



***UNIVERSITY OF MASSACHUSETTS
DARTMOUTH***

DEPARTMENT OF COMPUTER & INFORMATION SCIENCE

A Prototype Blockchain for Storing Trusted Predictive Models

CIS 600: MASTER'S PROJECT Final Project Report

PROJECT ADVISOR

DR. Haiping Xu.

PRESENTED BY

Aravind Pagadala

ABSTRACT

Machine Learning (ML) is essential for automating tasks, predicting outcomes, and aiding human decision making, significantly enhancing human understanding. However, as ML evolves, ensuring model reliability and transparency becomes increasingly critical. Users often rely on these models without a clear method to verify their accuracy or uncover hidden biases. To address this challenge, this project introduces an innovative solution by integrating blockchain technology with ML model management. Unlike traditional methods that depend on centralized databases or third-party validators, our approach uses the decentralized nature of blockchain to create a more trustworthy and transparent system for managing ML models. Each model is recorded as a series of transactions on the blockchain, preserving its history and integrity. This setup allows key components of ML models, such as neural network weights and configuration details, to be stored securely. Consequently, users across the network can independently access, review, and validate these models, thereby increasing trust. The main benefit of using blockchain is that it promotes customer trust through secure and transparent model management. The decentralized nature of blockchain also ensures redundancy, where multiple copies of models are stored in different locations to prevent data loss, thus allowing easy access to models with built-in redundancy. To demonstrate the feasibility of this approach, we developed a prototype blockchain for storing neural network predictive models. For efficiency, we further implemented a meta-block containing organized model information, such as the indexing details of the classifiers and their review comments. We validated this approach with a case study of storing real estate predictive models on blockchain, demonstrating that this system successfully provides a secure, transparent, and immutable way to manage ML models.

Table of Contents

1. Introduction	5
1.1 Introduction to the Project.....	5
1.2 Introduction to Blockchain	6
2. Project Overview	7
2.1 Concept and Scope	7
2.2 Project Architecture.....	8
2.3 Architecture Diagram	11
2.4 Class Diagram	13
2.5 Implementation Plan.....	14
3. Tools and Software Used	16
3.1 Java Programming.....	16
3.2 Java RMI.....	16
3.3 Spring Boot	16
3.4 Maven.....	16
3.5 Eclipse IDE.....	16
4. Project Installation.....	17
5. Application Walkthrough.....	22
5.1 Implementation Details.....	22
5.2 Running the servers	23
5.3 Running Spring boot Applications	24
6. Test Cases.....	26
6.1 Test Case 1: Block Creation	26

6.2 Test Case 2: Review Comments	28
6.3 Test Case 3: Extracting ML Classifier	29
6.4 Test Case 4: Extracting Reviews	30
6.5 Test Case 5: Search	31
6.6 Test Case 6: Transactions	32
7. Conclusion	33
8. References	34

1. Introduction

1.1 Introduction to the Project

Machine learning (ML) models have increasingly become the backbone of modern technology, influencing decision-making processes across various industries. However, the adoption of these models often requires users to place significant trust in the organizations that deploy them, with limited means to independently verify model accuracy or identify potential biases. This lack of transparency can lead to skepticism and hinder the broader acceptance of ML solutions.

In this project, we are building a blockchain from scratch using Java, complemented by full stack development with Spring Boot. Unlike existing approaches that rely on established blockchain platforms such as Ethereum and utilize smart contracts, our custom-built blockchain allows for a more tailored and flexible implementation.

Our approach involves storing the parameters of neural network-based predictive models directly on the blockchain. By designing efficient algorithms, we can securely upload and retrieve these parameters, ensuring that all nodes in the network maintain an identical and tamper-proof copy of the model data. This decentralized storage mechanism allows users to independently test, validate, and review the ML models, thereby promoting greater accountability and trust. To further enhance the efficiency of our system, we introduce a meta-block at the beginning of the blockchain. This meta-block contains indexing information about the stored predictive models and includes review comments, streamlining the retrieval process for users. By organizing and structuring this information, we aim to improve the accessibility and usability of blockchain-stored models.

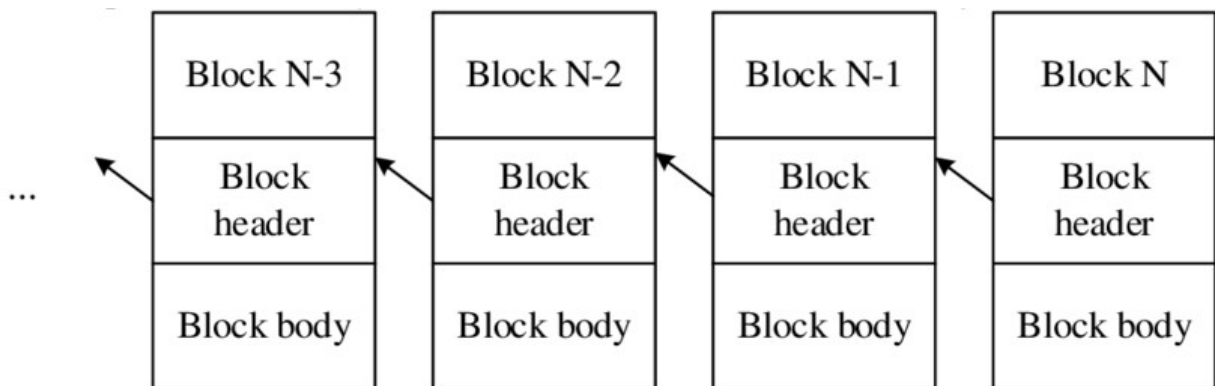
Through comprehensive case studies and simulations, we demonstrate the feasibility and effectiveness of our proposed solution. Our results show that integrating a custom-built blockchain with ML model storage not only enhances security and immutability but also fosters transparency and trustworthiness, addressing the key concerns associated with the deployment of ML models in various applications.

1.2 Introduction to Blockchain

Imagine a digital ledger that everyone can see but no one can alter. This is the essence of blockchain technology, a groundbreaking innovation that's reshaping how we think about trust and transparency in the digital age.

At its core, a blockchain is like a high-tech notebook. Each page of this notebook is a "block" that contains a list of transactions. Once a page is full, it gets added to a chain of previous pages, creating a continuous, unchangeable record of all transactions. What makes this system revolutionary is that it's not controlled by any single person or organization. Instead, it's maintained by a network of computers that work together to ensure the accuracy and security of the information.

This decentralized approach means that once a transaction is recorded on the blockchain, it becomes nearly impossible to alter. It's a bit like having a public diary where every entry is verified by multiple witnesses before it's written down. This feature is what makes blockchain particularly valuable for applications where trust and security are paramount, such as financial transactions, supply chain management, and even voting systems.



In simple terms, blockchain offers a new way to record and verify information, making processes more transparent and secure. It's a technology that promises to transform many aspects of our digital lives, from how we handle money to how we verify the authenticity of goods. As we delve deeper into this project, we'll explore how blockchain works and the myriad ways it's being used to solve real-world problems.

2. Project Overview:

In this project, we explore the innovative use of blockchain technology to enhance trust and transparency in Machine Learning (ML) predictive models for house price prediction. By securely storing ML models on a blockchain, we aim to provide customers with reliable, tamper-proof models they can trust. This prototype allows customers to review, download, and run the ML classifiers on their local machines, fostering greater confidence in the predictions.

This blockchain system is designed with a few or more super peers, who are responsible for uploading the ML classifiers onto the blockchain. Once a new classifier is uploaded, all other super peers are notified and given the opportunity to review and approve the classifier. Upon approval, the classifier is added to the blockchain, making it available to all peers in the network.

The goal of this project is to create a secure and transparent ecosystem where customers can find reliable housing price prediction models. By analyzing reviews and comments from other users, customers can make informed decisions about which models to trust. Additionally, customers are encouraged to provide their own feedback by reviewing and commenting on the models they use. This collaborative approach ensures that the best models are recognized and widely adopted.

Ultimately, our aim is to provide redundant and reliable ML models that customers can easily access and verify. By enabling customers to download, extract, and run the models on their local computers, we empower them to independently verify the predictions and trust in the technology.

2.1 Concept and Scope

The primary objective of this project is to develop a prototype that redefines how we handle and trust Machine Learning (ML) classifiers for house price prediction. Imagine an environment where experienced and trusted members, known as super peers, contribute their carefully crafted ML models to a shared blockchain.

When a super peer uploads a new ML classifier, it's not just added to the system without scrutiny. Instead, all other super peers are immediately notified. These peers then take the time to review the new classifier, checking its accuracy and reliability. This collaborative review process is crucial because it ensures that only the best and most trustworthy models are approved. Once a classifier passes this rigorous peer review, it is securely added to the blockchain, making it available to everyone in the network.

This hands-on approach not only builds trust but also encourages customers to provide feedback. They can share their experiences, leave reviews, and discuss the models with others, creating a

dynamic and interactive environment. This project aims to create a secure, transparent, and community-driven ecosystem for ML house price prediction models. By ensuring that only validated and trustworthy models are shared, we enhance customer confidence and contribute to more accurate and reliable predictions in the real estate market. This collaborative effort brings together technology and human interaction, fostering a space where everyone can participate and benefit from shared knowledge and trust.

2.2 Project Architecture

As mentioned in the project overview, the application is driven by four main components: super peers, regular peers, Machine Learning (ML) code, and blockchain. Each of these entities plays a crucial role in ensuring the system's functionality and reliability.

1. Super Peers

- Super peers are the backbone of the system, responsible for uploading ML classifiers into the blockchain.
- They act as the gatekeepers, ensuring that only high-quality models enter the system. Super peers also review and approve classifiers uploaded by others, adding an extra layer of scrutiny and trust.
- Super Peers can be able to Create a new Block in the Blockchain, retrieve them and add the reviews and comments to the blocks.

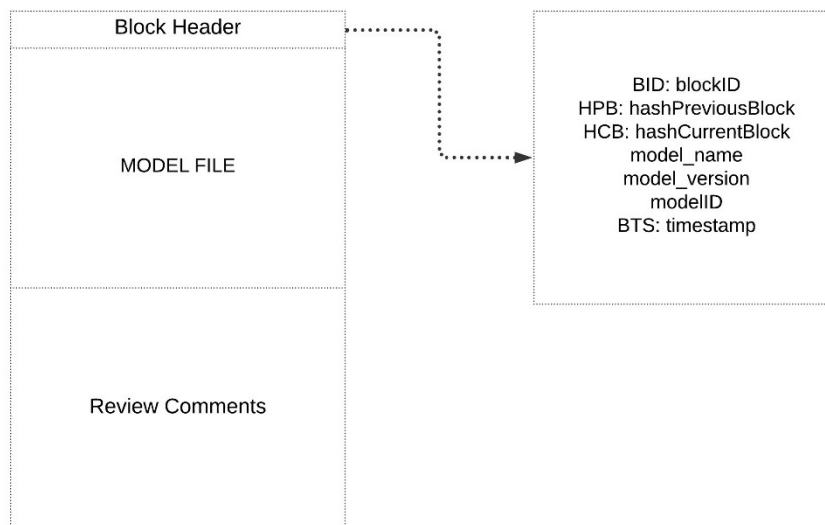


2. Regular Peers

- Regular peers are the end users who benefit from the models validated and shared by the super peers.
- They can download, run, and review the ML models on their local machines. Regular peers can also provide feedback, rate models, and participate in discussions, contributing to the community's collective knowledge and trust.
- Regular peers can connect to any super peers on the network and access the resources by requesting the super peer.

