PAYMENT ENGINE

Submitted in partial fulfilment of the requirements for the

award of

Bachelor of Engineering Degree in Computer Science and Engineering

by

PALLAVI RAJ (37110537)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC I 12B

Status by UGC I Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI - 600119

MARCH 2021

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEM
ED TO BE UNIVERSITY)
Accredited with Grade

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **PALLAVI RAJ (37110537)** who carried out the project entitled **"PAYMENT ENGINE"** under our supervision from **November 2020** to **March 2021**.

Internal Guide

Dr. L. Lakshmanan, M.E.,Ph.d.,

Head of the Department

Dr. S. Vigneshwari, M.E., Ph.D.,

Dr. L. Lakshmanan, M.E., Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner **External Examiner**

DECLARATION

I **PALLAVI RAJ** hereby declare that the Project Report entitled **"PAYMENT ENGINE"** done by me under the guidance of **Dr. L. Lakshmanan, M.E.,Ph.d.,** is submitted in partial fulfillment of the requirements for the award of Bachelor ofComputer Science and Engineering.

DATE: 06-04-2021

PLACE: **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D .,** Dean, School of Computing and **Dr. L. Lakshmanan M.E., Ph.D.,** and **Dr. S. Vigneshwari M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. L. Lakshmanan, M.E.,Ph.d., Head of the Department of Computer Science and Engineering** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of **COMPUTER SCIENCE AND ENGINEERING** who were helpful in many ways for the completion of the project.

fi

fi

ri

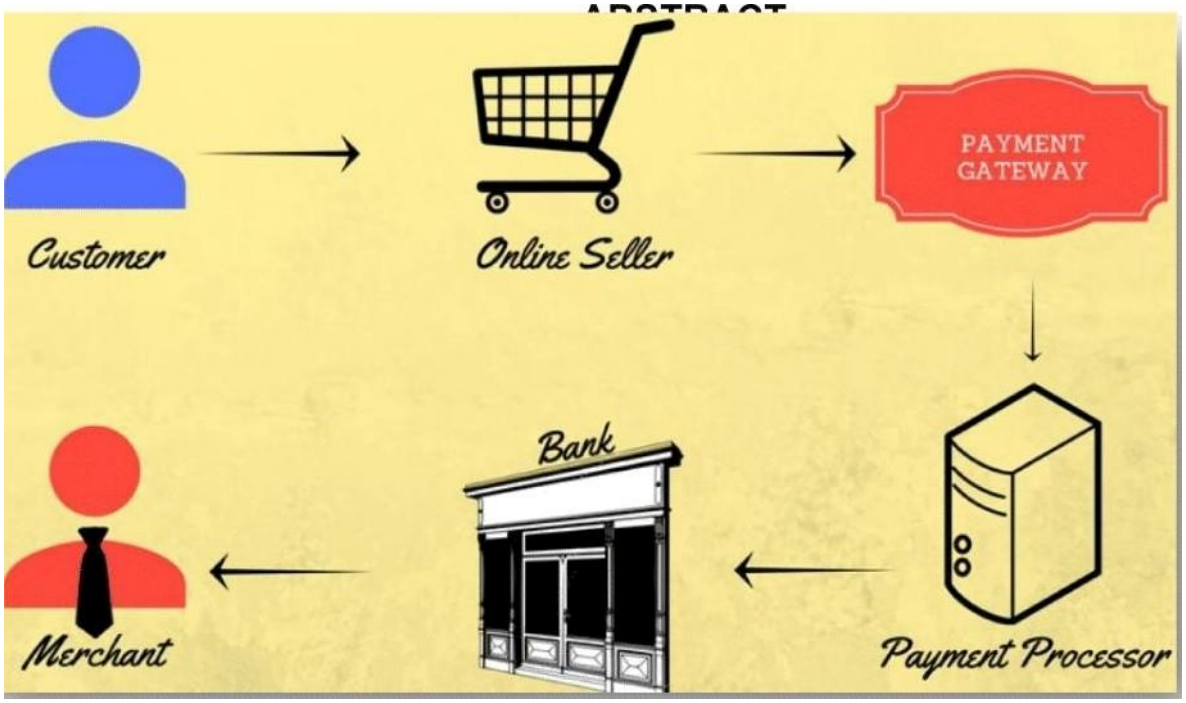

Customer
Online Seller
PAYMENT
GATEWAY
Bank
Merchant
Payment Processor

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

With the introduction of online payment in 2010 as Immediate Payment Services (IMPS) as instant, 24X7 interbank electronic fund transfer using mobile phones. With the rise of online payment, many payment gateways came into market with different features in it.

Since the dawn of history, there has been trading between two parties exchanging goods face-to-face. Eventually such trading became complicated and inconvenient; money was invented so that a buyer could acquire something he needed from a seller without necessarily exchanging goods. Security of the monetary systems was guaranteed by the local, regional, national, and eventually international banks controlling the printing of money. In course of time, new ways of payment such as payment orders, cheques, and later 'plastic' money were invented. These allow payment without 'actual' money. Mapping between the payment instrument and real money is still guaranteed by banks through secure financial clearing networks. Eventually, remote payment became possible using those same instruments, although security then started to become a challenge. Verifying a hand-written signature on a cheque or a credit card mail order is impossible when buyer and seller are not face-to- face; in this case the buyer must take the additional risk of sending in a payment before having received his purchase or the seller must send the purchased items before having received the payment. Phone order purchases are riskier since no signature at all can be provided: the seller runs the risk that the buyer may deny having made the purchase and demand a refund even after he has received the goods. Recently, there has been a great deal of interest in facilitating commercial transactions over open computer networks, such as the Internet. The introduction of open networks renders the security issues even more critical.

Due to the COVID-19 pandemic the world has to turn to digitalization. Payment methods has been changed tremendously. In the beginning, people use cash to pay. Then cards came in use and now it's all online. Many online payment engines or gateways are used to pay digitally. Each bank have their own credit or debit cards

which can be used for online transactions. In India, G-pay, Phone Pay and Paytm are the most used payment gateways. Mobile payment has become a trend all over the world and constitutes an increasingly substantial portion of payments.

Online payment system is a vast term, which shows various options for delivery through multiple electronic channel. It is used as an accurate characteristics of an online payment in literature. Electronic banking, electronic cash, internet banking, online banking etc all comes under online payments or can be different aspects of payments which can be performed online.

The revolution of IT industry in banking sector especially in payment systems has provided opportunities to banks to improve and bring new innovative services to the bank customers including reduction in operating costs. E-payment easily handles large volume of transactions in faster and efficient manner with minimum manual intervention. But still, in India a significantly large number of transactions are cash- based. The e-payment systems have made rapid strides in the last few years and the physical cheque based payments are growing at a lesser space. Introduction of technology in banks improved their functioning which include:

1) Low cost of operations.
2) Lesser burden for customers
3) Improvement in customer services
4) Efficiency in funds management
5) Better administration.

All things considered, recent researchers have demonstrated a few endeavours to come up with a definition of online payment. Online payment processes are ignored by traditional textbook, but it is important for students to understand online payment because it represents the most important part in payment in the last twenty decades.

The motive behind this project was to understand the working of any payment engine and payment gateways. During lockdown, online payment has increased its height and now became most popular among people for payment. People have faith that their money is secured. This faith is turned into curiosity to know about the system behind it.

