**INDUSTRIAL TRAINING**

Submitted in partial fulfilment of the

Requirements for the award of the degree

Of

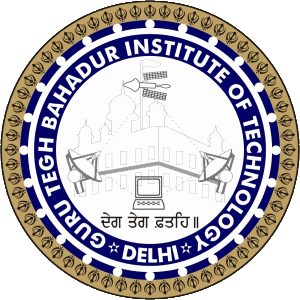
Bachelor of Technology

In

Computer Science & Engineering

  By:

Bhavnoor Singh **(**01813202717/CSE1**)**



Department of Computer Science & Engineering Guru Tegh Bahadur Institute of Technology

Guru Gobind Singh Indraprastha University Dwarka, New Delhi

Year 2017-2021

**BANKING AUTOMATION SYSTEM**

**USING PYTHON**

Duration

3rd June, 2019 – 17th July, 2019

By:

Bhavnoor Singh (01813202717/CSE1)

At

TechSquare Solutions Pvt. Ltd.

Ferozepur, India

**DECLARATION**

I hereby declare that all the work presented in this Industrial Training Report for the

Partial fulfilment of the requirements for the award of the degree of Bachelor of

Technology in Computer Science & Engineering, Guru Tegh Bahadur Institute of

Technology, affiliated to Guru Gobind Singh Indraprastha University Delhi is an

authentic record of our own work carried out at TechSquare Solutions Pvt. Ltd. Ferozepur City, Punjab, India from 3rd June, 2019 to 17th July, 2019.

Date:

Bhavnoor Singh

(01813202717 /CSE1)

**CERTIFICATE**

****

****

**ACKNOWLEDGEMENT**

I would like to express our great gratitude towards Er. Puneet Aggarwal who has given us support and suggestions. Without their help we could not have presented this work upto the present standard. We also take this opportunity to give thanks to all others who gave us support for the project or in other aspects of our study at Guru Tegh Bahadur Institute of Technology.

Bhavnoor Singh (01813202717 /CSE1)

Date:

bhav27noor@gmail.com

**ABSTRACT**

Requirements definition and management is recognized as a necessary step in the delivery of successful systems and software projects, discipline is also required by standards, regulations, and quality improvement initiatives. Creating and managing requirements is a challenge of IT, systems and product development projects or indeed for any activity where you have to manage a contractual relationship..The impact of a poorly expressed requirement can bring a business out of compliance or even cause injury or death.

The BANKING AUTOMATION SYSTEM undertaken as a project is based on relevant technologies. The main aim of this project is to develop software for bank management system. This project has been developed to carry out the processes easily and quickly, which is not possible with the manuals systems, which are overcome by this software. This project is developed using Python language and hence it provides the complete solution for the current management system. The main purpose of this software is to reduce the manual errors involved in the banking management process and make it convenient for the bank administrators to perform various banking operations. They can utilize this software to open, modify record/ bank accounts and perform various transactions.

**LIST OF FIGURES AND TABLES**

|  |  |  |
| --- | --- | --- |
| **Figure/ Table No.** | **Figure Name** | **Page No.** |
| Table 1 | Tkinter Widgets | 6-7 |
| Fig: 1.1 | Most-In- Demand Programming Language | 8 |
| Fig: 1.2 | PyMySQL | 9 |
| Fig: 1.3 | Waterfall Model Diagram | 17 |
| Fig: 1.4 | Use Case Diagram | 17 |
| Fig: 1.5 | Entity Relationship Diagram | 19 |
| Fig: 1.6 | Entity Relationship Diagram | 22 |

**CONTENTS**

**Chapter Page No**.

**Title Page** i

**Declaration and Certificate** ii-iv

**Acknowledgement** v

**Abstract** vi

**Tables and figures** vii

1. **introduction** 1-3
   1. About Project 2
   2. Objectives 2
   3. Purpose 3
   4. Scope of the project 3
2. **Development tools and Technologies** 4-10

2.1 Python 3 5

2.1.1 Tkinter Python 5-7

2.2 MySQL 8-9

2.3 Software System Attributes 9-10

1. **System Analysis** 11-19

3.1 SDLC 15-16

3.2 Use Case Diagram 17-18

3.3 Data Flow Diagram 18-19

3.4 ER Diagram 19

**4. Specific Requirements** 20-23

4.1 External interface requirements 21

4.1.1 User interface requirements 21

4.1.2 Hardware interface requirements 21

4.1.3 Software interface requirements 21

4.2 Software product feature 22

4.2.1 Functional requirements 22

4.2.2 Non-functional requirements 23

* 1. Database Requirements 23

**5. System design** 24-29

5.1 Design methodologies 27

5.2 Structured design 27

5.3 Module coupling 27-28

* 1. Module cohesion 28-29

**6. Conclusion & Scope of the project** 30-31

**References** 32-33

**Appendix A (Screenshots)** 34-43

**Appendix B (Source Code)** 44-60

**1. INTRODUCTION**

This project on Banking Management System is the automation of banking system. The system is able to provide much information about the banking tasks which are performed daily. The system also allows us to add records whenever an account is created or modified or any transaction has occurred. For data storage and retrieval we have used MySQL databasein python. It enables us to add any number of records in our database.

**1.1 About Project**

The project named “Banking Administration System” is written in python 3.6, mainly because of its suitability for this type of application. Its user friendly nature and in-built documentation, complication, error detection, binding facilities and interaction with other software packages make it most powerful tool for software development. Moreover python consists of all the technologies that help in creating and running robust, scalable and distributed packages.Python is a general-purpose object-oriented programming language.

Assistance is provided to the user at each and every step so that no problem is faced during using it. Further the details of every process and the user manuals attached in the report make it very easy to understand. Every possible care has been taken to make the software and the report clear, simple and error free which makes it so special and one of its kind.

Project deals with:

 Customers’ Accounts

 Transactions

**1.2 Objectives of the Project**

To provide some amount of automation in banking mangement system.

To help banking system in making their business more efficient.

An added attraction for their potential customers.

