LOAN ORIGINATION SYSTEM (HOME LOAN)

A PROJECT REPORT

Submitted By

KSHITIZ PANDEY
(2000290140060)

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Dr. Shashank Bhardwaj
Assistant Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(JUN 2022)

CERTIFICATE

Certified that **Kshitiz Pandey (2000290140060)** have carried out the project work having "**Loan Origination system (Home loan)**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Kshitiz Pandey (2000290140060)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Shashank Bhardwaj
Assistant Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Ajay Shrivastava
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

Loan Origination System is developed specially to support the loan application processing needs of banks and financial institutions. The process of issuing a loan involves a series of steps right from acquiring a borrower, to processing their information, to vetting their credit standing, documentation and then finally either approval and disbursal of funds or disapproving the loan.

Our loan origination system (LOS) can streamline all these processes offering a comprehensive solution to loan origination requirements while improving efficiency, mitigating risks and improving borrower relationships. Our Loan Origination Software provides banks complete front and back office processing functionality.

The solution is adaptable, agile and offers an enterprise class origination platform that is responsive to changing business needs in loans. It assists banks to meet service levels without conceding on cost and become more agile to be competitive, thus delivering significant return-on-investment. Financial institutions can deliver instant, automated decisions to applicants from any origination channel due to powerful decisioning, dynamic features, extensive third-party integrations and highly customizable applications.

Thus LOS equips you with all the tools required to ensure customer satisfaction – new competitive loan products, reduced processing time and faster disbursal while at the same time optimizing the bank's profit margins with improved efficiency, better credit analysis and reduced defects on loans.

ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Dr. Shashank Bhardwaj** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Kshitiz Pandey

Table of Contents

CERTIFICATE ii

ABSTRACT iii

ACKNOWLEDGEMENTS iv

Table of Contents v

Table of Figures vii

CHAPTER 1 INTRODUCTION 9

1.1 FUNCTIONALITIES PROVIDED 10

1.2 PURPOSE 10

1.3 MODULES 10

1.4 FEATURES 11

1.5 Main Roles 12

CHAPTER 2 SOFTWARE ANALYSIS 13

2.1 INTRODUCTION 13

2.2 IDENTIFICATION OF NEED 13

2.3 INPUT DATA AND VALIDATION 14

2.4 PRELIMINARY INVESTIGATION 16

2.5 FEASIBILITY STUDY 17

A. TECHNICAL FEASIBILITY 17

B. OPERATIONAL FEASIBILITY 18

C. ECONOMIC FEASIBILITY 19

CHAPTER 3 SOFTWARE REQUIREMENT SPECIFICATION 20

3.1 INTRODUCTION 20

3.2 DEVELOPERS RESPONSIBILITIES OVERVIEW 20

3.3 STAGES AND REQUIREMENTS OF THE PROPOSED SYSTEM 21

3.4 PERFORMANCE REQUIREMENTS 22

3.5 SOFTWARE REQUIREMENTS 22

3.6 HARDWARE REQUIREMENTS 23

CHAPTER 4 SOFTWARE DESIGN 24

4.1 INTRODUCTION 24

4.2 CLASS DIAGRAM 24

4.3 E-R DIAGRAM 29

4.4 ACTIVITY DIAGRAM 32

4.5 USECASE DIAGRAM 36

CHAPTER 5 CODING 39

5.1 Java Coding 39

CHAPTER 6 TESTING 70

6.1 LEVELS OF TESTING 70

6.2 STRATEGIC APPROACH TO SOFTWARE TESTING 71

..... 71

6.3 UNIT TESTING 72

6.4 INTEGRATION TESTING 72

6.5 VALIDATION TESTING 72

6.6 SYSTEM TESTING 73

6.7 WHITE BOX TESTING 73

6.8 BLACK BOX TESTING 74

6.9 OTHER TYPES OF TESTING 74

CHAPTER 7 IMPLEMENTATION AND MAINTENANCE 75

7.1 PROJECT SNAPSHOTS 75

7.2 MAINTENANCE 80

7.3 NEED FOR MAINTENANCE 81

7.4 TYPES OF MAINTENANCE 81

CHAPTER 8 CONCLUSION AND FUTURE WORK 83

8.1 CONCLUSION 83

8.2 FUTURE SCOPE 84

BIBLIOGRAPHY 86

RESEARCH PAPER (REFERENCES) 87

Table of Figures

Fig. 1 Class Diagram 29

Fig. 2 E-R Diagram 32

Fig. 3 Activity Diagram 35

Fig. 4 Use Case Diagram 38

Fig. 5 Testing Strategy 71

Fig. 6 Types of Maintenance 81

Chapter 1 INTRODUCTION

The main objective of developing this project is to handle the all details of Loans and Investments in the NBFC. The project has been developed to smoothen the processing of Loans in NBFC. Our proposed project automates the loan process from both, bankers as well as customer's side. Customer can apply for a loan and after approved it they can track their details from online. LOAN ORIGINATION SYSTEM is a very efficient process to handle all loan related transaction in a very accurate and convenient way. Loan origination system is an interface which facilitates a customer to apply for a loan online and to track the status from time-to-time. This system provides detail about the customers, their loan details, EMI details and its rate details. Getting a loan is a very tiring and complicated process in India. It may take weeks even months for loans to get approved and people have to visit the loan office again and again for document and verification. Using with this system admin can find customer details easily and it's a paperless system so workload is reduced. It is very helpful for those banking staffs who are in the charge of loan management, it provides a very reliable and convenient form for every loan and emi related transaction and their related details. Interest rates and the loan details are also available at the click of a mouse. Loan origination system is unique in such a way, it not only helps the customers but also the loan agency to check the pending, assign it to a departments, complete the formalities and procedures between the departments and arrive at decisions to very fact. System provides download option to download different type of loan form in MS word document. It also generates a very customer friendly and understandable form for their transaction information as the receive form after a transaction which contains all the information related to next emi date, remaining amount etc. After registration customer can use the system easily and also customer can view any query about loan details as well as EMI details in their profile. Most of the bank out-sources pre-loan process to loan agencies to reduce the burden and let the agencies pickup the information from customers and verify it before it is being forwarded to the actual bank for approval of loan. This system provides good communication for the customer and bank employee. This system provides a facility to generate the reports very easily.

1.1 FUNCTIONALITIES PROVIDED

- Provides the searching facilities based on various factors using advanced filters.
- Business development or pipeline development.
- Credit analysis and decisioning.
- Portfolio risk management.
- Allowance for loan and lease losses (ALLL) calculations or a fully compliance calculation under the current expected credit loss model, or CECL.
- Asset-liability management.
- The customer or member experience.
- Sound lending practices.
- Efficiency.

1.2 PURPOSE

- Applicant will submit his personal details through an third party application.
- Generating Loan Application Number (LAN) for each applicant's which is unique.
- Checking CIBIL score of applicant and coapplicant's.
- Making only eligible applicants to move further.
- Applicants will submit their Personal/Business KYC, Income documents, etc.
- Loan Origination.
- Define process and workflows.
- Case flow, management and tracking.
- Delivered on schedule within the budget.

1.3 MODULES

- **Studio**

Studio is used to manage all the details of loan and variables. It contains several other apps like -

- IDM App

- Task App
- Modeler App
- Admin App

- **Rendering Engine**

As the name suggests, this component is responsible for rendering a specific web page requested by the user on their screen. It interprets HTML and XML documents along with images that are styled or formatted using CSS, and a final layout is generated, which is displayed on the user interface.

- **Portal**

Portal is generally a back end part of our system. A page that is used as a point of entry to the customer, where information has been collected that will be useful to a applicant interested in home loan.

- **Integration Broker**

Also called an interface engine or a message broker, an IB is a third-party intermediary that facilitates interactions between applications. IBs minimally provide message transformation and routing services. They mostly communicate program to program; they integrate previously independent applications at the application-logic level of the software design.

- **Identity Access Management**

Identity and access management (IAM) is a framework of business processes, policies and technologies that facilitates the management of electronic or digital identities. With an IAM framework in place, information technology (IT) managers can control user access to critical information within their organizations. Systems used for IAM include single sign-on systems, two-factor authentication, multifactor authentication and privileged access management.

1.4 FEATURES

- Our product features.
- Client service.
- Client strength.

- Vendor Stability
- Support for customization.
- Ease of implementation and integration.
- A varieties of functionalities in the LOS.

1.5 Main Roles

- Admin
- Data entry
- Credit manager
- Technical Vendor
- Legal manager
- Legal Head
- Branch Manager

Chapter 2 SOFTWARE ANALYSIS

2.1 INTRODUCTION

After analysing the requirements of the task to be performed, the next step is to analyse the problem and understand its context. The first activity in the phase is studying the existing system and other is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system.

Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of existing running system is also difficult, improper understanding of present system can lead diversion from solution.

Software analysis is the process of totally understanding the current system by gathering and interpreting facts, diagnosing problems, and using the facts to improve the current system.it is conducted with the following objectives in mind:

- Identify the user's need;
- Evaluate the system concept for feasibility;
- Perform economic and technical feasibility;
- Allocate functions to hardware, software, people, database, and other system elements; and
- Create a system definition that forms the foundation for all subsequent Engineering works.

2.2 IDENTIFICATION OF NEED

The old manual system was suffering from a series of drawbacks. Since whole of the system was to be maintained with hands, the process of keeping, maintaining and retrieving the information was very tedious and lengthy. The records were never used to be in a systematic order. There used to be lots of difficulties in associating any particular transaction with a particular context. If any information was to be found, it was required to go through the different registers, documents. There would never exist anything like report generation. There would always be unnecessary consumption of time while entering records and retrieving records. One more problem was that it was very difficult to find

errors while entering the records. Once the records were entered it was very difficult to update these records.

The reason behind it is that there is lot of information to be maintained and have to be kept in mind while running the business. For this reason we have provided features present system which is partially automated (computerized). Actually existing system is quite laborious as one has to enter same information at three different places.

Following points should be well considered:

- Documents and reports that must be provided by the new system. There can also be few reports, which can help management in decision making and cost controlling. But since these reports do not get required attention, such kind of reports and information were also identified and given required attention.
- Details of the information needed for each document and report.
- The required frequency and distribution for each document.
- Probable sources of information for each document and report.
- With the implementation of computerized system, the task of keeping records in an organized manner will be solved. The greatest of all is the retrieval of information, which will be at the click of the mouse.
- The proposed system helps in saving the time in different operations and making information flow easy giving valuable reports.

2.3 INPUT DATA AND VALIDATION

- All the fields such as Name, Dob, co-applicants details, property details are validated and does not take invalid values.
- Each form for Data entry, Credit Manager, Technical vendor, cannot accept blank values fields.
- Avoiding errors in data.
- Controlling amount of input.
- Integration of all the modules in the system.
- Preparation of the test cases.
- Preparation of the possible test data with all the validation checks.
- Actual testing done manually.
- Recording of all the reproduced errors.

- Modifications done for the errors found during testing.
- Prepared the test result scripts after rectification of the errors.
- Functionality of the entire module.
- Validation for user input.
- Checking of the coding standards to be maintained during coding.
- Testing the module with all the possible test data.
- Testing of the functionality involving all type of calculations etc.
- Commenting standard in the source files.

2.4 PRELIMINARY INVESTIGATION

The Preliminary Investigation starts as soon as someone either a user or a member of a particular department recognizes a problem or initiates a request, to modify the current computerized system, or to computerize the current manual system. An important outcome of the preliminary investigation is determining whether the system requested is feasible or not.

The first step in the system development lifecycle is the preliminary investigation to determine the feasibility of the system. The purpose of the preliminary investigation is to evaluate project requests. It is not a design study nor does it include the collection of details to describe the business system in all respect. Rather, it is the collecting of information that helps committee members to evaluate the merits of the project request and make an informed judgement about the feasibility of the proposed project.

Analysts working on the preliminary investigation should accomplish the following objectives:

- Clarify and understand the project requests.
- Determine the size of the project.
- Assess cost and benefits of alternatives approaches.
- Determine the technical and operational feasibility of alternative approaches.
- Report the findings to management, with recommendations outlining the acceptance or rejection of the proposal.

BENEFIT TO ORGANIZATION

The organization will obviously be able to gain benefits such as savings in operating cost, reduction in paperwork, better utilization of human resources and more presentable image increasing goodwill.

THE INITIAL COST

The initial cost of setting up the system will include the cost of hardware, software (OS, add-on software, utilities) and labour (set-up and maintenance). The same has to bear by the organization.

RUNNING COST

Besides, the initial cost the long term cost will include the running cost for the system including the AMC, stationary charges, cost for human resources, cost for update/renewal of various related software.

2.5 FEASIBILITY STUDY

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational, and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they have unlimited resources and infinite time. Unfortunately, the development of computer based system is more likely to be plagued by a scarcity of resources and difficult delivery dates. It is both necessary and prudent to evaluate the feasibility of the project at the earliest possible time. Months or years of effort, money loss and untold professional embarrassment can be averted if we better understand the project at its study time. There are aspects in the feasibility study portion of the preliminary investigation.

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

A. TECHNICAL FEASIBILITY

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Do the necessary technology exist to do what is suggested?
- Do the proposed equipment have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?

- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow. Thus it provides an easy access to the user. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hardware requirements for the development of this project are not many and are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

B. OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system that will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to be test the operational feasibility of a project includes the following:

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

C. ECONOMIC FEASIBILITY

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as cost/benefit analysis.

The analyst raises various financial and economic questions during the preliminary investigation to estimate the following:

- The cost to conduct a full system investigation.
- The cost of hardware and software for the class of application being considered.
- The benefits in the form of reduced cost or fewer costly errors.
- The cost if the proposed system is not developed.

This project is economically feasible. It does not require any additional hardware or software. Since the interface for this system is developed using the existing resources and technologies available as free as open source, there is nominal expenditure and economic feasibility is certain.

Chapter 3 SOFTWARE REQUIREMENT SPECIFICATION

A **Software Requirement Specification (SRS)** – a requirements specification for a software system – is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional (supplementary) requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

3.1 INTRODUCTION

Purpose: The main purpose for preparing this document is to give a general insight into the analysis and requirements of the existing system or situation and for determining the operating characteristics of the system.

Scope: This document plays a vital role in Software Development Life Cycle (SDLC) and it describes the complete requirement of the system. It is meant for use by the developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

3.2 DEVELOPERS RESPONSIBILITIES OVERVIEW

The developer is responsible for:

- Developing the system which meets the SRS and solving all the requirements of the system.
- Demonstrating the system and installing the system at client's location after the acceptance testing is successful.
- Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.
- Conducting any user training that might be needed for using the system.
- Maintaining the system for a period of one year after installation.

3.3 STAGES AND REQUIREMENTS OF THE PROPOSED SYSTEM

- Pre-Qualification process like the potential borrower will receive a list of items they need to submit to the lender to get a loan.
- The borrower should complete the loan application.
- The application is received by the credit department and the first step done by the department is to review it for accuracy, genuine & Completeness. If all the required fields are not completed, the application will be returned to the borrower or the credit analyst and they will reach out the borrower to procure the missing information.
- When an application is totally completed, the underwriting process begins. Now Lender checks the application taking a variety of components into account: credit score, risk scores, and many lenders generate their own unique criteria for scoring that can be unique to their business or industry
- Depending on the results from the underwriting process, an application will be approved, denied or sent back to the originator for additional information.
- Since lending is highly regulated, the quality check stage of the loan origination process is critical to lenders. The application is sent to the quality control team, that analyze critical variables against internal and external rules and regulations. This is the last look at the application before it goes to funding.

- Most loans fund shortly after the loan documents are signed. Second mortgage loans, Business loans, Loan against property and lines of credit may require additional time for legal and compliance reasons. LOS can track funding and ensure that all necessary documents are executed before or together with funding.

3.4 PERFORMANCE REQUIREMENTS

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system.
- The system should be accurate.
- The system should be better than the existing system.

The existing system is completely dependent on the user to perform all the duties.

3.5 SOFTWARE REQUIREMENTS

Name of the component	**Specification**
Operating System	Windows 8 and Above
Technology	Flowable, Java, Springboot
Database	MySql
Tool	IntelliJ IDEA

3.6 HARDWARE REQUIREMENTS

Name of the component	Specification
Processor	Intel i3 or above
Hard Disk	100 GB Hard Disk Space
Ram	2 GB RAM or More

Chapter 4 SOFTWARE DESIGN

4.1 INTRODUCTION

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analysed, system design is the first of the three technical activities – design, code and test that is required to build any verified software.

The importance can be stated with a single word 'Quality'. Design is the place where quality is fostered in software development. Design provides us with representation of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed, reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

4.2 CLASS DIAGRAM

A class diagram models the static structure of a system. It shows relationships between classes, objects, attributes, and operations.
Basic notation and symbols:

- **Classes:**

Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.

Class Name
Attribute
Operation ()
Responsibility

Class

- **Active Classes:**

Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.

Class Name
Attribute
Operation ()

Active class

- **Visibility:**

Use visibility markers to signify who can access the information contained within a class. Private visibility, denoted with a - sign, hides information from anything outside the class partition. Public visibility, denoted with a + sign, allows all other classes to view the marked information. Protected visibility, denoted with a # sign, allows child classes to access information they inherited from a parent class.

Class Name
Attributes
+public operation -private operation #protected operation

Visibility

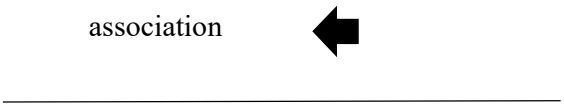
Marker	Visibility
+	Public
-	Private
#	Protected
~	Package

- **Association:**

Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.

Class Name
Attributes
Operations ()
Responsibility

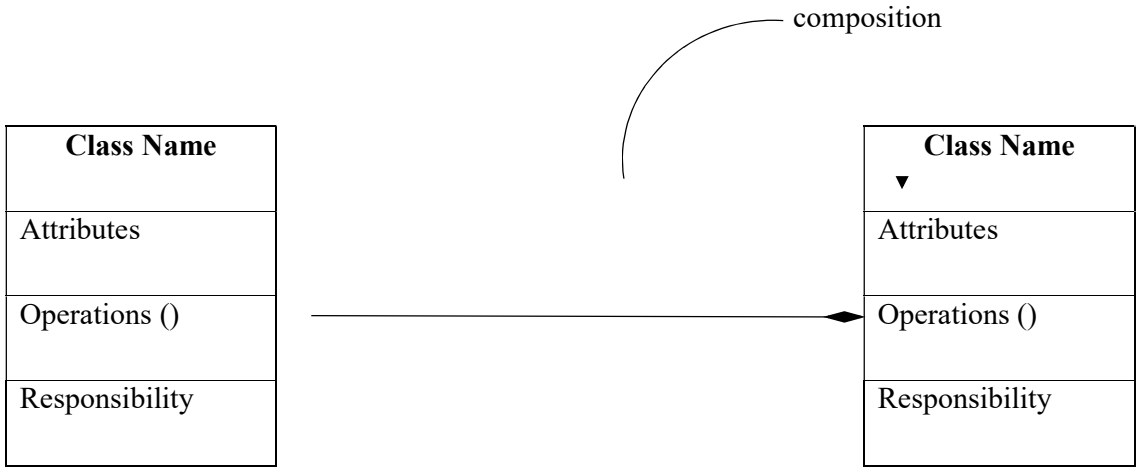
association

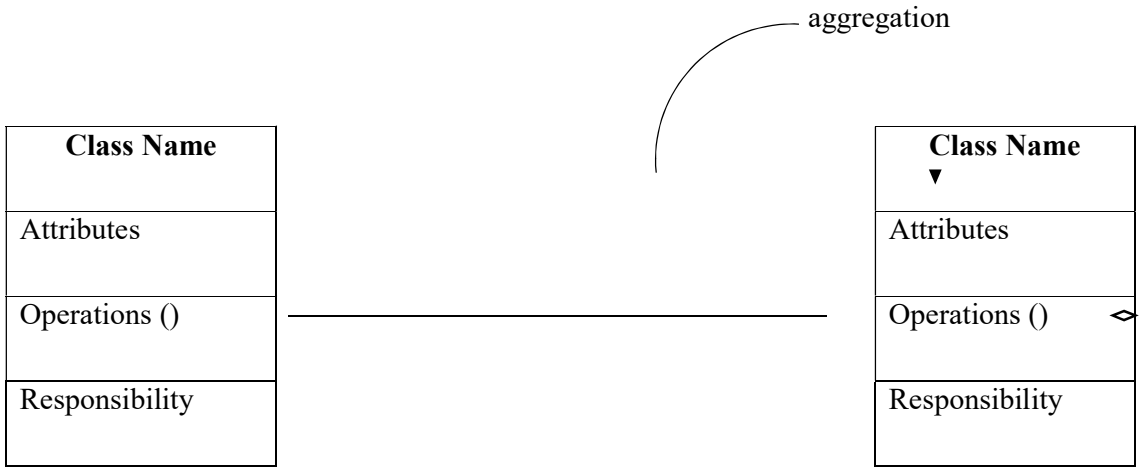


Class Name
Attributes
Operations ()
Responsibility

- **Composition and Aggregation**

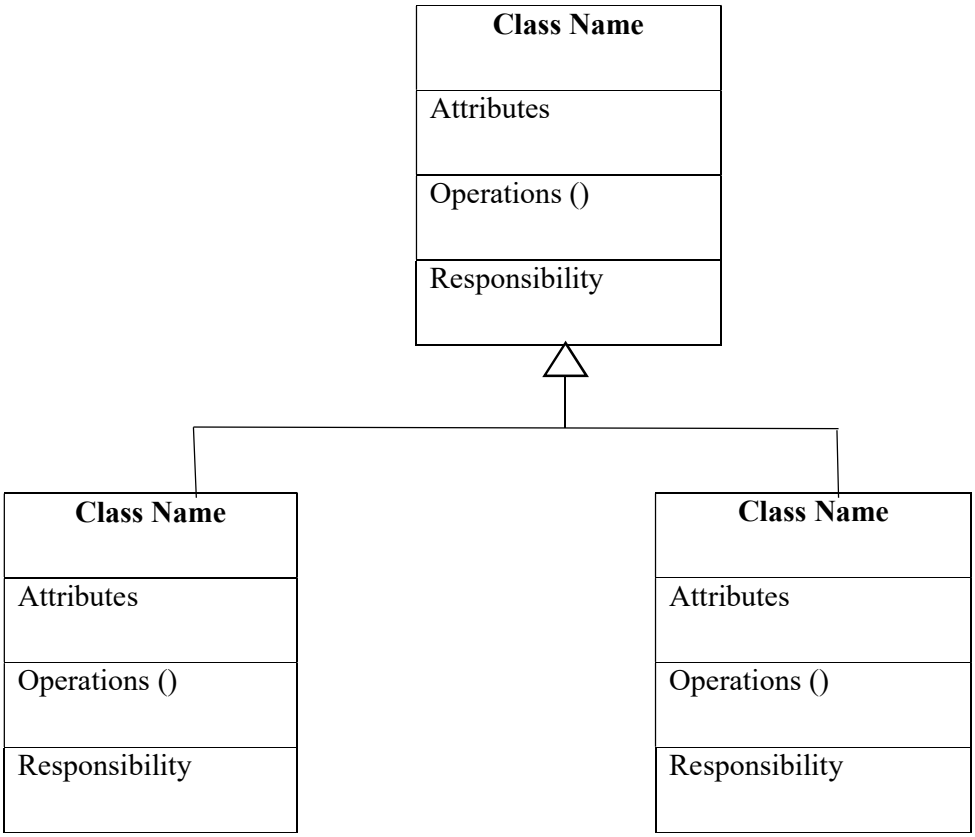
Composition is a special type of aggregation that denotes a strong ownership between classes. Use a hollow diamond to represent a simple aggregation relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other. The diamond ends in both composition and aggregation relationships point toward the "whole" class.





- **Generalization**

Generalization is another name for inheritance or an "is a" relationship. It refers to a relation between two classes where one class is a specialized version of another.


Class Name
Attributes
Operations ()
Responsibility
Class Name
Attributes
Operations ()
Responsibility
Class Name
Attributes
Operations ()
Responsibility

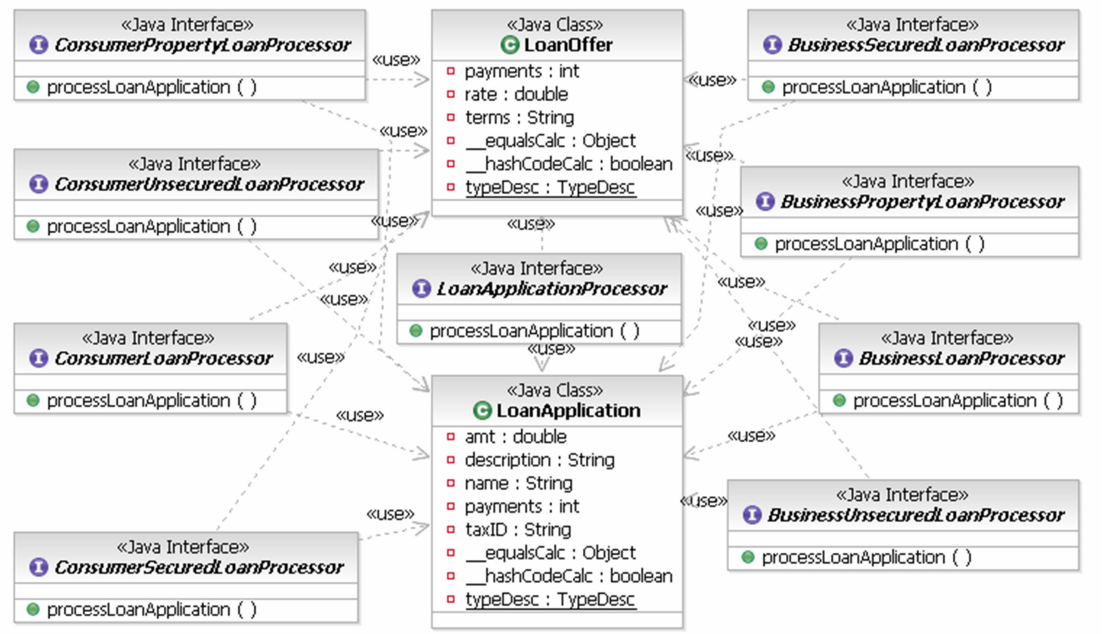
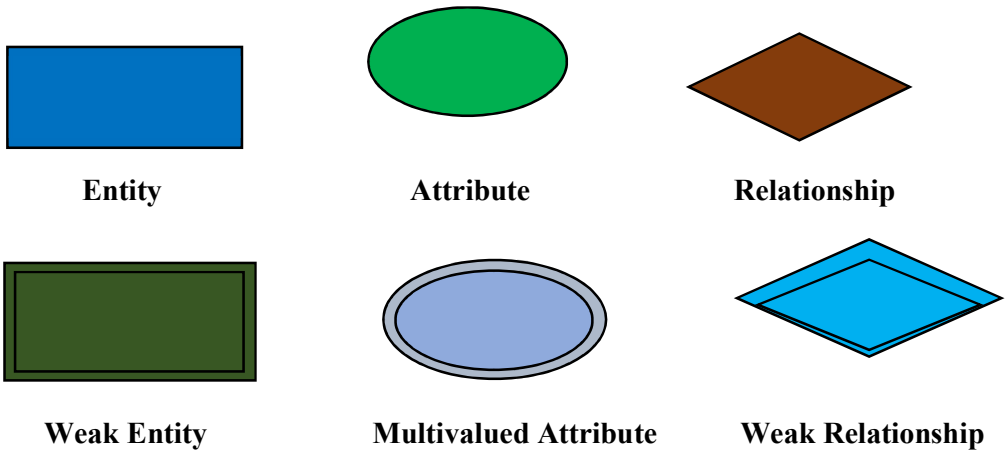


Fig. 1 Class Diagram

4.3 E-R DIAGRAM

An Entity Relationship Diagram (ERD) is a visual representation of different entities within a system and how they relate to each other.

Basic symbols and notations:



- **Entity:**

An entity can be a person, place, event, or object that is relevant to a given system. For example, a school system may include students, teachers, major courses, subjects, fees, and other items. Entities are represented in ER diagrams by a rectangle and named using singular nouns.

- **Weak Entity:**

A weak entity is an entity that depends on the existence of another entity. In more technical terms it can be defined as an entity that cannot be identified by its own attributes. It uses a foreign key combined with its attributed to form the primary key. An entity like order item is a good example for this.

- **Attribute:**

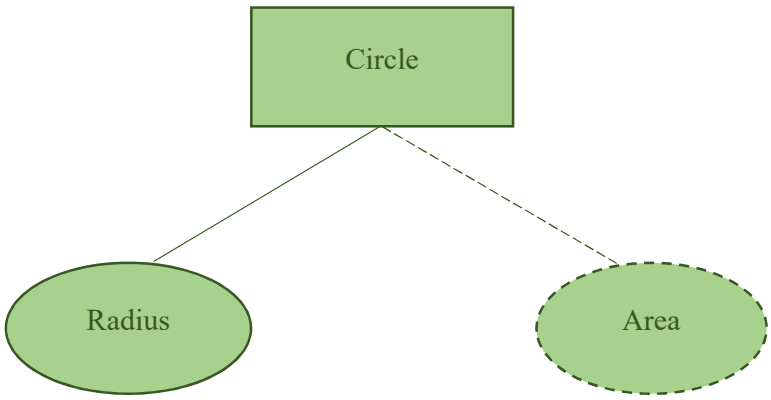
An attribute is a property, trait, or characteristic of an entity, relationship, or another attribute. For example, the attribute Inventory Item Name is an attribute of the entity Inventory Item. An entity can have as many attributes as necessary. Meanwhile, attributes can also have their own specific attributes.

- **Multivalued Attribute:**

If an attribute can have more than one value it is called a multi valued attribute. It is important to note that this is different to an attribute having its own attributes. For example, a teacher entity can have multiple subject values.

- **Derived Attribute:**

An attribute based on another attribute. This is found rarely in ER diagrams. For example, fora circle, the area can be derived from the radius.


Circle
Radius
Area

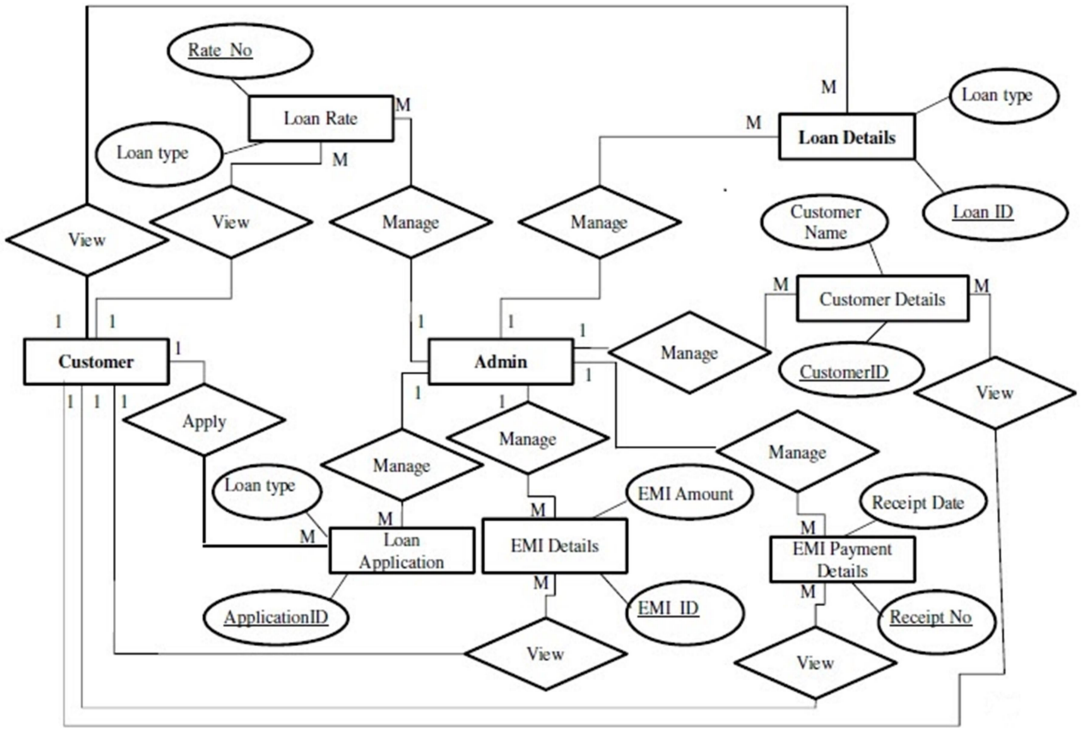


Fig. 2 E-R Diagram

4.4 ACTIVITY DIAGRAM

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modelling. They can also describe the steps in use case diagram. Activities modelled can be sequential and concurrent.

Basic Activity Diagram Notations and Symbols:

- **Initial State or Start Point:**

A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram. For activity diagram using swim lanes, make sure the start point is placed in the top left corner of the first column.

Start Point/Initial State

- **Activity or Action State:**

An action state represents the non-interruptible action of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.

Activity

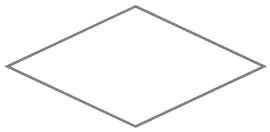
- **Action Flow:**

Action flows, also called edges and paths, illustrate the transitions from one action state to another. They are usually drawn with an arrowed line.

Action Flow

- **Decisions and Branching:**

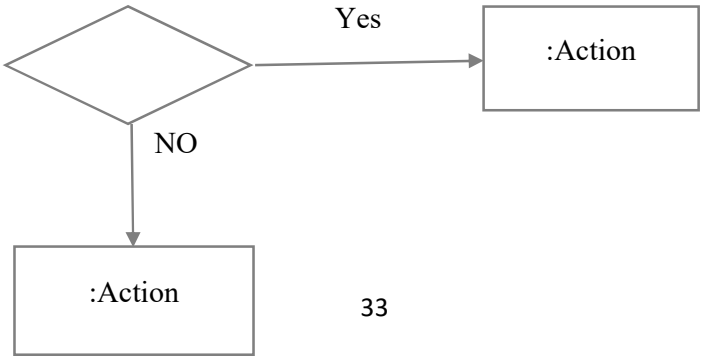
A diamond represents a decision with alternate paths. When an activity requires a decision prior to moving on to the next activity, add a diamond between the two activities. The outgoing alternates should be labelled with a condition or guard expression. You can also label one of the paths "else."



