SETU Carlow

HyperLedger Fabric

Auctioning System – Functional Spec

Student - Oisin Hickey – C00247185

11-30-2022

Table of contents

Document Versions	
First Version	08-01-2023
Final Version	16-04-2023

Abstract

An auction system is a market mechanism in which potential buyers bid against each other to purchase a particular item or service. Auctions can be conducted in various formats, such as sealed-bid, open-outcry, and online auctions.

In an auction, the seller sets a starting price or reserve price, and interested buyers then place bids on the item, each successive bid increasing the price until the seller accepts the highest bid. The seller may also set a time limit for the auction, and at the end of the auction, the highest bidder is typically awarded the item.

Auctions can be used for a wide range of items, including art, real estate, automobiles, and commodities. They can be conducted by individuals, auction houses, or through online platforms and are often used to facilitate the sale of unique or rare items. Auctions can also be used for charitable purposes, with proceeds from the auction going to support a specific cause or organisation.

This document will lay out the blockchain and combined auctioning system that will be built along with its project success metrics.

Introduction

Auctioning systems have been used since medieval times, and decentralised ledger systems have been used since the banking systems of ancient Rome. Although these systems are hugely popular, they are somewhat flawed as auctioning systems depend on their management or control by one company, and decentralised technologies are often highly complex, with steep learning curves involved in implementing the software.

Purpose

This functional specification lays out the intended design process, debugging process and architecture of the application-specific blockchain, along with the functionality it should allow for, from making basic currency transfers to making auctions on the chain. It will outline the intended platform and user authentication methods along with aspects of the backend structure. It will also outline some of the technologies used and the reasoning behind them. FURPS and metrics for project success will also be outlined in the document.

Functioning Process Overview

In the following section, I will explore the functioning processes of the technologies most relevant to running the kind of functional blockchain I will be building. The primary underlying system within this project that the network relies on is P2P. I will also explore front-end development, backend development and underlying application structure. I will also look at how an auctioning system works and the basic process my solution needs to attain. I will then explore my solution's proposed functioning process and how it will work in day-to-day usage by an organisation or user.

Auction System Functioning Process

Auctioning systems have been around since medieval times. In the context of computers, I will be making a virtual auctioning system. A typical virtual auctioning system consists of several fundamental functional aspects:

Auction Creation

- A user may add an item or service they own onto the platform as an auction.
- A user may specify a starting bid for an auction and other details like payment methods accepted or auction end time.

Auction Management

- A user may close an auction meaning they hold ownership of an item, but it is no longer available for bidding.
- A user may reinitiate an auction allowing them to put a previously closed or ended auction back up for auction
- A user may end an auction meaning the ownership of the item is passed to the highest bidder

User Interaction

- A user may place a bid on an item
- A user may retrieve an item's history

- A user may retrieve a wallet’s history

Typically, users must first register a valid account to access the auctioning database and the other items on the site. For this examination, I will talk about eBay.

Auction-style listings on eBay work best when a user submits a good choice of pictures and descriptions and relies on the power of suggestion to sell an item to another user. The items a user lists should be things that will sufficiently describe the item to other users.

The item will then appear publicly online for the rest of the user base to see, after which they can begin bidding with prices which increase from the starting price set by the auction creator. A user begins an auction by setting a starting price. If there is no starting bid, the seller sets one - usually half the item's value.

As users bid a price, they can only bid up from the starting bid, which will steadily rise as users compete for the winning bid. So, for example, if a user's starting bid is €5 and someone else outbids at €7, they would now be the item's owner, and the initial bidder would have lost that auction.

After the auction has hit the time set by the user or the owner ends the bidding, the item or service will be given to the account of the highest bidder, the individual who bided the highest amount.

Web Application Functioning Process

GUI

Most web applications run only in the browser and use it as their runtime environment. In this instance, I will use React to create my front end, with CSS used throughout for styling. This gives me room to wrap it with Electron for desktop usage, as React is compatible with Electron.

When the user tries to connect to a React or HTML-based site, the first thing to happen is a request is sent from the client machine to the server machine and from their data is returned from storage on the server to the client and displayed within the browser. All of this will occur locally in the Electron wrapper.

Components in React can best be described as grouping pieces of code into blocks and rendering or changing those blocks based on server or client-side responses or inputs which are read by the browser. The browser acts virtual I/O when interacting with React.

Some essential components most auctioning sites use are:

- Navbars are usually located at the top of the screen and assist the user in moving around the site.
- Auction item components, outlining how an item should be displayed after retrieving it from an external database
- Login screen, the method by which a user authenticates to the system

Several of these components will be used to build up the front end for the users to interact with the system, particularly the navbar, login screen and the components to render auction items.

