(DHANKAWADI, PUNE-43)



MINI PROJECT REPORT ON

Insurance Policy Management Portal for Customers, Agents & Administrators

SUBMITTED BY

**Sharan Rajani** (3961)

**Soham Ray** (3972)

**Jay Thanvi** (3976)

**Shreyas Waghmare** (3977)

(Third Year Information Technology)

UNDER THE GUIDANCE OF

Prof. S. A. Jakhete

Submitted To

**Pune Institute of Computer Technology**

**For Academic Year 2017-18**

|  |  |
| --- | --- |
| **TITLE** | **PAGE NO** |
| **1. Introduction** | **3** |
| **2. Scope** | **3** |
| **3. Modules** | **4** |
| **4. E-R Diagram (description)** | **5** |
| **5. Coding -Procedure and Triggers, (description)** | **6** |
| **5. Coding design (Important Function)** | **9** |
| **6. User Interface Snapshots** | **11** |
| **7. Conclusion** | **17** |

# INTRODUCTION

Problem Statement:

To make an Online Insurance Policy Management Portal for both Customers and Agents. Customers can review their policies, and make premium payments at their convenience. Agents can add new customers and choose from a variety of other functionalities.

# SCOPE

1. Customers may review their existing policies and their upcoming deadlines.
2. Customer is reminded to pay his premium at the earliest if he/she is fast approaching/already past the deadline.
3. Customers may update their personal details and surrender their account as per their wishes.
4. Customer may view all the available policies.
5. Customer may review all past invoices.
6. Customers may view their agents details.
7. Customer summary shows the customer all the important details per policy including maturity date, maturity amount and amount on surrender.
8. Agents are provided with a list of their customers.
9. Agents are provided with the functionality to add new customers or new policies for existing customers.
10. Agents can make payments for their customers.
11. Agents can view invoices for all past transactions.
12. Agents can view all available policies.
13. Agent summary provides the agents with an in-depth knowledge of which policies are the most popular as well as a customer-wise summary to know how much each customer is investing.

# MODULES

1] CUSTOMER

* Dashboard
* My Policies
* Policy Payment
* My Profile
* All Plans
* Agent Information
* Invoices
* Customer Summary

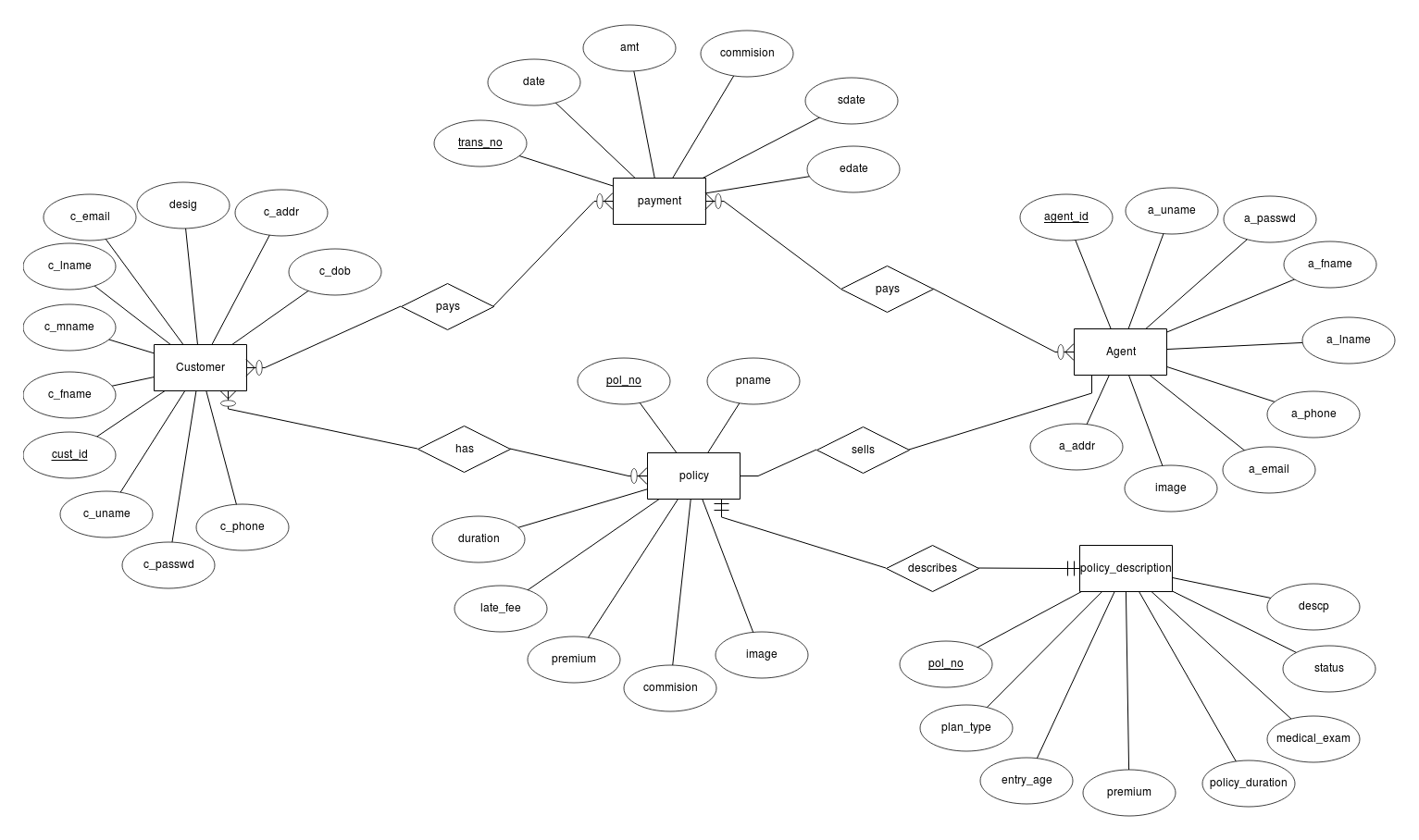
2] AGENT

* Dashboard
* My Profile
* My Customers
* Add Customers
* Add Policy
* Make Payment
* All Plans
* Agent Summary

3] ADMIN

* Dashboard
* All Agents
* Add new Plan
* Add Agent

# **ER DIAGRAM**

****

# **SCHEMA DESCRIPTION**

* The 4 main entities are Policy, Payment, Agent, Customer.
* There is an many to many relationship between Customer and Policy, Agent and Policy.
* There is one to many relationship between Customer and Payment, Agent and Payment.
* There is a one to one relationship between policy and policy-description.

# **TABLES DESCRIPTION**

# The DB has the following tables:

* Agent (Attributes: agent\_id, username, password, phone, email, address, name)
* Customer(Attributes: cust\_id, username, password, phone, email, address, name)
* Customer\_Agent\_Policy(Attributes: agent\_id, cust\_id, pol\_no)
* Policy(Attributes: pol\_no, name, commission, premium , late\_fee, duration)
* Payment(Attributes: agent\_id, cust\_id, pol\_no, date, amount, commision)
* Policy\_Description(Attributes: name, type, entry\_age, status, desc)

# **CODING**

**Triggers Used**:

1. Trigger used to update the the due-date for payment of the premium

delimiter $

create trigger p\_date

after insert on payment

for each row

begin

update customer\_policy set prem\_pdate =(select DATE\_ADD(prem\_pdate, INTERVAL 1 MONTH)) where customer\_policy.cust\_id=NEW.cust\_id AND customer\_policy.pol\_no=NEW.pol\_no;

end$

**Explanation:** As soon as the premium payment is being made, a months interval is added to the premium due date thus updating the premium due-date.

2. Trigger used to delete Customer dependencies

delimiter $

create trigger delcustpol

BEFORE delete from customer\_agent\_policy

for each row

begin

delete from customer\_policy where old.pol\_no=customer\_agent\_policy.pol\_no AND old.cust\_id=customer\_policy.cust\_id;

end$

delimiter ;

**Explanation:** This trigger ensures that each time before the record is deleted from the customer\_agent\_policy table, the corresponding record in the customer\_policy table is deleted.

**Procedure Used:**

1.Procedure is used to calculate Surrender Value of the given policy

delimiter $

create procedure surrender(INOUT value int)

begin

set value = value \* 1.05 ;

select @value;

end $

delimiter ;

**Explanation:** In this procedure we are calculating the surrender value using the total investment at this moment of time.

**Important Functions**:

1] A function to calculate the number of days left for the Premium payment due-date.

**public** **int** date(**int** cust\_id, **int** pol\_no) {

**int** date\_diff = 100;

String sql1 = "select prem\_pdate from customer\_policy where cust\_id=? AND pol\_no=?";

String sql2= "select datediff(?,curdate())";

Connection conn = **new** Connect().myDBConnect();

PreparedStatement stmt1 = conn.prepareStatement(sql1);

PreparedStatement stmt2 = conn.prepareStatement(sql2);

stmt1.setInt(1, cust\_id);

stmt1.setInt(2, pol\_no);

ResultSet rs1 = stmt1.executeQuery();

rs1.next();

Date p\_date= rs1.getDate("prem\_pdate");

System.***out***.println("Date:"+p\_date);

stmt2.setDate(1, p\_date);

ResultSet rs2= stmt2.executeQuery();

rs2.next();

date\_diff=rs2.getInt(1);

**return** date\_diff;}

**Explanation**: First the next premium payment due-date is fetched from the database and then it is the Current date is subtracted from it thus giving us the number of days left.

2] A function to calculate the term for which the premium is being paid

**public** Date term\_date(**int** cust\_id, **int** pol\_no{

Date prem\_term = **null**;

String sql1 = "select prem\_pdate from customer\_policy where cust\_id=? AND pol\_no=?";

String sql2= "select DATE\_ADD(?, INTERVAL 1 MONTH)";

**try** {

Connection conn = **new** Connect().myDBConnect();

PreparedStatement stmt1 = conn.prepareStatement(sql1);

PreparedStatement stmt2 = conn.prepareStatement(sql2);

stmt1.setInt(1, cust\_id);

stmt1.setInt(2, pol\_no);

ResultSet rs1 = stmt1.executeQuery();

rs1.next();

Date p\_date= rs1.getDate("prem\_pdate");

stmt2.setDate(1, p\_date);

ResultSet rs2= stmt2.executeQuery();

rs2.next();

prem\_term=rs2.getDate(1);

**catch** (Exception e) {

}

**return** prem\_term; }

**Expanation**: The premium due-date date is being added an interval of one month thus giving us the term for which the premium is being paid.

# **ERROR MESSAGES / ALERTS DESIGN**

* The login page displays “\*Incorrect Username Or Password” message on entering invalid login credentials.
* Each time a premium payment is made, the page displays “Transaction Successful”.
* Each time a Customer/Agent/Policy is successfully added, the page displays “Record Successfully Added”.
* Each the a customer surrenders a policy, the page displays “Policy Successfully Surrendered”.

# **TEST CASE DESIGN**

* For calculation of the days remaining to pay premium, the cases are as follows:

1. Due-date is today.

2. Due-date has been missed.

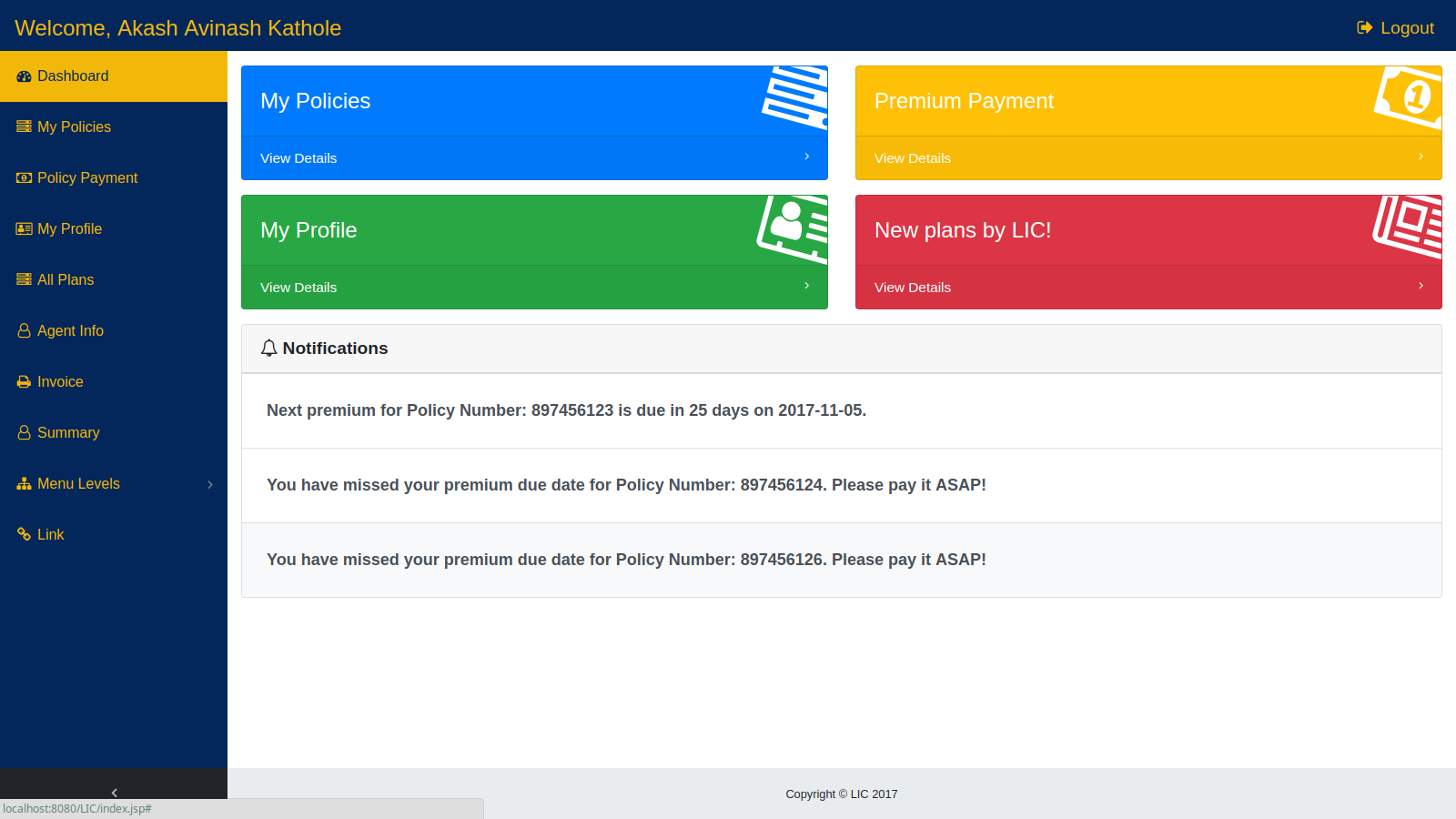
3. Due-date is upcoming

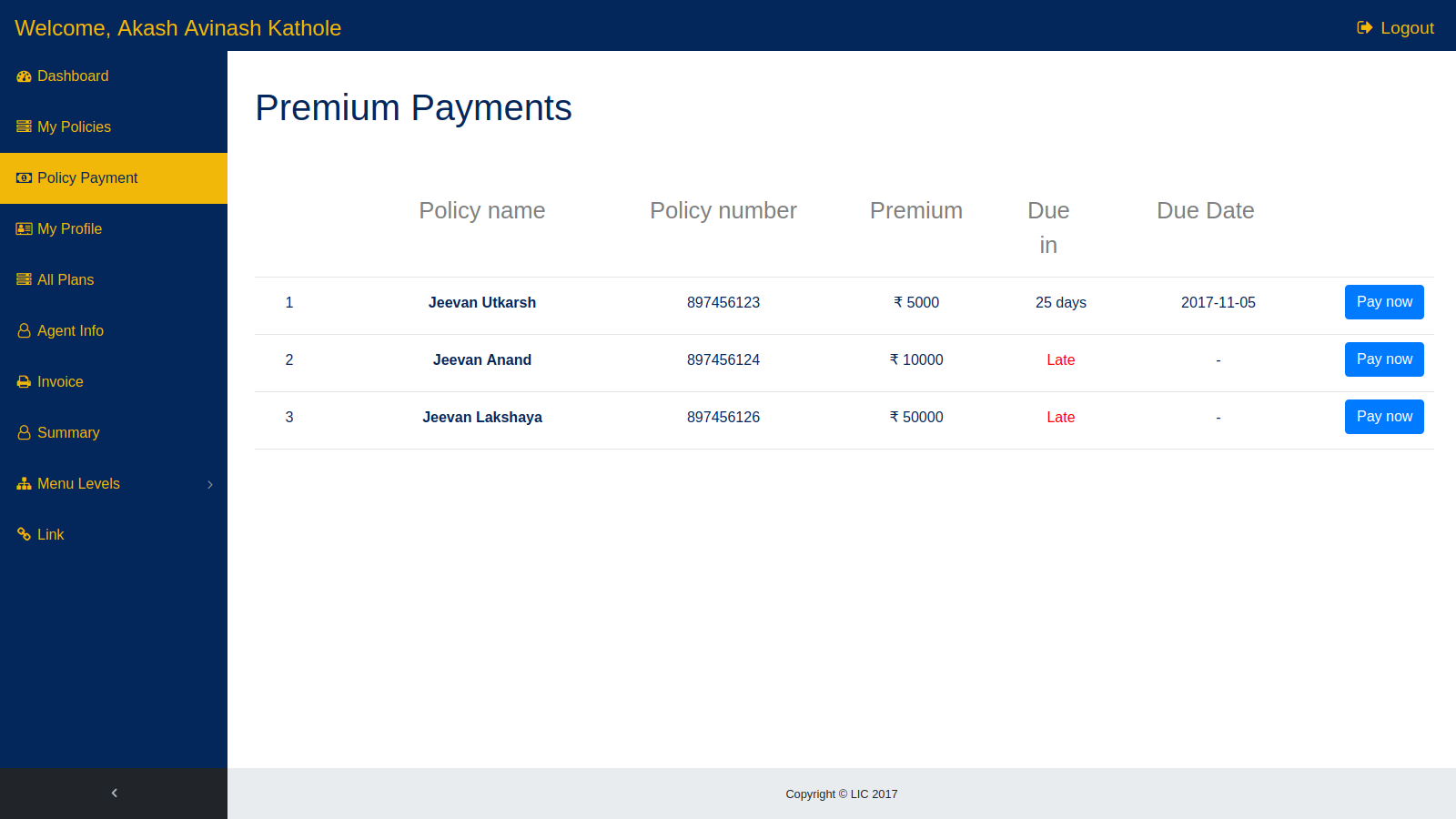
* For calculating whether there is a late fee or no, the cases are as follows:

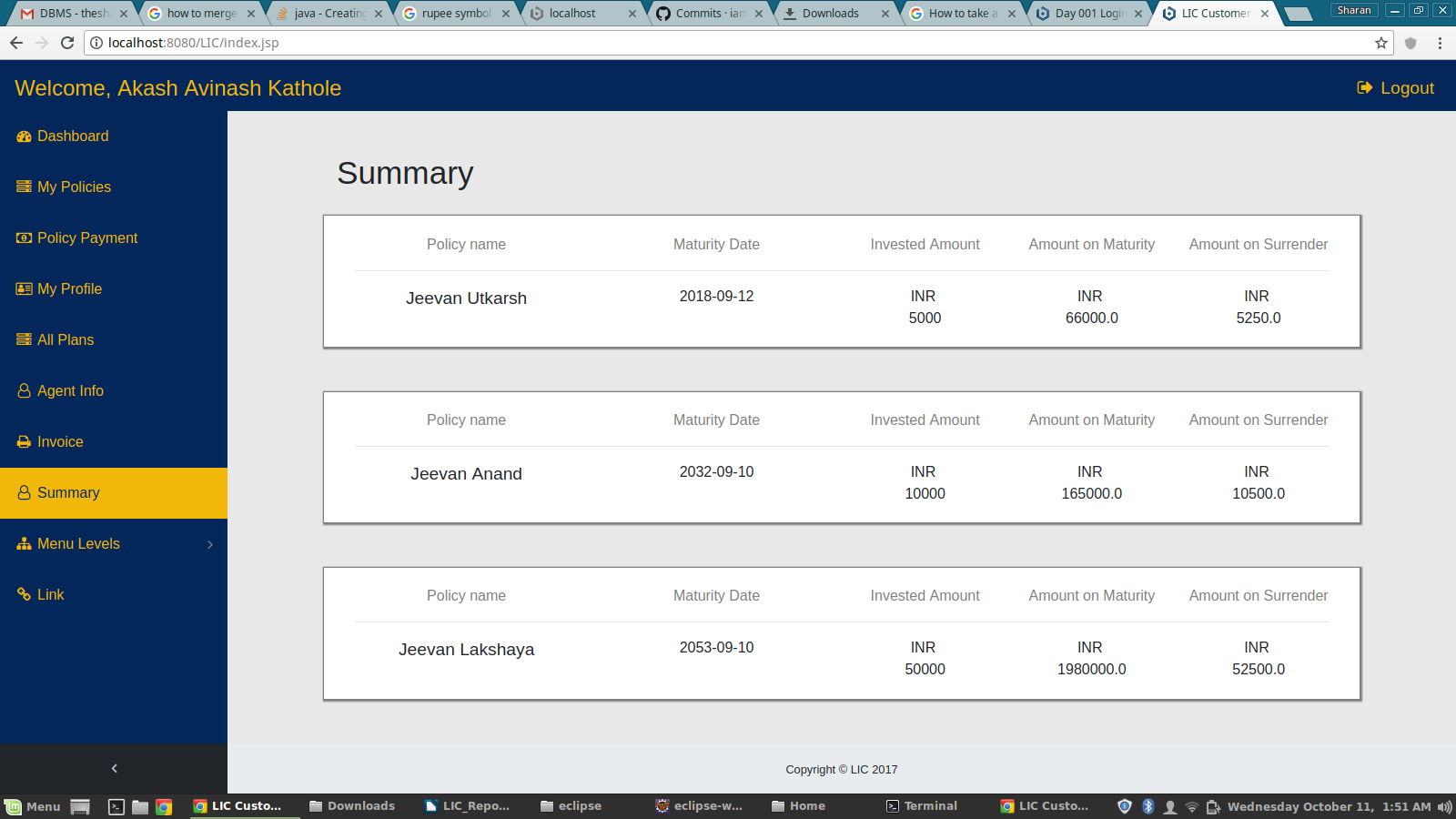
1. Due-date has been missed hence apply late fee.

2. Due-date has not been missed hence do not apply late fee.

# 





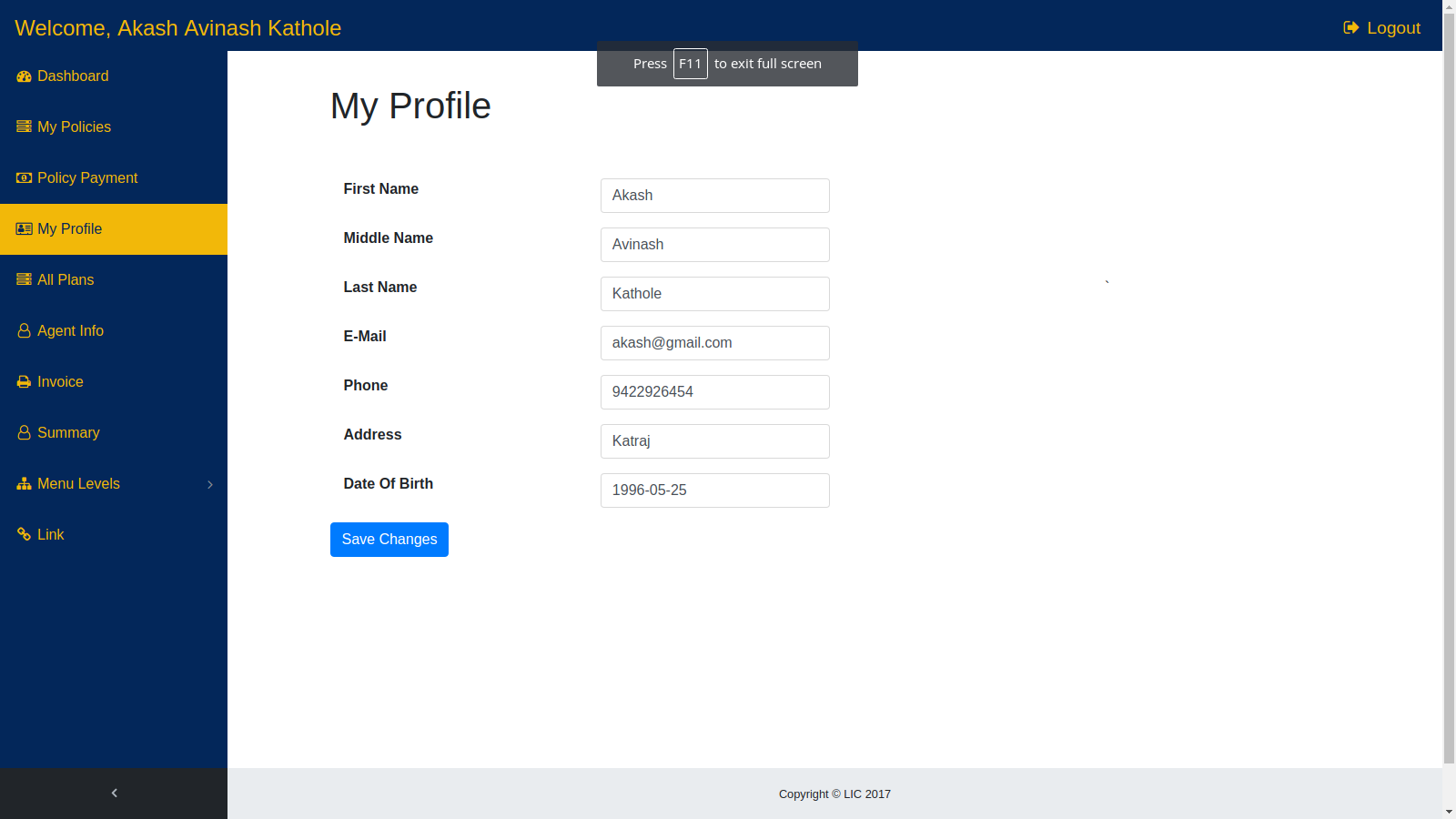


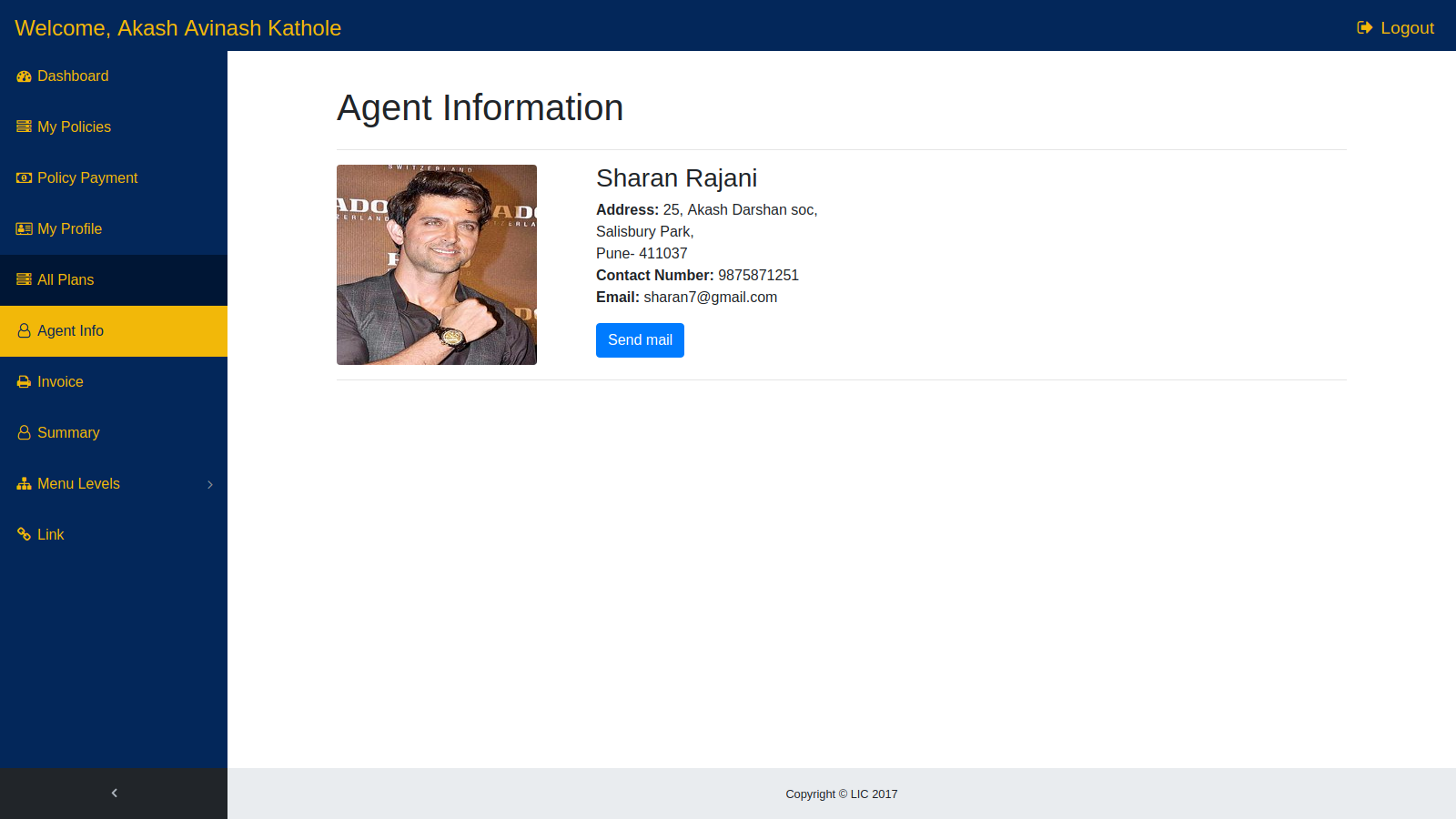
# 

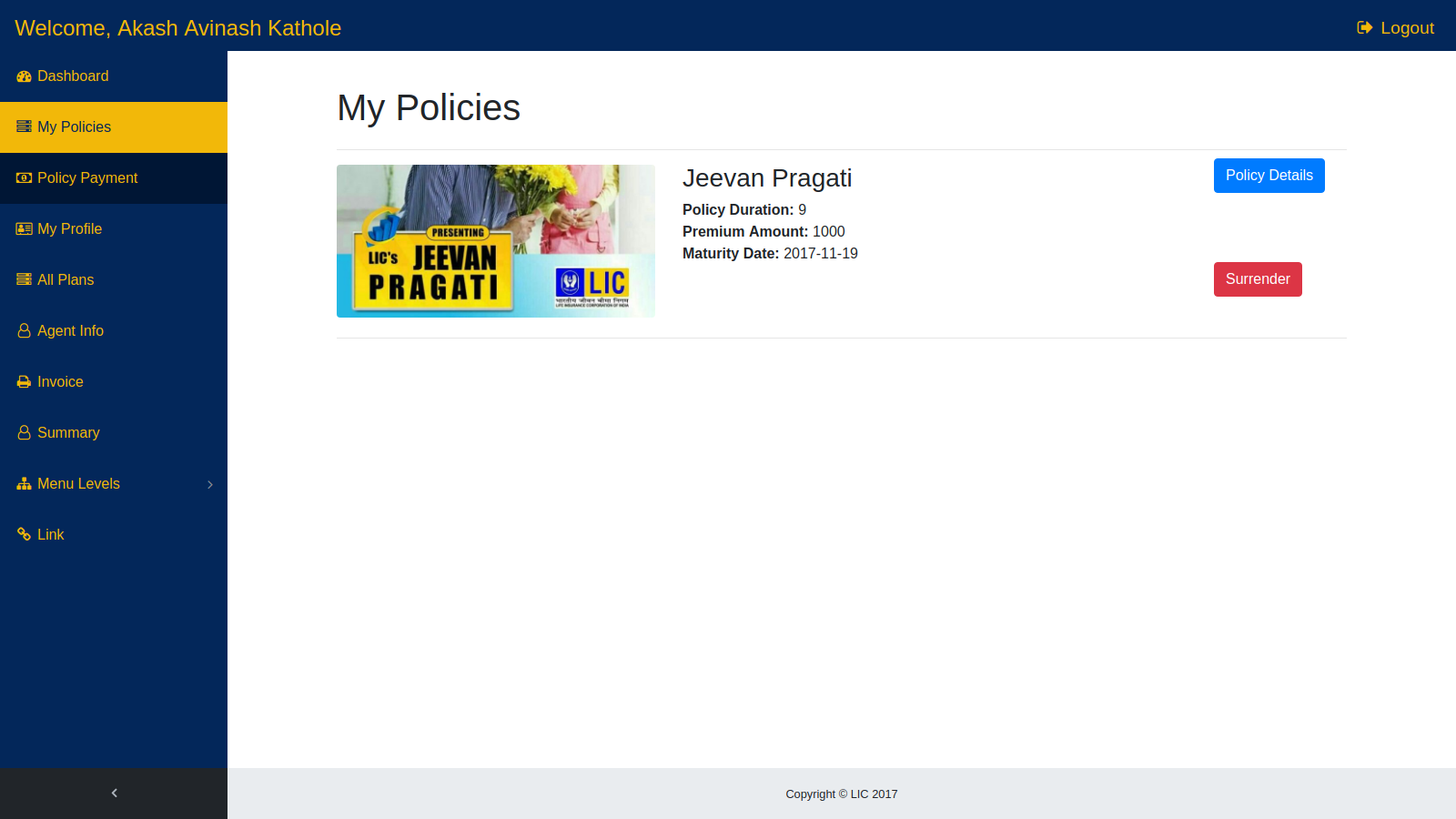
# 

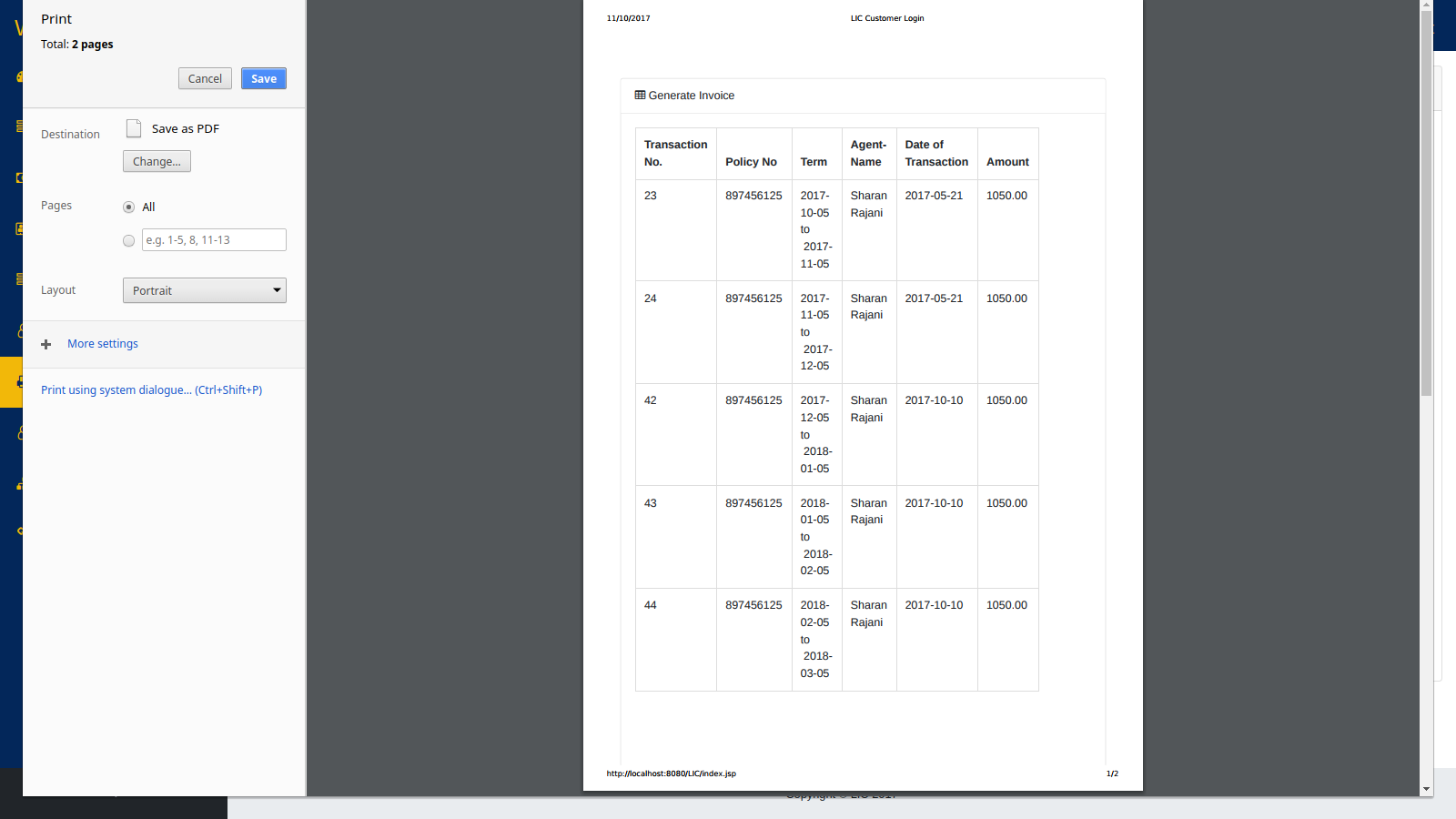
# 

# 









# CONCLUSION

The three parts which are essential for this project are User interface, Creation of relational database and SQL engine.

The Project is entirely based on database management system concepts. The back-end use for project is MySql and front-end is HTML, CSS, BOOTSTRAP and JQUERY. The Coding of Sql queries through Java is properly done. The project is very feasible.

The software engineering concepts are used to implement the project. The requirement analysis is understood and done for this project.