

AMAZON BLOCKCHAIN-BASED QLDB

INNDX TEAM – ITECH 7415

Supervisor
Dr. Azadeh Noori Hoshyar
Dr. Mustafa Hashmi

Saugat Paudel (30363035)
Melvin Flores (30352985)
Bhautikkumar Patel (30367311)
Chiranjivi Thunuguntla (30352950)

FEDERATION UNIVERSITY | BRISBANE

ABSTRACT

This report has been prepared for Inndox as a part of ITECH7415 master's project at Federation University. Inndox is providing a bridging software for secured document transmission between company/builder and property owner/buyer. It will help property builder or owner easily access and handover their property documents in secured and easier way.

The scope of this project is to provide more security and transparency to those properties' documents using a blockchain based framework which can in turn increase the trust of the customers. On top of that it is also made sure the blockchain based framework is suitable for current Inndox system. Amazon QLDB (Quantum Ledger Database) is finalized and used as the database that can provide proper security and can satisfy the client's needs. The project consists of Amazon QLDB which will have the connection to the current Inndox back-end and will store the transactional log after the property document is stored in current Inndox database. This report focuses on providing a detailed information about how Amazon QLDB created using PartiQL (Query Language) works for Inndox.

Table of Contents

1. Introduction	1
1.1 Project Introduction	1
1.2 Problem statement	1
1.3 Product Vision	2
1.4 Project Scope:	2
1.5 Project Objective:	2
2. Project Storyboard	3
2.1 Project ID and Title	3
2.2 Project Client	3
2.3 Project Outline	3
2.4 Project Timeline	4
2.5 Stakeholders	4
2.6 Project Communication Plan	5
2.7 Collaboration	5
2.8 Project Product	5
2.9 User Stories	6
2.10 Project Scope	6
2.11 Success Acceptance Criteria	6
3. Literature Review	7
3.1 Real Estate, Blockchain and Ledger Database	7
3.2 Tools and Techniques	9
3.3 Methodology	10
4. Project Approach	12
4.1 Agile Methodology/Scrum Framework	12
4.2 Tools and Technique	15
4.3 Project Development	15
1. Sprint 1	15
2. Sprint 2	16
3. Sprint 3	21
5. Product Demonstration	24

6. Errors and Resolution	34
6.1. Authorization Error in Creating Tables.....	34
6.2. Error in Building REST API.....	34
6.3. Configuration Error	35
6.4. Credential Error	35
7. Testing Phase.....	36
8. Applications, limitations, and future recommendations	50
8.1 Application of the product	50
8.2 Limitation of our project	50
8.3 Challenges	50
8.4 Future Improvements	51
9. Conclusion.....	52
References	53

List of Figures

Figure 1- Waterfall Model (Source: Curcio (2018))	10
Figure 2 - Scrum Framework (Source: Curcio (2018))	11
Figure 3 - Agile Methodology (Source: hub.packtpub.com)	12
Figure 4 - Scrum Framework (Source: zaynabzahrablog.wordpress.com)	13
Figure 5 - Microsoft Visual Studio 2019	17
Figure 6 – Code snippet showing table created on Amazon QLDB	18
Figure 7 - Showing the data verification as true.....	19
Figure 8 - Showing the data verification as false	19
Figure 9 - Export job successful in QLDB.	20
Figure 10 - Exported data in Amazon S3	20
Figure 11 - Etag is valid after executing the program.....	22
Figure 12 - Etag is invalid after it has been tampered with by executing the program.....	22
Figure 13 - Table and index created	24
Figure 14 - Connect to Amazon QLDB.....	25
Figure 15 - Inserting the record	26
Figure 16 - Updating the record	27
Figure 17 - Connect to AWS S3.....	28
Figure 18 - Return S3 path from QLDB.....	29
Figure 19 - Get Etag from Amazon QLDB	30
Figure 20 - Get Etag from AWS S3	31
Figure 21 - Process Etag	33
Figure 22 - API Gateway error	34
Figure 23 - Connect to Amazon QLDB.....	37
Figure 24 - Connect to AWS S3.....	38
Figure 25 - Create Inndox ledger in Amazon QLDB	39
Figure 26 - Create tables in the Inndox ledger in Amazon QLDB	40
Figure 27 - Insert the data in the tables of the Inndox ledger in Amazon QLDB.....	41
Figure 28 - Create a connection between our backend (Amazon QLDB) and AWS S3	42
Figure 29 - Upload the folder/files in AWS S3.....	43
Figure 30- Validate the S3 Return Path.....	45
Figure 31- Validate the Etag if Not found	46
Figure 32- Validate the Etag if valid.....	47
Figure 33- Validate the Etag if invalid.....	49

List of Tables

Table 1. Sprint Schedule	14
Table 2. Revised user stories.....	14

Acknowledgement

We would like to thank Federation University for this wonderful opportunity of doing this master's project with the collaboration of Innadox company. We would also like to thank our client Trish Mackie-Smith and Andrew Mackie-Smith for providing and assisting the team throughout the project. We would like to thank Innadox developer, Mr. John Henao for assisting and guiding us on technical parts throughout our project duration. We would also like to thank our course coordinator Ms. Taiwo Oseni, who has taught us to learn and understand the core concepts useful for this project. We would like to give special thanks to our supervisor Dr. Mustafa Hashmi and Dr. Azadeh Noori Hoshyar for their support and guidance throughout the project period.

1. Introduction

Buying or building a new property is a dream for many people, even more for those who use their life savings solely for this purpose. But getting new property by paying money does not end the challenges of owning a property can be for the buyer. Caveat Emptor (Buyer beware) law states that if you buy any property and face issues later than all costs regarding those issues will be from the buyer and will have no recourse from the vendor. That means property buyer must verify their documents and keep it safe for future if any problem arises regarding their documents in the future. On top of that documents which are in the paper form, email, USB, etc. are never safe. Data are crucial aspects for any one regardless what they do. On top of that there are a lot of cases of forged documents or fraudulent activity done, which will put the property owner into legal dispute. A property buyer who has bought a property after using his life savings if finds that there is problem in those documents and must bear loss which he/she cannot afford will be devastated. Currently Inndox is using Amazon S3 as the database for their customer. It provides some decent level of security, but acceptance level of data transparency is still an open challenge. So, our project, of creating a database on Amazon QLDB helps store these documents data and maintain data transparency as seen in Amazon Web Services, Inc. (2020).

1.1 Project Introduction:

Inndox is a company, driven to build a platform for property builder or owner to securely store and handover the documents related to the property. Over the years, construction company or owner provide the documents logbook after the finishing of the construction or significant renovation project. The building logbook is to give a basic outline report, portraying how the new or restored construction is expected to work and be overhauled. It likewise gives a way to log the vitality execution and support of the administrations inside the structure and an authentic record of building modifications, upkeep, and vitality execution. The Inndox project provides the solution of handover documents through Blockchain-based application and gives reliable trust to the client and owner.

1.2 Problem statement:

There are a lot of issues around property records in the real estate industry. Paper records and important information about the property are usually “handed over” by the construction company to the owner via email or on USB, when the owner goes to lease, insure, renovate, refinance, or sell. What makes it unreliable? Such unreliable records are subject to fraud and this causes confusion around the building’s safety and compliance. The property owner does not have to provide the information to the buyer, and this may bring injustice to future buyers. The Caveat Emptor law is an old-fashioned and unfair law, that allows defective and unsafe buildings to be sold to innocent buyers who have no legal remedy against the

dishonest seller. For example, data regarding the apartment building cladding can be proven to be safe on Inndox but if it is not safe and the records are not provided to the property owner, the buyer of the apartment in that building may suffer as a result of purchasing a dangerous building. The buyers can lose their life savings, suffer health issues, injuries, and even death if some accidents happened due to the non-compliant cladding material used in the building. Inndox will make sure every document must be submitted as per their checklist. If anything is missing it will inform property owner as well as future owners which would help buyers to have a complete awareness about the property issues.

1.3 Product Vision:

To integrate a blockchain-based technology into the current Inndox Company database, so that immutability can be achieved to ensure greater trust and transparency.

1.4 Project Scope:

Following are the scope for this project:

- Original property data captured by Inndox will be protected and immutable ensuring trust and transparency.
- Develop a database technology for integration into the current system.
- Set up the environments for developing, testing, and deploying.
- Complete development of a workable blockchain solution ready for deployment.

1.5 Project Objective:

- Find a proper blockchain- based technology for Inndox.
- Develop a suitable Blockchain-based technology for integration into the current system.
- Set up the environments for developing, testing, and deploying.

2. Project Storyboard

The following sections below describes the project storyboard for Inndox.

2.1 Project ID and Title

- **Release date:** 21 August 2020
- **Update author:** Inndox team (Flores, Patel, Poudel, Thunuguntla)

2.2 Project Client

- Inndox
- <https://www.Inndox.com/>
- trish@Inndox.com
- NDA signed and sent to client on 13 August 2020
- Slacks, Trello, GitHub, zoom (if needed)
- Client meeting every Monday at 2pm in RCL (River City Labs)
- Access to Inndox staging environment

2.3 Project Outline

The client, Inndox, provides a digital property logbook and a digital handover solution. Their customers are property developers and construction companies who provide Inndox with the original, highly valuable property data to meet their compliance and legal obligations to all future property owners.

There are a lot of issues around the real estate industry. History of buildings and property are unaccounted for. Paperwork's or flash drive/ USB sometimes go missing when it is really needed. Untrusted records issues arise because of document frauds. There is also Injustice for buyers, because the 'buyer beware' law(Caveat Emptor law) is an unfair law for buyers, whereas the seller of the property does not need to provide anything to the buyer, if eventually the buyer gets ripped off.

Now, Inndox system is housing these property records that you could trust. A trusted central location that everyone could know the history of the property, and the true value of that property.

These property handover documents include some of the following:

- Compliance forms
- Builder's warranty information
- Product warranty information

- Product user guides
- Color selections
- Material and finishes details
- Plans
- Contract Documents
- Contact details of suppliers

Currently, these property records are locked in the Inndox Amazon Web Server (AWS) so that it can never be deleted or amended. Inndox want to utilize the blockchain technology, so that these property records are stored in blockchain, achieving immutability which ensures greater trust and transparency.

By applying the blockchain technology without impacting the current Inndox system, Inndox wants the database to be more secured and trusted. Thus, the property handover documents will be housed in two storage locations:

- AWS servers
- Blockchain nodes

2.4 Project Timeline

Project Start Date:

- 13 August 2020

Project Finish date:

- 16 October 2020

2.5 Stakeholders

Founder/Client:

- Trish Mackie-Smith – Client Project Manager / Client Product Owner

Co-Founder/Client:

- Andrew Mackie-Smith

Inndox Developer:

- Jhon Henao

Project Supervisor:

- Mustafa, Hashmi (Team Supervisor)
- Dr. Azadeh Noori Hoshyar (Course Supervisor)

2.6 Project Communication Plan

- On Wednesday at 2pm-5pm, the team will work in the Inndox RCL space or FedUni RCL space.
- Other communication tools such as email, slacks, GitHub, Zoom/Microsoft team (if needed).

2.7 Collaboration

Collaboration strategy:

- Weekly meetings with the client will be conducted every Monday at 2pm-4pm in Inndox RCL space. After which the team will work, around 4pm-6pm in the Inndox RCL space or FedUni RCL space.
- On Wednesday at 2pm-4pm, the team will work in the Inndox RCL space or FedUni RCL space.
- Each team member is expected to upskill themselves in blockchain technology by learning online blockchain tutorials found in YouTube, Udemy, lynda.com, AWS self- paced training modules and google. The essential tutorial will also be added as an item in the product backlog to be accomplished by each team member.

Where and how the team will collaborate to execute the project:

- Inndox/FedUni RCL space, every Monday after client meeting, which is around 3pm-5pm in RCL (River City Labs)
- Inndox/FedUni RCL space, every Wednesday at 2pm-5pm in RCL (River City Labs)

Collaboration tool and communication tools:

- Slacks, Trello, GitHub, zoom (if needed), Microsoft Teams will always be available for collaboration purposes.

2.8 Project Product

- Blockchain recommendation report
- Dapp (Decentralized Application)
- Admin manual
- Testing report
- Deployment plan

2.9 User Stories

As an owner,

- I want to know which blockchain technology is suited for our current technology, so that it can be compatible and be easily integrated in the current Inndox application system.
- I want to have a secured signup and sign in functionality, so that user fraud and user identity theft is prevented.
- I should be able to store the data captured in the blockchain database, so that there is data correctness and consistency.
- I want to stored data in the database to be accessible as soon as command is given to do so, so that data is available for use.

As a user,

- I want to add some document files into my folders, so that I can store it in one place and retrieve it, if needed.
- I want to send all the documents securely through blockchain application, so that it is safely stored in one repository.
- I want to send and receive the confirmation mail on my ID and client ID, so that I am informed, aware and affirm that documents were stored.

2.10 Project Scope

Following are the Scope for this project:

- Original property data captured by Inndox will be secured and immutable ensuring trust and transparency.
- Determine the optimal blockchain technology for the project.
- Develop the suitable technology for integration into the current system.
- Set up the environments for developing, testing, and deploying.

2.11 Success Acceptance Criteria

- Clients assess the recommendation report. Client makes the decision in which blockchain framework is to be used by Inndox company.
- Current functionality of Inndox system remains the same. During a handover process, property records are stored in AWS. In addition, it is also stored in blockchain.
- Only subscribed Inndox customers can participate in the blockchain process and are secured with the blockchain authentication protocol.
- After the handover process, property records are already stored in AWS. as well as in blockchain. An API is used to query the property records in the blockchain network.
- Successful development and demonstration of a blockchain technology which is suitable for integration into the current system.

3. Literature Review

The Inndox student team scoured google scholar and FedUni library online for literatures regarding real estate, blockchain, ledger database, tools and techniques, and methodology. The following sections discussed the literatures and synthesized their importance.