It will also show the attitute of the management that they are aware to the newly

introduced technology and ready to adopt them.

1.3 Purpose of the project

Electronically handling of bank’s record to enhance the accuracy, flexibility,

reliability and to remove the human’s error.A Bank provides banking services for customers, generally with a recognized operating system.To provide accurate information about the addition, deletion and modification of bank accounts and transactions. To provide, efficient, accurate, reliable, fast, and robust structure that can handle any number of banking transactions.

**1.4 Scope of the project**

The project is based on “BANK MANAGEMENT SYSTEM”.

This software firm deals in developing software for its clients. This is

Bank Management software, using Python3.6 and My SQL Database , made for ease of clerks. TheProject “BANKINGAUTOMATION SYSTEM” is made for computerizing the banks. UsingComputer, the creation of records/accounts is easy task. Calculations and accounting may be donequickly and accurately by computer. We can store large amount of data on storage devicesavailable with computer and use this data in future when required. Updating and searching ofrecords is easy and quick in computerized system.Administrators or manager etc are able to access the database according to their privilegeafter Authentication using passwords.

**2**. **DEVELOPMENT TOOLS AND TECHNOLOGIES**

**2.1 PYTHON 3:**

Python is a general purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido Van Rossum during 1985-1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python is named after a TV show called ‘‘Monty Python’s Flying Circus’’. Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation.Python is a multi-paradigm programming language. Object-oriented programming and [structured programming](https://en.wikipedia.org/wiki/Structured_programming) are fully supported, and many of its features support functional programming and [aspect-oriented programming](https://en.wikipedia.org/wiki/Aspect-oriented_programming) (including by [metaprogramming](https://en.wikipedia.org/wiki/Metaprogramming)[]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-13-46)and [metaobjects](https://en.wikipedia.org/wiki/Metaobject) (magic methods)). Many other paradigms are supported via extensions, including design by contract and [logic programming](https://en.wikipedia.org/wiki/Logic_programming).[[50]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-17-50)Python uses [dynamic typing](https://en.wikipedia.org/wiki/Dynamic_typing) and a combination of [reference counting](https://en.wikipedia.org/wiki/Reference_counting) and a cycle-detecting garbage collector for [memory management](https://en.wikipedia.org/wiki/Memory_management). The language's core philosophy is summarized in the document *The Zen of Python* (*PEP 20*), which includes aphorisms such as:

* Beautiful is better than ugly.
* Explicit is better than implicit.
* Simple is better than complex.
* Complex is better than complicated.
* Readability counts.

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal. Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure.

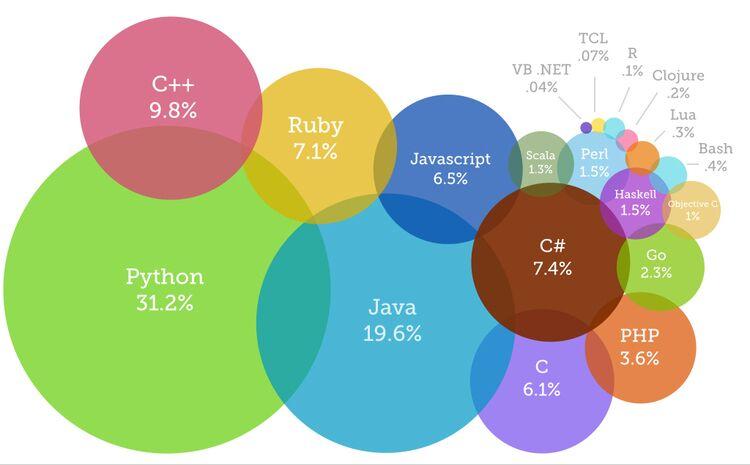
**2.1.1Tkinter Python**

Tkinter provides us variety of common GUI elements which can use to build our interface - such as buttons, menus and various kinds of entry fields and display areas. We call these elements widgets. We are going to construct a tree of widgets for our GUI – each widget will have a parent widget, all the way up to the root window of our application. For example :- a button or a text field needs to be inside some kind of containing window.

Tk is the class which we use to create the root window – the main window of application. Our application should have only one root, but it is possible for us to create other windows which are separate from the main window.

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| **1.** | [Button](https://www.tutorialspoint.com/python/tk_button.htm)  The Button widget is used to display buttons in your application. |
| **2.** | [Canvas](https://www.tutorialspoint.com/python/tk_canvas.htm)  The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application. |
| **3.** | [Checkbutton](https://www.tutorialspoint.com/python/tk_checkbutton.htm)  The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time. |
| **4.** | [Entry](https://www.tutorialspoint.com/python/tk_entry.htm)  The Entry widget is used to display a single-line text field for accepting values from a user. |
| **5.** | [Frame](https://www.tutorialspoint.com/python/tk_frame.htm)  The Frame widget is used as a container widget to organize other widgets. |
| **6.** | [Label](https://www.tutorialspoint.com/python/tk_label.htm)  The Label widget is used to provide a single-line caption for other widgets. It can also contain images. |
| **7.** | [Listbox](https://www.tutorialspoint.com/python/tk_listbox.htm)  The Listbox widget is used to provide a list of options to a user. |
| **8.** | [Menubutton](https://www.tutorialspoint.com/python/tk_menubutton.htm)  The Menubutton widget is used to display menus in your application. |
| **9.** | [Menu](https://www.tutorialspoint.com/python/tk_menu.htm)  The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton. |
| **10.** | [Message](https://www.tutorialspoint.com/python/tk_message.htm)  The Message widget is used to display multiline text fields for accepting values from a user. |
| **11.** | [Radiobutton](https://www.tutorialspoint.com/python/tk_radiobutton.htm)  The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time. |
| **12.** | [Scale](https://www.tutorialspoint.com/python/tk_scale.htm)  The Scale widget is used to provide a slider widget. |
| **13.** | [Scrollbar](https://www.tutorialspoint.com/python/tk_scrollbar.htm)  The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes. |
| **14.** | [Text](https://www.tutorialspoint.com/python/tk_text.htm)  The Text widget is used to display text in multiple lines. |
| **15.** | [Toplevel](https://www.tutorialspoint.com/python/tk_toplevel.htm)  The Toplevel widget is used to provide a separate window container |
| **16.** | [Spinbox](https://www.tutorialspoint.com/python/tk_spinbox.htm)  The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values. |
| **17.** | [PanedWindow](https://www.tutorialspoint.com/python/tk_panedwindow.htm)  A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically. |
| **18.** | [LabelFrame](https://www.tutorialspoint.com/python/tk_labelframe.htm)  A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts. |
| **19.** | [tkMessageBox](https://www.tutorialspoint.com/python/tk_messagebox.htm)  This module is used to display message boxes in your applications |