API

An API (Application Programming Interface) is a set of protocols, routines, and tools for building software applications. It provides a way for different software systems to interact, allowing developers to access and use the functionality and data of another system without having to understand the underlying code or implementation. An internal API will be used in the backend of this project to allow data to be transferred throughout the chain and retrieved from it. (*What Is an API?,* n.d.)

APIs can be public or private, but in the case of this project, it will be private and only used in this application. They can expose a wide range of functionality, such as data storage and retrieval, authentication and authorisation, and real-time messaging. Documentation will be provided regardless, in case users wish to adapt it. APIs can be accessed through various programming languages and protocols, including REST (Representational State Transfer), SOAP (Simple Object Access Protocol), and GraphQL. My API will steer towards being REST-based, relying on web technologies and JavaScript.

The API I intend to build for this project's backend will access a NoSQL-like database system with endpoints for each aspect of the project's functionality. (*Building A Distributed Database Step by Step…(Part 1) | by Rajdeep Das | Medium,* n.d.) An API keys system will not be used to authenticate to the API, and it will be accessed using the GUI front end to send POST and GET requests to its endpoints.

Solution's Proposed Blockchain Functioning Process

This solution will use an application-specific blockchain system as the main backbone of the application.

- The first part of the development will be dedicated to creating the chain object and the methods to update and verify the chain and its contents. Chains that are updated and sent to other peers must be validated and verified.
- The backend will also require mining components. New transactions to the chain must go into the transaction pool. They will be held in the transaction pool until users mine the pool and the new transactions are added to the chain.
- One of the core components linking these two objects together is the wallet. A wallet must have a public and private key. The public acts like a username or bank account number. It will be how users communicate actions with each other on the chain.
- P2P functionality will also be implemented using the common JavaScript practice of building a publisher-subscriber class to handle this communication. In addition, an external public API will be built upon to handle this functionality.
- Transactions will represent communications on the chain. Transactions will be processes which occur on the chain and are written into a block on that chain.
- The three primary transaction forms will be auction creation, bids and currency transfer.
- When the transaction pool is mined on a chain, it will be cleared, and the updates to the pool and chain will be broadcasted to other peers.
- Users will authenticate to the system using a valid BIP39 phrase.

Project Plan

I aim to allocate a month to my research and compile the document. From there, I plan to use the research I have collected to lay out my functional specification and some of the technologies I intend to use. Coding will begin after the first three months and will focus on getting basic chain functionality working, such as the block constructor and wallet constructor. Integrating the API with the wallet and transaction functionality will be allocated three weeks, and from there, I aim to test the chain across three weeks. Getting the base chain system working will use unit testing, as my research has helped me find great examples. However, I will aim to manually test everything I add to get a fully working system. Upon completing the system, I will finish the report within one month.

Task	Time
Research document	2 Months
Functional specification	1 Month
Begin initial work on block and chain	2 Weeks
Began work on API	3 Weeks
Finish backend work and test	3 Weeks
Test system	2 weeks
Finish report	1 Month

Use and Misuse cases

Use and misuse cases are vital aspects of any software engineering project, which is valid for this project. The following section will outline how the system is designed to be and should be used, some of the ways it can be misused, and how these threat scenarios are mitigated.

User definitions

A user possesses the software and attempts to interact with the chain. Depending on the context of their interaction with the chain, a user can be a seller, bidder or payee. All users share equal privilege to sell, buy and transfer currency throughout the chain, with the currency and items they own being retrieved and written to the chain.

Use Cases

Several valid use cases exist for the system regarding what functionality a user can avail of.

Use Case 1	
Actor	User
Function	Auction creation
Trigger	The user navigates to the auction creation panel, fills out the input fields, and hits submit.
Pre-conditions	The user is authenticated to the system via their wallet and on the create an auction panel.
Result	The created auction is added to the mining pool
Actor Context	Seller

Use Case 2	
Actor	User
Function	Bid creation
Trigger	The user navigates to the bid creation panel, fills out the input fields, and hits submit.
Pre-conditions	The user is authenticated to the system via their wallet and on the place a bid panel.
Result	The created bid transaction is added to the mining pool
Actor Context	Bidder

Use Case 3	
Actor	User
Function	Currency Transfer
Trigger	The user navigates to the currency transfer panel, fills out the input fields, and hits submit.
Pre-conditions	The user is authenticated to the system via their wallet and on the currency transfer panel.
Result	The currency transaction is added to the mining pool
Actor Context	Payee

