COMP4801

Final Year Project for Computer Science

The University of Hong Kong

Topic:

FYP18003

Blockchain and Smart Contract Application

Interim Report

LAU Siu Ming, Alex

YAM Mei Ki, Vanessa

Abstract

Blockchain has been one of the most trending areas among the field of technology in recent years. In particularly the financial industry has taken the advantage of such distributed ledger technology in creation of cryptocurrency and broadening of a new sector of the digital market. However, blockchain with its unique characteristic of immutability and transparency can provide much more potential usage in other industries. This project makes use of blockchain's security and traceability to create a platform that can better protect copyright and fair use of media by providing secure storage and usage tracking. This report iterates the project background and objective, and provides an assessment of the progress to date. There are also results from completed milestones, such as justified engineering decision on the development of web application; conceptual class model that supports the design of blockchain and use cases; user interface design that depicts the anticipated product; and an overview of the implementation of a preliminary demo.

Discussion of the difficulties encountered in the current stage is the smart contract construction in blockchain after the use case and system architecture design. The remaining work will be on the implementation and testing of the demo, thus indicating an ideal progress for demo implementation in the coming month.

Acknowledgment

We would like to express our greatest gratitude here to anyone who has helped me in completing this report, especially our supervisor Dr. Yuen for his guidance on the blockchain API and the system architecture of the project. Furthermore, a thank you to our english lecturer Cezar, who has given me a lot of advice and coaching on professional writing.

Table of Contents

1. **Overview** **4**
    1.1 Project Background 4
    1.2 Project Objective 5
    1.3 Project Features 5
    1.4 Report Outline 6
2. **Progress to date** **7**
3. **Methodology and Findings** **8**
    3.1 Blockchain design 8
    3.2 Website framework 9
    3.3 Cloud service 10
4. **Work Accomplished** **12**
    4.1 Design and Analysis 12
        4.1.1 Interface design 12
        4.1.2 System Architecture Design 15
        4.1.3 Use Case Design 17
    4.2 Demo building 19
        4.2.1 Environment and settings 20
        4.2.2 Features 20
5. **Difficulties encountered** **24**
    5.1 Use Case design 25
    5.2 System Architecture design 25
    5.3 Smart contract demo building 25
    5.4 Tracking feature implementation 25
6. **Future work** **25**
    6.1 Demo implementation 25
    6.2 Testing cloud service environment 26
7. **Conclusion** **27**

1. Overview

1.1. Project Background

Throughout the last decade, social media platforms have rapidly emerged and became a crucial part of our lives. People rely on Facebook in making connections, Instagram in sharing of photos and videos, Twitter in spreading of news… All of these platforms with their popularities share a common trait — centralization. Content that one shared, regardless of the effort one put in, ultimately is being ran and managed by a centralized authority. Thus it is often difficult for people that are used to such scheme of platform to question their ownership rights in what they have contributed, making concept of ownership on the internet unconceivable. Consequently, the nonexistence of concept in property right on the internet also leads to a lack of protection on original work or ideas. Even if one wishes to trace back to the author of a content, it is difficult with the ever-changing nature of the internet.

There is no guarantee in ownership nor originality to the content that one shared in the current media sharing platform, yet blockchain technology can change this. With the development of Bitcoin in 2008, increasing number of blockchains is being built and utilised in different aspects, such as finance and game industry. It introduces a relatively less familiar concept to the public, a transparent and decentralised way of record storing.

With the advantages of this technology, the project proposes a method for implementing blockchain technology into a media sharing platform where the record of uploading and sharing history is transparent to the users and they can trace the original work from a derivative work. It helps to issue the problems of digital sharing while providing a platform to prove the originality of an idea.

1.2. Project Objective

This project aims to use blockchain technology to implement a media sharing platform that enables a decentralised storage place for users to share their artworks. The scope of this project is to develop a web-based platform with its database supported by a hybrid approach of centralised cloud storage and existing blockchain architecture. The goal is to facilitate blockchain technology in the creative industry. By taking advantage of blockchain's immutability, eventual consistency and transparency, this project can drive the art market into a new era of online ownership. Providing reliable trading with cryptocurrency and a sharing platform that is protected by trustworthiness and security granted from the blockchain technology for artists worldwide.

1.3. Project Features

This project implements a web application using blockchain technology to achieve eight main features:

FE1: For users to create accounts on the website.

FE2: For users to manage their Profiles that are being displayed to others.

FE3: For users to upload artwork on to the website.

FE4: For users to downloading previous artwork and publish a derivative.

FE5: For users to be able to track the derivative and origin of an artwork.

FE6: For users to have the option in pricing their artwork.

FE7: For users to exchange ideas by giving comments.

FE8: For users to get tokens for giving contribution, such as artwork publishing and commenting in the community.

The scope is divided in two parts. One as the scope of this project, and the other one as a scope that are being hoped to establish in the future. Table 1 illustrates the two scopes in regards to each feature. FE1, FE5 and FE6 will be fully implemented in this project therefore do not have further development in scope of subsequent releases. FE2, FE3, FE4, FE7 and FE8 will be partially implemented in the initial scope and be extended in the future.

Features	Scope of this FYP	Scope of subsequent releases
FE1: Account creation	Fully implemented	-
FE2: Profile Management	Simple modification of user information should be included	To include more information that is useful for enhancing the transperancy of the community
FE3: Artwork uploading	Only limited to imagery form of work, such as file format of JPEG, PNG, BMP...etc	To be extended to other formats of work, such as text and video
FE4: Artwork downloading	Fully implemented for downloading the uploaded image one by one	To be extended to download multiple artworks at once for convinience
FE5: Artwork tracking	Fully implemented	-
FE6: Artwork selling	Fully implemented	-
FE7: Commenting	Only available under each individual artwork	Available under user profile, with option of private or public
FE8: Token generating	Token is represented as an internal point system	Token can be exchanged into crytocurrency

Table 1. Scope of features

1.4. Report Outline

This report is sectioned into progress demonstration, evaluation and conclusion. Chapter 1 demonstrates the background and overview of the project, Chapter 2 then gives an overview of the progress with timeline of each milestone. In chapter 3, methodology is explained regarding the design of website, blockchain architecture, and data storage with detailed justifications and findings. The work accomplished is presented in Chapter 4, including the design of user interface, blockchain architecture, use cases design and the implementation of the demo. Chapter 5 illustrates the difficulties encountered as well as limitation discover while designing the architecture of the application. A conclusion is given in chapter 6 by summarising the progress of the project and discussing the remaining work.

2. Progress to date

A tentative timeline of the project is demonstrated in Table 2, with progress of each milestone being represented in percentage. Requirement Specification and User Interface Design are both completed, with requirement specification presented in form of the Project Plan and User Interface Design being shown in Chapter 3.2. At this stage, the team is focusing on constructing smart contract and implementing front end of the website, the progress is being caught up with the completion of Design and Analysis.

Prog-ress	#	Milestones	Sep 18	Oct 18	Nov 18	Dec 18	Jan 19	Feb 19	Mar 19	Apr 19
Requirement										
100%	M1	Background research	▨							
100%	M2	Requirement Specification	▨							
Design and Analysis										
85%	M3	Use Case design		▨						
100%	M4	System architecture design		▨	▨					
100%	M5	Cloud service analysis		▨	▨					
100%	M6	User Interface design			▨					
Implementation										
35%	M7	Smart Contract building				▨	▨	▨	▨	
10%	M8	Front-end implementation				▨	▨	▨	▨	
0%	M9	Cloud services intergration						▨	▨	
0%	M10	System intergration						▨	▨	
Testing										
0%	M11	Unit testing							▨	
0%	M12	Integration & System testing							▨	▨

Table 2. Scheduled milestones and progress

3. Project Methodology and Findings

The following section gives a detailed justification on the decision made regarding each essential aspect that constitutes the project, including choice of Ethereum blockchain, the framework of the Single Page Application and Azure cloud service.