**Table 1: Tkinter Widgets**



**Fig: 1.1 Most-In- Demand Programming Language**

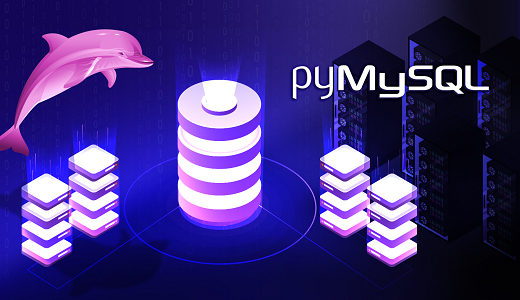
**2.2 MYSQL:**

MySQL was created by a Swedish company, MySQL AB, founded by David Axmark, Allan Larsson and Michael "Monty" Widenius. Original development of MySQL by Widenius and Axmark began in 1994. The first version of MySQL appeared on 23 May 1995. It was initially created for personal usage from mSQL based on the low-level language ISAM, which the creators considered too slow and inflexible. They created a new SQL interface, while keeping the same API as mSQL. By keeping the API consistent with the mSQL system, many developers were able to use MySQL instead of the (proprietarily licensed) mSQL antecedent.

MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer. MySQL works on many system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, macOS, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists.

In 2010, when Oracle acquired Sun, Widenius forked the open-source MySQL project to create MariaDB.

MySQL is a component of the LAMP web application software stack (and others), which is an acronym for *Linux, Apache, MySQL, Perl/PHP/Python*. MySQL is used by many database-driven web applications, including Drupal, Joomla, phpBB, and WordPress.



**Fig: 1.2 PyMySQL**

Python has an in-built support for SQLite. In this section, we would learn all the concepts using MySQL. MySQLdb module, a popular interface with MySQL is not compatible with Python 3. Instead, we shall use PyMySQL module. PyMySQL is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2.0 and contains a pure-Python MySQL client library. The goal of PyMySQL is to be a drop-in replacement for MySQLdb.

## 2.3 Software System Attributes

**Reliability:** When user wants to call the system over a given period of time, the system should correctly deliver services as expected by the user. The reliability of the system shall be good if it delivers services as specified.

**Availability:** When the system has any request at any given time, system should be available, it should be up and running and able to deliver useful service at this time. The availability of the system shall be good if it delivers services when it is requested. Otherwise, if requests are not responded at any given time then it implies bad availability.

**Security:** The system should resist accidental or deliberate intrusions, when users operate on the system. If the system should not resist accidental or deliberate intrusions, then important data – such as credit card number, id number, username, etc. – which belongs to user, shall be stolen by hacker. Thus, security of the system shall be low and trust of users shall be ruined. So, security of the system is very important for users.

**Maintainability:** When the system is used, new requirements may emerge. When these requirements are emerged, the system should be changeable to accommodate these requirements for maintaining the usefulness of the system.

**Performance**: The system should use the minimum part of memory. The processes of the system should use the processor most efficiently. User should finish operation in the least time interval.

**Reparability**: When the system is used, system failures are inevitable. The disruption caused by failure can be minimized if the system can be repaired quickly. So, the system should be possible to find the problem, access the component that failed.

**3. System Analysis**

**An Overview to system analysis**

The system analysis phase is considered to be one of the most important phases in the system development life cycle. It is immensely important that the software developer make through study of the existing system. Thorough study of the system is made and need i.e. features that are critical to system success and users wants (i.e. features that would be good but not essential) are brought out. The study will enable the developer to know the intricacies of the existing system.

Requirement analysis is done in order to understand the problem which the S/W system is to solve e.g., the problem could be automating the existing manual system or developing a completely new automated system or a combination of the two. For large systems having a large number of features and the need to perform many different tasks, understanding the requirement of the system is a major task. The emphasis in requirement analysis is on identifying what is needed from the system, and not how the system achieves its goal.

The main objective behind any business organization is to maximize its profit besides maintaining quality and strategic norms. This can be achieved by improving the efficiency of the system by providing more facilities using automation, by adopting faster data access, proper communication, whereas the main objective behind automation is not only to maximize profit but also to take care of passenger’s interest by providing them better facilities.

In this project we have used Rapid Application Development (RAD) model. RAD is an incremental software development process model that emphasizes an extremely short development cycle. The following phases are encompassed:

* **Business modelling:** All the information about the business functioning of the Banking department is collected, how the data and information is flow from one end to another end using the following questions: What information drives the department process? What information is generated? Who generates it? Where does the information go? Who process it?
* **Data modelling:** The information collected in Business modelling phase is refined into a set of data objects that are needed to support the project. The attributes of each object are identified and the relationships between these objects defined.
* **Process modelling:** Processing descriptions and functions like adding, modifying, deleting records, printing reports, providing information, file handling etc. are created.
* **Application generation:** The fourth generation techniques are used to generate application, like reusing the predefined functions or creating reusable components.
* **Testing:** Most of the functions are already tested, as they are predefined functions. However, new components or functions are also tested after application generation.

The development of a computer-based information system often comprises the use of a systems analyst. When a computer-based information system is developed, systems using computer hardware/software), what the system would be used for etc.analysis would constitute the following steps:

* The development of a feasibility study, involving determining whether a project is

economically, socially, technologically and organizationally feasible.