3.1 Real Estate, Blockchain and Ledger Database

Purchasing a real estate property can be a daunting task and knowing the history shows the true value of that property, thus saving you money for any future costs. Blockchain is a technology which revolutionize industries and have the following attributes. It is a decentralized distributed digital ledger, all transactions recorded are immutable, it is a 'trust less' system involving multiple parties, and only allows authentic transactions (Amazon Managed Blockchain). Ledger Database is a log-centric which means data is appended at the end of the current data and it is ordered sequentially by time. An example of a ledger database is Amazon Quantum Ledger Database (QLDB). It is an emerging technology, and a product by Amazon which provides record keeping in a centralized distributed digital ledger, with the same attributes of blockchain such as immutability and cryptographically verifiable history of data (Amazon Quantum Ledger Database).

Relevant literatures have been selected and explored regarding problems in the real estate industry. It is the hope of this literature review to provide a blockchain-based and ledger-based solution like Amazon QLDB, based on peer-reviewed journals and articles. Despite digitization and automation of property record keeping, these problems are still prevalent. Amazon QLDB could help solve or minimize this problem. The following literature review discussed the problems in the real estate industry, the proposed solution of using a blockchain-based and ledger-based database such as Amazon QLDB, and its disadvantages and advantages (Amazon Quantum Ledger Database).

Spielman (2016) pointed out three major issues that arises in property records. Firstly, the decentralized property record keeping wherein documents are kept in their respective county. Secondly, mistakes from paper-based recording practices which accounted for 30% of defective titles. Lastly is the high cost of property transactions mainly attributed from the first two issues previously mentioned. In addition, Veuger (2018) pointed out an important issue that is encountered by real estate fund managers. It is the issue of safeguarding an asset (e.g. ownership identification). Consequently, Chavez-Dreyfus (2016) described the issue of the time-consuming process in completing a real estate deal. In Sweden, it takes months to complete the real estate contract.

In his article, Spielman (2016) specified the disadvantage of using a blockchain-based solution: decentralized database is slower compared to a centralized database. And he doubted if blockchain is suitable for a high-speed and high-volume transactions of data. Nevertheless, the author introduced a blockchain-based solution for the property registry with the three advantages. First is its increased security, as encryption is used for its transaction which will

resolve fraud issues such as identity theft. Second is its inherent protection as hashes depend on its current transaction data as well as its previous transaction data, which results into data immutability. Third is transparency because transaction data needs to be consistent for it to be added to the blockchain, and multiple users can access that same data. Furthermore, Veuger (2018) concluded that trust is achieved in the real estate industry by using the blockchain technology which has the key advantage of transparency. In a blockchain, change cannot be done once a block has been added to the chain of blocks. Duplicate transactions are prevented, and shadow transactions never occur. Besides, Chavez-Dreyfuss (2016) observed that property transfers using blockchain technology will be less time-consuming. Assigning property title benefits all stakeholders like brokers, buyers, and sellers. They also can track it. These benefits address the need for a blockchain database. It focuses mainly on the considerations of cryptographic encryption, immutability, transparency, trust, and faster response time. To summarize, the advantages of using a blockchain database is greater than its disadvantage. And it can be noticed that in Amazon QLDB, the disadvantage of blockchain database mentioned by Spielman (2016) is addressed in Amazon QLDB because it is a centralized distribution.

In an article authored by Zhang et al. (2020), he noted a disadvantage of a ledger database: the volume of data grows over time leading to an expensive maintenance cost. The author concluded that it is costly to deploy and use a ledger database. However, in three separate articles, the authors agreed with the key concept of the advantages of a ledger database. Trusted data in a ledger database was stressed by Allen et al. (2019). Auditability is closely linked to the trusted data. Moreover, Raikwar et al. (2020) mentioned the importance of data immutability in a ledger database, which means data is tamper resistant. Likewise, Jacob et al. (2019) highlighted the promotion of data transparency in a ledger database. This means the visibility of data across the ledger and tracking each data changes. These advantages address the need for a ledger database. It revolves around the considerations of trust, immutability, and transparency. Overall, the benefits of using a ledger database outweighs the downside. And it can be noted that in Amazon QLDB, you pay only for what you use. Amazon based the payment on the following: writing of data, reading of data, transfer of data, and storage of data in the quantum ledger database.

Throughout the literature, the solutions provided in the related articles are a combination of a blockchain-based and ledger-based technology, so why not have the best of both worlds and keep the advantages offered by both technologies. Data control is important in an organizational setting, and it can be achieved by residing the data in the organization's domain, meaning it must be distributed centrally. Property record keeping is important in the real estate industry. It is in the best interest of all stakeholders, especially the property owner, to ensure reliable property records distributed centrally using a blockchain-based and ledger-based technology which corresponds to Amazon QLDB. Adopting this emerging technology makes the property records more trustworthy.

3.2 Tools and Techniques

Amazon Web Services

Amazon Web Services provides a wide variety of global cloud-based goods, computing, computing, databases, analytics, networking, cell phones, Tools, management tools, IoT, security, and business applications for developers: On-demand, available with pay-as-you-go pricing, in seconds. Out of Data Warehousing to management software, content distribution directories the facilities are available in AWS (Amazon Web Services).

Visual Studio

Visual studio code is a very powerful code-focused development environment expressly designed to make it easier to write web, mobile, and cloud applications using languages that are available to different development platforms and to support the application development lifecycle with a built-in debugger and with integrated support to the popular Git version control engine (Del Sole, 2018). With Visual Studio Code, you can work with individual code files or with structured file systems based on folders (Del Sole, 2018). It is a code-centric tool, which makes it easier to edit code files and folder-based project systems as well as writing cross- platform web and mobile applications over the most popular platforms, such as Node.js and .NET Core, with integrated support for a huge number of languages and rich editing features such as IntelliSense, finding symbol references, quickly reaching a type definition, and much more (Del Sole, 2018). After our research, as our project code will be folder-based, and Visual studio code has built in support for coding with C# we decided to use Visual Studio Code as our main IDE. Visual studio gives more flexibility for coding and advanced editing features and support for additional languages via extensibility makes it a perfect choice for our project. Overall, as per our project requirements, we need to use Amazon Web Services console to access Amazon QLDB and Amazon S3, and it gives more security, safety, and durability.

C# Program

Flood (2005) observed that C# supports both object-oriented and component-based approach. In object-oriented approach, the system is a collection of objects, and within the objects are classes, and within the classes are methods. For component-based approach, the system is a collection of components, and within the component are related functions and data. C# is also compatible with the programming language currently used by our client, Inndox, thus the team decided to use C#.

.Net Framework

For the framework, (Del Sole, 2018) talk about .Net to create applications that will run in the windows platform. And this framework supports the teams chosen programming language of C#.

The .Net framework is compatible with the client's development environment. therefore, we adopted the .Net framework in our development process.

3.3 Methodology

Waterfall Methodology

In Waterfall approach, all system development processes are carried out in a sequential order in which progress is seen as slowly flowing downward through the stages of: specifications, design of systems and applications, implementation and unit testing, integration and system testing, and operation and maintenance as examined by Curcio et al. (2018). It is natural to freeze sections of the development, such as the specification, in the waterfall approach. Problems are left for later resolution, ignored, or coded around, and some issues, such as poorly organized systems, can result as stated by Curcio et al. (2018).

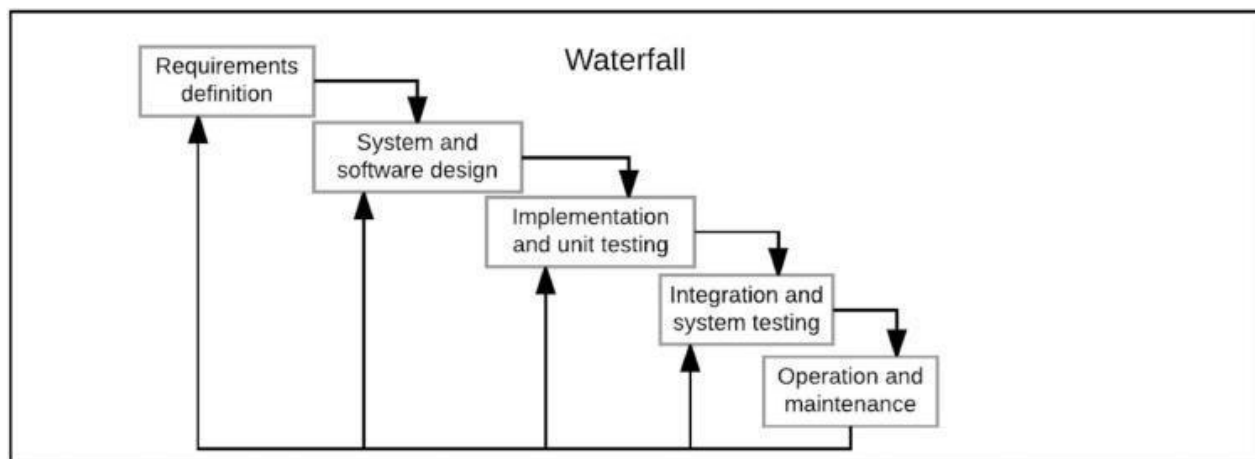


Figure 1- Waterfall Model (Source: Curcio (2018))

Agile Methodology

Agile software development generally operates with regular deliverables with tiny iterations, the process of development is dynamic. The specifications are initially identified by the customer and described in the format of a customer wish list; they are addressed every few weeks, better understood and reprioritized to determine the scope of the next iteration (Curcio et al., 2018).

The scrum system requires a scrum master, the owner of the product and the scrum squad. The key function of the Scrum Masters is to remove impediments. The scrum team is a cross- functional team consisting of developers, testers and other experts from different fields needed in development, contributing to a flexible and creative product that meets the customer's satisfaction as talked about by Srivastava et al. (2017).

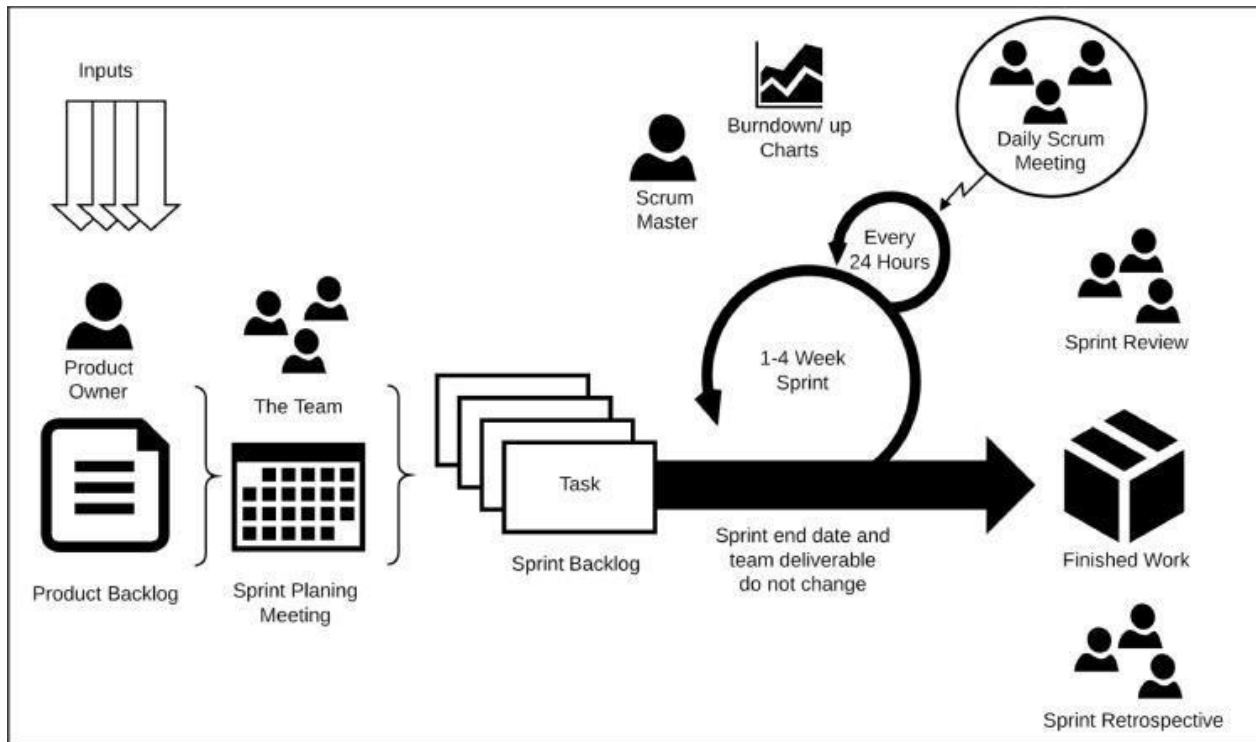


Figure 2 - Scrum Framework (Source: Curcio (2018))

The proposed scrum model aims to make scrum, over other agile methodologies, a more systematic methodology. The overall scrum cycle will prove to be complete in the sense of functionality and can accommodate changes within each iteration with separate treatment of conventional and creative tasks.

4. Project Approach

From our literature review we decided to use Scrum Agile method for our project approach. We will be using Agile as a methodology and Scrum as a framework for that methodology.

4.1 Agile Methodology/Scrum Framework

Based on our project requirements we decided to go with Agile methodology where we divided entire project into separate sprints. Agile development is done in iteration and as we planned to deliver the software in sprint, Scrum became our best choice. We had planned our software to be on 3 sprints. Because we used Agile scrum methodology even though there were changes in the requirement during the project development it was easier to adjust the changes. Average length of scrum project is around 11.6 weeks as explained by Scrum Alliance (2018), and it was perfectly suitable for us as our project time frame was also 12 weeks. During each weekly meeting we did project demonstration to our clients, supervisor and according to their feedback we moved our project forward. For each sprint we planned, designed, developed, and tested the functionalities.

