Electronics and Computer Science
Faculty of Physical Sciences and Engineering
University of Southampton

Thomas Smith, tcs1g20
December 9, 2022

Using Blockchain for Video Game Distribution

Project Supervisor: Leonardo Aniello
Second Examiner: tbd

A project report submitted for the award of
BSc Computer Science

Abstract

Video game developers will often have to rely on third party platforms for the distribution of their games; this comes at a large monetary cost to the developer and leaves users at a greater risk of censorship and with weak digital ownership that is reliant on the platform staying active. This project uses the Ethereum blockchain to facilitate the large-scale distribution and continuous updating of video games that allows developers to directly interact with their users, who will now have true digital ownership.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

Acknowledgements

I would like to thank my supervisor, Leonardo Aniello, for his support throughout this project.

Contents

Chapter 1

Problem Statement

1.1 The Problem

Video games are often large and highly popular pieces of software that are typically distributed for developers by a third party platform like Steam or Epic Games. Whilst these platforms provide benefits such as availability, and some social features they have some major downsides that include:

(a) taking a large cut of all revenue,
Steam take a 30% cut

(b) being vulnerable to censorship from governments,
The Chinese version of Steam is heavily censored

(c) the user's access to their games is linked to the platform.
If the platform shuts down, the user loses all their games

A blockchain-based, decentralised platform will provide greater profits to developers, eliminate the need for trust in a third party platform, and allow users greater control over the games they own as their access is not directly linked to one service.

1.2 Goals

The goal of this project is to implement a large-scale distribution platform that will allow game developers to release and continuously update their games on a public network by directly interacting with their users. The design should include:

- how data is shared between nodes in the network,
- how downloaded data can be verified using the network,
- how games are uploaded and updated,
- how users can be incentivised by developers to help distribute their games,
- how users can prove their contribution,
- how users purchase games,
- how users can prove they have purchased a game.

1.3 Scope

UNFINISHED

This project will look at how the Ethereum blockchain and smart contracts can be used to create a large-scale distribution platform for video games. This be deployed to a 'testnet', where Ether has no value and applications can be tested in a live environment. Whilst the focus on this application is on video games, this should be a valid solution for software of all types.

The application itself will consist of a set of smart contracts, written in Solidity, with tests and a basic GUI to be written using TypeScript and React.js. Tools like Ganache and MetaMask may also be useful for development.

Chapter 2

Background Research

2.1 BitTorrent

BitTorrent [4, 10] is the most popular p2p file-sharing platform, in which users will barter for chunks of files by downloading and uploading them in a tit-for-tat fashion, such that peers with a high upload rate will typically also have a high download rate. It is estimated that tens of millions of users use BitTorrent every day [13].

Download Protocol

For a user to download data from BitTorrent they would:

1. Find the corresponding .torrent file that contains metadata about the torrent.
2. The user will find peers, using a tracker identified in the .torrent, that are also interested in that content and will establish connections with them.
3. The user will download blocks [1], from peers, based upon the following priority:
   (a) **Strict Priority** Data is split into pieces and sub-pieces with the aim that once a given sub-piece is requested then all of the other sub-pieces in the same piece are requested
   (b) **Rarest First** Aims to download the piece that the fewest peers have to increase supply.
   (c) **Random First Piece** When a peer has no pieces, it will try to get one as soon as possible to be able to contribute.
4. The node will continuously upload blocks it has while active.

Availability

It is commonly suggested that availability of torrents is the biggest issue surrounding BitTorrent as *'38% of torrents become unavailable in the first month'* [4] and that *'the majority of users disconnect from the network within a few hours after the download has finished'* [10]. This paper [9] looks at how the use of multiple trackers for the same content and DHTs can be used to boost availability.

[1] nodes may reject downloads without the user providing data themselves in a tit-for-tat fashion

2.2 Ethereum

Ethereum is a Turing-complete, distributed, transaction-based blockchain that allows the deployment of decentralized applications through the use of smart contracts. Ether is the currency used on Ethereum and can be traded between accounts and is used to execute smart contract code on the network.

