**FinChain : Credit Recommendations for Financial Institutions with Blockchain-based System**

A PROJECT REPORT

in the partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

MAY 2023

**BONAFIDE CERTIFICATE**

**ABSTRACT**

In this paper, we propose FinChain, a deep-learningbased credit-recommender scheme for public blockchain to facilitate smart lending operations between prospective borrowers (PB) and prospective lenders (PL) to eliminate the need of third party credit-rating agencies (CRAs) for credit-score (CS) generation. Thus loan grants to PB from PL is secured, authorized, and automated so as to expedite the disbursement process. FinChain stores PB historical transactions, current assets, and liabilities as time-series sequenced data in a public blockchain. The sequenced data is fetched from blockchain by a long-short term memory (LSTM) model that generates CS for loan recommendations based on proposed lending algorithms for PB and PL. To ensure real-time updation of CS, edge-weights are updated based on boolean indicators from PB and PL, which indicates the successful repayments and loan-defaults. The process is iterated to improve the accuracy of edge-weights and generated CS o ensures the correct credibility of PB for future lending. Smart contracts (SC) are proposed for automatic setup of loan repayments between PB and PL. To model the LSTM recommender scheme, a German credit dataset from UCI repository is considered with 1000 credit-histories of PB, with 700 successful repayments and 300 defaults. FinChain achieves an accuracy of 97.5% in comparison to conventional approaches with an Fmeasure of 0.98304. The security evaluation of FinChain shows that it has computation cost of 20.96 ms and communication cost of 121 bytes compared to other state-of-the-art approaches

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER No** | **TITLE** | **PAGE** |
|  | LIST OF FIGURES | Vi |
|  | LIST OF ABBREVATIONS | Vii |
| **1** | **INTRODUCTION** |  |
| 1.1 | About the Project |  |
| **2** | **SYSTEM ANALYSIS** |  |
| 2.1 | Existing system |  |
| 2.2 | Proposed system |  |
| **3** | **REQUIREMENTS SPECIFICATION** |  |
| 3.1 | Introduction |  |
| 3.2 | Hardware and Software specification |  |
| 3.3 | Technologies Used |  |
| 3.3.1 | Java |  |
| 3.3.3.1 | Introduction to Java |  |
| 3.3.1.2 | Working of Java |  |
| **4** | **Project Purpose and Scope** |  |
| 4.1 | Purpose |  |
| 4.2 | Project Scope |  |
| 4.3 | Product Perspective |  |
| 4.4 | System Features |  |
| 4.5 | Design and Implementation Constraints |  |
| 4.6 | Other Nonfunctional Requirements |  |
| **5** | **SYSTEM DESIGN** |  |
| 5.1 | Architecture Diagram |  |
| **6** | **SYSTEM DESIGN – DETAILED** |  |
| 6.1 | Modules |  |
| 6.2 | Module explanation |  |
| **7** | **CODING AND TESTING** |  |
| 7.1 | Coding |  |
| 7.2 | Coding standards |  |
| 7.3 | Test procedure |  |
| 7.4 | Test data and output |  |
|  | **SOURCE CODE** |  |
|  | **SNAP SHOTS** |  |
|  | **REFERENCES** |  |

**LIST OF FIGURES**

5.1 System Design

5.2 Sequence Diagram

5.3 Use Case Diagram

5.4 Activity Diagram

5.5 Collaboration Diagram:

5.6 Data Flow Diagram

5.7 Class Diagram

**LIST OF ABBREVATIONS**

JDK Java Development Toolkit

DEX Dalvik Executables

TCP Transmission Control Protocol

IP Internet Protocol

**HTTP** Hyper Text Transfer Protocol

**ADT** Android Development Tool

**CHAPTER 1**

**INTRODUCTION**

**Aim:**

The main aim of this project to remove the third party agent between the perspective lenders and perspective borrowers.

**Synopsis:**

In this paper, we propose FinChain, a deep-learning based credit-recommender scheme for public blockchain to facilitate smart lending operations between prospective borrowers (PB) and prospective lenders (PL) to eliminate the need of third party credit-rating agencies (CRAs) for credit-score (CS) generation. Thus loan grants to PB from PL is secured, authorized, and automated so as to expedite the disbursement process. FinChain stores PB historical transactions, current assets, and liabilities as time-series sequenced data in a public blockchain. The sequenced data is fetched from blockchain by a long-short term memory (LSTM) model that generates CS for loan recommendations based on proposed lending algorithms for PB and PL. To ensure real-time updation of CS, edge-weights are updated based on boolean indicators from PB and PL, which indicates the successful repayments and loan-defaults. The process is iterated to improve the accuracy of edge-weights and generated CS to ensures the correct credibility of PB for future lending. Smart contracts (SC) are proposed for automatic setup of loan repayments between PB and PL. To model the LSTM recommender scheme, a German credit dataset from UCI repository is considered with 1000 credit-histories of PB, with 700 successful repayments and 300 defaults. FinChain achieves an accuracy of 97.5% in comparison to conventional approaches with an Fmeasure of 0.98304. The security evaluation of FinChain shows that it has computation cost of 20.96 ms and communication cost of 121 bytes compared to other state-of-the-art approaches.

**CHAPTER 2**

**SYSTEM ANALYSIS**

**2.1 EXISTING SYSTEM**

In existing system, d the overall loan disbursement cycle is slow and tedious in nature. This is due to complex co-relations that exist among diverse set of parameters like- previous credit histories, defaults, and length of repayment cycles. Also, loan grants are subjective to PL profit margins, with different levels of uncertainty, depending on PL prospective. For some PL, loss (defaults) is more critical than gains (repayments), and vice-versa. Thus, the recommender models might assign different scores to same PB, which results in the proportionality of the assessed risk by PL. This approach is flawed as even a missed repayment can affect CS resulting in the calculation of high risk. Also, CS generated through recommender models does not take into account new PB having a good reputation and high assets as their credit histories are absent.