* Conducting fact-finding measures, designed to ascertain the requirements of the

system's end-users. These typically span interviews, questionnaires, or visual

observations of work on the existing system.

* Gauging how the end-users would operate the system.
* It refers to the process of examining a business situation with the intent of improving itthrough better procedures and methods. Systems development can generally be thoughtof as having two major components: Systems Analysis and Systems Design.
* Systems design is the process of planning a new system or replace or complement anexisting system. But before this planning can be done, we must thoroughly understandthe existing system and determine how computers can best be used to make itsoperation more effective. Systems analysis, then, is the process of gathering andinterpreting facts, diagnosing problems and using the information to recommend

improvement to the system. In brief, we can say that analysis specifies what the systemshould do. Design states how to accomplish the objective.

* Analysis is a detailed study of the various operations performed by a system and theirrelationships within and outside of the system. A key question is: What must be doneto solve the problem? One aspect of analysis is defining the boundaries of the systemand determining whether or not a candidate system should consider otherrelatedsystems. During analysis, data are collected on the available files, decision points andtransactions handled by the present system. Thus, in systems design, we movefrom the logical to the physical aspects of the life cycle.
* The decision to acquire computer hardware or software must be handled in the sameway as any other business decision. The variety of sizes and types of computingresources available puts a burden on the analyst who must select suitable hardware,software or services and advise the top management accordingly.
* Today, selecting a system is a serious and time-consuming business. The time spent on the selection process is a function of the applications and whether the system is a basicmicro- computer or a mainframe. In either case, planning system selection andacquiring experienced help where necessary pay off in the long run.There are various important factors, which should be considered prior to systemselection.

They are:

* Define system capabilities that make sense for the business.
* Specify the magnitude of the problem, i.e., clarify whether selection entails a few

peripherals or a major decision concerning the mainframe.

* Assess the competence of the in-house staff.
* Hardware and software should be considered as a package.
* Develop a time frame for the selection process.
* Provide user indoctrination.

This is crucial, especially for first-time users. Selling the system to the user staff, provide adequate training and creating an environment conductive to implementation are prerequisites for system acquisition.

The selection process should be viewed as a project and a project team should be formed with the help of management. The selection process consists of several steps, which are discussed below:

* **Requirements analysis:** The first step in selection understands the user's requirement within the framework of the organization’s objectives and the environment in which the system is being installed.
* **System specifications:** System specifications must be clearly defined. These specifications must reflect the actual applications to be handled by the system and include system objectives, flowcharts, input-output requirements, file structure and cost.
* **Request for proposal:** After the requirement analysis and system specifications havebeen defined, a request for proposal is prepared and sent to selected vendors for

bidding.

* **Evaluation and validation:** The evaluation phase ranks various vendor proposals anddetermines the one best suited to the user's requirements. It looks into items such asprice, availability and technical support. System validation ensures that the vendor

can, in fact, match his/her claims, especially system performance.

* **Vendor selection:** This step determines the vendor with the best combination of

reputation, reliability, service record, training, delivery time, lease/finance terms. The

selected vendors are invited to give a presentation of their system. The system

chosen goes through contract negotiations before implementation.

### WORKING OF THE PROJECT

* Admin/Manager will login first and can perform various banking tasks.
* Admin/Manager can open, modify, search, etc. the bank accounts by clicking on account option.
* Admin/Manager can do various types of transactions by selecting transaction option .
* Admin/Manager can edit profile, security settings and logout using admin option.

**3.1 Software development life cycle (SDLC)**

In this project we have followed the Waterfall model.The waterfall model is the most familiar model. This model has five phases:

requirements analysis and specifications, design, implementation and unit testing,integration and system testing, and operation and maintenance.

**1. Requirements Analysis and Specification Phase**: The goal of this phase is to

understand the exact requirements of the customer and to document them properly.This activity is usually executed together with the customer, as the goal is to documentall functions, performance and interfacing requirements for the software. Therequirements describes the “what” of a system, not the “how”.

**2. Design phase**: The goal of this is to transform the requirements specification into astructure that is suitable for implementation in some programming language.

**3. Implementation and Unit Testing Phase**: During testing, the major activities arecentered around the examination and modification of the code. Initially, small modulesare tested in isolation from the rest of the software product. There are problemsassociated with testing a module in isolation. How do we run a module withoutanything to call it, to be called by it or, possibly, to output intermediate values obtainedduring execution? Such problems are solved in this phase and modules are tested afterwriting some overhead code.

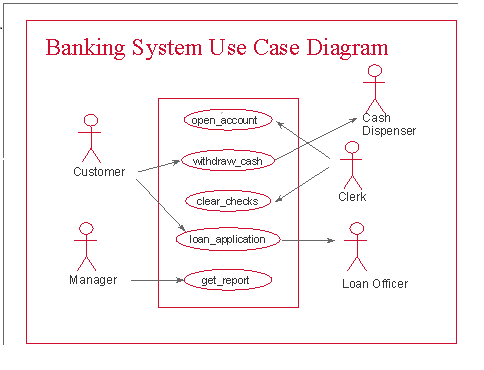
**4. Integration and System Testing Phase:** The purpose of unit testing is to determinethat each independent module is correctly implemented. This gives little chance todetermine that the interface between modules is also correct, and for this reasonintegration testing is performed. System testing involves the testing of the entire systemwhereas software is a part of the system. This is essential to build confidence in thedevelopers before software is delivered to the customer or released in the market.

**5. Operation and Maintenance Phase:** Software maintenance is a task that everydevelopment group has to face, when the software is delivered to the customer’s site,installed and is operational. Therefore, release of software inaugurates the operationand maintenance phase of the life cycle. The time spent and effort required to keep thesoftware operational after release is very significant.



