MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT AND ENGINEERING

Bank Loan Management System

Project Report

DATABASE MANAGEMENT SYSTEM: PROJECT 2021

Submitted by:

1. **Nihal Shetty (E050)**
2. **Khushi Vora (E061)**
3. **Arya Shah (E071)**

Submitted to: Prof. Ishani Saha

(Supervisor's signature)

……………………….

Prof. Ishani Saha

Date of submission: 05-04-2021

Table of Contents:

Abstract

The Loan Management System is important and helps to ensure success or failure of any credit institution. The scope of this project is to provide good communication and communication between the customer and the manager. The current system can be a user-friendly system, it can easily track information and contains the operation of fast-recovery information, such as customer data and all loan details. The system can make daily operations more efficient and provide faster response. Including adding, editing, retrieving customer information, maintaining and issuing new loans, changing the loan rate. It is also possible to apply for different kinds of loans such as vehicle loan, property loan, education loan, etc. This system will minimize errors while providing more control over the system and more robust management information in the form of implementation strategies. There are modular, scalable, and customizable components in this system that organizations can use for complete automation. It tries to replicate a Loan Management System which could be used by the banking sector.

List of figures

List of tables

The list of tables discussed throughout the Mini Project Report and the Bank Loan Management System are as follows:
Database File name storing all the tables- ‘LoanManager.db’

- Customer Table
- Loan Table
- AssignLoan Table
- RepayLoan Table

List of acronyms

The Project Report authors have tried their best to use as little acronyms as possible, to explain certain concepts, acronyms might have been used purposefully. The following list covers all the acronyms that the authors have used in the following report as well as the Relations present in the Bank Loan Management Database. The list of acronyms used are as follows:

GUI: Graphical User Interface
SQL: Structured Query Language
Re: Regular Expressions
CRUD: Create, Update, read, Delete
DOB: Date of Birth
EMI: Equated Monthly Installment
ER: Entity relation
PK: Primary Key
FK: Foreign Key
SSD: Solid State Drive
HDD: Hard Disk Drive
GHz: Gigahertz
Db: Database
NF: Normal Form
MB: Megabyte
GB: Gigabyte
UI: User Interface
SMS: Short Message Service

Chapter 1: Introduction to the system

1.1 Introduction

The project created by the group is a GUI based Loan Management System made using Tkinter in Python. The project is a desktop application developed using Python with the GUI framework Tkinter. For the backend database services, SQLite is used.

This system has been created to keep records about consumers who have taken out bank loans. An employee of the bank can add new customer accounts, manage old customer accounts and check other details. Other functionalities such as checking for a client's eligibility to get the loan and then accordingly, approving or denying the loan is also present. Once approved, the customer will receive the funds. There are also methods to evaluate the customer's loan amount, rate of interest and unpaid amount. This system has a simple easy-to-use user interface. Its many functionalities make it a useful tool. It can be the solution to many of the issues faced in the industry at large.

1.2 Problem Statement

To develop a system that will overlook the activities of transactions at a particular bank without manual processing. All transactions will be updated automatically using the information stored in the database record. Checking eligibility and other tasks too will be easier to carry out by using this system. The main motive behind this project is to develop a system which will be able to handle the overall tasks going inside the institutions without much effort.

1.3 Functional requirements of the system

1. Loan Product-
    - Create, Read operations for Loan Product.
    - Select loan type, check eligibility (age), apply fixed and floating interest rate
    - Fines - preclosing fine and late EMI fine

2. <u>Customer-</u>
    - Create, Read operations for Customer.
    - Customer details - customer ID, name, address, DOB, email, phone number
3. <u>Loan Assignment-</u>
    - Assignment of loan amount based on loan type, salary, fixed obligation values
    - Loan is granted only after eligibility check is done
4. <u>Loan Repayment-</u>
    - Update Loan Repayment amount as each EMI is made.
    - Late EMI fine is applied whenever late payment is made.