3.1. Blockchain design

In order to process data using blockchain, it is necessary to select an approach that is most suitable to our need among various types of blockchain that are available in the market. Ethereum is used to facilitate the back-end development of this project, for its already well established cryptocurrency — Ether, and its better supported flexibility.

Comparison of Ethereum with other blockchains

Comparing Ethereum to other popular blockchains such as Bitcoin, it offers a better degree of customization to users via Smart Contracts. While Bitcoin was built with predefined operations that support transaction type of activities, Ethereum was built as "*a meta-protocol on top of Bitcoin*"[1], with its Smart Contract allowing storage of any executable codes. This type of blockchain can secure not just values or records but any type of functionalities, such as storing of image and interaction with users info that are required in this project. While there are also other built blockchains, such as Stratis and EOS, that can support similar flexibility and offer a higher transaction rate, Ethereum has been the most popular platform to be used to build decentralized applications, with the availability of Ethereum Virtual Machine that offers powerful computing power and adaptability to different programming languages that better supports the construction of this project. Furthermore, such popularity also makes its cryptocurrency — Ether widely adopted, hence beneficial to users that wish to use this platform as a market to generate financial value from their artworks.

Comparison of Ethereum with self-build blockchain

While option of building an original blockchain can offer tailor-made functionality for the project without any restriction of existing cryptocurrency as well as a faster transaction rate, these features are considered less crucial as compared to the advantages from using Ethereum. Even though building our own blockchain would eliminate the restriction of significant low transaction rate from Ethereum, it would require extra effort in implementing the wallet and mining technology that are already extensively developed in Ethereum. Taking into account that this application need not be real-time to be able to operate, the benefits of a well-established platform from Ethereum outweighs the advantage of performance and flexibility from a customized blockchain.

3.2. Website framework

For ease of navigation and to provide a smooth experience to the users, the project will implement a Single Page Application as it web-based application. As compared to Multiple Page Application, Single Page Application can eliminate the hassle in page loading by refreshing only section of the content that the users require. It also enhances the interaction between users and website by being responsive.

Areas of concern

Numerous frameworks are available for developing the front-end, where user interacts; the back-end, where the website operates and functions; and the database, where data utilised by the website is stored and retrieved from. Since implementation of blockchain will not restrict the choice of website development, the project will implement with frameworks that gives the most outstanding user experience and that the team is most comfortable working with.

Front-end development

For front-end implementation, AngularJS and ReactJS were both taken into consideration for their popularities and well supported resources in developing Single Page Application. More findings reveal that ReactJS is a better option, for its greater degree of interactiveness that enhances user experience and its easier integration to different platform than AngularJS that make it a more suitable framework for achieving the project’s ultimate goal of implementation on multi-platforms.

Data storage management

For database construction, as opposed to fully operate using blockchain, the project will take a hybrid approach by using both blockchain, database and cloud service. Such approach is to minimize the cost and processing time in storing potentially large image. Storing image is different to typical data storage on blockchain such as plain text and string, as it requires a pixel by pixel data record and thus will be extremely expensive in processing time if done entirely using blockchain. The hybrid approach will make use of the cloud service for storing the actual image and the blockchain for storing a hashed linked which can be used to verify the authenticity of the image on cloud storage server. The user details and metadata will be stored in the database. The approach can share the storage to improve the efficiency on the data processing and ensure a high level of security and traceability from blockchain while solving the issue of a costly storage occupation.

Back-end development

For back-end development, a decision is yet to be finalised for the vast amount of available frameworks that can fit into other determined sectors of the web development including ReactJS and hybrid storage approach. A JavaScript framework such as Node.js will likely be utilised since the front-end development by ReactJS also makes use of JavaScript. The unification of programming languages across different sectors of the development can provide a steady learning curve to the team.

3.3. Cloud service

Cloud services are investigated with a few criteria that are crucial to this projects, one is its capacity for storing images that are uploaded to the website, and the other is the functionality of an image recognition service that can facilitate filtering of inappropriate photos or images that might have invaded copyright when uploaded by users. Table 3 illustrate the details of considered cloud services and gives an comparison in areas of usage, storage service and image recognition service.

Service provider	Google [2]	Amazon Web Services [3]	Microsoft Azure [4]

Storage Service	CloudSQL	Cloud Storage	Amazon Elastic File System	File Storage
Usage	Provide web framwork and content manage -ment	Storage for multimedia and blob objects	Content management and web serving, as well as database backup	Enable usage of Windows and Representational State Transfer Application programming interface which is useful for web development
Image Recog -nition	Cloud Vision		Amazon Rekognition	Computer Vision

Table 3. Cloud service analysis

In this project, Azure cloud from Microsoft is selected since it provides a trial version of cloud storage and it is sufficient for storing image files. It is convenient and feasible among the choices which provides an integrated environment for testing, developing and deploying application. Since image filtering function is not the most main objective in the project, a simple cloud storage platform during testing phase is more suitable for simplifying the process without having any extra costs.

4. Work Accomplished

This chapter demonstrates the work accomplished at the current stage and gives an overview result from each ongoing or completed milestones. The chapter is divided in two sections, one in discussing the milestones completed in the Design and Analysis phase; and the other in discussing the demo implementation.

4.1. Design and Analysis

4.1.1. Interface design

The following are the interface design for the application and it is used to visualize the ideas of the content sharing platform which becomes the reference for building of the website. The website development thus will be heavily relied on the blueprint of the following figure 1 to 6.

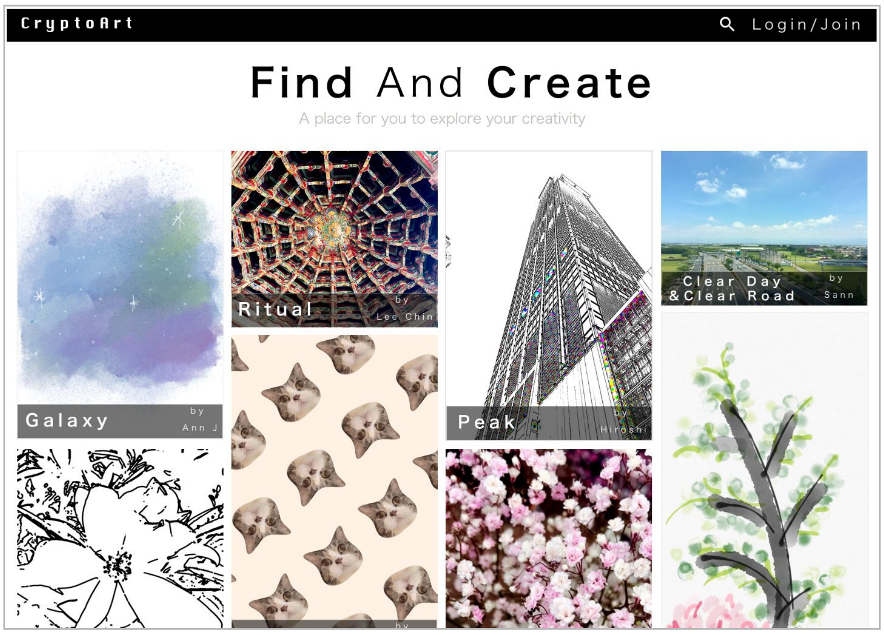
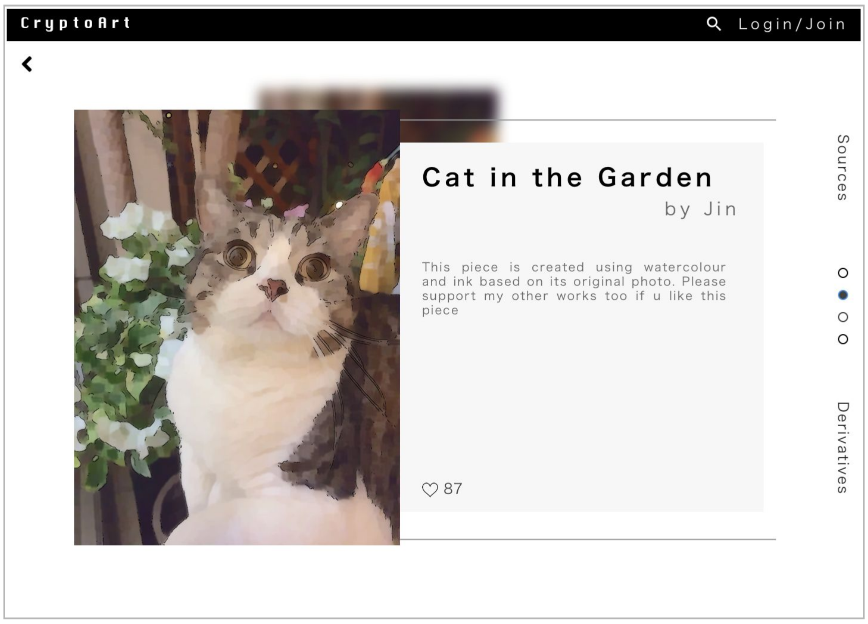