**Fig: 1.3 Waterfall Model Diagram**

**3.2 USE CASE DIAGRAM**



**Fig: 1.4 Use Case Diagram**

**3.3 DATA FLOW DIAGRAM**

Data flow diagrams are commonly used during problem analysis. Data flow diagrams are quite general and not limited to problem analysis for software requirement specification. A DFD shows the flow of data through a system. It views a system a function that transforms the inputs into desired outputs. Any complex system does not perform this transformation into a single step and a data will typically undergo a series of transformation before it becomes an output. The DFD aims to capture the transformations that take place within a system to the input data so that eventually the output data is produced.

The agent that performs the transformation of data from one state to another is called a process. So, a DFD shows the movement of data through the different transformations or processes in the system. Named circles show the processes and data named arrows entering or leaving the bubbles represent flows.

Process Activity

The rectangle represents a source and sink and is a net originator or consumer of data. A source or sink is typically outside the main system of study.

Originator or Consumer of data

All external files are shown as a labeled straight line.

File name

The need of multiple data flows by a process is represented by a “\*” between the data flows.the symbol represents the AND relationship.for example, if there is a “\*” between the two input data flows A and B for a process,it means that A AND B are needed for the process.

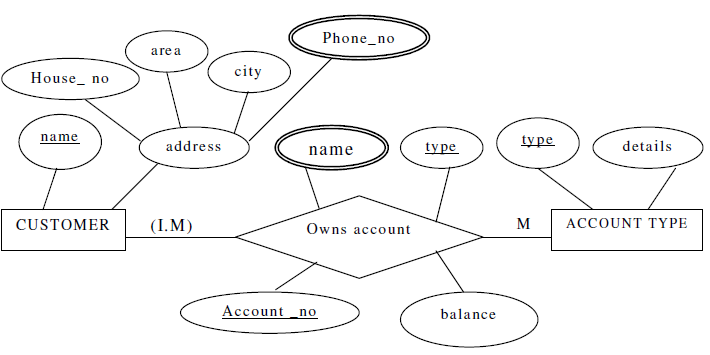
A

\*

B

**3.4 ER DIAGRAM**

An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes. If the application is primarily a database application, the entity-relationship approach can be used effectively for modeling some parts of the problem. The main focus in ER modeling is the Data Items in the system and the relationship between them. It aims to create conceptual scheme for the Data from the user’s perspective. The model thus created is independent of any database model. The ER models are frequently represented as ER diagram. Here we present the ER diagram of the Banking Automation System.



**Fig: 1.5 Entity Relationship Diagram**

**4. Specific Requirements**

**4.1 External interface Requirements**

### 4.1.1 User Interface Requirements

User interface is used to provide communication between users and system. Our product should have communication between them. Because, Banking Management System is a desktop application and it should get input from users for processing. Firstly the system should ask to its admin/clerkto login and then they can perform various banking tasks. He/ She will select an option for performing the tasks. The system gets these inputs by using user interface. After that, the system will give appropriate output for users via user interface. Finally users will see the optimized path at their screen according to their selected option.The user interface should be easy to learn.

The interface actions and elements should be consistent. When users press any button, required actions should be done by the system. The screen layout and color of the user interface should be appealing. When users look at the screen, it will have a nice vision. Colors will be selected clearly, thus eyes of users won’t be tired.

**4.1.2 Hardware Interface Requirements**

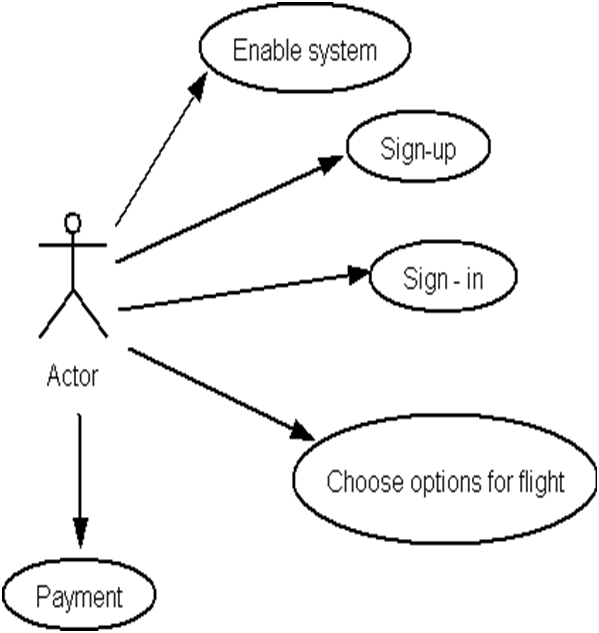
Banking Management System is a desktop application based project. So any personal computer, which has RAM-4GB or more, Graphics Card , is enough to use this system.

**4.1.3 Software Interface Requirements**

In Banking Management System, users will use application program via the user interface program. When database management system access is required, the system establishes a connection to the database management system, once the connection is created; the client program can communicate with the database management system. Most database management system vendors provide ODBC drivers for their systems. A user can actually connect to several database management system and send query and transaction requests using the Open Database Connectivity (ODBC) API, which are then processed at the server site. Any query results are sent back to user, which can process or display the result as needed.

**4.2 Software Product Features**

**4.2.1 Functional Requirements**



**Fig: 1.6 Entity Relationship Diagram**

## 4.2.2 Non-Functional Requirements

## Since the project is a desktop application there is no need of web-based system. Users may or may not use the system via internet. Thus, when they want to do any banking task efficiently, there is no need to have internet.Each user should have a user account. The system should ask the username and password to users. It doesn’t permit to unregistered access.

**4.3 Database Requirements**

In our project, there are three types of databases. One of them keeps customerinformation. Information is customer name, account number, city, gender, mobile number, customer’s father’s name,customer’s mother’s name ,account opening date, e-mail address and address. It is necessary to keeping users information in the database.

Another type keeps transactions occured information. Information is transaction id, transaction source account,transaction destination account,transaction type, transaction amount, transaction time andtransaction date.

Another type keeps the admin’sinformation. Information is admin name, admin id, admin password, security question, security answer, admin email, admin phone number, address of the admin.