Use Case 4	
Actor	User
Function	Item history retrieval
Trigger	The user navigates to the item history panel, fills out the input fields, and hits submit.
Pre-conditions	The user is authenticated to the system via their wallet and on the item history panel.
Result	The item history is retrieved from the chain and returned to the user
Actor Context	User

Use Case 5	
Actor	User
Function	Wallet history retrieval
Trigger	The user navigates to the wallet history panel, fills out the input fields, and hits submit.
Pre-conditions	The user is authenticated to the system via their wallet and on the wallet history panel.
Result	The wallets history is retrieved from the chain and returned to the user
Actor Context	User

Use Case 5	
Actor	User
Function	Login
Trigger	The user opens the software and navigates to the passphrase login panel
Pre-conditions	The user is authenticated to the system via their wallet and on the wallet history panel.
Result	The wallets history is retrieved from the chain and returned to the user
Actor Context	Unauthenticated User

Misuse Cases

Several misuse cases exist for various components of the system, and mitigations are implemented to handle several of these.

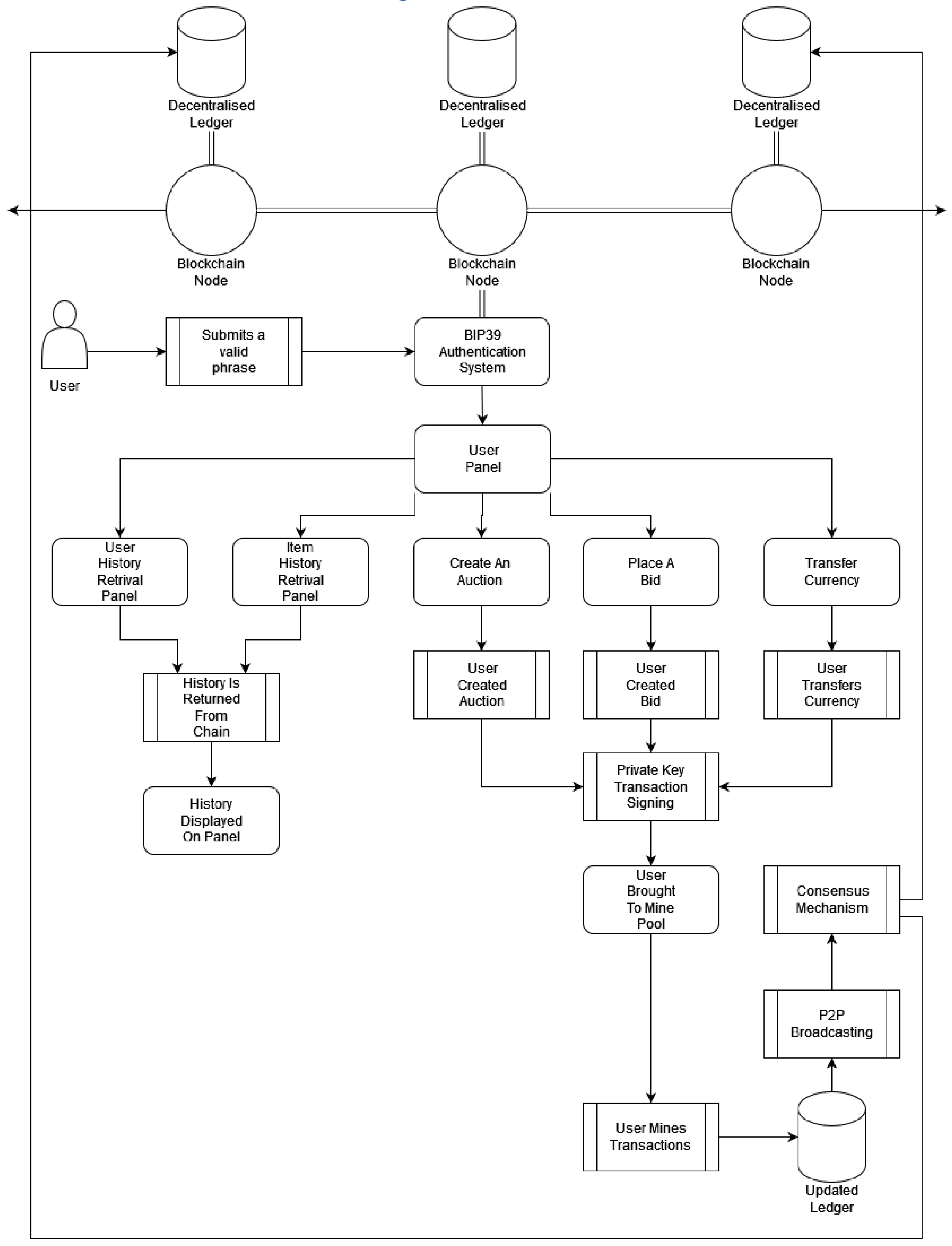
Misuse Case 1	
Actor	Threat Actor
Threat Goal	Send valid transactions using another user's public key
Trigger	The threat actor sends a request to a transaction endpoint using a different user's public key.
Pre-conditions	The user is authenticated to the system and possesses another user's public key.
Result	The threat actor creates a transaction with ownership assigned to the wrong user.
Mitigation	Transactions are signed on creation with the current wallet's private key