Figure 1. Front page of the website

The user interface make uses of a minimalist design, so that it can provide a clean layout for users to focus on the artworks rather than the website itself. Figure 1 illustrates the interface to the front page of the website, where popular artworks are displayed with an infinite scrolling that allow users to view more artworks by scrolling to the bottom of the page.



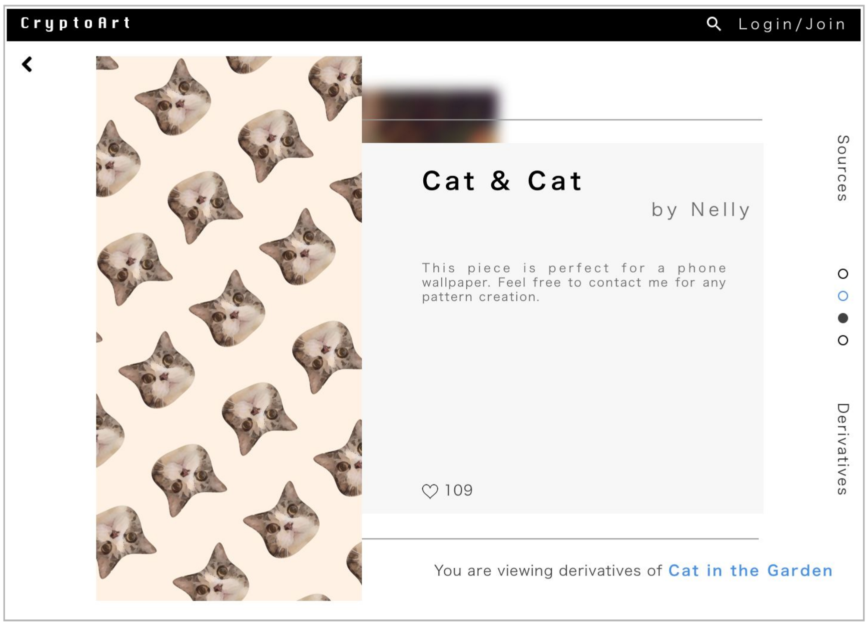
Cat in the Garden

by Jin

This piece is created using watercolour and ink based on its original photo. Please support my other works too if u like this piece

♡ 87

Figure 2. Artwork portfolio



Cat & Cat

by Nelly

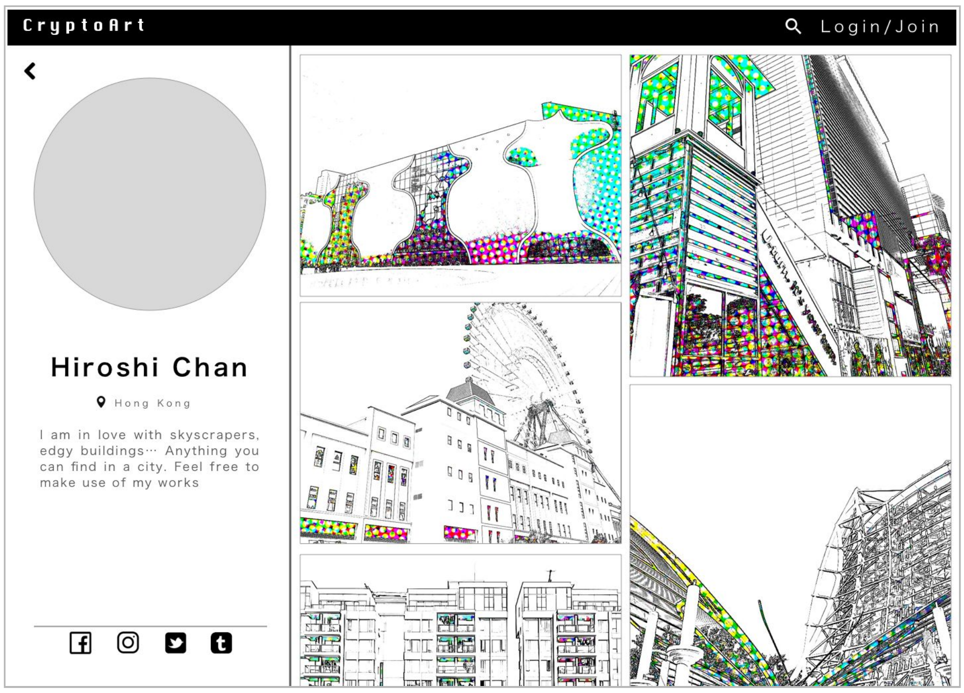
This piece is perfect for a phone wallpaper. Feel free to contact me for any pattern creation.

♡ 109

You are viewing derivatives of Cat in the Garden

Figure 3. Viewing derivative of an artwork

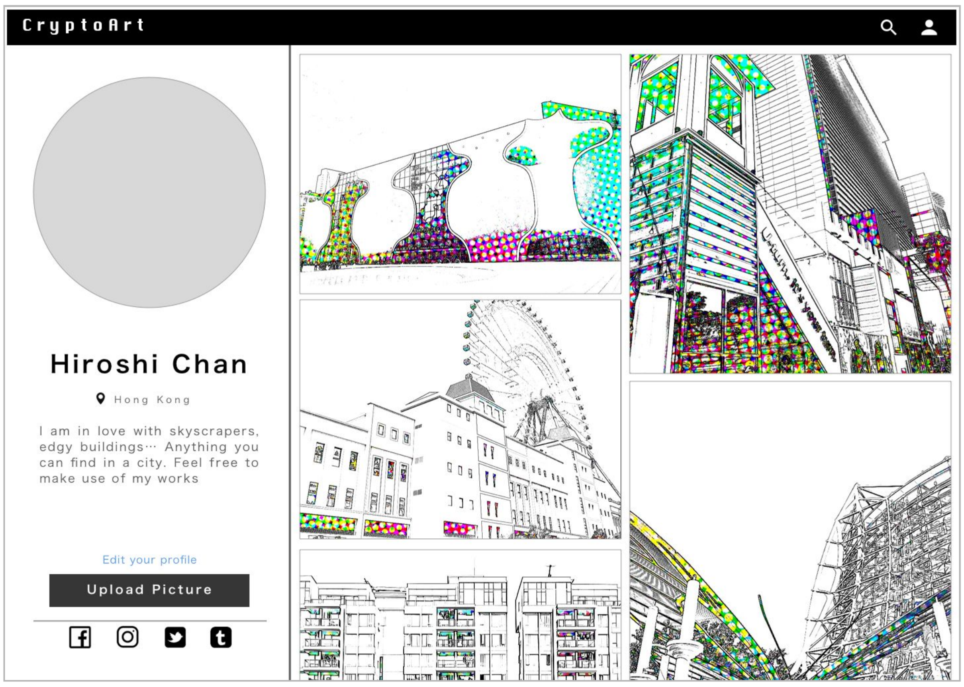
Figure 2 and Figure 3 display the web pages design for viewing an artwork and its derivative works, which is a key feature of the web application. The artwork portfolio contains artwork with its name, description and creator, as well as its popularity presented as the number of "like" given by other users. The page also contains a page-bar in form of dots at the side for navigation between sources and derivatives regarding to the selected artwork. Users can view the entire chain to know the evolution from the root of the artwork to its derivative works. Figure 2 gives an example of portfolio for a selected artwork, and Figure 3 shows an illustration for viewing a derivative portfolio.