**Problem Definition**

* Overall loan disbursement cycle is slow and tedious in nature.
* This approach is flawed as even a missed repayment can affect CS resulting in the calculation of high risk.

**2.2PROPOSED SYSTEM**

In proposed System, loan disbursement cycle is high. Perspective lenders and perspective borrowers have direct communication. So, that they can know their details clearly. In this approach we implement a sentiment analysis to read a borrowers reviews. For using sentiment analysis we can find out the total percentage of positive reviews negative reviews. Based on the reviews and positive negative reviews percentage PL will decide to lend the loan to that particular user or not. So that maximum users can get a loan without any difficulties. We will store the perspective lender lending details and perspective borrower’s payment history in block-chain.

**Advantage**

* Every perspective borrower has a possibility to get a loan from lender.
* Perspective lenders and perspective borrowers had to know each other’s identity.

**CHAPTER 3**

**REQUIREMENT SPECIFICATIONS**

**3.1 INTRODUCTION**

IN modern financial institutions, loan disbursement to PB by financial stakeholders like banks, insurance agencies, and PL are facilitated through electronic loan records. The records consists of PB sensitive information, such as their unique identification number, address, income-tax returns, and financial assets. PL shares the loan records to third-party CRAs that evaluate risk in granting loan to PB by generating a CS for PB. To generate CS, CRAs charge hefty amounts from PL that it enforces on PB once loan grants are finalized. CRAs forms recommender model on PB historical data of financial transactions that includes details like- age, loan purpose, previous credit and lending histories, nature of job, current assets and liabilities, and bank account balance to compute CS. To facilitate the same, recommender systems employ two set of filtering mechanisms on PB loan records- collaborative and content based filters. Collaborative filters analyze PB current assets and liabilities, job type (business/salaried), and age to determine risk in loan grants, whereas content filters analyzes PB previous repayments and defaults to compute a lending score. The risk values and lending scores goes through multiple iterations (cycles) of statistical inferences to finalize CS for PB. Based on CS, PL forms an informed decision on amount of loan grant, interest-rate fixation, and repayment period of the granted loan. The current loan recommender models are complex, and the overall loan disbursement cycle is slow and tedious in nature. This is due to complex co-relations that exist among diverse set of parameters like- previous credit histories, defaults, and length of repayment cycles. Also, loan grants are subjective to PL profit margins, with different levels of uncertainty, depending on PL prospective. For some PL, loss (defaults) are more critical than gains (repayments), and vice-versa. Thus, the recommender models might assign different scores to same PB, which results in the proportionality of the assessed risk by PL. This approach is flawed as even a missed repayment can affect CS resulting in the calculation of high risk. Also, CS generated through recommender models does not take into account new PB having a good reputation and high assets as their credit histories are absent. This assigns low CS to new PB, which effects future lending scope of such PB. Apart from complexity of recommender models, credit repositories of PB are stored in central servers and are subject to privacy and security issues. CRAs, PL and other financial stakeholders like insurance agencies, and banks, fetches PB sensitive information from these repositories using an open unsecured channels, i.e., the Internet [1]. Thus, any malicious intruder can also gain access to these repositories, which raises security concerns like identity theft, selling of PB records for personal motives, and data breaches. As per the reports of the Federal Trade Commission, 148 million customers data was breached at Equifax, one of the largest CRAs in the United States, amounting to $425 billion loss to the industry [2]. According to United States InfoSearch, Experian databases were also breached by running malicious service daemons on workstations that allowed identity theft of 200 million consumer records, with the loss approximated close to $1.3 billion to the CRA [3].

**3.2 HARDWARE AND SOFTWARE SPECIFICATION**

**3.2.1 HARDWARE REQUIREMENTS**

* Hard Disk : 80GB and Above
* RAM : 4GB and Above
* Processor : P IV and Above

**3.2.2 SOFTWARE REQUIREMENTS**

* Windows 7 and above(64 bit)
* JDK 1.8
* Python 3.6.3
* Tomcat 9.0.26
* MySQL

**3.3.JAVA**

Java is an object-oriented programming language developed initially by James Gosling and colleagues at Sun Microsystems. The language, initially called Oak (named after the oak trees outside Gosling's office), was intended to replace C++, although the feature set better resembles that of Objective C.

**3.3.1.1 INTRODUCTION TO JAVA**

Java has been around since 1991, developed by a small team of Sun Microsystems developers in a project originally called the Green project. The intent of the project was to develop a platform-independent software technology that would be used in the consumer electronics industry. The language that the team created was originally called Oak.

The first implementation of Oak was in a PDA-type device called Star Seven (\*7) that consisted of the Oak language, an operating system called GreenOS, a user interface, and hardware. The name \*7 was derived from the telephone sequence that was used in the team's office and that was dialed in order to answer any ringing telephone from any other phone in the office.

Around the time the First Person project was floundering in consumer electronics, a new craze was gaining momentum in America; the craze was called "Web surfing." The World Wide Web, a name applied to the Internet's millions of linked HTML documents was suddenly becoming popular for use by the masses. The reason for this was the introduction of a graphical Web browser called Mosaic, developed by ncSA. The browser simplified Web browsing by combining text and graphics into a single interface to eliminate the need for users to learn many confusing UNIX and DOS commands. Navigating around the Web was much easier using Mosaic.

It has only been since 1994 that Oak technology has been applied to the Web. In 1994, two Sun developers created the first version of Hot Java, and then called Web Runner, which is a graphical browser for the Web that exists today. The browser was coded entirely in the Oak language, by this time called Java. Soon after, the Java compiler was rewritten in the Java language from its original C code, thus proving that Java could be used effectively as an application language. Sun introduced Java in May 1995 at the Sun World 95 convention.

Web surfing has become an enormously popular practice among millions of computer users. Until Java, however, the content of information on the Internet has been a bland series of HTML documents. Web users are hungry for applications that are interactive, that users can execute no matter what hardware or software platform they are using, and that travel across heterogeneous networks and do not spread viruses to their computers. Java can create such applications.

