PROJECT REPORT
ON

Gold Loan Finance Management System

By

THAKKAR ADITYA M. (CE-130) (19CEUOS131)
THAKKAR SAGAR D. (CE-131) (19CEUOG026)

B.Tech-CE Semester-IV
Subject: Software Project

Guided by:
Prof. Brijesh S. Bhatt
Prof. Jigar M. Pandya
Prof. Pinkal C. Chauhan
Dept. of Computer Engineering.

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

Table of Contents

1. Abstract

Gold Loan Finance Management System is an online gold loan system. The software is proposed to develop for any individual businessman/any enterprise for managing the loans lent to the customers who mortgage their gold for specific period of time. The customers use this software for applying for the loan and can view the application status. After application is accepted by enterprise, they can track the loan amount they need to pay in form of instalments and the amount already paid. The customers could mortgage up-to few hundred grams of gold.

2. Introduction

2.1 Brief Introduction

Any customer can check the present interest rate of loan, present market value of gold and after that can calculate EMI for specific principal, interest rate and Loan tenure. Also, by deciding if they meet the eligibility criteria, they can apply for loan thereafter.

When any customer issues/requests/applies for gold loan, they should be able to register the customer details like its name, address, contact number, Aadhaar card number/pan card number(any one proof id), weight of gold(in gm's) mortgaged. After applying for loan, the customer can check the status of loan application which would be approved by the enterprise/admin. Admin can see the pending list of applications of loan applied by the customers and based on their decision, they can approve or reject the loan application.

The loan would be lent to the customers at the present interest rate of the loan set by administration. Loan amount given to customer should be equal to particular-percentage(Percentage would also be set by individual enterprise lending the loan) of equivalent amount of market value of gold. The software should allow to set and update market value of gold by the administration. The administration can see all the list of customers and their details.

If the enterprise approves the loan application by the customer, they would be provided by unique id which software should automatically generate and with the password which they can update anytime themselves. They can login by this unique id provided by enterprise and can see all the details regarding the date of lending the loan, how many instalments are left for payment and next due date for instalment and how many instalments he already paid.

Customer can pay for the due instalment, or upcoming instalments in advance from their login menu online. When customer pays for the instalment the software should show the date of instalment paid and the amount should be debited from the loan amount. If the instalment date is already due and still after the due date customer doesn't pay the instalment, extra charges per day(fixed by administration) would be added to the ongoing instalment by the software.

2.2 Tools/Technologies Used

Technologies:

- Django
- Python
- MySQL
- Bootstrap
- JavaScript
- HTML
- jQuery

Tools:

Visual Studio Code

Platform

Local development server

3. Software Requirement Specifications

3.1 End Users

- Customer
- Admin
- Guest User

3.2 System Functional Requirements

R1.MASTER ENTRY

R.1.1 : ADMIN LOGIN

Description : Admin can login by entering username and password . If username and password match correctly then confirmation message should be displayed.

Input : Username and Password of Admin.

Output : Confirmation message.

Processing: Password validation.

R.1.2 : UPDATE GOLD VALUE

Description : If there is a change of gold value in market then only admin can change gold value by updating the new market value of gold and confirmation message should be displayed.

Input : New value of gold.

Output : Update gold value and display confirmation message

R.1.3 : UPDATE INTEREST RATE OF GOLD

Description : If admin wants to change interest rate of gold then it can be changed by updating new rate value and so from now onwards the customers applying for loan will be lent on this updated rate of interest.

Input : New value of interest rate

Output : confirmation message.

R.1.4 : APPROVE LOAN APPLICATION

Description : Admin can view the list of the loan applications which are applied by the guest users. They can be approved or rejected by admin. If admin approves or rejects the loan application of specific user then loan application status of that particular guest user should be updated based on that.

Input : Admin selection for approval or rejection.

Output : If admin approves ,then new unique id should be generated for that customer which should be used for Customer Login. And if it is rejected then rejection reason should be displayed on Application Status of guest user. Loan application status of that particular guest user should be updated based on that.

Processing: If admin approves then NEW UNIQUE ID which is generated by system should be displayed which should be used for customer login and from here guest user is a fresh customer and all loan transactions will be started from here on for that customer.

2. CUSTOMER MANAGEMENT

R.2.1 : CUSTOMER DETAILS

Description : Customers' details like name , contact number , email address, documents details(like pan card/Aadhaar card), nominee name etc. are hidden from outside world can be viewed by admin.

Input : Admin selection

Output : Details of customer like Name , contact no. , email, documents details, nominee name etc. are displayed

R.2.2 : ACTIVE LOANS

Description : Admin can see the active loans and details of the same based on Loan Tenure(number of years) or unique id assigned to customer or by name of customer or interest rate of loan given at any instance of time.

Input : Admin selection based on Loan Tenure or unique id of customer or name of customer or interest rate of loan given at any instance of time.

Output : Customer Details.

Processing: Search the loans based on selected keyword i.e. Loan Tenure, unique id of customer, name of customer, interest rate of loan.

3. LOAN MANAGEMENT

3.1 : EMI CALCULATOR

Description : Any user who wants to calculate EMI can calculate by selecting principal amount, rate of interest and number of years(Loan Tenure).

Input : User selection

Output: Total Interest to be paid for selected principal amount and EMI for selected Loan Tenure.

3.2 : ELIGIBILITY CRITERIA

Description : There are some specific eligibility conditions for gold loan like age, maximum gold weight can customer can mortgage etc. should be displayed.

Input : User click

Output: Eligibility Criteria set by system for gold loan.

3.3 : APPLY FOR LOAN

Description : Any interested user can apply for gold loan by filling His/ Her details and if the details meet the eligibility criteria then some confirmation message that shows the details are correct or wrong that should be displayed.
Input : Name , contact no. , email, documents details like , nominee name etc.

Output: Confirmation message with Guest Username and password.

Processing: If all details are authentically filled then this application of loan of customer should be sent to admin and confirmation message should be displayed OR if the customer is not eligible then message accordingly should be displayed.

3.4 : LOAN APPLICATION STATUS

Description : Guest users can see the status of loan application by logging in details which they created while applying for loan.

Input : Guest User name and password.

Output: PENDING or APPROVED or REJECTED details.

Processing: Password validation.

4. USER MANAGEMENT

4.1 : CUSTOMER LOGIN

DESCRIPTION : Customer can login by entering username and password . If username and password match correctly then confirmation message should be displayed.

INPUT : Username and Password of customer.

OUTPUT: Customer Details.

PROCESSING: Password validation

NEXT FUNCTION: R.4.2 or R.4.3 based on user selection if password is valid and R.4.1 if password is invalid.

4.2 : INSTALMENT PAYMENT

DESCRIPTION : Customer can pay the instalments which is due OR also can pay remaining instalments in advance by filling amount and details of account. If the conditions are satisfied confirmation message should be displayed.

INPUT : Details of account and number of instalment(s) .

OUTPUT: Confirmation message.

PROCESSING: If all details are filled correctly then status for instalment(s) should be updated as a successful transaction and if transaction could not be completed then display unsuccessful transaction.

4.3 : TRACK PAYMENT DETAILS

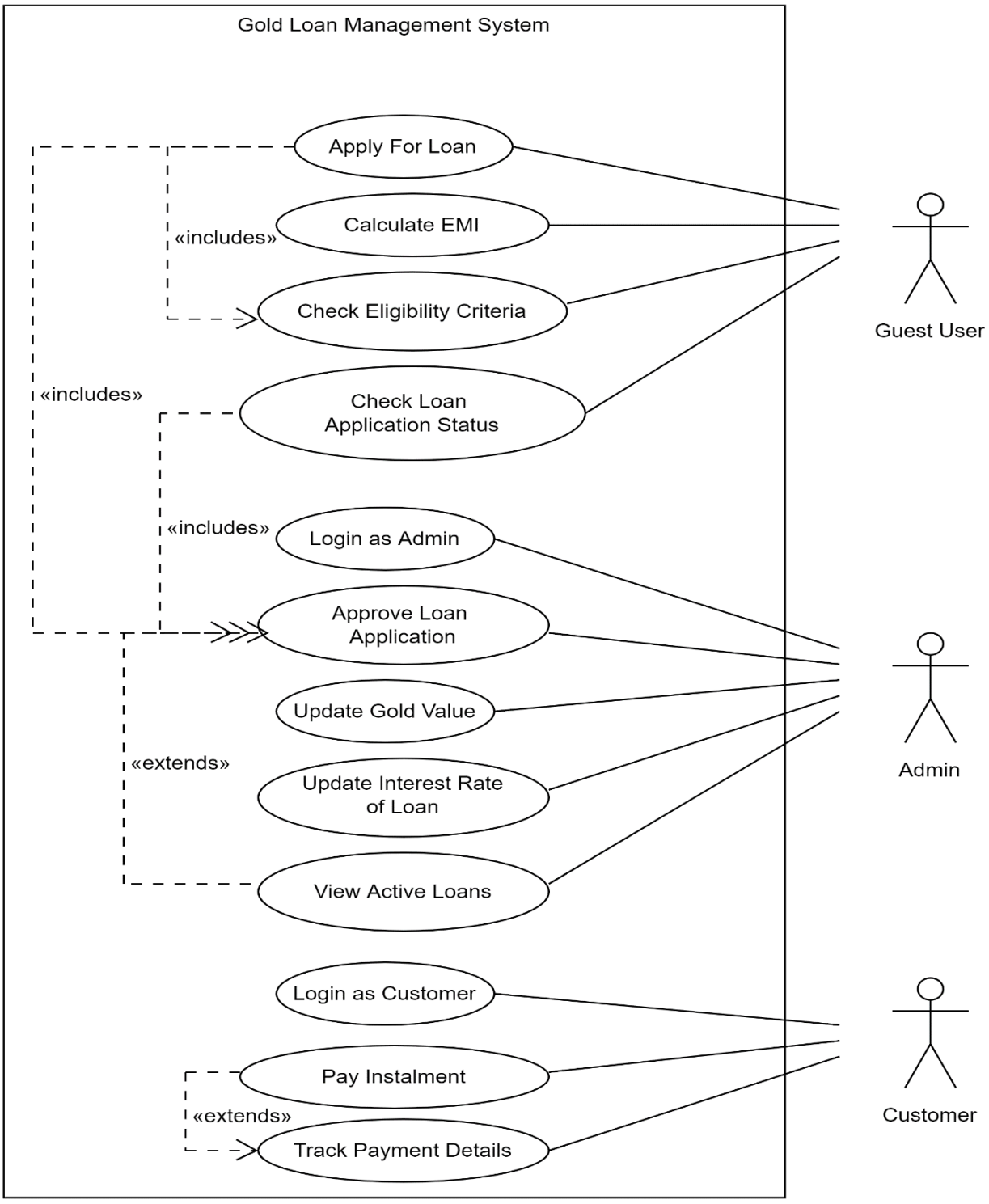
DESCRIPTION : Customer can track the details like Information of instalments , Next due of instalment, the interest rate at which customer is lent loan etc.

INPUT : User Selection.

OUTPUT : As per user selection.

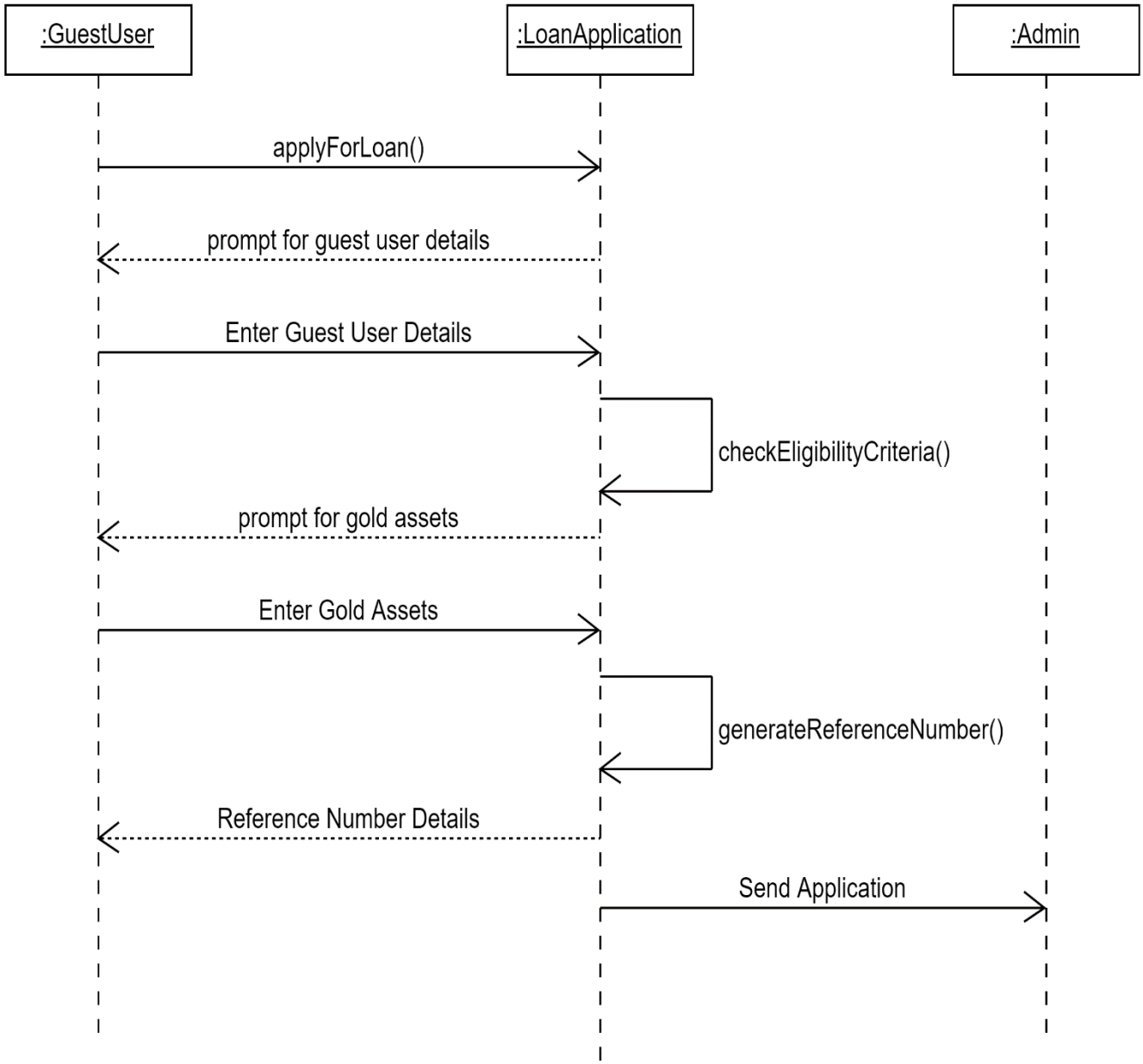
4. Design

4.1 Use-Case Diagram

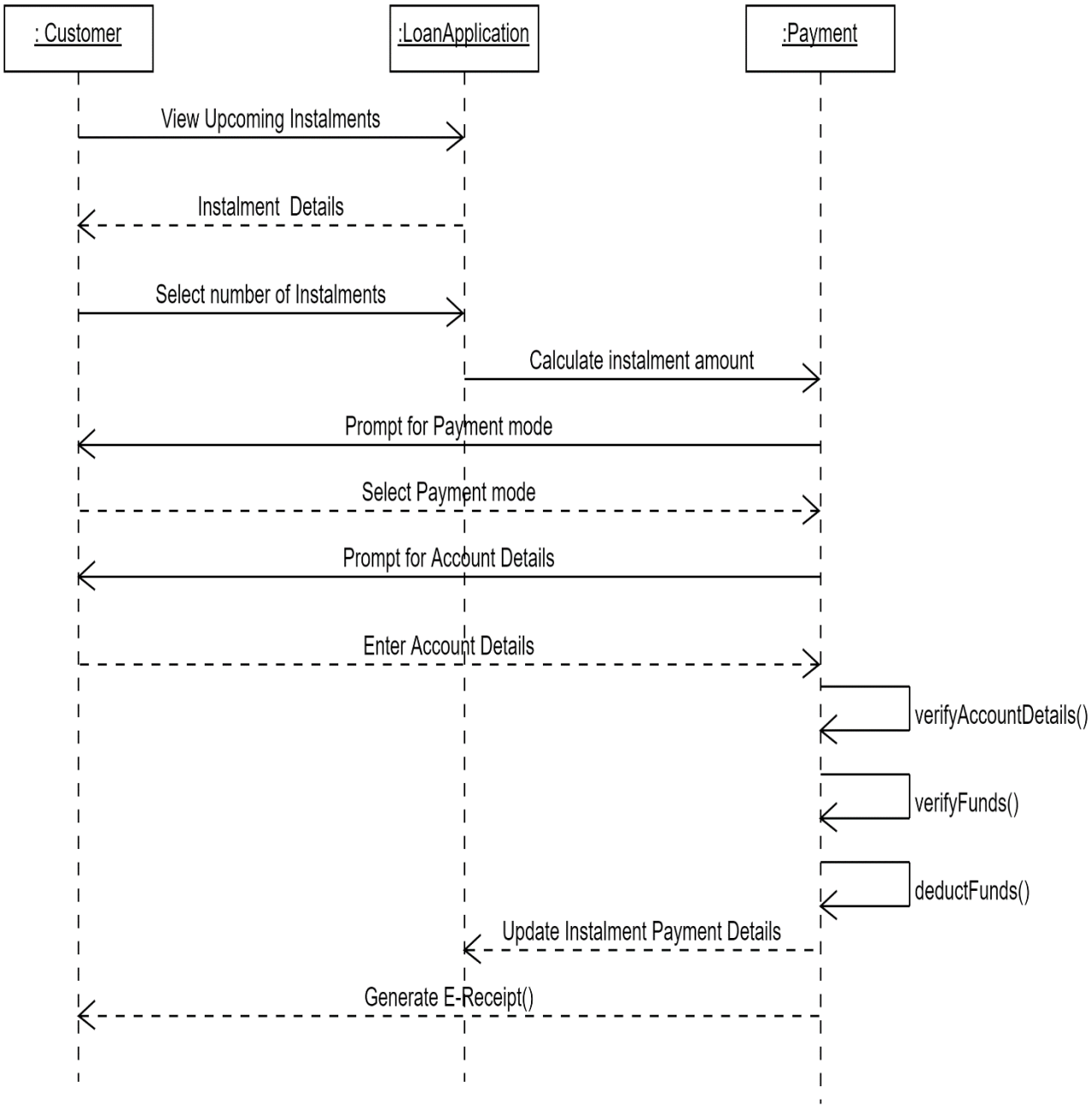


4.2 Sequence Diagram

4.2.1 Sequence Diagram for Apply for Loan

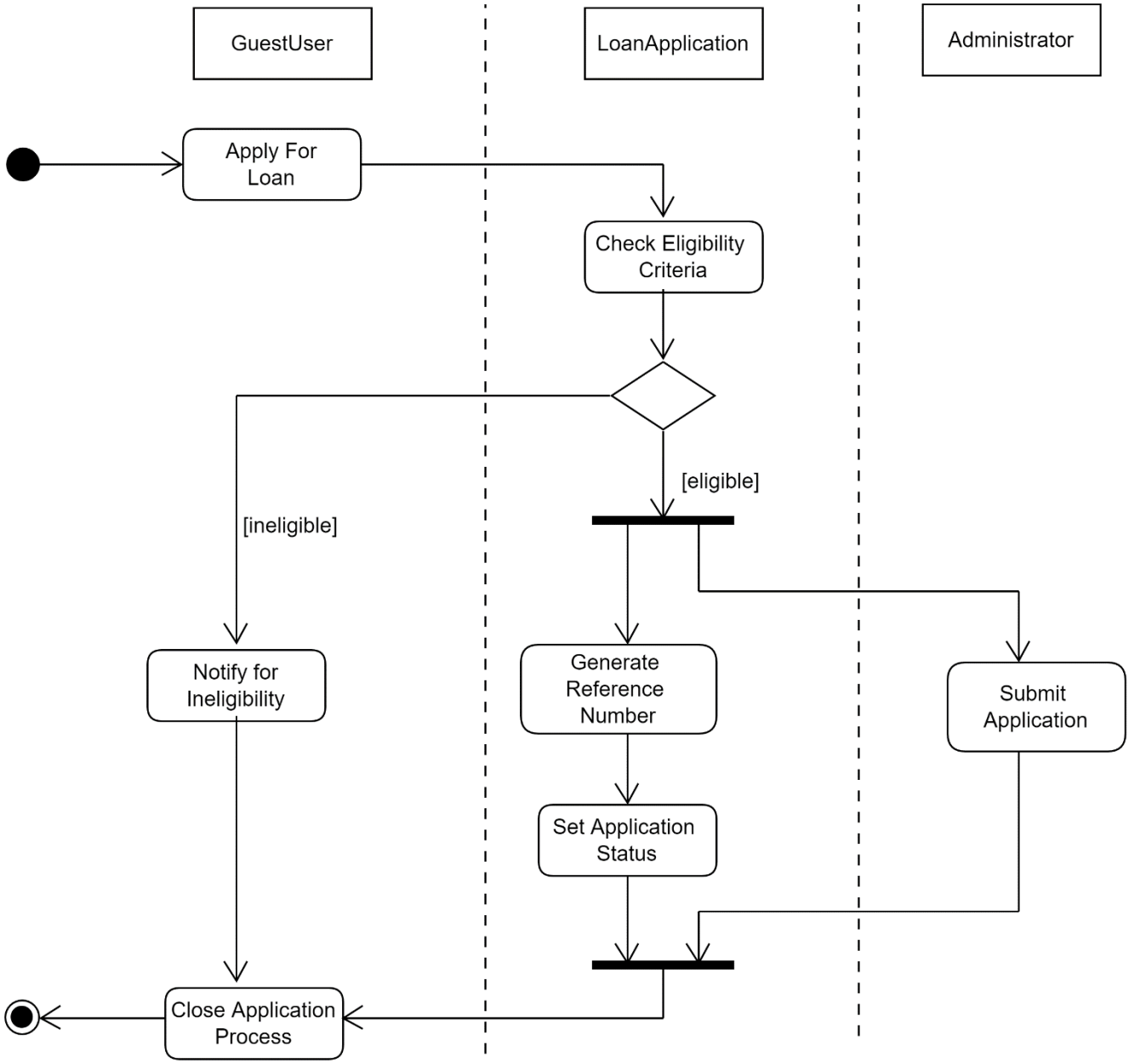


4.2.2 Sequence Diagram for Instalment Payment

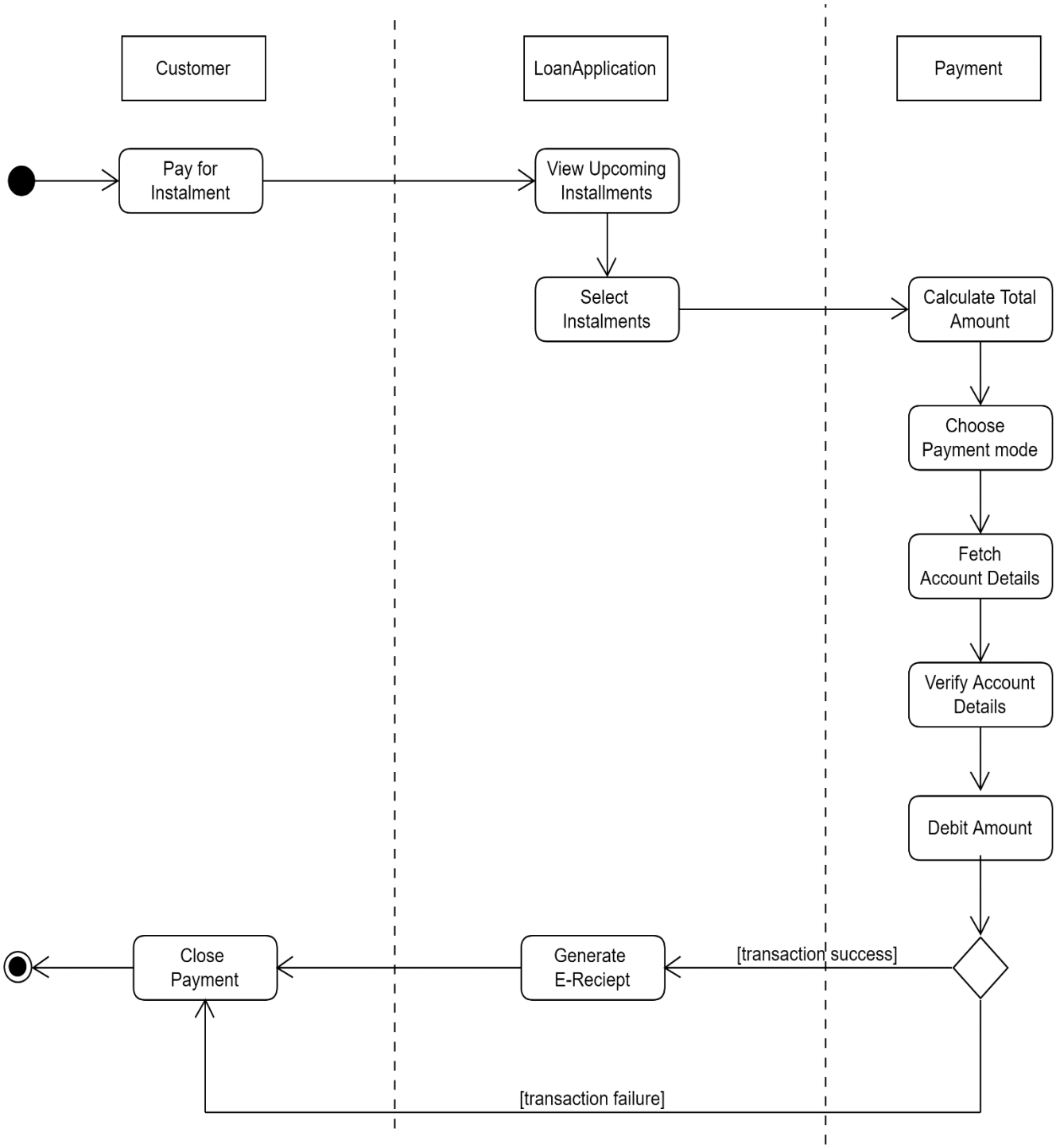


4.3 Activity Diagram

4.3.1 Activity Diagram for Apply for Loan



4.3.2 Activity Diagram for Apply for Loan



4.4 Data Dictionary

Customer						
Sr no.	Field Name	Data Type	Unique	PF/FK	Referred Table	Description
1	username	varchar	yes	PK	-	-
2	password	varchar	-	-	-	-
3	customerName	varchar	-	-	-	-
4	customerId	int	yes	FK	-	Auto Increment
5	aadhaarId	decimal	yes	-	-	-
6	nomineeName	varchar	-	-	-	-
7	income	int	-	-	-	-
8	address	varchar	-	-	-	-
9	mobile	decimal	-	-	-	-
10	dob	date	-	-	-	-
11	applicationDate	date	-	-	-	-

GoldAsset						
Sr no.	Field Name	Data Type	Unique	PF/FK	Referred Table	Description
1	assetID	int	yes	PK	-	Auto Increment
2	weight	decimal	-	-	-	-
3	goldType	varchar	-	-	-	-
4	customerId	int	yes	FK	Customer	-

LoanApplication						
Sr no.	Field Name	Data Type	Unique	PF/FK	Referred Table	Description
1	loanId	Int	yes	PK	-	Auto Increment
2	principalAmount	decimal	-	-	-	-
3	lentRateOfinterest	Decimal	-	-	-	-
4	lentLoanTenure	smallint	-	-	-	-
5	loanApplicationStatus	Varchar	-	-	-	-
6	lentGoldValue	Decimal	-	-	-	-
7	assetID	int	Yes	FK	GoldAsset	-
8	customerId	int	Yes	FK	Customer	-
9	emi	decimal	-	-	-	-
10	paid	decimal	-	-	-	-
11	totalLoanAmount	decimal	-	-	-	-
12	Emipaid	Int	-	-	-	-
13	lentltvRatio	decimal	-	-	-	-

Payment						
Sr no.	Field Name	Data Type	Unique	PF/FK	Referred Table	Description
1	id	int	yes	PK	-	Auto Increment
2	cardtype	varchar	-	-	-	-
3	cardnumber	int	Yes	-	-	-
4	cvc	int	Yes	-	-	-
5	expmonth	int	-	-	-	-
6	expyear	varchar	-	-	-	-
7	customerId	Int	yes	FK	Customer	-

LoanRates						
Sr no.	Field Name	Data Type	Unique	PF/FK	Referred Table	Description
1	id	int	Yes	Yes	-	Auto Increment
2	GoldValue	decimal	-	No	-	-
3	RateOfInterest	Decimal	-	No	-	-
4	ltvRatio	Decimal	-	yes	-	-
5	Date	date	-	Yes	-	-

5. Implementation Details

The system consists of 3 basic modules namely

1. Apply for Loan Module
2. EMI Payment Module
3. Admin Module
4. Login Module

Each module consists of several methods to implement the required functionality. Implementation is done using Django. Database used in these modules is MySQL.

Apply for Loan Module

This module allows customer to apply for loan. Customer is made to fill necessary details related to Loan Application like his/her personal details, gold loan details, etc.

EMI Payment Module

This module is for payment of EMI's which is used to pay monthly or in-advance EMI's through online banking. The payment details can only be seen by the customer.

Admin Module

This module allows admin for taking action on Loan Applications of Customers, for updating Loan Rates etc.

Login Module

This module is the base for authentication and authorization to ensure the security aspect of the user.

5.1 Function prototypes

```
def addCustomer(request):
    if request.method == 'POST':
        username = request.POST.get('email')
        password = request.POST.get('password')
        customerName = request.POST.get('customerName')
        aadhaarId = request.POST.get('aadhaarId')
        nomineeName = request.POST.get('nomineeName')
        income = request.POST.get('income')
        address = request.POST.get('address')
        mobile = request.POST.get('mobile')
        dob = request.POST.get('dob')
        c = Customer(username=username,password = password,customerName = customerName,aadhaarId = aadhaarId,nomineeName = nomineeName,
        income = income,address = address,mobile = mobile,dob=dob)
        c.save()
        user = User.objects.create_user(username = username, password = password)
        user.save()
        goldtype = request.POST.get('goldtype')
        weight = request.POST.get('goldweight')
        customerId = c.customerId
        g = GoldAsset(goldtype=goldtype,weight=weight,customerId_id=customerId)
        g.save()
        loanamount = float(request.POST.get('loanamount'))
        tenure = float(request.POST.get('tenure'))
        assetId = g.assetID
        t = tenure*12
        r = 9.5/(12*100)
        emi = loanamount * r * pow((1+r),t)/(pow((1+r),t)-1)
        totalLoanAmount = emi*t
        rate = LoanRates.objects.last()
        if rate is None:
            loanrate = LoanRates()
            loanrate.save()
            rate = LoanRates.objects.last()
        l = LoanApplication(principalAmount = loanamount,lentLoanTenure = tenure,lentGoldValue=rate.GoldValue,lentRateOfInterest=rate.RateOfInterest,lentltvRatio=rate.ltvRatio,
        totalLoanAmount = totalLoanAmount,emi=emi,customerId_id = customerId,assetId_id = assetId)
        l.save()
        cardtype = request.POST.get('cardtype')
        cardnumber = request.POST.get('cardnumber')
        cvc = request.POST.get('cvc')
        expmonth = request.POST.get('expmonth')
        expyear = request.POST.get('expyear')
        p = Payment(cardtype = cardtype,cardnumber = cardnumber,cvc = cvc,expmonth =  expmonth,expyear = expyear,customerId_id = customerId)
        p.save()
        # return render(request,"success.html")
        messages.success(request,"Application submitted successfully!!")
        messages.success(request,"Login with your username and passsword.")
        return HttpResponseRedirect('/loginmodule/login')
    else:
        return render(request,"fail.html",{"message":"First , Please  fill  the details"})

```

Apply for loan

```python
@login_required(login_url='/loginmodule/login')
def pay_emi(request):
    if request.method == "POST":
        customerId = request.POST.get('customerId')
        month = int(request.POST.get('month'))
        l = LoanApplication.objects.get(customerId = customerId)
        if month == l.lentLoanTenure*12:
            l.emipaid = month
            l.paid = l.emi + l.paid
            l.save()
            messages.success(request,"Congratulations,Full Loan Payment done!!")
        elif month < l.lentLoanTenure*12:
            l.emipaid = month
            l.paid = l.emi + l.paid
            l.save()
            payment_done="Payment done successfully for month = " + str(month)
            messages.success(request,payment_done)
        return redirect('/loginmodule/loggedin_customer/')
    else:
        return redirect('/loginmodule/login')

```

Pay EMI

```python
@login_required(login_url='/loginmodule/login')
def approve(request):
    if request.method == "POST":
        customerId = request.POST.get('customerId')
        l = LoanApplication.objects.get(customerId = customerId)
        l.loanApplicationStatus = "APP"
        l.save()
        return HttpResponseRedirect('/loginmodule/loggedin_admin/')
    else:
        return redirect('/loginmodule/login')

@login_required(login_url='/loginmodule/login')
def reject(request):
    if request.method == "POST":
        customerId = request.POST.get('customerId')
        l = LoanApplication.objects.get(customerId = customerId)
        g = GoldAsset.objects.get(customerId = customerId)
        p = Payment.objects.get(customerId = customerId)
        c = Customer.objects.get(customerId = customerId)
        username = c.username
        password = c.password
        user = auth.authenticate(username=username,password=password)
        user.delete()
        l.delete()
        g.delete()
        p.delete()
        c.delete()
        return HttpResponseRedirect('/loginmodule/loggedin_admin/')
    else:
        return redirect('/loginmodule/login')
```

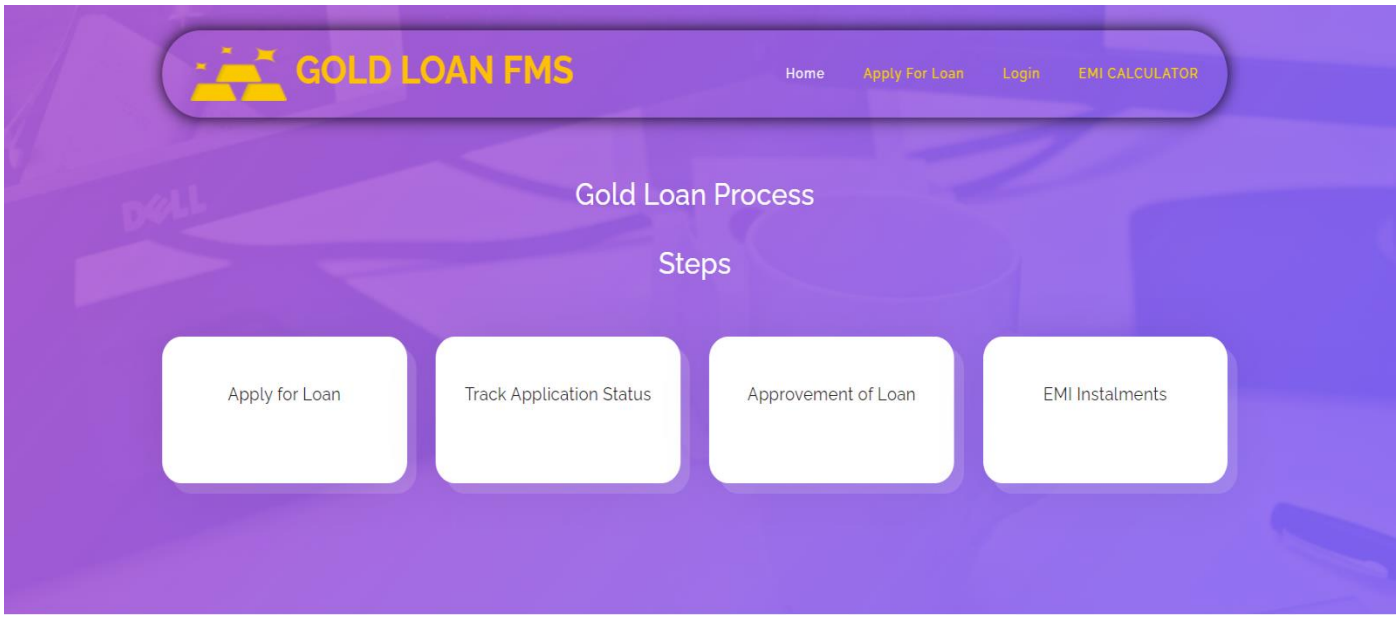
Loan Application Approve/Reject

```python
@login_required(login_url='/loginmodule/login')
def update_rates(request):
    if request.method == "POST":
        goldvalue = request.POST.get('goldvalue')
        interest = request.POST.get('interest')
        ltv_ratio = request.POST.get('ltv_ratio')
        rate=LoanRates(GoldValue=goldvalue,RateOfInterest=interest,ltvRatio=ltv_ratio)
        rate.save()
        return HttpResponseRedirect('/loginmodule/loggedin_admin/')
    else:
        return redirect('/loginmodule/login')
```

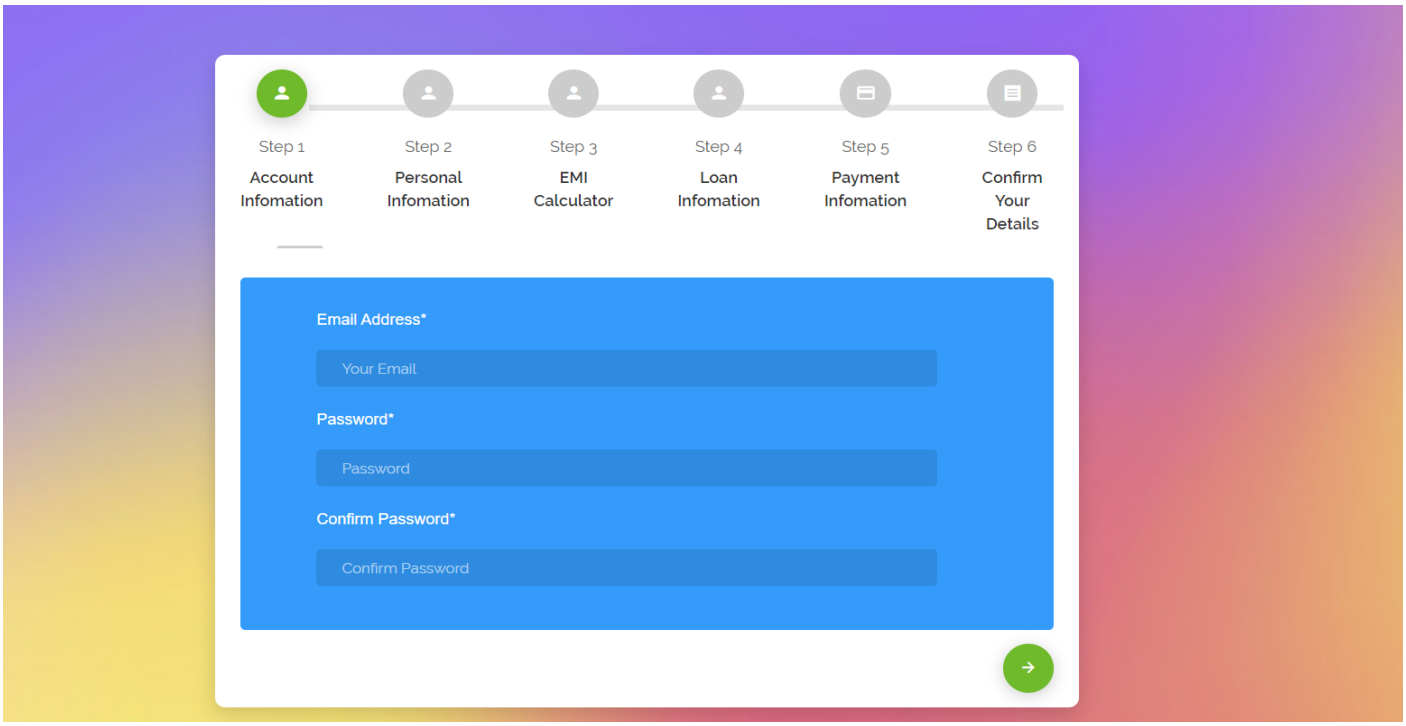
Update Loan Rates

7. Screenshots

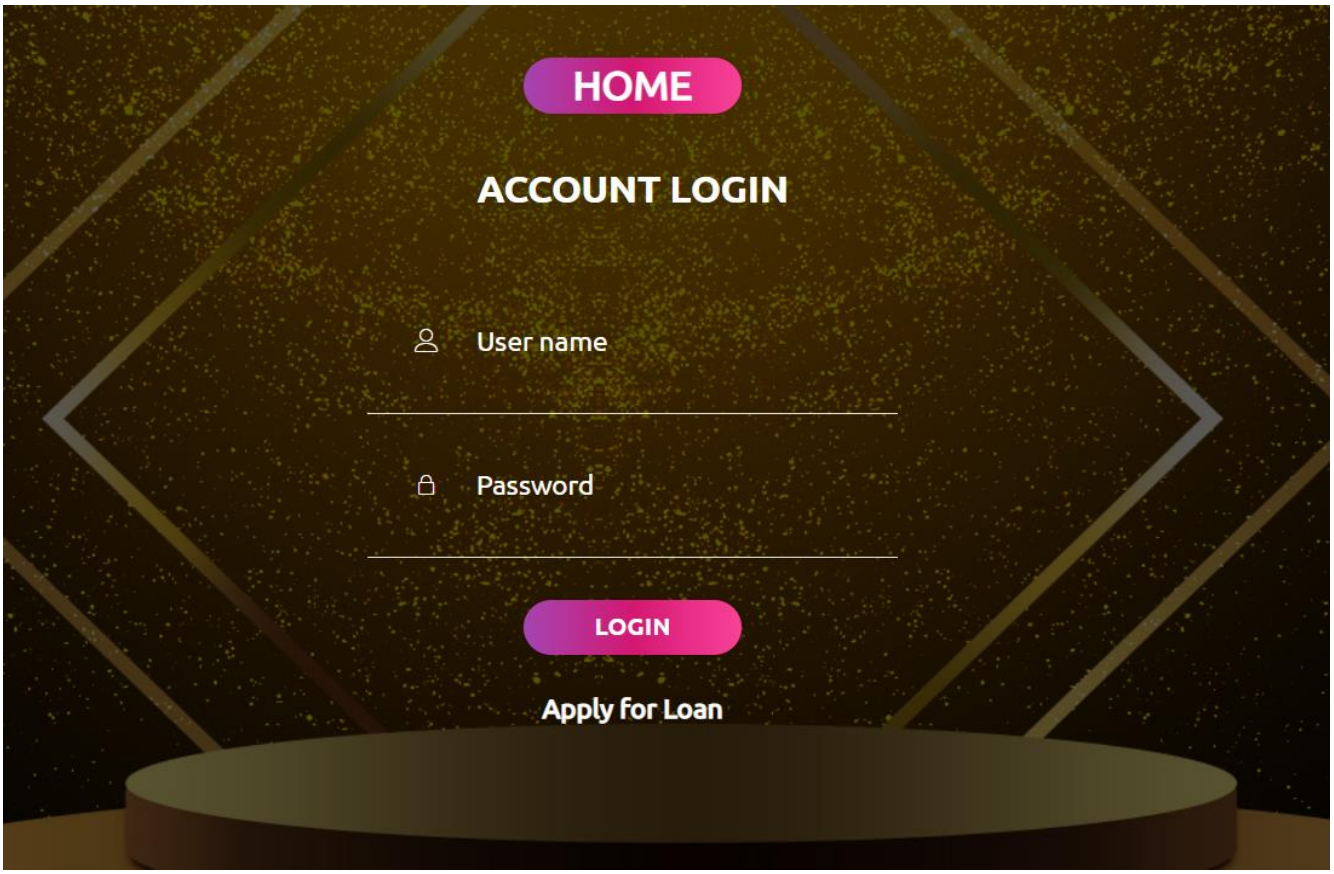
Home Page



Apply for Loan



Login Page



Loan Applications

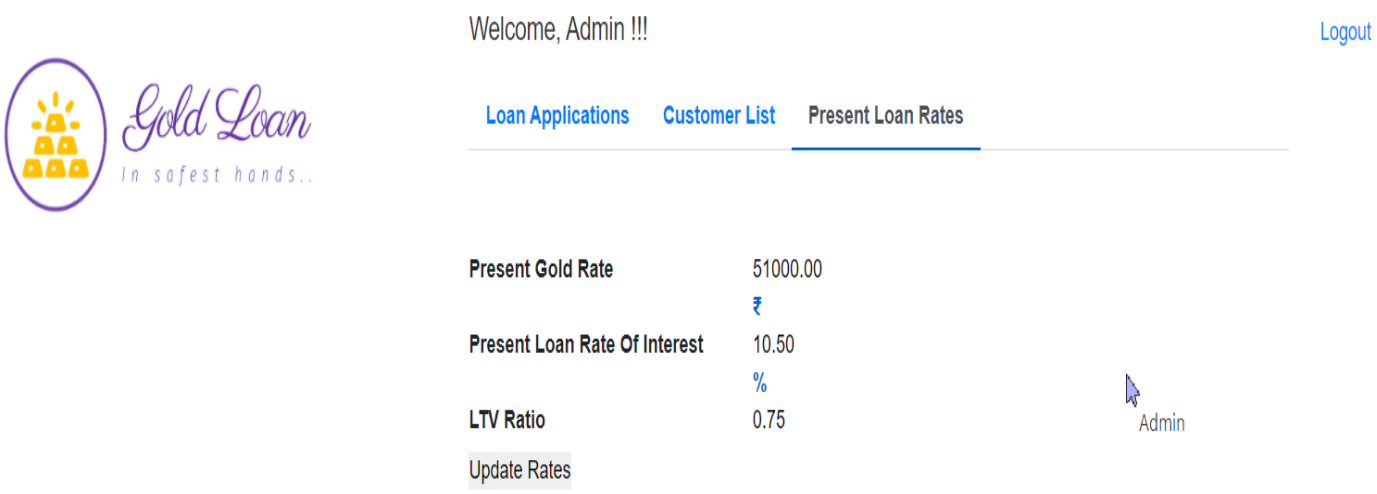
Welcome, Admin !!!

Logout

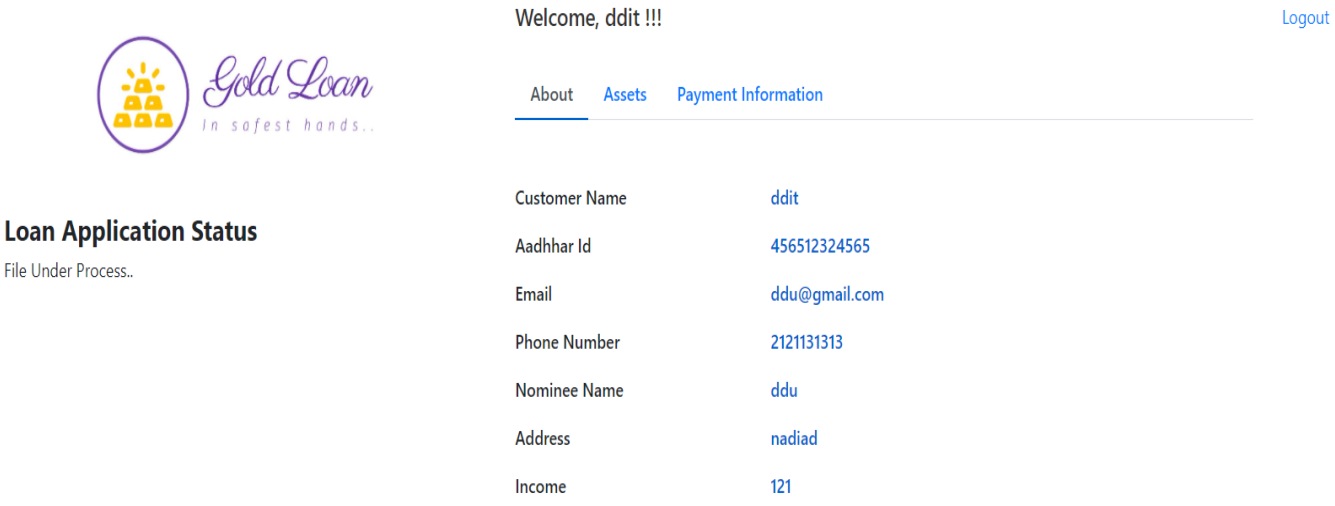
Loan Applications | Customer List | Present Loan Rates

ID	NAME	AADHHAR ID	NOMINEE NAME	INCOME	ADDRESS	MOBILE	APPLICATION DATE	GOLD WEIGHT	GOLD TYPE	LOAN AMOUNT	TENURE	LENT GOLD VALUE	EMI	STATUS		
110	aditya	121236365645	asas	21	12	5868541420	March 27, 2021	21.00 gms	B	121212.00	12	50000.00	1413.78	FUP	APPROVE	REJECT
111	jakjsak	456512324565	ajskja	121	12	2121131313	March 27, 2021	12.00 gms	B	21212.00	12	50000.00	247.41	FUP	APPROVE	REJECT

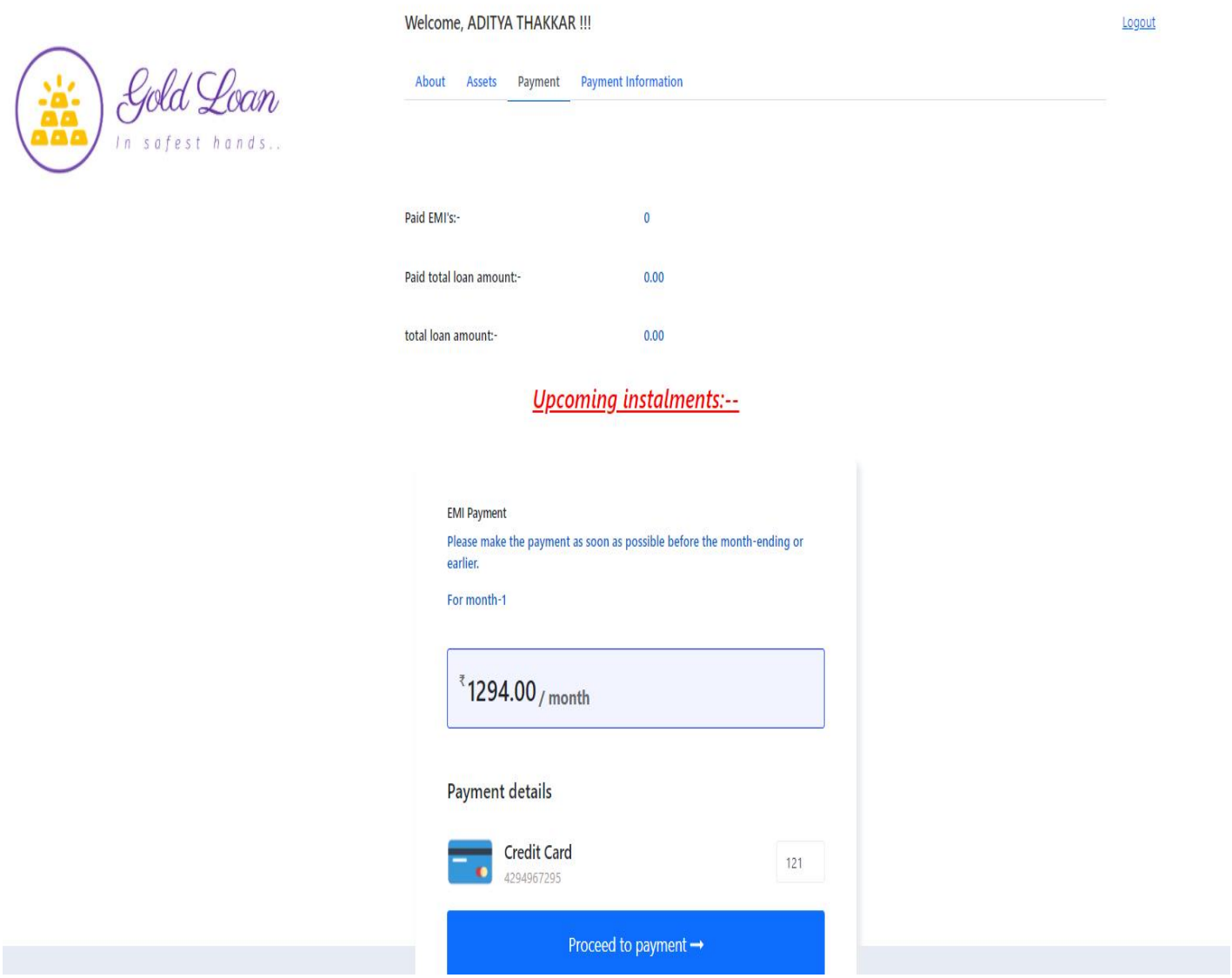
Update Loan Rates



Customer Home Page(Appl. Pending)



Customer(Appl. Approved)



8. Conclusion

The functionalities are implemented in system after understanding all the system modules according to the requirements. Functionalities that are successfully implemented in the system are:

- Application for Loan containing all the necessary validation on field
- Login
- User authentication
- Logout
- Approval of Loan and Rejection of Loan by Administration
- View Customers and their Application list by admin
- Update Loan Rates
- Track Application Status
- Track payment details and pay according to the instalments
- EMI Calculator

After the implementation and coding of system, comprehensive manual testing was performed on the system to determine the errors and possible flaws in the system.

9. Limitations and Future Enhancements

We are able to implement the functionality model of the “Gold Loan FMS”. We aim to make this product ready to be used in live gold-loan markets. Currently, the project supports only one loan application be one customer.

The project can be extended to support multiple-loan applications by single customer.

We can enhance more as per future requirements.

10. Bibliography

Following links and websites were referred during the development of this project:

stackoverflow.com

docs.djangoproject.com