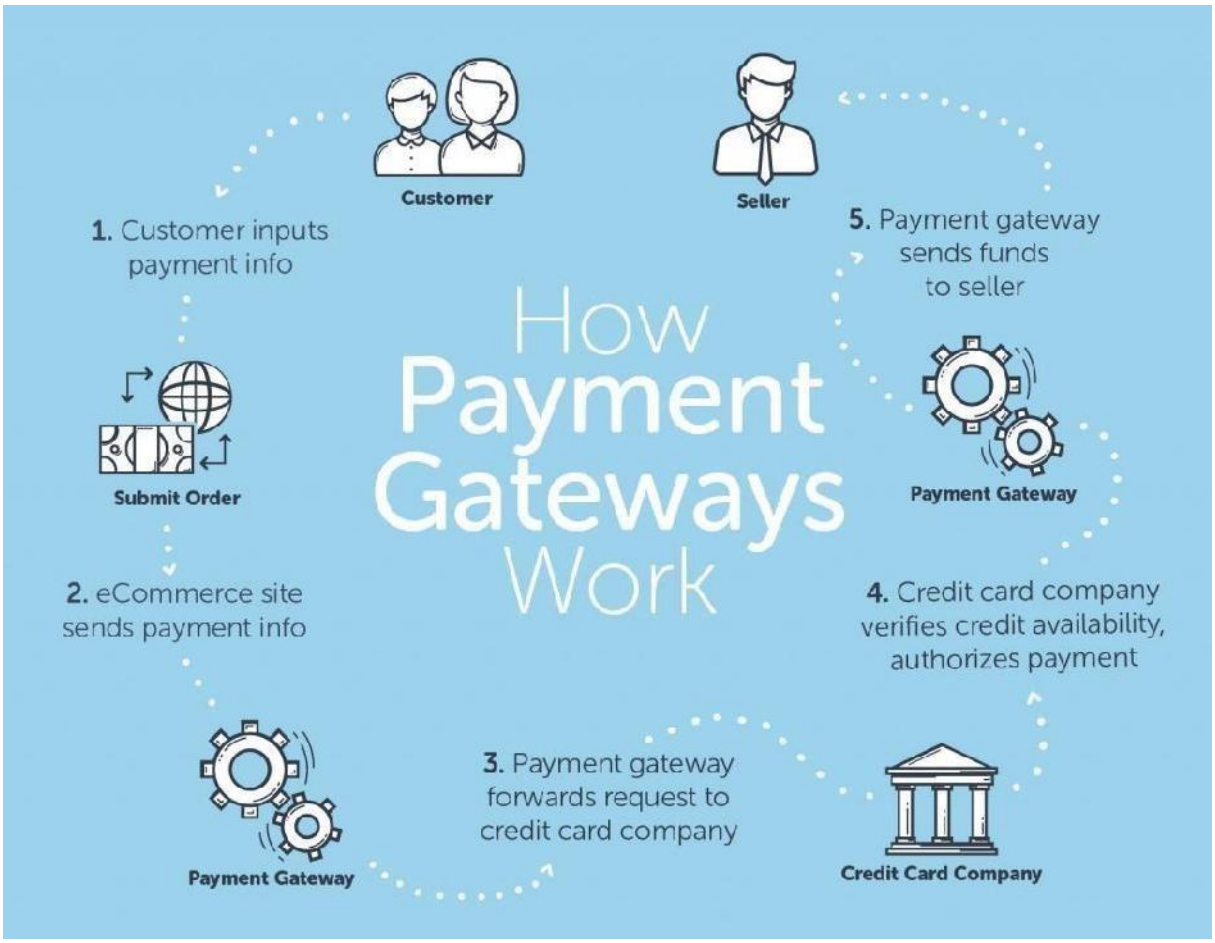


Fig.1.1 : Working of Payment Gateway

1.1.1 Role of a Payment Gateway

The main role of an online Payment Gateway is to approve the transaction process between merchant and customer. It plays a vital role in the online transaction process and authorizes transactions between merchants and customers.

It helps the e-commerce platform aggravate its existence with ease of payments to offer to its customers. Besides, it also leads to the e-commerce platform gaining rapport for leading to not only quick and secure payments but also convenience and success with the same every time.

A payment gateway service can be provided by banks directly or a payment service provider authorized by a bank.

1.2 PROBLEM STATEMENT

A payment gateway is a merchant service provided by an e-commerce application service provider that authorizes credit card or direct payments processing for e-

businesses, online retailers, bricks and clicks, or traditional brick and mortar.[1] The payment gateway may be provided by a bank to its customers, but can be provided by a specialised financial service provider as a separate service, such as a payment service provider.

A payment gateway facilitates a payment transaction by the transfer of information between a payment portal (such as a website, mobile phone or interactive voice response service) and the front end processor or acquiring bank.

Building a payment engine application that can help the users to understand the transaction process. This will help the students also to understand the new technology behind the online transactions. It will show how any fund is getting transferred to another account with different steps. This will help us to know about the payment methods used by various payment gateways for online payment.

1.3 LITERATURE REVIEW

Burhan Ul Islam Khan et al proposed a study on online systems: past developments, present impact and future considerations which is aimed at the present status and growth of online payment. The paper gives a deep survey on the electronic payment by analysing various different papers. It also shows the details about the several payment services and tells us about the security reasons too.

In a conference, Emir Husni et al, an online payment system using SMS Gateway and line application was posed for rural areas. This will help the people in rural areas for the online payments just by using SMS. It will not only help urban areas but majorly focuses on rural areas in any country. This research shows that people can use online payment system even without internet connection just by the SMS or line API. They can purchase, do online transaction, bank transfer etc.

Yunhao Xia et al proposed a third party payment system in which they try to solve the reason behind the security of any online payment system. They came up with a signature scheme based on number theory research unit (NTRU). Their project works efficiently by using this algorithm and can raise the speed and probability of generating signature value.

Wenzheng Liu et al proposed another security related to online payment systems. It gives a review about a secure mobile payment. They have divided payments into TPC- led mobile payment and bank-led mobile payment and on basis of this they made a structure of mobile payment. They gave a detailed view about the security and compared different mobile applications.

Through these Literature Surveys, it was made possible to understand how any payment system works and how it gets implemented. It also helps us to understand that security is the major concern in online payment systems.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM OF THE PROJECT

The aim of this project is to understand the working of a payment engine and develop a demo payment engine. Payment engine is designed to show all the transactions status.

Before analysing a topic it is necessary to have common definitions of its vocabulary. This chapter helps aligning previous assumptions.

2.2 OBJECTIVES

- To build a payment engine.
- To get the track of the transactions made.
- To be used in real life if it satisfies the requirements of the industries.

2.3 SCOPE

This model will help in building the strategies to understand the transaction flow and the technology behind the payment engine. This model can help students also to understand/study about the new technology. Future is with digitalization, so we need to upgrade ourselves and be ready to conquer it.

2.4 ADVANTAGES

- Understanding of payment system.
- Transaction flow which runs in the back-end.
- Transaction Services.
- Front-end is designed to show the status of transaction for the users.

2.5 What is a “Payment Engine”?

A Payment Engine is a standalone system that enables the connection of multiple internal and external channels. This facility allows the extension of existing applications to interface with additional functionality as required. For example, one can connect complementary, standalone services such as; fraud prevention and anti money laundering services, or a report writing capability. Alternatively, an external repository could be used for the storage of operational files and audit trails to support separate enquiry or management information systems. A Payment Engine is used to accept, validate, authenticate, sort, direct, clear and settle payment transactions received from a diverse range of originating systems and deliver these payments, in the required format, on the appropriate date or time, to an equally diverse range of destination systems. A Payment Engine is required to manage the financial accounting and reconciliation of payments received and dispatched. Audit Trails, archival records, and operational reports are produced to support system operations and administration. A Payment Engine provides a high level of straight through processing capability, real time and batch processing, and 24 x 7 availability. Value added and real time services may also be provided independently of any other systems included in a particular configuration. For example, high value transactions may be identified by the processing rules of the Payment Engine and consequently referred to a separate anti money laundering capability for validation purposes. Subject to the response code returned to the Payment Engine, the transaction will continue to be processed as required. A Payment Engine usually utilises transparent, flexible and automated workflows to execute predefined processes applicable to particular payment types. These workflows, which incorporate a diverse range of originator/recipient determined rules, are pre-configured by expert users. Transactions processed by a Payment Engine are reconfigured and distributed utilising the format, frequency and communication protocol prescribed by a particular recipient. Such transmissions may be proprietary, or may comply with one or other industry standard. Payment Engine operations are administered by a range of highly disciplined control routines. Because of the intermediary role performed between other participating systems, a very high level of importance is placed on the ability to reconcile inputs to outputs and vice versa. Reconciliation records and audit trails are maintained to enable the reconstruction of

any function or process retrospectively, at the lowest level required. Enquiries received can be resolved quickly and conclusively.

Although cash has provided the major form of payment in the past, cash substitutes such as Bank drafts, cheques, credit transfers and debit orders have emerged to displace actual currency. With the advent of modern technology and electronic communication, an even larger number of alternative electronic payment methods have been created. These include credit cards, debit cards, credit orders, debit orders, direct debits, direct credits, internet banking payments and e-commerce payments. Standardisation has allowed for some alternative payment services to grow nationally and even globally. For example, SWIFT, the credit card industry and the ATM industry. Alternatively, many Country, Bank and product specific payment services continue to exist which are unable to interact or communicate with each other at either an intra- bank or inter-bank level. Many of these bespoke systems have been enhanced with complex and sometimes ingenious developments that enable interoperability with other systems and services. Unfortunately, these “solutions” are normally tenuous and do not readily lend themselves to changed circumstances or needs. Consequently, many of these “brilliant solutions” sometimes cause “insoluble problems”.

CHAPTER 3

SYSTEM DESIGN & METHODOLOGY

3.1 EXISTING SYSTEM

Technology adoption and usage literature have recognized the influence of various factors. Environments pose a threat of risk and uncertainty, and that is when contextual factors play a significant role in the behaviours of individuals. There are a very different set of studies which are not particularly focussed on mobile payments as an information technology, but on its growth in India post demonetization. Demonetization and the inevitable transformation from 'physical cash to digital cash' has been anecdotally examined in most of the works around the period of the shock. Digital payments referred to any types of payments using digital instruments, which include mobile payment, mobile wallets, cryptocurrency, and electronic payment.