**3.3.1.2 WORKING OF JAVA**

For those who are new to object-oriented programming, the concept of a class will be new to you. Simplistically, a class is the definition for a segment of code that can contain both data (called attributes) and functions (called methods).

When the interpreter executes a class, it looks for a particular method by the name of **main,** which will sound familiar to C programmers. The main method is passed as a parameter an array of strings (similar to the argv [] of C), and is declared as a static method.

To output text from the program, we execute the **println** method of **System.out,** which is java’s output stream. UNIX users will appreciate the theory behind such a stream, as it is actually standard output. For those who are instead used to the Wintel platform, it will write the string passed to it to the user’s program.

Java consists of two things :

* + Programming language
  + Platform

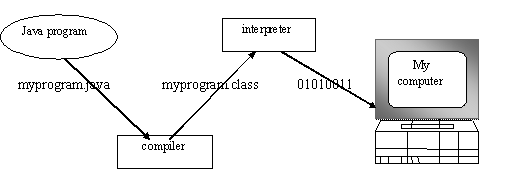
**3.3.1.3 THE JAVA PROGRAMMING LANGUAGE**

Java is a high-level programming language that is all of the following:

* + Simple
  + Object-oriented
  + Distributed
  + Interpreted
  + Robust
  + Secure
  + Architecture-neutral
  + Portable
  + High-performance
  + Multithreaded
  + Dynamic

The code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Java is unusual in that each Java program is both co implied and interpreted. With a compiler, you translate a Java program into an intermediate language called **Java byte codes** – the platform independent codes interpreted by the Java interpreter. With an interpreter, each Java byte code instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. This figure illustrates how it works:

****

**Fig.3.1**

You can think of Java byte codes as the machine code instructions for the **Java Virtual Machine (JVM).** Every Java interpreter, whether it’s a Java development tool or a Web browser that can run Java applets, is an implementation of JVM. That JVM can also be implemented in hardware. Java byte codes help make “write once, run anywhere” possible.

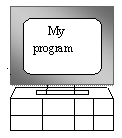
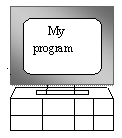
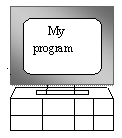
You can compile your Java program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the JVM. For example, that same Java program can e run on Windows NT, Solaris and Macintos

Complier

Interpreter

Interpreter

Interpreter

**  **

**PC-Compatible Sun Ultra Solaris Power macintosh**

**Windows NT System 8**

**3.3.1.4 THE JAVA PLATFORM**

A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components :

* The Java Virtual Machine (JVM)
* The Java Application Programming Interface (Java API)

You’ve already been introduced to the JVM. It’s the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries **(packages)** of related components. The following figure depicts a Java program, such as an application or applet, that’s running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.

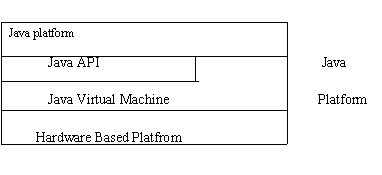
****

Fig.3.3

As a platform-independent environment, Java can be a bit slower than native code. However, smart compliers, weel-tuned interpreters, and just-in-time byte compilers can bring Java’s performance close to that of native code without threatening portability.

**3.3.2 APACHE TOMCAT SERVER**

Apache Tomcat (formerly under the Apache Jakarta Project; Tomcat is now a top level project) is a web container developed at the Apache Software Foundation. Tomcat implements the servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.

**Environment**  
 Tomcat is a web server that supports servlets and JSPs. Tomcat comes with the Jasper compiler that compiles JSPs into servlets.

The Tomcat servlet engine is often used in combination with an Apache web server or other web servers. Tomcat can also function as an independent web server. Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high-availability environments.

Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM.

**Product features**

Tomcat 3.x (initial release)

* implements the Servlet 2.2 and JSP 1.1 specifications
* servlet reloading
* basic HTTP functionality Tomcat 4.x
* implements the Servlet 2.3 and JSP 1.2 specifications
* servlet container redesigned as Catalina
* JSP engine redesigned as Jasper
* Coyote connector
* Java Management Extensions (JMX), JSP and Struts-based administration
* Tomcat 5.x
* implements the Servlet 2.4 and JSP 2.0 specifications
* reduced garbage collection, improved performance and scalability
* native Windows and Unix wrappers for platform integration
* faster JSP paring

**History** Tomcat started off as a servlet specification implementation by James Duncan Davidson, a software architect at Sun. He later helped make the project open source and played a key role in its donation by Sun to the Apache Software Foundation.

Davidson had initially hoped that the project would become open-sourced and, since most open-source projects had O'Reilly books associated with them featuring an animal on the cover, he wanted to name the project after an animal. He came up with Tomcat since he reasoned the animal represented something that could take care of and fend for itself. His wish to see an animal cover eventually came true when O'Reilly published their Tomcat book with a tomcat on the cover.

### 3.3.2

Introduction to Blockchain

With the emergence of Digital Currency (aka Crypto currency), several enterprises or financial institutions are experimenting with the Distributed Ledger system as a trusted way to track the ownership of the assets without any central authority.

The core system behind the new currency system is Blockchain technology. A walkthrough of the basic building blocks of the Blockchain technology is described below.

A Blockchain is basically a chain of Blocks. Blocks are hashed using SHA-256 hashing algorithm to generate the signature of the data associated with it.

Imagine a Blockchain as a linked-list whose node contains below attributes:

1. Block number – a sequence number (monotonically increasing) assigned to the block
2. Nonce – a random number which is used to generate Hash (as in #5) value which starts with 4 zeroes (0000). The process of generating this Nonce is called Mining.
3. Data – the actual user data associated with the block
4. Prev – contains the Hash of the previous block (e.g. current block # -1). The value for the first block in the chain is 64 zeroes (0000000000000000000000000000000000000000000000000000000000000000).
5. Hash – current block’s Hash value (generated using SHA-256). All of the above attributes excluding Hash e.g. Block #, Nonce, data, Prev are used to calculate the Hash of this block.