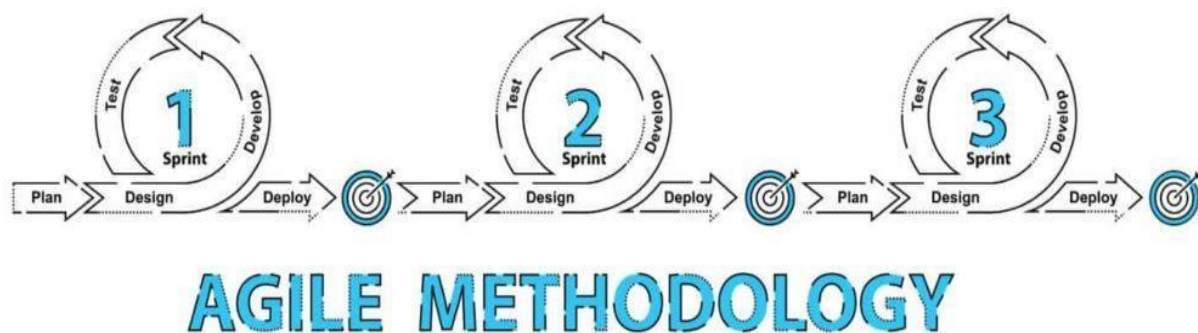


Figure 3 - Agile Methodology (Source: hub.packtpub.com)

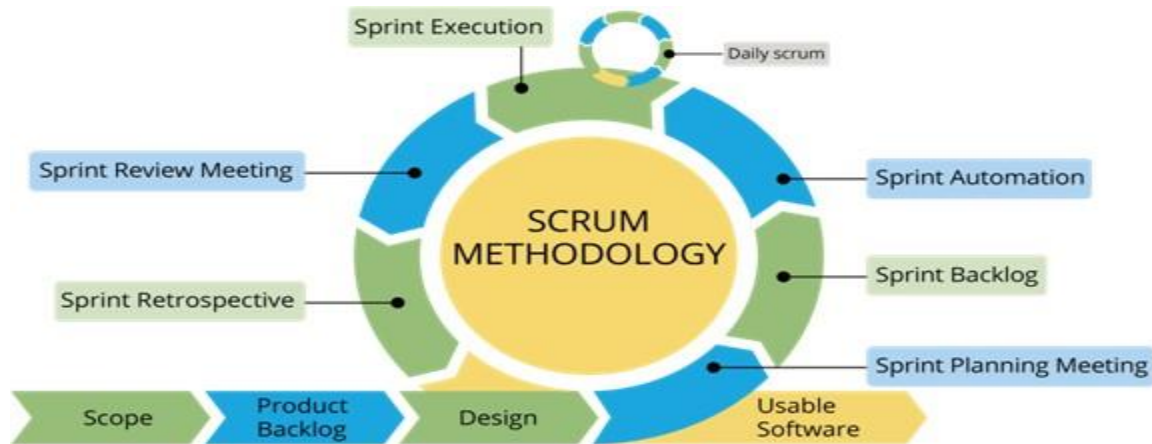


Figure 4 - Scrum Framework (Source: zaynabzahrablog.wordpress.com)

SPRINT SCHEDULE

Table 1 below describes the sprint schedule for the Inndox team.

Product Backlog	Requirement Priority	Weeks 1-6 (10 Aug - 04 Sep)	Weeks 7-8 (31 Aug - 18 Sep)	Weeks 8-10 (21 Sep - 03 Oct)	Weeks 11-12 (05 Oct - 16 Oct)
SPRINT 1					
Blockchain Framework Decision	High				
Blockchain-based Infrastructure - Test Environment	High				
Mapping of Business Process	Medium				
Create Test Cases and Test Data	Low				
SPRINT 2					
Create blockchain entities	High				
Create a sample API	Medium				
Test sample API	Low				
Create APIs	High				
Execute Test Cases	Medium				
Fix bugs and retest	Medium				
Create Master Test Report	Low				
SPRINT 3					
Blockchain-based Infrastructure - Production Environment	High				

Deploy API's in Production	High														
Monitor execution in production	Medium														
Fix production bugs	Medium														
Create Master Production Report	Low														

Table 1. Sprint Schedule

Revised User Stories

The revised user stories for the Inndox team are described in Table 2.

Requirement	User Story	Acceptance Criteria
Blockchain recommendation report. Research & report on blockchain technology specifically Ethereum, AWS (BaaS), Azure (BaaS) and Hyperledger and its suitability for enhancing the Inndox solution.	As a client, I want to know which blockchain technology is suited for our current technology, so that it can be compatible and be easily integrated in the current Inndox application system	Clients assess the recommendation report. Client makes the decision in which blockchain framework is to be used by Inndox company.
Minimum or no impact to current system of Inndox. Additional functionality of property records being stored in blockchain during a handover process,	As a client, I only want the blockchain technology to be integrated in our current database, so that front-end will not be impacted and will have minimum or no impact on back-end	Current functionality of Inndox system remains the same. During a handover process, property records are stored in AWS. In addition, it is also stored in blockchain.
Inndox customers signup and sign in, is secured in the blockchain.	As a client, I want to have a secured signup and sign in functionality in the blockchain network, so that user fraud and user identity theft is prevented	Only subscribed Inndox customers can participate in the blockchain process and are secured with the blockchain authentication protocol.
Inndox customer data (property records) are stored in blockchain, achieving immutability which ensures greater trust and transparency.	As a client, I want to have stored data in the database to be accessible as soon as command is given to do so, so that data is available for use.	After a handover process, property records are already stored in AWS. as well as in blockchain. An API is used to query the property records in the blockchain network.

Table 2. Revised user stories

4.2 Tools and Technique

From our research and literature review following were the tools and the technique that we used during our project development:

1. Amazon Web Services (AWS) Console.
2. Visual Studio Code as IDE (Integrated Development Environment).
3. C# as programming language, asp.net as framework.

We used Trello for project management, Slack for communication, zoom for online meeting and Confluence for document sharing as per the request of our client.

4.3 Project Development

From our literature review we decided to go with Scrum Agile framework. We divided our task into three sprints. We first designed our product backlog and sprint backlog. Then we did our sprint according to our sprint backlog and for each sprint after completion we reviewed it and did sprint retrospective. As for our whole software development phase, we divided into five phases, i.e.:

- 1) Analysis
- 2) Design
- 3) Development
- 4) Testing
- 5) Release

We divided each phases of our software development into three separate sprints before developing the actual Product.

1. Sprint 1

Sprint 1 runs on 10th of August 2020 up to 18th of September 2020. Product backlogs and stakeholder responsible are listed in which the high priorities are in the top of the list. See list below.

- Blockchain-based Framework (All stakeholders)
 - Identify the blockchain-based framework to be integrated to the Inndox business process.
- Blockchain-based Infrastructure - Test Environment (Project team)
 - Set up the environments for developing and testing.
- Mapping of Business Processes (Project team)
 - Mapping of Inndox business process identifying Inndox business process (participants, assets, transactions, and events).
- Create test cases and test data (Project team)
 - Identification of all possible user inputs and create a test plan for each. Create test data to be used for the identified test plan.

We focused mainly on research and analysis for our sprint 1.

a. Analysis

After the initial project description was handed to our team, all our team members together drafted upcoming week goal as our initial phase was of research. We then planned, according to what our research outcome we get, we will do project scheduling for each week and prepare our sprint goals. Our first task was to determine the optimal blockchain-based technology for Inndox. Through extensive research, discussion with our clients and with the Amazon tech personnel we chose Amazon QLDB as the best blockchain-based technology which can provide proper immutability, trust, and transparency to the user of Inndox. Our project schedule was pushed 1 week behind while finalizing Amazon QLDB. Amazon QLDB is a fully managed ledger database that provides a transparent, immutable, and cryptographically verifiable transaction log owned by a central trusted authority. Amazon QLDB can be used to track each application data change and maintains a complete and verifiable history of changes over time, per Amazon AWS (2020).

2. Sprint 2

For our sprint 2 we did the following task:

- Create blockchain-based entities (Project team)
 - Create participants, assets, transactions, and events in blockchain-based framework.
- Create a sample API (Project team)
 - Create a sample API that will process basic blockchain-based entities.
- Test sample API (Project team)
 - Test the blockchain-based network with minimal participants, assets, transactions, and events using the sample API and/or smart contract.
- Create APIs (Project Team)
 - Create APIs that will process the Inndox business data
- Execute Test Cases (Project Team)
 - Execute the test cases identified in sprint 1, using the created test plan and test data
- Fix bugs and retest (Project Team)

For failed test cases, analyze the bug and fix it, then re-ran the failed test case. Repeat this step until it passed.

We did most of our design phase in this sprint while also started with our development phase.

a. Design

Through analysis we became clear on our objectives. We were clear on the product that we will be developing and the outcome we can expect after developing it. But next task for us was to

design the desired features and operation in details. We designed how our ledger will look like and how the data will be stored in it. As for operation part we focused on:

- Table and its operation like insert, update, delete, etc.
- Verifying the data in the ledger through console.
- Verifying the data in the ledger through our code.
- Exporting the journal from QLDB to Amazon S3.

As Amazon QLDB was relatively new to us, we only designed these operations at the start of our project. These were the most crucial operations that we had to design. But just in case if any new minor requirements pop up during the development phase, we were confident to accommodate them as we were using Agile development method.

b. Development

To get started with QLDB first we were provided with an account on AWS to access AWS console by our client. We encountered a lot of authentication and access level error, which made it hard for us to properly run AWS console. Then later we created our own AWS account with proper authentication and got started with Amazon QLDB. PartiQL is the query language which can be used for QLDB. We learned PartiQL and created ledger, created table, updated table data. At the same time, we also upskilled ourselves on C# using which our first task was to create and connect QLDB driver. We used visual studio 2019, see Figure 5, and visual studio code respectively on two separate computer and completed the first task.

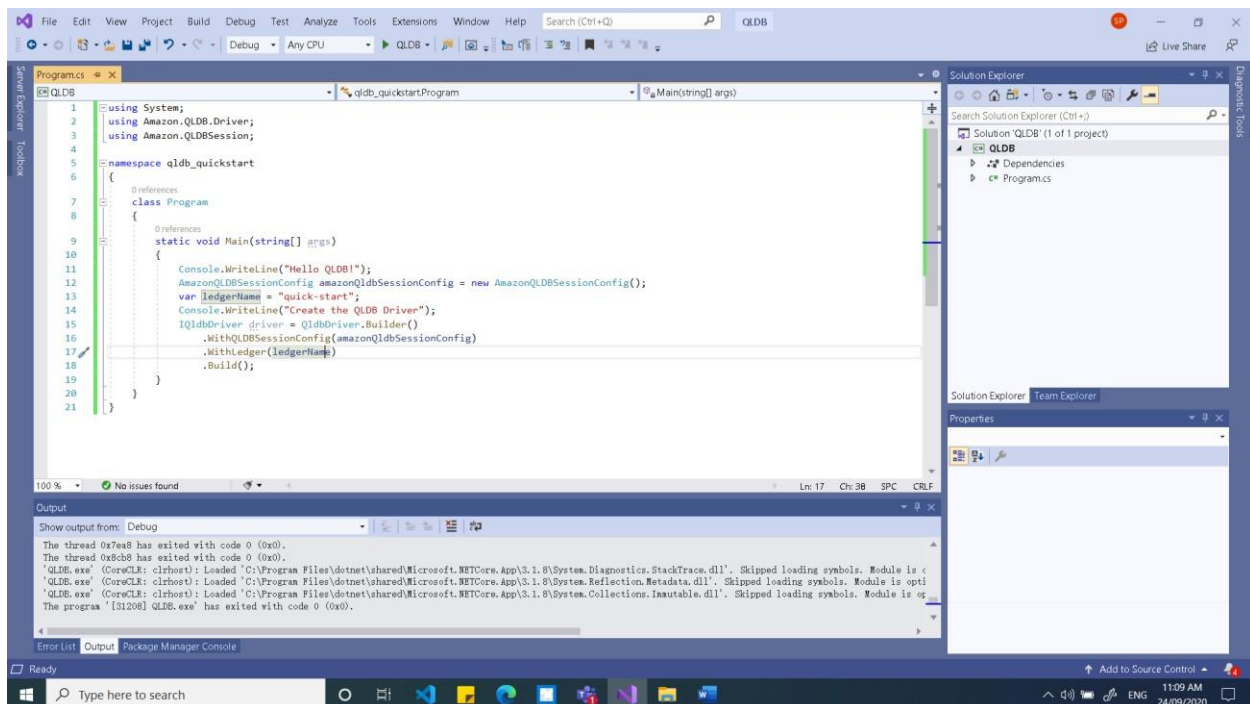


Figure 5 - Microsoft Visual Studio 2019

After that, our task was to create ledger and create tables into that ledger. We were also successful on inserting data on the table and update those data. As QLDB is a ledger to maintain the transaction log, we had to verify the integrity of the data after some transaction is done. On top of that as actual file is store in Amazon S3, we created a S3 bucket to link it with our QLDB. Our vision was to include all the details of account holder, properties related to that account holder and the metadata and file path of actual documents from S3. Our table on Amazon QLDB is shown in the Figure 6 below:

```
{
  PropertyId: "P001",
  CompanyId: "C001",
  HandoverDate: "2020-10-10T12:02:43.804Z",
  DocumentId: "D001",
  DocumentFileName: "D001",
  DocumentUploadedBy: "UploaderOfDocument001",
  DocumentPath: "FolderP001/D001.txt",
  DocumentMetadata: "d5752f16bc6be903f3d9358a6bc2c0bf",
  OwnerId: O002
}
```

Figure 6 – Code snippet showing table created on Amazon QLDB

As it can be seen our table consists of all information related to the property, documents related to the property, their Owner ID, document ID and most importantly the metadata related to that document that is crucial for maintaining data integrity is also stored.

Using the console, we first used digest to find out if the data integrity can be verified or not. It can be seen in the figure below for two different cases:

1. When the metadata is same after handover and the data integrity can be verified.
2. When the metadata is changed after handover without all party's consent then the verification process is failed.