Mobile devices now serve to pay for goods and services by means of the transmission of data, a system known as mobile payments. Mobile payment is receiving growing attention globally, from consumers to merchants, as an alternative to using cash, check, or credit cards. Most encryption techniques applied in mobile payment are based on traditional public key infrastructure. However, the traditional public key encryption algorithm has higher requirements for hardware, which is not suitable for mobile terminals of limited computing resources.

The current payment engine is like a secret to world. At present many people don't know how any payment engine and payment gateway actually works. They are just using the application. It is more important for any student with IT background to know the working and technology behind any new trending application or technology

So, I am trying to learn something new from the existing system through this project.

Drawbacks

- Unwanted payment fail.
- Not having friendly UI.
- No details about any back-end process.

3.2 PROPOSED SYSTEM

In my project, I have developed a payment engine by using JAVA, HTML and CSS. IntelliJ IDEA is the IDLE used for the project. Google Chrome is used for display. Hyper-Text Markup Language (HTML) is the standard language of the Internet and is used for Front-End Web Development while Cascading Style Sheet (CSS) is used for styling the Web Pages and to give it a proper User-Interface. Java is an object oriented programming language which is widely used as back end development projects. In addition, IntelliJ IDEA is an integrated environment for Java. Google Chrome is a browser used for web part.

Advantages

- Understanding of payment system.
- Front-end is designed to show the status of transaction for the users.

3.3 REQUIREMENT SPECIFICATION

This proposed software runs effectively on a computing system that has the minimum requirements. Undertaking all the equipment necessities are not satisfied but rather exist in their systems administration between the customer's machines already. So the main need is to introduce appropriate equipment for the product.

The requirements are split into two categories, namely:

Software Requirements

The basic software requirements to run the program are:

Software	**IntelliJ**
Language	**JAVA**
Database	**SQL**
Front-End	**HTML, CSS**
Browser	**Chrome**

Hardware Requirements

The basic hardware required to run the program are:

System	**DELL Core i3**
Hard Disk	**256 GB**
Monitor	**15” LED**
Input Device	**Keyboard, Mouse**
RAM	**8GB**

3.4 SOFTWARE DESCRIPTION

3.4.1 JAVA

Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general- purpose programming language intended to let application developers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but

has fewer low-level facilities than either of them. The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages. As of 2019, Java was one of the most popular programming languages in use according to GitHub, particularly for client-server web applications, with a reported 9 million developers.

Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems' Java platform. The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun had relicensed most of its Java technologies under the GNU General Public License. Oracle offers its own HotSpot Java Virtual Machine, however the official reference implementation is the OpenJDK JVM which is free open source software and used by most developers and is the default JVM for almost all Linux distributions.

3.4.2 SPRING MAVEN

The **Spring Framework** is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE (Enterprise Edition) platform. Although the framework does not impose any specific programming model, it has become popular in the Java community as an addition to the Enterprise JavaBeans (EJB) model. The Spring Framework is open source.

Java programs are complex and feature many heavyweight components. Heavyweight means the components are dependent on the underlying operating system (OS) for their appearance and properties.

Spring is considered to be a secure, low-cost and flexible framework. Spring improves coding efficiency and reduces overall application development time because it is lightweight -- efficient at utilizing system resources -- and has a lot of support.

Spring removes tedious configuration work so that developers can focus on writing business logic. Spring handles the infrastructure so developers can focus on the application.

A web application (layered architecture) commonly includes three layers:

1. Presentation/view layer (UI) - This is the outermost layer which handles the presentation of content and interaction with the user.
2. Business logic layer - The central layer that deals with the logic of a program.
3. Data access layer - The deep layer that deals with data retrieval from sources.

Each layer is dependent on the other for an application to work. In other words, the presentation layer talks to the business logic layer, which talks to the data access layer. Dependency is what each layer needs to perform its function. A typical application has thousands of classes and many dependencies.

Without a Spring Framework, application code tends to be tightly coupled (interdependent), which is not considered good coding practice. Loose coupling is ideal because loosely coupled components are independent, meaning changes in one will not affect the operation of others.

Spring's core logic is dependency injection. Dependency injection is a programming pattern that allows developers to build more decoupled architectures. Dependency injection means that Spring understands the different Java annotations that a developer puts on top of classes. Spring knows that the developer wants to create an instance of a class and that Spring should manage it. Spring also understands the dependency and makes sure that all instances created have properly populated dependencies.

For the Spring Framework to instantiate objects and populate the dependencies, a programmer simply tells Spring which objects to manage and what the dependencies are for each class. A developer does so by using annotations like:

@component - Lets Spring know which classes to manage (create). Marks the beans (objects) as managed components, which means that Spring will autodetect these classes for dependency injection.

@autowired - Tells Spring how to handle the instantiation of the class (so it starts looking for that dependency among components/classes to find a match). This spares developers from wiring with code and allows Spring to find what needs to be injected where.

The first version was written by Rod Johnson, who released the framework with the publication of his book *Expert One-on-One J2EE Design and Development* in October 2002. The framework was first released under the Apache 2.0 license in June 2003. The first production release, 1.0, was released in March 2004. The Spring 1.2.6 framework won a Jolt productivity award and a JAX Innovation Award in 2006. Spring
2.0 was released in October 2006, Spring 2.5 in November 2007, Spring 3.0 in December 2009, Spring 3.1 in December 2011, and Spring 3.2.5 in November 2013. Spring Framework 4.0 was released in December 2013. Notable improvements in Spring 4.0 included support for Java SE (Standard Edition) 8, Groovy 2, some aspects of Java EE 7, and WebSocket.

Spring Framework 4.2.0 was released on 31 July 2015 and was immediately upgraded to version 4.2.1, which was released on 01 Sept 2015. It is *"compatible with Java 6, 7 and 8, with a focus on core refinements and modern web capabilities"*.

Spring Framework 4.3 has been released on 10 June 2016 and will be supported until 2020. It *"will be the final generation within the general Spring 4 system requirements (Java 6+, Servlet 2.5+), [...]"*.

Spring 5 is announced to be built upon Reactive Streams compatible Reactor Core.

MAVEN

Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project.

Maven addresses two aspects of building software: how software is built, and its dependencies. Unlike earlier tools like Apache Ant, it uses conventions for the build procedure, and only exceptions need to be written down. An XML file describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plug-ins. It comes with pre- defined targets for performing certain well-defined tasks such as compilation of code and its packaging. Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories such as the Maven 2 Central Repository, and stores them in a local cache. This local cache of downloaded artifacts can also be updated with artifacts created by local projects. Public repositories can also be updated.

Maven is built using a plugin-based architecture that allows it to make use of any application controllable through standard input. A C/C++ native plugin is maintained for Maven 2.

Alternative technologies like Gradle and sbt as build tools do not rely on XML, but keep the key concepts Maven introduced. With Apache Ivy, a dedicated dependency manager was developed as well that also supports Maven repositories.

Apache Maven has support for reproducible builds.

3.4.3 IntelliJ

IntelliJ IDEA is an integrated development environment (IDE) written in Java for developing computer software. It is developed by JetBrains (formerly known as IntelliJ), and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition. Both can be used for commercial development.

The first version of IntelliJ IDEA was released in January 2001, and was one of the first available Java IDEs with advanced code navigation and code refactoring capabilities integrated.

In a 2010 *InfoWorld* report, IntelliJ received the highest test center score out of the four top Java programming tools: Eclipse, IntelliJ IDEA, NetBeans and JDeveloper.

In December 2014, Google announced version 1.0 of Android Studio, an open- source IDE for Android apps, based on the open source community edition of IntelliJ IDEA. Other development environments based on IntelliJ's framework include AppCode, CLion, DataGrip, GoLand, PhpStorm, PyCharm, Rider, RubyMine, WebStorm, and MPS.

3.4.4 JDBC

Java Database Connectivity (**JDBC**) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database, and is oriented toward relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the Java virtual machine (JVM) host environment.

Sun Microsystems released JDBC as part of Java Development Kit (JDK) 1.1 on February 19, 1997. Since then it has been part of the Java Platform, Standard Edition (Java SE).

The JDBC classes are contained in the Java package java.s and javax.sql .

Starting with version 3.1, JDBC has been developed under the Java Community Process. JSR 54 specifies JDBC 3.0 (included in J2SE 1.4), JSR 114 specifies the JDBC Rowset additions, and JSR 221 is the specification of JDBC 4.0 (included in Java SE 6).

JDBC 4.1, is specified by a maintenance release 1 of JSR 221 and is included in Java SE 7.

JDBC 4.2, is specified by a maintenance release 2 of JSR 221 and is included in Java SE 8.

The latest version, JDBC 4.3, is specified by a maintenance release 3 of JSR 221and is included in Java SE 9.

JDBC ('Java Database Connectivity') allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT.

3.4.5 CHROME (as browser for UI)

Google Chrome is a cross-platform web browser developed by Google. It was first released in 2008 for Microsoft Windows, and was later ported to Linux, macOS, iOS, and Android where it is the default browser built into the OS. The browser is also the main component of Chrome OS, where it serves as the platform for web applications.