Finally, we have to create databases. If they are invalid, system can’t work properly. We can only store information in the databases.

**5. SYSTEM DESIGN**

The systems objectives outlined during the feasibility study serve as the basis from which the work of system design is initiated. Much of the activities involved at this stage is of technical nature requiring a certain degree of experience in designing systems, sound knowledge of computer related technology and thorough understanding of computers available in the market and the various facilities provided by the vendors. Nevertheless, a system cannot be designed in isolation without the active involvement of the user. The user has a vital role to play at this stage too. As we know that data collected during feasibility study will be utilized systematically during the system design. It should, however, be kept in mind that detailed study of the existing system is not necessarily over with the completion of the feasibility study. Depending on the plan of feasibility study, the level of detailed study will vary and the system design stage will also vary in the amount of investigation that still needs to be done. This investigation is generally an urgent activity during the system design as the designer needs to study minute’s details in all aspects of the system.

**System Design Considerations:**

The system design process is not a step-by-step adherence of clear procedures and guidelines. Though, certain clear procedures and guidelines have emerged in recent days, but still much of design work depends on knowledge and experience of the designer.

When designer starts working on system design, he will face different type of problems. Many of these will be due to constraints imposed by the user or limitations of the hardware and software available in the market. However, following considerations should be kept in mind during the system-designing phase:

**The primary objective of the design:** Of course, is to deliver the requirements as specified in the feasibility report. In general, the following design objectives should be kept in mind:

* **Practicality**: The system must be stable and can be operated by people with average
* **Efficiency**: This involves accuracy, timeliness and comprehensiveness of the system.
* **Cost**: it is desirable to aim for a system with a minimum cost subject to the condition that it must satisfy all the requirements.
* **Flexibility**: The system should be modifiable depending on the changing needs of the user. Such modifications should not entail extensive reconstructing or recreation of software. It should also be portable to different computer systems.
* **Security**: This is very important aspect of the design and should cover areas of hardware reliability, fall back procedures, physical security of data and provision for detection of fraud and abuse.

System design involves first logical design and then physical construction of the system. The logical design describes the structure and characteristics of features, like the outputs, inputs, files, databases and procedures. The physical construction, which follows the logical design, produces actual program software, files and a working system.

The designer normally will work under following constraints:

* **Hardware**: The existing hardware will obviously affect the system design.
* **Software**: The available software (operating system, utilities, language etc.) in the market will constrain the design.
* **Budget**: The budget allocated for the project will affect the scope and depth of design.
* **Time-scale**: The new system may be required by a particular time (e.g. the start of a financial year). This may put a constraint on the designer to find the best design.
* **Interface with other systems**: The new system may require some data from another computerized system or may provide data to another system in which case the files must be compatible in format and the system must operate with a certain processing cycle.

**Processing Techniques:**

The processing options available to the designer are:

* Batch processing
* Real-time processing
* On-line processing
* A combination of all the above

You are already aware of these techniques. It is quite interesting to note, however, that a combination of these is often found to be ideal in traditional data processing applications. This increases throughput of the system as also brings down the response time of on-line activities. In most of die business applications, 24-hour data is acceptable enough and hence it is possible to update voluminous data after office-hours in batch mode.

**5.1 DESIGN METHODOLOGIES**

The scope of the systems design is guided by the framework for the new system developed during analysis. More clearly defined logical method for developing system that meets user requirements has led to new techniques and methodologies that fundamentally attempt to do the following:

* Improve productivity of analysts and programmers
* Improve documentation and subsequent maintenance and enhancements.
* Cut down drastically on cost overruns and delays
* Improve communication among the user, analyst, designer, and programmer.
* Standardize the approach to analysis and design
* Simplify design by segmentation.

**5.2 STRUCTURED DESIGN**

Structured design is a data flow based methodology. The approach begins with a system specification that identifies inputs and outputs and describes the functional aspects of the system. The specifications then are used as a basis for the graphic representation. The next step is the definition of the modules and their relationships to one another in a form called a structure chart, using a data dictionary and other structured tools.

Thus, structured design is an attempt to minimize the complexity and make a problem manageable by subdividing it into smaller segments, which is called Modularization or decomposition. In this way, structuring minimizes intuitive reasoning and promotes maintainable provable systems.

**5.3 MODULE COUPLING**

Coupling is the measure of the degree of interdependence between modules. Two modules with high coupling are strongly interconnected and thus, dependent on each other. Two modules with low coupling are not dependent on one another. “Loosely coupled” systems are made up of modules which are relatively independent. “Highly coupled” systems share a great deal of dependence between modules.

**Data Coupling**

The dependency between module A and B is said to be coupled it their dependency is based on the fact they communicate by only passing of data.

**Stamp Coupling**

Stamp coupling occurs between module A and B when complete data structure is passed from one module to another.

**Control Coupling**

Module A and B are said to be control coupled if they communicate by passing of control information.

**External Coupling**

A form of coupling in which has a dependency to other module, external to the software being developed or to a particular type of hardware.

**Common Coupling**

With common coupling, module A and B have shared data. Global data areas are commonly found in programming languages. Making a change to the common data means tracing back to all the modules which access that data to evaluate the effect of change.

**Content Coupling**

Content coupling occurs when module A changes data of module B or when control is passed from one to the middle of another.

**5.4 MODULE COHESION**

Cohesion is a measure of the degree to which the elements of a module are functionally related. A strongly cohesive module implements functionality that is related to one feature of the solution and requires little or no interaction with other modules.

**Functional Cohesion**

X and Y are part of a single functional task. This is very good reason for them to be contained in the same procedure.

**Sequential Cohesion**

X output some data which forms the input to Y. This is the reason for them to be contained in the same procedure.

**Communicational Cohesion**

X and Y both operate on the same input data or contribute towards the same output data. This is okay, but we might consider making them separate procedures.

6. CONCLUSION& FUTURE SCOPE

**Conclusion:-**

We can hereby conclude that:

* The system effectively automated the functions involved in the processes beinghandled manually before.
* The cost & benefit analysis shows that the system was quite successful in saving costsfor the bank & generate equivalently huge benefits.
* The system is secure & scalable. The system design has been done keeping user-friendliness and efficiency in mind.

**Future Scope:-**

Scope of this project is to replace the manual work of banking management system with the new advanced computerized system. User does not need much training to use this software, as this software is very user friendly and easy to use. It replaces all the paper work also. In this software we can store thousands of records. It replaces all the calculation works also as it automatically calculates and store the record in the database.

Some main advantages are:-

* Customer numbers are generated automatically by system itself.
* If any invalid entry done by the users which is nature of human being but checks bythe machine so this software gives the error message to the users to indicate about theinvalid entry.
* While transfer of any customer such as incoming customer, it will update both the fileand incoming customer so that if we want to know number of incoming customer, itwill be very easily for us.
* E-banking
* Maintaining records of all branches
* Issuing and keeping record of credit cards, Atm cards , debit cards
* Locker facilitites can be available
* Record of currency or whole per day transaction including all branches

**REFERENCES**

**BOOK REFERENCES**

1. Doug Hellmann, The Python Standard Library by Example (2 Volume Set), Copyright 1998.
2. Python as a Discrete Event Simulation environmentby F. Oliver Gathmann
3. Python Vuh: Mayan Calendrical Mathematics with Pythonby Ivan Van\Laningham, Callware Technologies, Inc.
4. The ‘C’ Programming language by Kernighan & Ritchie,Prentice Hall Publishing house,Copyright 1988.
5. Teach yourself byYashwantKanetkar, BPb Publishing house edition-6th.
6. Software engineering , K.K. Aggrawal, new age international publisher.

# Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming.

# Python Crash Course, Eric Matthes.

# Think Python: How to Think Like a Computer Scientist, 2nd edition, Allen B. Downey

# [10]Learn Python 3 the Hard Way, Zed A. Shaw (Addison-Wesley, 2016)

**WEB REFERENCES**

**[11**] [www.wikipedia.org](http://www.wikipedia.org)

**[12**] [www.google.com](http://www.google.com)

**[13**] <https://docs.python.org/3.7/library/tkinter.html>

**[14**] www.tutorialspoint.com/python/

**[15**] [www.python.org/doc/tutorial/](http://www.python.org/doc/tutorial/)

**[16**]www.coronadoenterprises.com/tutorials/index.html/

**[17**] https://www.python.org/download/releases/3.0/

**[18**]<https://www.geeksforgeeks.org/python-3-basics/>

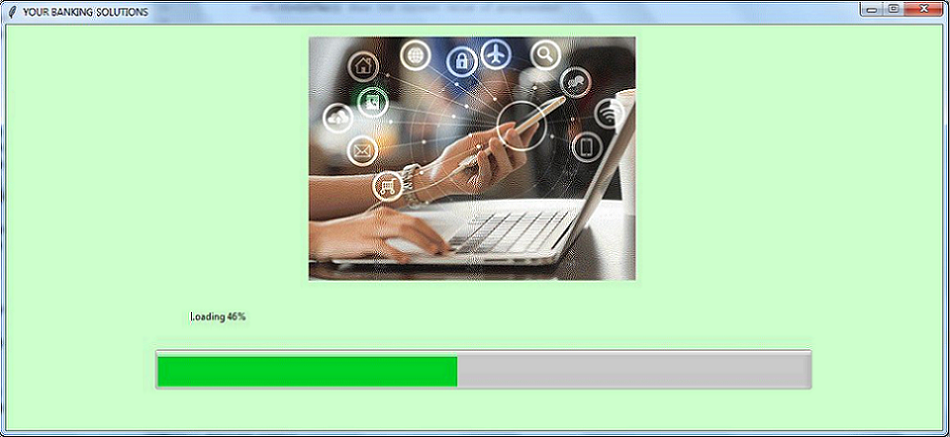
**[19**] <https://pythonprogramming.net/python-3-tkinter-basics-tutorial/>

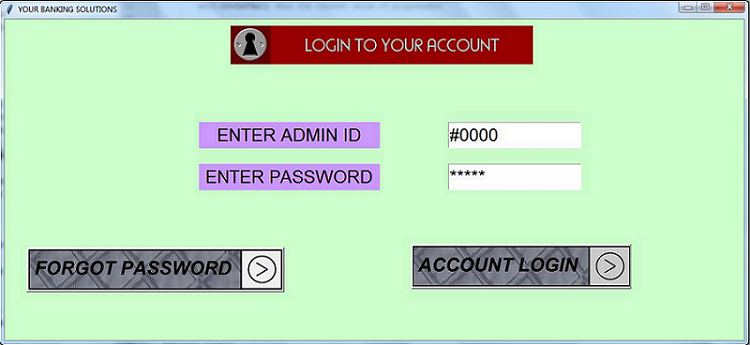
**[20]**<https://www.w3schools.com/python/python_mysql_getstarted.asp>