Misuse Case 2	
Actor	Threat Actor
Threat Goal	Send valid transactions using an incorrect balance
Trigger	The threat actor requests a transaction endpoint using an incorrect balance amount.
Pre-conditions	The user is authenticated into the system
Result	The threat actor creates a transaction with an incorrect balance causing the chain data to be falsified.
Mitigation	Wallet balance for a wallet is calculated based on existing chain data

Misuse Case 3	
Actor	Threat Actor
Threat Goal	Mine a block using easier or harder difficulties
Trigger	The threat actor sends a request to mine the transaction pool with greater or lower difficulty.
Pre-conditions	The user is authenticated into the system
Result	The threat actor manipulates the difficulty of mining blocks overall on the chain.
Mitigation	The difficulty is calculated using a timestamp and the previous block's difficulty level.

Misuse Case 4	
Actor	Threat Actor
Threat Goal	Submit an invalid chain to other peers
Trigger	The threat actor sends a request to submit an invalid chain
Pre-conditions	The user is authenticated into the system
Result	The threat actor manipulates the P2P system to submit an invalid chain
Mitigation	The consensus algorithm verifies the chain before broadcast

Use Case & Mis-Use Case Diagrams



The blockchain operates as nodes, with a copy of the ledger stored on each node. Upon a user making any form of transaction, their private key is used to sign it, verifying it to be valid and genuine from the user. All transactions are added to the mining pool, and when mined, an updated copy of the ledger is made for broadcast. Requesting history retrieves from the chain, so verification or signing is not required. The consensus mechanism checks broadcasted chains to ensure their validity. Users authenticate to the system using a valid BIP39 phrase to generate a public and private key.

Solution Resources

For this project, I will use visual studio code on my local machine to write my code and GitHub to back it up and help verify that I made the source. In addition, I will use sync back for syncing to my local backups, and I will host a single root node for demonstration purposes. For hosting, I plan to use Heroku.

Hardware

The entirety of the development will be hosted on my personal computer due to its high specs allowing me to run a development environment with few limitations. In addition, it allows me to also gauge the overall usability of the project as I can run several nodes on my machine to test it. (*Legion 5 Series | Gaming Laptops | Lenovo IE*, n.d.)

Host Machine Specifications:	
Machine Name:	Lenovo Legion 5
Ram:	32 GB
Storage:	4 TB
CPU:	AMD Ryzen 7 5800H
GPU:	Nivida GeForce RTX 3060
Operating System:	Windows 11 Version: 22H2

Figure 1: Host Machine Specifications

Software

For this project, I intend to use only JavaScript libraries, as integration with other languages makes things needlessly complex. Also, only using JavaScript libraries makes hosting it on Heroku more affordable as the more technologies or languages I use can potentially increase the overall cost of hosting. Finally, using only JavaScript libraries and technology makes debugging more efficient and straightforward.

Node Js

Node JS is a Linux foundation-produced piece of software which allows for JavaScript code to be run locally on a machine. Node has support for HTTP server creation, and the usage of GUI creation libraries like React and Electron and its virtual machine architecture allows things to be run in a control flow loop. This means objects can be observed stored in heap managers, like a regular PC. (*About | Node.Js*, n.d.) Node will be a vital part of this project allowing for the JavaScript languages base functionality to be run in a desktop environment.

NPM – Node Package Manager

One of the biggest software registries in the world is npm (Node Packcage Manager). All open-source JavaScript developers use npm to exchange and borrow packages, while many businesses use npm to oversee private development. (*About Npm | Npm Docs*, n.d.) In this project, I will be using NPM through Windows command line. NPM will allow the JavaScript language base functionality to be extended. Some of the critical packages I'll be using are outlined below.

React

React is an open-source JavaScript library for building user interfaces or UI components for web applications. Developed by Facebook, React was designed to simplify the process of building complex, interactive, and reusable UI components by breaking them down into smaller, manageable pieces. (*React-Bootstrap · React-Bootstrap Documentation*, n.d.)