Decision Symbol

- **Guards:**

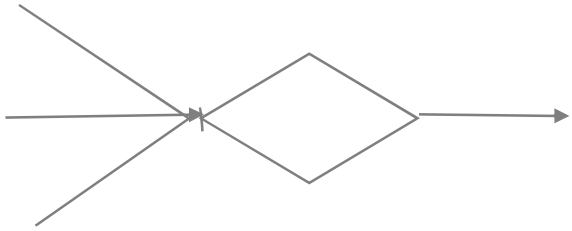
In UML, guards are a statement written next to a decision diamond that must be true before moving next to the next activity. These are not essential, but are useful when a specific answer, such as "Yes, three labels are printed," is needed before moving forward.



Guard Symbols

- **Merge Event:**

A merge event brings together multiple flows that are not concurrent.



Merge

- **Final State or End Point:**

An arrow pointing to a filled circle nested inside another circle represents the final action state.

End Point Symbol

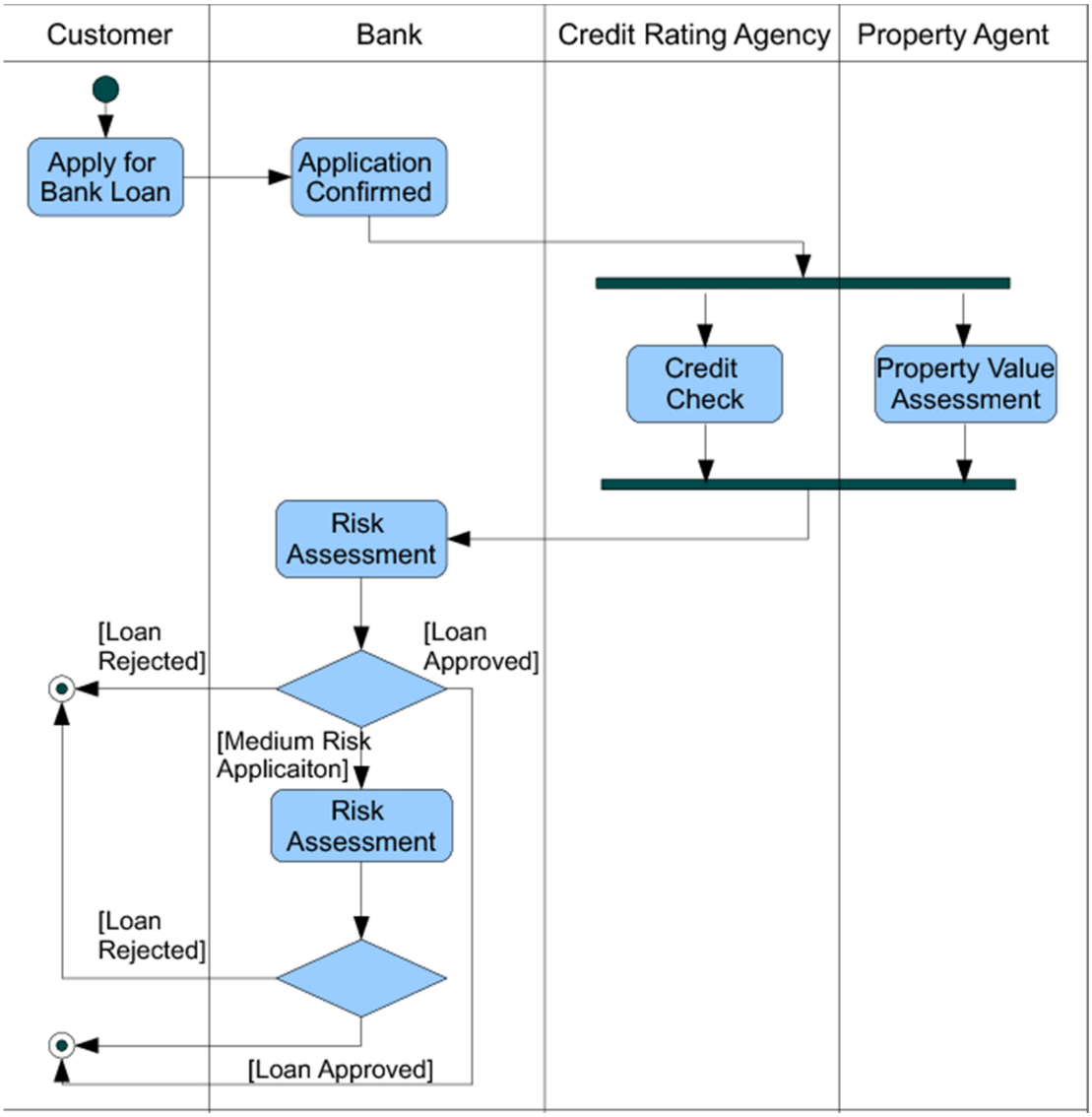


Fig. 3 Activity Diagram

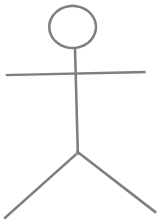
4.5 USECASE DIAGRAM

Use case diagram is a behavioural UML diagram type and frequently used to analyse various systems. They enable you to visualize the different types of roles in a system and how those roles interact with the system.

Basic diagram notation and symbols:

- **Actor:**

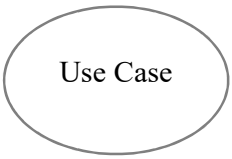
Actor in a use case diagram is any entity that performs a role in one given system. This could be a person, organization or an external system and usually drawn like skeleton shown below.



Actor

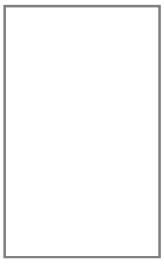
- **Use Case:**

A use case represents a function or an action within the system. It's drawn as an oval and named with the function.



- **System:**

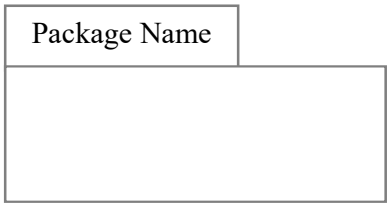
The system is used to define the scope of the use case and drawn as a rectangle. This an optional element but useful when you're visualizing large system.



System

- **Package:**

The package is another optional element that is extremely useful in complex diagrams. Similar to class diagrams, packages are used to group together use cases. They are drawn like the image shown below.



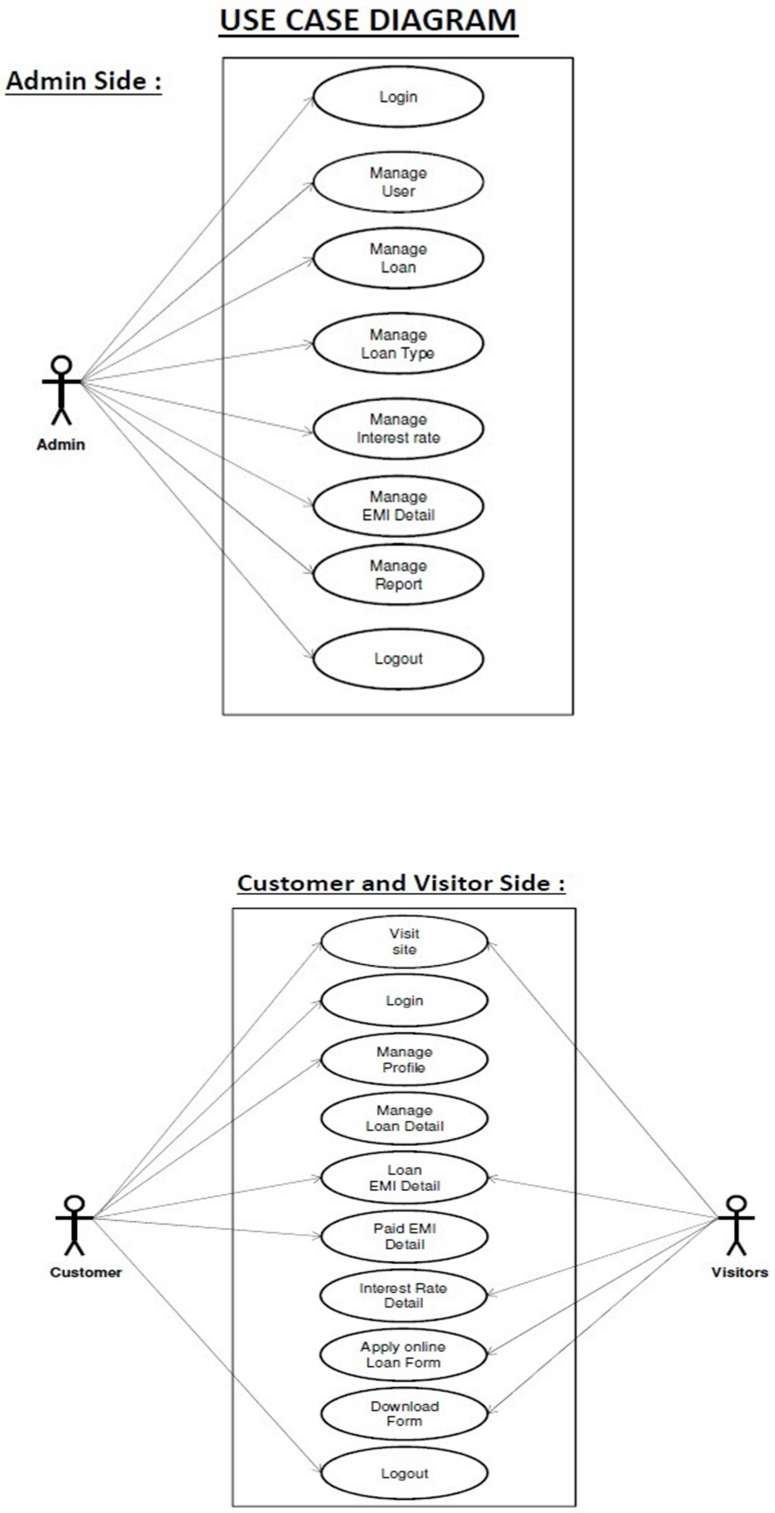


Fig. 4 Use Case Diagram

Chapter 5 CODING

5.1 Java Coding

- **Consumer App Service**

```
package com.kuliza.workbench.service;

import com.google.gson.Gson;
import com.kuliza.lending.common.pojo.ApiResponse;
import com.kuliza.lending.common.utils.CommonHelperFunctions;
import com.kuliza.workbench.model.PragatiAppDetails;
import com.kuliza.workbench.repository.PragatiAppDetailsRepository;
import java.util.Map;
import org.flowable.engine.RuntimeService;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;

@Service
public class ConsumerAppApiService {
  private static final Logger logger =
LoggerFactory.getLogger(ConsumerAppApiService.class);

  @Autowired PragatiAppDetailsRepository pragatiAppDetailsRepository;
  @Autowired RuntimeService runtimeService;

  public ApiResponse getConsumerAppStatus(Map<String, String> request)
      throws JSONException, JSONException {
    String appStatus = "";
    String ucicNoExist = "";
    JSONObject details = new JSONObject();
    int applicantCount = -1;
    String ucic = request.get("ucic").toString();
    try {
      PragatiAppDetails ucicExists = pragatiAppDetailsRepository.findByUcic(ucic);
      ucicNoExist = ucicExists.getUcic();
    } catch (Exception e) {
    }
    if (ucicNoExist.equalsIgnoreCase("")) {
      return new ApiResponse(HttpStatus.NOT_FOUND, "Customer Number doest
not exist");
    } else {
```

```
    PragatiAppDetails pragatiAppDetails =
pragatiAppDetailsRepository.findByUcic(ucic);
    logger.info("pragatiAppDetails :-{}",
pragatiAppDetails.getProcessInstanceId());
    String processInstanceId = pragatiAppDetails.getProcessInstanceId();
    Map<String, Object> variables =
runtimeService.getVariables(processInstanceId);
    JSONObject object = new JSONObject(new Gson().toJson(variables));
    logger.info("variables :-{}", object);

    try {
     applicantCount = object.getInt("applicantCount_hl");
    } catch (Exception e) {
    }

    if (applicantCount == -1) {
     return new ApiResponse(HttpStatus.NOT_FOUND, "Inquiry Number doest
not exist");
    }
    try {
     appStatus = object.getString("consumerAppStatus_hl");
     logger.info("appStatus.......:- {}", appStatus);
    } catch (Exception e) {
     e.printStackTrace();
    }

    JSONArray ucicArray = new JSONArray();
    JSONObject ucicObject = new JSONObject();
    String ucicId = "";

    for (int i = 0; i < applicantCount; i++) {
     try {
      ucicId = object.getString("applicant" + (i + 1) + "UcicId_hl");
     } catch (Exception e) {
     }

     ucicObject.put("applicant" + (i + 1), ucicId);
    }
    ucicArray.put(ucicObject);

    JSONArray coApplicantName = new JSONArray();
    JSONObject coApplicantNameObject = new JSONObject();
    for (int i = 1; i < applicantCount; i++) {
     coApplicantNameObject.put(
        "CoApplicant" + i, object.getString("applicant" + (i + 1) + "Name_hl"));
    }
    coApplicantName.put(coApplicantNameObject);

    if (appStatus.equalsIgnoreCase("")) {
```

```
        return new ApiResponse(HttpStatus.NOT_FOUND, "Consumer App Status
Not Found");
      }

      if (appStatus.equalsIgnoreCase("Under Progress")) {
        details.put("TypeOfLoan", object.getString("endUseOfLoanString_hl"));
        details.put("LANId", object.getString("loanId_hl"));
        details.put("CustomerId", ucic);
        details.put("Status", object.getString("consumerAppStatus_hl"));
        details.put("ApplicantName", object.getString("applicant1Name_hl"));
        details.put("CoApplicantName", coApplicantName);
        logger.info("UnderProgressDetails.....:- {}", details);
      }

      if (appStatus.equalsIgnoreCase("Rejected")) {
        details.put("TypeOfLoan", object.getString("endUseOfLoanString_hl"));
        details.put("LANId", object.getString("loanId_hl"));
        details.put("CustomerId", ucic);
        details.put("Status", object.getString("consumerAppStatus_hl"));
        try {
          String cml1Remarks = object.getString("cml1Remarks_hl");
          if (cml1Remarks.isEmpty() || cml1Remarks.equals("")) {
            details.put("RejectedRemarks", "NA");
          } else {
            details.put("RejectedRemarks", cml1Remarks);
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        details.put("ApplicantName", object.getString("applicant1Name_hl"));
        details.put("CoApplicantName", coApplicantName);
        logger.info("RejectedDetails.....:- {}", details);
      }
    }
    Map<String, Object> res =
CommonHelperFunctions.getHashMapFromJsonString(details.toString());
    return new ApiResponse(HttpStatus.OK, "SUCCESS", res);
  }
}
```

- **Consumer app controller**

```
package com.kuliza.workbench.controller;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.kuliza.lending.common.pojo.ApiResponse;
import com.kuliza.workbench.service.ConsumerAppApiService;
```

```
import java.util.Map;
import org.json.JSONException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ConsumerAppApiController {
  @Autowired ConsumerAppApiService consumerAppApiService;

  @PostMapping("/consumerAppStatus")
  public ApiResponse consumerAppStatus(@RequestBody Map<String, String>
request)
      throws JSONException, JsonProcessingException {

    return consumerAppApiService.getConsumerAppStatus(request);
  }
}
```

- **Cibil Controller**

```
package com.kuliza.workbench.controller;

import com.kuliza.lending.common.pojo.ApiResponse;
import com.kuliza.workbench.model.FinalCibilRequest;
import com.kuliza.workbench.service.CibilApiService;
import com.mashape.unirest.http.exceptions.UnirestException;
import org.json.JSONException;
import org.json.simple.parser.ParseException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import javax.xml.transform.TransformerException;
import java.io.IOException;

@RestController
public class CibilApiController {
  private static final Logger logger = LoggerFactory.getLogger(CibilApiController.class);

  @Autowired private CibilApiService cibilApiService;

  @RequestMapping(method = RequestMethod.POST, value = "/cibil")
  public ApiResponse getResponseFromIb(@RequestBody FinalCibilRequest
```

```
cibilRequest)
    throws UnirestException, JSONException, IOException, ParseException,
TransformerException {
   return cibilApiService.cibilApi(cibilRequest);
  }

}
```

- **Digio Controller**

```
package com.kuliza.workbench.controller;

import com.kuliza.lending.common.pojo.ApiResponse;
import com.kuliza.workbench.model.PragatiAppDetails;
import com.kuliza.workbench.repository.PragatiAppDetailsRepository;
import com.kuliza.workbench.service.DigioService;
import java.io.IOException;
import java.util.List;
import java.util.Map;
import org.flowable.engine.RuntimeService;
import org.flowable.engine.TaskService;
import org.flowable.task.api.Task;
import org.json.JSONException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

@RestController
public class DigioController {
  private static final Logger logger =
LoggerFactory.getLogger(DigioController.class);

  @Autowired DigioService digioService;
  @Autowired PragatiAppDetailsRepository pragatiAppDetailsRepository;

  @Autowired RuntimeService runtimeService;

  @Autowired TaskService taskService;
```

```
  @PostMapping(value = "/fileUpload")
  public ApiResponse uploadAgreement(MultipartFile uploadFile) throws
IOException {
    ApiResponse response = new ApiResponse(HttpStatus.BAD_REQUEST, "file
not valid", null);

    if (uploadFile.isEmpty()) return response;
    response = digioService.digioFileToPdf(uploadFile);
    return response;
  }

  @PostMapping(value = "/product/saveVariables")
  public ApiResponse saveVariables(@RequestBody Map<String, Object> request)
throws IOException {
    ApiResponse response = new ApiResponse(HttpStatus.OK, "not valid", null);
    String processInstanceId = request.get("processInstanceId").toString();
    try {
      logger.info("Inside try block.....");
      PragatiAppDetails pragatiAppDetails =
        pragatiAppDetailsRepository.findByKulizaAppId(
          request.get("lasid").toString()); // lasid is shared with frontend
      logger.info(
        "Pragati App Details Process Instance Id......:- {}",
        pragatiAppDetails.getProcessInstanceId());
      String caseInstanceId =
        runtimeService
          .getVariable(pragatiAppDetails.getProcessInstanceId(), "caseInstanceId")
          .toString();
      logger.info("Case Instance Id.....:- {}", caseInstanceId);
      runtimeService.setVariable(processInstanceId, "oldCaseInstanceId",
caseInstanceId);
      List<Task> allTasks =

taskService.createTaskQuery().processInstanceId(processInstanceId).active().list();
      taskService.complete(allTasks.get(0).getId());
    } catch (Exception e) {
      e.printStackTrace();
    }
    return response;
  }

  @PostMapping(value = "/createDocFromTemplate")
  public ApiResponse docFromTemplate() throws IOException, JSONException {
    return digioService.generateDocFromTemplates();
  }
}
```

- **Create Mandate Model Class**

```
package com.kuliza.workbench.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;

@Entity
@Table(name = "Createmandate")
public class CreateMandate extends BaseModel {
  String customer_identifier;
  String auth_mode;
  String mandate_type;
  String corporate_config_id;

  @Column(unique = true)
  String processInstanceId;

  String enachStatus;
  boolean notifyCustomer;

  int reTriggerCount;
  String mandateId;

  @OneToOne(cascade = javax.persistence.CascadeType.ALL)
  @Cascade(CascadeType.ALL)
  MandateData mandate_data;

  public CreateMandate() {}

  public String getCustomer_identifier() {
    return customer_identifier;
  }

  public void setCustomer_identifier(String customer_identifier) {
    this.customer_identifier = customer_identifier;
  }

  public String getAuth_mode() {
    return auth_mode;
  }

  public void setAuth_mode(String auth_mode) {
    this.auth_mode = auth_mode;
  }

  public String getMandate_type() {
```

```
 return mandate_type;
}

public void setMandate_type(String mandate_type) {
 this.mandate_type = mandate_type;
}

public String getCorporate_config_id() {
 return corporate_config_id;
}

public void setCorporate_config_id(String corporate_config_id) {
 this.corporate_config_id = corporate_config_id;
}

public String getProcessInstanceId() {
 return processInstanceId;
}

public void setProcessInstanceId(String processInstanceId) {
 this.processInstanceId = processInstanceId;
}

public String getEnachStatus() {
 return enachStatus;
}

public void setEnachStatus(String enachStatus) {
 this.enachStatus = enachStatus;
}

public boolean isNotifyCustomer() {
 return notifyCustomer;
}

public void setNotifyCustomer(boolean notifyCustomer) {
 this.notifyCustomer = notifyCustomer;
}

public int getReTriggerCount() {
 return reTriggerCount;
}

public void setReTriggerCount(int reTriggerCount) {
 this.reTriggerCount = reTriggerCount;
}

public MandateData getMandate_data() {
 return mandate_data;
```

```
  }

  public void setMandate_data(MandateData mandate_data) {
    this.mandate_data = mandate_data;
  }

  public String getMandateId() {
    return mandateId;
  }

  public void setMandateId(String mandateId) {
    this.mandateId = mandateId;
  }
}
```

- **Posidex Model Class**

```
package com.kuliza.workbench.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;

@Entity
@Table(name = "Posidex")
public class Posidex extends BaseModel {

  private String assetSrNo;

  @Column(length = 6000)
  private String posidexResponse;

  private String caseInstanceId;

  public Posidex() {}

  public Posidex(String assetSrNo, String posidexResponse) {
    this.assetSrNo = assetSrNo;
    this.posidexResponse = posidexResponse;
  }

  public String getAssetSrNo() {
    return assetSrNo;
  }

  public void setAssetSrNo(String assetSrNo) {
    this.assetSrNo = assetSrNo;
```

```
  }

  public String getPosidexResponse() {
    return posidexResponse;
  }

  public void setPosidexResponse(String posidexResponse) {
    this.posidexResponse = posidexResponse;
  }

  public String getCaseInstanceId() {
    return caseInstanceId;
  }

  public void setCaseInstanceId(String caseInstanceId) {
    this.caseInstanceId = caseInstanceId;
  }
}
```

- **Final Submit Service Class**

```
package com.kuliza.workbench.service;

import com.google.gson.Gson;
import com.kuliza.lending.common.pojo.ApiResponse;
import com.kuliza.lending.common.utils.CommonHelperFunctions;
import com.kuliza.workbench.model.PragatiAppDetails;
import com.kuliza.workbench.repository.PragatiAppDetailsRepository;
import org.flowable.engine.RuntimeService;
import org.flowable.engine.TaskService;
import org.flowable.task.api.Task;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Map;

@Service
public class FinalSubmitApiService {
  private static final Logger logger =
LoggerFactory.getLogger(FinalSubmitApiService.class);
```

```
  @Autowired private RuntimeService runtimeService;
  @Autowired private TaskService taskService;
  @Autowired PragatiAppDetailsRepository pragatiAppDetailsRepository;

  public com.kuliza.lending.common.pojo.ApiResponse finalSubmit(Map<String,
Object> request)
      throws JSONException {
    PragatiAppDetails pragatiAppDetails =

pragatiAppDetailsRepository.findByKulizaAppId(request.get("los_id").toString());
    logger.info("pragatiAppDetails :-{}", pragatiAppDetails.getProcessInstanceId());
    String los_id = pragatiAppDetails.getProcessInstanceId();
    runtimeService.setVariable(
        los_id, "finalSubmitRequestBody",
CommonHelperFunctions.getJsonString(request));

    try {
      request = CommonHelperFunctions.fromJson(mappingVariables(request));
      runtimeService.setVariables(los_id, request);
      try {
        String customerNo = runtimeService.getVariable(los_id,
"applicant1UcicId_hl").toString();
        logger.info("CUSTOMER NUMBER :-{}", customerNo);
        pragatiAppDetails.setUcic(customerNo);
        pragatiAppDetailsRepository.save(pragatiAppDetails);
      } catch (Exception e) {
      }
      List<Task> tasks =
          taskService
              .createTaskQuery()
              .processInstanceId(los_id)
              .active()
              .orderByTaskCreateTime()
              .desc()
              .list();
      taskService.complete(tasks.get(0).getId());
      return new ApiResponse(HttpStatus.OK, "Data Saved");
    } catch (Exception e) {
      e.printStackTrace();
      return new ApiResponse(HttpStatus.BAD_REQUEST, "Bad Request");
    }
  }

  public JSONObject mappingVariables(Map<String, Object> request) throws
JSONException {
    JSONObject object = new JSONObject(new Gson().toJson(request));
    logger.info("request :- {}", object);
    String losId = object.get("los_id").toString();
```

```
    String newLosId = "LN" + losId.substring(2);
    object.put("loanId_hl", newLosId);
    String createdAt = object.get("created_at").toString();
    String date = createdAt.substring(0, 10);
    object.put("applicationCreationDate_hl", date);
    object.put("enquiryDate_hl", object.get("created_at").toString());
    object.put("isSentToTechApp_hl", "no");
    object.put("applicationId_hl", object.get("application_id"));
    //    JSONArray applicantsArray = object.getJSONArray("applicants");
    //    logger.info("applicantsArray : {}", applicantsArray);
    //    object.put("applicantsArray_hl", applicantsArray.toString());

    JSONObject basicDetails = object.getJSONObject("basic_details");
    int loanAmount =
basicDetails.getJSONObject("loan_eligibility").getInt("loan_amount");
    object.put("loanAmount_hl", loanAmount);
    int tenureInMonths =
basicDetails.getJSONObject("loan_eligibility").getInt("tenure_in_months");
    object.put("loanTenure_hl", tenureInMonths);
    double expectedInterestRate =

basicDetails.getJSONObject("loan_eligibility").getInt("expected_interest_rate");
    object.put("expectedRoi_hl", expectedInterestRate);
    int grossSalary =
basicDetails.getJSONObject("loan_eligibility").getInt("gross_salary");
    object.put("grossSalary_hl", grossSalary);
    object.put("salaryBand_hl", grossSalary);
    String maritalStatusOfPrimaryApplicant =
        basicDetails.get("marital_status_of_primary_applicant").toString();
    object.put("applicant1MaritalStatus_hl", maritalStatusOfPrimaryApplicant);

    JSONObject propertyDetails = object.getJSONObject("property_details");
    String end_use_of_loan = propertyDetails.get("end_use_of_loan").toString();
    object.put("endUseOfLoanString_hl", end_use_of_loan);
    logger.info("endUseOfLoanString_hl : {}", end_use_of_loan);
    String propertyAddress =
        propertyDetails.getJSONObject("property_address").get("address").toString();
    String districtOrCity =

propertyDetails.getJSONObject("property_address").get("district_or_city").toStrin
g();
    object.put("propertyCity", districtOrCity);

    String state =
propertyDetails.getJSONObject("property_address").get("state").toString();
    object.put("propertyState", state);

    int pinCode =
propertyDetails.getJSONObject("property_address").getInt("pincode");
```

```
    object.put("propertyPinCode", pinCode);

    String fullPropertyAddress =
        propertyAddress + ", " + districtOrCity + ", " + state + ", " + pinCode;
    object.put("propertyAddress_hl", fullPropertyAddress);
    logger.info("fullPropertyAddresss : {}", fullPropertyAddress);

    JSONArray propertyTechVerification = new JSONArray();
    JSONArray propertyLegalVerification = new JSONArray();
    JSONObject propertyTechVerificationObject = new JSONObject();
    JSONObject propertyLegalVerificationObject = new JSONObject();
    propertyTechVerificationObject.put("propertyAddress_hl",
fullPropertyAddress);
    propertyLegalVerificationObject.put("propertyAddress_hl",
fullPropertyAddress);
    propertyLegalVerification.put(propertyLegalVerificationObject);
    propertyTechVerification.put(propertyTechVerificationObject);
    object.put("propertyTechVerification_hl", propertyTechVerification);
    object.put("propertyLegalVerification_hl", propertyTechVerification);

    String propertyOwner =

CommonHelperFunctions.getStringValue(propertyDetails.get("property_owner"));
    object.put("propertyOwner_hl", propertyOwner);

    JSONArray applicant1PropertyDetails = new JSONArray();
    JSONObject applicant1Property = new JSONObject();
    String propertyType = propertyDetails.get("property_type").toString();
    applicant1Property.put("applicant1DocumentType_hl", propertyType);
    applicant1PropertyDetails.put(applicant1Property);
    object.put("applicant1PropertyDetails_hl", applicant1PropertyDetails);

    try {
      JSONObject location = object.getJSONObject("location");
      String address = location.get("address").toString();
      object.put("applicant1Location_hl", address);
      Double lat = location.getDouble("lat");
      object.put("applicant1Latitude_hl", lat);
      Double lng = location.getDouble("lng");
      object.put("applicant1Longitude_hl", lng);
    } catch (Exception e) {
    }

    JSONArray applicants = object.getJSONArray("applicants");
    int applicantCount = applicants.length();
    object.put("applicantCount_hl", applicantCount);
    logger.info("applicantCount :- {}", applicantCount);

    int count = 1;
```

```
   for (int i = 0; i < applicantCount; i++) {
     JSONObject applicant = (JSONObject) applicants.get(i);
     String applicantType = applicant.get("applicant_type").toString();

     object.put("applicant" + (i + 1) + "ApplicantType_hl", applicantType);
     try {
       JSONArray applicantBStatementDetails = new JSONArray();
       JSONObject applicantBStatement = new JSONObject();
       String applicantBankStatementUrl =
          applicant.getJSONObject("bank_statements").get("url").toString();
       applicantBStatement.put(
          "applicant" + (i + 1) + "6MBankStatement_hl",
applicantBankStatementUrl);
       applicantBStatementDetails.put(applicantBStatement);
       object.put(
          "applicant" + (i + 1) + "BStatementDetails_hl",
applicantBStatementDetails.toString());
     } catch (Exception e) {
       e.printStackTrace();
     }
     String applicantName = applicant.get("applicant_name").toString();
     object.put("applicant" + (i + 1) + "Name_hl", applicantName);
     try {
       String ucic = applicant.get("ucic").toString();
       object.put("applicant" + (i + 1) + "UcicId_hl", ucic);
     } catch (Exception e) {
       e.printStackTrace();
     }
     int applicantId = applicant.getInt("applicant_id");
     object.put("applicant" + (i + 1) + "_id", applicantId);
     String applicantMobile = applicant.get("mobile").toString();
     object.put("applicant" + (i + 1) + "MobileNumber_hl", applicantMobile);
     String applicantRelationshipWithPrimaryApplicant =
        applicant.get("relationship_with_primary_applicant").toString();
     object.put("applicant" + (i + 1) + "Relation_hl",
applicantRelationshipWithPrimaryApplicant);
     try {
       String applicantOfficialEmailId =

applicant.getJSONObject("employment_check").get("official_email_id").toString()
;
       object.put("applicant" + (i + 1) + "Email_hl", applicantOfficialEmailId);
       object.put("applicant" + (i + 1) + "OfficialEMail_hl",
applicantOfficialEmailId);
       String applicantDesignation =

applicant.getJSONObject("employment_check").get("designation").toString();
       object.put("applicant" + (i + 1) + "Designation_hl", applicantDesignation);
       String applicantEmployerName =
```

```
applicant.getJSONObject("employment_check").get("employer_name").toString();
      object.put("applicant" + (i + 1) + "EmployerName_hl",
applicantEmployerName);
      String applicantEmployerType =

applicant.getJSONObject("employment_check").get("employer_type").toString();
      object.put("applicant" + (i + 1) + "Constitution_hl", applicantEmployerType);
      if (applicantEmployerType.equalsIgnoreCase("Pvt Ltd")
         || applicantEmployerType.equalsIgnoreCase("Public")) {
        object.put("is" + "Applicant" + (i + 1) + "GovtEmp", "Yes");
      }

      String applicantEmployerLocalAddress =
         applicant
            .getJSONObject("employment_check")
            .getJSONObject("employer_local_address")
            .get("address")
            .toString();
      String applicantEmployerDistrictOrCity =
         applicant
            .getJSONObject("employment_check")
            .getJSONObject("employer_local_address")
            .get("district_or_city")
            .toString();
      String applicantEmployerState =
         applicant
            .getJSONObject("employment_check")
            .getJSONObject("employer_local_address")
            .get("state")
            .toString();
      String applicantEmployerPinCode =
         applicant
            .getJSONObject("employment_check")
            .getJSONObject("employer_local_address")
            .get("pincode")
            .toString();
      String applicantFinalEmployerAddress =
         applicantEmployerLocalAddress
            + ", "
            + applicantEmployerDistrictOrCity
            + ", "
            + applicantEmployerState
            + ", "
            + applicantEmployerPinCode;
      object.put("applicant" + (i + 1) + "EmployerAddress_hl",
applicantFinalEmployerAddress);
      Boolean applicant_26asAvailable =
```

```
applicant.getJSONObject("employment_check").getBoolean("26as_available");
      object.put("applicant" + (i + 1) + "Form26AS_hl", applicant_26asAvailable);
      String employmentClass =

applicant.getJSONObject("employment_check").get("employment_class").toString
();
      object.put("employmentType1_app", employmentClass);
    } catch (Exception e) {
      e.printStackTrace();
    }
    String applicantGender =
applicant.getJSONObject("aadhaar").get("gender").toString();
    object.put("applicant" + (i + 1) + "Gender_hl", applicantGender);
    String aadhaarName =
       CommonHelperFunctions.getStringValue(
          applicant.getJSONObject("aadhaar").get("aadhaar_name"));
    object.put("applicant" + (i + 1) + "AadhaarName_hl", aadhaarName);
    JSONArray applicantAadharDetails = new JSONArray();
    JSONObject applicantAadhar = new JSONObject();
    String applicantAadhaarName =
       applicant.getJSONObject("aadhaar").get("aadhaar_name").toString();
    applicantAadhar.put("applicant" + (i + 1) + "AadharName_hl",
applicantAadhaarName);
    String applicantAadhaarNumber =
       applicant.getJSONObject("aadhaar").get("aadhaar_number").toString();
    applicantAadhar.put("applicant" + (i + 1) + "AadharNumber_hl",
applicantAadhaarNumber);
    applicantAadharDetails.put(applicantAadhar);
    object.put("applicant" + (i + 1) + "AadharDetails_hl", applicantAadharDetails);
    try {
      JSONArray applicantPanDetails = new JSONArray();
      JSONObject applicantPan = new JSONObject();
      String applicantPanName =
applicant.getJSONObject("pan").get("pan_name").toString();
      applicantPan.put("applicant" + (i + 1) + "PanName_hl", applicantPanName);
      String applicantPanNumber =
applicant.getJSONObject("pan").get("pan").toString();
      applicantPan.put("applicant" + (i + 1) + "PanNumber_hl",
applicantPanNumber);
      //     try {
      //       String applicantPanUrl =
applicant.getJSONObject("pan").get("pan_url");
      //       applicantPan.put("applicant"+(i+1)+"PanCard_hl", applicantPanUrl);
      //     } catch (Exception e) {
      //       e.printStackTrace();
      //     }
      applicantPanDetails.put(applicantPan);
      object.put("applicant" + (i + 1) + "PanDetails_hl", applicantPanDetails);
    } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        JSONArray applicantAddressDetails = new JSONArray();
        JSONObject applicantAddress = new JSONObject();
        String applicantCurrentAddress =
            applicant.getJSONObject("current_address").get("address").toString();
        String applicantCurrentAddressDistrict =

applicant.getJSONObject("current_address").get("district_or_city").toString();
        String applicantCurrentAddressState =
            applicant.getJSONObject("current_address").get("state").toString();
        String applicantCurrentAddressPinCode =
            applicant.getJSONObject("current_address").get("pincode").toString();
        String applicantFinalCurrentAddress =
            applicantCurrentAddress
                + ", "
                + applicantCurrentAddressDistrict
                + ", "
                + applicantCurrentAddressState
                + ", "
                + applicantCurrentAddressPinCode;
        applicantAddress.put(
            "applicant" + (i + 1) + "CurrentAddress_hl", applicantFinalCurrentAddress);
        String applicantPermanentAddress =
            applicant
                .getJSONObject("residential_verification")
                .getJSONObject("permanent_address")
                .get("address")
                .toString();
        String applicantPermanentAddressPinCode =
            applicant
                .getJSONObject("residential_verification")
                .getJSONObject("permanent_address")
                .get("pincode")
                .toString();
        String applicantFinalPermanentAddress =
            applicantPermanentAddress + ", " + applicantPermanentAddressPinCode;
        applicantAddress.put(
            "applicant" + (i + 1) + "PermanentAddress_hl",
applicantFinalPermanentAddress);
        String applicantCurrentAddressVintage =
            applicant
                .getJSONObject("residential_verification")
                .get("current_address_vintage")
                .toString();
        String years = "", months = "";
        int totalMonths = 0;
        double totalYears;
        applicantCurrentAddressVintage =
```

```
applicantCurrentAddressVintage.replaceAll("\\s", "");
    if (applicantCurrentAddressVintage.contains("Y")) {
     years =
        applicantCurrentAddressVintage.substring(
           0, applicantCurrentAddressVintage.indexOf("Y"));
    }
    if (applicantCurrentAddressVintage.contains("M")) {
     months =
        applicantCurrentAddressVintage.substring(
           applicantCurrentAddressVintage.indexOf("s") + 1,
           applicantCurrentAddressVintage.indexOf("M"));
    }
    totalMonths = (Integer.parseInt(years) * 12) + Integer.parseInt(months);
    totalYears = (double) totalMonths / 12;
    applicantAddress.put(
       "applicant" + (i + 1) + "YearsAtCurrentResidency_hl",
Math.round(totalYears));
    applicantAddressDetails.put(applicantAddress);
    object.put("applicant" + (i + 1) + "AddressDetails_hl",
applicantAddressDetails);
    String applicantDependentsCounts =

applicant.getJSONObject("residential_verification").get("dependents_counts").toSt
ring();
    object.put("applicant" + (i + 1) + "DependentsCount_hl",
applicantDependentsCounts);
    object.put("applicant" + (i + 1) + "Dependents_hl",
applicantDependentsCounts);
    String applicantJobNature =

applicant.getJSONObject("residential_verification").get("job_nature").toString();
    object.put("applicant" + (i + 1) + "JobNature_hl", applicantJobNature);
    String applicantEducation =

applicant.getJSONObject("residential_verification").get("education").toString();
    object.put("applicant" + (i + 1) + "EducationQualification_hl",
applicantEducation);
    object.put("applicant" + (i + 1) + "currentresidenceinYears_hl",
Math.round(totalYears));

    //     JSONArray applicantSalarySlipDetails1 = new JSONArray();
    //     JSONObject applicantSalarySlip = new JSONObject();
    //     String applicantPayslipsUrl =
applicant.getJSONObject("payslips").get("url");
    //     applicantSalarySlip.put("applicant"+(i+1)+"SalarySlip_hl",
applicantPayslipsUrl);
    //     applicantSalarySlipDetails1.put(applicantSalarySlip);
    //     object.put("applicantSalarySlipDetails1_hl", applicantSalarySlipDetails1);
    logger.info("ObjectForApplicant" + (i + 1) + " :- {}", object);
```

```
      if (applicantType.equals("Primary Applicant")) {
        object = saveDetails(object, applicant, applicantType, 0);
      }
    }

    for (int i = 0; i < applicantCount; i++) {
      JSONObject applicant = (JSONObject) applicants.get(i);
      String applicantType = applicant.get("applicant_type").toString();
      if (!applicantType.equals("Primary Applicant")) {
        object = saveDetails(object, applicant, applicantType, count);
        count++;
      }
    }

    JSONArray references = object.getJSONArray("references");
    int referenceCount = references.length();
    logger.info("referenceCount :- {}", applicantCount);
    for (int i = 0; i < referenceCount; i++) {
      JSONObject reference = (JSONObject) references.get(i);
      try {
        String referenceFullName = reference.get("full_name").toString();
        object.put("reference" + (i + 1) + "Name_hl", referenceFullName);
        String referenceResidentialAddress =
            reference.getJSONObject("residential_address").get("address").toString();
        object.put("reference" + (i + 1) + "ResidAddress_hl",
referenceResidentialAddress);
        String referenceOfficeAddress =
            reference.getJSONObject("office_address").get("address").toString();
        object.put("reference" + (i + 1) + "OfficeAddress_hl",
referenceOfficeAddress);
        String referenceMobileNumber = reference.get("mobile_number").toString();
        object.put("reference" + (i + 1) + "PhoneNumber_hl",
referenceMobileNumber);
        String referenceRelationWithApplicant =
reference.get("relation_with_applicant").toString();
        object.put("reference" + (i + 1) + "Relationship_hl",
referenceRelationWithApplicant);
        String referenceKnownSince = reference.get("known_since").toString();
        object.put("reference" + (i + 1) + "KnownSince_hl", referenceKnownSince);
        String referenceOccupation = reference.get("occupation").toString();
        object.put("reference" + (i + 1) + "Occupation_hl", referenceOccupation);
      } catch (Exception e) {
        e.printStackTrace();
      }
    }

    JSONObject pdInput = object.getJSONObject("pd_input");
    int monthlyHouseholdIncome = pdInput.getInt("monthly_household_income");
```

```
    object.put("pdIncome_hl", monthlyHouseholdIncome);
    String anyonePoliticallyAffiliatedInFamily =
        pdInput.get("anyone_politically_affiliated_in_family").toString();
    object.put("pdPoliticalAffliation_hl", anyonePoliticallyAffiliatedInFamily);
    String employersSector = pdInput.get("employers_sector").toString();
    object.put("pdEmploymentSector_hl", employersSector);
    String employmentType = pdInput.get("employment_type").toString();
    object.put("pdEmploymentType_hl", employmentType);
    int monthlyDiscretionaryInvestments =
pdInput.getInt("monthly_discretionary_investments");
    object.put("pdDiscInvestment_hl", monthlyDiscretionaryInvestments);
    int monthlyNonDiscretionaryInvestments =
        pdInput.getInt("monthly_non_discretionary_investments");
    object.put("pdNonDiscInvestment_hl", monthlyNonDiscretionaryInvestments);
    int totalNetWorth = pdInput.getInt("total_net_worth");
    object.put("pdMarketValue_hl", totalNetWorth);
    String contributionToPurchaseInPercent =
        pdInput.get("contribution_to_purchase_in_percent").toString();
    object.put("pdOwnContribution_hl", contributionToPurchaseInPercent);
    object.put("ownContribution_hl", contributionToPurchaseInPercent);
    String source_of_contribution =
pdInput.get("source_of_contribution").toString();
    object.put("pdSource_hl", source_of_contribution);
    String severeMedicalConditionInFamily =
        pdInput.get("severe_medical_condition_in_family").toString();
    object.put("pdSevereCondition_hl", severeMedicalConditionInFamily);
    String religion = pdInput.get("religion").toString();
    object.put("pdReligion_hl", religion);
    String caste = pdInput.get("caste").toString();
    object.put("pdCaste_hl", caste);

    JSONObject inquiry = object.getJSONObject("inquiry");
    String source = inquiry.get("source").toString();
    object.put("source_hl", source);
    String status = inquiry.get("status").toString();
    object.put("enquiryStatus_hl", status);

    JSONObject branch =
object.getJSONObject("inquiry").getJSONObject("branch");
    String branchName = branch.get("branch_name").toString();
    object.put("loanBranch_hl", branchName);
    object.put("branchName_hl", branchName.replaceAll("\\s", ""));

    //    PragatiAppDetails pragatiAppDetails = new PragatiAppDetails();
    //    pragatiAppDetails.setApplicationId(application_id);
    //    String los_id = (String) request.get("los_id");
    //    pragatiAppDetails.setLosId(los_id);
    //    //    pragatiAppDetails.setFinalSubmitRequest(object.toString());
    //    pragatiAppDetailsRepository.save(pragatiAppDetails);
```

```
    return object;
  }

  public JSONObject saveDetails(
      JSONObject object, JSONObject applicant, String applicantType, int i) throws
JSONException {
    object.put("applicant" + (i + 1) + "ApplicantType_hl", applicantType);
    try {
      JSONArray applicantBStatementDetails = new JSONArray();
      JSONObject applicantBStatement = new JSONObject();
      String applicantBankStatementUrl =
          applicant.getJSONObject("bank_statements").get("url").toString();
      applicantBStatement.put(
          "applicant" + (i + 1) + "6MBankStatement_hl", applicantBankStatementUrl);
      applicantBStatementDetails.put(applicantBStatement);
      object.put(
          "applicant" + (i + 1) + "BStatementDetails_hl",
applicantBStatementDetails.toString());
    } catch (Exception e) {
      e.printStackTrace();
    }
    String applicantName = applicant.get("applicant_name").toString();
    object.put("applicant" + (i + 1) + "Name_hl", applicantName);
    try {
      String ucic = applicant.get("ucic").toString();
      object.put("applicant" + (i + 1) + "UcicId_hl", ucic);

    } catch (Exception e) {
      e.printStackTrace();
    }

    int applicantId = applicant.getInt("applicant_id");
    object.put("applicant" + (i + 1) + "_id", applicantId);
    String applicantMobile = applicant.get("mobile").toString();
    object.put("applicant" + (i + 1) + "MobileNumber_hl", applicantMobile);
    String applicantRelationshipWithPrimaryApplicant =
        applicant.get("relationship_with_primary_applicant").toString();
    object.put("applicant" + (i + 1) + "Relation_hl",
applicantRelationshipWithPrimaryApplicant);
    try {
      String applicantOfficialEmailId =

applicant.getJSONObject("employment_check").get("official_email_id").toString()
;
      object.put("applicant" + (i + 1) + "Email_hl", applicantOfficialEmailId);
      object.put("applicant" + (i + 1) + "OfficialEMail_hl",
applicantOfficialEmailId);
      String applicantDesignation =
```

```
applicant.getJSONObject("employment_check").get("designation").toString();
    object.put("applicant" + (i + 1) + "Designation_hl", applicantDesignation);
    String applicantEmployerName =

applicant.getJSONObject("employment_check").get("employer_name").toString();
    object.put("applicant" + (i + 1) + "EmployerName_hl",
applicantEmployerName);
    String applicantEmployerType =

applicant.getJSONObject("employment_check").get("employer_type").toString();
    object.put("applicant" + (i + 1) + "Constitution_hl", applicantEmployerType);
    String applicantEmployerLocalAddress =
       applicant
          .getJSONObject("employment_check")
          .getJSONObject("employer_local_address")
          .get("address")
          .toString();
    String applicantEmployerDistrictOrCity =
       applicant
          .getJSONObject("employment_check")
          .getJSONObject("employer_local_address")
          .get("district_or_city")
          .toString();
    String applicantEmployerState =
       applicant
          .getJSONObject("employment_check")
          .getJSONObject("employer_local_address")
          .get("state")
          .toString();
    String applicantEmployerPinCode =
       applicant
          .getJSONObject("employment_check")
          .getJSONObject("employer_local_address")
          .get("pincode")
          .toString();
    String applicantFinalEmployerAddress =
       applicantEmployerLocalAddress
          + ", "
          + applicantEmployerDistrictOrCity
          + ", "
          + applicantEmployerState
          + ", "
          + applicantEmployerPinCode;
    object.put("applicant" + (i + 1) + "EmployerAddress_hl",
applicantFinalEmployerAddress);
    Boolean applicant_26asAvailable =

applicant.getJSONObject("employment_check").getBoolean("26as_available");
```

```
    object.put("applicant" + (i + 1) + "Form26AS_hl", applicant_26asAvailable);
    String employmentClass =

applicant.getJSONObject("employment_check").get("employment_class").toString
();
    object.put("employmentType" + (i + 1) + "_app", employmentClass);
    String totalExperience =

applicant.getJSONObject("employment_check").get("total_experience").toString();
    String years = "", months = "";
    int totalMonths = 0;
    double totalYears;
    totalExperience = totalExperience.replaceAll("\\s", "");
    if (totalExperience.contains("Y")) {
     years = totalExperience.substring(0, totalExperience.indexOf("Y"));
    }
    if (totalExperience.contains("M")) {
     months =
        totalExperience.substring(
          totalExperience.indexOf("s") + 1, totalExperience.indexOf("M"));
    }
    totalMonths = (Integer.parseInt(years) * 12) + Integer.parseInt(months);
    totalYears = (double) totalMonths / 12;
    object.put("applicant" + (i + 1) + "workExperience_hl",
Math.round(totalYears));
   } catch (Exception e) {
    e.printStackTrace();
   }
   String applicantGender =
applicant.getJSONObject("aadhaar").get("gender").toString();
   object.put("applicant" + (i + 1) + "Gender_hl", applicantGender);
   String aadhaarName =
      CommonHelperFunctions.getStringValue(
        applicant.getJSONObject("aadhaar").get("aadhaar_name"));
   object.put("applicant" + (i + 1) + "AadhaarName_hl", aadhaarName);
   JSONArray applicantAadharDetails = new JSONArray();
   JSONObject applicantAadhar = new JSONObject();
   String applicantAadhaarName =
applicant.getJSONObject("aadhaar").get("aadhaar_name").toString();
   applicantAadhar.put("applicant" + (i + 1) + "AadharName_hl",
applicantAadhaarName);
   String applicantAadhaarNumber =
     applicant.getJSONObject("aadhaar").get("aadhaar_number").toString();
   applicantAadhar.put("applicant" + (i + 1) + "AadharNumber_hl",
applicantAadhaarNumber);
   applicantAadharDetails.put(applicantAadhar);
   object.put("applicant" + (i + 1) + "AadharDetails_hl", applicantAadharDetails);
   try {
    JSONArray applicantPanDetails = new JSONArray();
```

```
    JSONObject applicantPan = new JSONObject();
    String applicantPanName =
applicant.getJSONObject("pan").get("pan_name").toString();
    applicantPan.put("applicant" + (i + 1) + "PanName_hl", applicantPanName);
    String applicantPanNumber =
applicant.getJSONObject("pan").get("pan").toString();
    applicantPan.put("applicant" + (i + 1) + "PanNumber_hl",
applicantPanNumber);
    //     try {
    //       String applicantPanUrl =
applicant.getJSONObject("pan").get("pan_url");
    //       applicantPan.put("applicant"+(i+1)+"PanCard_hl", applicantPanUrl);
    //     } catch (Exception e) {
    //       e.printStackTrace();
    //     }
    applicantPanDetails.put(applicantPan);
    object.put("applicant" + (i + 1) + "PanDetails_hl", applicantPanDetails);
   } catch (Exception e) {
    e.printStackTrace();
   }
   JSONArray applicantAddressDetails = new JSONArray();
   JSONObject applicantAddress = new JSONObject();
   String applicantCurrentAddress =
      applicant.getJSONObject("current_address").get("address").toString();
   String applicantCurrentAddressDistrict =
      applicant.getJSONObject("current_address").get("district_or_city").toString();
   String applicantCurrentAddressState =
      applicant.getJSONObject("current_address").get("state").toString();
   String applicantCurrentAddressPinCode =
      applicant.getJSONObject("current_address").get("pincode").toString();
   String applicantFinalCurrentAddress =
      applicantCurrentAddress
         + ", "
         + applicantCurrentAddressDistrict
         + ", "
         + applicantCurrentAddressState
         + ", "
         + applicantCurrentAddressPinCode;
   applicantAddress.put("applicant" + (i + 1) + "CurrentAddress_hl",
applicantFinalCurrentAddress);
   String applicantPermanentAddress =
      applicant
         .getJSONObject("residential_verification")
         .getJSONObject("permanent_address")
         .get("address")
         .toString();
   String applicantPermanentAddressPinCode =
      applicant
         .getJSONObject("residential_verification")
```

```
            .getJSONObject("permanent_address")
            .get("pincode")
            .toString();
    String applicantFinalPermanentAddress =
        applicantPermanentAddress + ", " + applicantPermanentAddressPinCode;
    applicantAddress.put(
        "applicant" + (i + 1) + "PermanentAddress_hl",
applicantFinalPermanentAddress);
    String applicantCurrentAddressVintage =
        applicant
            .getJSONObject("residential_verification")
            .get("current_address_vintage")
            .toString();
    String years = "", months = "";
    int totalMonths = 0;
    double totalYears;
    applicantCurrentAddressVintage =
applicantCurrentAddressVintage.replaceAll("\\s", "");
    if (applicantCurrentAddressVintage.contains("Y")) {
      years =
          applicantCurrentAddressVintage.substring(0,
applicantCurrentAddressVintage.indexOf("Y"));
    }
    if (applicantCurrentAddressVintage.contains("M")) {
      months =
          applicantCurrentAddressVintage.substring(
              applicantCurrentAddressVintage.indexOf("s") + 1,
              applicantCurrentAddressVintage.indexOf("M"));
    }
    totalMonths = (Integer.parseInt(years) * 12) + Integer.parseInt(months);
    totalYears = (double) totalMonths / 12;
    applicantAddress.put(
        "applicant" + (i + 1) + "YearsAtCurrentResidency_hl",
Math.round(totalYears));
    applicantAddressDetails.put(applicantAddress);
    object.put("applicant" + (i + 1) + "AddressDetails_hl", applicantAddressDetails);
    String applicantDependentsCounts =

applicant.getJSONObject("residential_verification").get("dependents_counts").toSt
ring();
    object.put("applicant" + (i + 1) + "DependentsCount_hl",
applicantDependentsCounts);
    object.put("applicant" + (i + 1) + "Dependents_hl", applicantDependentsCounts);
    String applicantJobNature =

applicant.getJSONObject("residential_verification").get("job_nature").toString();
    object.put("applicant" + (i + 1) + "JobNature_hl", applicantJobNature);
    String applicantEducation =
```

```
applicant.getJSONObject("residential_verification").get("education").toString();
   object.put("applicant" + (i + 1) + "EducationQualification_hl",
applicantEducation);
   object.put("applicant" + (i + 1) + "currentresidenceinYears_hl",
Math.round(totalYears));

   //     JSONArray applicantSalarySlipDetails1 = new JSONArray();
   //     JSONObject applicantSalarySlip = new JSONObject();
   //     String applicantPayslipsUrl =
applicant.getJSONObject("payslips").get("url");
   //     applicantSalarySlip.put("applicant"+(i+1)+"SalarySlip_hl",
applicantPayslipsUrl);
   //     applicantSalarySlipDetails1.put(applicantSalarySlip);
   //     object.put("applicantSalarySlipDetails1_hl", applicantSalarySlipDetails1);
   logger.info("ObjectForApplicant" + (i + 1) + " :- {}", object);

   try {
    if (applicantType.equalsIgnoreCase("Non-Financial Co-Applicant")) {
     object.put("isApplicant" + (i + 1) + "NF", "yes");
    } else {
     object.put("isApplicant" + (i + 1) + "NF", "no");
    }
   } catch (Exception e) {
   }

   return object;
  }
}
```

- **Initiate Disbursal Service Class**

```
package com.kuliza.workbench.service;

import com.kuliza.lending.common.pojo.ApiResponse;
import com.kuliza.lending.common.utils.CommonHelperFunctions;
import com.kuliza.workbench.util.CapriUtilService;
import org.flowable.cmmn.api.CmmnRuntimeService;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;

import java.text.DecimalFormat;
```

```
import java.util.Map;

@Service
public class InitiateDisbursalService {
  private static final Logger logger =
LoggerFactory.getLogger(InitiateDisbursalService.class);
  private static final DecimalFormat df = new DecimalFormat("0.00");

  @Autowired CmmnRuntimeService cmmnRuntimeService;
  @Autowired CapriUtilService capriUtilService;

  public ApiResponse initiate(Map<String, Object> request) throws
JSONException {

    logger.info("Inside multi tranche table.....");
    String caseInstanceId = request.get("caseInstanceId").toString();
    ;
    double disbursedPercentageTillDateSummary = 0;
    String disbursalAmountSummaryString =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId,
"disbursalAmountSummary_hl"));
    logger.info("disbursalAmountSummary in string.....:- {}",
disbursalAmountSummaryString);
    JSONArray disbursalAmountSummary = new JSONArray();
    try {
      disbursalAmountSummary = new
JSONArray(disbursalAmountSummaryString);
    } catch (Exception e) {
    }

    logger.info("disbursalAmountSummary_hl.....:- {}",
disbursalAmountSummary);
    JSONObject object = new JSONObject();
    String recommendedStageOfConstruction =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId,
"recommendedStageOfConstruction_hl"));
    String currentTranchePercent =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId,
"currentTranchePercent_hl"));
    logger.info("currentTranchePercent.....:- {}", currentTranchePercent);
    String finalLoanAmount =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId, "finalLoanAmount"));
    logger.info("finalLoanAmount.....:- {}", finalLoanAmount);
    logger.info("recommended Stage Of Construction.....:- {}",
recommendedStageOfConstruction);
```

```
   object.put("recommendedStageOfConstSummary_hl",
recommendedStageOfConstruction);
   object.put("disbursalDate_hl", capriUtilService.getCurrentDate("dd-MM-
yyyy"));
   double sumAmountDisbursed =
      (Double.parseDouble(currentTranchePercent) / 100) *
Integer.parseInt(finalLoanAmount);
   object.put("disbursedAmountSummary_hl", sumAmountDisbursed);
   logger.info("sumAmountDisbursed 1st.....:- {}", sumAmountDisbursed);

   if (!disbursalAmountSummaryString.equalsIgnoreCase("")) {
    logger.info("Inside if condition.....");
    for (int i = 0; i < disbursalAmountSummary.length(); i++) {
     String amountDisbursed =
        CommonHelperFunctions.getStringValue(
          cmmnRuntimeService.getVariable(caseInstanceId,
"disbursedAmountSummary_hl"));
      sumAmountDisbursed = sumAmountDisbursed +
Integer.parseInt(amountDisbursed);
    }
   }
   logger.info("sumAmountDisbursed.....:- {}", sumAmountDisbursed);

   disbursedPercentageTillDateSummary =
      sumAmountDisbursed / (Double.parseDouble(finalLoanAmount) * 100);
   logger.info(
      "disbursedPercentageTillDateSummary.....:- {}",
disbursedPercentageTillDateSummary);

   object.put("disbursedAmountTillDateSummary_hl", sumAmountDisbursed);
   logger.info("after amount disbursed.....");

   try {
    object.put(
       "disbursedPercentageTillDateSummary_hl",
df.format(disbursedPercentageTillDateSummary));
   } catch (Exception e) {
   }
   logger.info("after percentage.....");

   try {
    object.put(
       "amountPendingDisbursedSummary_hl",
       Integer.parseInt(finalLoanAmount) - sumAmountDisbursed);
   } catch (Exception e) {
   }
   logger.info("after pending summary.....:- {}", object);

   disbursalAmountSummary.put(object);
```

```
    logger.info("disbursalAmountSummary_hl after population.....:- {}",
disbursalAmountSummary);
    logger.info("after everything.....");

    cmmnRuntimeService.setVariable(
        caseInstanceId, "disbursalAmountSummary_hl",
disbursalAmountSummary.toString());
    Map<String, Object> res =
CommonHelperFunctions.getHashMapFromJsonString(disbursalAmountSummary.
toString());

    return new ApiResponse(HttpStatus.OK, "Success", res);
  }
}
```

- **Insurance Api class**

```
package com.kuliza.workbench.service;

import com.kuliza.lending.common.pojo.ApiResponse;
import com.kuliza.lending.common.utils.CommonHelperFunctions;
import com.kuliza.workbench.util.IbHelper;
import com.mashape.unirest.http.exceptions.UnirestException;
import org.flowable.cmmn.api.CmmnRuntimeService;
import org.flowable.common.engine.impl.util.Financials;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;

import java.text.DecimalFormat;
import java.util.HashMap;
import java.util.Map;

@Service
public class InsurancePremiumService {
  private static final Logger logger =
LoggerFactory.getLogger(InsurancePremiumService.class);
  private static final DecimalFormat df = new DecimalFormat("0.00");

  @Autowired CmmnRuntimeService cmmnRuntimeService;
  @Autowired IbHelper ibHelper;
  @Autowired FinalLoanAmountService finalLoanAmountService;
```

```
  public void calculatePremium(Map<String, Object> request) throws
JSONException, UnirestException {
    String caseInstanceId = request.get("caseInstanceId").toString();
    String loanAmount =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId, "loanAmount_hl"));
    logger.info("Loan Amount from final submit:- {}", loanAmount);
    int insurancePremium = (int) (0.036 * Integer.parseInt(loanAmount));
    cmmnRuntimeService.setVariable(caseInstanceId, "insurancePremium_hl",
insurancePremium);
    finalLoanAmountService.calcFinalLoanAmount(request);
    String rate =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId, "updatedROI"));
    String finalLoanAmount =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId, "finalLoanAmount"));
    String loanTenure =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId, "loanTenure_hl"));
    logger.info("final ROI Maximum Capping :- {}", rate);
    logger.info("finalLoanAmount :- {}", finalLoanAmount);
    logger.info("loanTenure :- {}", loanTenure);
    double rateD = Double.parseDouble(rate);
    rateD = rateD / 100;
    logger.info("final ROI Maximum Capping D :- {}", rateD);
    double finalLoanAmountD = Double.parseDouble(finalLoanAmount);
    finalLoanAmountD = -(finalLoanAmountD);
    logger.info("finalLoanAmountD :- {}", finalLoanAmountD);
    int loanTenureD = Integer.parseInt(loanTenure);
    logger.info("loanTenureD :- {}", loanTenureD);
    double pmt = Financials.pmt(rateD, loanTenureD, finalLoanAmountD, 0, 1);
    cmmnRuntimeService.setVariable(caseInstanceId, "emi_hl", df.format(pmt));
    logger.info("PMT VALUE :- {}", pmt);

  }

  private Map<String, Object> payload(String caseInstanceId) throws
JSONException {
    String dob =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId, "applicant1Dob_hl"));
    String loanDisbDate =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId, "loanDisbDate"));
    String loanAmount =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId, "loanAmount_hl"));
```

```
    String policyTerm =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId, "policyTerm"));
    String planId =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId, "planId"));
    String gender =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId,
"applicant1Gender_hl"));
    String sumAssured =
        CommonHelperFunctions.getStringValue(
            cmmnRuntimeService.getVariable(caseInstanceId, "sumAssured"));
    Map<String, Object> payload = new HashMap<>();
    payload.put("dob", "06/19/1988");
    payload.put("loanDisbDate", "03/03/2021");
    payload.put("originalLoanAmount", 180000);
    payload.put("policyTerm", 2);
    payload.put("planId", "012");
    payload.put("gender", "Male");
    JSONArray jsonArray = new JSONArray();
    JSONObject jsonObject = new JSONObject();
    jsonObject.put("sumAssured", 50000);
    jsonArray.put(jsonObject);
    payload.put("productDataList", jsonArray);

    return payload;
  }

  public ApiResponse checkStatus(Map<String, String> request) {

    return new ApiResponse(HttpStatus.OK, "success");
  }
}
```

Chapter 6 TESTING

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

This tutorial will give you a basic understanding on software testing, its types, methods, levels, and other related terminologies. (tutorials point, n.d.)

6.1 LEVELS OF TESTING

There are different levels during the process of testing. Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are −

- Functional Testing
- Non-functional Testing.

Functional Testing:

This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of a software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

Non-functional Testing:

This section is based upon testing an application from its non-functional attributes. Non-functional testing involves testing a software from the requirements which are non-functional in nature but important such as performance, security, user interface, etc.

6.2 STRATEGIC APPROACH TO SOFTWARE TESTING

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behaviour, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implement in source code. Testing progress by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.

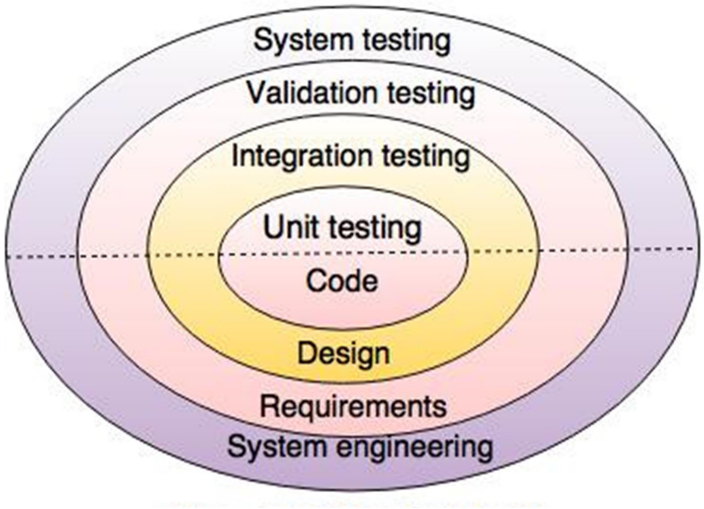


Fig. 5 Testing Strategy

6.3 UNIT TESTING

Each module is considered independently. It focuses on each unit of software as implemented in the source code. It is white box testing.

Steps in the software testing:

The steps involved during unit testing are as follows:

- Preparation of the test cases.
- Preparation of the possible test data with all the validation checks.
- Complete code review of the module.
- Actual testing done manually.
- Modifications done for the errors found during testing.
- Prepared the test result scripts.

The unit testing done included the testing of the following items:

- Functionality of the entire module.
- Validations for user input.
- Checking of the coding standards to be maintained during coding.
- Testing the module with all the possible test data.
- Testing of the functionality involving all type of calculations etc.
- Commenting standard in the source files.

After completing the unit testing of all the modules, the whole system is integrated with all its dependencies in that module. While system integration, we integrated the modules one by one and tested the system at each step. This helped in reduction of errors at the time of the system testing.

6.4 INTEGRATION TESTING

Integration testing aims at constructing the program structure while at the same constructing tests to uncover errors associated with interfacing the modules. Modules are integrated by using the top down approach.

6.5 VALIDATION TESTING

Validation testing was performed to ensure that all the functional and performance requirements are met.

6.6 SYSTEM TESTING

It is executing programs to check logical changes made in it with intention of finding errors. A system is tested for online response, volume of transaction, recovery from failure, etc. System testing is done to ensure that the system satisfies all the user requirements.

The steps involved during system testing are as follows:

- Integration of all the modules in the system.
- Preparation of all test cases.
- Preparation of the possible test data with all the validation checks.
- Actual testing done manually.
- Recording of all the reproduced errors.
- Modifications done for the errors found during testing.
- Prepared the test result scripts after rectification of the errors.

The system testing done included the testing of the following items:

- Functionality of the entire system as whole.
- User interface of the system.
- Testing the dependent modules together with all the possible test data scripts.
- Verification and validation testing.
- Testing the reports with all its functionalities.

After the completion of the system testing, the next following phase was the Acceptance testing. Thus, we reached the final phase of the project.

6.7 WHITE BOX TESTING

In this technique, the close examination of the logical parts through the software are tested by cases that exercise species sets of conditions. All logical parts of the software checked once. Errors that can be corrected using this techniques are typographical errors, logical expressions which should be executed once may be getting executed more than once and error resulting by using wrong controls. When the white box testing tests all the independent part within a module a logical decions on their true and false side are exercised, all bounds within their operational bounds were exercised and internal data structure to ensure their validity were exercised once.

6.8 BLACK BOX TESTING

This method enables the software engineer to device sets of input techniques that fully exercise all functional requirements for a program. Black box testing tests the input, the output and the external data. It checks whether the input data is correct and whether we are getting the desired output.

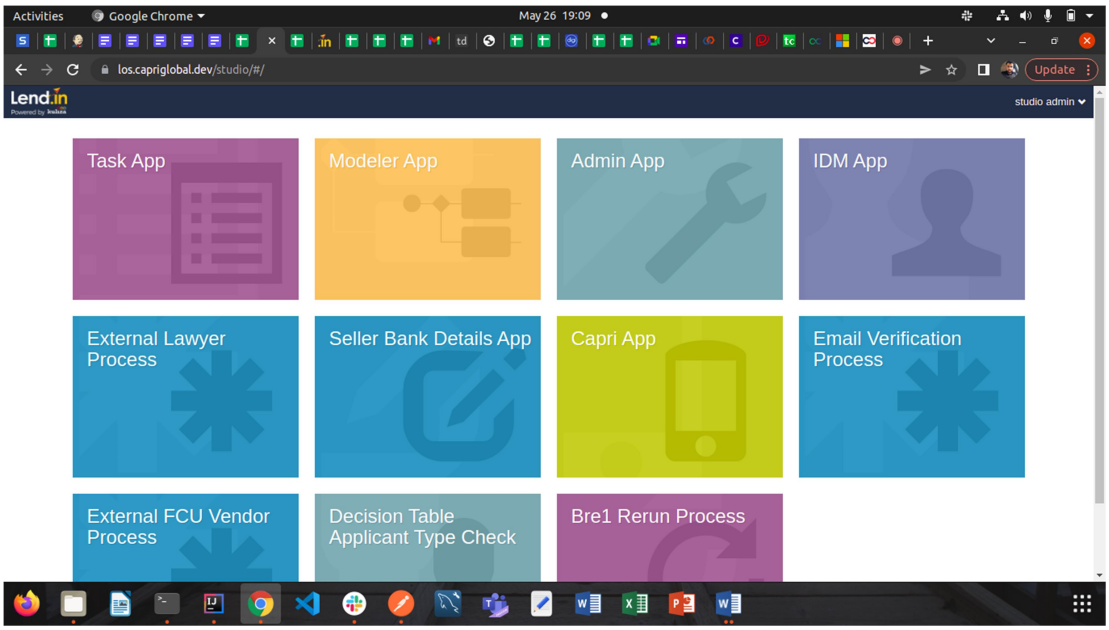
6.9 OTHER TYPES OF TESTING

- **Peak Load Test:** It determines whether the system will handle the volume of activities that occur when the system is at the peak of its processing demand. For example, test the system by activating all terminals at the same time.
- **Storage Testing:** It determines the capacity of the system to store transaction data on a disk or in other files.
- **Performance Time Testing:** It determines the length of the time system used by the system to process transaction data. This test is conducted prior to implementation to determine how long it takes to get a response to an inquiry, make a backup copy of a file, or send a transmission and get a response.
- **Recovery Testing:** This testing determines the ability of user to recover data or re-start after failure. For example, load backup copy of data and resume processing without data or integrity loss.
- **Procedure Testing:** It determines the clarity of documentation on operation and usues of system by having users do exactly what manual requests. For example, powering down system at the end of week or responding to paper-out light on printer.
- **Human Factors Testing:** It determines how users will use the system when processing data or preparing reports.

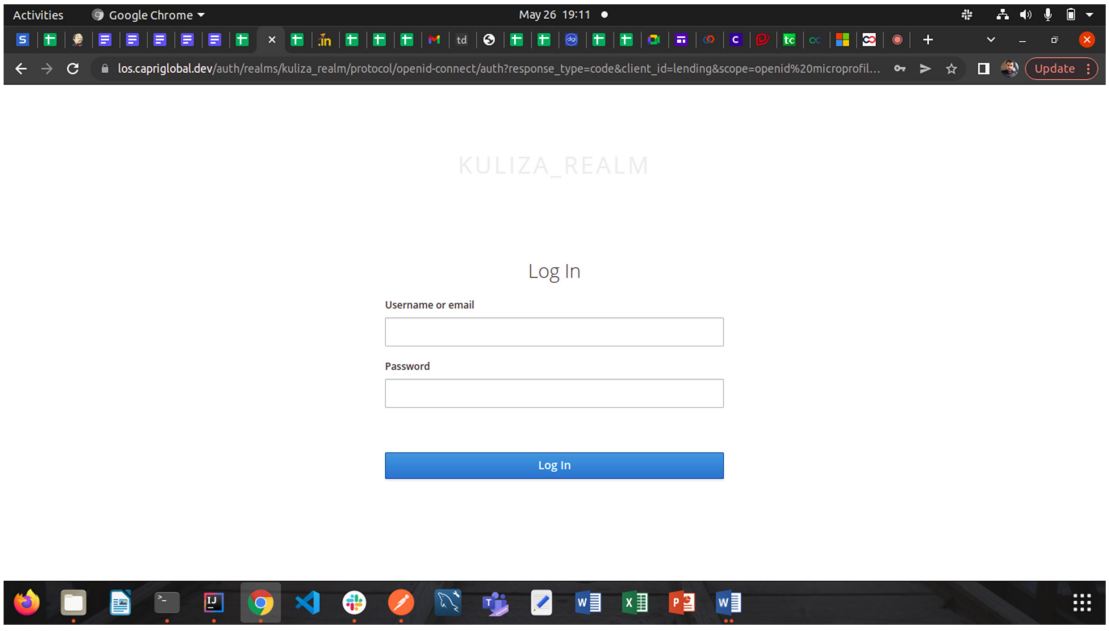
Chapter 7 IMPLEMENTATION AND MAINTENANCE

7.1 PROJECT SNAPSHOTS

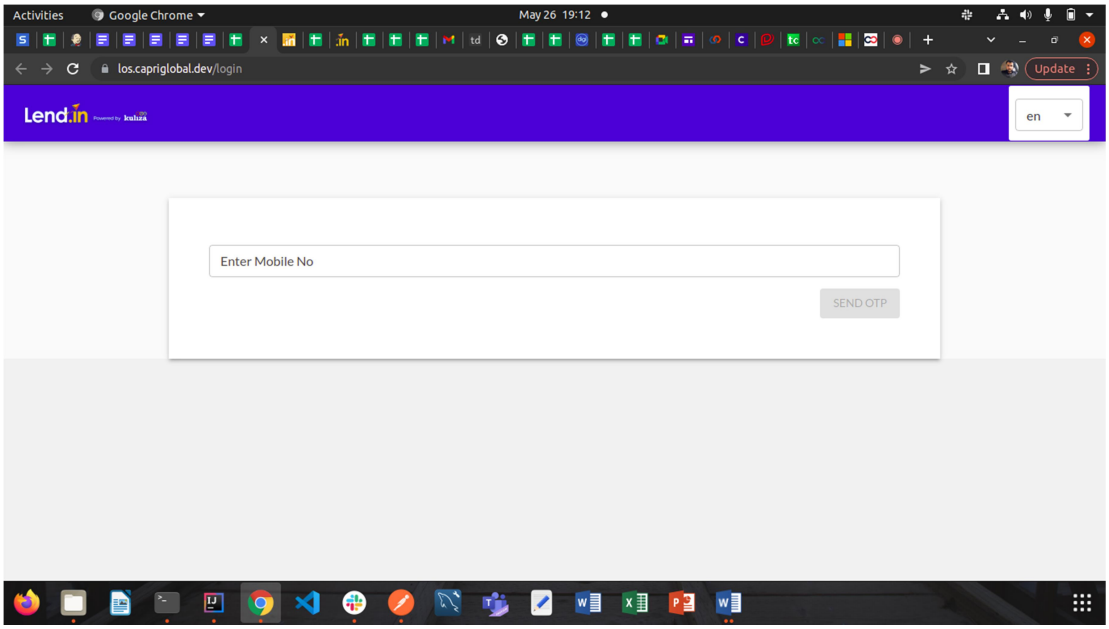
- **STUDIO**



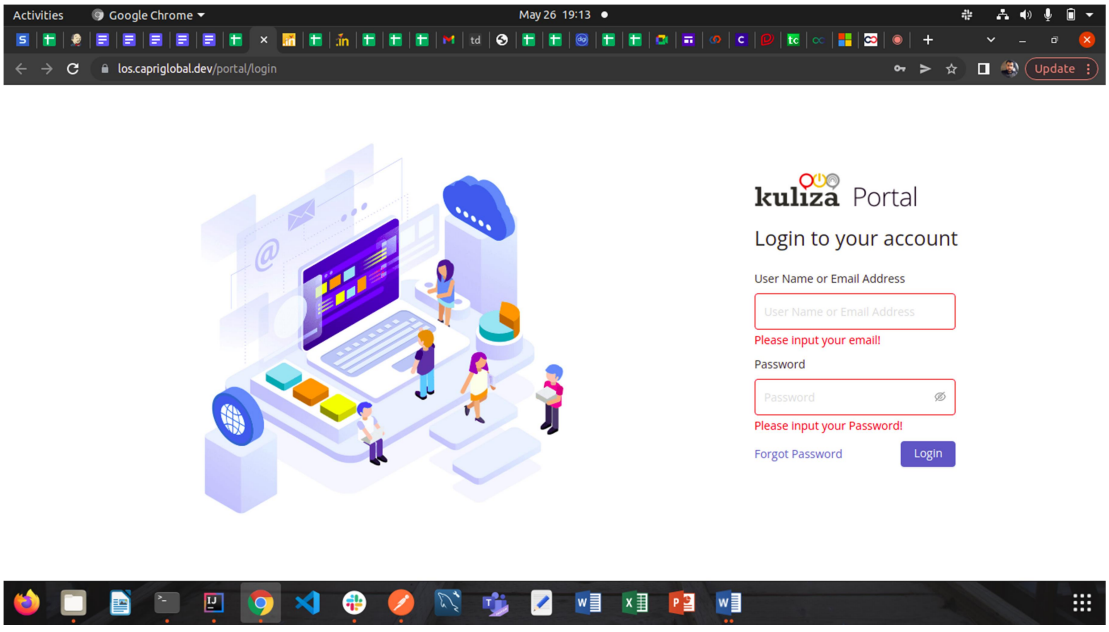
- **ADMIN LOGIN**



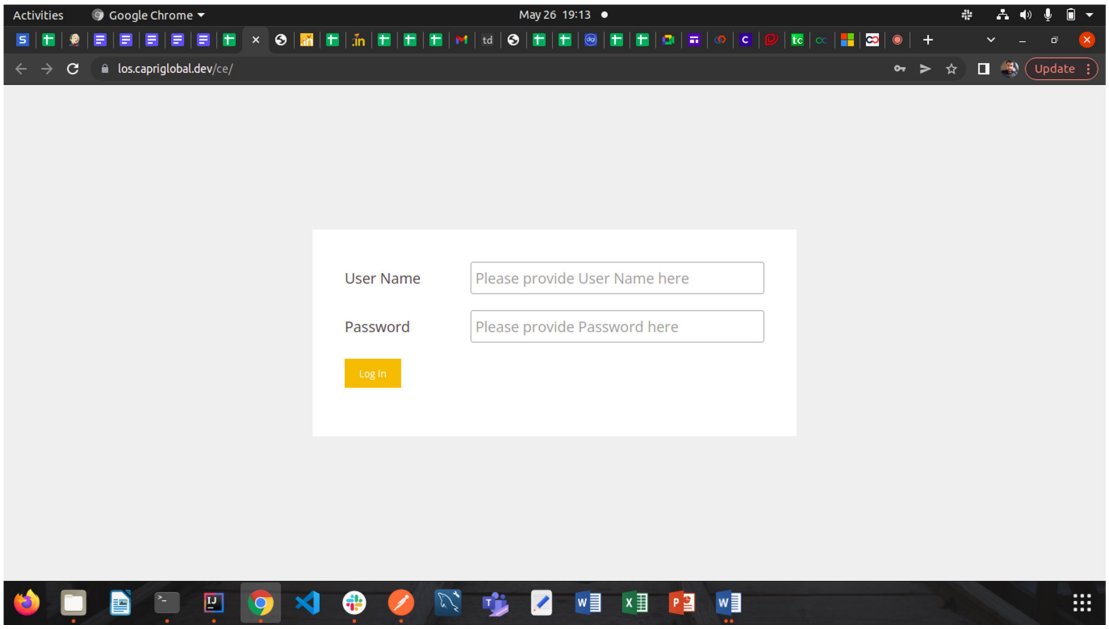
- **RENDERING ENGINE LOGIN**



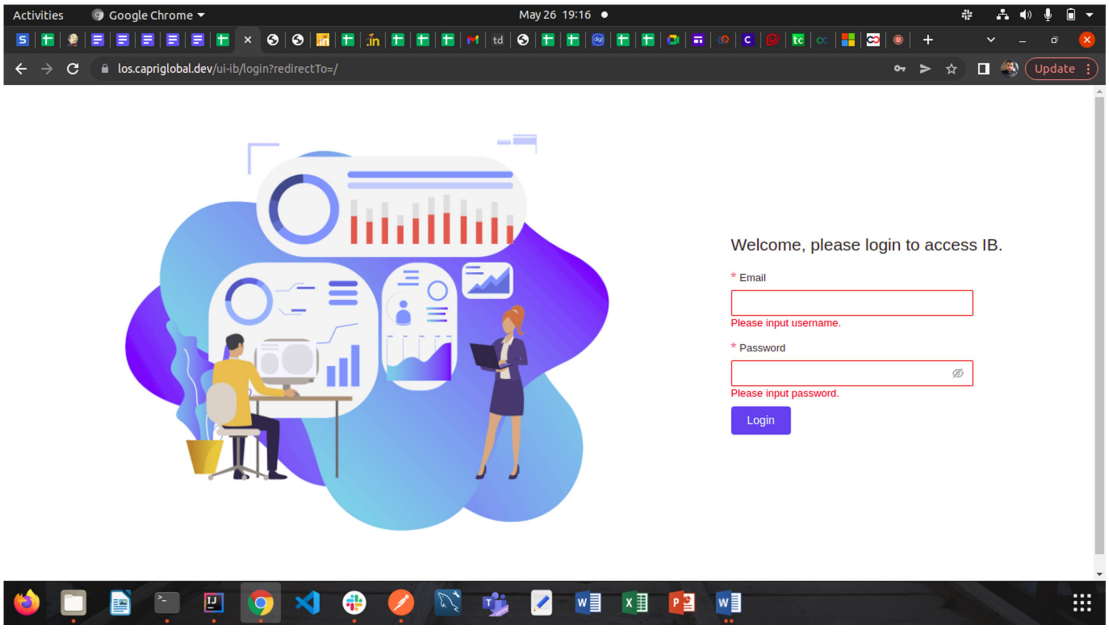
- **PORTAL LOGIN**



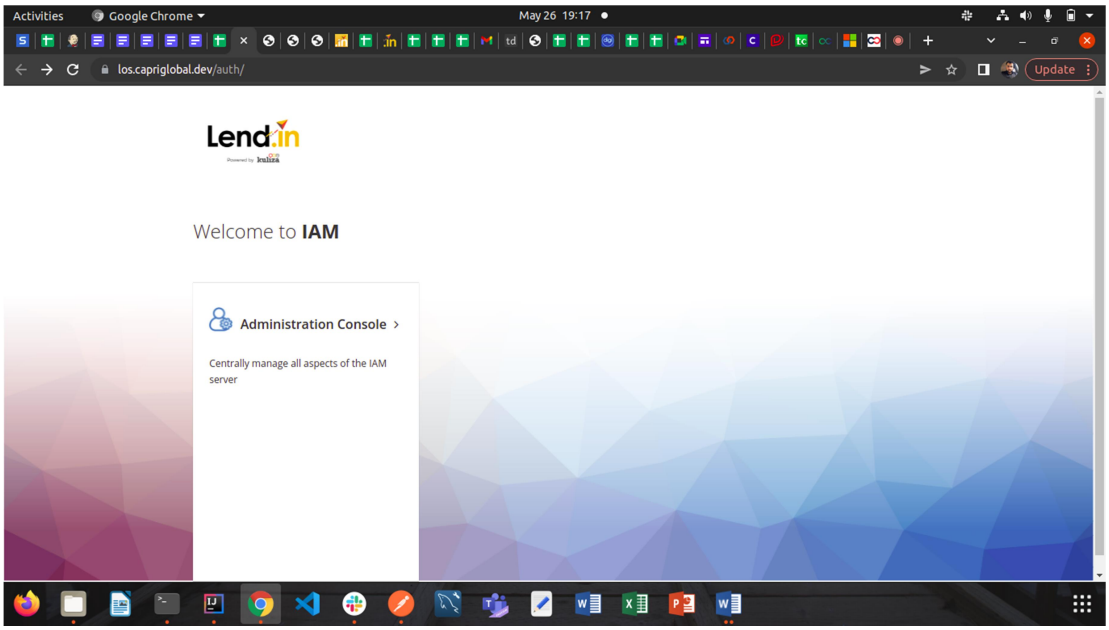
- **CREDIT ENGINE LOGIN**



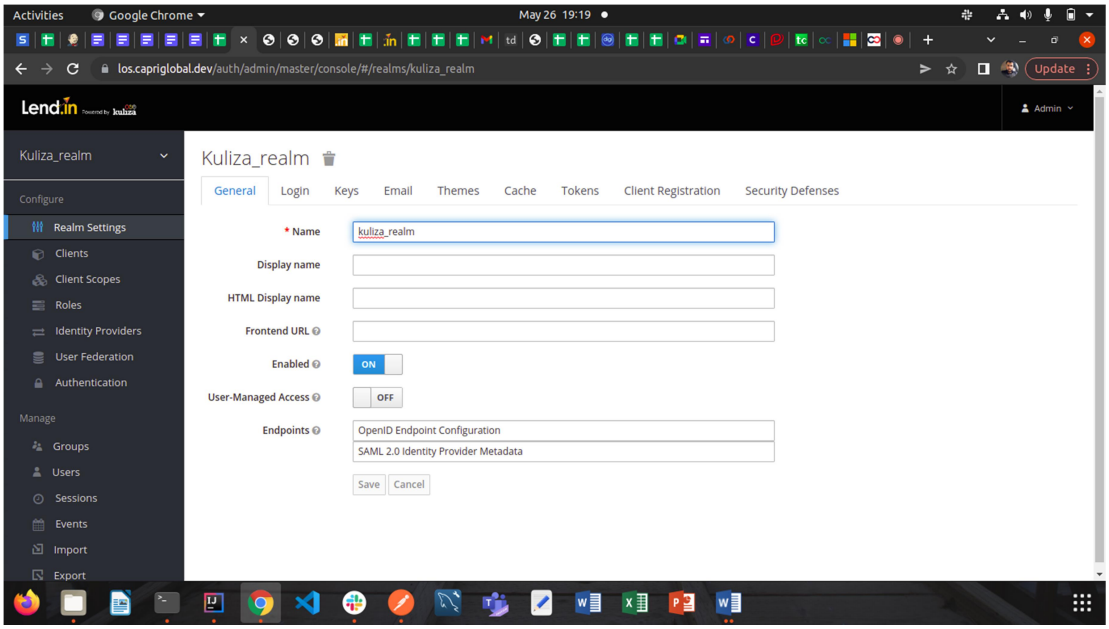
- **INTEGRATION BROKER LOGIN**



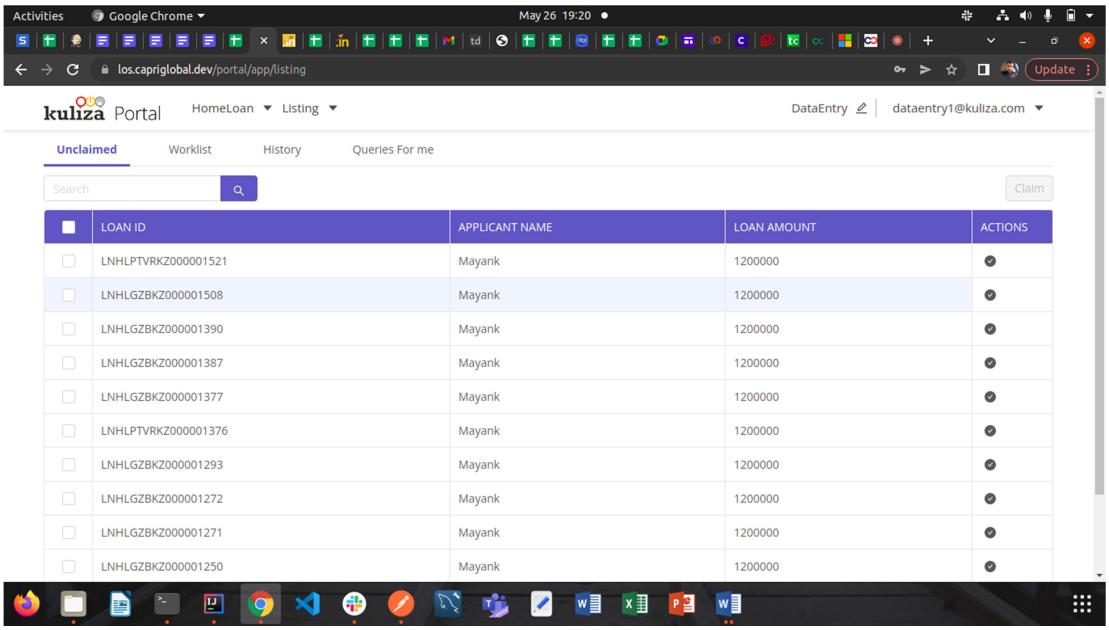
- **IDENTITY ACCESS MANAGEMENT**



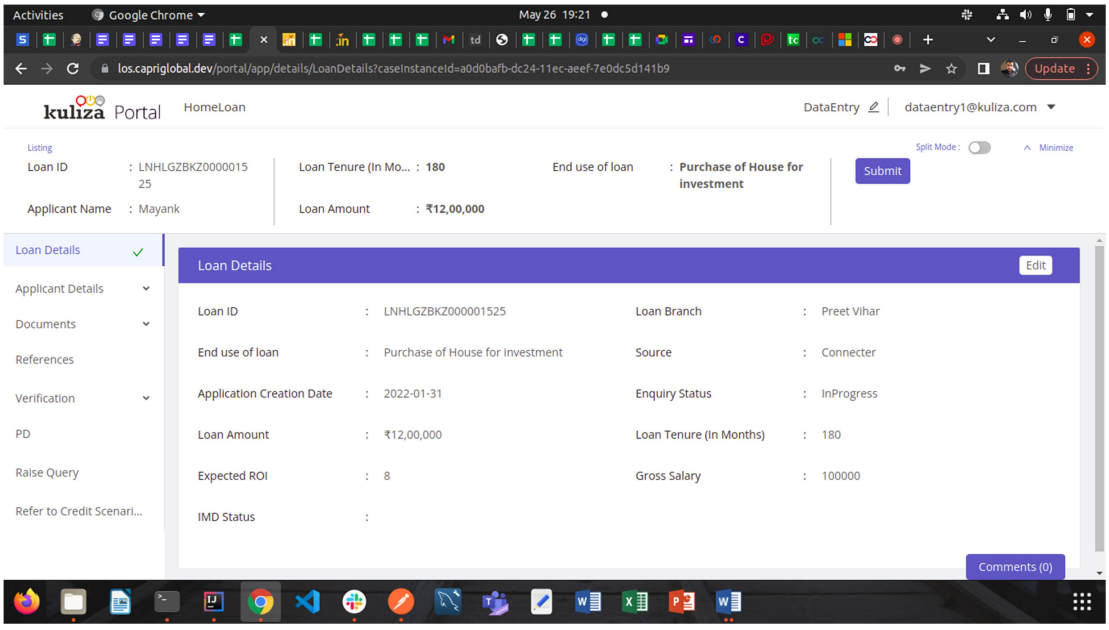
- **IAM AFTER LOGIN**



- **DATA ENTRY AFTER LOGIN**



- **APPLICANTS DETAILS IN DATA ENTRY ROLE**



7.2 MAINTENANCE

Software Maintenance is the process of modifying a software product after it has been delivered to the customer. The main purpose of software maintenance is to modify and update software application after delivery to correct faults and to improve performance. Software is a model of the real world. When the real world changes, the software require alteration wherever possible.

Software Maintenance is an inclusive activity that includes error corrections, enhancement of capabilities, deletion of obsolete capabilities, and optimization.

Software maintenance is widely accepted part of SDLC now a days. It stands for all the modifications and updations done after the delivery of software product. There are number of reasons, why modifications are required, some of them are briefly mentioned below:

- **Market Conditions** - Policies, which changes over the time, such as taxation and newly introduced constraints like, how to maintain bookkeeping, may trigger need for modification.
- **Client Requirements** - Over the time, customer may ask for new features or functions in the software.
- **Host Modifications** - If any of the hardware and/or platform (such as operating system) of the target host changes, software changes are needed to keep adaptability.
- **Organization Changes** - If there is any business level change at client end, such as reduction of organization strength, acquiring another company, organization venturing into new business, need to modify in the original software may arise.

7.3 NEED FOR MAINTENANCE

Software Maintenance is needed for:-

- Correct errors
- Change in user requirement with time
- Changing hardware/software requirements
- To improve system efficiency
- To optimize the code to run faster
- To modify the components
- To reduce any unwanted side effects.

Thus the maintenance is required to ensure that the system continues to satisfy user requirements. (geeksforgeeks, n.d.)

7.4 TYPES OF MAINTENANCE

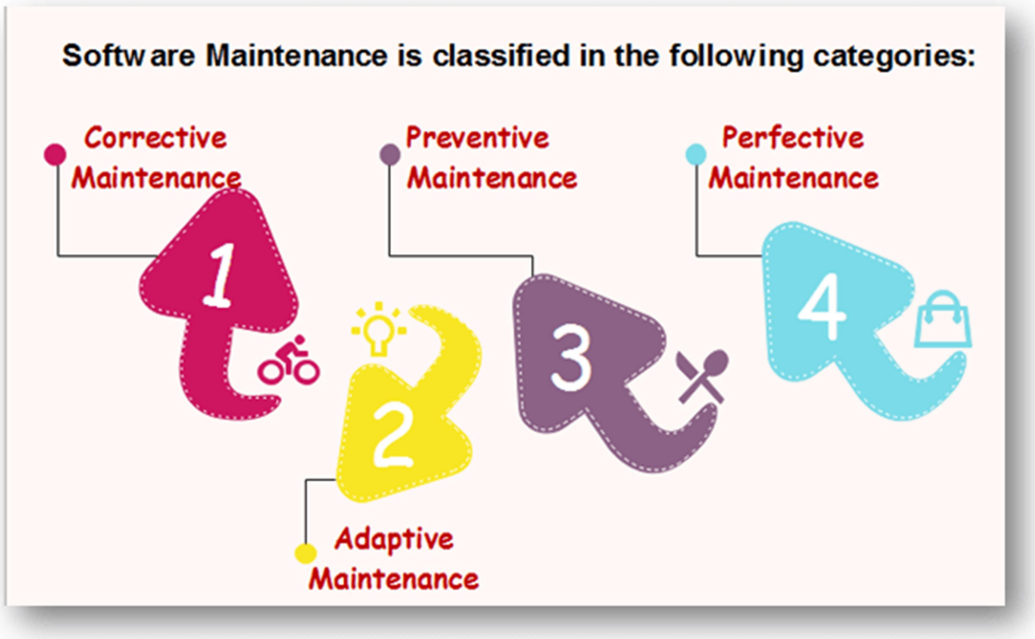


Fig. 6 Types of Maintenance

In a software lifetime, type of maintenance may vary based on its nature. It may be just a routine maintenance tasks as some bug discovered by some user or it may be a large event in itself based on maintenance size or nature. Following are some types of maintenance based on their characteristics:

- **Corrective Maintenance** - This includes modifications and updations done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.
- **Adaptive Maintenance** - This includes modifications and updations applied to keep the software product up-to date and tuned to the ever changing world of technology and business environment.
- **Perfective Maintenance** - This includes modifications and updates done in order to keep the software usable over long period of time. It includes new features, new user requirements for refining the software and improve its reliability and performance.
- **Preventive Maintenance** - This includes modifications and updations to prevent future problems of the software. It aims to attend problems, which are not significant at this moment but may cause serious issues in future.

Chapter 8 CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

Our project is only a humble venture to satisfy the needs to manage the project work. Several user friendly coding have also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the NBFC. The objective of software planning is to provide a framework that enables the manager to make reasonable estimates made within a limit time frame at the beginning of the software project and should be updated regularly as the project progresses.

At the end it is concluded that we have made effort on following points:

- A description of the background and context of the project and its relation to work already done in the area.
- Made statement of the aims and objectives of the project.
- The description of purpose, scope and applicability.
- We define the problem on which we are working in the project.
- We describe the requirement specifications of the system and the actions that can be done on these things.
- We understand the problem domain and produce a model of the system, which describes operations that can be performed on the system.
- We included features and operations in detail, including screen layouts.
- We designed user interface and security issues related to system.
- Finally the system is implemented and tested according to test cases.

Benefits:

The project is identified by the merits of the system offered to the user. The merits of this project are as follows:

- It is a web enabled project.
- This project offers user to enter the data through simple and interactive forms. This is very helpful for applicants to enter the all the information through so much simplicity.

- User is provided the option of monitoring the records he entered earlier. He can see the desired records with the variety of options provided by him.
- Data storage and retrieval will become faster and easier to maintain because data is stored in a systematic manner and in a single database.
- Decision making process would be greatly enhanced because of faster processing of information since data collection from information available on computer takes much less time then manual system.
- Allocating of sample results becomes much faster because at a time the user can see the records of last years.
- Easier and faster data transfer through latest technology associate with the computer and communication.
- Through these features it will increase the efficiency, accuracy and transparency.

8.2 FUTURE SCOPE

In a nutshell, it can be summarized that the future scope of the project circles around maintaining informations regarding:

- We can give more advance software for loan origination system including more facilities.
- Integrate multiple load balancers to distribute the load of the system.
- Create the master and slave database structure to reduce the overload of the database queries.
- Implement the backup mechanism for taking backup of codebase and database on regular basis on different servers.

The above mentioned points are the enhancements which can be done to increase the applicability and usage of this project. Here we can maintain the records of applicants and their applications. Also, as it can be seen that nowadays the players are versatile, i.e, so there is a scope for introducing a method to maintain the campus LOS. Enhancement can be done to maintain all the roles including admin, CML, dataentry, legal head, technical vendor, etc.

We have left all the options open so that if there is any other future requirement in the system by the user for the enhancement of the system then it is possible to implement them. In the last, we would like to thank all the persons involved in the development of the system directly or indirectly. We hope that the project will serve its purpose for which it is develop there by underlining success of process.

BIBLIOGRAPHY

https://www.baeldung.com/spring-boot

Achyut Godbole. (2019). *Web Technologies.* chennai: McGraw-Hill.

Paul Holmes-Higgin (2017-04-20). "BPMNext 2017: Making Business Processes Dance". BPMNext. Retrieved 2017-05-09.

"Rajesh Kumar - CEO/Managing Director, TransUnion CIBIL Ltd". Bloomberg. Retrieved 24 May 2022.

https://www.geeksforgeeks.org

https://www.jetbrains.com/idea/

testing. (n.d.). Retrieved from tutorials point: https://www.tutorialspoint.com

https://www.tutorialspoint.com/software_engineering/software_maintenance_overview.htm

https://www.w3schools.com/java/

https://www.mysql.com/downloads/

https://www.w3schools.com/mySQl/default.asp

https://www.flowable.com/open-source/docs/oss-introduction

RESEARCH PAPER (REFERENCES)

1. https://www.researchgate.net/publication/318131748_An_Overview_of_Blockchain_Technology_Architecture_Consensus_and_Future_Trends/link/59d71faa458515db19c915a1/download

2. https://www.researchgate.net/publication/228742449_New_technologies_for_web_development

3. https://www.researchgate.net/publication/49777804_Automatically_Activated_Attitudes_as_Mechanisms_for_Message_Effects_The_Case_of_Alcohol_Advertisements?_sg%5B0%5D=M9HUUf1ygFL93tUDHhMYCU-ddnds1WmPqkDbkloIcZIF_rs1qlNPtEzHM_a-LZkbbA0WPNb8Ff3NDTUMWzuhv5OuWQ.yK-VDZy--aSgW4Tdeg1ZH_YB1QjB4McCuoPSoXI-YJah5F2ITZGNuLYD8ubAuwT7-4SH1qs5d11g3qBpxD9J0Q

4. https://www.researchgate.net/publication/265336387_Maintaining_the_Correctness_of_Transactional_Memory_Programs

5. https://www.researchgate.net/publication/258879322_The_implementation_of_the_intelligence_simulation_system_for_automated_homes

6. https://www.researchgate.net/publication/36192738_Utilization_of_Web_services_to_improve_communication_of_operational_information

7. https://www.researchgate.net/publication/331429981_React_Native_Application_Development

8. https://www.researchgate.net/publication/338497042_Prospects_for_Using_React_Native_for_Developing_Cross-platform_Mobile_Applications

9. https://www.researchgate.net/publication/224132834_It's_about_Time_to_Take_JavaScript_More_Seriously

10. https://www.researchgate.net/publication/221321024_Automated_Construction_of_JavaScript_Benchmarks

11. https://www.researchgate.net/publication/328906451_Research_Paper_Static_JavaScript_Call_Graphs_A_Comparative_Study

12. https://cs.brown.edu/~sk/Publications/Papers/Published/gsk-essence-javascript/paper.pdf

13. https://www.researchgate.net/publication/344153528_JavaScript_How_we_got_here_An_indepth_history

14. https://www.researchgate.net/publication/352551347_Internal_Quality_Evolution_of_Open-Source_Software_Systems

15. https://www.researchgate.net/publication/342232789_What_Java_Developers_have_talked_about_An_empirical_study_on_Stack_Overflow

16. https://www.researchgate.net/publication/319890267_What_are_Software_Engineers_asking_about_Android_Testing_on_Stack_Overflow

17. https://www.researchgate.net/publication/354638969_An_empirical_study_of_COVID-19_related_posts_on_Stack_Overflow_Topics_and_technologies

18. https://www.researchgate.net/publication/337675610_Testing_tools_for_Android_context-aware_applications_a_systematic_mapping

19. https://www.researchgate.net/publication/351088042_Java_Programming_Language_Report

20. https://www.researchgate.net/publication/254008906_On_the_Analysis_of_Cascading_Style_Sheets

21. https://www.researchgate.net/publication/353403841_Online_Food_Ordering_Management_System

22. https://www.researchgate.net/publication/306279796_Make_Your_Own_Web_App_A_Massive_Open_Online_Course

23. https://www.researchgate.net/publication/267936252_Applying_Formal_Concept_Analysis_to_Cascading_Style_Sheets

24. https://www.researchgate.net/publication/233145547_HTML_for_Beginners_Public_Services_Emphasis

25. https://www.researchgate.net/publication/267936252_Applying_Formal_Concept_Analysis_to_Cascading_Style_Sheets