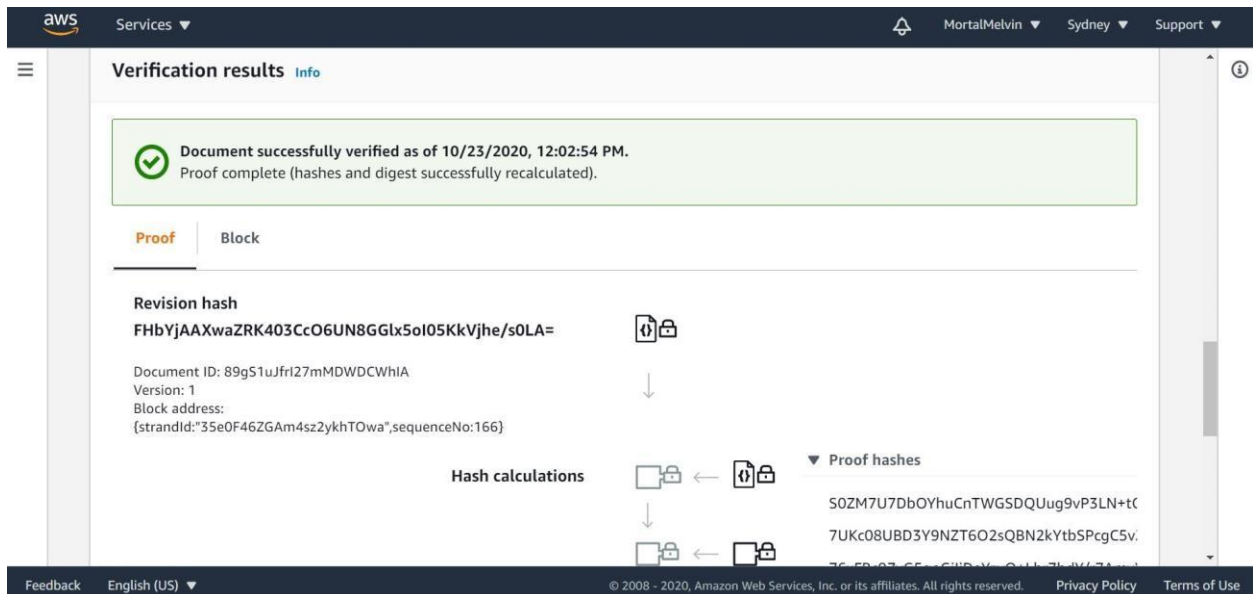


Figure 7 - Showing the data verification as true

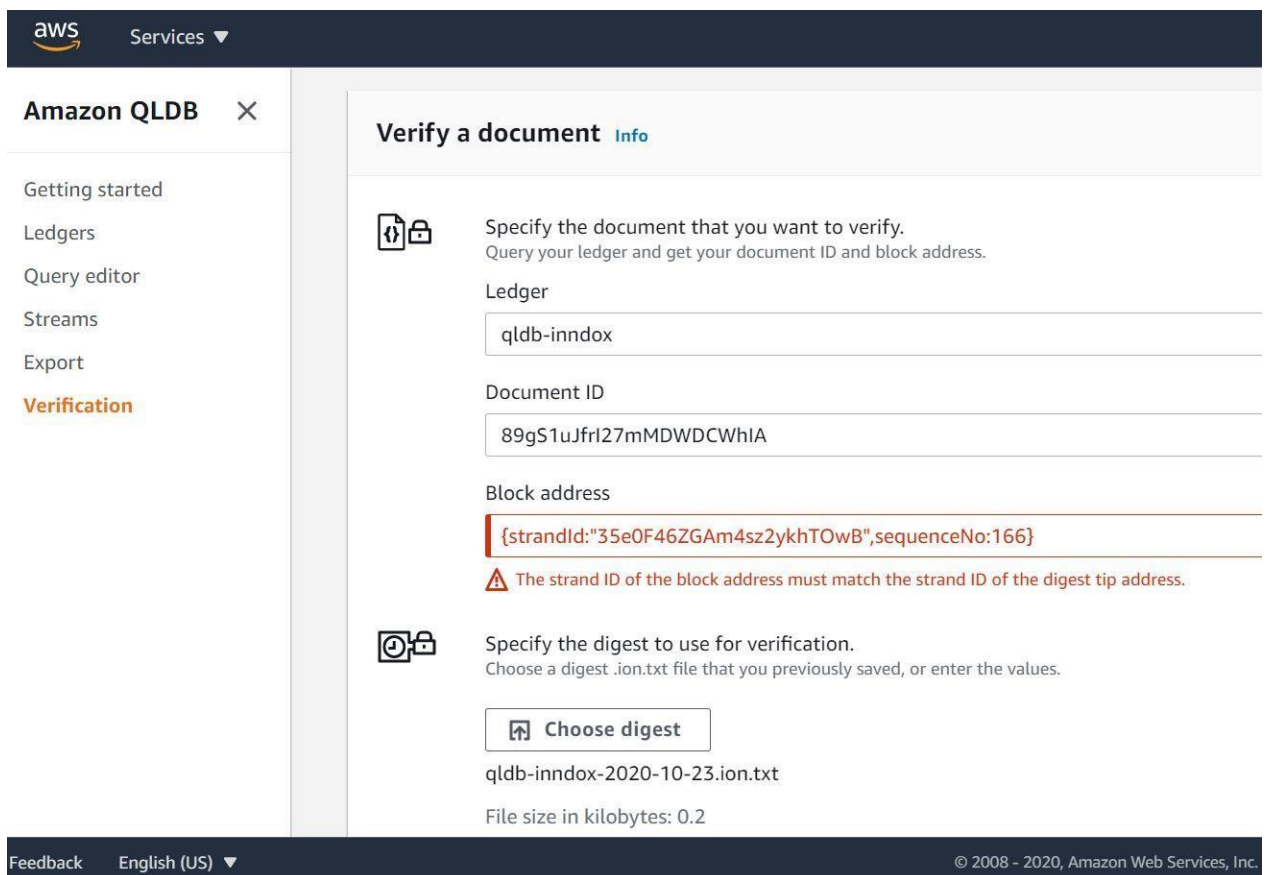


Figure 8 - Showing the data verification as false

As it can be seen in the Figure 7 the data verification is successful. And if the data verification failed, it can be seen in Figure 8. As we are developing a separate ledger for data transparency, whereas actual file is stored in Amazon S3 it may be time consuming and create difficulties for user to navigate from one database to another to access their ledger. This led us to find a way to export the journal from Amazon QLDB to Amazon S3, which we were able to. Using console, it was easier to do that but because of lack of IAM role we got stuck for about 3 days. By consulting with Amazon technical support finally we could rewrite the program to give appropriate IAM role to our User allowing us to export the journal from QLDB to S3. We can see that in the figures below:

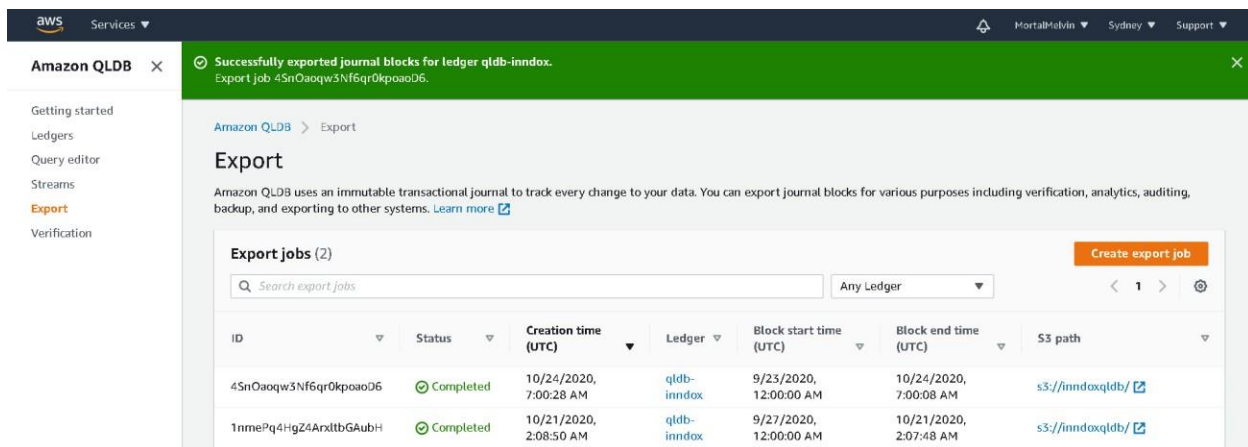


Figure 9 - Export job successful in QLDB.

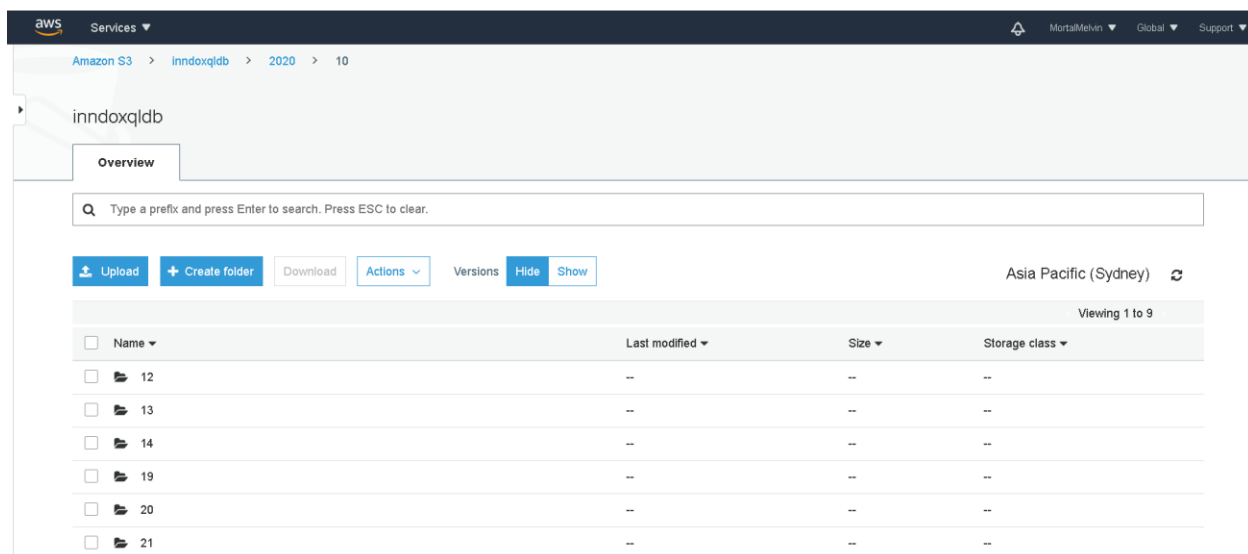


Figure 10 - Exported data in Amazon S3.

As in all these processes mentioned above creating a ledger, table and inserting data are done through code while verification process and storing the file path on Amazon S3 is done manually, our client gave us a new requirement to develop an API to make above mentioned process easier which was then transferred to Sprint 3.

3. Sprint 3

Sprint 3 occurs from 5th of October 2020 and finishes on 16th of October 2020. Product backlogs include:

- Blockchain-based Infrastructure - Production Environment (Project Team and Innadox developer)
 - Set up the environment for deployment
- Deploy API's in Production (Project Team and Innadox developer)
 - Deploy API's in Production, using Innadox actual business data
- Monitor execution in production (Project Team)
 - Check and record the results of the execution which used the actual business data
- Fix Production Bugs (Project Team)
 - Analyze the production bug and identify the production fix. Recreate the bug in test environment and retest until it passed. Deploy production fix.
- Create Master Production Report (Project Team)
 - Create a final report on the production execution and bug fixes

We continued with our development phase for sprint 3. We were given two new requirements by our client which are as follows:

1. By giving the input of any parameter like Property ID, Account ID, File name and Uploader ID getting the output as file path of the stored file, see figure 6.
2. Automatically check whether the given Etag of the file is found or not, and if it is found whether the found Etag is valid (as seen in Figure 11) or not (as seen in Figure 12) through the code.

We were able to perform both tasks mentioned above with the short time frame we were given. Only shortcomings that we had was we were not able to perform these tasks for multiple files all at once. As we had problem with Amazon QLDB access restrictions, we were not able to code inside the console to develop an API, so we wrote the code on Visual Studio code. Figures below shows the output we got after running our program for both tasks.

```
Create the inndox QLDB Driver
Querying a table
count: 1
QLDB Etag found: d5752f16bc6be903f3d9358a6bc2c0bf
Document selected
{
  DocumentMetadata: "d5752f16bc6be903f3d9358a6bc2c0bf"
}
QLDB Etag found given the folowing parameter: PropertyId = P
QLDB Etag: "d5752f16bc6be903f3d9358a6bc2c0bf"
  S3 Etag found given the folowing parameters: Bucketname =
  S3 Etag: "d5752f16bc6be903f3d9358a6bc2c0bf"

Processing of QLDB Etag and S3 Etag
=====
QLDB Etag:"d5752f16bc6be903f3d9358a6bc2c0bf"
  S3 Etag:"d5752f16bc6be903f3d9358a6bc2c0bf"
Etag is valid
PS C:\Users\flore> █
```

Figure 11 - Etag is valid after executing the program

```
Create the inndox QLDB Driver
Querying a table
count: 1
QLDB Etag found: d5752f16bc6be903f3d9358a6bc2c0bf
Document selected
{
  DocumentMetadata: "d5752f16bc6be903f3d9358a6bc2c0bf"
}
QLDB Etag found given the folowing parameter: PropertyId = P
QLDB Etag: "d5752f16bc6be903f3d9358a6bc2c0bf"
  S3 Etag found given the folowing parameters: Bucketname =
  S3 Etag: "dedb568fb495e253d049811b79bcb4ca"

Processing of QLDB Etag and S3 Etag
=====
QLDB Etag:"d5752f16bc6be903f3d9358a6bc2c0bf"
  S3 Etag:"dedb568fb495e253d049811b79bcb4ca"
Etag is invalid
PS C:\Users\flore> █
```

Figure 12 - Etag is invalid after it has been tampered with by executing the program

a. Testing

During the development phase we did manual testing for each functionality that we developed successfully. But most of the testing part was done on our sprint 3. We had created test cases and test plan during our sprint 1, using which we tested each functionality of our product. For most of the testing part, we did acceptance testing by comparing the expected result and the actual result. As for the acceptance criteria for each test cases it was determined according to the user requirements through user stories. We will be demonstrating our test cases and test result later in this report.

b. Product release

After successful tests of our product we released our product and demonstrated the features and functionality we were able to implement to our client. After the initial demonstration of our product during sprint 3 phase, our client gave us two more requirements as mentioned during development phase, which we implemented during the next week of development. We successfully released our final product after accommodating two new requirements given by our client. As per our client requirements, we were able to do data verification which will maintain data integrity and data transparency giving rise to trust among the related parties.

5. Product Demonstration

We have developed our product and handed it over to our client, Innadox. Our product has the following amazing features.

1. Creation of Table and index in Amazon QLDB.

A class called Create Table was created. Inside that class, a method called Add Table was also created. Table name and index needs to be specified so that it can be created. See sample code snippet below.

Code snippet:

```
// Creates the table and the index in the same transaction

Console.WriteLine("Creating the tables");

    string tableName = "Properties";

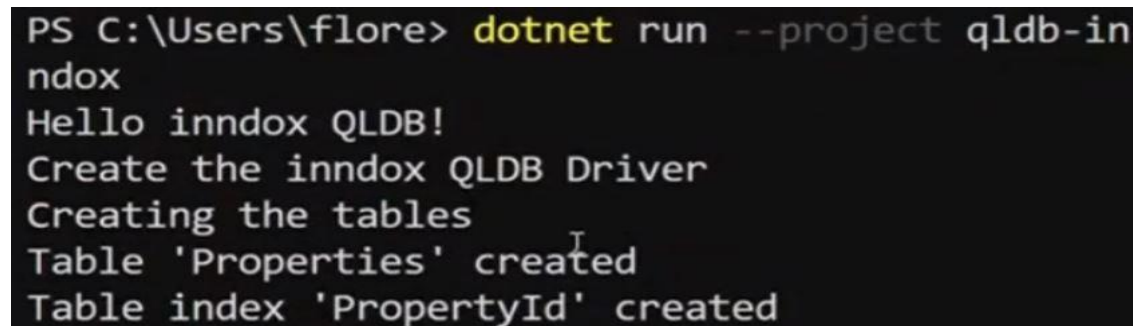
    string tableIndex = "PropertyId";

driver.Execute(txn =>
{
    txn.Execute($"CREATE TABLE {tableName}");

    txn.Execute($"CREATE INDEX ON {tableName}({tableIndex})");

});
```

Actual program run:



```
PS C:\Users\flores> dotnet run --project qlldb-inndox
Hello inndox QLDB!
Create the inndox QLDB Driver
Creating the tables
Table 'Properties' created
Table index 'PropertyId' created
```

Figure 13 - Table and index created

2. Connect to Amazon QLDB

A class called DriverQLDB was created. Inside that class, a method called Get Driver was also created. Specify the ledger name. Credentials with AWS private key and secret key also needs to be specified, as well as the AWS region, so that it can be connected to Inndox ledger in Amazon QLDB. See sample code snippet below.

Code snippet:

```
//Create driver for Amazon QLDB

Console.WriteLine("Create the Inndox QLDB Driver");

IQldbDriver driver = QldbDriver.Builder()

    .WithAWSCredentials(awsCredentials)

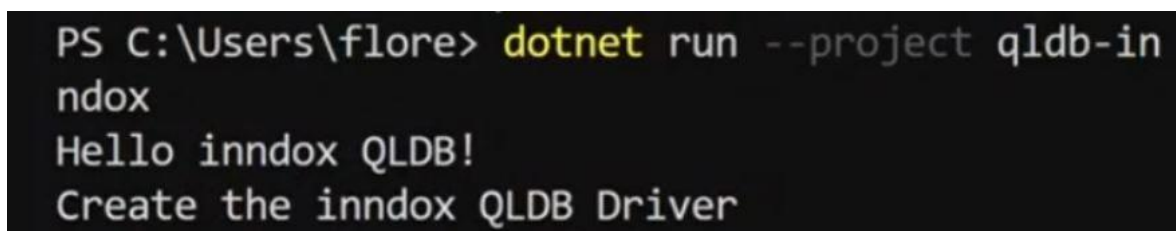
    .WithQLDBSessionConfig(amazonQldbSessionConfig)

    .WithLedger(ledgerName)

    .Build();

return driver;
```

Actual program run:



```
PS C:\Users\flores> dotnet run --project qlldb-inndox
Hello inndox QLDB!
Create the inndox QLDB Driver
```

Figure 14 - Connect to Amazon QLDB

3. Inserting the Inndox record in Amazon QLDB

A class called InsertTable was created. Inside that class, a method called AddDocument was also created. Specify the table name and the Inndox record that needs to be inserted. The program checks if the record exists, then it will return an error message. If the record does not exist, then it will insert the record. See sample code snippet below.

Code snippet:

```
        Console.WriteLine("Inserting a document");

        IIonValue insertResult = driver.Execute(txn =>

            {

                IResult result = txn.Execute($"INSERT INTO
{insertTableName} VALUE "+sb);

                foreach (var row in result)

                {

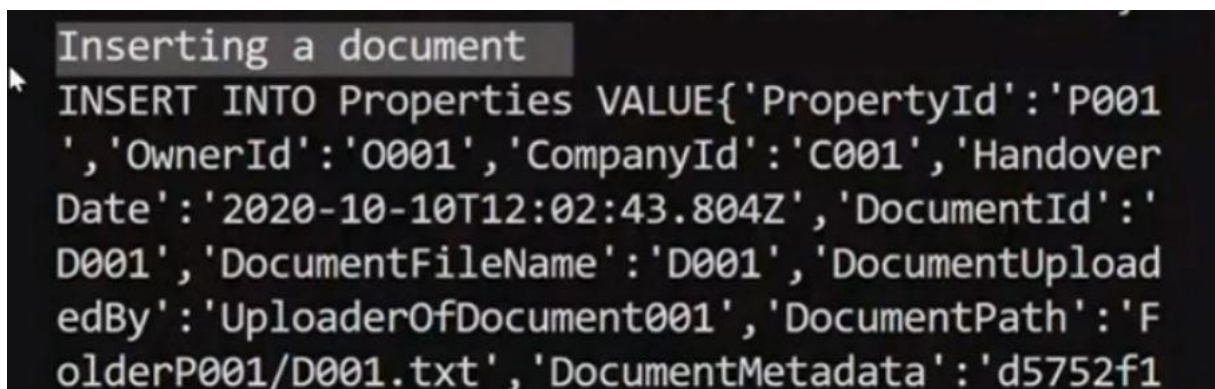
                    return row;

                }

                return null;

            });
```

Actual program run:



```
Inserting a document
INSERT INTO Properties VALUE{'PropertyId':'P001', 'OwnerId':'O001', 'CompanyId':'C001', 'Handover
Date':'2020-10-10T12:02:43.804Z', 'DocumentId':'D001', 'DocumentFileName':'D001', 'DocumentUploa
dedBy':'UploaderOfDocument001', 'DocumentPath':'FolderP001/D001.txt', 'DocumentMetadata':'d5752f1
```

Figure 15 - Inserting the record

4. Updating the Inndox record in Amazon QLDB

A class called InsertTable was created. Inside that class, a method called UpdateDocument was also created. Specify the tablename, Inndox record and the field that needs to be updated. The program checks if the record does not exist, then it will return an error message. If the record exists, then it will update the record. See sample code snippet below.

Code snippet:

```

        Console.WriteLine("Updating a document");

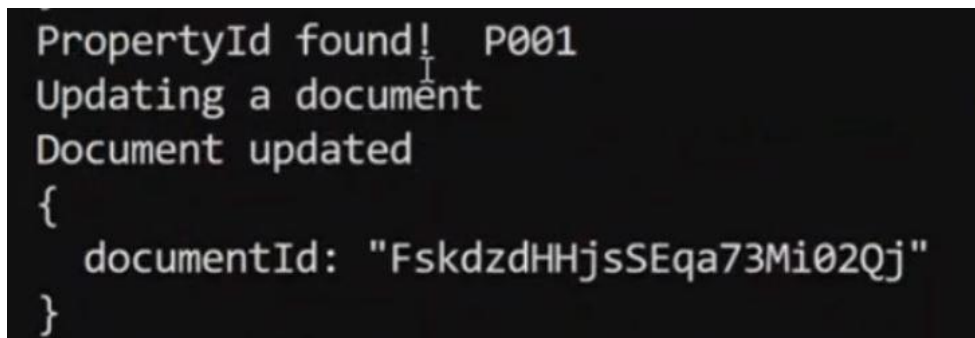
        IIonValue updateResult = driver.Execute(txn =>
        {
            IResult result = txn.Execute($"UPDATE {updateTableName}
SET OwnerId = ? WHERE PropertyId = ?", ionUpdateOwnerId,
ionUpdatePropertyId);

            foreach (var row in result)
            {
                return row;
            }

            return null;
        });

```

Actual program run:



```

PropertyId found! P001
Updating a document
Document updated
{
  documentId: "FskdzdHHjsSEqa73Mi02Qj"
}

```

Figure 16 - Updating the record

5. Connect to AWS S3

A class called SelectS3Etag was created. Inside that class, a method called ReadObjectDataAsync was also created. Credentials with AWS private key and secret key also needs to be specified, as well as the AWS region. Also specify the bucket name and the key, so that it can be connected to Inndox AWS S3. See sample code snippet below.

Code snippet:

```
// Create a client by providing security credentials.
using (client = new AmazonS3Client(tempCredentials,
bucketRegion))
{
    Amazon.S3.Model.GetObjectRequest request = new
Amazon.S3.Model.GetObjectRequest();

    {
        request.BucketName = "Inndoxqlldb";
        request.Key = "FolderP001/D001.txt";
    };
}
```

Actual program run:

```
Create the inndox QLDB Driver
Querying a table
count: 1
S3 Document Path found: FolderP001/D001.txt
Document selected
{
    DocumentPath: "FolderP001/D001.txt"
}
```

Figure 17 - Connect to AWS S3

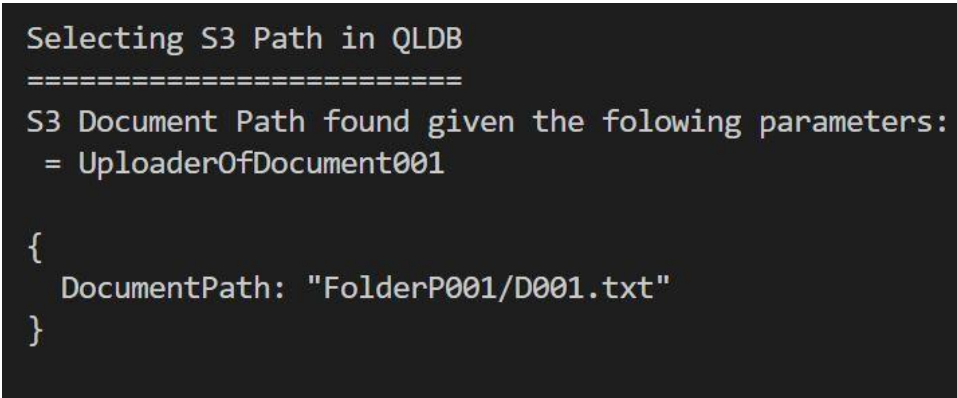
6. Return S3 path from QLDB

A class called SelectS3Path was created. Inside that class, a method called GetS3Path was also created. The following parameters must be specified: PropertyId, DocumentId, DocumentFileName, and DocumentUploadedBy. If the S3 path is not found, it will display an error message. If the S3 path is not found, it will return the S3 path from QLDB. See sample code snippet below.

Code snippet:

```
        IResult result = txn.Execute("SELECT DocumentPath FROM  
Properties WHERE PropertyId = ? AND DocumentId = ? AND DocumentFileName = ?  
AND DocumentUploadedBy = ?", ionSelectPropertyId, ionSelectDocumentId,  
ionSelectDocumentFileName, ionSelectDocumentUploadedBy);  
  
        foreach (var row in result)  
        {  
            count ++;  
            return row;  
        }  
  
        return null;  
    });
```

Actual program run:



```
Selecting S3 Path in QLDB  
=====  
S3 Document Path found given the following parameters:  
= UploaderOfDocument001  
  
{  
  DocumentPath: "FolderP001/D001.txt"  
}
```

Figure 18 - Return S3 path from QLDB

7. Get Etag from Amazon QLDB

A class called SelectQLDBEtag was created. Inside that class, a method called GetQLDBEtag was also created. The parameter PropertyId must be specified. If the Etag is not found, it will display an error message. If the Etag is found, it will return the Etag from AWS QLDB. See sample code snippet below.

Code snippet:

```
        IIonValue selectResult = driver.Execute(txn =>
        {
            // Selecting a document

            IIonValue ionSelectPropertyId =
            IonLoader.Default.Load(queryPropertyId);

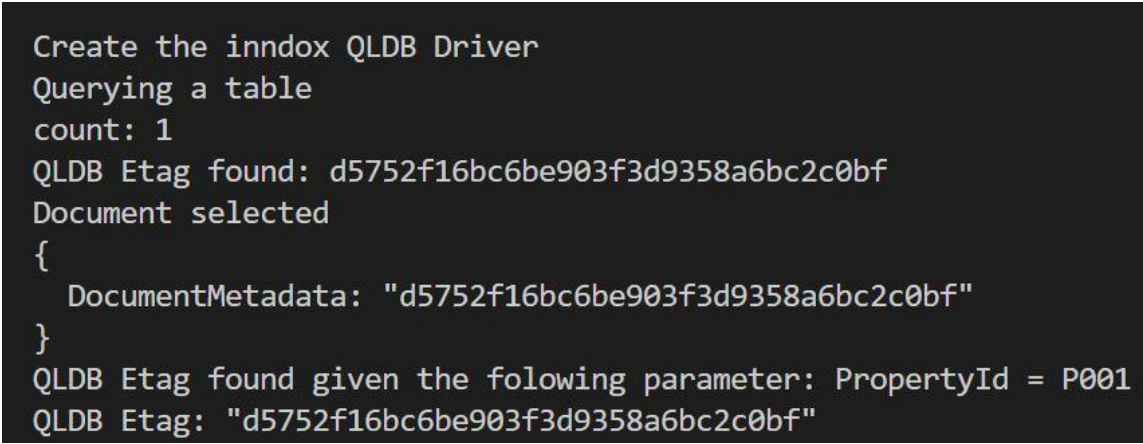
            IResult result = txn.Execute("SELECT DocumentMetadata FROM
            Properties WHERE PropertyId = ?", ionSelectPropertyId);

            foreach (var row in result)
            {
                count ++;

                return row;
            }

            return null;
        });
```

Actual program run:



```
Create the inndox QLDB Driver
Querying a table
count: 1
QLDB Etag found: d5752f16bc6be903f3d9358a6bc2c0bf
Document selected
{
  DocumentMetadata: "d5752f16bc6be903f3d9358a6bc2c0bf"
}
QLDB Etag found given the folowing parameter: PropertyId = P001
QLDB Etag: "d5752f16bc6be903f3d9358a6bc2c0bf"
```

Figure 19 - Get Etag from Amazon QLDB

8. Get Etag from AWS S3

A class called SelectS3Etag was created. Inside that class, a method called ReadObjectDataAsync was also created. Credentials with AWS private key and secret key also needs to be specified, as well as the AWS region. Also specify the bucket name and the key. If the Etag is not found, it will display an error message. If the Etag is found, it will return the Etag from AWS S3. See sample code snippet below.

Code snippet:

```

        using (GetObjectResponse response = await
client.GetObjectAsync(request))

        using (Stream responseStream = response.ResponseStream)

        using (StreamReader reader = new
StreamReader(responseStream))

        {

            string etag = response.ETag;

            var S3Etag = etag;

            S3.eTag = etag;

            Console.WriteLine($"  S3 Etag found given the
following parameters: Bucketname = {request.BucketName}, Key={request.Key}");

            Console.WriteLine($"  S3 Etag: {S3Etag}");

        }

```

Actual program run:

```

S3 Etag found given the following parameters: Bucketname = inndoxqldb, Key=FolderP001/D001.txt
S3 Etag: "d5752f16bc6be903f3d9358a6bc2c0bf"

```

Figure 20 - Get Etag from AWS S3

9. Process Etag

The main program called 'Program' was created. Inside that program, the Etag were processed. If the Etag in AWS S3 is not found, it will display a not found message. If the Etag from both AWS S3 and QLDB matched, then it will display that the Etag is valid. If the Etag from both AWS S3 and QLDB does not matched, then it will display that the Etag is invalid. See sample code snippet below.

Code snippet:

```
//Process Etags of QLDB and S3

Console.WriteLine("\n");

Console.WriteLine("Processing of QLDB Etag and S3 Etag");

Console.WriteLine("=====");

Console.WriteLine($"QLDB Etag:{QLDB.eTag}");

Console.WriteLine($"  S3 Etag:{S3.eTag}");


if (S3.eTag == null)

    Console.WriteLine("Etag not found in S3");
else
{
    if (QLDB.eTag == S3.eTag)

        Console.WriteLine("Etag is valid");
    else

        Console.WriteLine("Etag is invalid");
}
```

Actual program run:

```
Processing of QLDB Etag and S3 Etag
=====
QLDB Etag:"d5752f16bc6be903f3d9358a6bc2c0bf"
  S3 Etag:"d5752f16bc6be903f3d9358a6bc2c0bf"
Etag is valid
PS C:\Users\flore> █
```

Figure 21 - Process Etag

6. Errors and Resolution

A lot of time is spent in solving software errors that appears during development. A resilient mind is needed in fixing software bugs. Solving it in a panic mode just introduces more bugs in the long run. First step is to identify which line of code is causing the problem. Second step is to understand why the error is happening. Third step is to apply obvious and common-sense fix. Fourth step would be to ask technical support by opening a case in AWS support center, if applicable. And if the fix is not working, we are in a digital age where that same bug has been encountered by other developers in the world and fourth step is, we need to google and understand how they fixed it. Last and final step is to apply the fix. In this section, errors encountered, and its resolution are discussed.

6.1. Authorization Error in Creating Tables

Actual error displayed: User: arn:aws:iam::152028372420:user/BlockchainUser is not authorized to perform: qlldb:ExecuteStatement on resource: arn:aws:qlldb:ap-southeast-2:152028372420:ledger/InndoxLedger

Steps for resolution: Contacted AWS technical support

Solution: Attached the correct policy of AmazonQLDBConsoleFullAccess to user Blockchain User

6.2. Error in Building REST API

Actual error displayed:



⊗ User: arn:aws:iam::152028372420:user/BlockchainUser is not authorized to perform: apigateway:GET on resource: arn:aws:apigateway:ap-southeast-2::/account

Figure 22 - API Gateway error

Steps for resolution: Contacted AWS technical support

Solution: Ask David, Inndox account administrator to add IAM permissions to user Blockchain User to perform the needed actions on API Gateway. In the end, the team decided that it was easier to create our own personal AWS account and attach needed policy.

6.3. Configuration Error

Actual error displayed: Unhandled exception. Amazon.Runtime.AmazonClientException: No RegionEndpoint or ServiceURL configured at Amazon.Runtime.ClientConfig.Validate()

Steps for resolution: Googled the error

Solution: Applied code fix

Code fix snippet:

```
var awsCredentials = new BasicAWSCredentials("<AWS Private  
Key>", "<AWS Secret Key>");  
Console.WriteLine("Create the QLDB Driver");  
IQldbDriver driver = QldbDriver.Builder()  
.WithAWSCredentials(awsCredentials)
```

6.4. Credential Error

Actual error displayed: Unhandled exception. Amazon.Runtime.AmazonServiceException: Unable to get IAM security credentials from EC2 Instance Metadata Service.

Steps for resolution: Googled the error

Solution: Applied code fix

Code fix snippet:

```
amazonQldbSessionConfig.RegionEndpoint  
Amazon.RegionEndpoint.GetBySystemName("ap-southeast-2");
```

7. Testing Phase

An incremental approach to testing requires Agile testing. Features are tested in this kind of software testing as they are built. In Agile development method, testing needs to occur early and regularly. So, instead of waiting for production to be completed before testing starts, when features are introduced, testing happens continuously. Much like user stories, tests are prioritized. In an iteration, testers try to get through as many experiments as they can. Adding automated testing software can assist testers get through more of the backlog of testing.

User Acceptance Testing (UAT) is the final stage of the testing process for applications and has always been regarded as a very important stage. Real software users test the software during UAT to ensure that, as defined, it can handle necessary tasks in real-world scenarios. For Blockchain-based software, we used the acceptance testing procedure.

There are several test cases and test plan covered in this project is following:

Test Case ID: 01

Connect to Amazon QLDB

Purpose: The purpose of this test script is to validate that the developer can connect with Amazon QLDB.

Step	Description	Test Steps	Expected Results	Actual Results	Status
1	Prerequisite: Developer should have the appropriate IAM user and policy attached.	Go to the Website of Amazon QLDB Read the appropriate IAM user policy	Amazon QLDB Console Full Access policy should be attached to username Blockchain User	Amazon QLDB Console Full Access policy attached to username Blockchain User	Pass
2	Login to the AWS IAM (Identity Access Management)	Enter User ID and Password	User should login into an application	Login Successfully	Pass
3	Navigate to the AWS service: Amazon QLDB	Run the various services of Amazon QLDB	Amazon QLDB panel should be displayed	Amazon QLDB panel displayed	Pass

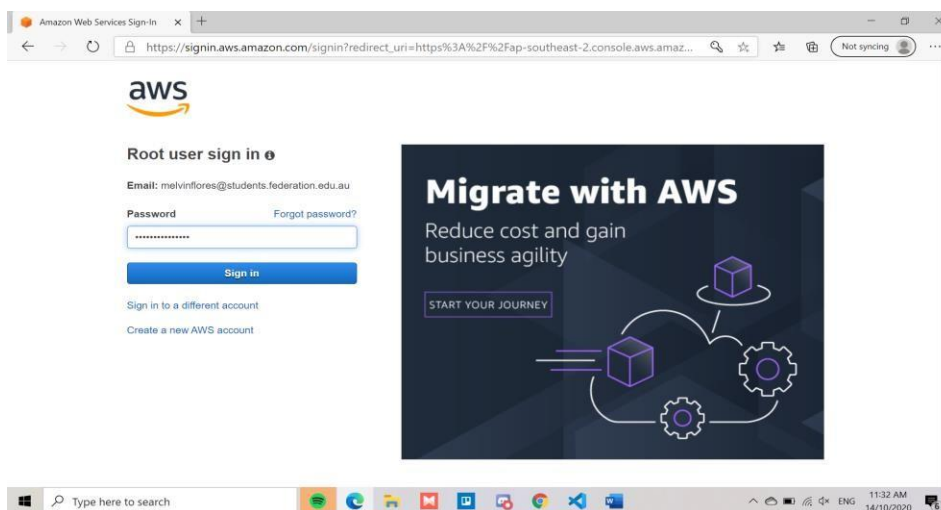


Figure 23 - Connect to Amazon QLDB

Test Case ID: 02

Connect to AWS S3

Purpose: The purpose of this test script is to validate that the developer can connect with AWS S3.

Step	Description	Test Steps	Expected Results	Actual Results	Status
1	Prerequisite: Developer should have the appropriate IAM user and policy attached.	Go to the Website of Amazon S3 Read the appropriate IAM user policy	AWS S3 Console Full Access policy should be attached to username Blockchain User	AWS S3 Console Full Access policy attached to username Blockchain User	Pass
2	Login to the AWS IAM (Identity Access Management)	Enter User ID and Password	User should login into an application	Login Successfully	Pass
3	Navigate to the AWS service: AWS S3	Run the various services of AWS S3	AWS S3 panel should be displayed	AWS S3 panel displayed	Pass

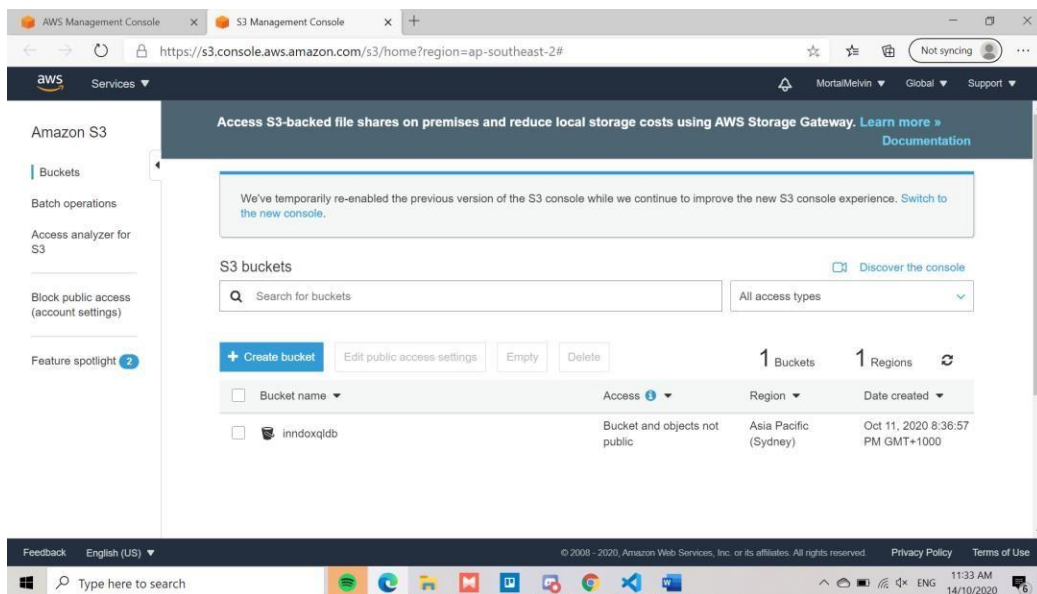


Figure 24 - Connect to AWS S3

Test Case ID: 03

Create Inndox ledger in Amazon QLDB

Purpose: The purpose of this test script is to validate that the developer can create Inndox ledger

Step	Description	Test Steps	Expected Results	Actual Results	Status
1	Navigate to the Create ledger panel	Go the Amazon QLDB	Create ledger panel should be displayed	Create ledger panel displayed	Pass
2	Populate ledger information	Check the ledger information	Ledger information should be populated	Ledger information populated	Pass
3	Ledger created with the ledger information	Insert the name of ledger	Ledger should be created	Ledger created	Pass

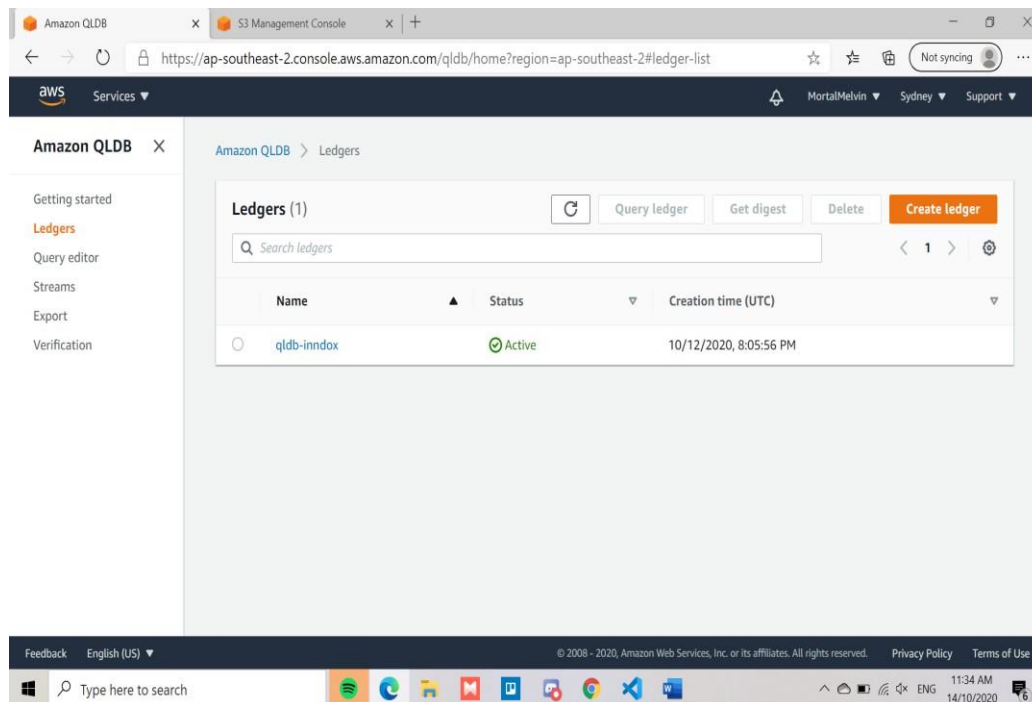


Figure 25 - Create Inndox ledger in Amazon QLDB

Test Case ID: 04

Create tables in the Inndox ledger in Amazon QLDB

Purpose: The purpose of this test script is to validate that the developer can create the tables inside the Inndox ledger in Amazon QLDB.