3. Block:

Block is a representation of piece of confined resources that in this case stores the ML file and maintains the information about the file, it also stores the review comments, the detailed diagram is as follows:



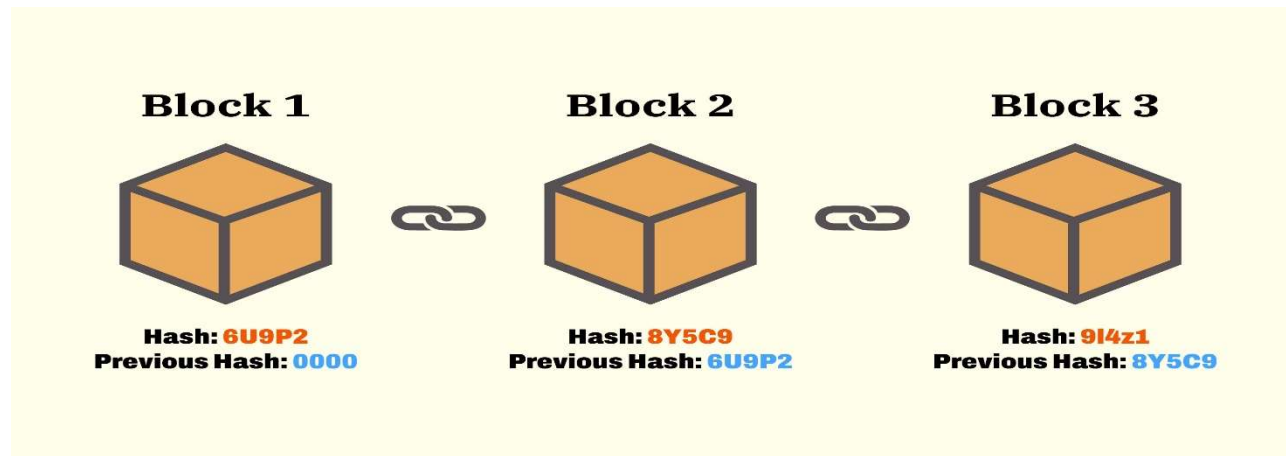
4. Machine Learning Code

The ML code is the core of our predictive models. These models are trained to provide accurate house price predictions. By storing these models on the blockchain, we ensure their integrity and allow users to verify their validity independently. The ML code undergoes rigorous testing and validation by super peers before it is shared with the wider community.

5. Blockchain

Blockchain technology provides the secure, immutable ledger that underpins the entire system. It records every transaction, from the upload and review of ML models to the feedback provided by regular peers. This transparency ensures that all actions are traceable and verifiable, building trust

among all participants.



6. Supporting Components

In addition to these main components, several supporting elements help maintain the system's reliability and redundancy:

- **Transactions:** Every action, from uploading a new model to providing feedback, is recorded as a transaction on the blockchain. This ensures a transparent and auditable history of all activities.
- **Meta Block:** These blocks contain metadata about the ML models, including their version, validation status, and review history. Meta blocks support the organization and retrieval of information, making the system more efficient and user-friendly.

7. Synchronization

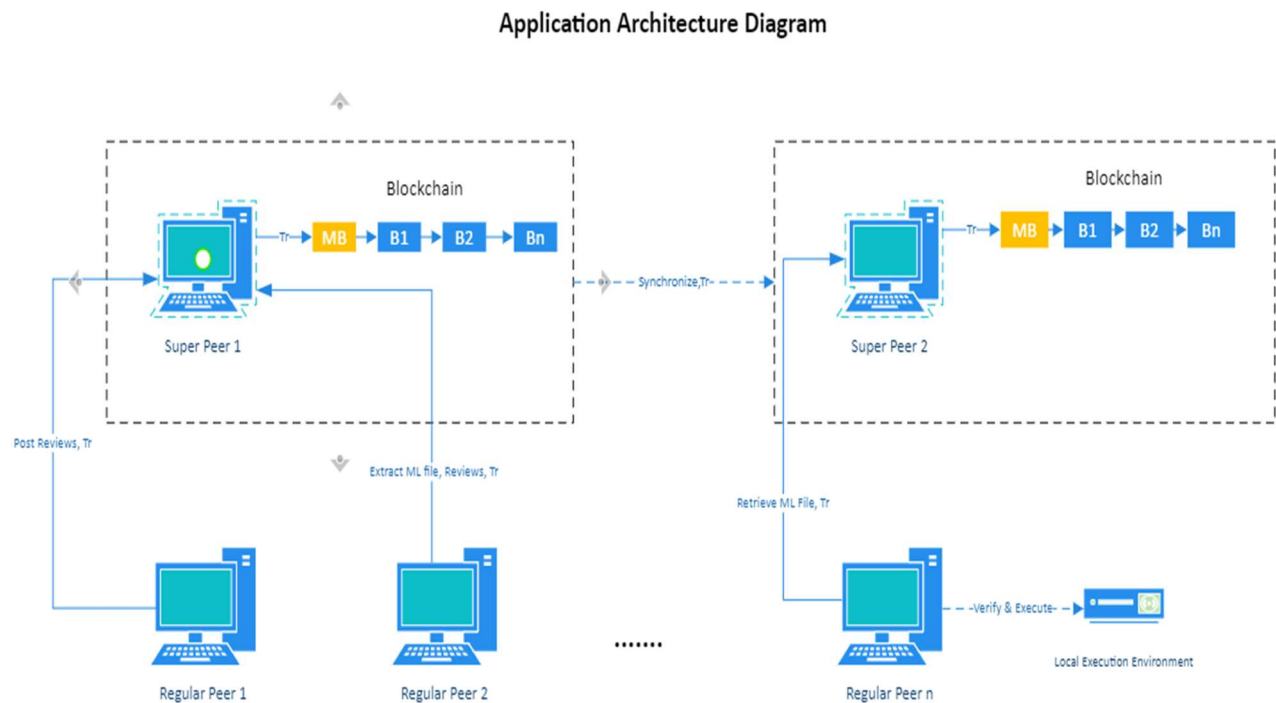
Synchronization is a critical aspect of this project, ensuring that data is consistent, reliable, and redundant across the entire network. By maintaining a copy of the blockchain in every super peer, we guarantee that the system remains robust and trustworthy, even in the face of potential failures or attacks.

Redundant Data Storage

Each super peer maintains a complete copy of the blockchain. This redundancy ensures that the loss or compromise of any single peer does not affect the integrity or availability of the data. With every super peer holding a full copy, the network can quickly recover and continue functioning smoothly, maintaining the trust of all participants.

By combining advanced ML techniques with the transparency and security of blockchain, we provide a reliable and collaborative platform for all users.

2.3 Architecture Diagram:



The diagram provides a visual representation of the key components and interactions within our blockchain-based system for ML house price prediction models.

Super Peers:

Super Peer 1 and Super Peer 2:

- Each super peer maintains a complete copy of the blockchain (B1, B2, B3, ... Bn).
- Super peers are responsible for uploading new ML classifiers to the blockchain.
- They also review and validate classifiers uploaded by other super peers.
- If Super peer 1 wants to create a block in blockchain, it will first create a temporary block in a temporary folder. Then the consensus algorithm runs, and the peers approve the new block.
- Only after then, the block is added to the blockchain of each peer.

Blockchain:**Blockchain Copy:**

- Both Super Peer 1 and Super Peer 2 maintain a synchronized copy of the blockchain.
- Each block (B1, B2, B3, ... Bn) contains an ML classifier, a header, Review comments.

Regular Peers:

Regular Peer 1, Regular Peer 2, and Regular Peer n:

- Regular peers can request and retrieve ML classifiers from super peers.
- They can extract the ML file and associated reviews.
- Regular peers verify and execute the ML classifier in their local execution environment.
- They also post reviews and feedback on the classifiers.

Interactions:

Upload and Synchronization:

When Super Peer 1 uploads a new classifier, it is synchronized with Super Peer 2.

- Every operation has Tr noted next to it, Where Tr is a transaction.

Review Process:

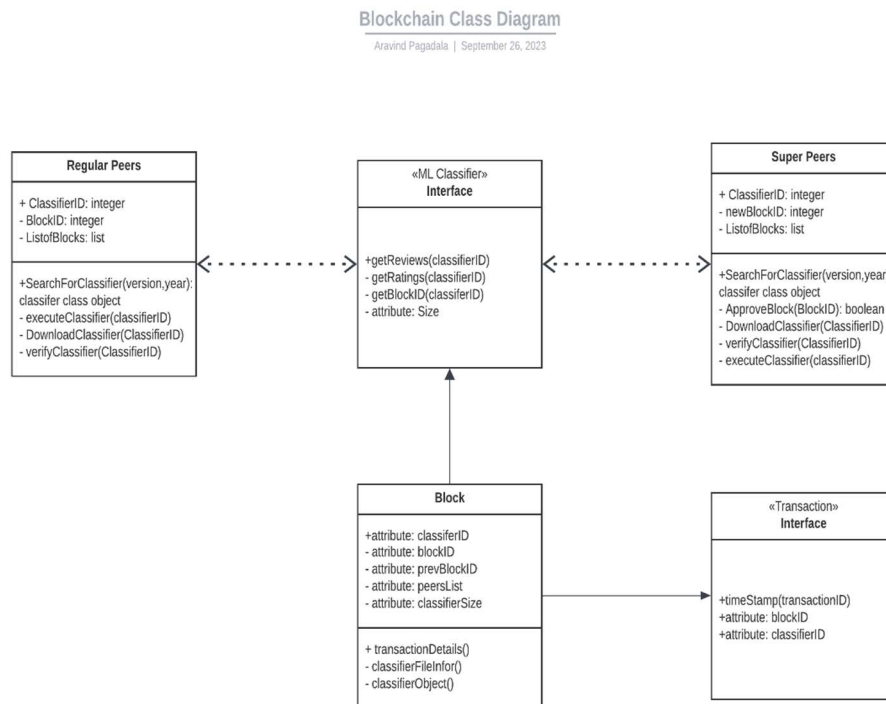
Super peers review and approve or reject the new classifier before it is added to the blockchain.

Data Access and Execution:

Regular peers retrieve ML classifiers from super peers, verify them, and run them locally.

They provide feedback and reviews which are shared across the network.

2.4 Class Diagram



ML Classifier Blockchain

This blockchain template contains the representation of class diagram of the ML classifier blockchain. The class diagram shows there are three classes named block, super peers and regular peers. It also contains the two interfaces which are ML classifier and transaction. Super peers can be able to verify, upload, execute, search, download the classifier.

Regular peers be able to only search and execute the classifier.

All super peers have to be synchronized

Each Block has its previous and its own BlockID hash

ML classifier is sytored as a piece of code in a block

Diagram Insights:

- **Synchronization:**
 - All super peers must be synchronized to ensure consistency across the network.
- **Block Structure:**
 - Each block has a blockID and contains references to both its previous block (prevBlockID) and a list of peers (peerList).
 - Blocks store classifier information as encrypted code.
- **Class Interactions:**
 - Regular peers can search, download, verify, and execute classifiers.
 - Super peers can approve new blocks, synchronize updates, and ensure the integrity of the blockchain.
- **Attributes and Methods:**
 - The attributes and methods in each class facilitate the various operations needed for maintaining and utilizing the blockchain for ML classifiers.