I am in love with skyscrapers, edgy buildings… Anything you can find in a city. Feel free to make use of my works

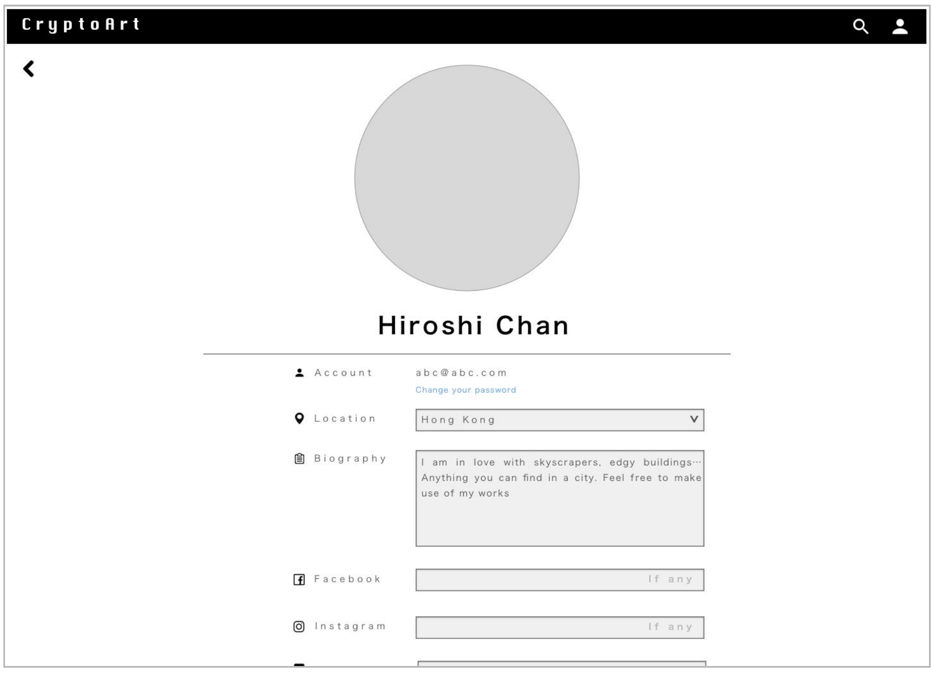
.

Figure 4. Artist portfolio



I am in love with skyscrapers, edgy buildings… Anything you can find in a city. Feel free to make use of my works

Figure 5. Viewing self portfolio



Hiroshi Chan

Account	abc@abc.com Change your password
Location	Hong Kong
Biography	I am in love with skyscrapers, edgy buildings… Anything you can find in a city. Feel free to make use of my works
Facebook	If any
Instagram	If any

Figure 6. Profile Editing

Figure 4 and Figure 5 illustrate the webpage layout for viewing artist portfolio. The portfolio contains artworks and biography of artist, as well as different social means for connecting with the artist. The two figures are different in terms of viewer of the page. Figure 4 is a portfolio page in perspective of other users whereas Figure 5 is in view of the artist himself with options to upload artwork or edit profile. Once the artist presses the edit profile link, he will be directed to the edit page as shown in Figure 6, in which he can change or his personal details except his username.

4.1.2. System Architecture Design

In order to model blockchain architecture and simulate its interactions with the web application and cloud service, a conceptual class model is designed as shown in Figure 7. The diagram illustrates the responsibility of each class and the relationship between different classes across systems. MongoDB holds the classes exist in the web application database, including user details and the access key to artwork information in form of hashing of the user and image ID; Ethereum holds the uRecord class in blockchain that represents the uploaded artwork record and stores artwork metadata including the access link; Cloud contains the image uploaded on cloud service; and MetaMask refers to an external wallet that is connected to the users.

The responsibility of each class is designed in a way so that it aids the implementation of the back-end system of the application as well as efficiently minimises the data processing cost on the blockchain. For example, to avoid data redundancy while verifying the data integrity of each artwork, the Artwork class in MongoDB does not store any artwork metadata but only the key to uRecord class. In such way the integrity of each artwork is automatically protected by Ethereum and does not require cross checking to verify data on either side.

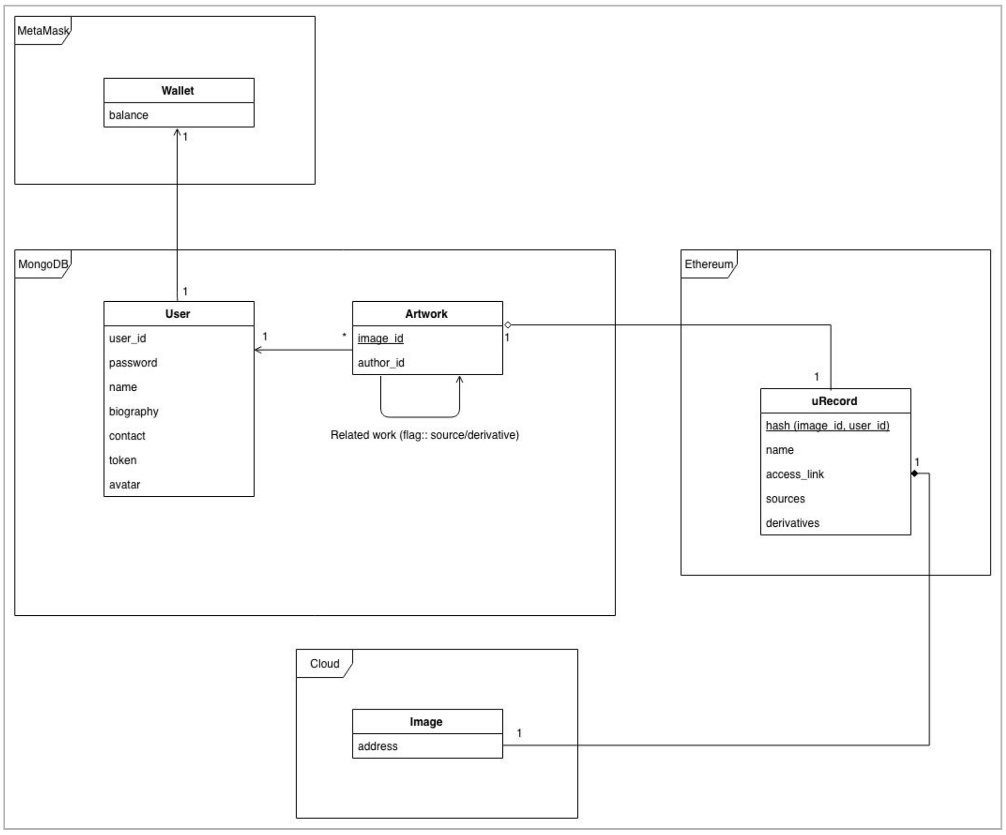


Figure 7. Conceptual Class Diagram of the web application with blockchain and cloud service

4.1.3. Use Cases Design

Use case design simulates the interaction between different system for each use case that involved the users and is represented in a System sequence diagram. Figure 8 illustrates the System sequence diagram for a user to upload an image to the web application. It initiates with user interacting with the front-end, which then sends an HTTP request to the back-end and being handled by calling corresponding service providers from cloud service, web application database on MongoDB, and blockchain on Ethereum.