Smart Contracts

A smart contract is an executable piece of code, written in Solidity, that will automatically execute on every node in the Ethereum network when certain conditions are met. Smart contracts are enforced by the blockchain network and remove the need for intermediaries and reduce the potential of contractual disputes.

Gas is used to measure the computational effort of running a smart contract and must be paid before being processed and added to the blockchain. This helps prevent DoS attacks and provides economic incentives for users to behave in a way that benefits the whole network.

Example Use Cases

Some examples of applications that can be deployed to the Ethereum network are:

- Financial applications, such as decentralised exchanges and payment systems,
- supply chain management and tracking,
- voting and governance systems,
- unique digital asset systems, and
- data storage and sharing platforms.

Chapter 3

Literature Review

3.1 Blockchain-Based Cloud Storage

Blockchain technology can be leveraged for distributed cloud storage to provide both public and private storage. In table 3.1, I detail some examples of how blockchain has been used to create cloud storage platforms or supplement existing ones:

One gap found when researching these solutions was that few offered file versioning that would allow a user to view previous versions of uploaded data. File versioning is an essential part of my application and it will be likely that users will have varying versions of the same game.

Paper	Description of Solution
Blockchain Based Data Integrity Verification in P2P Cloud Storage [15]	This paper uses Merkle trees to help verify the integrity of data within a P2P blockchain cloud storage network as well as looking at how different structures of Merkle trees effect the performance of the system.
Deduplication with Blockchain for Secure Cloud Storage [6]	This paper describes a deduplication scheme that uses the blockchain to record storage information and distribute files to multiple servers. This is implemented as a set of smart contracts.
Block-secure: Blockchain based scheme for secure P2P cloud storage [5]	A distributed cloud system in which users divide their own data into encrypted chunks and upload those chunks randomly into the blockchain, P2P network.
Blockchain-Based Medical Records Secure Storage and Medical Service Framework [2]	Describes a secure and immutable storage scheme to manage personal medical records as well as a service framework to allow for the sharing of these records.
A Blockchain-Based Framework for Data Sharing With Fine-Grained Access Control in Decentralized Storage Systems [14]	This solution uses IPFS, Ethereum and ABE technology to provide distributed cloud storage with an access rights management system using secret keys distributed by the data owner.

Blockchain based Proxy Re-Encryption Scheme for Secure IoT Data Sharing [7]	An IoT distributed cloud system for encrypted IoT data that uses a proxy re-encryption scheme that allows the data to only be visible to the owner and any persons present in the smart contract.

Table 3.1: Examples of blockchain cloud storage systems [12]

3.2 P2P File Sharing

These applications involve a distributed network of computers that share data with each other without the need for a central party. Table 3.2 shows some example p2p file-sharing networks.

One of the main issues with these networks come from their anonymity property in that you can never fully trust that what you're downloading isn't malicious. On top of this, these platforms don't offer a way for user to pay for the content they are downloading, which means that legal, proprietary content is rarely distributed over them.

System	**Description of Solution**
IPFS [1]	IPFS is a content-addressable, block storage system which forms a Merkle DAG, a data structure that allows the construction of versioned file systems, blockchains and a Permanent Web.
BitTorrent [10]	BitTorrent is a p2p file-sharing system that has user bartering for chunks of data in a tit-for-tat fashion, which provides incentive for users to contribute to the network. More on BitTorrent can be found in Section 2.1
AFS [8, 3]	The Andrew File System was a prototype distributed system by IBM and Carnegie-Mellon University in the 1980s that allowed users to access their files from any computer in the network.
Napster [11]	Napster uses a cluster of centralized servers to maintain an index of every file currently available and which peers have access to it. A node will maintain a connection to this central server and will query it to find files; the server responds with a list of peers and their bandwidth and the node will form a connection with one or many of them and download the data.
Gnutella [11]	Gnutella nodes form an overlay network by sending *ping-pong* messages. When a node sends a *ping* message to their peers, each of them replies with a *pong* message and the *ping* is forwarded to their peers. To download a file, a node will flood a message to its neighbors, who will check if they have and return a message saying so; regardless, the node will continue to flood their request till they find a suitable node to download off of.

Table 3.2: Various global distributed file systems.

Chapter 4

Design

4.1 Stakeholders & Requirements

4.1.1 Stakeholders

UNFINISHED

Game Developers PRIMARY
This group will use the application to release their games and its updates to their users, who they will reward for helping to distribute it.

Players PRIMARY
This group will use this application to downloaded and update their games off of. They may also contribute to the distribution of the games to other players for an incentive provided by the developers.

Game Distribution Platforms SECONDARY
This group consists of platforms like Steam or Epic Games, which serve as the main competitor to this application. It is likely that as more developers choose this application, this group will see a loss in revenue.

TBD TERTIARY

4.1.2 Requirements

Tables 4.1 and 4.2 show the functional and non-functional requirements of this project organized using MoSCoW prioritisation.

Functional Requirements

ID	Description
	Must
F_M1	Store software metadata on a blockchain

F_M2	A node must request individual shards from its peers
F_M3	A node must be able to discover peers relevant to the software it wants
F_M4	Software must be updatable through the blockchain
F_M5	A node must be able to upload software
F_M6	A node must be able to download software in its entirety from nodes in the same network.
F_M7	A node must be able to verify the integrity of each block it downloads
F_M8	The application should run on the Ethereum network
F_M9	Users must be able to purchase games from developers over the network
F_M10	Users must be able to prove they have purchased a game
Should	
F_S1	Seeders should have a way to prove how much data they have seeded
F_S2	Seeders will only upload content to users who have a valid proof of purchase
Could	
F_C1	Allow users to request specific software versions

Table 4.1: These requirements define the functions of the application in terms of a behavioural specification

Non-Functional Requirements

UNFINISHED

ID	Description
Must	
NF_M1	The application is decentralized and cannot be controlled by any one party
NF_M2	Any user must be able to join and contribute to the network
NF_M3	Game uploaders should be publicly identifiable
NF_M4	Metadata required to download the game should be immutable
Should	
NF_S1	This application must be scalable, such that many users can upload and download the same game at the same time.
NF_S2	Only the original uploader can upload an update to their game
Could	

Table 4.2: Requirements that specify the criteria used to judge the operation of this application

4.2 Design Considerations

Architecture

Figure 4.1 shows the architecture of this application:

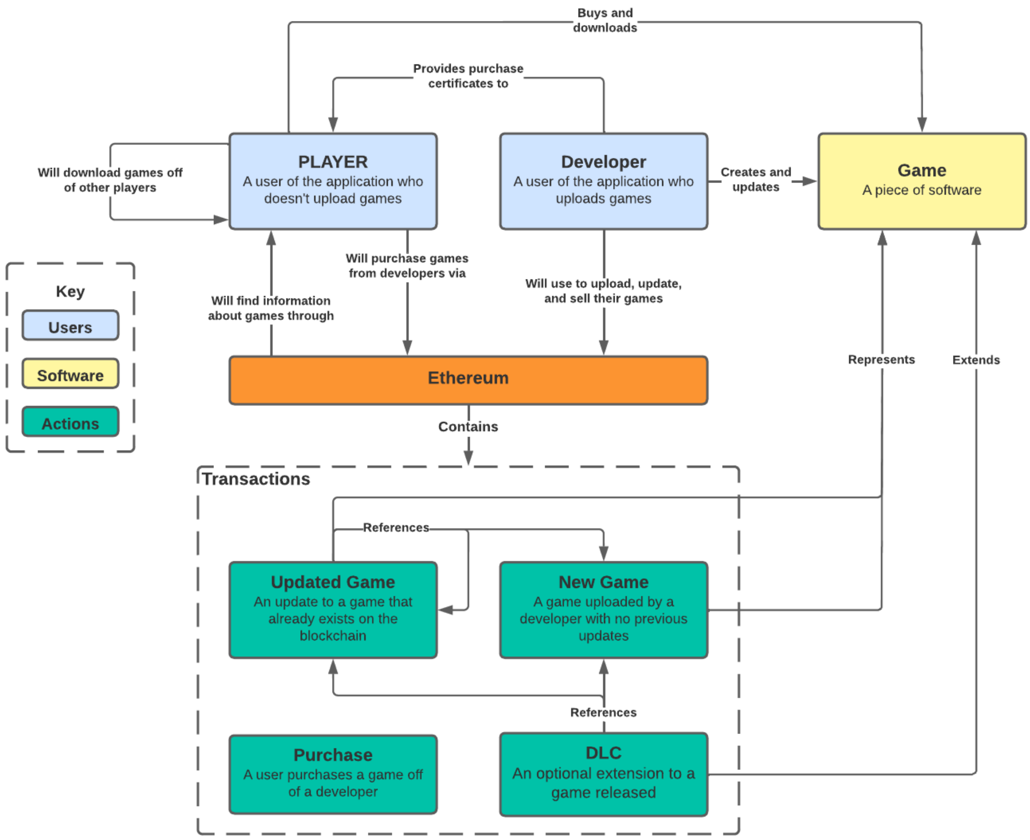

Figure 4.1: Architecture of the application

Type of Blockchain

To satisfy **NF_M1** and **NF_M2**, we will need to use a public blockchain, which will benefit our project by:

- being accessible to a larger user-base, which should boost availability and scalability (**NF_S1**),
- reducing the risk of censorship (**NF_M1**), and
- providing greater data integrity (**NF_M4**)

Ethereum is a public blockchain that allows developers to publish their own distributed applications to it. It comes with an extensive development toolchain so is an obvious choice for this project (**F_M8**).

Uploading Content

For developers to upload their game (**F_M5**), they must provide a digital certificate to prove their identity (**NF_M3**) as well as the required metadata (**F_M1**) for identifying[1], downloading[2] and verifying their game[3]. The developer is then expected to allow users to purchase the game off them and seed the game to at least an initial group of users.

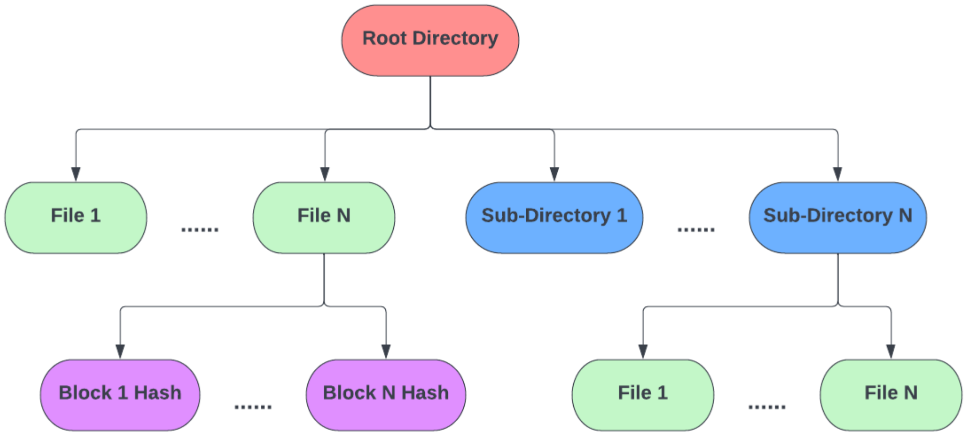

Figure 4.2: How shard data is stored.

Purchasing Content

Users will purchase content from developers using Ether (**F_M9**) and will be provided with a proof of purchase (**F_M10**) that is encrypted with the developer's private key. By using public key infrastructure, it gives proof of purchases authenticity that allows them to be validated by any node in the network. The proof of purchase will include: the root hash of the title purchased, the developer's address, the purchaser's address, a timestamp, and a unique seeder token to be used for proving distribution.

Downloading Content

Games will be content addressable, using their root hash stored on the blockchain. This will allow nodes to connect with each other based on the content they are interested in (**F_M3**). Once a user connects to a node it will:

1. Send their proof of purchase (**F_S2**).
2. request individual shards from the node using the shard's hash (**F_M2**),
3. use the metadata from the blockchain to verify the shard's contents (**F_M7**),
4. send a confirmation message that proves the successful transfer of a block (**F_S1**),
5. and repeat this until the entirety of the game is installed (**F_M6**).

Shards will be downloaded in a similar order to that of BitTorrent, which is described in Section 2.1.

[1] Such as the name, release date, version number, creator and price.

[2] Such as the address of where to purchase the game, the digital certificate of who you're purchasing off and a tracker

[3] The root hash and the hashes of each block of data

Updating Content

To satisfy **F_M4**, developers will perform the same steps as in **Uploading Content** but will also include the hash of a previous block that contains the older version of the game. This will include the restriction that only the original uploader can upload an update for their game (**NF_S2**).

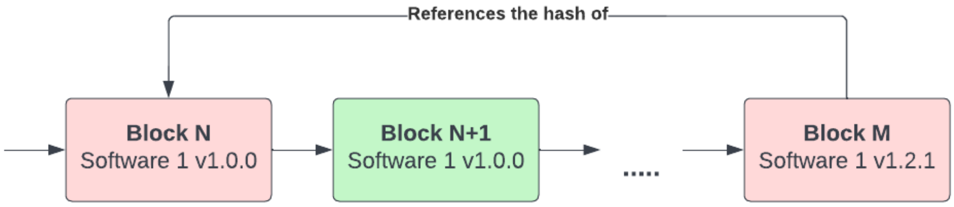

Figure 4.3: How blocks can relate to older blocks.

It is likely that many shards will persist between versions so a node will only ever download the changed or new data. To satisfy **F_C1**, a node may optionally keep older shards that have been removed or changed.

Downloadable Content

UNFINISHED
Downloadable Content DLC provides extensions to the base game that typically cost money. This could be implemented in the same way that updates are but be separately purchasable and include a dependency that means a user cannot purchase this without already owning the base game.

Proving Contribution

When a user purchases a piece of software they will be granted a unique seeder token. When a user successfully downloads a shard of data off of a peer they will reply with a confirmation message, containing their seeder token, that is encrypted using the developers public key. When a user wants to prove that they have contributed to the distribution of a game, they will send a collection of these messages to the developer, who will judge their validity.

Sequence Diagram

Figure 4.4, shows the main interactions for a game uploaded to the network and how different actors will interact with each other.

4.3 Limitations

This project will not attempt to mimic any of the social features (friends, achievements, message boards, etc.) provided by platforms like Steam. This may leave users with an inconsistent social experience for games as they will have to rely on the developers own implementations, other social platforms or just not have the features at all.

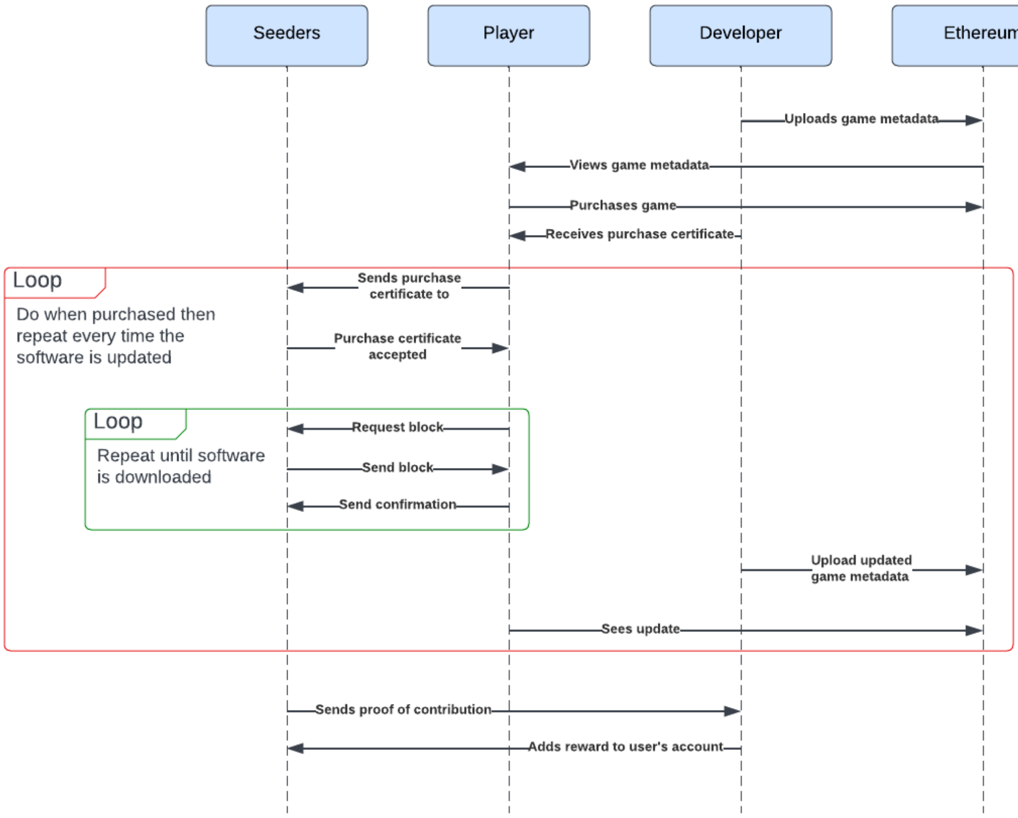

Figure 4.4: The main interactions within the application.

Like mentioned in Section 2.1, the main issue with p2p sharing is about the availability of content. The long-term distribution of a video game will often be reliant on the community, and the incentives provided by the developers, which may leave less popular titles unavailable.

Chapter 5

Project Management

5.1 Risk Assessment

Risk	Loss	Prob	Risk	Mitigation
No suitable large scale test environment	2	5	10	I do not have the infrastructure to test this project on a large network, however small scale tests will be possible.
Difficulty with blockchain development	2	3	6	I will seek advice from my supervisor about how to tackle certain problems and if necessary, what aspects of my project I should change.
Personal illness	3	2	6	Depending on the amount of lost time, I may have to not complete some of the SHOULD or COULD requirements.
Laptop damaged or lost	3	1	5	All work is stored using version control and periodic backups will be made and stored locally and in cloud storage. I have other devices that could be used to continue development.
The application is not finished	1	3	3	Using agile development will ensure that I will at least have a minimal working application. If I feel that I am running out of time, I will focus on expanding test cases and improving the write-up.

Table 5.1: The risk assessment of this project

5.2 Work to Date

My work has primarily been on research, looking at how blockchain has been used to build and supplement cloud storage systems as well as how various peer-to-peer functioned and performed. I have proposed a design for the application to be built on the EVM.

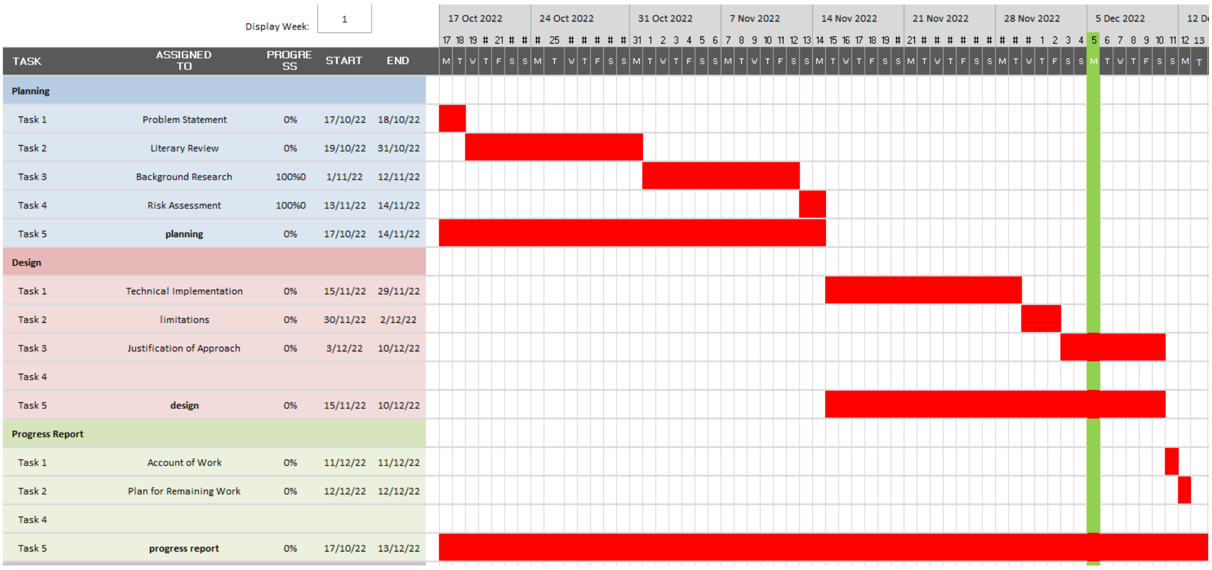

Figure 5.1: A Gantt chart for my work up until the progress report.

5.3 Plan of Future Work

Implementation & Testing

This phase I will use the agile development methodology to build my application. My sprints will all be structured into three phases:

1. **Preparation** Deciding on the set of requirements to complete and making any initial design decisions and diagrams,
2. **Implementation** using test-driven development, I will work on requirements based on their prioritization, and
3. **Review** I will discuss the completed work in that sprint including design choices, what was completed, and any issues.

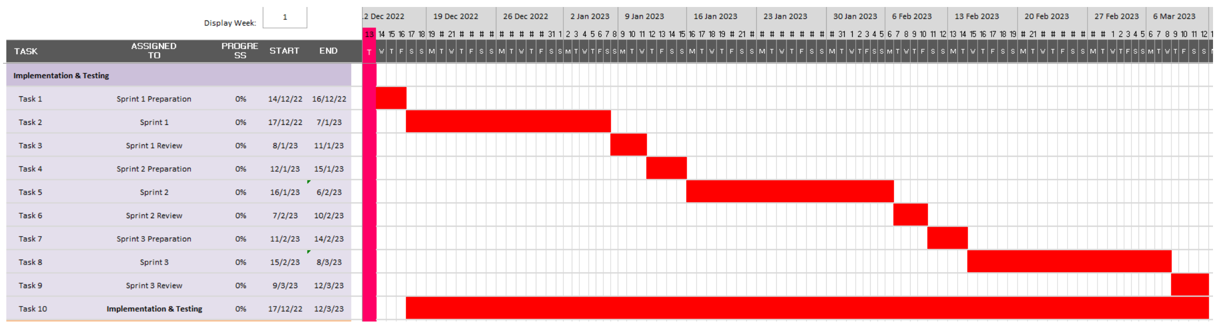

Figure 5.2: The plan for my three sprints

Testing Strategy and Results

This phase will consists of several sections:

1. **Testing Strategy** The approach I used for writing tests throughout the implementation phase and how these were used to evaluate the completion of the requirements set out in Section 4.1.2.
2. **Test Results** A report on the results of my automated and manual testing.

Evaluation

This phase will have me evaluating how successful my project was, as a whole, by focusing on several key areas:

1. **Project Organisation** How successfully did I structure my time in this project?
2. **Outcome of the Application** How successful was my application in regards to a solution to the problem set out in Section 1.1 and in terms of the requirements set out in Section 4.1.2?
3. **Limitations and Future Improvements** What were the limitations of my project and what would I change about it if I had more time or were to start again?

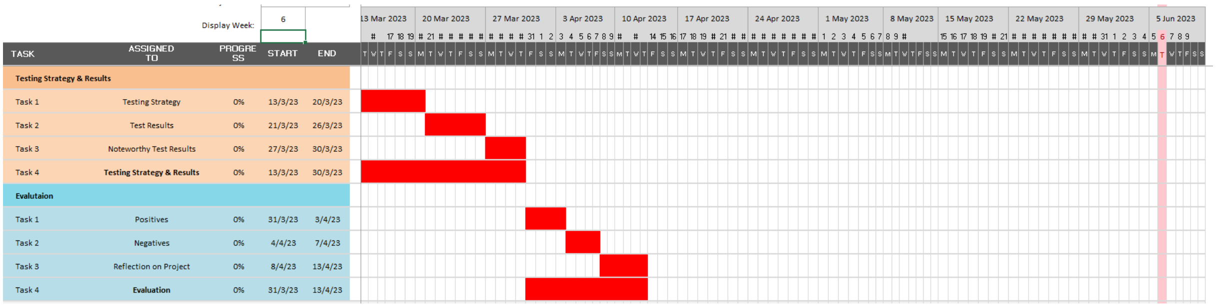

Figure 5.3: The plan for my testing and evaluation phases

Leftover Time

UNFINISHED

In Figure 5.3, you'll notice there is a large period of time at the end where nothing has been planned. This has been to done to allow some extra time where certain things take longer than expected or new things need to be added that weren't taken into account before. It also means that all of the items on my risk assessment in Section 5.1 have the time to be resolved without affecting my project.

Bibliography

[1] Benet, J. IPFS - content addressed, versioned, p2p file system.

[2] Chen, Y., Ding, S., Xu, Z., Zheng, H., and Yang, S. Blockchain-based medical records secure storage and medical service framework. 5.

[3] Howard, J. H., Kazar, M. L., Menees, S. G., Nichols, D. A., Satyanarayanan, M., Sidebotham, R. N., and West, M. J. Scale and performance in a distributed file system. 51–81.

[4] Kaune, S., Rumín, R. C., Tyson, G., Mauthe, A., Guerrero, C., and Steinmetz, R. Unraveling BitTorrent's file unavailability: Measurements and analysis. In *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, pp. 1–9. ISSN: 2161-3567.

[5] Li, J., Wu, J., and Chen, L. Block-secure: Blockchain based scheme for secure p2p cloud storage. 219–231.

[6] Li, J., Wu, J., Chen, L., and Li, J. Deduplication with blockchain for secure cloud storage. In *Big Data*, Z. Xu, X. Gao, Q. Miao, Y. Zhang, and J. Bu, Eds., Communications in Computer and Information Science, Springer, pp. 558–570.

[7] Manzoor, A., Liyanage, M., Braeke, A., Kanhere, S. S., and Ylianttila, M. Blockchain based proxy re-encryption scheme for secure IoT data sharing. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 99–103.

[8] Morris, J. H., Satyanarayanan, M., Conner, M. H., Howard, J. H., Rosenthal, D. S., and Smith, F. D. Andrew: a distributed personal computing environment. 184–201.

[9] Neglia, G., Reina, G., Zhang, H., Towsley, D., Venkataramani, A., and Danaher, J. Availability in BitTorrent systems. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pp. 2216–2224. ISSN: 0743-166X.

[10] Pouwelse, J., Garbacki, P., Epema, D., and Sips, H. The bittorrent p2p file-sharing system: Measurements and analysis. In *Peer-to-Peer Systems IV*, M. Castro and R. van Renesse, Eds., Lecture Notes in Computer Science, Springer, pp. 205–216.

[11] Saroiu, S., Gummadi, P. K., and Gribble, S. D. Measurement study of peer-to-peer file sharing systems. In *Multimedia Computing and Networking 2002*, vol. 4673, SPIE, pp. 156–170.

[12] SHARMA, P., JINDAL, R., AND BORAH, M. D. Blockchain technology for cloud storage: A systematic literature review. 1–32.

[13] WANG, L., AND KANGASHARJU, J. Measuring large-scale distributed systems: case of BitTorrent mainline DHT. In *IEEE P2P 2013 Proceedings*, pp. 1–10. ISSN: 2161-3567.

[14] WANG, S., ZHANG, Y., AND ZHANG, Y. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. 38437–38450. Conference Name: IEEE Access.

[15] YUE, D., LI, R., ZHANG, Y., TIAN, W., AND PENG, C. Blockchain based data integrity verification in p2p cloud storage. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 561–568. ISSN: 1521-9097.