2.5 Implementation Plan

This project implementation is divided into three phases to ensure a structured and comprehensive development process. Each phase focuses on a specific aspect of the system, ultimately leading to a fully functional and reliable platform for ML house price prediction models.

Phase 1: Implementing Super Peers

Super Peers and Blockchain:

- Blockchain Development:

- Develop a simple blockchain architecture where each block contains an ML classifier.
- Implement synchronization mechanisms to ensure all super peers have an up-to-date copy of the blockchain.

- User Interface:

- Create a front-end interface for super peers to upload and review classifiers.
- Implement login functionality to secure super peer access.

- System Setup:

- Use two systems to simulate super peers and their interactions.
- Ensure that these systems can communicate and synchronize the blockchain effectively.

Synchronization and Validation:

- When a super peer uploads a new classifier, the system will:
 1. Notify all other super peers about the new block.
 2. Allow super peers to review the classifier and approve or reject it.
 3. Upon approval, synchronize the classifier across all super peers' copies of the blockchain.

Benefits of Using Blockchain:

- Transparency: All peers can track the history and origin of each classifier.
- Security: Immutable records prevent tampering with the classifiers.
- Trust: Peer review and validation processes ensure the reliability of the classifiers.

Additional Details:

- ML Model: House price prediction.
- Storage: Classifiers are stored in blocks as encrypted code.
- Synchronization: Super peers add new blocks containing classifiers, which are then synchronized across all super peers.
- Data Access: Super peers maintain a complete copy of the blockchain, while regular peers request specific classifiers from super peers.

Phase 2: Building Blockchain, Transactions, and Meta Block

Blockchain:

- Implement a robust blockchain structure to handle the secure storage and synchronization of ML classifiers.
- Develop transaction mechanisms to record every upload, review, and synchronization action.

Transactions:

- In a blockchain system, transactions are the fundamental units of operation. They represent the transfer of data or assets from one party to another.
- Once a transaction is included in a block and added to the blockchain, it cannot be altered or deleted. This ensures the integrity of the data.
- Transaction contains Requestor, Provider, DES for description of the transaction, modelID.

Meta Block:

- Create meta blocks that contain metadata about each ML classifier, including version, validation status, and review history.
- Ensure meta blocks support efficient organization and retrieval of information.

Phase 3: Building Regular Peers

Regular Peers:

- Implement the functionality for regular peers who will download, run, and review ML classifiers.
- Develop an interface for regular peers to request classifiers from super peers.
- Ensure regular peers can provide feedback, rate models, and participate in discussions.

System Integration:

- Integrate regular peers into the blockchain network, ensuring seamless interaction with super peers.
- Implement synchronization mechanisms to keep regular peers updated with the latest classifiers and reviews.

3. Tools and Software Used/Learned

3.1 Java JDK 17

Java Development Kit (JDK) 17 is a robust and widely-used development environment for building Java applications. As a Long-Term Support (LTS) release, JDK 17 provides enhanced performance, security, and stability, making it an ideal choice for developing modern, scalable applications. It includes the Java Runtime Environment (JRE), compiler, and various libraries and tools necessary for Java development. The updated features and improvements in JDK 17 ensure efficient coding practices and support for the latest Java language enhancements.

3.2 Java RMI

Java Remote Method Invocation (RMI) is a Java API that enables the creation of distributed applications. With RMI, you can invoke methods on objects located on remote servers as if they were local. This technology simplifies the development of networked applications by handling the complexities of remote communication. It's particularly useful in scenarios where you need to build systems that require communication between different parts of a distributed network.

3.3 Spring Boot

Spring Boot is an extension of the Spring framework that simplifies the development of production-ready applications. It provides a suite of tools and conventions for creating stand-alone, production-grade Spring-based applications with minimal configuration. Spring Boot is designed to make it easy to set up and run new applications, with features like embedded servers, automatic configuration, and a wide range of starter projects that streamline the development process. Its ease of use and extensive ecosystem make it a popular choice for building modern web services and applications.

3.4 Maven

Apache Maven is a build automation tool used primarily for Java projects. It simplifies the process of managing project dependencies, building applications, and generating documentation. Maven uses a Project Object Model (POM) file to define project structure, dependencies, and build configurations. By providing a standardized build lifecycle and a central repository for libraries, Maven helps ensure consistency and efficiency in the development process.

3.5 Eclipse IDE Latest

Eclipse Integrated Development Environment (IDE) is a powerful and versatile tool for Java development, among other languages. The latest version of Eclipse IDE offers an extensive set of features, including code editors, debugging tools, and integration with various build systems and version control systems. It provides a user-friendly environment for writing, testing, and managing code, making it a popular choice among developers for its flexibility and support for a wide range of plugins and extensions.

4. Project Installation

Here is a step-by-step installation guide for setting up Java Maven project in Eclipse IDE on Windows: This project uses Java 17 version of jdk.

Step 1: Install Java Development Kit (JDK)

1. Download JDK:

- Go to the Oracle JDK download page.
- Download the latest JDK version suitable for Windows.

2. Install JDK:

- Run the downloaded installer.
- Follow the on-screen instructions to complete the installation.
- Note the installation directory (e.g., C:\Program Files\Java\jdk-17).

3. Set Environment Variables:

- Open the Control Panel, go to System and Security -> System -> Advanced system settings.
- Click on Environment Variables.
- Under System Variables, click New and add JAVA_HOME with the path to the JDK installation directory (e.g., C:\Program Files\Java\jdk-17).
- Find the Path variable, select it, and click Edit. Add a new entry with %JAVA_HOME%\bin.

4. Verify Installation:

- Open Command Prompt and type java -version and javac -version to ensure Java is installed correctly.

Step 2: Install Eclipse IDE

1. Download Eclipse:

- Go to the Eclipse download page.
- Download the "Eclipse IDE for Java Developers" package.

2. Install Eclipse:

- Extract the downloaded zip file to a desired location (e.g., C:\Eclipse).
- Navigate to the Eclipse directory and run eclipse.exe.

Step 3: Install Maven

1. Download Maven:

- Go to the Maven download page.
- Download the latest binary zip archive.

2. Install Maven:

- Extract the zip archive to a desired location (e.g., C:\Program Files\Maven).

3. Set Environment Variables:

- Open the Control Panel, go to System and Security -> System -> Advanced system settings.
- Click on Environment Variables.
- Under System Variables, click New and add MAVEN_HOME with the path to the Maven installation directory (e.g., C:\Program Files\Maven).
- Find the Path variable, select it, and click Edit. Add a new entry with %MAVEN_HOME%\bin.

4. Verify Installation:

- Open Command Prompt and type mvn -version to ensure Maven is installed correctly.

Step 4: Create a Maven Project in Eclipse

1. Open Eclipse:

- Launch Eclipse by running eclipse.exe (or use start button).

2. Install Maven Integration for Eclipse:

- Go to Help -> Eclipse Marketplace.
- In the Eclipse Marketplace dialog, search for "Maven Integration for Eclipse" (also known as m2e).
- Click Go and install the plugin.

3. Create a New Maven Project:

- We might not need to create a Maven project, instead we open an already existing Spring Boot project.

- Go to File -> Import. In Import, select 'Existing Maven Project', in the next page under Root directory -> choose Browse and open the project which contains pom.xml file.

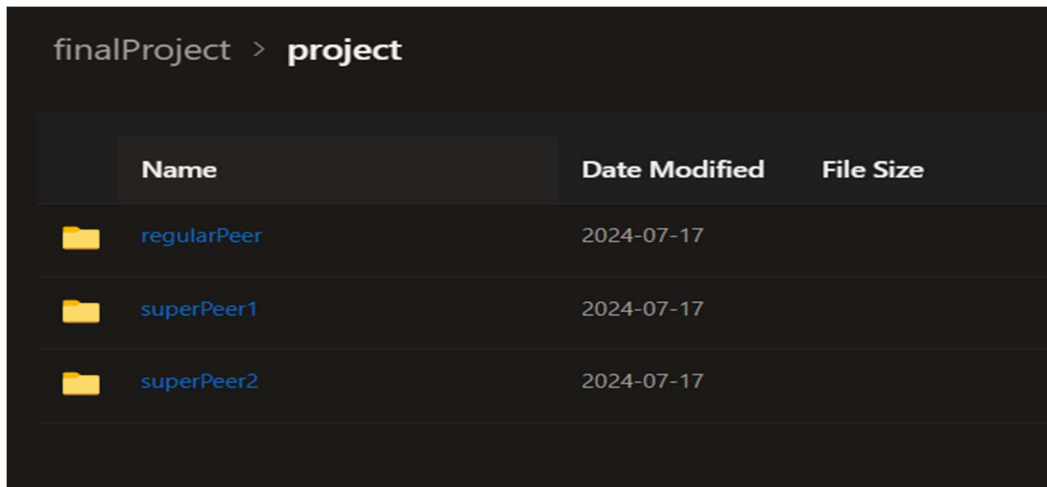
Steps to Install the project and Project Setup:




Please follow the below steps to set up the project.

I assume the above steps led you to install all the software tools to configure this project. As we have installed Java, Maven and Eclipse IDE, let's continue to setup.

Step1: Download the Source code

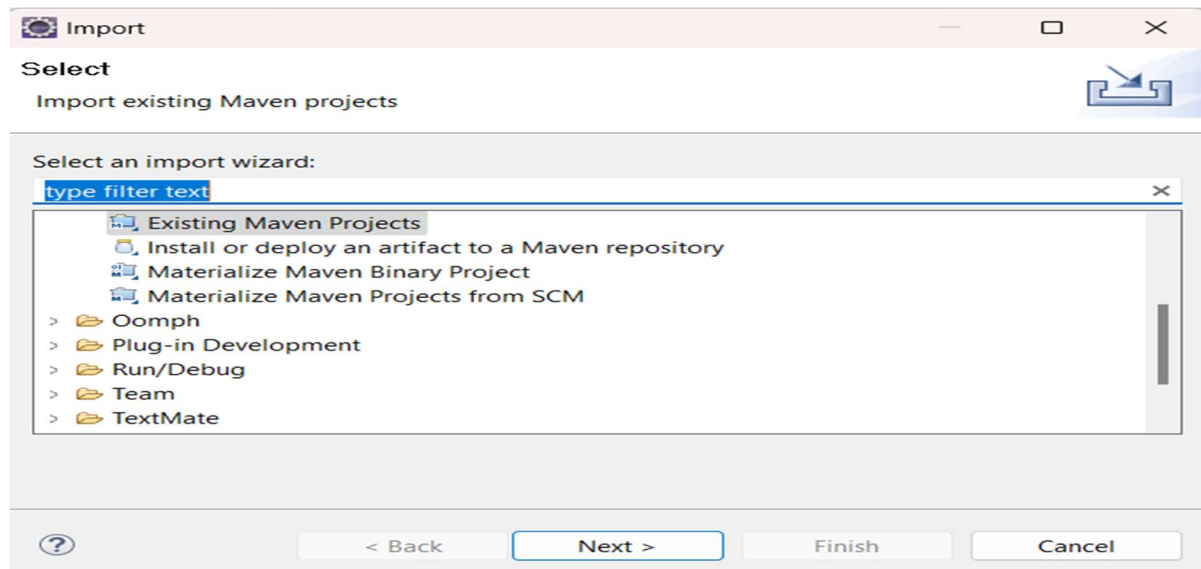
- Please download the source code for this project, I have sent the code in a .zip file.
- Extract the project files. (zip file should have three folders, extract them all).



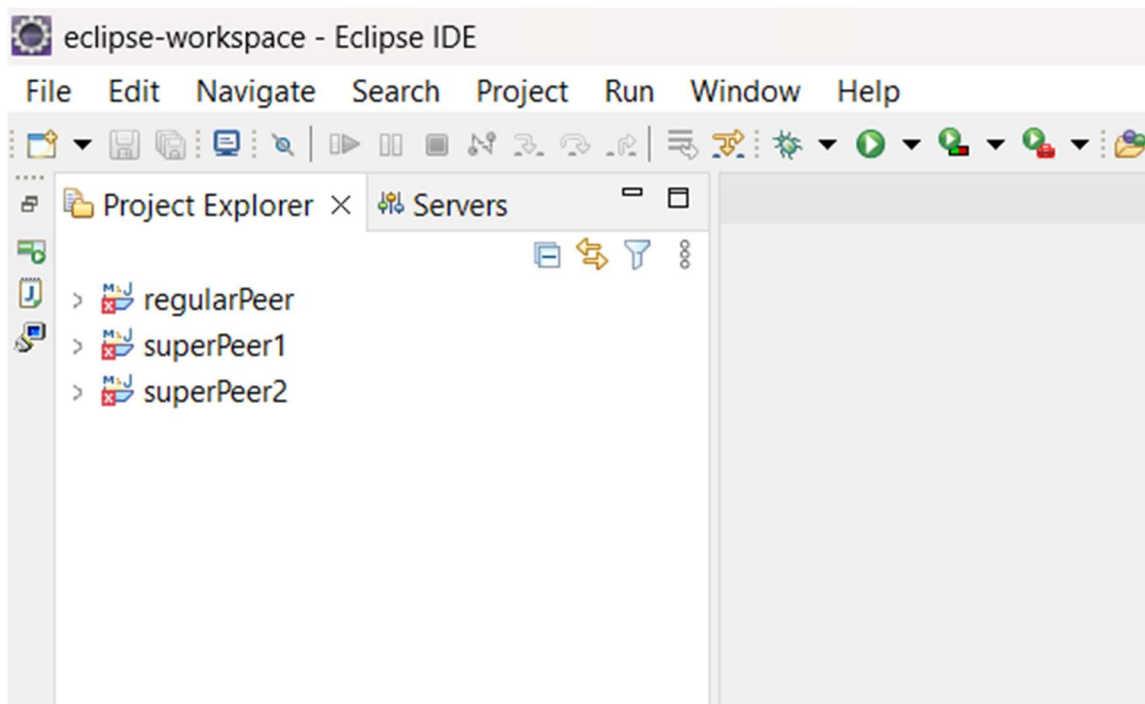
finalProject > project			
	Name	Date Modified	File Size
	regularPeer	2024-07-17	
	superPeer1	2024-07-17	
	superPeer2	2024-07-17	

Step 2: Importing Projects

- Open the Eclipse IDE, go to File → 'import'.
- Choose 'Existing Maven Project', click next.



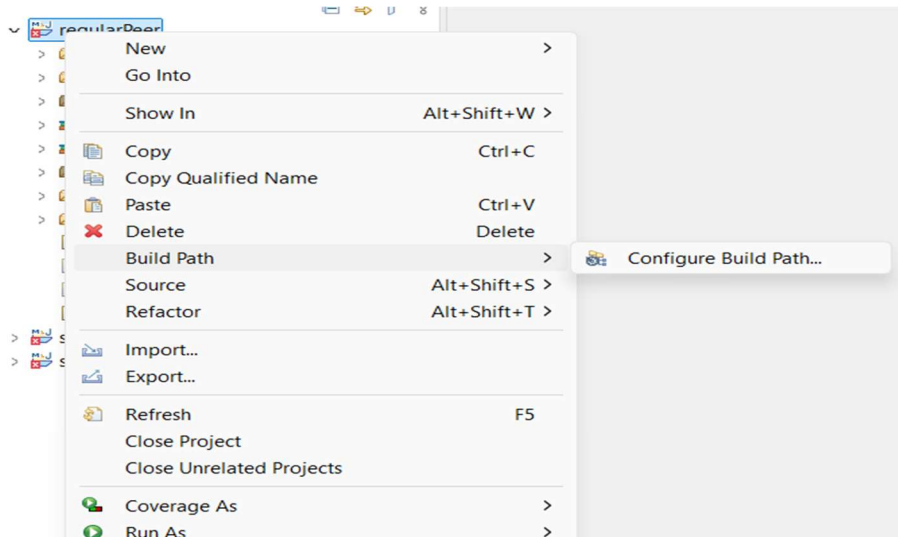
- Import all the three projects present in the extracted folder of source code, Click Finish.
- By now the Eclipse IDE should look like this. (Contains Maven Projects regularPeer, superPeer1, superPeer2)



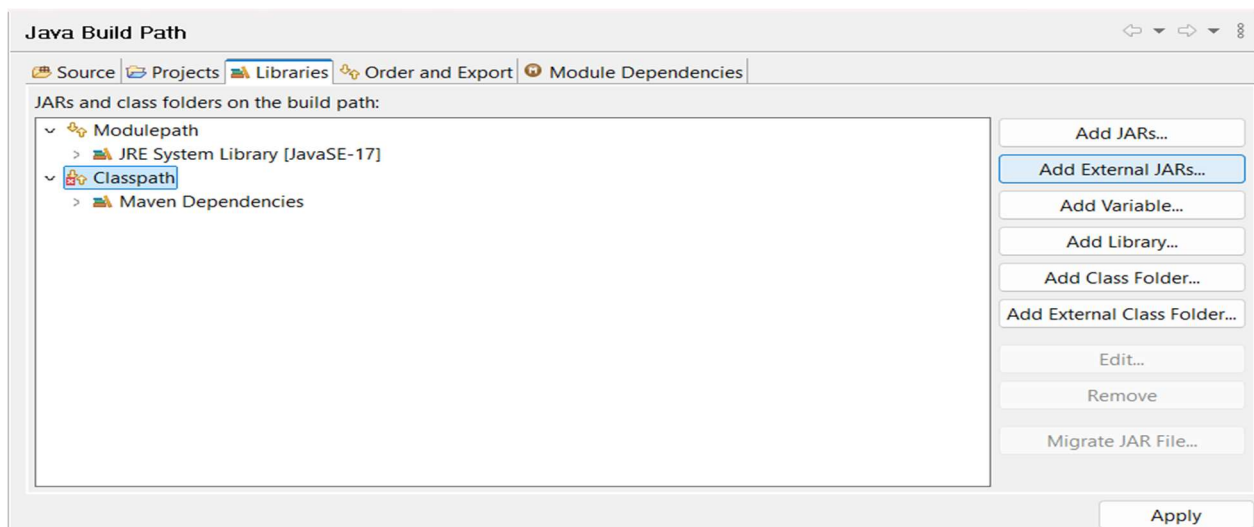
Step 3: Add dependencies

- Download the dependencies file that is sent in email, or you can also find the file in source code .zip. It contains commonInterface.jar file only.

- **Note: This jar file contains only the common Interface, no Implementation code included.**
- Extract the dependencies zip file to local PC.
- As of now you can see errors in the projects due to missing dependency file, we will include the jar file into each other projects.
- Right-click on regular peer project, go to build path → configure Build Path.



- Under libraries, click Classpath, Click add External JARs on the right menu.



- Do same for superPeer1 and superPeer2, include commonInterface.jar file in classpath.

5. Application Walkthrough

In this application, we have three Main components that are Super Peer, Regular Peer and a dependency Common Interface.

These three components interact with each other to reach the desired outcome. Both super peers and regular peers have a common interface, this interface will be injected as common dependency for all the 3 sub projects. This common interface is stored in the package: **rmi.common.Interface**

This common interface is named 'BlockchainInterface' which will be implemented by the super peers.

The blockchain interface extending Remote class is as follows:

```
package rmi.common.Interface;

import java.rmi.Remote;

public interface BlockchainInterface extends Remote {
    void initializeBlockchain(String[] blockFilePaths) throws RemoteException;

    byte[] extractMLFile(String modelID) throws RemoteException;
    byte[] extractReviews(String modelID) throws RemoteException;
    String[] search(String modelID) throws RemoteException;
    int bcSize() throws RemoteException;
    void createBlock(String modelName, String modelVersion,
        String modelFileBase64, String modelReviewsBase64) throws RemoteException;
    void addReviews(String modelID, String text) throws RemoteException;

    void receiveFile(byte[] file, String fileType, String modelID) throws RemoteException;
```

5.1 Implementation Details

The project contains 3 sub projects which are Regular Peer, SuperPeer1, SuperPeer2. Each sub project is running on 3 different ports on localhost, regular peer is on **localhost:8082**, super peer1 is on **localhost:8080**, whereas super peer 2 is on **localhost: 8081**. This is because all these sub projects are running on the same machine.

The project has two super peers, and their servers should run on two different RMI registers, so below are the RMI objects for the super peers.

Object Names for Super peers in RMI registers:

Super Peer 1: rmi:localhost:1099: superPeer1

Super Peer 2: rmi:localhost:1098: superPeer2

Login Credentials:

For Regular Peer:

Username: **admin**

Password: **password**

For Super Peer 1:

Username: **superPeer1**

Password: **password**

For Super Peer 2:

Username: **superPeer2**

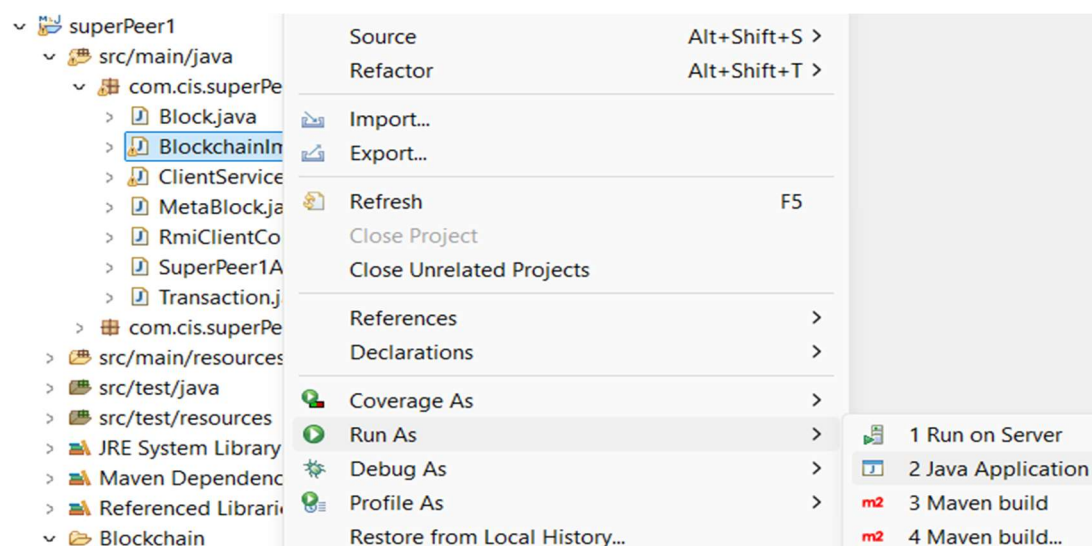
Password: **password**

5.2 Running the servers

Run the Server Applications:

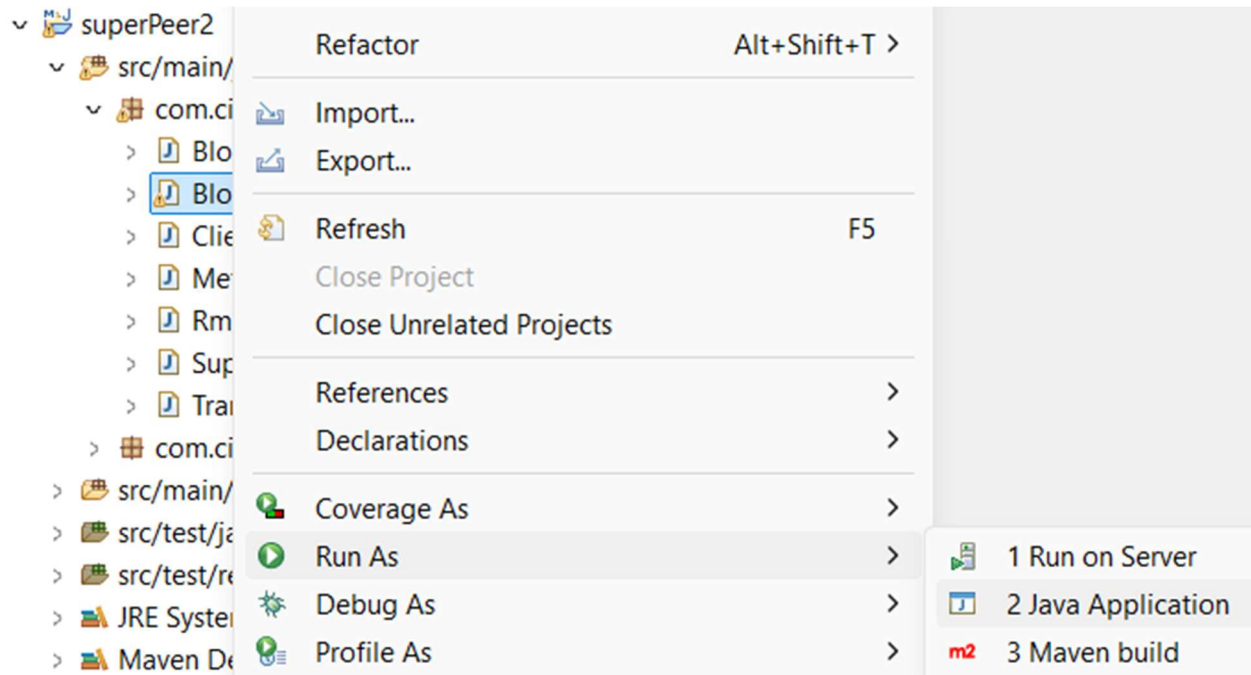
1. Run com.cis.superPeer1/BlockchainImpl:

- In the superPeer1 Project, open the com.cis.superPeer1 package
- Right-click on the class BlockchainImpl.
- Select Run As > Click Java Application.



2. Run com.cis.superPeer2/BlockchainImpl:

- In the superPeer2 Project, open the com.cis.superPeer2 package
- Right-click on the class BlockchainImpl.
- Select Run As > Click Java Application.

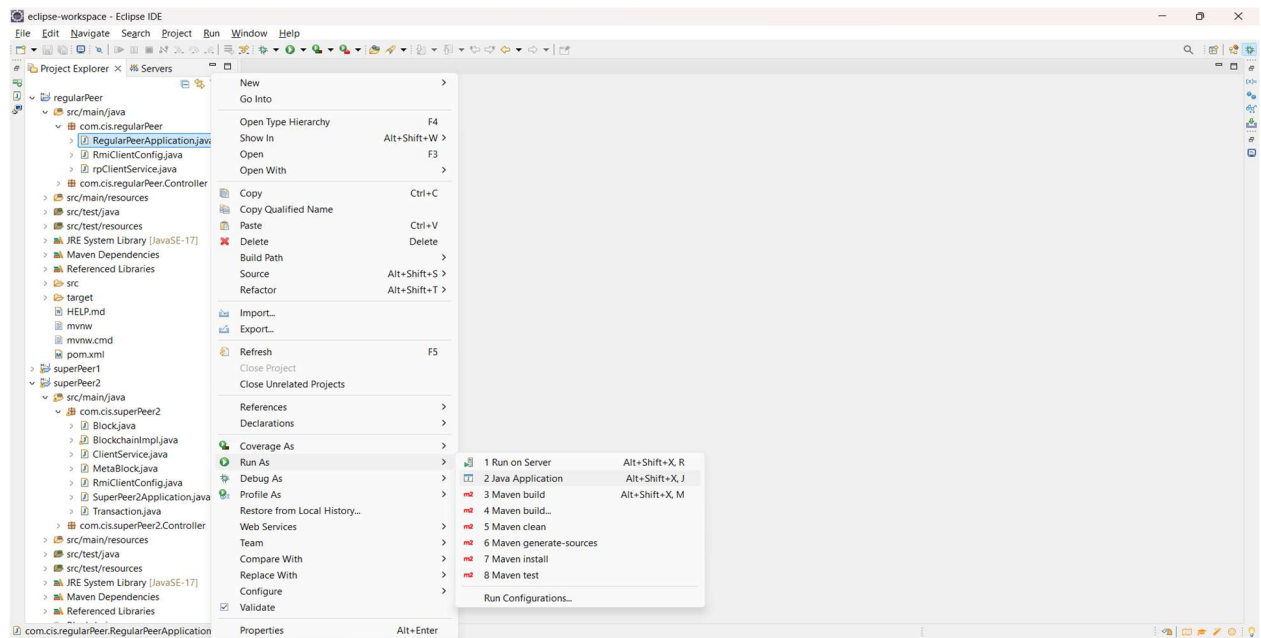


5.3 Running Spring boot Applications

Run Client Configurations:

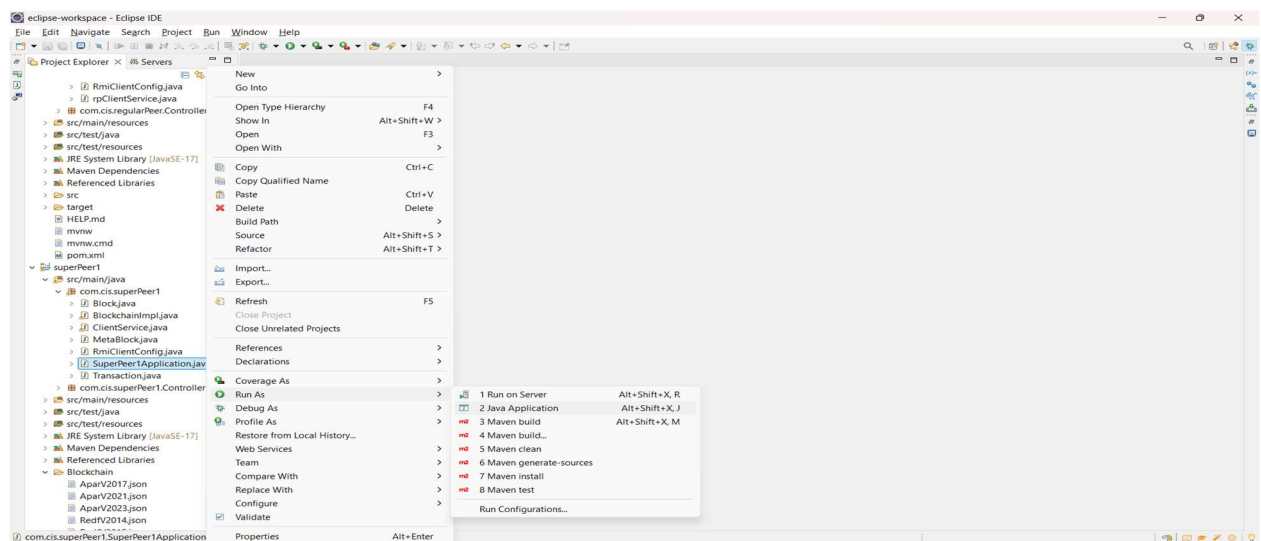
1. Run the Regular Peer Application:

- In the regularPeer Project, open the com.cis.regularPeer package
- Right-click on the RegularPeerApplication.java
- Select Run As > Click Java Application.



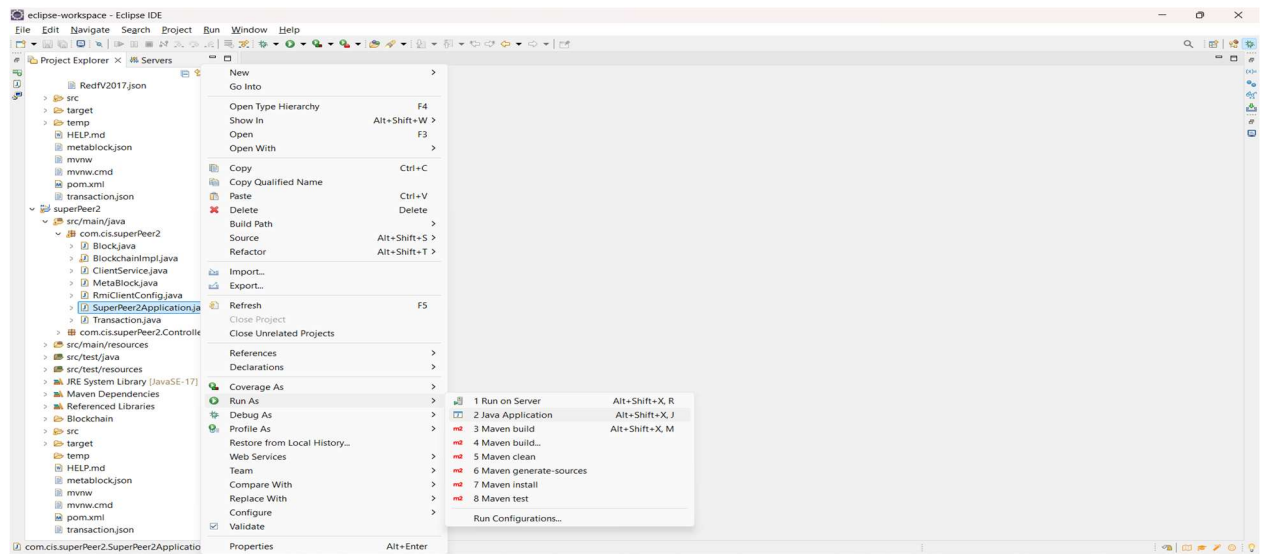
2. Run the Super Peer 1 Application:

- In the superPeer1 Project, open the com.cis.superPeer1 package
- Right-click on the SuperPeer1Application.java
- Select Run As > Click Java Application.



3. Run the Super Peer 2 Application:

- In the superPeer2 Project, open the com.cis.superPeer2 package
- Right-click on the SuperPeer2Application.java
- Select Run As > Click Java Application.



6. Test Cases

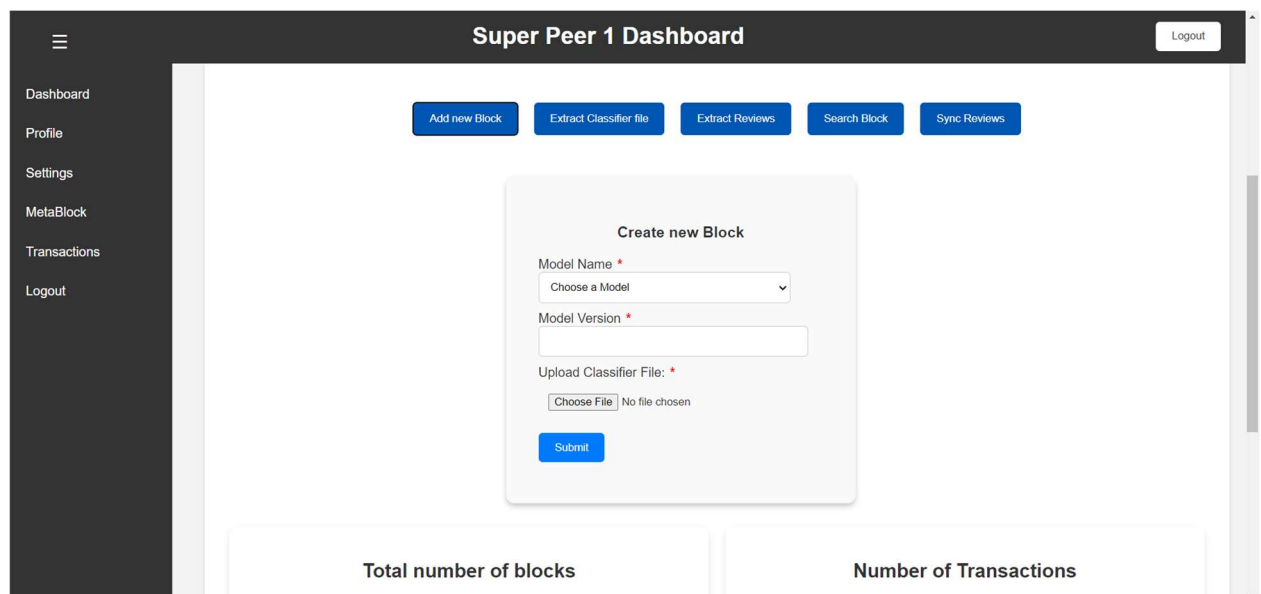
6.1 Test Case 1: Block Creation

Creating a new Block:

Let's go ahead and add a new Block,

Step 1: we first need to login to one of our super peers to do that.

Step 2: So, after login there should be a button on the dashboard saying, 'Add new block' and after clicking it the container is shown as below:



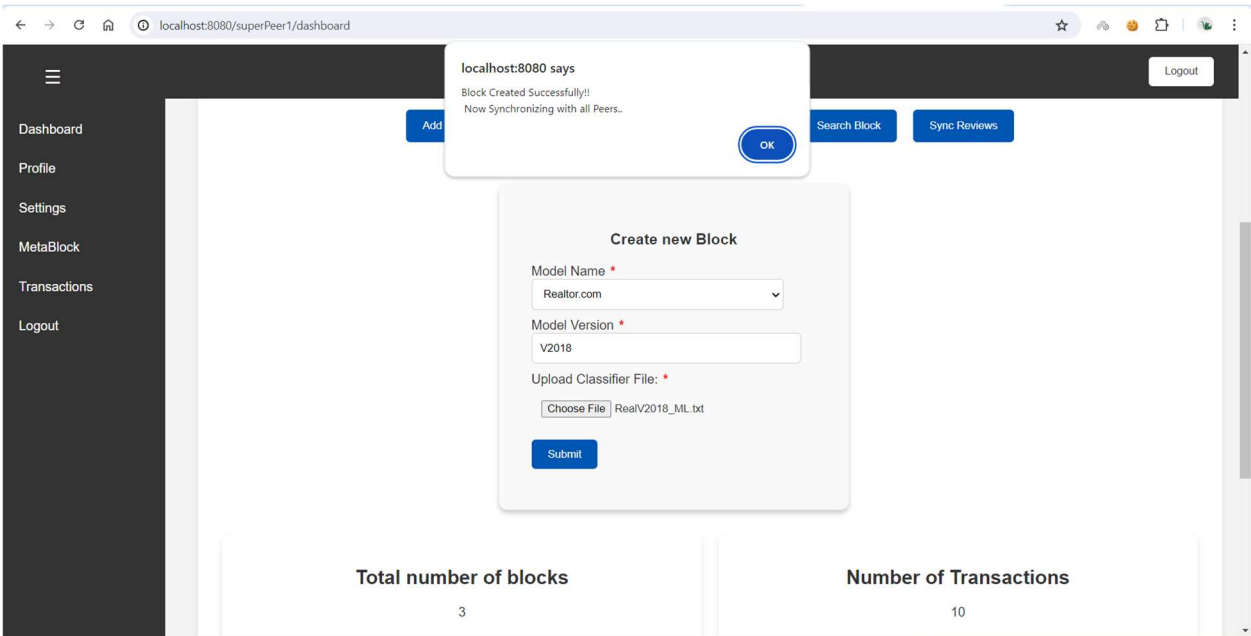
Step 3: Choose the model you want to create, now remember not to create an already existing model. You can always look for what's in the blockchain by navigating to 'MetaBlock' on the left panel.

Step 4: Enter a model Version

Step 5: Upload the ML classifier file from the files folder I've sent and click submit.

I suggest you add these blocks in the order as shown on the dashboard.

As you can see in the below screenshot, the block has been created and now it is synchronizing with all the other peers on the network. Click OK. Will return to dashboard.



You can verify this by going to transactions page first, it will be as follows:

Trans11	superPeer1	superPeer1	Temp Block Created	RealV2018
Trans12	superPeer1	All peers	New Block is added to Blockchain	RealV2018

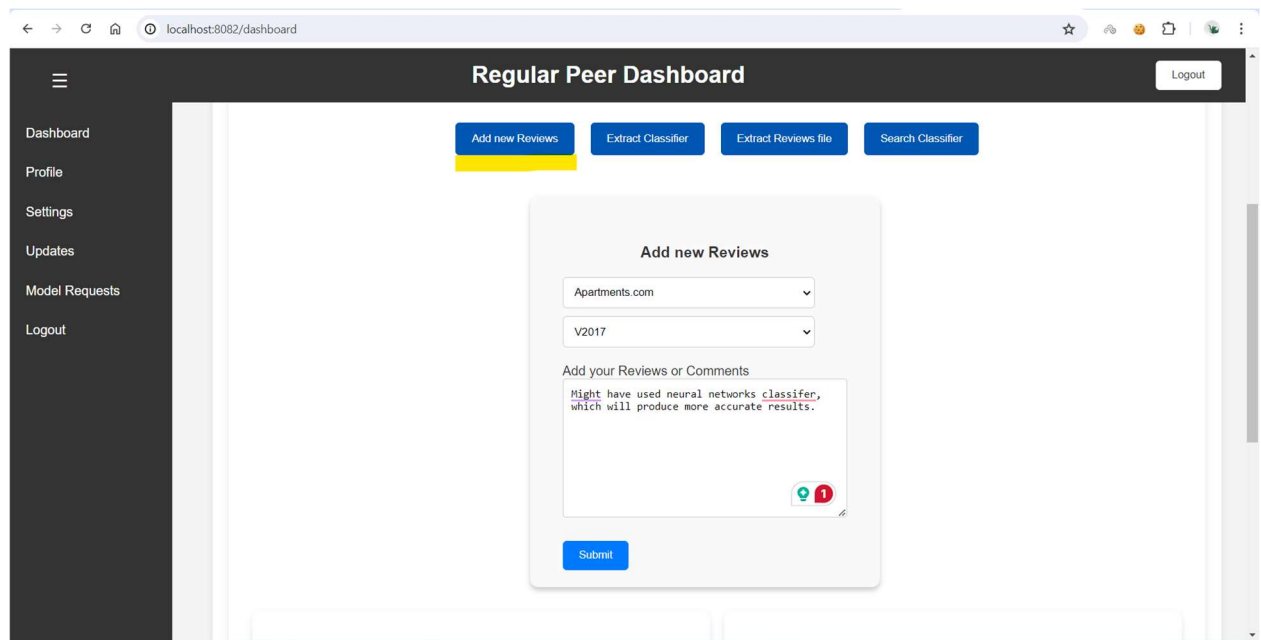
You can also check the actual blockchain folder:
superPeer1 → Blockchain or superPeer2 → Blockchain

AparV2017.json	7/29/2024 10:09 AM	JSON Source File	63 KB
AparV2021.json	7/28/2024 5:29 PM	JSON Source File	18 KB
AparV2023.json	7/29/2024 9:58 AM	JSON Source File	20 KB
RealV2018.json	7/29/2024 11:26 AM	JSON Source File	72 KB

6.2 Test Case 2: Review Comments:

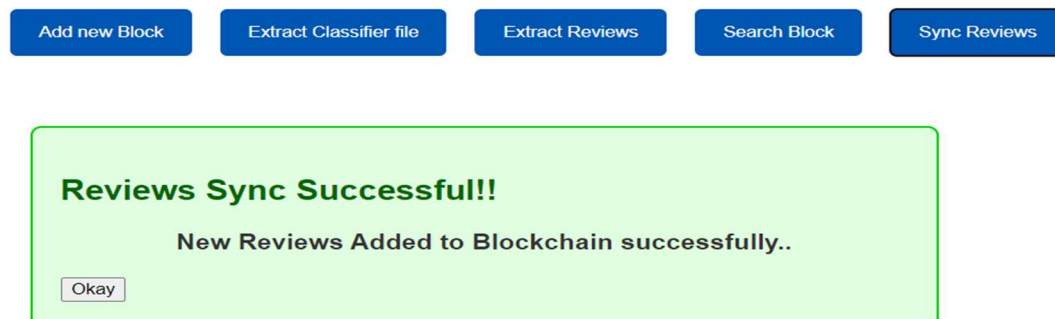
Step 1: As the Regular peer application has started, let's login to Regular peer running on localhost:8082 with the credentials above.

Step 2: You will be on the regular peer dashboard, once you login. Find the Add reviews button. Here as a test case, I'm adding reviews for Apartments.com model, version V2017, click submit



The reviews are sent to super peer 1, it will now add the reviews to blockchain only after you synchronize the reviews this is done next step:

Step 3: go to the super peer dashboard, you can see the 'sync reviews' button, click it. Success container is as shown, click Okay.



The reviews are now synchronized with all the peers. You can verify this in transactions page:

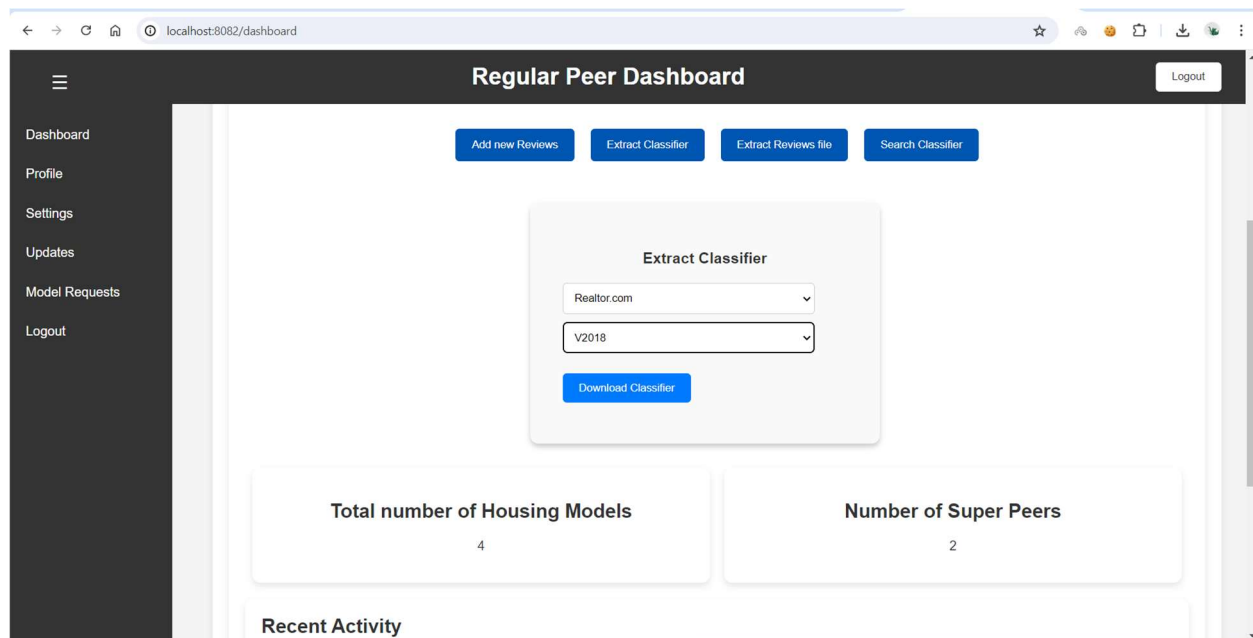
Trans15	Regular Peer	superPeer1	Reviews received for	AparV2017
Trans16	superPeer1	All peers	New reviews Synced	AparV2017

6.3 Test Case 3: Extracting ML Classifier

Step 1: Go to Regular peer dashboard, you will find the 'Extract Classifier' button, click it.

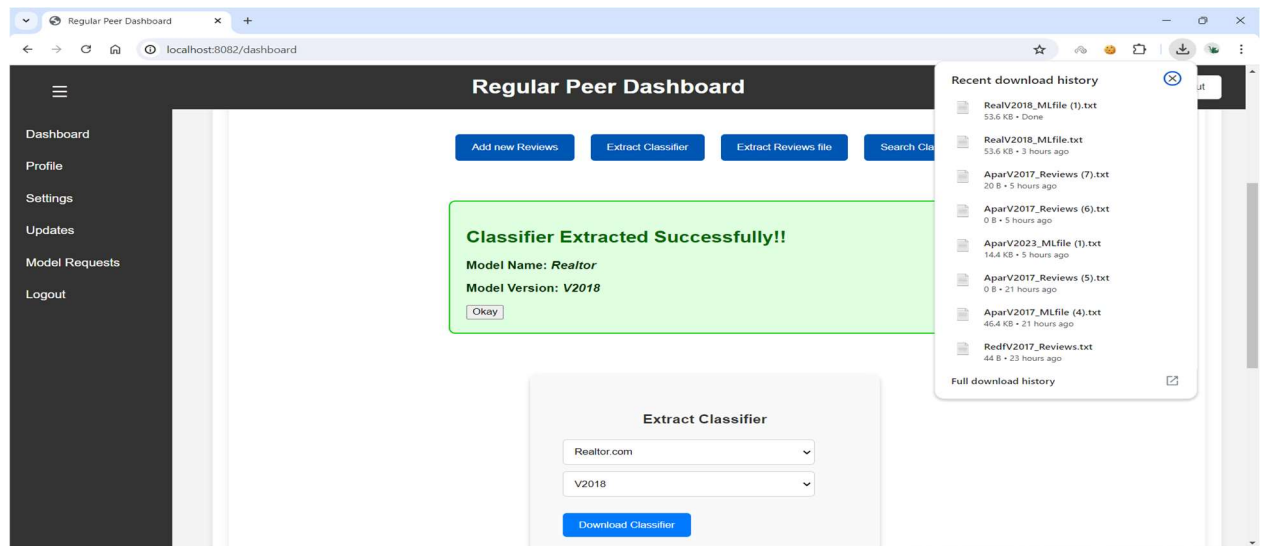
Step 2: Container is as shown, choose the model file you want to extract, version of it.

Let's extract ML file for Realtor.com and version V2018.



Step 3: after you select all the options, click 'Download Classifier', a success message is as shown. Click Okay. ML file is downloaded as a text readable file, you will find this file in your Downloads folder.

Below is the picture showing the success page for downloading a classifier:

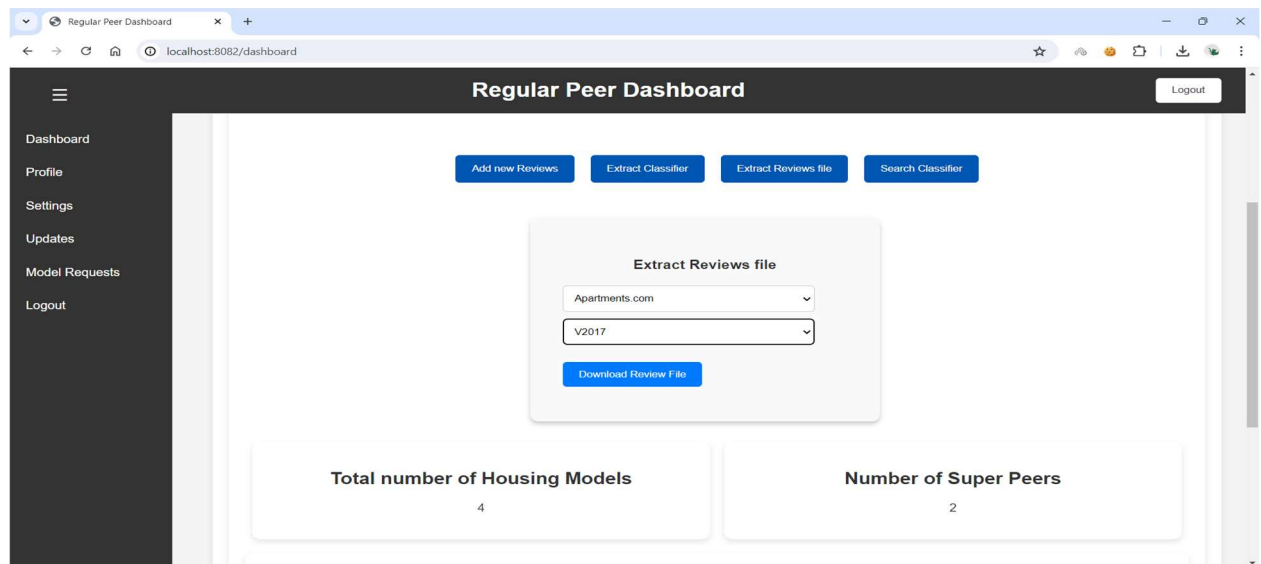


6.4 Test Case 4: Extracting Reviews

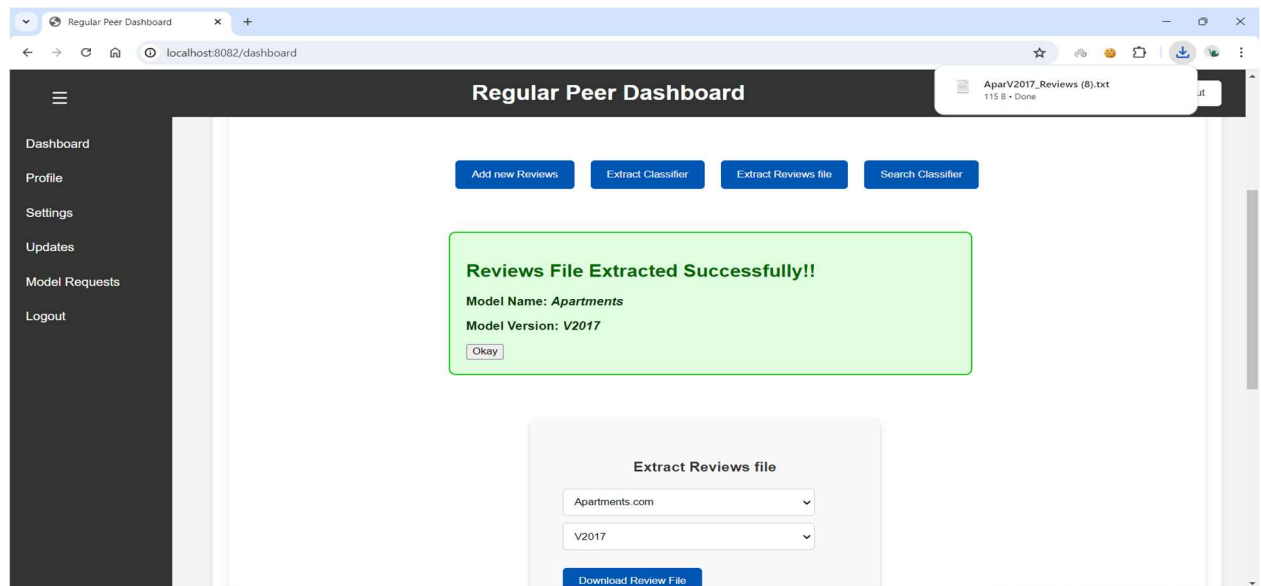
Step 1: Go to Regular peer dashboard, you will find the 'Extract Reviews file' button, click it.

Step 2: Container is as shown, choose the model file you want to extract, version of it.

Let's extract Reviews file for Apartments.com and version V2017.

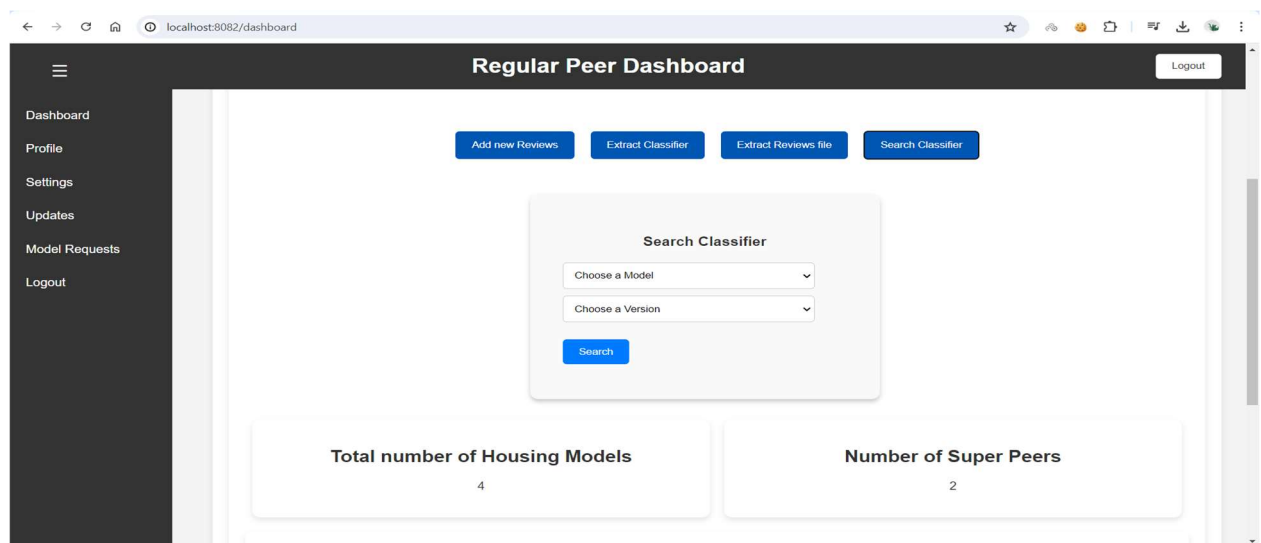


Step 3: after you select all the options, click 'Download Reviews file', a success message is as shown. Click Okay. Reviews file is downloaded as a text readable file, you will find this file in your Downloads folder.