Most of Chrome's source code comes from Google's free and open-source software project *Chromium*, but Chrome is licensed as proprietary freeware. WebKit was the original rendering engine, but Google eventually forked it to create the Blink engine; all Chrome variants except iOS now use Blink.

As of November 2020, StatCounter estimates that Chrome has a 70% worldwide browser market share (after peaking at 72.38% in November 2018) on personal computers (PC), and 66.12% across all platforms. Because of this success, Google has expanded the "Chrome" brand name to other products: Chrome OS, Chromecast, Chromebook, Chromebit, Chromebox, and Chromebase.

CHAPTER 4

SOFTWARE DEVELOPMENT METHODOLOGY

4.1 DESCRIPTION OF DIAGRAM

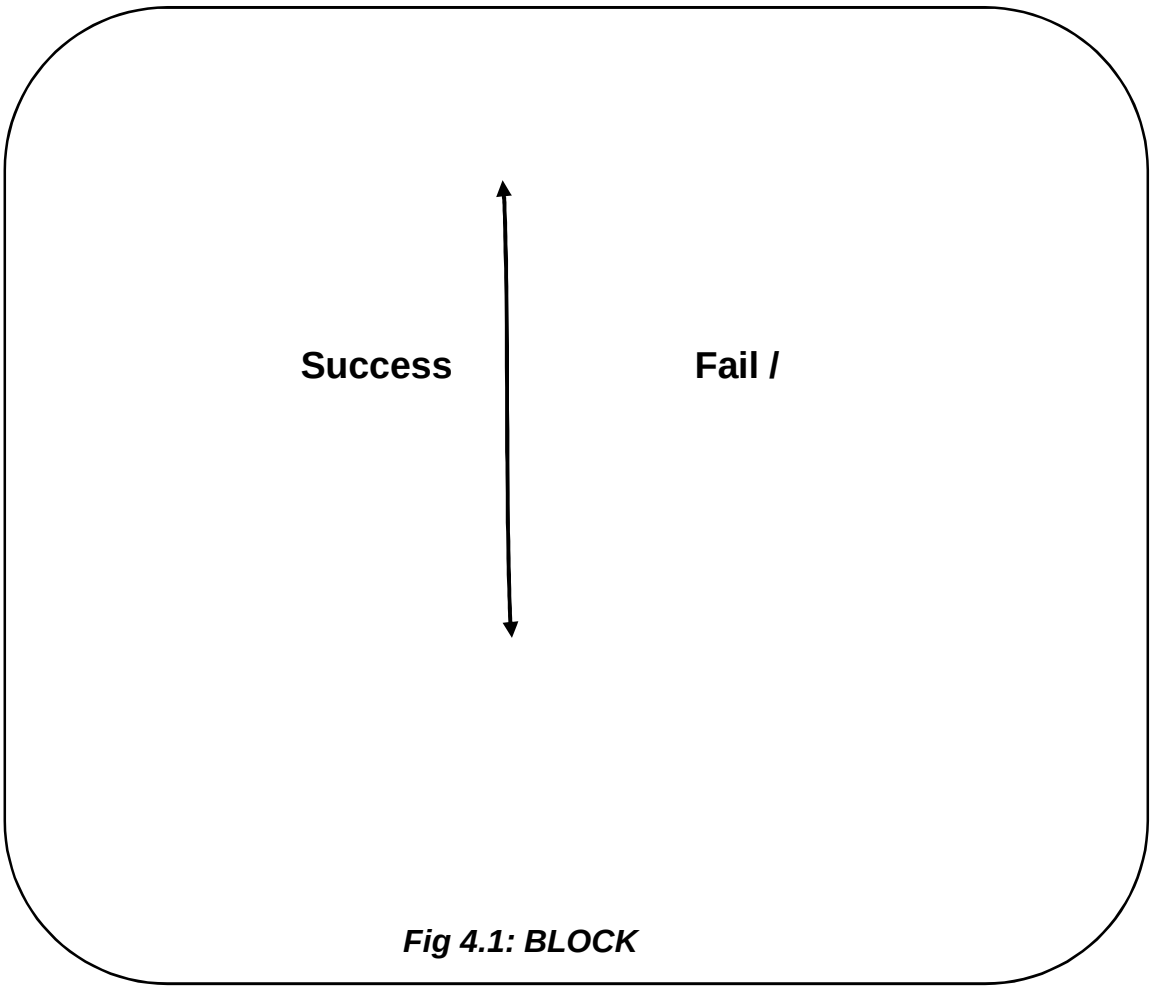


Fig 4.1: BLOCK

In fig 4.1 the block diagram represents the rough presentation of the system designed. The arrows represents the flow of the process generated from the end user side to the payment engine. The grey box presents as the user interface or end user side. The yellow box represents the whole payment engine. Within the system, we will observe that the user will request a transaction and the payment engine will go through different steps to process the transaction. The transaction will fail or gets pending if any issue occurred. It will get successful if it will pass all the steps in the payment engine. The inputs to the proposed system are given in the XML format. User interface is used as

output screen of the proposed system. The transaction details will be displayed on the user end.

4.2 ARCHITECTURAL DESIGN

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components. It is an important tool as it provides an overall view of the physical deployment of the software system and its evolution roadmap.

It consists of :

- Pre-processing the data
- Aggregation
- Transaction
- Sanction
- Debit and Credit

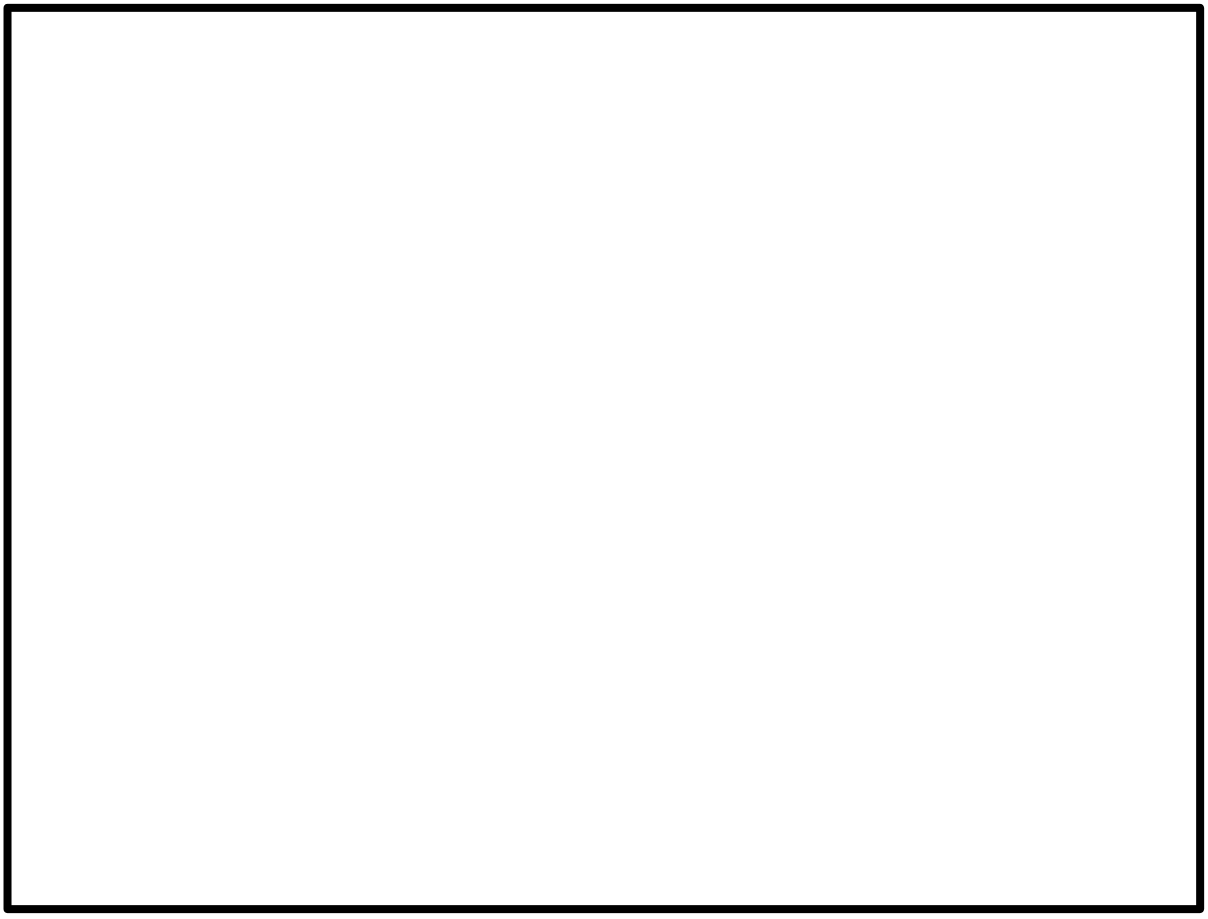


Fig 4.2: Activity Diagram

4.3 IDENTIFYING THE ACTORS

The actors within the system are the user, payment engine and thus the system will access the transaction request and implements the steps. The user can request for any kind of transaction it can be for sending it to overseas, within local area or any place possible. The payment engine plays the major role for processing the transaction and delivering back to the user.