React uses a declarative programming approach, where the developer specifies what the UI should look like based on the current state of the application rather than how to update the UI in response to user input or changes in the application's state. This makes it easier to develop and maintain complex UIs, enabling faster rendering and better performance. React is widely used in web development and has a large and active community of developers contributing to its ongoing development and improvement.

React is one of the critical tools I intend to use for my GUI. Although React uses a markup language-like structure, it is not a markup language or its language as it is simply a way of creating GUI's that a browser can interpret. (*React*, n.d.)

Express

The Express library is a popular Node.js framework for building web applications. (*Express - Npm*, n.d.)It provides features and tools that simplify building scalable, secure, high-performance web applications. Some of the key features of Express include:

- Routing: Express provides a flexible and powerful routing system that allows developers to define URL patterns and associate them with specific actions or functions.
- Middleware: Express middleware is a series of functions executed in a specific order while processing a request. Middleware can be used to perform everyday tasks such as logging, authentication, or error handling.
- Templating Engines: Express supports a variety of templating engines that allow developers to create dynamic HTML pages.
- Error handling: Express provides a robust system that allows developers to handle errors gracefully and prevent server crashes.
- Security: Express includes various security features, including support for HTTPS, CSRF protection, and XSS protection.

Express will be used to build an API for this system, and routing, middleware and error handling will all be used. Templating engines will not be used, and security will not be a focus of the API backend as it will be run locally and sync over a publisher-subscriber structure. Several users will not all access the same instance of the software, each with its instance. Axios is also a good way to fulfil this part of development.

BIP39

BIP39 is a Bitcoin Improvement Proposal (BIP) that outlines a standard way to create a mnemonic sentence or seed phrase for generating deterministic wallets. A mnemonic sentence is a series of words, typically 12 or 24, representing a random sequence of bits that can generate private keys for cryptocurrencies. (*Bip39 - Npm*, n.d.)

BIP39 defines a list of 2048 words used to create the mnemonic sentence. Each word represents 11 bits of entropy, and a 12-word mnemonic sentence provides 128 bits of entropy, which is considered sufficient to secure the private keys for most cryptocurrencies. In addition, the same set of words can be used to generate multiple private keys, making it easier to back up and restore wallets.

BIP39 also includes a checksum that is used to ensure the accuracy of the mnemonic sentence. The checksum is generated by taking the first few bits of the SHA-256 hash of the entropy bits and adding them to the end of the mnemonic sentence. This makes it possible to detect and correct any errors in the mnemonic sentence.

BIP39 will bring great security benefits to this system. It will remove the need for the system to store user authentication details anywhere on the chain but still allow access to the exact user details every time.

Electron

The Electron framework is a popular open-source framework for building desktop applications using web technologies such as JavaScript, HTML, and CSS. It allows developers to create cross-platform desktop applications for Windows, macOS, and Linux using a single codebase.

Electron was created by GitHub in 2013 as a way to build the Atom text editor, but it has since grown in popularity. Many other companies and developers now use it to build desktop applications. Electron applications are built using Chromium, the open-source browser project that powers Google Chrome, and Node.js, a popular JavaScript runtime for building server-side applications. (*Electron JS + React JS + Express JS | by Keshav Agrawal | Medium*, n.d.)

One of the advantages of using Electron is that it allows developers to leverage their existing web development skills to build desktop applications rather than having to learn new languages and frameworks for each platform. Electron also provides a rich set of APIs for interacting with the underlying operating system and hardware, making it easy to create native-like desktop applications. (*Build Cross-Platform Desktop Apps with JavaScript, HTML, and CSS | Electron*, n.d.)

Although Electron is looked at as a whole here, it will only be used to wrap the developed web application, and none of its more functional or extended APIs or components will be used.

HTML, CSS

HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are two fundamental languages used to create websites and web applications.

HTML is a markup language used to create the structure of a web page. It consists of a set of tags and attributes that define the various elements of a web page, such as headings, paragraphs, images, links, forms, and tables. HTML provides the basic structure of the page, but it does not control the visual appearance of the elements. (*HyperText Markup Language (HTML): What It Is, How It Works*, n.d.)

On the other hand, CSS is a style sheet language used to define the visual appearance of a web page. It is used to control the layout, font, colour, and other visual elements of a web page. In addition, CSS is used to style HTML elements by selecting them with selectors and then applying styles. CSS is designed to separate the presentation of a web page from its content, which makes it easier to maintain and update the visual appearance of a website. (*What Is CSS? - Learn Web Development | MDN*, n.d.)