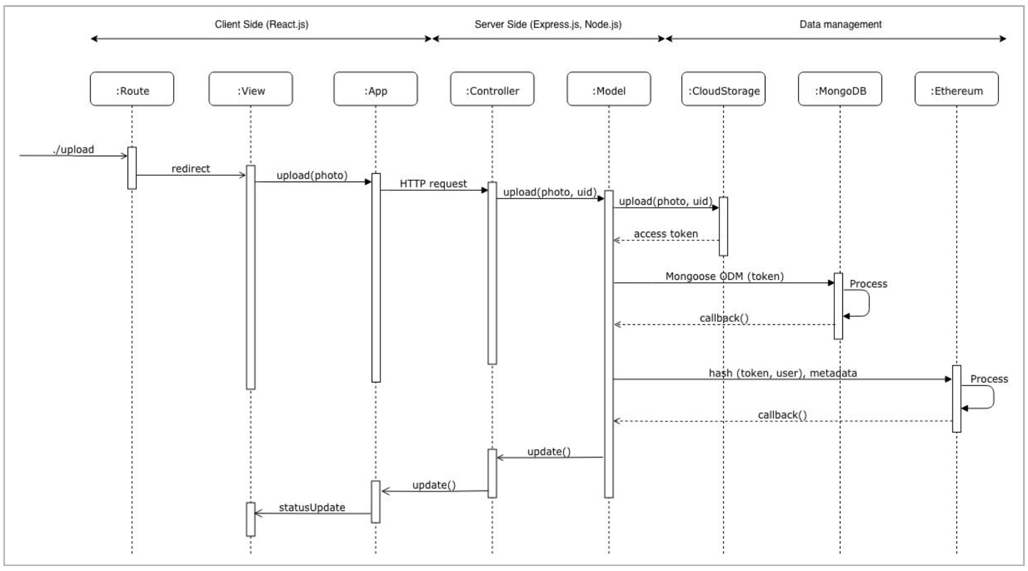


Figure 8. System sequence diagram (SSD) for uploading an image

Figure 9 demonstrates the System sequence diagrams for searching artworks by specific author and Figure 10 illustrates the tracking of a given artwork.
Both use cases initiates with a similar line of interactions that forwards the HTTP request to the back-end and queries the MongoDB for user id (uid) of user or artwork id (aid) of artwork accordingly. The uid and aid together will be used as a key to retrieve artworks from Ethereum. For case of searching artwork by specific user, artworks are queried in a recursive calling to Ethereum; while for case of artwork tracking, a query of related work is first called to Ethereum, and followed by a recursive call to retrieve related artworks while verify the integrity of each related artwork.

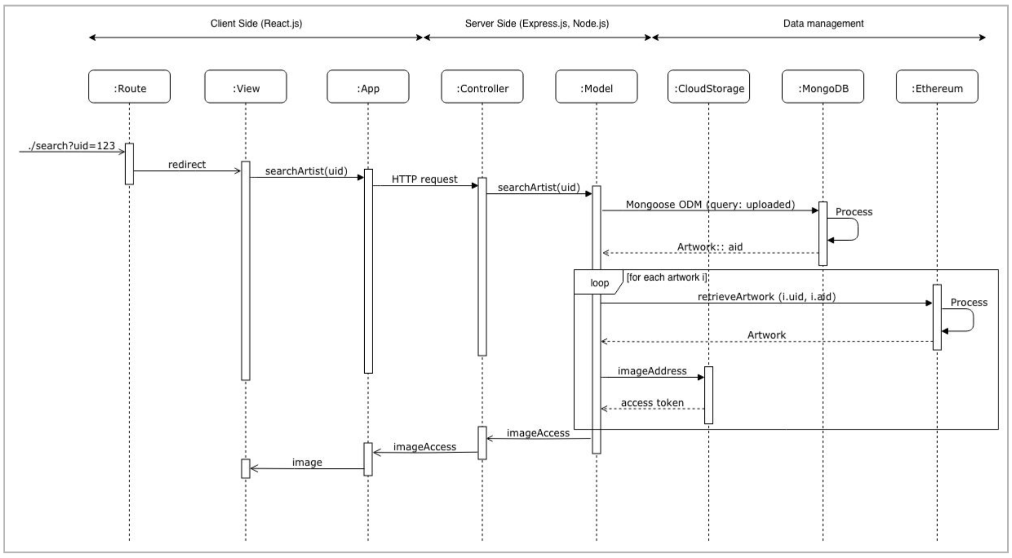


Figure 9. System sequence diagram (SSD) for view artworks by specific artist

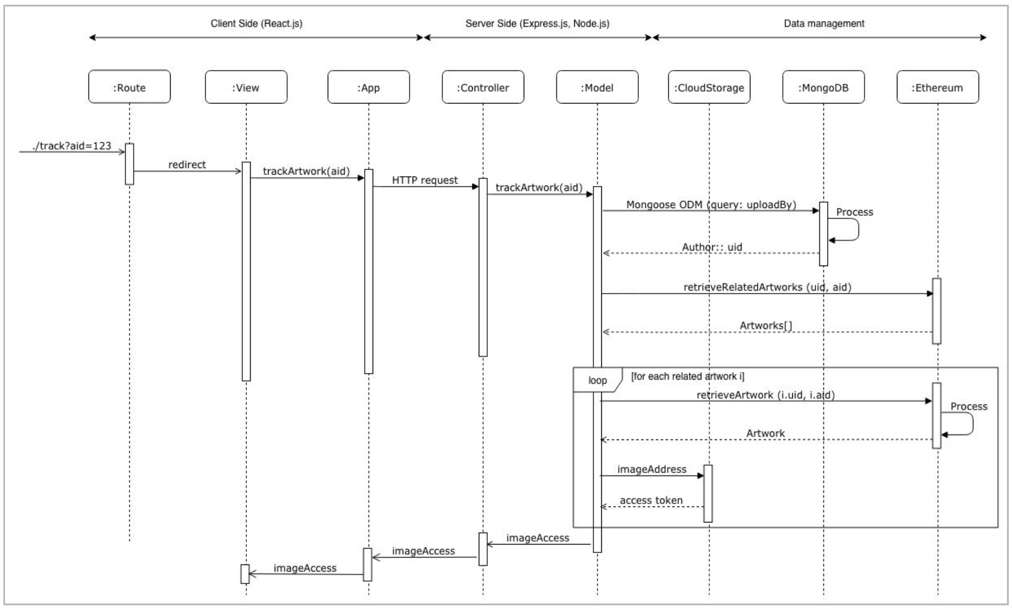


Figure 10. System sequence diagram (SSD) for tracking an artwork

4.2. Demo implementation

The demo implementation aims to develop a simple web application using smart contracts written in Solidity that serves as a proof of concept in demonstrating the feasibility of the current design and methodology. It shall also act as a bridge between the Design and Analysis phase and the Implementation phase for which the building of the smart contract can be based on the system diagram from Figure 8 to 10.

4.2.1. Environment and settings

Different frameworks and libraries were utilised in the building of the demo application. Smart contracts are written in Solidity in accordance to the decided methodology. The application also makes use of tools provided by the Truffle suite in managing the smart contract, including Truffle framework that compiles and builds contract; and Ganache, a local private Ethereum blockchain for which the contracts can be deployed on. Node JS is also used for the server environment as addressed in chapter 3.2, while web3.js is used to manage the communication between web application and Ethereum.

Since the team has no experience on building decentralised application, online tutorials were used as a reference to initiate the demo implementation, in particularly the tutorial from the Truffle Suite [5] on which the project has its source codes based on.

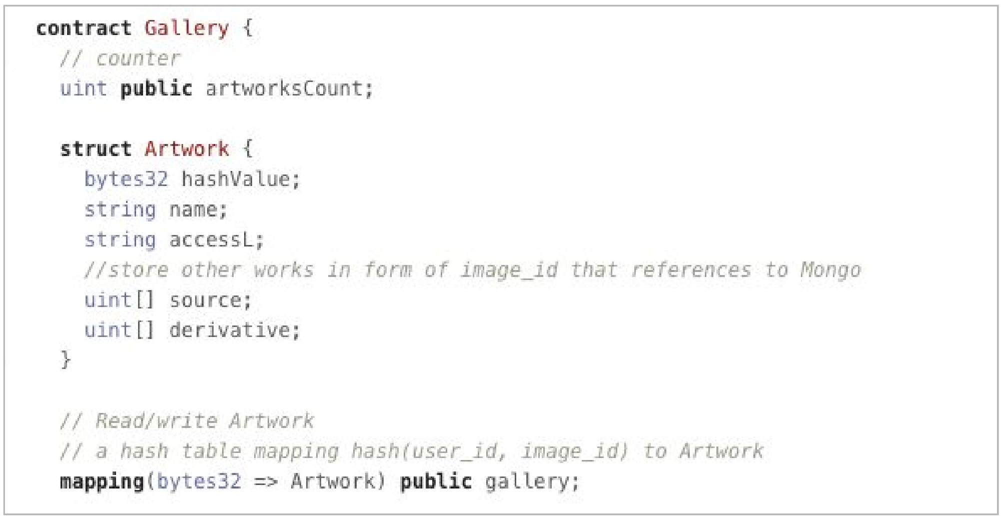
4.2.2. Features

Since the demo application serves as a demonstration of feasibility, the focus is placed on the function of smart contract and its communication to the web application. Thus the front-end is written in plain HTML and JavaScript, and serves as a primitive render of data. Cloud service is also not concerned in this stage, such that for this demo application the image addresses are addresses pointing to local storage.

Smart Contract

At the current stage, the team found that a single contract, *Gallery*, is sufficient in storing all artwork records and handling all CRUD operations.

As seen in Figure 10, Artwork records are represented in form of a struct, consisting of its metadata and storing sources and derivatives in form of an array of their image id. The artwork is then mapped to key of user id and image id by a mapping, unique data structure in Solidity which is indeed a hash table . Figure 11 also shows that functions for basic database operation have been written and are available to the web application, for example creating an Artwork instance, retrieving Artwork information from key of user id and image id, and adding a source.

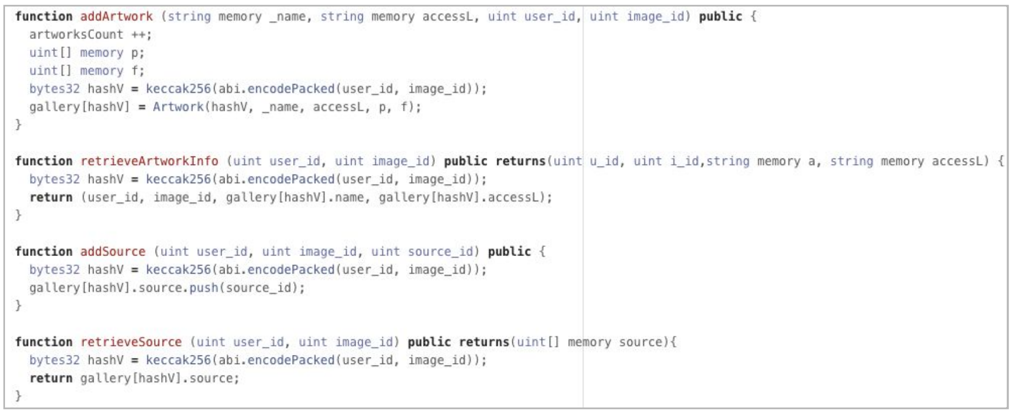


```
contract Gallery {
  // counter
  uint public artworksCount;

  struct Artwork {
    bytes32 hashValue;
    string name;
    string accessL;
    //store other works in form of image_id that references to Mongo
    uint[] source;
    uint[] derivative;
  }

  // Read/write Artwork
  // a hash table mapping hash(user_id, image_id) to Artwork
  mapping(bytes32 => Artwork) public gallery;
```

Figure 11. Snapshot of the code in smart contract – variables, from *Gallery.sol*



```
function addArtwork (string memory _name, string memory accessL, uint user_id, uint image_id) public {
  artworksCount ++;
  uint[] memory p;
  uint[] memory f;
  bytes32 hashV = keccak256(abi.encodePacked(user_id, image_id));
  gallery[hashV] = Artwork(hashV, _name, accessL, p, f);
}

function retrieveArtworkInfo (uint user_id, uint image_id) public returns(uint u_id, uint i_id,string memory a, string memory accessL) {
  bytes32 hashV = keccak256(abi.encodePacked(user_id, image_id));
  return (user_id, image_id, gallery[hashV].name, gallery[hashV].accessL);
}

function addSource (uint user_id, uint image_id, uint source_id) public {
  bytes32 hashV = keccak256(abi.encodePacked(user_id, image_id));
  gallery[hashV].source.push(source_id);
}

function retrieveSource (uint user_id, uint image_id) public returns(uint[] memory source){
  bytes32 hashV = keccak256(abi.encodePacked(user_id, image_id));
  return gallery[hashV].source;
}
```

Figure 12. Snapshot of the code in smart contract – functions, from *Gallery.sol*

Web applicatioin

By calling functions from the built contracts, the website is able to display existing artworks, retrieve artwork information as well as display sources and derivatives. Figure 12 displays the user interface to the front page, on which list of artworks are displayed and is responsible by the section of code as illustrated in Figure 13.

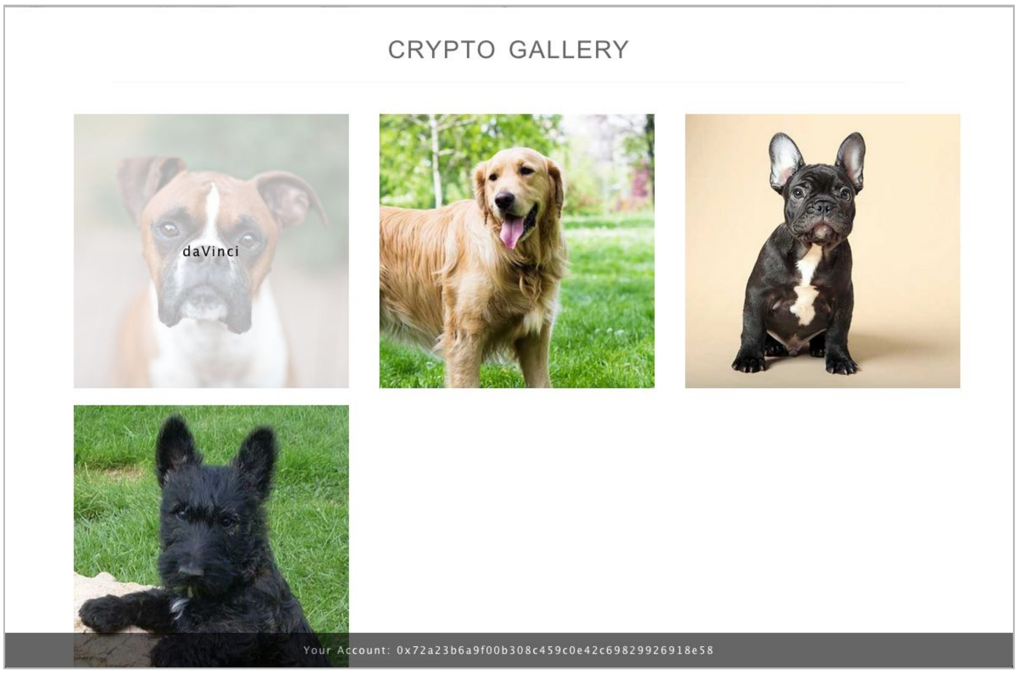
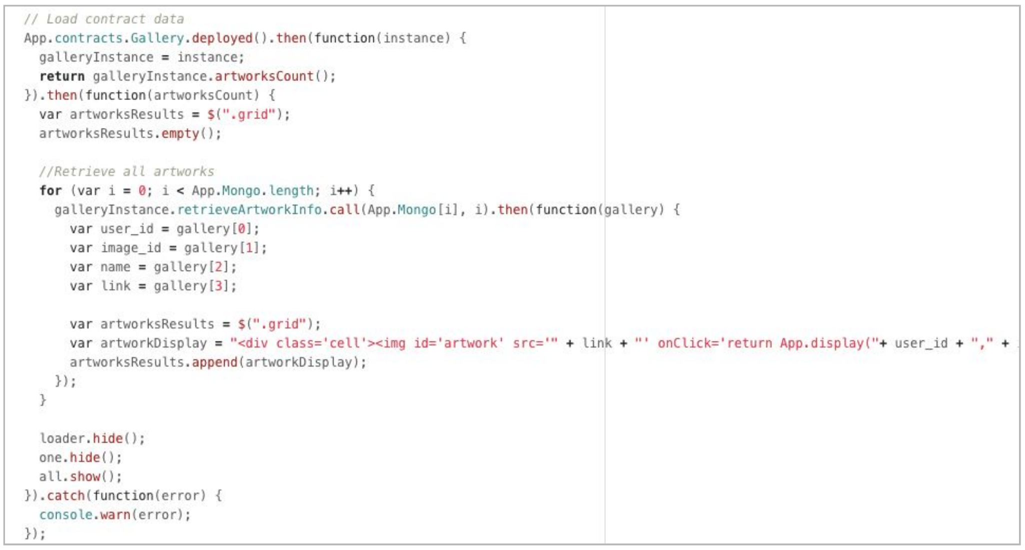


Figure 13. Frontpage of the web application



```
// Load contract data
App.contracts.Gallery.deployed().then(function(instance) {
  galleryInstance = instance;
  return galleryInstance.artworksCount();
}).then(function(artworksCount) {
  var artworksResults = $(".grid");
  artworksResults.empty();

  //Retrieve all artworks
  for (var i = 0; i < App.Mongo.length; i++) {
    galleryInstance.retrieveArtworkInfo.call(App.Mongo[i], i).then(function(gallery) {
      var user_id = gallery[0];
      var image_id = gallery[1];
      var name = gallery[2];
      var link = gallery[3];

      var artworksResults = $(".grid");
      var artworkDisplay = "<div class='cell'><img id='artwork' src='" + link + "' onClick='return App.display("+ user_id + "," +
      artworksResults.append(artworkDisplay);
    });
  }

  loader.hide();
  one.hide();
  all.show();
}).catch(function(error) {
  console.warn(error);
});
```

Figure 14. Code responsible for displaying all artwork from the smart contract, from *app.js*

Figure 14 illustrate the artwork portfolio page after user clicking on an image, in which artwork information is displayed together with its sources and derivatives, and Figure 15 demonstrates section of code that is in charged of the such function.

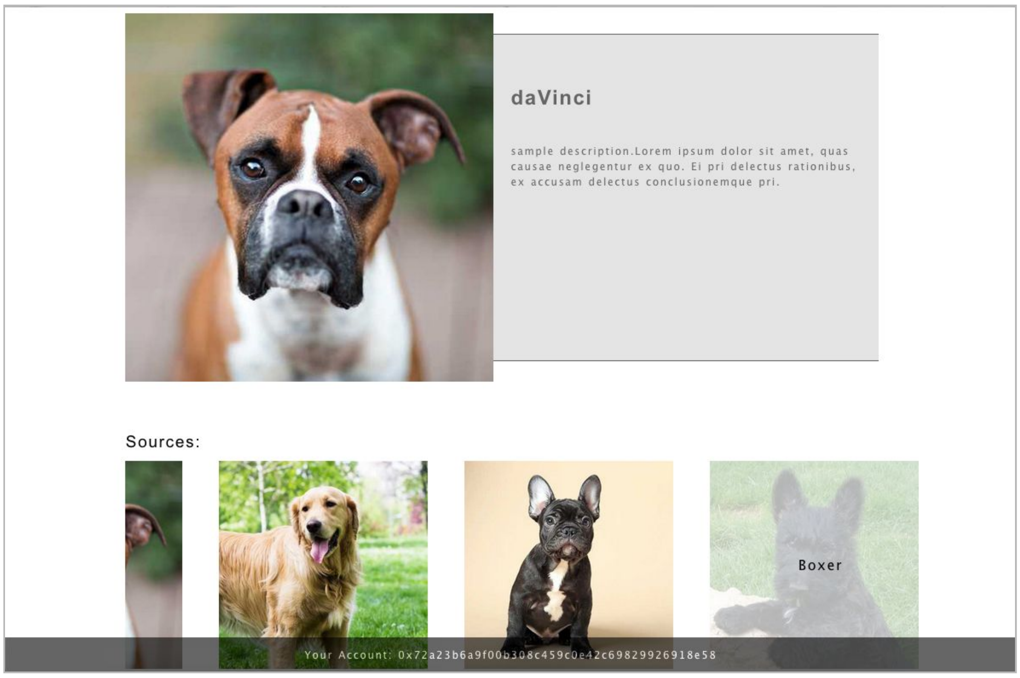
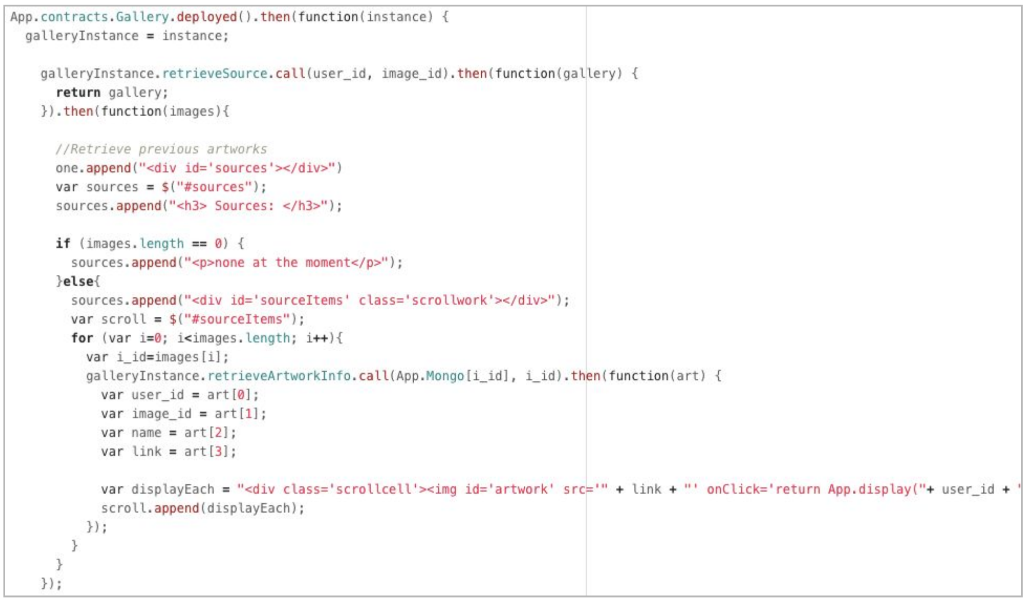

daVinci
sample description.Lorem ipsum dolor sit amet, quas causae neglegentur ex quo. Ei pri delectus rationibus, ex accusam delectus conclusionemque pri.
Sources:
Boxer
Your Account: 0x72a23b6a9f00b308c459c0e42c69829926918e58

Figure 15. Artwork portfolio of the web application



```
App.contracts.Gallery.deployed().then(function(instance) {
  galleryInstance = instance;

    galleryInstance.retrieveSource.call(user_id, image_id).then(function(gallery) {
      return gallery;
    }).then(function(images){

      //Retrieve previous artworks
      one.append("<div id='sources'></div>")
      var sources = $("#sources");
      sources.append("<h3> Sources: </h3>");

      if (images.length == 0) {
        sources.append("<p>none at the moment</p>");
      }else{
        sources.append("<div id='sourceItems' class='scrollwork'></div>");
        var scroll = $("#sourceItems");
        for (var i=0; i<images.length; i++){
          var i_id=images[i];
          galleryInstance.retrieveArtworkInfo.call(App.Mongo[i_id], i_id).then(function(art) {
            var user_id = art[0];
            var image_id = art[1];
            var name = art[2];
            var link = art[3];

            var displayEach = "<div class='scrollcell'><img id='artwork' src='" + link + "' onClick='return App.display("+ user_id + '
            scroll.append(displayEach);
          });
        }
      }
    });
```

Figure 16. A section of code that is responsible for displaying an artwork and its related works, from *app.js*

5. Difficulties encountered

Going through each milestone has given the team several difficulties to be resolved, especially in the part of design and analysis. Currently the team is facing most challenges from system design and demo implementation.

5.1. Use Case design

Use case design is one of the milestones that the team most struggle on. It also has the lowest completion rate as compared to the other scheduled milestones. Use case design is the procedure of simulating users' interactions with the system as well as interactions between different internal systems, and representing the models with System Sequence Diagram and User Case Diagram. So far the team has designed just one system sequence diagram for the case of uploading image, which is considered a slow progress by taking account into the fact that eight different user cases are required for the stage. Therefore this milestone is marked with a low completion rate of 20% as compared to other milestones. The slow progress is due to the nature that each use case takes an enormous amount of time to construct. Not only that the process of use case designing for a web application is complicated by the different elements involved from blockchain and cloud service, it is also a design that the implementation will heavily relies on such that it must be accurate and precise.

5.2. System Architecture design

Although the team was able to develop a conceptual class model that simulates the design of the system and kickstart the use case design process, there are still a few more designs required to complete the blockchain architecture design. In figure 7, the attributes have to be clarified during design stage. There were some issues on distributing them to the proper system. There were redundant attributes in more than one party. The analysis of the system architecture has to be performed in order to minimize the cost of implementation. Therefore, the discussion with the supervisor provided a clearer picture to design figure 7 by explaining the way of exchanging data between different parties. For example, using 'image_id' and 'user_id' to connect database and blockchain storage for facilitating the record searching. And Ethereum is the middle-man between cloud service and database for retrieving the actual image.

5.3. Smart contract demo building

The team also attempted to build a demo in order to understand the architecture of blockchain comprehensively. With online tutorials, building a simple web application and constructing a basic functioning smart contract in blockchain were successful, however challenges encountered when the team is trying to integrate the two parts into one single application. The solution is to examine different existing solutions used by other applications to relieve the difficulty and unfamiliarity in integrating the applications.

5.4. Tracking feature implementation

The tracking function is to show the related previous and successive works from the targeted image. Our team has been working on building the list of array to store the related work. The issue is not being able to retrieve the corresponding data and display on the demo website. The implementation involves with the interaction between javascript, html and smart contract structure. The proposed solution is to investigate the how the data pass through to the application and extract the correct variables to the front end. It is in progress for further testing.

6. Future work

This section is about the work to be done in the second semester. Following with Table 2, after finishing the Design and Analysis stage, the team will focus on the demo implementation as it is in progress with the support of the accomplished work. After finishing the demo, the enhancement is required to be done by integrating the front end and cloud service implementations.

6.1. Demo implementation

Since the demo is not fully implemented with the proposed functions, the team is going to build the application with the completed documents such as conceptual class diagram and system sequence diagram from Figure 7-10 to enhance completeness of the demo. Hence, the most recent objective is to implement the functions from the system sequence diagram from Figure 8 to 10. They are the foundation of the web application.

The main objective of building a demo is to demonstrate the functions if it is workable before constructing a complex structure of the application. It helps the project to foreseen any challenges and difficulties that can be fixed as soon as possible.

Moreover, the current stage is using a local environment to test the functionality of the demo. After the enhancement on the demo, the demo will be established to the test net of Ethereum for testing.

Besides the smart contract building part, the front end development is also in progress during developing the demo. Hence, the demo part is involved with front end website design and the back end blockchain implementation.

6.2. Testing cloud service environment

Since the current stage is using a local file for storing all the images for the demo, the cloud service uploading will be tested separately for preparing the implementation on the actual image uploading which will be in the Cloud service integration on February. The plan is to implement image uploading function for establishing the connection to the azure cloud. This approach will be processed in parallel with the demo implementation.

7. Conclusion

This report summarises the progress of this Blockchain and Smart Contract Application to date, with justified design and engineering choices; design of conceptual model; Use cases design; and illustration of user interface. Background research and findings support the implementation decision for developing a responsive Single Page Application with ReactJS as it front-end and hybrid approach of cloud storage, database and blockchain as its data storage. The user interface demonstrates the developing web application and simulate users' interactions with the website on different functionalities. It also serves as a solid foundation and reference for future front-end development to rely on. The conceptual class model also helps to partially understand the blockchain architecture and institutes the process of use case design. The completion of use cases aids the understanding of the flow of system operations.

As reflected by the progress appraisal, an application demo is under the development stage which is following the use case and system architecture design. Therefore the progress is considered moderately positive with comprehensive construction of the smart contract.

The next major step shall be on enhancing the demo application following with the designed architecture and interface design, also the cloud service testing.

List of Figures

1. Front page of the website 12

2. Artwork portfolio 13

3. Viewing derivative of an artwork 13

4. Artist portfolio 14

5. Viewing self portfolio 15

6. Profile Editing 15

7. Conceptual Class Diagram of the web application with blockchain and cloud service 16

8. System sequence diagram (SSD) for uploading an image 17

9. System sequence diagram (SSD) for view artworks by specific artist 18

10. System sequence diagram (SSD) for tracking an artwork 18

11. Snapshot of the smart contract – variables 20

12. Snapshot of the smart contract – functions 20

13. Frontpage of the web application 21

14. Code responsible for displaying all artwork from the smart contract, from app.js 21

15. Artwork portfolio of the web application 22

16. A section of code that is responsible for displaying an artwork and its related works, from *app.js* 22

List of Tables

1. Scope of features 5 - 6

2. Scheduled milestones and progress 7

3. Cloud service analysis 11

Reference

[1] Buterin V. A next-generation smart contract and decentralized application platform [Internet]. 2014 [cited 2018 Sep 28]; Available from: https://github.com/ethereum/wiki/wiki/White-Paper

[2] Google [Internet]. Cloud Storage Options; [cited 2018 Oct 20]. Available from: https://cloud.google.com/products/storage/

[3] Amazon [Internet]. Cloud Storage Services – Amazon Web Services (AWS); [cited 2018 Oct 20]. Available from: https://aws.amazon.com/products/storage/

[4] Microsoft Azure [Internet]. Azure Storage - Secure cloud storage; [cited 2018 Oct 20]. Available from: https://azure.microsoft.com/en-us/services/storage/

[5] Truffle Suite [Internet]. Ethereum Pet Shop – Your First Dapp; [cited 2018 Jan 20]. Available from: https://truffleframework.com/tutorials/pet-shop