times, and within 5 seconds for the rest of the times.

   How test will be performed: A team of testers will be asked to navigate to different pages of the application on devices varying from mobile to desktop, and from slow to fast. Regardless of the device, if the web page successfully navigates to other pages in less than 1 second

for 95% of times and in less than 5 seconds for the rest of the times, then it should be considered a success.

3.3 Traceability Between Test Cases and Requirements

Test Case ID	Requirement ID
FT-HP-1	FR-1
FT-HP-2	FR-2
FT-HP-3	FR-2
FT-HP-4	FR-5
FT-HP-5	FR-5
FT-HP-6	FR-6
FT-DP-1	FR-3
FT-CP-1	FR-4
FT-CP-2	FR-4
FT-FD-1	FR-7
FT-FD-2	FR-8
FT-FD-3	FR-9
FT-FD-4	FR-11
FT-UI-1	FR-12
FT-UI-2	FR-10
FT-UI-3	FR-13
NFT-LF-11	NFR-3
NFT-UH-1	NFR-6
NFT-UI-2	NFR-9
NFT-UI-3	NFR-10
NFT-P-1	NFR-11
NFT-P-2	NFR-12

Table 4: **Traceability Matrix**

4 Tests for Proof of Concept

4.1 Table/Card View Testing

Table and Card View Tests

1. POC-TCV-1

   Type: Functional, Dynamic, Manual

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The user clicks on the 'Table' icon and then on the time tag to toggle between 24h, last week, and last month, last year.

   Output/Result: The system shall display every cryptocurrency's relevant information in a table-like manner based on the interval specified in the time tag.

   How test will be performed: This test will be performed manually by running the API calls separately to verify that the information matches what is displayed in the table-view card appropriately.

2. POC-TCV-2

   Type: Functional, Dynamic, Manual

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The user clicks on the 'Card' icon and then on the time tag to toggle between 24h, last week, and last month, last year.

   Output/Result: The system shall display every cryptocurrency's relevant information in a card-like manner based on the interval specified in the time tag.

   How test will be performed: This test will be performed manually by running the API calls separately to verify that the information matches what is displayed in the table-view card appropriately.

4.2 Search Feature Testing

1. POC-SFT-1

   Type: Functional, Dynamic, Manual

   Initial State: The server for the NextJS app is started and the web page is opened.

   Input/Condition: The user clicks on the search bar and starts typing their expected search term.

   Output/Result: The system shall filter the cards according to the search term entered in the search box and only the cards with matching names would be displayed on the screen.

   How test will be performed: The tests will be performed using the cypress library by comparing the data output of the search filter and comparing it to the expected results.

5 Comparison to Existing Implementation

There have been several changes and new additions separating CryptoMetrics, our project, from Cryptodash, the existing implementation. The existing application primarily displayed a list of cryptocurrencies in tabular form, whereas CryptoMetrics allows users to view this list in both a card view and a table view containing even more columns of information and a mini graph displaying cryptocurrency price changes in the past user specified time. CryptoMetrics also added a product page which allows users to get a detailed view of every cryptocurrency listed in the web app, which was not a part of the existing implementation. Furthermore, the project also adds a page dedicated to comparing cryptocurrencies with one another to indicate changes in projected price increases and decreases to better advise users so they make informed investment decisions. The original implementation did not include any testing within the project's files. Our implementation will be verified rigorously through the use of a test report.

6 Unit Testing Plan

This project will use the Jest library to do unit testing.

6.1 Unit testing of internal functions

Unit testing will be done on internal utility methods that return a specific output value. These methods will be given specific input values for normal, edge, and boundary cases and the actual and expected outputs will be compared for validation. Unit testing for this application does not require stubs, drivers or coverage metrics.

6.2 Unit testing of output files

The application will not be generating any output files. Therefore, there will be no unit testing of output files.

7 Appendix

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance. N/A

7.2 Usability Survey Questions

The following survey will be filled out by members of the survey group.

CryptoMetrics Survey

Please fill out the survey after using the application.

Time spent using software:

Provide a rating on a scale from 1 to 5 (where 1 represents a poor review and 5 represents a strong review) in each of these questions by filling in the number of circles corresponding to your desired rating.

How easy is the website to use?	○	○	○	○	○
	1	2	3	4	5
How appealing is the app visually?	○	○	○	○	○
	1	2	3	4	5
How intuitive is the website?	○	○	○	○	○
	1	2	3	4	5
How responsive is the website?	○	○	○	○	○
	1	2	3	4	5
How trustworthy does the website look?	○	○	○	○	○
	1	2	3	4	5