Together, HTML and CSS are used to create a web page's content and visual design. HTML provides the page's basic structure and content, while CSS controls the layout and appearance of the content.

The project's React front end will be styled using CSS; however, it will not be written using HTML. The structure of React components is likened to HTML; hence it is essential to mention it. When React code is compiled and run, it will be interpreted by a user's browser or the Electron web wrapper as HTML and CSS, allowing for broader compatibility with various devices and browsers.

JavaScript

JavaScript is a high-level, dynamic, interpreted programming language used primarily to add interactivity to web pages. It is a client-side scripting language that runs in the user's web browser rather than on a web server. JavaScript is one of the three essential technologies of the World Wide Web, along with HTML and CSS.

JavaScript allows developers to create interactive web pages by responding to user actions, such as clicking a button or filling out a form. It can be used to manipulate the content of a web page, validate user input, create animations, and communicate with web servers to dynamically update the content without requiring a page reload.

JavaScript is a universal language, and it can be used for a wide range of applications beyond web development, such as server-side programming, desktop and mobile applications, game development, and more. All major web browsers, including Chrome, Firefox, Safari, and Internet Explorer, support it. (*What Is JavaScript? - Learn Web Development | MDN*, n.d.)

Javascript is the primary language used throughout the project and is a vital technology.

Heroku

Heroku is a cloud-based Platform-as-a-Service (PaaS) provider that allows developers to build, run, and operate applications entirely in the cloud. It was founded in 2007 and acquired by Salesforce in 2010. (*About Heroku | Heroku*, n.d.)

Heroku provides a streamlined development process by allowing developers to focus on their code rather than worrying about the underlying infrastructure. As a result, developers can quickly deploy and scale their applications without worrying about the complexity of setting up and maintaining servers.

Heroku supports various programming languages like Node.js, Ruby, Python, Java, and more. It also provides various third-party add-ons and services easily integrated into applications, such as databases, monitoring, analytics, and more.

One of the key benefits of Heroku is its ease of use. Developers can quickly deploy their code by using the Heroku CLI (Command Line Interface) or by integrating with a Git repository. Heroku also provides a web-based dashboard that allows developers to manage their applications and resources. (*Personal Apps | Heroku*, n.d.)

Heroku is the perfect low-cost solution to host a root peer on.

Development Environment Configuration

I will be using Node, NVM and NPM to help manage installed dependencies and updates for them and run the codebase on my local machine.

I will use Visual Studios' built-in terminal functionality to run Node and NPM commands when using Node. In addition, I will be able to configure specific script commands or functions using the scripts section of the projects package.json file.

Any progress will be backed up to my GitHub, verifying my progress and allowing me to roll back changes, if need be, to older commits. On my Windows host OS, I will use GitHub desktop as it's a more straightforward method than git command line on Windows and Ubuntu; I'll use the command line tool; however, most of my files will be exported from Ubuntu first for me to work with them.

Jest

Jest is a popular open-source JavaScript testing framework created by Facebook. It has been commonly used for testing JavaScript code, especially code that runs in a browser or interacts with a browser's Document Object Model (DOM). Jest provides developers with a comprehensive toolset for writing, running, and debugging tests, making it an excellent choice for both beginners and experienced developers. (*Jest - Npm,* n.d.)

Jest uses automated and manual testing techniques to ensure your JavaScript code is functioning correctly. It supports various tests, including unit, integration, and snapshot tests. Jest also comes with a built-in test runner that can execute tests in parallel, making it an efficient choice for testing large projects. For this project, I intend to use Jest for unit tests.

Some of the main features of Jest include:

- Easy setup and configuration: Jest can be easily integrated into a project using npm or Yarn, and it comes with a default configuration that works well for most projects.
- Powerful mocking capabilities: Jest allows developers to create mock functions and modules, making complex simulating interactions between components easy.
- Snapshot testing: Jest allows developers to capture snapshots of their components and compare them to previous versions, making it easy to detect changes that may have introduced bugs.
- Code coverage reporting: Jest can generate code coverage reports showing which parts of your code are being tested and which are not.

Jest will be used to help me ensure I've met most of the base criteria of the chain system I want to build, and once I have a basic chain system, I can verify its outputs using Jest and then build upon it for my full-fledged auctioning system.

Nodemon

Nodemon is a command-line utility that helps developers to monitor changes in the source code of a Node.js application and automatically restart the application when changes are detected. It eliminates the need to restart the server each time you change your code manually. This can save developers time and increase productivity.