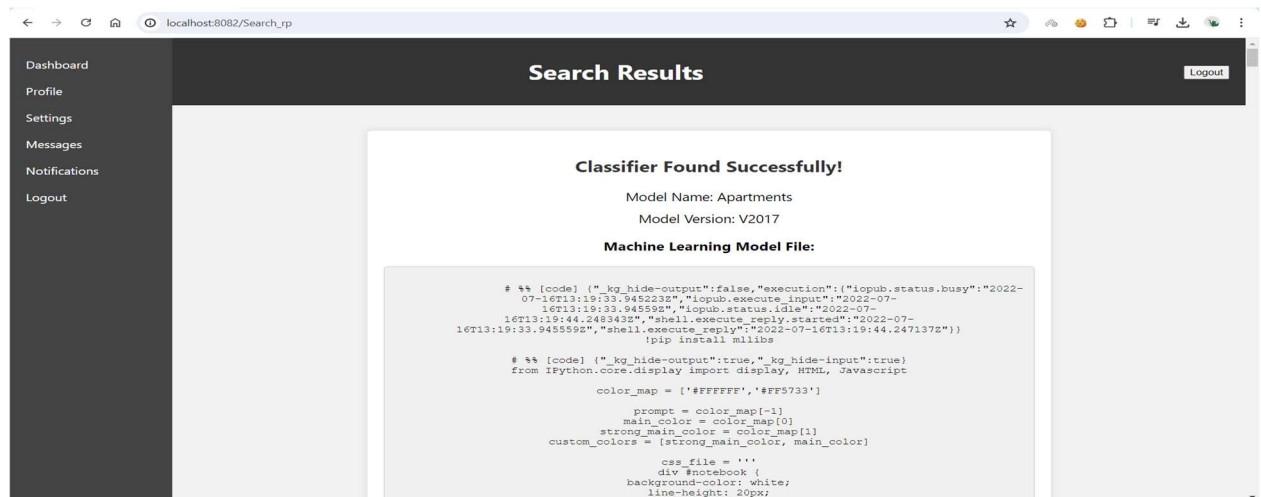


6.5 Test Case 5: Search

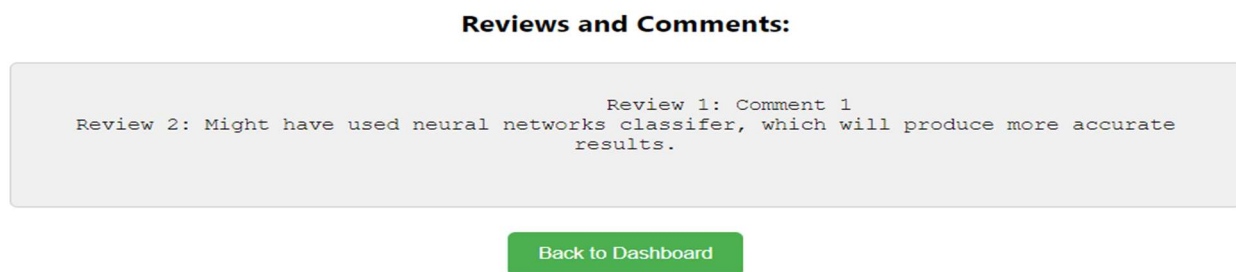
Step 1: In the Regular Peer we have the last option that is to search, find a block content based on the model's name and model version, click Search Classifier.



Step 2: After choosing a model name and model version, click Search. You will be redirected to the success page with ML file and Reviews file of the model you choose.



Reviews and Comments Section:



6.6 Test Case 6: Transactions

To fetch the transactions, please follow the below steps:

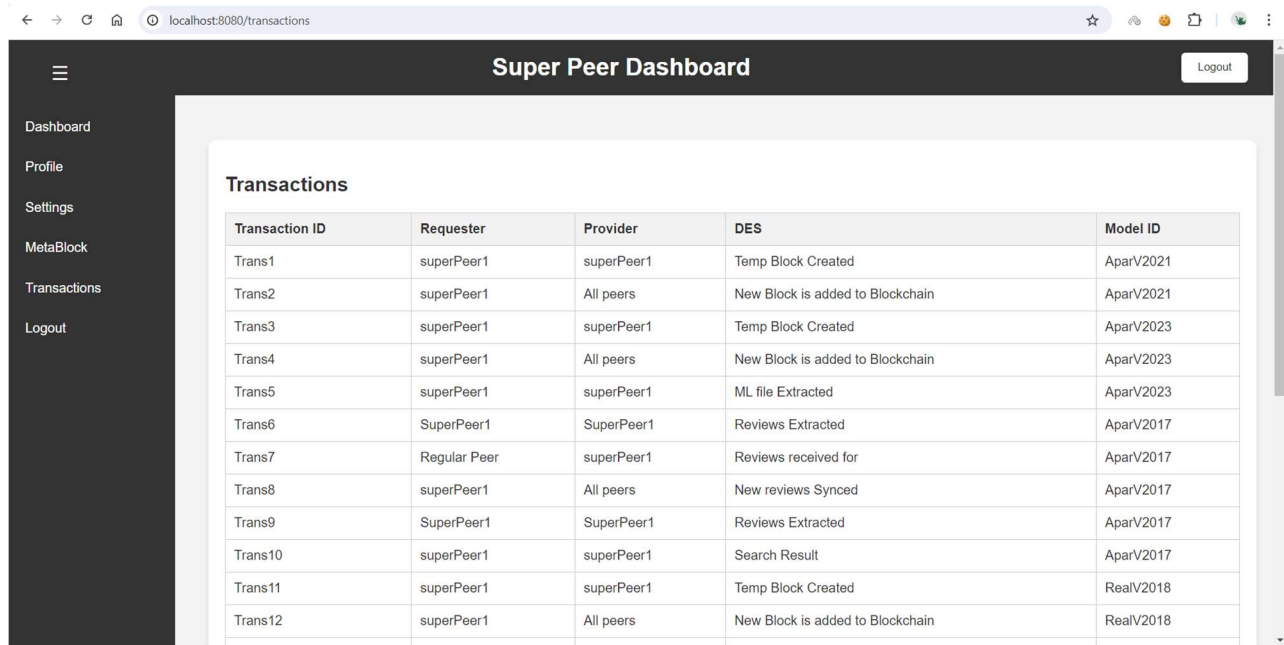
After logging into the super peer(any):

Step 1: In the Super Peer Dashboard: On the left navigation panel you will find the 'Transactions' button.

Step 2: Click the button, this will take you to the transactions page.

In the transactions page you will find all the transactions that are made previously.

Below is the screenshot of the transactions page:



The screenshot shows a web browser at localhost:8080/transactions displaying the 'Super Peer Dashboard'. The dashboard has a dark sidebar with navigation links: Dashboard, Profile, Settings, MetaBlock, Transactions, and Logout. The main content area is titled 'Transactions' and contains a table with 12 rows of transaction data.

Transaction ID	Requester	Provider	DES	Model ID
Trans1	superPeer1	superPeer1	Temp Block Created	AparV2021
Trans2	superPeer1	All peers	New Block is added to Blockchain	AparV2021
Trans3	superPeer1	superPeer1	Temp Block Created	AparV2023
Trans4	superPeer1	All peers	New Block is added to Blockchain	AparV2023
Trans5	superPeer1	superPeer1	ML file Extracted	AparV2023
Trans6	SuperPeer1	SuperPeer1	Reviews Extracted	AparV2017
Trans7	Regular Peer	superPeer1	Reviews received for	AparV2017
Trans8	superPeer1	All peers	New reviews Synced	AparV2017
Trans9	SuperPeer1	SuperPeer1	Reviews Extracted	AparV2017
Trans10	superPeer1	superPeer1	Search Result	AparV2017
Trans11	superPeer1	superPeer1	Temp Block Created	RealV2018
Trans12	superPeer1	All peers	New Block is added to Blockchain	RealV2018

7. Conclusion

In conclusion, this project has successfully demonstrated the integration of blockchain technology with machine learning to enhance transparency, security, and trust in ML predictive models, particularly in the context of house price prediction. By leveraging a decentralized architecture with super peers and regular peers, we have created a robust platform where ML classifiers can be securely uploaded, validated, and distributed.

Key Achievements:

- 1. Transparency and Trust:** The blockchain ensures that all ML classifiers and their updates are transparent and traceable. Customers can trust the integrity of the classifiers due to the peer-review and approval processes embedded within the blockchain.
- 2. Security and Immutability:** The use of blockchain technology provides a secure and immutable record of all transactions, ensuring that ML classifiers cannot be tampered with once they are added to the blockchain.
- 3. Collaboration and Validation:** Super peers play a crucial role in reviewing and validating new classifiers, ensuring that only high-quality, reliable models are distributed. This collaborative approach enhances the overall quality and reliability of the models.
- 4. Scalability and Redundancy:** The architecture ensures scalability by allowing new peers to join the network and redundancy by maintaining multiple copies of the blockchain across super peers.

This ensures data consistency and availability.

Implementation Insights:

- **Super Peers:** Successfully implemented a prototype with super peers who can upload, review, and synchronize ML classifiers.
- **Blockchain Infrastructure:** Developed a simple yet effective blockchain that securely stores ML classifiers as executable code.
- **User Interface:** Created user-friendly interfaces for both super peers and regular peers to interact with the system.
- **Data Synchronization:** Ensured that all peers have access to the most up-to-date data, maintaining consistency across the network.

By integrating blockchain technology with machine learning, we have taken a significant step towards building a more transparent, secure, and collaborative future for predictive modeling. This project showcases the potential of combining these advanced technologies to create a system that benefits all stakeholders, from developers to end-users, ensuring that ML models are reliable, validated, and trustworthy.

8. References:

“Deploying Trusted and Immutable Predictive Models on a Public Blockchain Network”:

<https://www.astesj.com/v09/i03/p07/>

<https://link.springer.com/article/10.1186/S40854-019-0147-Z>

<https://www.baeldung.com/java-blockchain>

<https://www.geeksforgeeks.org/implementation-of-blockchain-in-java/>

<https://medium.com/programmers-blockchain/create-simple-blockchain-java-tutorial-from-scratch-6eeed3cb03fa>

<https://www.kaggle.com/code>