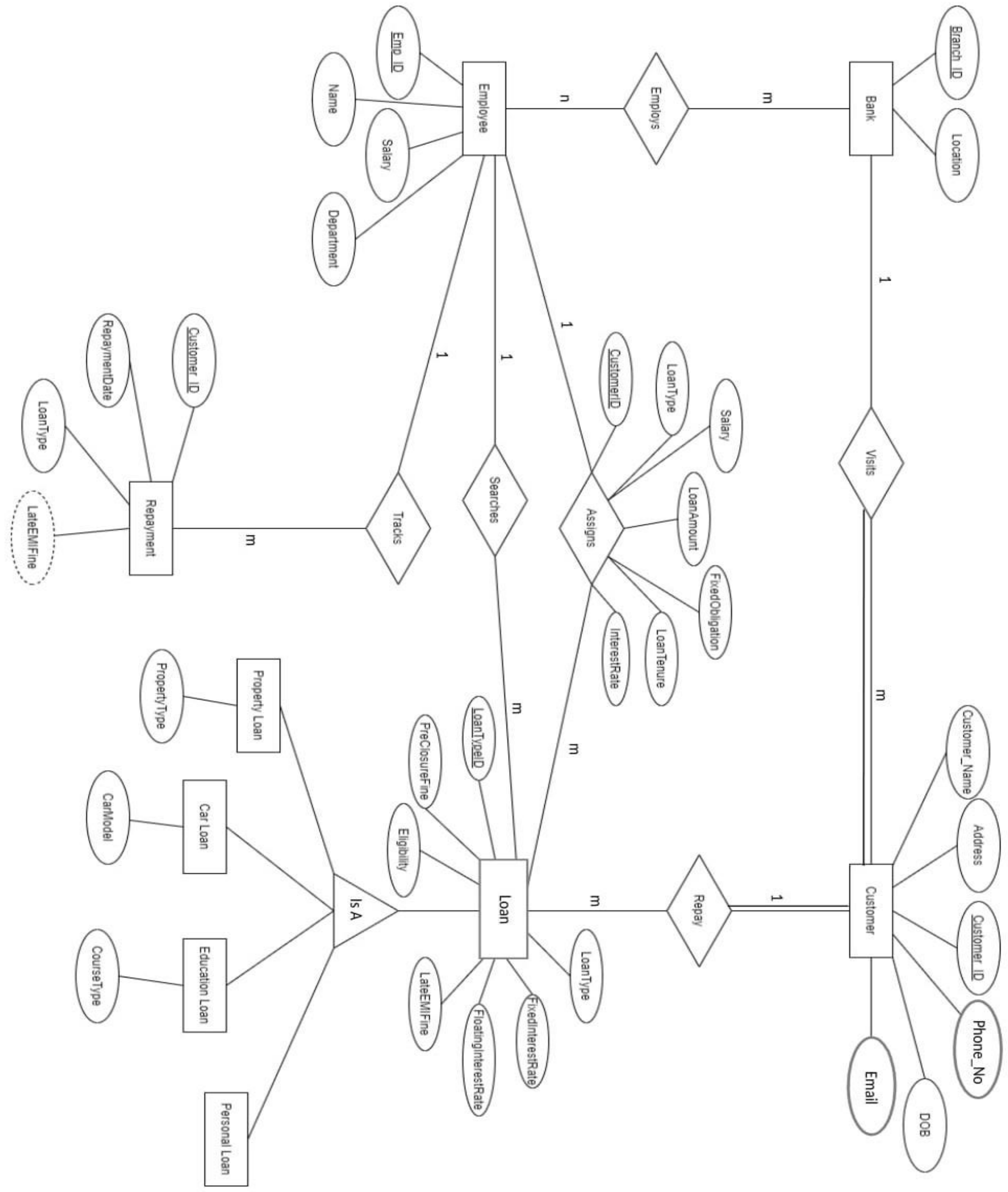
1.4 Users of the system

The users of the system primarily consist of employees of the bank namely:

- Clerks
- Mid Level Managers
- Senior Level Managers

Chapter 2: System Design and constraints

2.1 ER Model

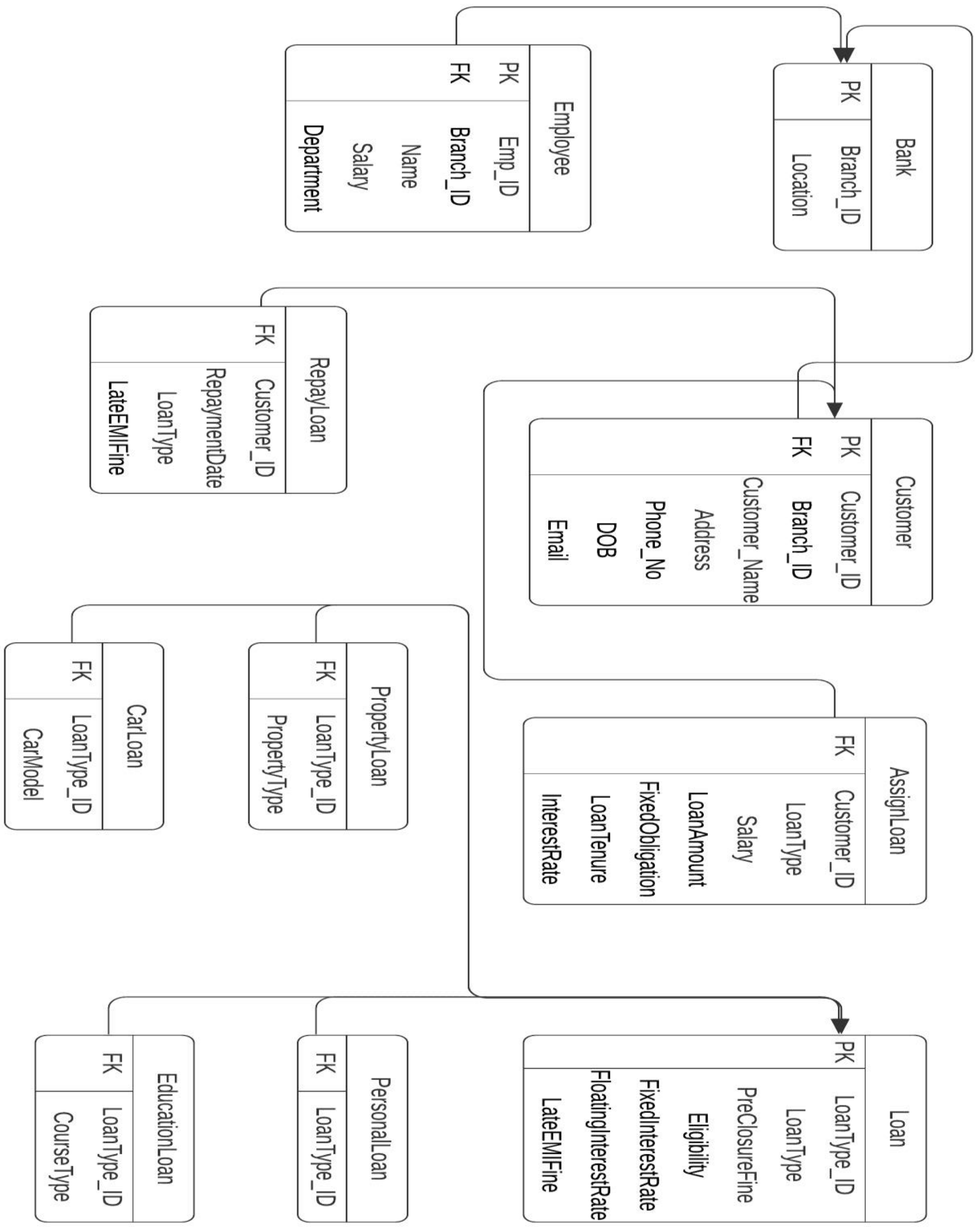


2.2 Reduction of ER model to Relational Model

The reduction to Relational Model is as follows:

1. Bank(Branch_ID, Location)
2. Employee(Emp_ID, Name, Salary, Department)
3. Customer(Customer_ID, Customer_Name, Address, Phone_No, DOB, Email)
4. Loan(Loantype_ID, LoanType, PreClosureFine, Eligibility, FixedInterestrate, FloatingInterestrate, LateEMIFine)
5. AssignLoan(Customer_ID, LoanType, Salary, LoanAmount, FixedObligation, LoanTenure, InterestRate)
6. RepayLoan(Customer_ID, RepaymentDate, LoanType, LateEMIFine)
7. PropertyLoan(LoanType_ID, PropertyType)
8. CarLoan(LoanType_ID, CarModel)
9. EducationLoan(LoanType_ID, CourseType)
10. PersonalLoan(LoanType_ID)

2.3 Schema Diagram



2.4 Constraints

Following are the constraints of the Bank Loan Management System:

- Only the Database Admin and the Employee having the admin rights can access the system
- In order to assign a loan to a customer, the customer should be already present in the customer table of the database
- There are total 4 types of Loans available for the customers: Car Loan, Educational Loan, Property Loan and Personal Loan
- In order for the repayment of loan the customer should exist in the customer table of the database and the loan must be assigned to the customer from the assign loan table of the database
- In order to Search loans, the respective loan type must be created and should be present in the loan table of the database.
- A loan can be assigned only if the customer is found to be eligible for the loan based on the entry fields. If ineligible, appropriate message should be displayed and decisions be taken
- Any entry which does not follow the particular format according to the regular expression defined will simply not reflect on the GUI window and get committed in the database hence appropriate care must be taken in following the correct format of the required entries to be filled
- The format of DOB for entries in the customer table is DD/MM/YYYY only and any other format will not reflect on the GUI window as well as in the Database
- The system requires the customer to be identified as an Indian national only since this database assumes the fact that the bank is Indian and all branches are present within India with no overseas connections whatsoever.
- The customers must provide valid documents and accept that the information provided by them in the entry sections of the GUI window are valid to the best of their ability and all the documents are verified and all the financial conditions are met apart from the system detecting the customer as eligible or ineligible for the loan.

2.5 Normalization techniques applied on relational model

The relational model of the Bank Loan Management System for reference is as follows:

Bank(Branch_ID, Location)
Employee(Emp_ID, Name, Salary, Department)
Customer(Customer_ID, Customer_Name, Address, Phone_No, DOB, Email)
Loan(Loantype_ID, LoanType, PreClosureFine, Eligibility, FixedInterestrate, FloatingInterestrate, LateEMIFine)
AssignLoan(Customer_ID, LoanType, Salary, LoanAmount, FixedObligation, LoanTenure, InterestRate)
RepayLoan(Customer_ID, RepaymentDate, LoanType, LateEMIFine)

PropertyLoan(LoanType_ID, PropertyType)
CarLoan(LoanType_ID, CarModel)
EducationLoan(LoanType_ID, CourseType)
PersonalLoan(LoanType_ID)

A. Since none of the Relations have multivalued attributes, hence the given relational model satisfies 1NF Normalization

Conclusion: The Relation is said to be in 1NF.

B. In the 2NF, relational must be in 1NF. In the second normal form, all non-key attributes are fully functional dependent on the primary key

In the given relational model, all the relations are fully functional dependent on the primary key:
Bank: Primary Key- Branch_ID
Employee: Primary Key- Emp_ID
Customer: Primary Key- Customer_ID
Loan: Primary Key- Loantype_ID
AssignLoan: Foreign Key- Customer_ID
RepayLoan: Foreign Key- Customer_ID

PropertyLoan: Foreign Key- LoanType_ID
CarLoan: Foreign Key- LoanType_ID
EducationLoan: Foreign Key- LoanType_ID
PersonalLoan: Foreign Key- LoanType_ID
Since all the conditions satisfy, Hence the relational model is said to be in 2NF Form.

C. A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency. 3NF is used to reduce the data duplication. It is also used to achieve data integrity. If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.
   A relation is in third normal form if it holds at least one of the following conditions for every non-trivial functional dependency X → Y.

   - X is a super key.
   - Y is a prime attribute, i.e., each element of Y is part of some candidate key.

Since all the rules for 3NF are verified for the relational model under consideration hence the relational model is said to be in 3NF Form.

Chapter 3: Implementation

3.1 Hardware and Software details

The Bank Loan Management System consists of two components namely Frontend and Backend. The Frontend part deals with the Graphical user interface and the Backend part deals with the connection to the database and storage and retrieval of data from the database.

Hardware Specifications:
The hardware interface for the project should be a Personal Computer/Laptop with a Windows 10 Operating System. There are no specific hardware component requirements since our project is not a memory intensive or high storage consuming in nature. The purpose of this PC/Laptop is to provide information of the data entered by the user.
The minimum system requirements needed for the software to run are as follows:

● Processor: 1 gigahertz (GHz) or faster processor
● Hard Disk Space (HDD/SSD): 500 MB
● RAM: 1GB and above
● Display:800 x 600
● Graphics: 512MB with DirectX 9.0 and later versions

Display Unit:
A Display unit is necessary for the input and output operations. It is used to display greetings to welcome the user, help him/her with the interface, display information about the software and to
Software Requirements Specification for Digit Recognition System Page 17
accept the input and display the output that has been obtained based on it. It can also display video/audio media as a form of entertainment for the user.

Printer (Optional):
The Printer can print the transcripts of the user interaction and final output that was obtained in a standard format.