Nodemon can be installed globally or locally on a specific project, and it can be used with various Node.js frameworks, such as Express, which I intend to use. When you run

Nodemon, it monitors the file system for any changes in the source code files, and if any changes are detected, it automatically restarts the application.

Nodemon can be customised with various options, such as ignoring specific files or directories, specifying a delay before restarting the application, or setting up custom scripts to run before or after the application restarts. (*Nodemon - Npm*, n.d.)

I will use Nodemon when testing the codebase outside the electron wrapper.

JSDoc

JSDoc is a markup language used to annotate JavaScript code. It allows developers to document their code, specifying the types of function parameters and return values, describing what a function does, and providing examples of how to use it. Other developers can use this documentation to understand how to use a function or class. Automated tools can also use to generate API documentation, type-checking, and code analysis.

JSDoc is based on the syntax of JavaDoc, a similar documentation system for Java code. JSDoc uses special comments that start with a /** sequence and end with a */ sequence to indicate that the following code is a documentation comment. Inside the comment, tags provide information about the documented code, such as @param to describe a function parameter or @returns to describe the return value.

Many popular JavaScript tools, such as ESLint and TypeScript, support JSDoc comments. Additionally, there are tools like JSDoc itself that can generate HTML documentation from JSDoc comments. (*Use JSDoc: Index*, n.d.)

As my project will be written using Javascript, JSDoc is an excellent way for me to make good documentation.

FURPS

Functionality, Usability, Reliability, Performance, and Supportability (FURPS) are software quality factors that Hewlett-Packard created. The FURPS quality criteria extensively use prior work in defining the characteristics listed below for each of the five main elements. These have been integrated into ISO 9126. (*Software Engineering-FURPS - Best Online Tutorials | Source Codes | Programming Languages*, n.d.)

Functionality

Functionality in terms of FURPS defines all of the functions which the software should be able to perform. A lot of these are essential functions; however, due to the different components involved in my project, the software's functionality can be broken down into three sections:

Blockchain Functionality Within Project

- Facilitate the connection between all nodes
- Perform automatic node registry (*Node.Js - How to Get My External IP Address in Node.Js App? - Stack Overflow*, n.d.)
- Store user-made transactions on the chain
- Store unverified transactions in a transaction pool
- Allow for mining of a transaction pool so its transactions are added to the chain

- Automatically clear the valid transactions from the pool
- Automatically broadcast changes to the pool or chain (*How to Manage Chat Message Timestamps | PubNub*, n.d.)
- Allow for the access or creation of valid wallets to the chain
- Use wallet keys to sign transactions as valid

Web Application Functions within the Project

- Allows a user to log in with a previous wallet or generate a new one by interfacing with API
- Allow a user to create an auction item
- Allow a user to place a bid on an existing item
- Allow a user to send another user chain currency
- Allow a user to close, reinitiate or end an auction on the chain
- Allow a user to search a wallet for its history
- Allow a user to search an auction item for its history
- Allow a user to mine transactions on their machine

Usability

Usability sets out the ease of use and access the software should maintain. In the instance of this project, the following should hold true:

- The software should utilise a GUI to communicate data and changes to the user
- Buttons should be used to make navigation and functionality easier for a user
- The software should be able to run on a desktop
- The software should ideally have documentation in the event a user wishes to modify or fork the code
- The system should ideally run as a portable exe
- The system GUI should ideally be compatible with multiple desktop platforms or browsers
- Pagination should make large amounts of data easier to navigate (*React Bootstrap Pagination Example (Forked) - CodeSandbox*, n.d.)

Reliability

In terms of reliability, the application should achieve several milestones:

- The system should use consensus to verify that the incoming chain is valid
- Cryptography should be used to ensure data integrity
- The system should have low downtimes due to its P2P structure
- The system should not store any user authentication details for security
- The system should utilise error-handling methods correctly
- Utilise a proof of work system for mining

Performance

Performance defines the speed and efficiency of the software under the hood. The following should be achievable for this project:

- The system should be low in resource consumption
- The system should be able to run on low-spec devices
- The system should be able to run a valid node in a web environment for demonstration purposes
- Mining a block should take around 1000 milliseconds or so on a new chain
- Sync chains and pools automatically

Supportability

Supportability should define how broadly the system is supported by other hardware or software, e.g. a more commonly unused browser or operating system.

- The system should use standard technologies like JavaScript, which various browsers and web technologies should widely support.
- The system should be well documented to make changes or improvements easier.
- The system should use a clear folder structure to allow users to navigate its code base.

Metrics of Project Success

There are several vectors of project success. Most of these vectors are functional requirements of the software, but failure to get them working would be a core breakdown in base system functionality.