4.4 USE CASES

In the payment engine application system, users are any person who wants do any type of transaction. The proposed system has different users, different roles and different use cases for instance of identifying the utilization cases, which allows us to take the transactions (users input to the system).

4.5 SYSTEM IMPLEMENTATION

The modules included in this system are:

4.5.1 Transaction (XML file)

- XML is the file format used to take the transaction details.
- This file will be used by the user to give the transaction details to the system.
- This file will be pre-processed further.
- All payments will be converted into xml file.
- Using these payment details verification will be done from the database about the client.
- Otherwise reject the payment and will be referred .

4.5.2 Payment Aggregation

- If payment is done internationally, then the currency should be changed in this process.
- Prepare the transaction.
- Validate each payment.
- Reject in case of any issues.

4.5.3 Payment Sanction

- Check the KYC for the client for the transaction proceedings.
- If the KYC is done, proceed for the debit.

4.5.4 Payment Debit

- After sanction and aggregation approval debit the payment from the client's account.

4.5.5 Payment Credit

- After the debit is done, credit will done to the receiver's account.

CHAPTER 5

RESULTS AND DISCUSSION

Fig 5.1: XML file

Fig 5.2: Transaction is completed when status is Credit End

Fig 5.3: Transaction details above

Fig 5.4: If transaction have any error it will be referred

Fig 5.5: All transaction details can be seen on the front end above

Fig 5.6: All transaction details with error can be seen on the front end above

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 CONCLUSION

In my approach, I have designed a payment engine which can state the transaction status and display it to the user. This can help to give the detailed status of any transaction. Users can understand the reason and take decision as per the transaction fault. Nowadays users cannot find the reason behind the transaction fail or pending. This application can help users for this and helps the student to understand this new technology working and their features.

In conclusion, electronic transfer funds have been around for many years and the economy has greatly benefited from this technological advance. An electronic payment system such as credit cards has facilitated monetary transactions and even provides a way to finance everyday purchases through credit. Because of this, bitcoins are gaining popularity but there are still many questions and considerations of a virtual economy. However, the risk of identity thefts, market euphoria, and privacy issues will always exists. As history has showed us, new technology can cause irrational exuberance that only leads to overvalued securities and ultimately end in a financial collapse. Nonetheless, new financial technology is not yet perfected and can be very costly. But with new innovations and proper usage, financial technology can be the key to successfully managing one's money.

6.2 FUTURE ENHANCEMENT

- UI can be modified and can add more features.
- Security measures can be added.

REFERENCES

[1] Abhipsa Pal “A review of contextual factors affecting mobile payment adoption and use” Springer, 2019.

[2] Burhan Ul Islam Khan “A Compendious Study of Online Payment Systems: Past Developments, Present Impact, and Future Considerations” IJACSA, Vol. 8, No. 5, 2017.

[3] Emir Husni “E-Payment System Using SMS Gateway and Line Application” IEEE Conference Publication, 2018.

[4] Francisco Li`ebana-Cabanillas “ Assessment of mobile technology use in the emerging market: Analyzing intention to use m-payment services in India” Elsevier, 2020.

[5] Oussama Tounekti “Users Supporting Multiple (Mobile) Electronic Payment Systems in Online Purchases:An Empirical Study of Their Payment
Transaction Preferences” IEEE, vol. 8, 2020.

[6] S Fatonah “A Review of E-Payment System in E-Commerce” IOP Publishing, 2020.

[7] Venkatasamy Sureshkumar “A lightweight two-gateway based payment protocol ensuring accountability and unlinkable anonymity with dynamic identity” Elsevier, 2016.

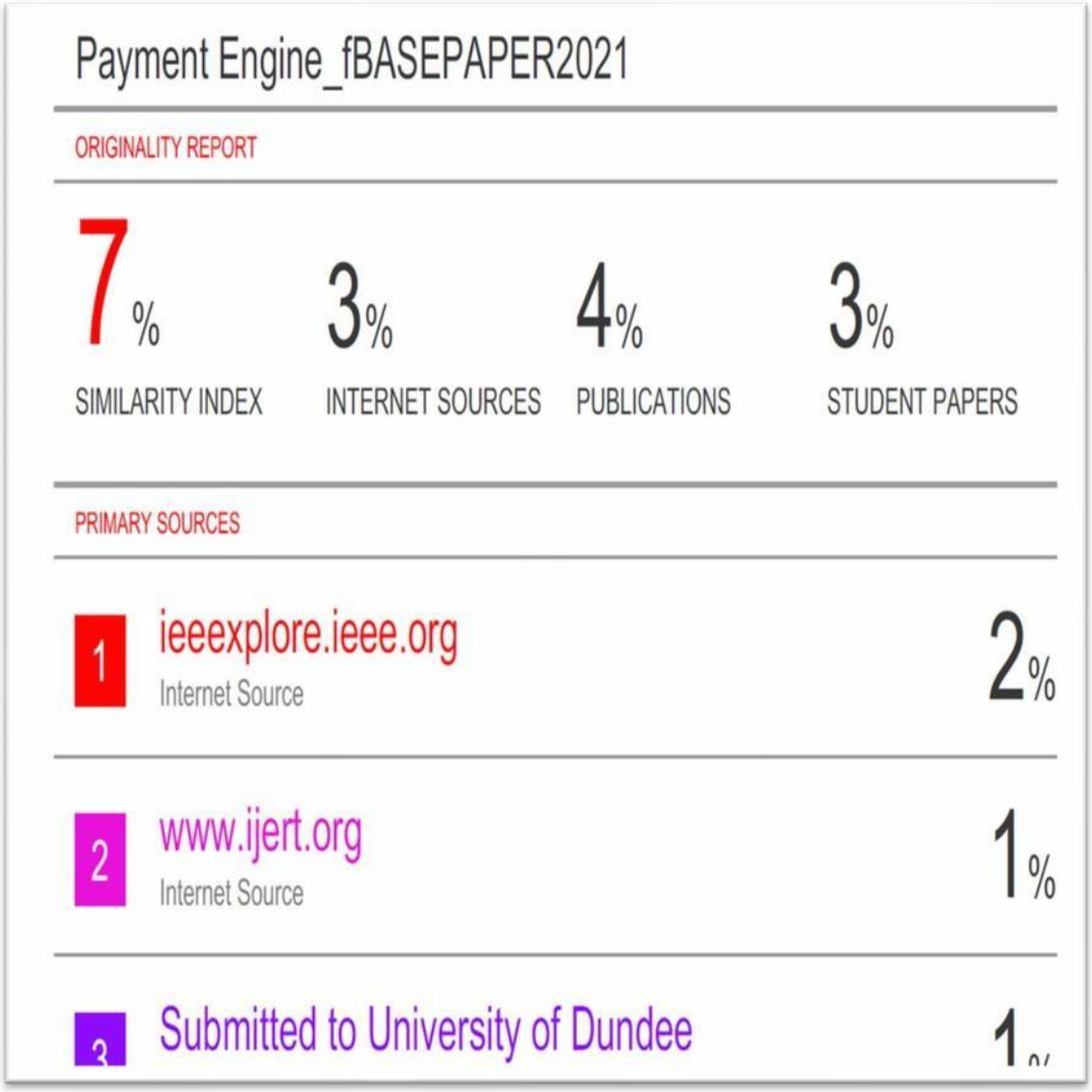
[8] Wassan Abdullah “Digital payment and banking adoption research in Gulf countries: A systematic literature review” Elsevier, 2020.

[9] Wenzheng Liu “State of the Art: Secure Mobile Payment” IEEE Publication, vol. 8, 2020.

[10] Yunhao Xia “A Third-Party Mobile Payment Scheme Based on NTRU Against Quantum Attacks” IEEE Publication, vol. 7, 2019

APPENDIX

A. PLAGIARISM REPORT



Payment Engine_fBASEPAPER2021

ORIGINALITY REPORT

7%	3%	4%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ieeexplore.ieee.org Internet Source	2%
2	www.ijert.org Internet Source	1%
3	Submitted to University of Dundee	1%

B. JOURNAL PAPER

Importance and rise of Payment Engine during the pandemic COVID-19

Pallavi Raj
Department of Computer Science and Engineering, Sathyabama institute of science and technology, Chennai, India
pallavi9598@gmail.com

Dr. L Lakshmanan
Department of Computer Science and Engineering, Sathyabama Institute of science and technology, Chennai, India
lakshmanan.cse@sathyabama.ac.in

Abstract –

Payment Engine provides standardized processes for payment transactions and flexible rules for all payment activities to be always ready for rapidly changing customer demands. Many e-commerce websites will use online payments and it's time for digitalization. Nowadays, online payments are replacing cash. So, online payment systems are grabbing attention of everyone in the world. The alternative to online payments are cash, check, and credit/debit cards. This paper proposes the working of a payment engine and understanding the transaction journey through front-end. In my project we have designed a State-of-the-Art payment engine which can help users to understand the transaction flow. Finally, this will be useful for the end users to get the reason behind their transactions fail or about their pending transactions.

Keywords: Payment Engine, online payments, digitalization

I. INTRODUCTION

With the introduction of online payment in 2010 as Immediate Payment Services (IMPS) as instant, 24X7 interbank electronic fund transfer using mobile phones. With the rise of online payment, many payment gateways came into market with different features in it.

Due to the COVID-19 pandemic the world has to turn to digitalization. Payment methods has been changed tremendously. In the beginning, people use cash to pay. Then cards came in use and now it's all online. Many online payment engines or gateways are used to pay digitally. Each bank have their own credit or debit cards which can be used for online transactions. In India, G-pay, Phone Pay and Paytm are the most used payment gateways[8]. Mobile payment has become a trend all over the world and constitutes an increasingly substantial portion of payments.

Online payment system is a vast term, which shows various options for delivery through multiple electronic channel. It is used as an accurate characteristics of an online payment in literature. Electronic banking, electronic cash, internet banking, online banking etc all comes under online payments or can be different aspects of payments which can be performed online.

All things considered, recent researchers have demonstrated a few endeavours to come up with a definition of online payment. Online payment processes are ignored by traditional textbook, but it is important for students to understand online payment because it represents the most important part in payment in the last twenty decades.

The motive behind this project was to understand the working of any payment engine and payment gateways. During lockdown, online payment has increased its height and now became most popular among people for payment. People have faith that their money is secured. This faith is turned into curiosity to know about the system behind it.

In msg paper. I progned n Pujnient Engine" o•hieh oJ -hon bori• aEJ' fund z getting trmferied to another zeeoimt math diJferezt step.. T will kelp in to hues about the paJmaent ñe&ods u.sed by various payment gateways for online payment.

II. LITERATURE SURVEY

Buu'han fifl Islam **Kbzn** 1] et z. proposed n study on online system:: post émm*opmeam , premnt impact and fuNre consideration which i: did ai the pre sent status and gloom of online payment. The pape r gii es a deep suit e* on the e'eerr onic pastfi I bx' zs*:ysxg x arim di£fer cut pays. Ii also skoivs the details about the cm<ral payment settees and Ie.1s us about the seewiPv ieasozis too.

In a conference, Enir HP ct oL an one payment system using SKIS Gates n* and .se zpplie ation o os posed for rural news. This red help the seople' ie iura. area s for the onbe s zJnneno ;u*t by tag FMS. It o 1: nor only help urban areas but major • foxes on niral mean in any eoimty . imezrcli fore- iJ:nt people em use on'ice ga¿aeet cyctezs e\ ea sYitbout uztet:aet coz+z+ectioc ji:rst by- be SSU or 1fize .&J. W¿• caa j wc6a:e. do ootfize dozsacaoo baAk tzza:fer etc.

Yunhno Xia [6] ct a. Mr oposed n d in pms payment s* stem in which thej' rrv to s oh e the read on behind the suntn- of znj' online paxnDent ystem . They came up Auth a : zrure scheme based .on number tbeow reseaieh imii TRi'3. lieu project o o rks e£fieient:y b* be oritbm met can roe the speed and probability of generating signature value.

Wenzheng Liu [5] et al proposed another security related to online payment systems. It gives a review about n sure mobi.e pajmext . They km e di.dded pajmects etc UPC -led mobile payment as:i i›znk- led mobile payment and on basis of this they made a structure of mobile payment. They gave a detailed view about the security and compared different mobile applications.

Through these Literature Surveys, it was made poscib.e to us:ierstan box- and pzJ :uent system o ords mcd kou- it eels top.emented. It aLo he:ps us to u£.dent and rhat securiD• is t1•e major concern ir online payment systems.

DE SISIElfIñIPLElIEñZ.\HO1

A. *Existing System*

The current payment engine is like a secret to world. At present many people don't know how any payment engine and payment gateway actually works. They are just using the application. It is more important for any student with IT background to know the working and technology behind any new trending application or technology[9].

5o, I m Mmgto lsaz somelâze ne bmotâe existing system through this project.

B. /'ruposz‹lSt-sfnrH

In mv pin{eei, I have dcx elooed n payment engine by be JAN'.? , llTifL met C US . Irites! JDK A fi the IDLE used for be project. Gooele Chrome is u red for Simply.

Hyper-Text Markup Language (HTML) is the ended iza ;e of the Internet and is eé for Front-End fi' eb Dec. elopnaent while 'Czscadmg Sw 'e Sheet (C'SSâ is md for ring the V'e-' Pages and to px< it a proher User-Inter-face. Jan-z u ac object orie:etefi: prcm ammicg lanruzee o Lick is w'der md as but end dmvlopment pin.'ects. In addition InieâiJ IDA is an inleerated enxnronineni for Jan-a. Google €'Ro£iz in a browser used for web parl.

The pic'ject zi ebitecNue bz: been articulated below:

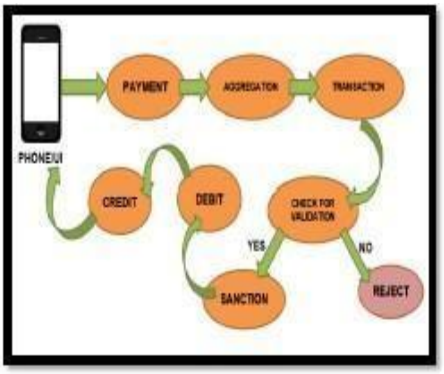


Fig. 1 System Architecture

C. Advantages of the proposed system

- Understanding of payment system.
- Front-end is designed to show the status of transaction for the users.

D. Implementation

The modules included in the project are:

- **XML File:**

  XML file is used to take the details of the transaction.

- **Payment Aggregation:**

  This method is used to prepare the transaction and validate it. Reject the transaction in case of any issue.

- **Process Sanction:**

  Check the KYC for the client for the transaction proceedings.

- **Process Debit:**

  After sanction and aggregation approval debit the payment from the client's account.

- **Process Credit:**

  After the debit is done, credit will done to the receiver's account.

IV. RESULT AND DISCUSSION

The payment engine was tested on IntelliJ for back-end and Google Chrome for front-end. A positive response was identified with the same. Once the code was executed, all the details from the XML file was converted and stored into their respective tables in database.

The code was tested with more than 50 types of transactions file so that we don't miss any aspects.

Some of the test images has been displayed below:

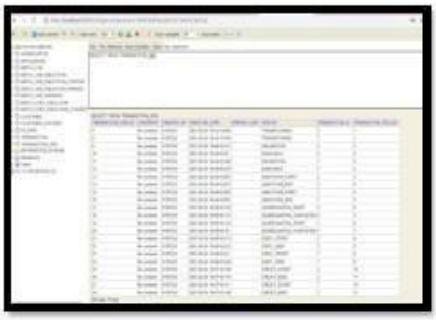


Fig. 2 Status of transactions

Fig.2 clearly shows the transaction status of the payments done.

Fig. 3 UI for Users

Fig.3 shows the transactions details on front-end to the end users.

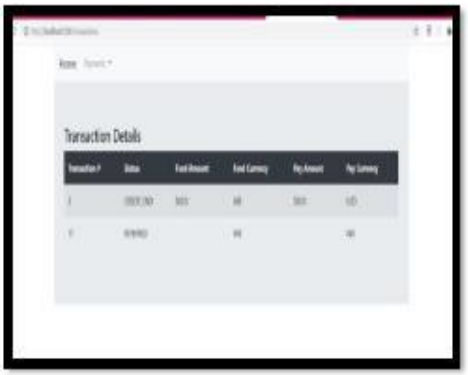


Fig. 4 UI showing failed transaction

Fig.4 shows the fault in the transaction to the end user to know the reason of transaction fail or pending.

V. CONCLUSION

In my approach, I have designed a payment engine which can state the transaction status and display it to the user. This can help to give the detailed status of any transaction. Users can understand the reason and take decision as per the transaction fault. Nowadays users cannot find the reason behind the transaction fail or pending. This application can help users for this and helps the student to understand this new technology working and their features.

In future we can add some more details on the front end for the user and update security.

VI. REFERENCES

[1] Abhipsa Pal "A review of contextual factors affecting mobile payment adoption and use" Springer, 2019.

[2] Burhan Ul Islam Khan "A Compendious Study of Online Payment Systems: Past Developments, Present Impact, and Future Considerations" IJACSA, Vol. 8, No. 5, 2017.

[3] Emir Husni "E-Payment System Using SMS Gateway and Line Application" IEEE Conference Publication, 2018.

[4] Francisco Li`ebana-Cabanillas " Assessment of mobile technology use in the emerging market: Analyzing intention to use m-payment services in India" Elsevier, 2020.

[5] Oussama Tounekti "Users Supporting Multiple (Mobile) Electronic Payment Systems in Online Purchases:An Empirical Study of Their Payment Transaction Preferences" IEEE, vol. 8, 2020.

[6] S Fatonah "A Review of E-Payment System in E-Commerce" IOP Publishing, 2020.

[7] Venkatasamy Sureshkumar "A lightweight two-gateway based payment protocol ensuring accountability and unlinkable anonymity with dynamic identity" Elsevier, 2016.

[8] Wassan Abdullah "Digital payment and banking adoption research in Gulf countries: A systematic literature review" Elsevier, 2020.

[9] Wenzheng Liu "State of the Art: Secure Mobile Payment" IEEE Publication, vol. 8, 2020.

[10] Yunhao Xia "A Third-Party Mobile Payment Scheme Based on NTRU Against Quantum Attacks" IEEE Publication, vol. 7, 2019.

C. SOURCE CODE

```
package com.pallavi.app.entity;

import javax.persistence.*;
import java.util.Date;

@Entity
public class Aggregation {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name="aggregation_id")
    private long aggregationId;

    @Column(name="batch_file_id")
    private String batchFileId;

    @Column(name="creation_date")
    private Date creationDate;

    @Column(name="created_by") private
    String createdBy;

    @Column(name="error_code") private
    String errorCode;

    @Column(name="status")
    private String status;

    @Column(name="fund_customer_id") private
    String fundCustomerId;

    @Column(name="pay_customer_id")
```

```
private String payCustomerId;

@Column(name="fund_currency")
private String fundCurrency;

@Column(name="pay_currency")
private String payCurrency;

@Column(name="fx_rate")
private Double fxRate;

@Column(name="fund_amount")
private Double fundAmount;

@Column(name="pay_amount")
private Double payAmount;

@Column(name="pay_value_date")
private String payValueDate;

@Column(name="sanction_status")
private String sanctionStatus;

@Column(name="debit_status")
private String debitStatus;

@Column(name="credit_status")
private String creditStatus;


public Aggregation() {
}

public Aggregation(long aggregationId, String batchFileId, Date creationDate,
```

```
String createdBy, String errorCode, String status, String fundCustomerId, String payCustomerId,
String fundCurrency, String payCurrency, Double fxRate, Double fundAmount, Double
payAmount, String payValueDate, String sanctionStatus, String debitStatus, String creditStatus) {
        this.aggregationId = aggregationId;
        this.batchFileId = batchFileId;
        this.creationDate = creationDate;
        this.createdBy = createdBy;
        this.errorCode = errorCode; this.status
        = status;
        this.fundCustomerId = fundCustomerId;
        this.payCustomerId = payCustomerId;
        this.fundCurrency = fundCurrency;
        this.payCurrency = payCurrency;
        this.fxRate = fxRate;
        this.fundAmount = fundAmount;
        this.payAmount = payAmount;
        this.payValueDate = payValueDate;
        this.sanctionStatus = sanctionStatus;
        this.debitStatus = debitStatus;
        this.creditStatus = creditStatus;
    }

    public long getAggregationId() {
        return aggregationId;
    }

    public void setAggregationId(long aggregationId) {
        this.aggregationId = aggregationId;
    }

    public String getBatchFileId() {
        return batchFileId;
    }
```

```
public void setBatchFileId(String batchFileId) {
    this.batchFileId = batchFileId;
}

public Date getCreationDate() {
    return creationDate;
}

public void setCreationDate(Date creationDate) {
    this.creationDate = creationDate;
}

public String getCreatedBy() {
    return createdBy;
}

public void setCreatedBy(String createdBy) {
    this.createdBy = createdBy;
}

public String getErrorCode() {
    return errorCode;
}

public void setErrorCode(String errorCode) {
    this.errorCode = errorCode;
}

public String getStatus() {
    return status;
}

public void setStatus(String status) {
```

```
    this.status = status;
}

public String getFundCustomerId() {
    return fundCustomerId;
}

public void setFundCustomerId(String fundCustomerId) {
    this.fundCustomerId = fundCustomerId;
}

public String getPayCustomerId() {
    return payCustomerId;
}

public void setPayCustomerId(String payCustomerId) {
    this.payCustomerId = payCustomerId;
}

public String getFundCurrency() {
    return fundCurrency;
}

public void setFundCurrency(String fundCurrency) {
    this.fundCurrency = fundCurrency;
}

public String getPayCurrency() {
    return payCurrency;
}

public void setPayCurrency(String payCurrency) {
    this.payCurrency = payCurrency;
}
```

```
public Double getFxRate() {
    return fxRate;
}

public void setFxRate(Double fxRate) {
    this.fxRate = fxRate;
}

public Double getFundAmount() {
    return fundAmount;
}

public void setFundAmount(Double fundAmount) {
    this.fundAmount = fundAmount;
}

public Double getPayAmount() {
    return payAmount;
}

public void setPayAmount(Double payAmount) {
    this.payAmount = payAmount;
}

public String getPayValueDate() {
    return payValueDate;
}

public void setPayValueDate(String payValueDate) {
    this.payValueDate = payValueDate;
}

public String getSanctionStatus() {
```

```
    return sanctionStatus;
}

public void setSanctionStatus(String sanctionStatus) {
    this.sanctionStatus = sanctionStatus;
}

public String getDebitStatus() {
    return debitStatus;
}

public void setDebitStatus(String debitStatus) {
    this.debitStatus = debitStatus;
}

public String getCreditStatus() {
    return creditStatus;
}

public void setCreditStatus(String creditStatus) {
    this.creditStatus = creditStatus;
}


@Override
public String toString() {
    return "Aggregation{" +
            "aggregationId='" + aggregationId + '\'' + ",
            batchFileId='" + batchFileId + '\'' +
            ", creationDate=" + creationDate + ",
            createdBy='" + createdBy + '\'' + ",
            errorCode='" + errorCode + '\'' + ",
            status='" + status + '\'' +
            ", fundCustomerId='" + fundCustomerId + '\'' +
```

```
                ", payCustomerId='" + payCustomerId + '\'' + ",
                fundCurrency='" + fundCurrency + '\'' +
                ", payCurrency='" + payCurrency + '\'' + ",
                fxRate=" + fxRate +
                ", fundAmount=" + fundAmount +
                ", payAmount=" + payAmount +
                ", payValueDate=" + payValueDate +
                ", sanctionStatus='" + sanctionStatus + '\'' + ",
                debitStatus='" + debitStatus + '\'' +
                ", creditStatus='" + creditStatus + '\'' + '}';
    }
}
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Set;
import java.util.stream.Collectors;
import java.util.stream.Stream;

@Service
public class FileReceiverServiceImpl implements FileReceiverService {

    private static String FOLDER_PATH = "C:\\apps"; private
    static String INBOUND_PATH = "\\inbound\\"; private static
    String OUTBOUND_PATH = "\\outbound\\"; private static
    String FAILED_PATH = "\\failed\\";

    @Autowired
    BatchFileService batchFileService;
```

```
@Autowired
TransactionService transactionService;

@Override
public void readFile(){
    System.out.println("Starting readFile()         " + System.currentTimeMillis());

    try {
        //Read all files from inbound folder and return only filename Set<String>
        fileNames = listFiles(FOLDER_PATH + INBOUND_PATH);

        if (fileNames != null && fileNames.size() > 0) { System.out.print("fileNames
            size " + fileNames.size());
            //Iterate all file name and process the files for
            (String inputFileName : fileNames) {
                processFile(inputFileName);
            }
        } else {
            System.out.println("No Files available at " + System.currentTimeMillis());
        }
    } catch (Exception e){
        System.out.println("Exception in readFile() " + e);
    }
    System.out.println("End readFile()         " + System.currentTimeMillis());
}

//This method is used to get the name of files under directory
private Set<String> listFiles(String dir) {
    Set<String> fileNames = null;
    System.out.println("Inside listFiles() "+dir);
    try{
        fileNames = Stream.of(new File(dir).listFiles())
                .filter(file -> !file.isDirectory())
```

```
                .map(File::getName)
                .collect(Collectors.toSet());
        }catch (Exception e){ System.out.println("Exception
            in listFiles "+e);
        }
        return fileNames;
    }
    private BatchFileDO populateBatchFileDO(BatchFileXmlDO batchFileXmlDO){
        //Batch File DO populate
        BatchFileDO batchFile = new BatchFileDO();
        batchFile.setBatchNumber(batchFileXmlDO.getBatchNumber());
        batchFile.setBatchFileName("ABC");

        //transaction DO populate
        TransactionDO transaction = null;
        for(TransactionXmlDO transactionXmlDO : batchFileXmlDO.getTransactions())
{
            transaction = new TransactionDO();

            transaction.setFundAmount(transactionXmlDO.getDebtor().getAmount());
            transaction.setFundCurrency(transactionXmlDO.getDebtor().getCurrency());
            if(transactionXmlDO.getDebtor().getCustomerId() != null
                && !transactionXmlDO.getDebtor().getCustomerId().isEmpty()){

transaction.setFundCustomerId(transactionXmlDO.getDebtor().getCustomerId());
            }

            transaction.setPayAmount(transactionXmlDO.getCreditor().getAmount());