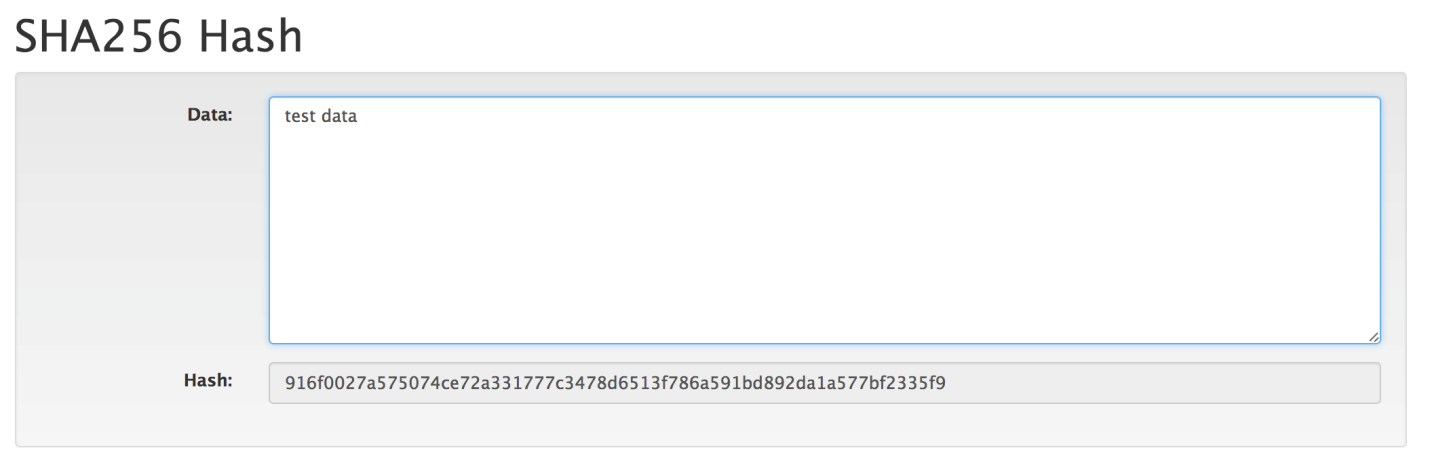
[#=1, Nonce=3409, Data=x, Prev=00..0, Hash=0000ffgr5rg67j] <- [#=2, Nonce=4986, Data=x, Prev=0000ffgr5rg67j, Hash=000045tggr5rg..77yh] <-……and the chain goes on…

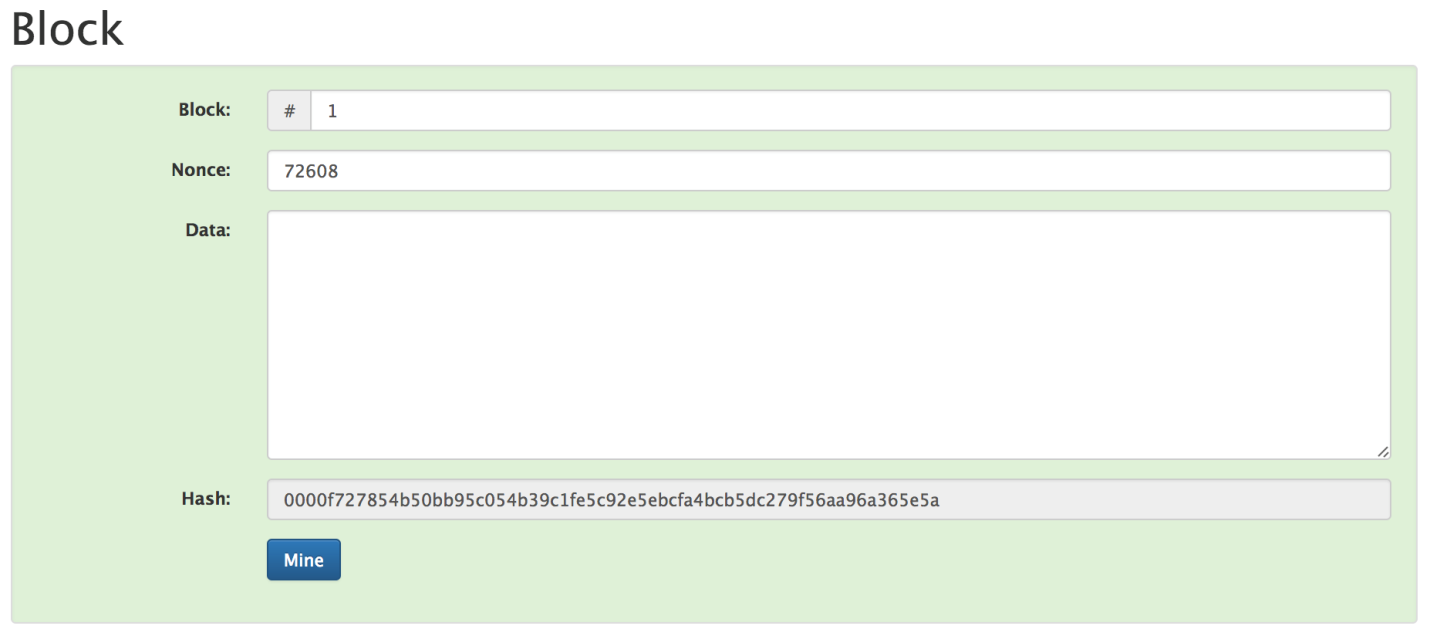
e.g. in above block #1, the value for Hash=0000ffgr5rg67j is generated using the values 1,3409,x,00..0. In case value for any of these 4 attributes changes, it will change the Hash value of this block. Once the Hash value of this Block changes (e.g. from 0000ffgr5rg67j to 34sdffgr5rg67j), it will break the next Block (#2) as its Prev field will point to invalid Hash (0000ffgr5rg67j doesn’t exist anymore). This leads to a ripple effect and turns whole chain as invalid/tampered.