- Successful CRUD operations on the chain, such as making, ending or bidding on an auction
- Successful user authentication to the system via BIP39 with its implementation leading to the same wallet being successfully retrieved every time
- Successful syncing of chain between nodes in the context of both the current chain and the transaction pool
- Successful chain validation across multiple validation criteria
- Successful minimal GUI integration so data is displayed to the user in an easily interpretable way
- Successful implementation of API to GUI communications and functionality

Testing

Testing the project overall will consist of several base steps and procedures. These tests will help me verify project functionality, find and patch bugs, and measure functionality at different points of the development process to help identify potential problems before they become giant bugs or issues.

- Running a root node on Heroku
- Testing wallet creation
- Testing consistent wallet recovery using a phrase
- Testing P2P transaction pool broadcast

- Testing submission of chain transactions
- Testing calculation of wallet balance
- Testing the API endpoints
- Testing recovery of a wallet or item history from chain
- Testing desktop client functionality
- Testing consensus algorithm
- Testing average work rate
- Testing mining rewards system
- Testing constructor and function return using unit tests

References

About | Node.js. (n.d.). Retrieved December 1, 2022, from https://nodejs.org/en/about/

About Heroku | Heroku. (n.d.). Retrieved April 13, 2023, from https://www.heroku.com/about

About npm | npm Docs. (n.d.). Retrieved December 1, 2022, from https://docs.npmjs.com/about-npm

bip39 - npm. (n.d.). Retrieved April 12, 2023, from https://www.npmjs.com/package/bip39

Build cross-platform desktop apps with JavaScript, HTML, and CSS | Electron. (n.d.). Retrieved April 12, 2023, from https://www.electronjs.org/

Building A Distributed Database Step by Step…(Part 1) | by Rajdeep Das | Medium. (n.d.). Retrieved April 11, 2023, from https://rajdeep-das.medium.com/building-a-distributed-database-step-by-step-part-1-abc7d944e52d

Electron JS + React JS + Express JS | by Keshav Agrawal | Medium. (n.d.). Retrieved April 6, 2023, from https://medium.com/@keshavagrawal/electron-js-react-js-express-js-b0fb2aa8233f

express - npm. (n.d.). Retrieved April 12, 2023, from https://www.npmjs.com/package/express

How to manage chat message timestamps | PubNub. (n.d.). Retrieved April 6, 2023, from https://www.pubnub.com/how-to/message-timestamps/

HyperText Markup Language (HTML): What It Is, How It Works. (n.d.). Retrieved April 13, 2023, from https://www.investopedia.com/terms/h/html.asp

jest - npm. (n.d.). Retrieved April 13, 2023, from https://www.npmjs.com/package/jest

Legion 5 Series | Gaming Laptops | Lenovo IE. (n.d.). Retrieved April 12, 2023, from https://www.lenovo.com/ie/en/c/laptops/legion-laptops/legion-5-series/?orgRef=https%253A%252F%252Fwww.google.com%252F

Node.js - How to get my external IP address in node.js app? - Stack Overflow. (n.d.). Retrieved April 7, 2023, from https://stackoverflow.com/questions/20273128/node-js-how-to-get-my-external-ip-address-in-node-js-app

nodemon - npm. (n.d.). Retrieved April 13, 2023, from https://www.npmjs.com/package/nodemon

Personal apps | Heroku. (n.d.). Retrieved April 12, 2023, from https://dashboard.heroku.com/apps

React. (n.d.). Retrieved April 12, 2023, from https://react.dev/

React Bootstrap pagination example (forked) - CodeSandbox. (n.d.). Retrieved April 11, 2023, from https://codesandbox.io/s/react-bootstrap-pagination-example-forked-nxci1?fontsize=14&hidenavigation=1&theme=dark&file=/src/App.css

React-Bootstrap · React-Bootstrap Documentation. (n.d.). Retrieved April 11, 2023, from https://react-bootstrap.github.io/components/buttons/

Software Engineering-FURPS - Best Online Tutorials | Source codes | Programming Languages. (n.d.). Retrieved December 12, 2022, from https://www.1000sourcecodes.com/2012/05/software-engineering-furps.html

Use JSDoc: Index. (n.d.). Retrieved April 12, 2023, from https://jsdoc.app/

What is an API? (n.d.). Retrieved April 12, 2023, from https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces

What is CSS? - Learn web development | MDN. (n.d.). Retrieved April 13, 2023, from https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS

What is JavaScript? - Learn web development | MDN. (n.d.). Retrieved April 13, 2023, from https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript