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

Thomas Smith, tcs1g20
February 9, 2023

Using Blockchain for Video Game Distribution

Project Supervisor: Leonardo Aniello
Second Examiner: Heather Packer

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 [9, 2]
(b) being vulnerable to censorship from governments,
The Chinese version of Steam is heavily censored [15]
(c) the user's access to their games is linked to the platform.
If the platform shuts down, the user loses all their games

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. This is in the aim to provide greater profits to developer's, freedom from censorship, and better digital ownership for the user.

1.3 Scope

This project will be broken down into two distinct components:

1. **On-Chain** This component will consist of a set of Solidity Smart Contracts written for the Ethereum blockchain that will allow users to view metadata about and purchase games. It will be tested using a local test-net like Ganache using TypeScript. It will later be deployed to the Ethereum test-net to showcase the application in a live network.
2. **Off-Chain** This component will be what users will actually run. Each user will join a peer-to-peer network in which they can upload and download games off of other users. This will interface with the blockchain to allow users access to game metadata. See Section **??** for details about how this will be tested.

For both of these, a series of acceptance tests, that directly correlate to individual requirements, will be run and include a series of integration tests to show that my ap-

plication can meet the requirements and goals I set out. A more detailed description is given in Section **??**.

Chapter 2

Background Research

2.1 BitTorrent

It is an unrealistic to expect that every game uploaded to the network will be downloaded by every user so only a subset of users will have the game installed and available to share. In this section and Section 3.2, I will look at how various peer-to-peer file-sharing networks allow users to discover and download content that is fragmented across the network.

BitTorrent [5, 12] 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 [16].

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

One of the most significant issues facing BitTorrent is the availability of torrents, where *'38% of torrents become unavailable in the first month'* [5] and that *'the majority of users disconnect from the network within a few hours after the download has finished'* [12]. This paper [11] 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, in Ether, 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:

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 a particularly important to this project as users will likely all have varying versions of the same software.

Paper	**Description of Solution**
Blockchain Based Data Integrity Verification in P2P Cloud Storage [18]	This paper uses Merkle trees to help verify the integrity of data within a P2P blockchain cloud storage network. It also looks at how different structures of Merkle trees effect the performance of the system.
Deduplication with Blockchain for Secure Cloud Storage [7]	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 [6]	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 [3]	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 [17]	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 [8]	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 [14]*

3.2 P2P File Sharing

It is unreasonable to expect every node to have a copy of each game uploaded to the blockchain so data will be fragmented across the network. This project will use ideas from various P2P file-sharing networks to help connect nodes interested in the same content Table 3.2 shows some example p2p file-sharing networks.

The main issues involving these networks are:

1. **Trust** Nodes are typically anonymous and you can never fully trust that what you're downloading isn't malicious, and
2. **Payment** These platform don't allow users to pay for content and are generally large sources of piracy.

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 [12]	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 [10, 4]	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 [13]	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 [13]	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

Stakeholders

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.

Other 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.

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 game metadata on a blockchain
F_M2	A node must download data as constant-sized shards from its peers
F_M3	A node must be able to discover peers who have their desired game installed
F_M4	Games must be updatable through the blockchain
F_M5	A node must be able to upload games

F_M6	A node must be able to download games in their entirety from nodes in the 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
F_S3	Allow for the distribution of Downloadable Content (DLC) for games
	Could
F_C1	Allow users to request specific game versions
F_C2	Provide a simple GUI for interacting with the blockchain

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

Non-Functional Requirements

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
NF_S3	Only the original uploader can upload a DLC for their game
	Could
NF_C1	The application could have an intuitive GUI

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

4.2 Design Considerations

Data

The first consideration is what kind of data we are going to be storing and where is it going to be stored.

Data	Size	Location	Explanation
Game Metadata		Ethereum	This data represents information about the game that help identify it, such as its title, developer, version, root hash, etc. This information should allow for the unique identification of every game uploaded, whilst remaining minimal. Due to the minimal amount of storage required and the fact that every user should be able to discover every game, this data is best stored on the blockchain.
Game Hash Data		IPFS	This data will be the hash tree of a given game and will be used by a player to verify the contents of each block of data they download as well as the expected structure of the applications contents. Due to this data's moderate size and the fact that not every user will need to view every game's hash tree, there is no need to upload this to the blockchain as this will add an unnecessary expense in publishing games. IPFS is ideal for this as we do not need to restrict access to the data but still need to easily share it and we can simply include the IPFS ID within our game metadata stored on the blockchain.
Game Data	avg. 60GB	Peers	This will be the actual data for the game that is used to run it. To play a game, a user will need access to all of its data. More information about this is in Section **??**. An important property for uploaded games is that all users do not have access to all games without having payed for them first so we shouldn't store them on public platforms like the blockchain or IPFS. Section **??** will discuss how this can be achieved using a distributed peer-to-peer file sharing model.

Architecture

Figure 4.1 shows the architecture of this application:

Sequence Diagram

Figure 4.2, shows the main interactions between actors in the application.

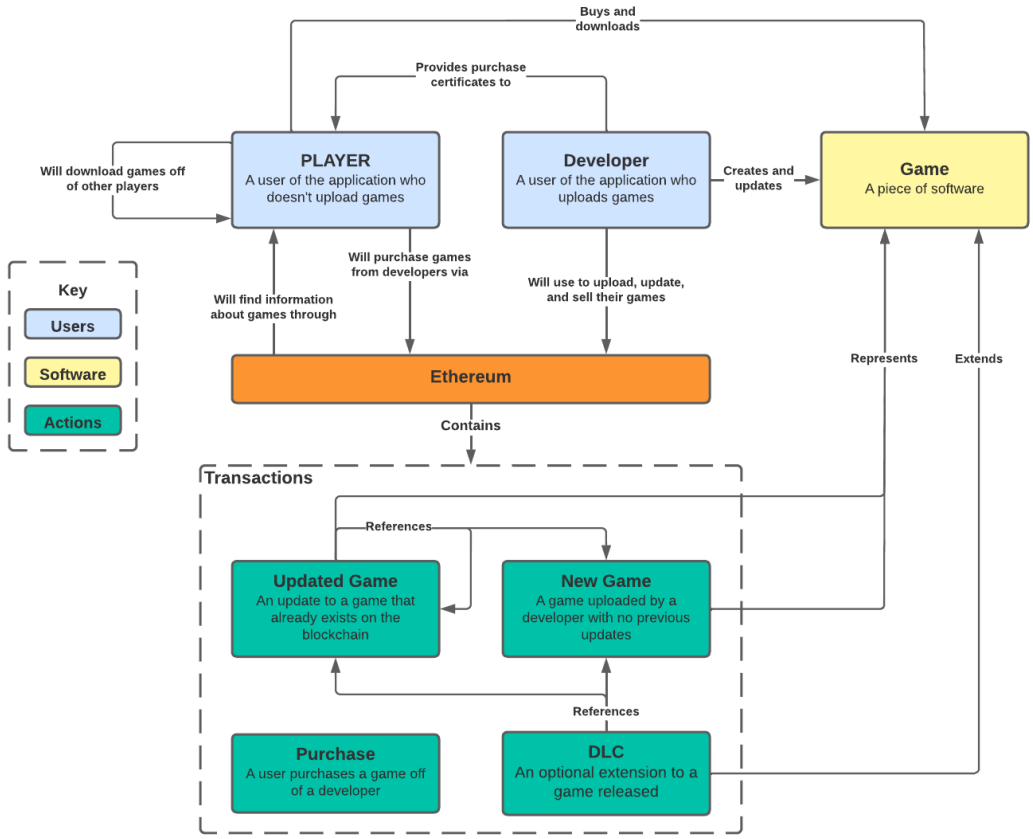

Figure 4.1: *Architecture of the application*

4.2.1 Blockchain

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**).

Data to Store

Like mentioned in Section 4.2, the data stored on the blockchain is used for the identification of each game and will consist of the following fields, where *italic* fields will be automatically-generated:

Name	Description
title	The name of the game.
version	A version number of the game.
release date	When the game was released.
developer	The name of the developer releasing the game.
uploader	The Ethereum address of the developer.
root hash	The root hash of the game that uniquely identifies it and is based upon its contents.
IPFS address	The ID of the hash tree of IPFS that can be used to download the hash tree before starting a download of a game.

Purchasing Content

Users will purchase content from developers over Ethereum using Ether (**F_M9**) and this will be recorded on the blockchain (**F_M10**). Any user can see which other users have purchased the game users can prove this between each other using their public/private keys.

4.2.2 Distributed File Sharing

Hash Tree

The hash tree of a given directory is a tree object that stores information about its structure and contents. This is used to tell users what information they need to download, where it goes and what its contents should be. For every file, the hash tree stores a series of SHA-256 hashes.

Uploading Content

For a developer to upload their game (**F_M5**) they must provide the required metadata outlined in Section 4.2.1 as well as a hash tree that is generated when they create the game.

A developer is expected to host their hash tree on IPFS indefinitely and seed the game data at least to an initial group of people.

Downloading Content

Updating Content

Downloadable Content

Proving Contribution

Downloading Content

Games will be content addressable, using their root hash stored on the blockchain. This will be used to help connect nodes who are interested in the same content (**F_M3**). Once a user connects to a node it will:

1. Send their proof of purchase (**F_S2**).

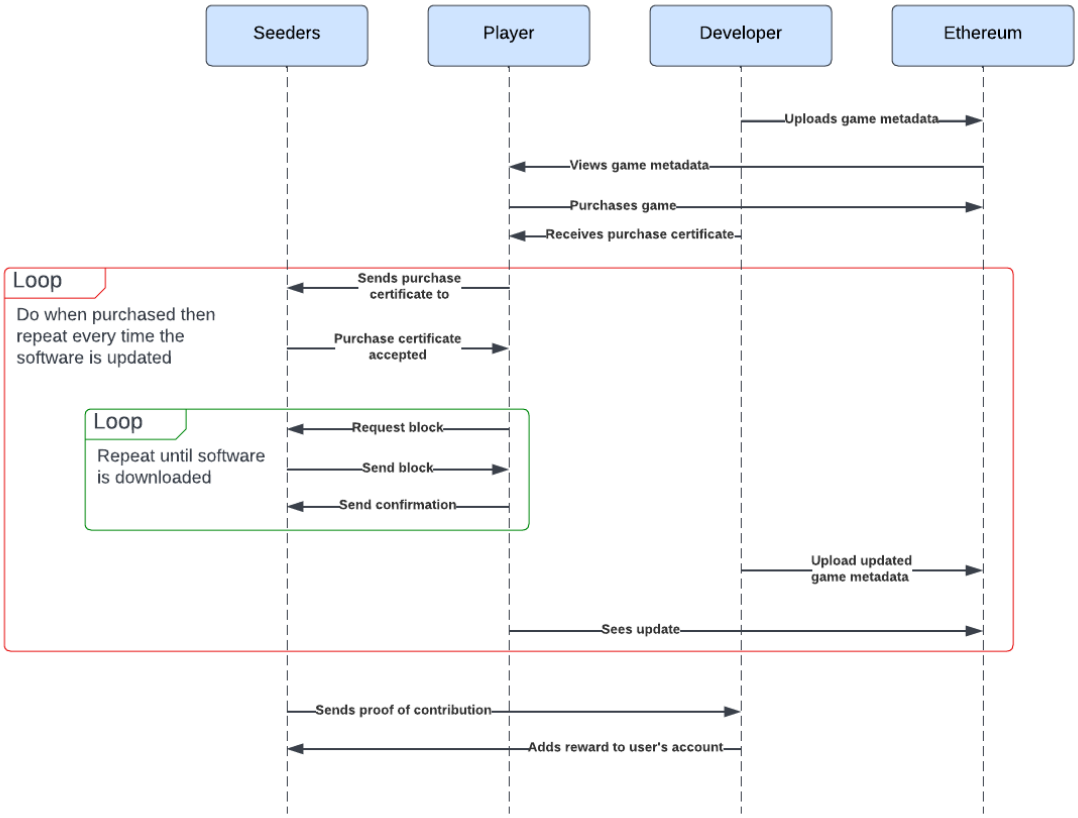

Figure 4.2: *A sequence diagram showing some of the main interactions within this application*

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**).

As the blockchain will only be used to store metadata about games uploaded to the network, not every node will have each game installed. Instead, this project will take ideas from networks like BitTorrent, where nodes will seek out peers who have the game installed using a tracker hosted by the uploader.

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 relationship will allow users ownership to all versions of a game they have purchased and not just a singular version. This will include the restriction that only the original uploader can upload an update for their game (**NF_S2**).

Each version will be considered as a standalone game and will require users to download the entirety by scratch. Whilst this isn't reflective of how updates are typically managed, this will be acceptable for the scope of this project and any changes will be considered as a future extension to this project.

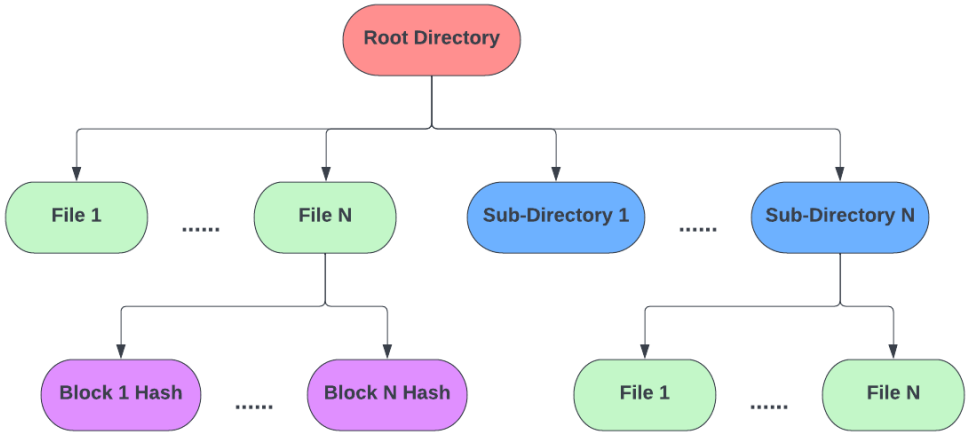

Figure 4.3: *THe structure of a hash tree*

Downloadable Content

Developers will typically offer paid or free DLC (**F_S3**) for their games that users will purchase separately. Each DLC will need:

1. **Dependency** A reference to the oldest version of the game they apply to, and
2. **Previous Version** A pointer to the previous version of the DLC.

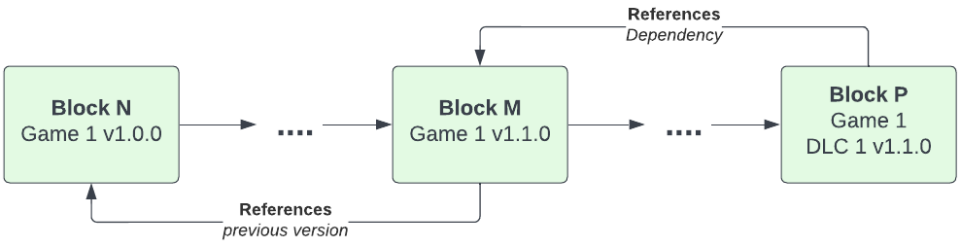

Figure 4.4: *How blocks relate to each other. An update will reference the previous version whilst a DLC will reference, which piece of software and version it is dependent on.*

Proving Contribution

A purchase certificate will include a confidential *seeder token*. When a user successful downloads a shard of data they will send a confirmation message, including their seeder token, that is encrypted with the game developer's public key. A user will prove their contribution (**F_S1**) by sending a collection of these messages to the developer, who can decrypt and validate the tokens.

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.

Section 2.1 highlights the issue of availability in p2p sharing networks and this platform will likely suffer from similar issues. These can be mitigated by having an active community or good incentives to help distribute the game.

Chapter 5

Project Management

5.1 Risk Assessment

Risk	Loss	Prob	Risk	Mitigation
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 ignore 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 testing and finalising the report.

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 cloud storage systems as well as how various peer-to-peer function and perform. I have proposed a design for the application to be built on the EVM.

5.3 Plan of Future Work

Below is a high level plan for the work I have remaining.

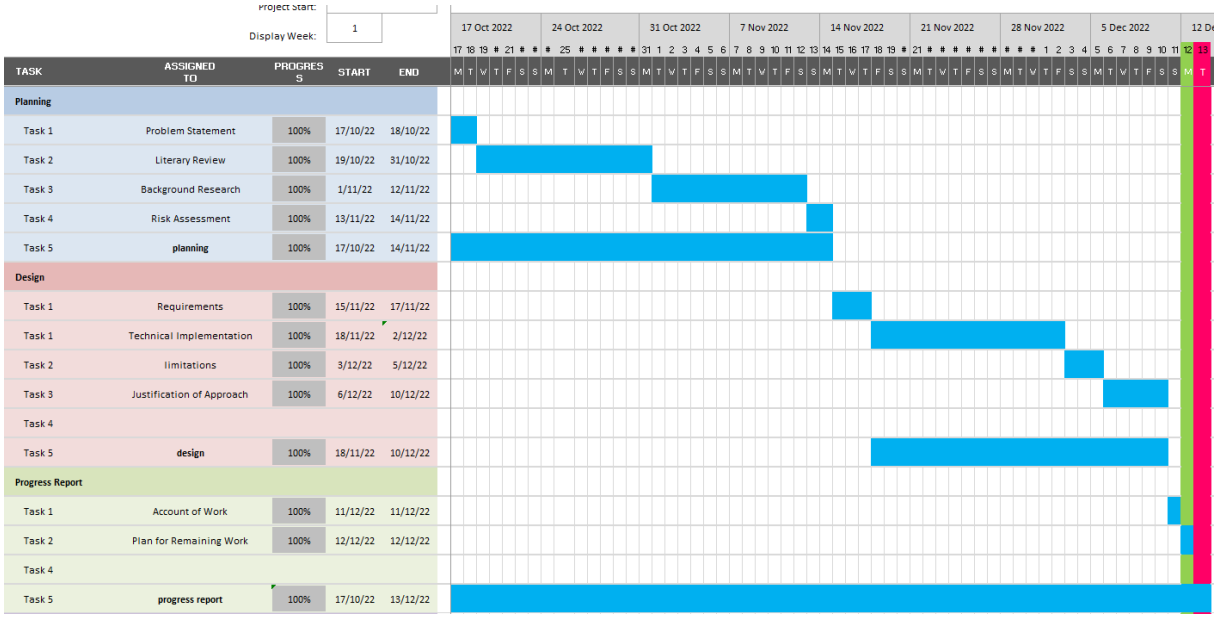

Figure 5.1: *A Gantt chart for my work up until the progress report.*

Implementation & Testing

In 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.

By using a preparation and review phase with each sprint, I can make detailed notes on my implementation as I go so when I come to write the full report I have a strong starting point.

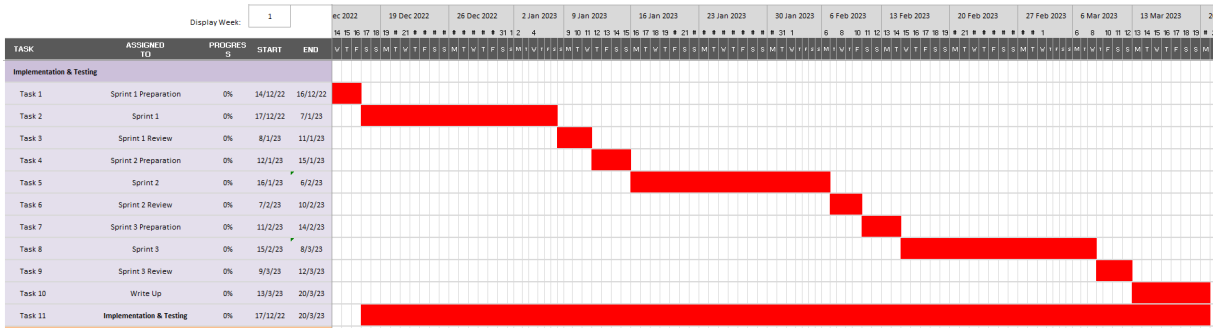

Figure 5.2: *The plan for my three sprints*

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. **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?
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?

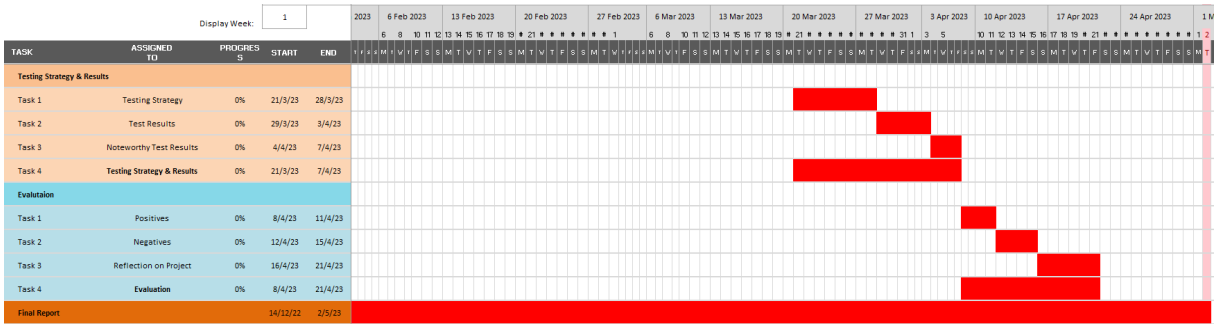

Figure 5.3: *The plan for my testing and evaluation phases*

Final Report

I will be writing the first draft of my report throughout the project, aiming to finalise sections at the end of each phase of this project. Any leftover time at the end will be focused on finishing the report.

Chapter 6

Implementation

6.1 Sprint 1

Planning

Game Smart Contract

A smart contract can be created for each game with the following fields:

Attribute	Description
name	The name of the video game being published
version	The version number of the game
author	The address of the author
certificate	A digital signature from the author
price	The price in Ether
previousVersion	A reference to a previous version of the game if one exists

Chapter 7

Testing

7.1 Unit Testing

7.2 Acceptance Testing

Strategy

Each requirement must complete a series of acceptance tests to be considered complete. These tests should be run in a live environment that uses the platform as a whole and demonstrates that each section of the project the requirement covers is performing correctly.

Results

7.3 Benchmarking

Strategy

Benchmarking tests will be used to demonstrate how the scalability and performance of my application in a live environment, by running 'expensive' tasks given several different contexts:

1. **Locally** A peer will interact with many other peers being run on the same device,
2. **Over devices** A peer will interact with other peers over the internet that have been deployed to cloud services or are running on local machines.

Bibliography

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

[2] Brown, A. Valve defends taking 30 per cent cut of steam sales in response to lawsuit.

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

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] Marks, T. Report: Steam's 30% cut is actually the industry standard.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] Sharma, P., Jindal, R., and Borah, M. D. Blockchain technology for cloud storage: A systematic literature review. 1–32.

[15] SteamDB. Steam china officially launched.

[16] 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.

[17] 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.

[18] 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.