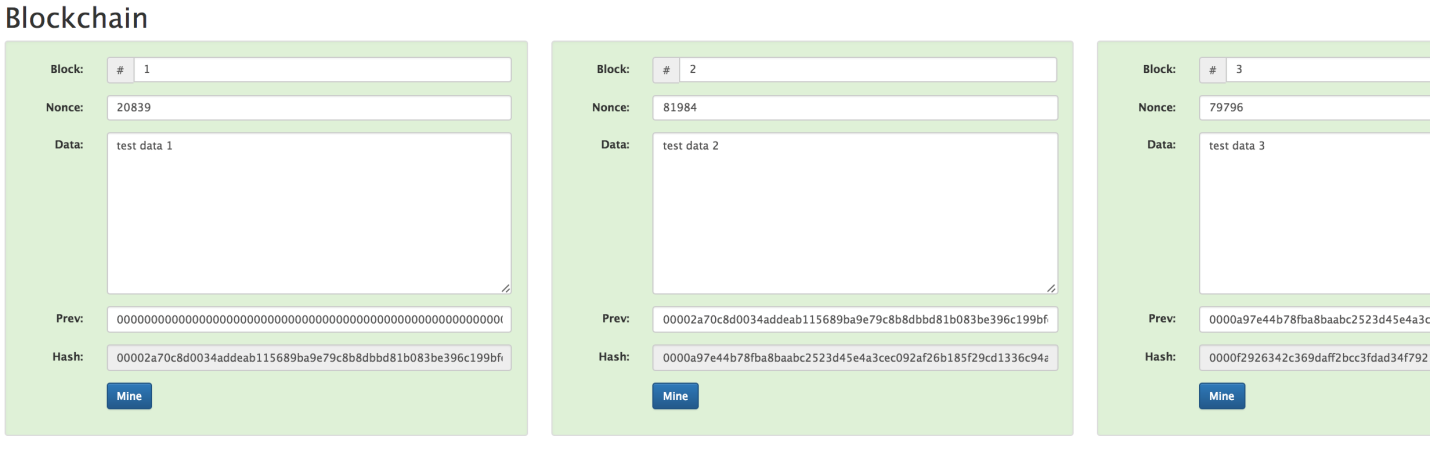
One way to fix it is to run mining and recalculate the Hash value of Block #1 which basically will generate new value for Nonce and hence leading to a valid Hash value which starts with 4 zeroes. Copying this to next Block #2’s Prev field will fix these 2 Blocks. However in order to fix the whole Blockchain, we need to continue with this process for all the Blocks in the chain so that all Blocks point to new & valid Hash codes of their previous blocks.

The cost of fixing the tampered Blockchain as described in above process is very high. Because we have to go and fix the Chain from the starting Block to the last one. In case the Chain is large, it becomes costly operation. In case of Distributed Blockchain where several Peers are involved in the process and keeping the copy of the Blockchain, the repairing the Blocks becomes even more costly operation.

The other and more efficient process is to come up with the compensating data and add this Block at the end of the Chain. E.g. In case your Chain contains the financial transaction (money movement) in Data field of the Block, then instead of fixing each of the Block’s Data with corrected financial transaction, come up with the adjusted financial transaction (aka compensating transaction) and create a Block (with Data=adjusted transaction record) and add this Block to the Blockchain (adds to the end of the Chain).







**CHAPTER 4**

**4.1 Project Purpose and Scope**

**4.1 Purpose**

The main aim of this project to remove the third party agent between the perspective lenders and perspective borrowers.

**4.2 Project Scope**

In this paper, we propose FinChain, a deep-learning based credit-recommender scheme for public blockchain to facilitate smart lending operations between prospective borrowers (PB) and prospective lenders (PL) to eliminate the need of third party credit-rating agencies (CRAs) for credit-score (CS) generation. Thus loan grants to PB from PL is secured, authorized, and automated so as to expedite the disbursement process. FinChain stores PB historical transactions, current assets, and liabilities as time-series sequenced data in a public blockchain. The sequenced data is fetched from blockchain by a long-short term memory (LSTM) model that generates CS for loan recommendations based on proposed lending algorithms for PB and PL. To ensure real-time updation of CS, edge-weights are updated based on boolean indicators from PB and PL, which indicates the successful repayments and loan-defaults. The process is iterated to improve the accuracy of edge-weights and generated CS to ensures the correct credibility of PB for future lending. Smart contracts (SC) are proposed for automatic setup of loan repayments between PB and PL. To model the LSTM recommender scheme, a German credit dataset from UCI repository is considered with 1000 credit-histories of PB, with 700 successful repayments and 300 defaults. FinChain achieves an accuracy of 97.5% in comparison to conventional approaches with an Fmeasure of 0.98304. The security evaluation of FinChain shows that it has computation cost of 20.96 ms and communication cost of 121 bytes compared to other state-of-the-art approaches.

**4.3 Product Perspective**

Smart lending operations in modern financial institutions are based on recommender models to generate CS to facilitate loan grants to PB. To ensure credibility in generation of CS, and transparency in evaluation of loan grants, the proposed scheme FinChain integrates blockchain and deep-learning to facilitate a trust based recommender ecosystem. In FinChain, LSTM model is applied to ensure accurate predictions on PB historical repositories and blockchain offers immutability in stored records. To improve the decision analytics over time, and minimize bias, a feedback system is evolved that updates edge-weights of LSTM in real-time, so CS are accurate and up-to-date. SC necessitates automated re-payments from PB to PL. The proposed results indicate the efficacy of the scheme against existing credit schemes. However, the LSTM model operates on feedbacks to update CS that induce latency overheads.

* 1. **4.4 SystemFeatures**

In existing system, d the overall loan disbursement cycle is slow and tedious in nature. This is due to complex co-relations that exist among diverse set of parameters like- previous credit histories, defaults, and length of repayment cycles. Also, loan grants are subjective to PL profit margins, with different levels of uncertainty, depending on PL prospective. For some PL, loss (defaults) is more critical than gains (repayments), and vice-versa. Thus, the recommender models might assign different scores to same PB, which results in the proportionality of the assessed risk by PL. This approach is flawed as even a missed repayment can affect CS resulting in the calculation of high risk. Also, CS generated through recommender models does not take into account new PB having a good reputation and high assets as their credit histories are absent.