Step	Description	Test Steps	Expected Results	Actual Results	Status
1	Navigate to the Query editor panel	Go to the Amazon QLDB and open ledger	Query editor panel should be displayed	Query editor panel displayed	Pass
2	Choose the Inndox ledger information	Select ledger information	Inndox ledger should be chosen	Inndox ledger chosen	Pass
3	Issue the create table statement in the query editor and hit run	Insert the query in query editor panel	Table should be created	Table created	Pass

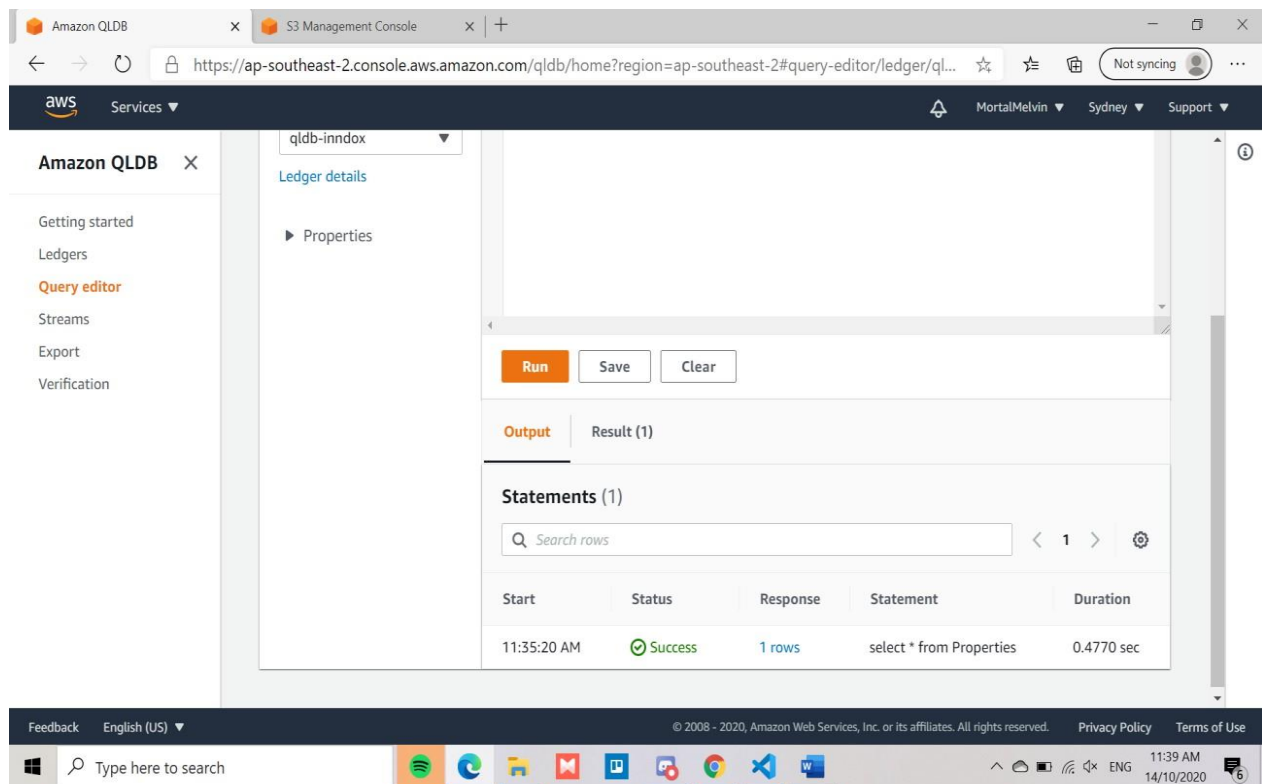


Figure 26 - Create tables in the Inndox ledger in Amazon QLDB

Test Case ID: 05

Insert the data in the tables of the Inndox ledger in Amazon QLDB

Purpose: The purpose of this test script is to validate that the developer can insert the data in the tables inside the Inndox ledger in Amazon QLDB.

Step	Description	Test Steps	Expected Results	Actual Results	Status
1	Navigate to the Query editor panel	Go to the Amazon QLDB and open ledger	Query editor panel should be displayed	Query editor panel displayed	Pass
2	Choose the Inndox ledger information	Select Inndox Ledger	Inndox ledger should be chosen	Inndox ledger chosen	Pass
3	Insert the data into table	Insert the data into table	Data should be inserted	Data inserted	Pass

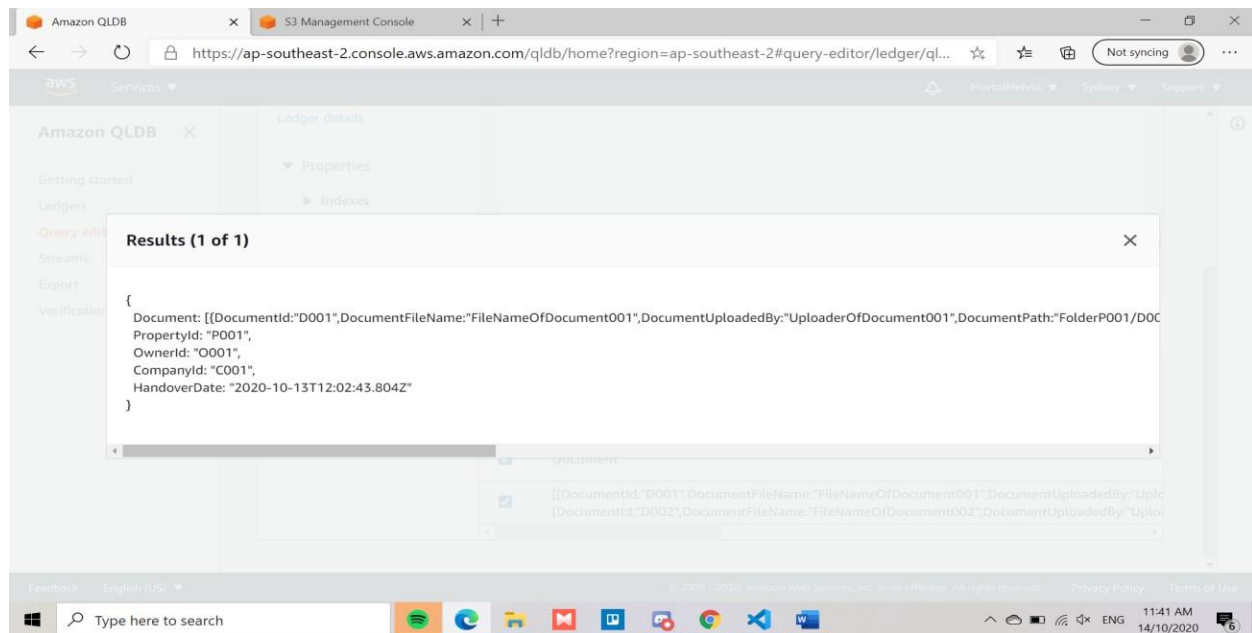


Figure 27 - Insert the data in the tables of the Inndox ledger in Amazon QLDB

Test Case ID: 06

Create a connection using AWS console, between our backend (Amazon QLDB) and AWS S3

Purpose: The purpose of this test script is to validate that the developer can create the API gateway between Amazon QLDB and AWS S3

Step	Description	Test Steps	Expected Results	Actual Results	Status
1	Navigate to the Amazon API Gateway panel	Go to the Amazon API Gateway panel	Amazon API Gateway panel should be displayed	Amazon API Gateway panel displayed	Pass
2	Create an API endpoint between Amazon QLDB and AWS S3 by populating the required configuration	Insert the select coding to develop API between Amazon QLDB and AWS S3	API endpoint between Amazon QLDB and AWS S3 should be created	API endpoint between Amazon QLDB and AWS S3 created	Pass

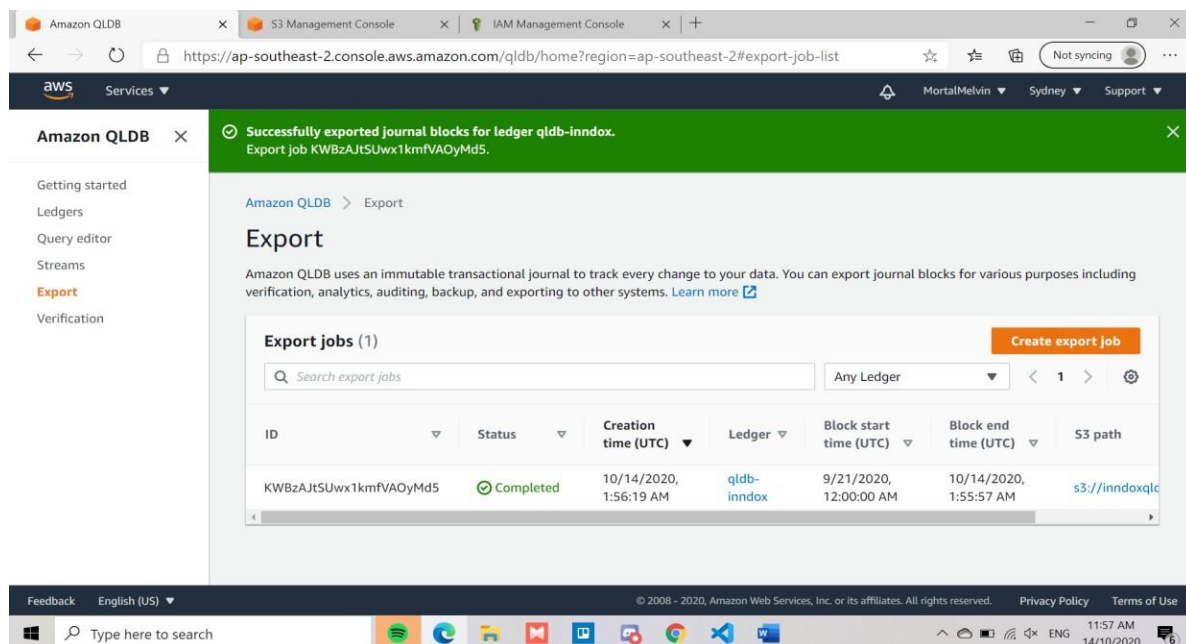


Figure 28 - Create a connection between our backend (Amazon QLDB) and AWS S3

Test Case ID: 07

Upload the folder/files in AWS S3

Purpose: The purpose of this test script is to validate that the developer can store the folder/files.

Step	Description	Test Steps	Expected Results	Actual Results	Status
1	Developer successfully executed the POST feature of the AMAZON API gateway	Go to Amazon API	Amazon API should be executed	Amazon API executed	Pass
2	Check if the folder/files exist in AWS S3	Check if the folder/files exist in AWS S3	Folder/files should exist in AWS S3	Folder/files exist in AWS S3	Pass
3	Apply the object lock configuration in the folder/files	Upload the file in S3 with object lock configuration	Object lock configuration in the folder/files should be applied	Object lock configuration in the folder/files is applied	Pass

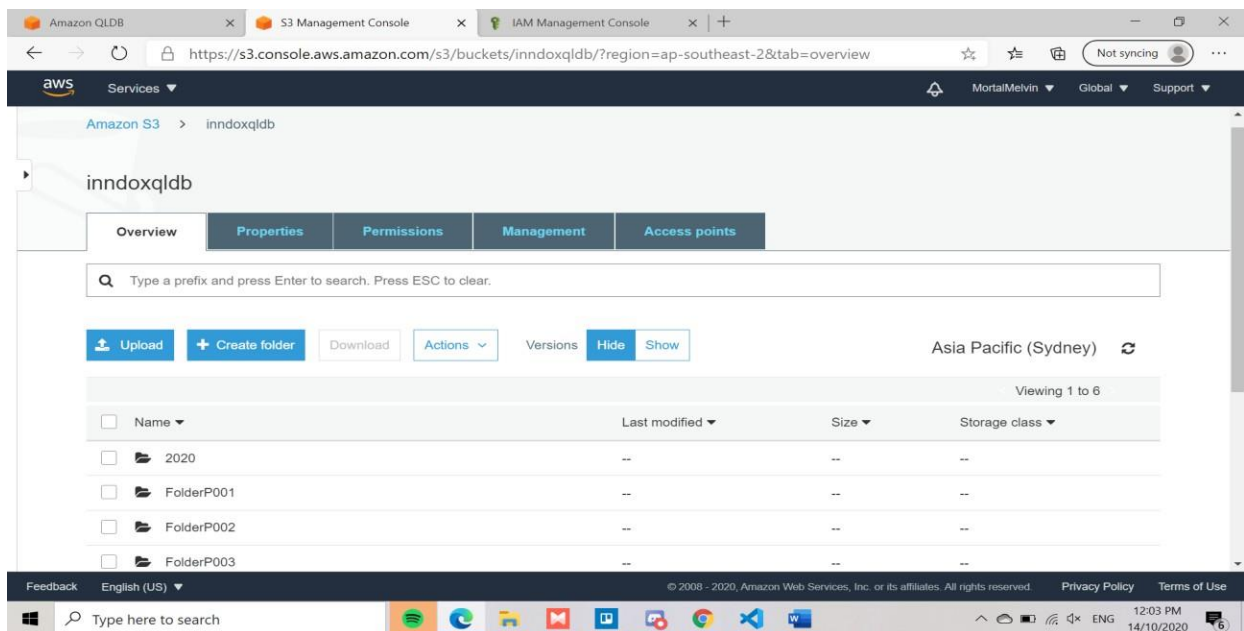


Figure 29 - Upload the folder/files in AWS S3

Test Case ID: 08

Validate the S3 Return Path

Purpose: The purpose of this test script is to validate that the developer can verify the correctness of the returned S3 path folder/files.

Step	Description	Test Steps	Expected Results	Actual Results	Status
1	Navigate to the AWS S3 panel	Go to the AWS S3 panel	AWS S3 panel should be displayed	AWS S3 panel displayed	Pass
2	Check if the folder/files exist in AWS S3	Go to the folder section AWS S3	Folder/files should exist in AWS S3	Folder / Files exist in AWS S3	Pass
3	Specify the document that you want to verify and check its attributes	Select the documents and verify	Document is specified and attribute of the documents should be displayed.	Document is specified and attribute of the documents displayed	Pass
4	Check the attribute 'Key' of the document in AWS S3	Select the attributes in AWS S3	The attribute 'Key' shows the return path, which is the folder location of the document	The attribute 'Key' shows the return path, which is the folder location of the document	Pass
5	Run the C# program. The S3 return path was correctly retrieved from QLDB.	Select C# program and go to the S3 path	Returned S3 path from QLDB by the C# program corresponds to the 'Key' attribute in AWS S3	Returned S3 path from QLDB by the C# program corresponds to the 'Key' attribute in AWS S3	Pass
6	Specify the document that you want to verify and provide document details.	Select the documents	Document is specified and document details should be provided.	Document is specified and document details provided	Pass

```
PropertyId found! P001
Updating a document
Document updated
{
  documentId: "HgXAx70kLq056KZRbaDBf4"
}
Create the inndox QLDB Driver
Querying a table
count: 1
S3 Document Path found: FolderP001/D001.txt
Document selected
{
  DocumentPath: "FolderP001/D001.txt"
}
```

Figure 30- Validate the S3 Return Path

Test Case ID: 09

Validate the Etag if Not found

Purpose: The purpose of this test script is to validate that the developer can verify that the Etag is not found

Step	Description	Test Steps	Expected Results	Actual Results	Status
1	Check if the folder/files exist in AWS S3	Go to the folders in AWS S3	Folder/files should exist in AWS S3	Folder/files exist in AWS S3	Pass
2	Specify the document that you want to verify	Select the documents	Document specified should not exist in AWS S3	Document specified does not exist in AWS S3	Pass
3	Run the C# program. The S3 Etag was not found in S3	Select C# program and run S3	S3 Etag was not found in S3 and error message should be displayed	S3 Etag was not found in S3 and error message displayed	Pass

```

Create the inndox QLDB Driver
Querying a table
count: 1
QLDB Etag found: d5752f16bc6be903f3d9358a6bc2c0bf
Document selected
{
  DocumentMetadata: "d5752f16bc6be903f3d9358a6bc2c0bf"
}
QLDB Etag found given the following parameter: PropertyId = P001
QLDB Etag: "d5752f16bc6be903f3d9358a6bc2c0bf"
S3 Etag:
S3 Etag not found
Error encountered ***. Message: 'The specified key does not exist.'

Processing of QLDB Etag and S3 Etag
=====
QLDB Etag: "d5752f16bc6be903f3d9358a6bc2c0bf"
S3 Etag:
Etag not found in S3
PS C:\Users\flores>

```

Figure 31- Validate the Etag if Not found

Test Case ID: 10

Validate the Etag if valid

Purpose: The purpose of this test script is to validate that the developer can verify that the Etag is valid.

Step	Description	Test Steps	Expected Results	Actual Results	Status
1	Check if the folder/files exist in AWS S3	Go to the folders in AWS S3	Folder/files should exist in AWS S3	Folder/files exist in AWS S3	Pass
2	Specify the document that you want to verify	Select the documents	Document should be specified exist in AWS S3	Document specified exist in AWS S3	Pass
3	Run the C# program. The Etag was correctly retrieved from both AWS S3 and QLDB.	Select C# program and run in AWS S3 and QLDB	Etag should correctly retrieved from both AWS S3 and QLDB	Etag correctly retrieved from both AWS S3 and QLDB	Pass
4	The Etag correctly matched from both AWS S3 and QLDB.	Check Etag	Etag should correctly matched from both AWS S3 and QLDB. Valid Etag message	Etag correctly matched from both AWS S3 and QLDB. Valid Etag message	Pass

			should be displayed	displayed	
--	--	--	------------------------	-----------	--

```
Create the inndox QLDB Driver
Querying a table
count: 1
QLDB Etag found: d5752f16bc6be903f3d9358a6bc2c0bf
Document selected
{
  DocumentMetadata: "d5752f16bc6be903f3d9358a6bc2c0bf"
}
QLDB Etag found given the folowing parameter: PropertyId = 
QLDB Etag: "d5752f16bc6be903f3d9358a6bc2c0bf"
  S3 Etag found given the folowing parameters: Bucketname = 
  S3 Etag: "d5752f16bc6be903f3d9358a6bc2c0bf"

Processing of QLDB Etag and S3 Etag
=====
QLDB Etag:"d5752f16bc6be903f3d9358a6bc2c0bf"
  S3 Etag:"d5752f16bc6be903f3d9358a6bc2c0bf"
Etag is valid
PS C:\Users\flore>
```

Figure 32- Validate the Etag if valid

Test Case ID: 11

Validate the Etag if invalid

Purpose: The purpose of this test script is to validate that the developer can verify that the Etag is invalid.

Step	Description	Test Steps	Expected Results	Actual Results	Status
1	Check if the folder/files exist in AWS S3	Go to the folder in AWS S3	Folder/files should exist in AWS S3	Folder/files should exist in AWS S3	Pass
2	Specify the document that you want to verify	Select the documents	Document should be specified exist in AWS S3	Document specified exist in AWS S3	Pass
3	Run the C# program. The Etag were correctly retrieved from both AWS S3 and QLDB.	Select the C# program and run AWS S3 and QLDB	Etags should correctly retrieved from both AWS S3 and QLDB	Etags correctly retrieved from both AWS S3 and QLDB	Pass
4	The Etag did not correctly matched from both AWS S3 and QLDB	Check Etag in AWS S3 and QLDB	There was a mismatch between the Etags from both AWS S3 and QLDB. Invalid Etag message should be displayed	There was a mismatch between the Etag from both AWS S3 and QLDB. Invalid Etag message displayed	Pass


```
Create the inndox QLDB Driver
Querying a table
count: 1
QLDB Etag found: d5752f16bc6be903f3d9358a6bc2c0bf
Document selected
{
  DocumentMetadata: "d5752f16bc6be903f3d9358a6bc2c0bf"
}
QLDB Etag found given the following parameter: PropertyId = 1
QLDB Etag: "d5752f16bc6be903f3d9358a6bc2c0bf"
S3 Etag found given the following parameters: Bucketname = inndox
S3 Etag: "dedb568fb495e253d049811b79bcb4ca"

Processing of QLDB Etag and S3 Etag
=====
QLDB Etag:"d5752f16bc6be903f3d9358a6bc2c0bf"
S3 Etag:"dedb568fb495e253d049811b79bcb4ca"
Etag is invalid
PS C:\Users\flores>
```

Figure 33- Validate the Etag if invalid

8. Applications, limitations, and future recommendations

In this topic we will be discussing about the application, limitations, and future improvements of our product.

8.1 Application of the product

This product can be used by Inndox to protect the integrity of their documents and to maintain data transparency. It can be done in three steps:

- a. Document: Inndox when working with their client can take the property documents that needs to be stored.
- b. Amazon S3: Those property documents taken by Inndox can be stored in Amazon S3 for the security purposes. Unique Etag is generated for each document.
- c. Amazon QLDB: For maintaining data integrity and transparency unique Etag of each document is stored along with Document Owner ID, Property ID, Document S3 file path. Now user can use our program for data verification whenever there is any need, which will protect data integrity. As for data transparency the unique Etag can be shared with all the related parties to that document like, property owner, builder and Inndox themselves. If anybody changes the document without other parties' consent than through data verification it can be known. This will maintain data transparency and increase trust among all the parties.

8.2 Limitation of our project

Even though the project was deemed mostly successful as we were able to complete our main requirement, i.e. being able to verify the integrity of the document and maintaining proper transaction log for properties owner, there were still some limitations to our project.

- Our product can only handle single document at one time, for verification process and not the multiple files at once.
- As Amazon QLDB is relatively new, there was not enough guidance for us to develop an API in AWS console itself. Doing so would have made this software a lot easier for client to work with.

8.3 Challenges

We faced lots of challenges while completing out project. First challenge we faced was to research the proper blockchain based database suitable for our client. Given, the initial user requirements at the beginning of the project and the change in requirement in later stage of project development, we took long time to finalize Amazon QLDB as our database. Our project schedule was pushed back due to these reasons, which gave us less amount of time to develop the actual database.

As Amazon QLDB is relatively new technology and finding bare minimum proper documentation and guidance was another challenge that we faced. Accessing AWS console given by our client to start our project was obstructed by levels of access and authorization error. After we got started by creating our own AWS account, further error even while following guidance provided by Amazon lead us to find the solution elsewhere on internet. This not only affected our project schedule but also gave us loads of pressure. Creating API on Amazon QLDB to connect with Amazon S3 was another daunting task for us, for which we implemented alternate way to achieve the same functionality. Verifying the data integrity using code was the main challenge that we faced and were able to overcome it.

8.4 Future Improvements

During the project development and even after we finished our project new idea came to us which we thought can be done as future improvements. The best idea we have is to link Amazon S3 and Amazon QLDB directly so that when any file is stored in Amazon S3, then that file unique Etag will be generated and stored in QLDB along with other parameters i.e. document features. For the feature mentioned above, help from Amazon may be needed, but still it will be a good feature if implemented. Another future improvement that we can visualize and is possible with little bit of extra time is to work with multiple documents. As we have our limitations, so do the future improvement on it. Creating a proper API and making it possible to scan multiple documents and show their Amazon S3 path and validation of Etag will make this project beneficial to any business that maintains transaction ledger.

9. Conclusion

Owning a property is always a crucial aspect in anyone life. Property documents come attached with the property which the owner must manage themselves. As those documents in paper format is too much of a hassle to handle, in this digitalized world, property handover is done through either email, USB or any other technical format. After receiving the property documents then comes the crucial aspect of security and transparency of those documents. As the world has become digitalized so has the need of data security. On top of that due to lots of cases of property fraud being increased there has been even more need of data transparency between the related parties to maintain trust between them.

We as a team in our project have researched, designed, and developed a way to maintain security, transparency of property documents and help increase the trust between the related parties. Document related to the property owner will be stored in Amazon S3 database for proper security, while the transaction log of those documents between various parties is stored in Amazon QLDB. Amazon QLDB has cryptographic hash function which will make all those transactional data immutable. These data can never be deleted and maintained throughout the ledger lifecycle. We have also made it possible to find the ledger data using parameters like Property owner account id and property id. User can find their document path in QLDB by providing their ID and easily access those documents from Amazon S3. Amazon QLDB has made it possible to verify data stored in QLDB through the means of console. But as this task takes too much time and labor, we wrote a program to automatically verify the data integrity by comparing the unique Etag created while storing the document in S3 and the Etag stored in Amazon QLDB. Using this program user can easily access and find their Etag. If Etag is not found then it will let user know Etag is not found, which means the file has been deleted from S3. If the Etag is found, then it can further verify the integrity of that Etag. If the file in S3 has been altered without user permission then the verification process will fail, letting user know their document has been compromised or tampered with and they can take needed action to rectify it. Ledger created on QLDB can also be exported to Amazon S3 for further security as the ledger content can be accessed from both Amazon S3 and Amazon QLDB. This feature also makes it easier for user to access the data they needed easily. The Etag generated while storing the file on Amazon S3 can be shared by INNDOX with all the related parties, be they Property owner or builder. After that each of these related parties can verify these documents integrity when needed which will maintain the data transparency and trust between them.

References

- Allen, L., Antonopoulos, P., Arasu, A., Gehrke, J., Hammer, J., Hunter, J., ... & Setty, S. (2019).
- Amazon Managed Blockchain. Retrieved from <https://aws.amazon.com/managed-blockchain/>
- Amazon Quantum Ledger Database. Retrieved from <https://aws.amazon.com/qldb/>
- Amazon Web Services. Retrieved from <https://aws.amazon.com/>
- Amazon Web Services, Inc. (2020). *Start Building on AWS Today*. Retrieved from <https://aws.amazon.com/qldb/>
- Curcio, K., Navarro, T., Malucelli, A., & Reinehr, S. (2018). Requirements engineering: A systematic mapping study in agile software development. *The Journal of Systems and Software*, 139, 32–50. <https://doi.org/10.1016/j.jss.2018.01.036>
- Del Sole, A. (2018). *Visual Studio Code Distilled: Evolved Code Editing for Windows, MacOS, and Linux*. Apress.
- Flood, G. (2005). C# programming language. *IT Training*, 66–67.
- Jacob, B., Porter Jr, J., Khemraj, R., Lasecki, D., & Miller Jr, E (2019). What can we learn from early adopters?
- Raikwar, M., Gligoroski, D., & Velinov, G. (2020). Trends in the Development of Databases and Blockchain. *arXiv preprint arXiv:2003.05687*.
- Scrum Alliance, Inc. (2020). *2017-18 State of Scrum Report*. Retrieved from <https://www.scrumalliance.org/learn-about-scrum/state-of-scrum>
- Srivastava, A., Bhardwaj, S., & Saraswat, S. (2017). SCRUM model for agile methodology. 2017 International Conference on Computing, Communication and Automation (ICCCA), 2017, 864–869. <https://doi.org/10.1109/CCAA.2017.8229928>
- Veritas: Shared verifiable databases and tables in the cloud. In 9th Biennial Conference on Innovative Data Systems Research (CIDR).
- Zhang, M., Xie, Z., Yue, C., & Zhong, Z. (2020). Spitz: a verifiable database system. *arXiv preprint arXiv:2008.09268*.