Software Specifications consisting of The Frontend and Backend part of the system is discussed in details as follows:

A. Frontend Details

- The Frontend or the Graphical User Interface of the Bank Loan Management system has been made using the Python programming language and supported graphical user interface framework Tkinter
- The Frontend makes use of the concept of switching frames in Tkinter Window where each functionality opens a new frame within the system
- The dimensions of the user interface are set at 400x400 (height x width) and the background is given a shade of blue colour.
- The frontend of the Loan Management System also has the feature of pop up error messages to be displayed in case of any error from the system backend or user input generated error

B. Backend Details

- The backend of the Bank Loan Management System makes use of the Python Programming language with the database connectivity facilitated by the SQLite3 Library in Python and the use of Regular Expressions to maintain the correct passage of information through user input has been incorporated. The backend also makes use of the datetime library to keep record of the time of database entries made.
- The Backend automatically creates a .db database file in case not found on the system and starts the creation of tables as soon as the user starts interacting with the system and adds tuples in the database relations once the connection to the database is established.
- In order to connect the backend to the database and the frontend the concept of cursors was used
- Each table functionality was made into a module and cursors for each were called for the respective module. Throughout the program code, the concept of Object Oriented Programming has been maintained
- Once the cursor is created, the required tables were created and in order for the changes to reflect in the database, a commit was done for every relation created.
- Queries for each relation have been added with the cursor execute method and after each completion of the module the cursor is closed.

- The buttons on the homepage lead to the change of frames in the GUI window and each entry filled in the GUI layout by the user passes through the validation check with the help of regular expressions and relevant error messages are shown in case the entry made by the user is incorrect

3.2 Tools or library used

The Bank Loan Management System has been constructed using the following libraries and frameworks:

1. Programming Language used: Python 3.8

The integrated development environment (IDE) used was Jupyter Notebook with Python 3.8 kernel.

2. Library for Graphical User Interface: Tkinter

The GUI for the system has been developed using Tkinter with additional features like messagebox from Tkinter for displaying error messages, warning messages and completion messages.

Syntax:
Syntax: import tkinter as tk
from tkinter import messagebox

3. Library for Database Connectivity: SQLite3

The database connectivity has been done with the help of SQLite3 Library for Python.

Syntax:
import sqlite3

4. Library for validation of entered strings: Regex

All the inputs entered by the user have been verified with the use of regular expressions or regex so that the data entered is in correct format which is accepted by the system

Syntax:

import re

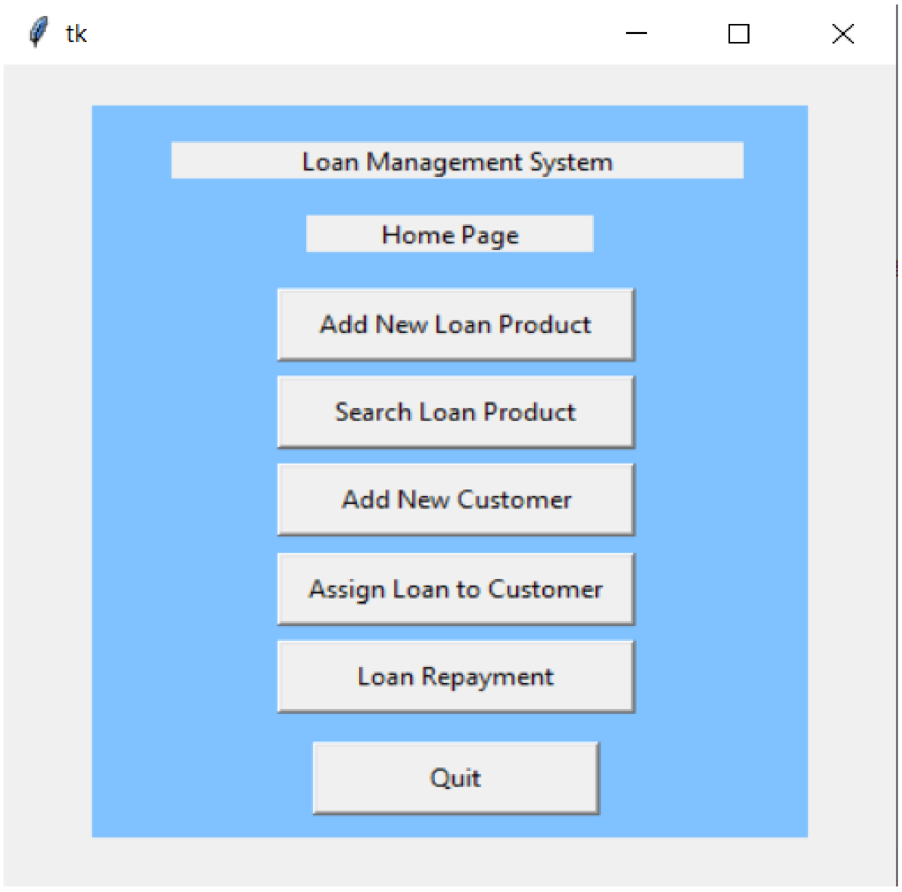
5. Library for Date and Time: datetime

Inbuilt library called datetime has been used for dealing with Date and Time inputs in the Bank Loan Management System

Syntax:
import datetime
from datetime import date

3.3 Screenshots and description

Figure A:

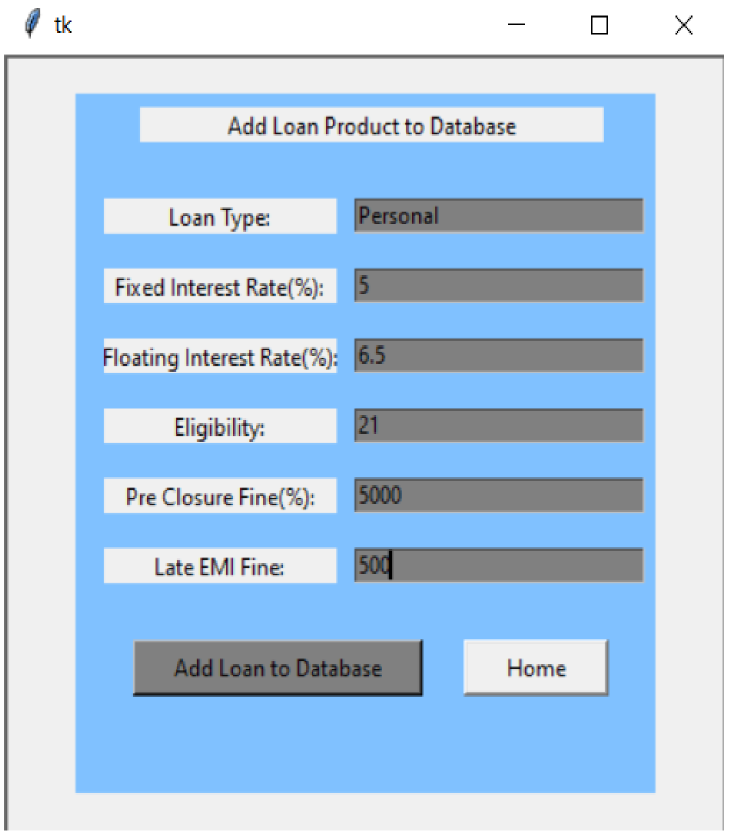


Description: The above image shows the home page of the Bank Loan Management System. The System has dimensions of 400x400 and on running the system the following GUI starts. The title is given as Loan Management System and the Home page as shown above consists of Buttons namely:

- Add New Loan Product
- Search Loan Product
- Add New Customer
- Assign Loan to Customer
- Loan Repayment
- Quit

Each button opens a new page in the existing system window for the respective functionality associated with it as discussed further.

Figure B:

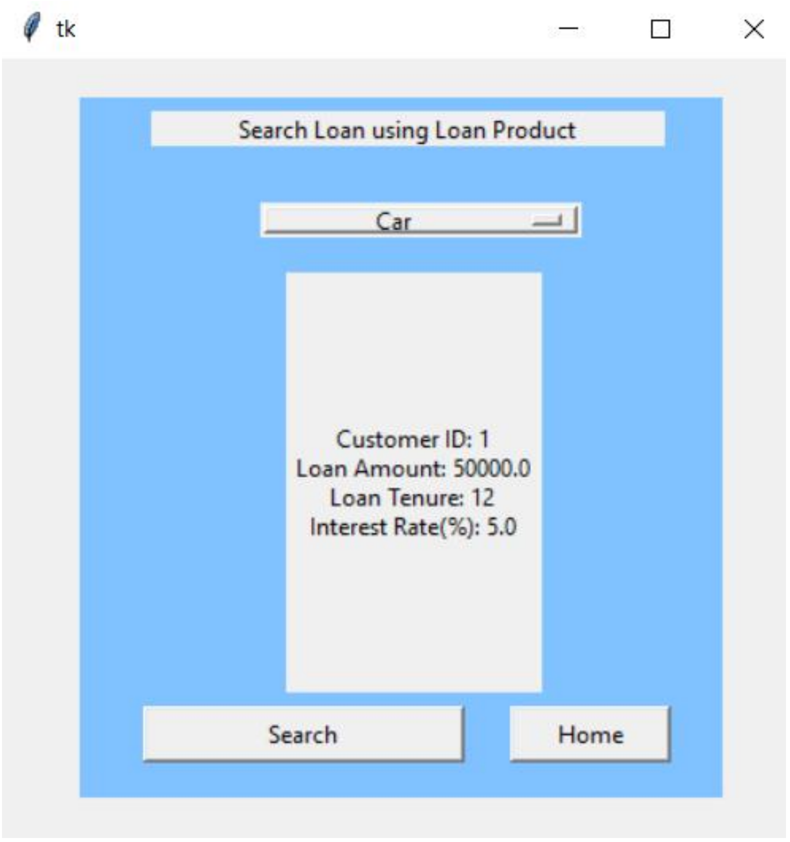


Loan Type:	Personal
Fixed Interest Rate(%):	5
Floating Interest Rate(%):	6.5
Eligibility:	21
Pre Closure Fine(%):	5000
Late EMI Fine:	500

Description: The above image shows the Add Loan Product to Database functionality. The System page shows certain fields that need to be entered and then the bottom left button can be clicked to add the Loan type to the Database. The fields that need to be entered while adding the loan product to the database is as follows:

- Loan Type
- Fixed Interest Rate
- Floating Interest Rate
- Eligibility
- Pre Closure Fine
- Late EMI Fine

Figure C:

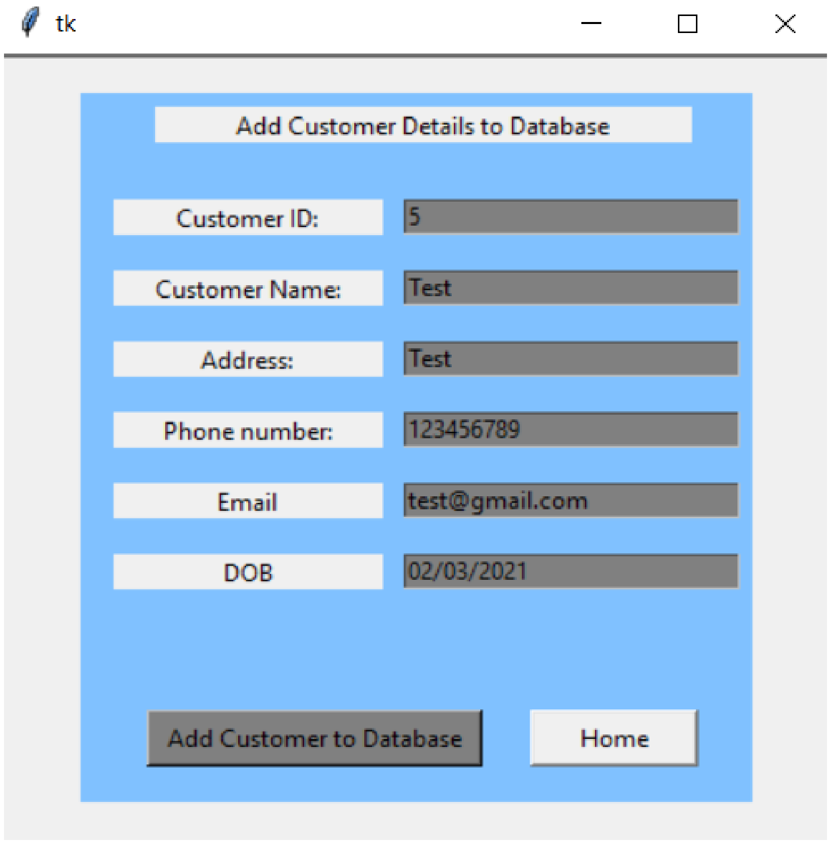


Customer ID: 1
Loan Amount: 50000.0
Loan Tenure: 12
Interest Rate(%): 5.0

The above figure shows the functionality of searching a loan using the Loan Product. On clicking on the Rectangle, a drop down menu of all the available loan products appears and on selecting a particular loan type, the details of that particular loan is displayed on the grey rectangular area. The items displayed on the search query are as follows:

- Customer ID
- Loan Amount
- Loan Tenure
- Interest Rate

Figure D:



tk

Add Customer Details to Database

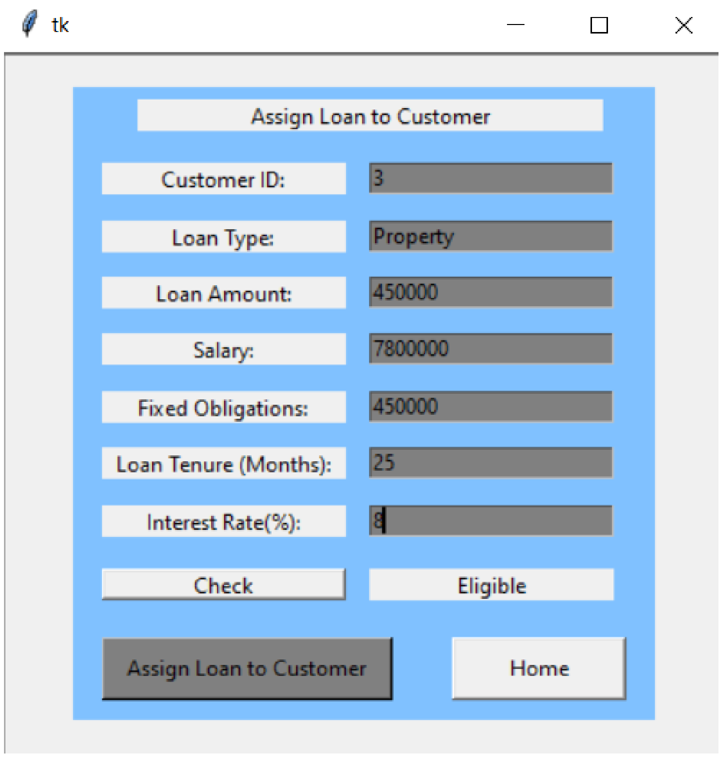
Customer ID:	5
Customer Name:	Test
Address:	Test
Phone number:	123456789
Email	test@gmail.com
DOB	02/03/2021

The above figure shows the functionality of the Add Customer Details to Database. With the help of this frame, the system user is able to add customers in the database who are potential customers needing a loan. The fields for entry in the database are as follows:

- Customer ID
- Customer Name
- Address
- Phone Number
- Email
- DOB

Finally there is a Add Customer to Database button which commits the entry in the Customer Table of the database.

Figure E:



tk

Assign Loan to Customer

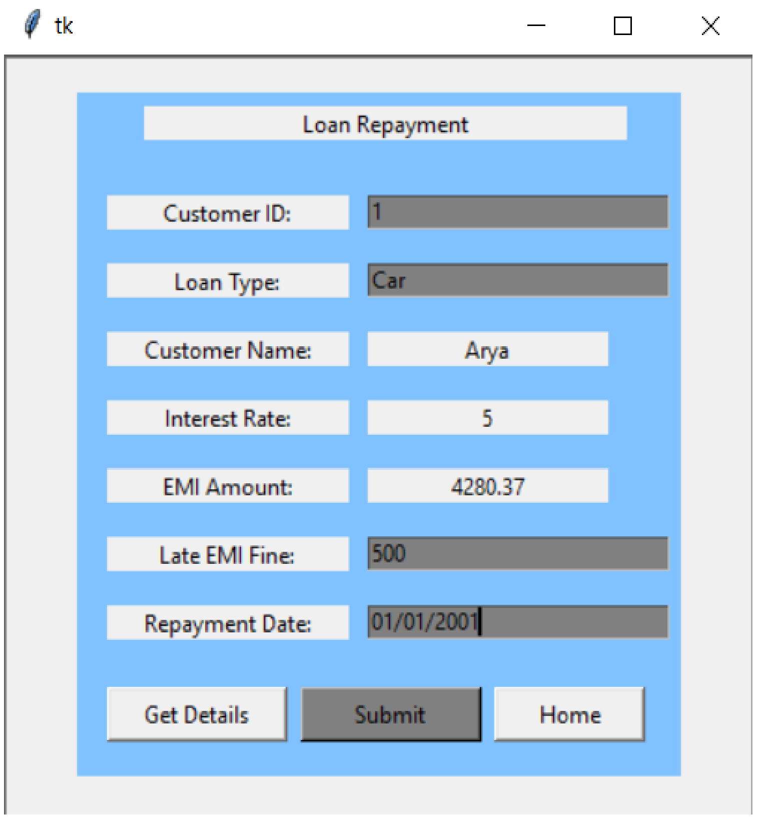
Customer ID:	3
Loan Type:	Property
Loan Amount:	450000
Salary:	7800000
Fixed Obligations:	450000
Loan Tenure (Months):	25
Interest Rate(%):	8
Check	Eligible

The above figure shows the Assigning loan to customer functionality, with the help of this functionality, the already registered customers from the customer table can be assigned loan to already added loan products from the loan product table. The entries in this frame of window are as follows:

- Customer ID
- Loan Type
- Loan Amount
- Salary
- Fixed Obligations
- Loan Tenure
- Interest Rate

Finally the “check” button also allows the system user to check whether the added details will lead to the customer being eligible or ineligible for the loan. Once the customer is fit to be eligible then the system user can make use of the 'Assign Loan to Customer' button and the loan will be assigned to that customer and the entry will reflect in the assign loan table of the database.

Figure F:



tk

Loan Repayment

Customer ID:	1
Loan Type:	Car
Customer Name:	Arya
Interest Rate:	5
EMI Amount:	4280.37
Late EMI Fine:	500
Repayment Date:	01/01/2001

Finally, the above figure shows the Loan repayment functionality whereby the repayment of assigned loans is recorded and carried out. The entries for this frame are as follows:

- Customer ID
- Loan Type
- Late EMI Fine
- Repayment date

At the same time, 3 other fields are displayed based on the early 4 entries as stated above, these three fields help identify the customer name, the total EMI amount to be paid and the Interest rate which was charged. The “Get Details” button helps retrieve that information.

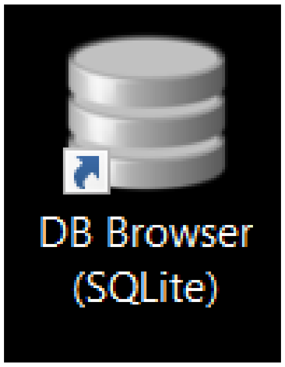
Once the details have been found, the user of the system can proceed to click on the “Submit” button and the loan repayment will be successful and the same entry will reflect in the Loan repayment table of the database.

From this point the user of the system can go back to the Home page, and repeat the process or simply quit the system.

3.4 Database structure

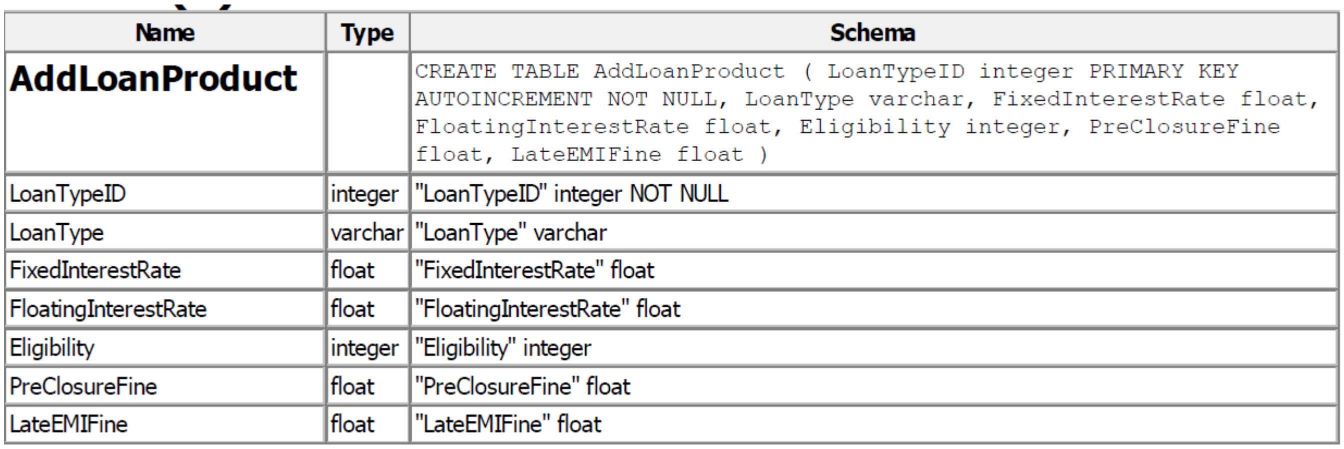
The database structure of the whole project can be viewed from the database file created named as “LoanManager.db”. The SQLite package that is used also has a dedicated SQL database browser through which the database structure can be clearly viewed. In this section the database structure shall be discussed along with the screenshots of the overall database tables and individual tables with the tuples entered so far.

Software used: DB Browser(SQLite)



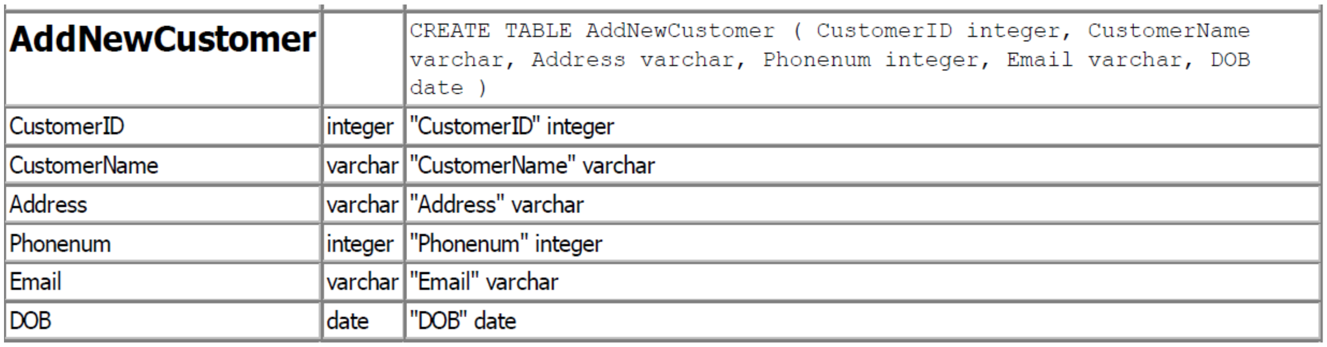
The following software is a database browser for SQLite wherein one can browse through the database file namely the .db files.
One can also view the overall database structure and view all the entries in individual tables of the database.
Here, we make use of the DB Browser to view the overall database structure of “LoanManager.db” and also view all the relations in the database and verify live entries made through the Graphical User Interface.

Overall Database Structure:
TABLE 1:



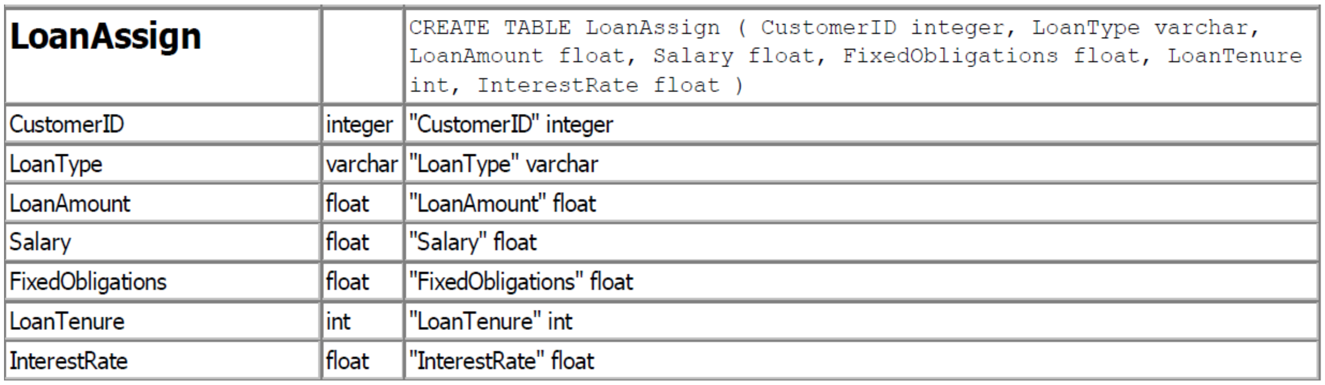
Name	Type	Schema
AddLoanProduct		`CREATE TABLE AddLoanProduct ( LoanTypeID integer PRIMARY KEY AUTOINCREMENT NOT NULL, LoanType varchar, FixedInterestRate float, FloatingInterestRate float, Eligibility integer, PreClosureFine float, LateEMIFine float )`
LoanTypeID	integer	"LoanTypeID" integer NOT NULL
LoanType	varchar	"LoanType" varchar
FixedInterestRate	float	"FixedInterestRate" float
FloatingInterestRate	float	"FloatingInterestRate" float
Eligibility	integer	"Eligibility" integer
PreClosureFine	float	"PreClosureFine" float
LateEMIFine	float	"LateEMIFine" float

TABLE 2:



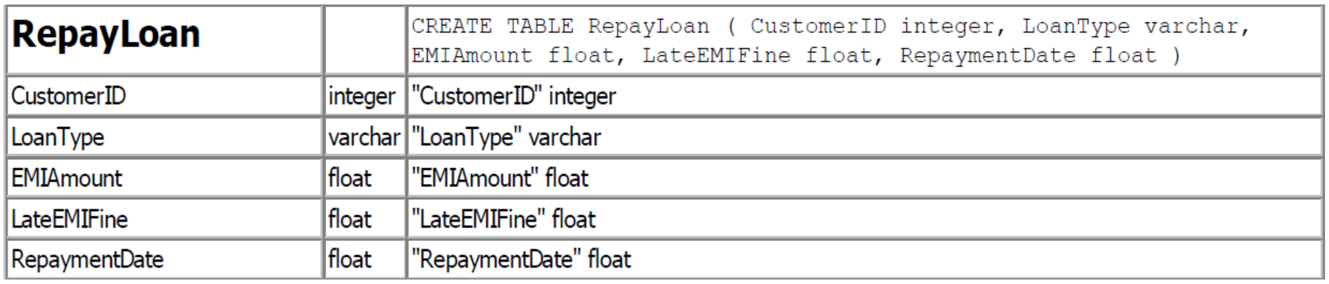
AddNewCustomer		`CREATE TABLE AddNewCustomer ( CustomerID integer, CustomerName varchar, Address varchar, Phonenum integer, Email varchar, DOB date )`
CustomerID	integer	"CustomerID" integer
CustomerName	varchar	"CustomerName" varchar
Address	varchar	"Address" varchar
Phonenum	integer	"Phonenum" integer
Email	varchar	"Email" varchar
DOB	date	"DOB" date

TABLE 3:



LoanAssign		`CREATE TABLE LoanAssign ( CustomerID integer, LoanType varchar, LoanAmount float, Salary float, FixedObligations float, LoanTenure int, InterestRate float )`
CustomerID	integer	"CustomerID" integer
LoanType	varchar	"LoanType" varchar
LoanAmount	float	"LoanAmount" float
Salary	float	"Salary" float
FixedObligations	float	"FixedObligations" float
LoanTenure	int	"LoanTenure" int
InterestRate	float	"InterestRate" float

TABLE 4:



RepayLoan		`CREATE TABLE RepayLoan ( CustomerID integer, LoanType varchar, EMIAmount float, LateEMIFine float, RepaymentDate float )`
CustomerID	integer	"CustomerID" integer
LoanType	varchar	"LoanType" varchar
EMIAmount	float	"EMIAmount" float
LateEMIFine	float	"LateEMIFine" float
RepaymentDate	float	"RepaymentDate" float

Chapter 4: Conclusion and Future work

To conclude, the Loan Management System proves to be an effective tool to manage and provide loans to the customers that visit the bank. Bank Employees can easily operate the software owing to its friendly User Interface and robust Back-End code. The software reduces the amount of data import done manually and provides increased performance. Its program is very friendly and can be easily used by anyone. It also reduces the time it takes to document customer information and other data.

Future Work:
Based on the observations of this project, we have suggested a few implementations as future work so as to improve the effectiveness of the program:

1. Implementation of Multiple entries for Customer Details: The Bank Employee can fill in multiple phone numbers, email addresses and other personal information of the customer.
2. Enhanced UI: User Interface of the system could be updated to match with present-day design trends and features.
3. Online Transactions: Bank Employees can use the program to perform transactions online.
4. SMS Service: The program could be used to send weekly/monthly SMS reminders to inform customers about upcoming or pending payments as their deadlines are approaching.
5. Cloud Support: A distributed database could be added as an advanced functionality so that all the bank branches can communicate with each other and have access to a common database relation.

THANK YOU