**4.5 Design and Implementation Constraints**

**4.5.1 Constraints in Analysis**

* Constraints as Informal Text
* Constraints as Operational Restrictions
* Constraints Integrated in Existing Model Concepts
* Constraints as a Separate Concept
* Constraints Implied by the Model Structure

**4.5.2 Constraints in Design**

* Determination of the Involved Classes
* Determination of the Involved Objects
* Determination of the Involved Actions
* Determination of the Require Clauses
* Global actions and Constraint Realization

**4.5.3 Constraints in Implementation**

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

* 1. **Other Nonfunctional Requirements**

**4.6.1 Performance Requirements**

The application at this side controls and communicates with the following three main general components.

* embedded browser in charge of the navigation and accessing to the web service;
* Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

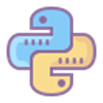
Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

**4.6.2 Safety Requirements**

* 1. The software may be safety-critical. If so, there are issues associated with its integrity level
  2. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.
  3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.
  4. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.
  5. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.
  6. Systems with different requirements for safety levels must be separated.
  7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

**CHAPTER 5**

**5.1 Architecture Diagram:**



Lender

Borrower

Register

Register



Upload Documents

Upload Documents

Analyze the user Request

Send loan Request to Lender

Analyze the reviews

Sentiment analysis

Based on borrower details lender can accept or decline the request

Accepted

Declined

Add monthly payment details and amount

Once get the money start to pay monthly interest for upcoming months.



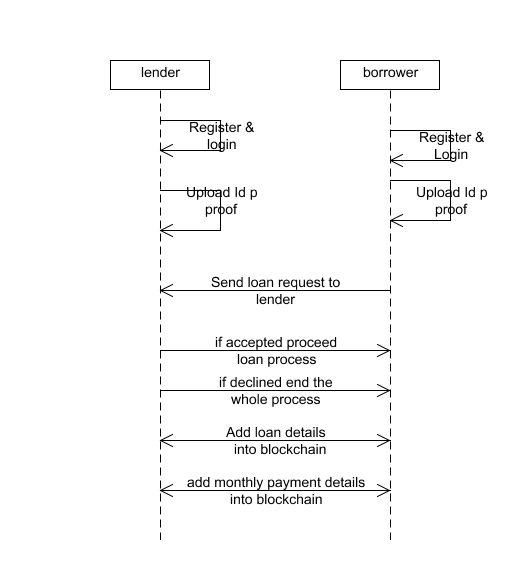
Store these Details into blockchain

End of process

**Fig: 5.1**

**5.2 Sequence Diagram:**

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.

****

**5.3 Use Case Diagram:**

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

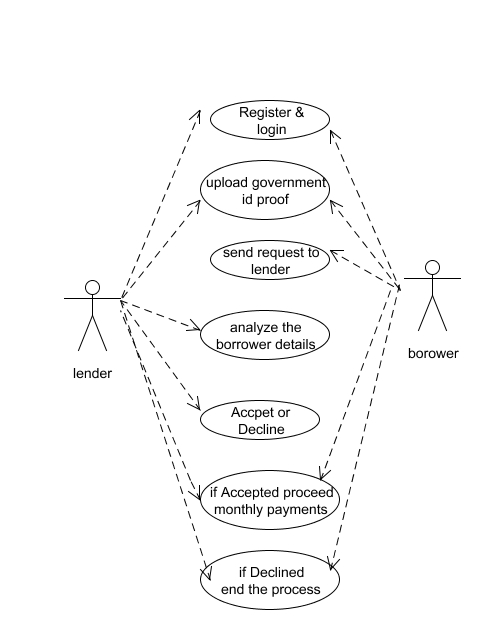
**5.3.1. USECASE DIAGRAM**

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

**Use case:** A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

**Actor:** An actor is a person, organization or external system that plays a role in one or more interaction with the system.

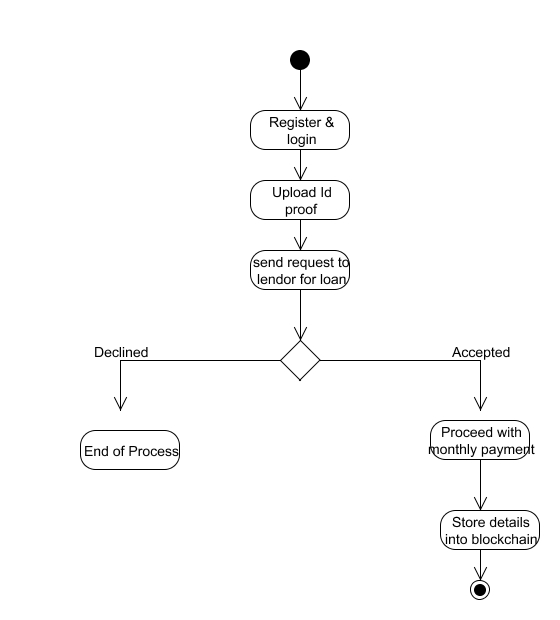
****

**5.4 Activity Diagram:**

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

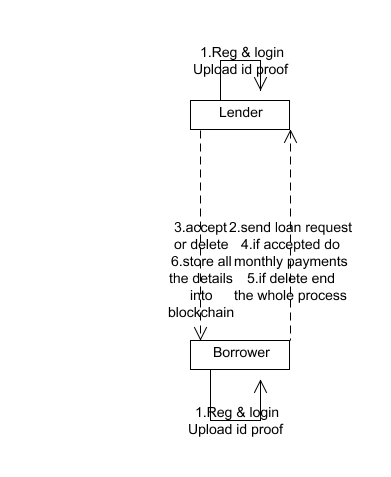
The most important shape types:

* Rounded rectangles represent activities.
* Diamonds represent decisions.
* Bars represent the start or end of concurrent activities.
* A black circle represents the start of the workflow.
* An encircled circle represents the end of the workflow.

****

**5.5 Collaboration Diagram:**

UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.

****

**5.6 DATA FLOW DIAGRAM:**

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its aspects. It is a preliminary step used to create an overview of the system which can later be elaborated DFDs can also be used for visualization of data processing.

**Level 0:**

Lender

Register

Login

Borrower

Register

Login

**Level 1:**

Lender

Register

Login

Borrower

Register

Login

Send loan request to lender

**Level 2:**

Lender

Register

Login

Borrower

Register

Login

Send loan request to lender

Analyze the requested borrower details

Accept or delete the request

**Level 3:**

Lender

Register

Login

Borrower

Register

Login

Send loan request to lender

Analyze the requested borrower details

Accept or delete the request

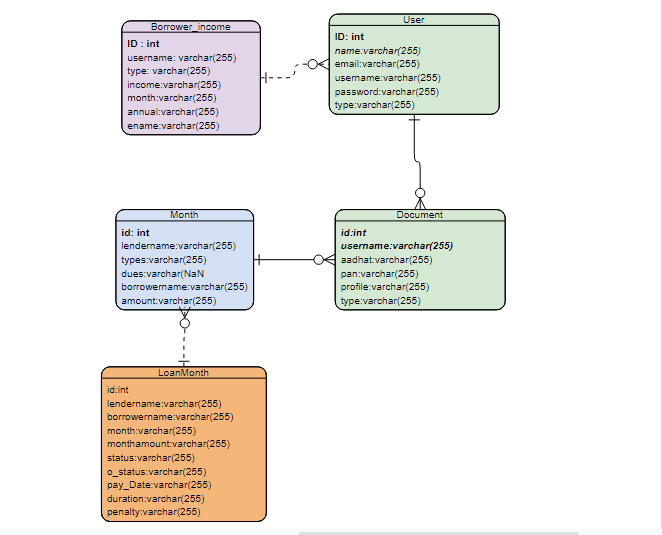
If accepted do monthly payments

If deleted end the whole process

Store all transaction details into blockchain

**5.7 Class Diagram**

A Class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

****

**CHAPTER 6**

**SYSTEM DESIGN**

In proposed System, loan disbursement cycle is high. Perspective lenders and perspective borrowers have direct communication. So, that they can know their details clearly. In this approach we implement a sentiment analysis to read a borrowers reviews. For using sentiment analysis we can find out the total percentage of positive reviews negative reviews. Based on the reviews and positive negative reviews percentage PL will decide to lend the loan to that particular user or not. So that maximum users can get a loan without any difficulties. We will store the perspective lender lending details and perspective borrower’s payment history in block-chain.

**6.1 MODULES**

* PB and PL Registration
* Loan Request by PB
* Sentiment Analysis
* Payment Tracking

**6.2 MODULE EXPLANATION:**

**6.2.1 PB and PL Registration**

In first module, Perspective lender and perspective borrower will register their personal information as well as income details. And also they will upload government issued documents for registration process.

**Loan Request by PB**

In second module when Perspective borrower login the registered perspective lenders details will shown as list. Borrower can choose the lender and he can check the lenders personal details and government Id proofs. Then he can send the request to the lenders. One PB should send request to one PL only.

**Sentiment Analysis**

In third module the requester borrower’s details will be list down in lenders side. Lender can check borrower’s personal info id proofs. Once he checked with the basic steps he will move on to sentiment analysis. Lender can check the positive reviews percentage, negative reviews percentage by using sentiment analysis.

**Payment Tracking**

In final module the borrower’s monthly payment will be stored in block-chain. So that lenders can know about the date of payment and whether it is late payment or on time payment.

**CHAPTER 7**

**CODING AND TESTING**

**7.1 CODING**

Once the design aspect of the system is finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

**7.2 CODING STANDARDS**

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Program should be simple, clear and easy to understand.

Naming conventions

Value conventions

Script and comment procedure

Message box format

Exception and error handling

**7.2.1 NAMING CONVENTIONS**

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

**Class names**

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

**Member Function and Data Member name**

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

7**.2.2 VALUE CONVENTIONS**

Value conventions ensure values for variable at any point of time. This involves the following:

* Proper default values for the variables.
* Proper validation of values in the field.
* Proper documentation of flag values.

**7.2.3 SCRIPT WRITING AND COMMENTING STANDARD**

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

**7.2.4 MESSAGE BOX FORMAT**

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

* X – User has performed illegal operation.
* ! – Information to the user.

**7.3 TEST PROCEDURE**

SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

* Static analysis is used to investigate the structural properties of the Source code.
* Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

**7.4 TEST DATA AND OUTPUT**

**7.4.1 UNIT TESTING**

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

**7.4.2 FUNCTIONAL TESTS**

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

* Performance Test
* Stress Test
* Structure Test

**7.4.3 PERFORMANCE TEST**

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

**7.4.4 STRESS TEST**

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

**7.4.5 STRUCTURED TEST**

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

* Exercise all logical decisions on their true or false sides.
* Execute all loops at their boundaries and within their operational bounds.
* Exercise internal data structures to assure their validity.
* Checking attributes for their correctness.
* Handling end of file condition, I/O errors, buffer problems and textual errors in output information

**7.4.6 INTEGRATION TESTING**

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

**7.5 TESTING TECHNIQUES / TESTING STRATERGIES**

**7.5.1 TESTING**

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

**7.5.1.1 WHITE BOX TESTING**

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

* Flow graph notation
* Cyclometric complexity
* Deriving test cases
* Graph matrices Control

**7.5.1.2 BLACK BOX TESTING**

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

* Graph based testing methods
* Equivalence partitioning
* Boundary value analysis
* Comparison testing

**7.5.2 SOFTWARE TESTING STRATEGIES:**

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

* Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
* Different testing techniques are appropriate at different points in time.
* The developer of the software and an independent test group conducts testing.
* Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

**7.5.2.1 INTEGRATION TESTING:**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

**7.5.2.2 PROGRAM TESTING:**

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

**7.5.2.3 SECURITY TESTING:**

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

**7.5.2.4 VALIDATION TESTING**

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

\* The function or performance characteristics confirm to specifications and are accepted.

\* A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

**7.5.2.5 USER ACCEPTANCE TESTING**

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

* Input screen design.
* Output screen design.

**Source Code**

**Screenshots:**

**REFERENCES**

[1] N. Kabra, P. Bhattacharya, S. Tanwar, and S. Tyagi, “Mudrachain Blockchain-based framework for automated cheque clearance in financial institutions,” Future Generation Computer Systems, vol. 102,

pp. 574–587, 2020.

[2] “Equifax data breach settlement.” <https://www.ftc.gov/enforcement/> cases-proceedings/refunds/equifax-data-breach-settlement. Accessed: 2019-12-27.

[3] “Experian lapse allowed id theft service access to 200 million consumer records.” <https://krebsonsecurity.com/2014/03/experian-lapseallowed-> id-theft-service-to-access-200-million-consumer-records/. Accessed: 2019-12-27.

[4] P. Bhattacharya, S. Tanwar, U. Bodke, S. Tyagi, and N. Kumar, “Bindaas: Blockchain-based deep-learning as-a-service in healthcare 4.0 applications,” IEEE Transactions on Network Science and Engineering, pp. 1–1, 2019.

[5] H. Wang, C. Guo, and S. Cheng, “Loc — a new financial loan management system based on smart contracts,” Future Generation Computer Systems, vol. 100, pp. 648 – 655, 2019.

[6] S. Chakraborty, S. Aich, S. J. Seong, and H. Kim, “A blockchai based credit analysis framework for efficient financial systems,” in 201 21st International Conference on Advanced Communication Technology (ICACT), PyeongChang, Korea (s), pp. 56–60, Feb 2019.

[7] V. Thakur, M. Doja, Y. K. Dwivedi, T. Ahmad, and G. Khadanga, “Land records on blockchain for implementation of land titling in india,” International Journal of Information Management, p. 101940, 2019. V. Hassija, G. Bansal, V. Chamola, N. Kumar, and M. Guizani, “Secure lending: Blockchain and prospect theory-based decentralized credit scoring model,” IEEE Transactions on Network Science and Engineering, vol. PP, 03 2020.

[9] T. Yuan, H. Wu, J. Zhu, L. Shen, and G. Qian, “Ms-ucf: A reliable recommendation method based on mood-sensitivity identification and user credit,” in 2018 International Conference on Information Management and Processing (ICIMP), London, UK, pp. 16–20, Jan 2018.

[10] K. Ren and A. Malik, “Recommendation engine for lower interest borrowing on peer to peer lending (p2pl) platform,” in 2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI), Thessaloniki Greece, pp. 265–269, Oct 2019.

[11] N. Morar, C. Baber, F. McCabe, S. D. Starke, I. Skarbovsky, A. Artikis, and I. Correai, “Drilling into dashboards: Responding to computer

recommendation in fraud analysis,” IEEE Transactions on Human- Machine Systems, vol. 49, pp. 633–641, Dec 2019.

[12] S. Cai and J. Zhang, “Exploration of credit risk of p2p platform based on data mining technology,” Journal of Computational and Applied Mathematics, vol. 372, p. 112718, 2020.

[13] P. Pławiak, M. Abdar, J. Pławiak, V. Makarenkov, and U. R. Acharya, “Dghnl: A new deep genetic hierarchical network of learners for prediction of credit scoring,” Information Sciences, vol. 516, pp. 401 – 418, 2020.

[14] U. Bodkhe, P. Bhattacharya, S. Tanwar, S. Tyagi, N. Kumar, and M. S. Obaidat, “Blohost: Blockchain enabled smart tourism and hospitality management,” in 2019 International Conference on Computer, Information and Telecommunication Systems (CITS), China, pp. 1–5, Aug 2019.

[15] A. Srivastava, Bhattacharya, A. Singh, A. Mathur, O. Prakash, and R. Pradhan, “A distributed credit transfer educational framework based on blockchain,” in 2018 Second International Conference on Advances in Computing, Control and Communication Technology (IAC3T), Allahabad, India, pp. 54–59, IEEE, 2018.

[16] D. Dua and C. Graff, “UCI machine learning repository,” 2017.

[17] S. Guo, H. He, and X. Huang, “A multi-stage self-adaptive classifier ensemble model with application in credit scoring,” IEEE Access, vol. 7, pp. 78549–78559, 2019.

[18] Y. Liu, A. Ghandar, and G. Theodoropoulos, “A metaheuristic strategy for feature selection problems: Application to credit risk evaluation in emerging markets,” in 2019 IEEE Conference on Computational Intelligence for Financial Engineering Economics (CIFEr), Shenzhen, China, pp. 1–7, May 2019.

[19] B. Zhu, W. Yang, H. Wang, and Y. Yuan, “A hybrid deep learning model for consumer credit scoring,” in 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), Las Vegas, USA, pp. 205– 208, May 2018.

[20] “Mythril security analysis framework on evm bytecodes to detect security vunerabilities of smart contracts.” <https://github.com/ConsenSys/> mythril. Accessed: 2020-01-26.

[21] M. D. Angelo and G. Salzer, “A survey of tools for analyzing ethereum smart contracts,” 2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON), pp. 69–78, 2019.

[22] J. Srinivas, A. K. Das, N. Kumar, and J. Rodrigues, “Cloud centric authentication for wearable healthcare monitoring system,” IEEE Transactions on Dependable and Secure Computing, 2018.

[23] V. Odelu, A. K. Das, M. Wazid, and M. Conti, “Provably secure authenticated key agreement scheme for smart grid,” IEEE Transactions on Smart Grid, vol. 9, no. 3, pp. 1900–1910, 2018.