            transaction.setPayCurrency(transactionXmlDO.getCreditor().getCurrency());
            if(transactionXmlDO.getCreditor().getCustomerId() != null
                && !transactionXmlDO.getCreditor().getCustomerId().isEmpty()){

transaction.setPayCustomerId(transactionXmlDO.getCreditor().getCustomerId());
            }
```

```
            batchFile.getTransactions().add(transaction);
        }
        return batchFile;
    }
            System.out.println("File moved to "+target.toString());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    //This method will be call as scheduler; need to create scheduler public
    void readFileScheduler(){
        String cronExpression = "0 0/5 * 1/1 * ? *"; // Call for Every 5 mins

        //readFile();
    }


}
public class AggregationScheduler implements Tasklet {

    AggregationService aggregationService;

    public AggregationScheduler(AggregationService aggregationService){ this.aggregationService
        = aggregationService;
    }

    public RepeatStatus execute(StepContribution contribution, ChunkContext
chunkContext) throws Exception
    {
        System.out.println("AggregationScheduler start.."); try
        {
```

```
            aggregationService.processAggregation();
        } catch (Exception e){
            System.out.println("Exception while AggregationScheduler "+ e);
        }
        System.out.println("AggregationScheduler done..");
        return RepeatStatus.FINISHED;
    }
import java.util.stream.Collectors;

@Service
public class TransactionServiceImpl implements TransactionService {
    @Autowired
    AggregationService aggregationService;

    @Autowired
    TransactionRepository transactionRepository;

    @Autowired
    TransactionVerService transactionVerService;

    @Override
    public void createTransaction(BatchFileDO batchFileDO){
        System.out.println("Starting createTransaction()          ");
        List<Aggregation> aggregationDOs = new ArrayList<Aggregation>();

        for(TransactionDO transactionDO : batchFileDO.getTransactions()) {
            Transaction transaction = populateTransactionEntity(transactionDO);
            transaction.setBatchFileId(batchFileDO.getBatchFileId());

            this.populateAggregation(transaction, batchFileDO); Transaction

            transSaved = this.saveTransaction(transaction);
```

```
transactionVerService.saveTransactionVer(populateTransactionVerEntity(transSave d));
        }

        System.out.println("End createTransaction()          ");
    }

        private Transaction populateTransactionEntity(TransactionDO transactionDO){
        Transaction transaction = new Transaction(); transaction.setTransactionId(10011);
        transaction.setAggregationId("20011");
        transaction.setCreatedBy("Pallavi"); transaction.setCreationDate(new
        Date()); transaction.setErrorCode("");
        transaction.setFundAmount(transactionDO.getFundAmount());
        transaction.setFundCurrency(transactionDO.getFundCurrency());
        transaction.setPayAmount(transactionDO.getPayAmount());
        transaction.setPayCurrency(transactionDO.getPayCurrency());
        transaction.setStatus(Constants.STATUS_VER_TRANSFORMED);

        return transaction;
    }

    private TransactionVer populateTransactionVerEntity(Transaction transactionDO){

        //Batch File DO populate for Entity/Table TransactionVer
        transactionVer = new TransactionVer();
        transactionVer.setTransactionVerId(20011);
        transactionVer.setTransactionId(""+transactionDO.getTransactionId());
        transactionVer.setTransactionVerNo(1);
        transactionVer.setCreatedBy("SYSTEM");
        transactionVer.setCreationDate(new Date()); transactionVer.setErrorCode("");
```

```
        transactionVer.setStatus(Constants.STATUS_VER_TRANSFORMED);
        transactionVer.setContents("file contents");

        return transactionVer;
    }

    private void populateAggregation(Transaction transaction,BatchFileDO batchFileDO){
        System.out.println("populateAggregation :: Size
"+batchFileDO.getAggregationDOs().size());
        List<Aggregation> aggregationList = batchFileDO.getAggregationDOs();

        Aggregation aggregationDO = this.getExistingAggregation(transaction,
aggregationList);
        if(aggregationDO != null){
            //populate same transaction.setAggregationId(""+aggregationDO.getAggregationId());

            //update aggragation with amount
            aggregationDO.setFundAmount(transaction.getFundAmount().doubleValue()
+ aggregationDO.getFundAmount().doubleValue());
            aggregationDO.setPayAmount(transaction.getPayAmount().doubleValue() +
aggregationDO.getPayAmount().doubleValue());

            aggregationService.saveAggregation(aggregationDO);

        } else {
            SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("yyyyMMdd");
            //New Aggregation create
            Aggregation aggregation = new Aggregation();
            aggregation.setPayValueDate(simpleDateFormat.format(new Date()));
            aggregation.setSanctionStatus(Constants.NO); aggregation.setDebitStatus(Constants.NO);
```

```
            aggregation.setCreditStatus(Constants.NO);

aggregation.setFundCustomerId(batchFileDO.getTransactions().get(0).getFundCust omerId());

aggregation.setPayCustomerId(batchFileDO.getTransactions().get(0).getPayCustom erId());

            Aggregation agg = aggregationService.saveAggregation(aggregation);
            aggregationList.add(aggregation);
            batchFileDO.setAggregationDOs(aggregationList);

            transaction.setAggregationId(""+agg.getAggregationId());
            System.out.println("populateAggregation ::: Size
"+batchFileDO.getAggregationDOs().size());
        }

        System.out.println("End populateAggregation :::::::::::: Size
"+batchFileDO.getAggregationDOs().size());
    }

    private Aggregation getExistingAggregation(Transaction transactionDO,List<Aggregation>
aggregationDOs){
        System.out.println("getExistingAggregation :: FundCuurency
"+transactionDO.getFundCurrency()
                + " PayCurrency " + transactionDO.getPayCurrency() + " Size
"+aggregationDOs.size());
        for ( Aggregation aggregationDO : aggregationDOs){

if(transactionDO.getFundCurrency().equalsIgnoreCase(aggregationDO.getFundCurr ency())
                    &&
transactionDO.getPayCurrency().equalsIgnoreCase(aggregationDO.getPayCurrency ())
```

```
                //&&
transactionDO.getCustomerId().equalsIgnoreCase(aggregationDO.getCustomerId())
                ){
            System.out.println("Aggregation found");
            return aggregationDO;
        }
    }
    System.out.println("Aggregation NOT found");
    return null;
}

@Override
public Transaction saveTransaction (Transaction transaction){
    System.out.println("Starting saveTransaction()          ");
    Transaction trans = transactionRepository.save(transaction);

    System.out.println("End saveTransaction()          ");
    return trans;
}

@Override
public List<Transaction> allTransaction() {
    return (List<Transaction>) transactionRepository.findAll();
}

@Override
public Transaction getTransaction(long transactionId) {
    return transactionRepository.findById(transactionId).orElse(null);
}

@Override
public Transaction getTransactionByAggregationId(String aggregationId) { return
    allTransaction()
            .stream()
```

```
            .filter(txn -> (aggregationId.equalsIgnoreCase(txn.getAggregationId())))
            .collect(Collectors.toList())
            .stream()
            .findAny().orElse(null);
    }

    @Override
    public void validateTransaction(){
        System.out.println(" validate validateTransaction ");
        boolean isTransactionValid = false;

        List<Transaction> transactionList = allTransaction()
                .stream()
                .filter( transactions ->
Constants.STATUS_VER_TRANSFORMED.equalsIgnoreCase(transactions.getStat us()))
                .collect(Collectors.toList());

        if(transactionList != null && transactionList.size() >0) {
            System.out.println(" Transformed transactionList :: " + transactionList.size());
transactionVerService.captureTransactionVerEntry(""+txn.getTransactionId(),
Constants.STATUS_VER_VALIDATION);

                //Validate Transactions
                isTransactionValid = false;
                if(txn.getFundAmount() != null && txn.getFundAmount().doubleValue() > 0 &&
                        !txn.getFundCurrency().isEmpty()){
                    isTransactionValid = true;
                }
                txn.setStatus(isTransactionValid ? Constants.STATUS_VER_ENRICHED :
Constants.STATUS_VER_REFERRED);
                saveTransaction(txn);

                //Insert entry in Transaction_ver table with validation pass/fail
```

```
transactionVerService.captureTransactionVerEntry(""+txn.getTransactionId(), isTransactionValid
                        ? Constants.STATUS_VER_ENRICHED :
Constants.STATUS_VER_REFERRED);

                //Update Aggregation table
                if(isTransactionValid) {

aggregation.setStatus(Constants.STATUS_VER_SANCTION_PENDING);
                    aggregation.setSanctionStatus(Constants.STATUS_PENDING);
                    aggregationService.saveAggregation(aggregation);
                }

            }

        } else {
            System.out.println(" No transactions for validate");
        }

        System.out.println(" validate validateTransaction Completed !!!");
    }
}
```