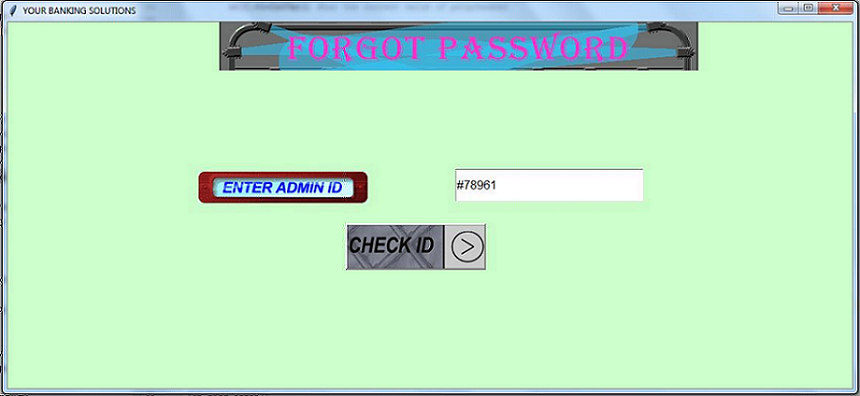
**APPENDIX –A**

**SCREENSHOTS**

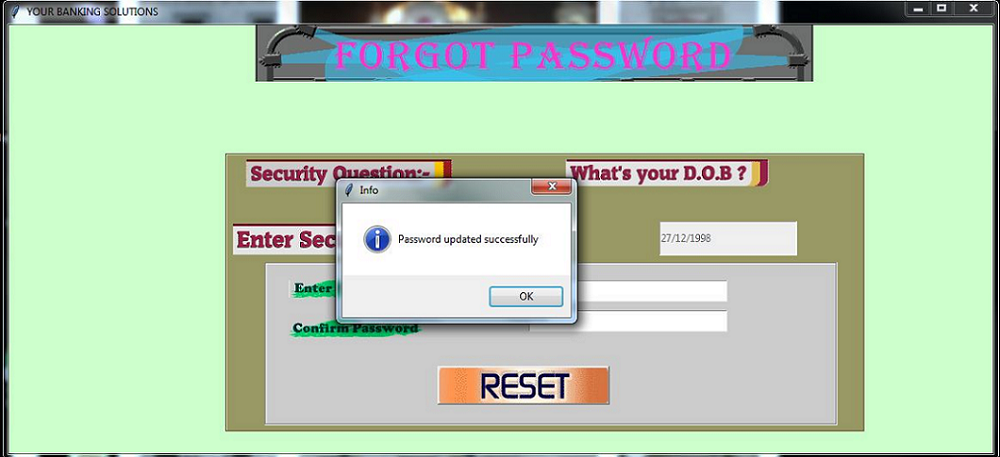
****

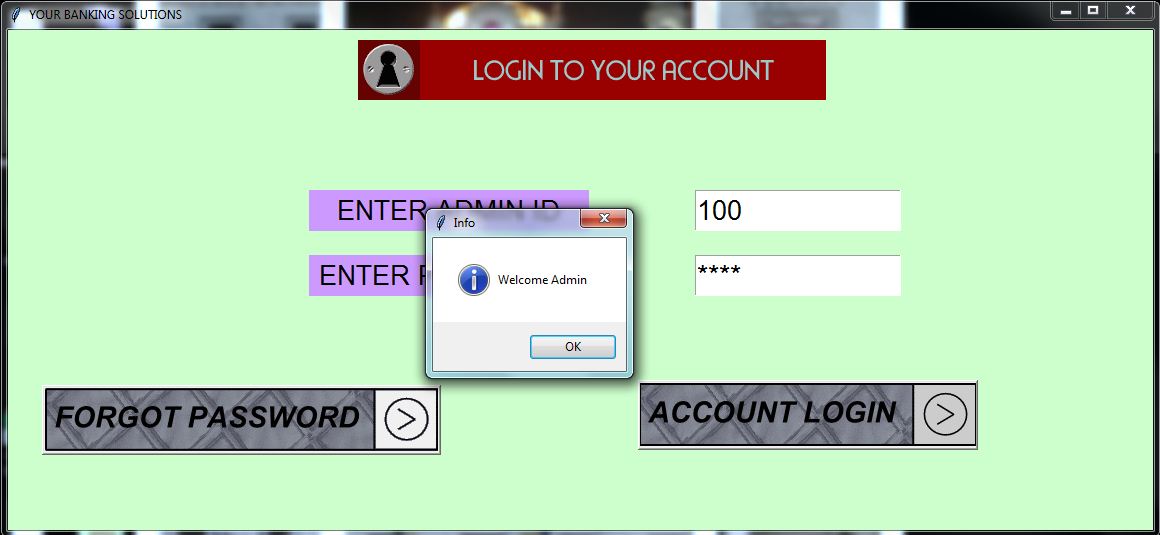
****

****

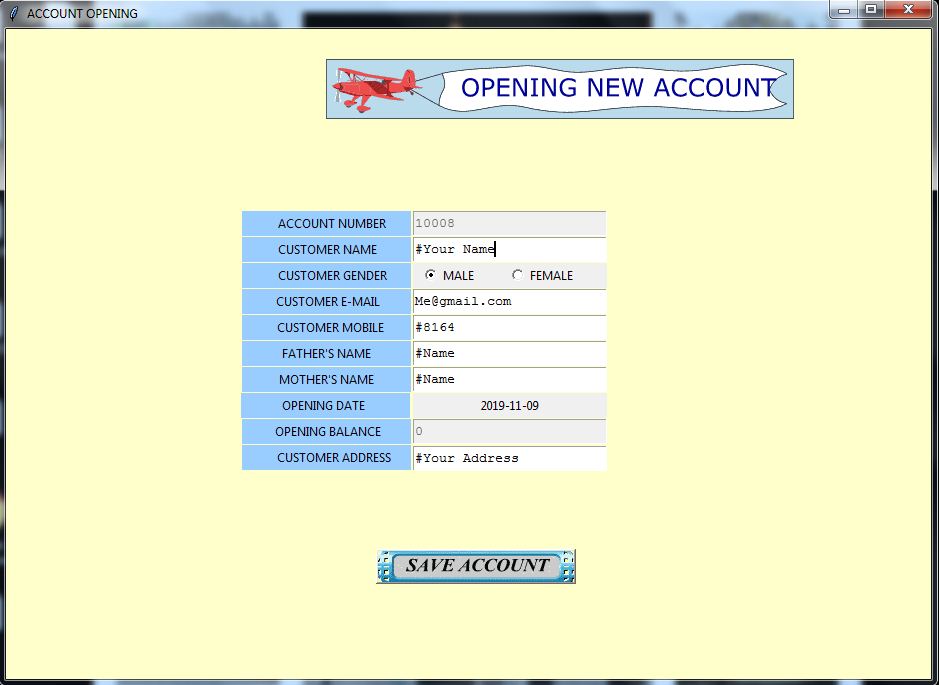
****

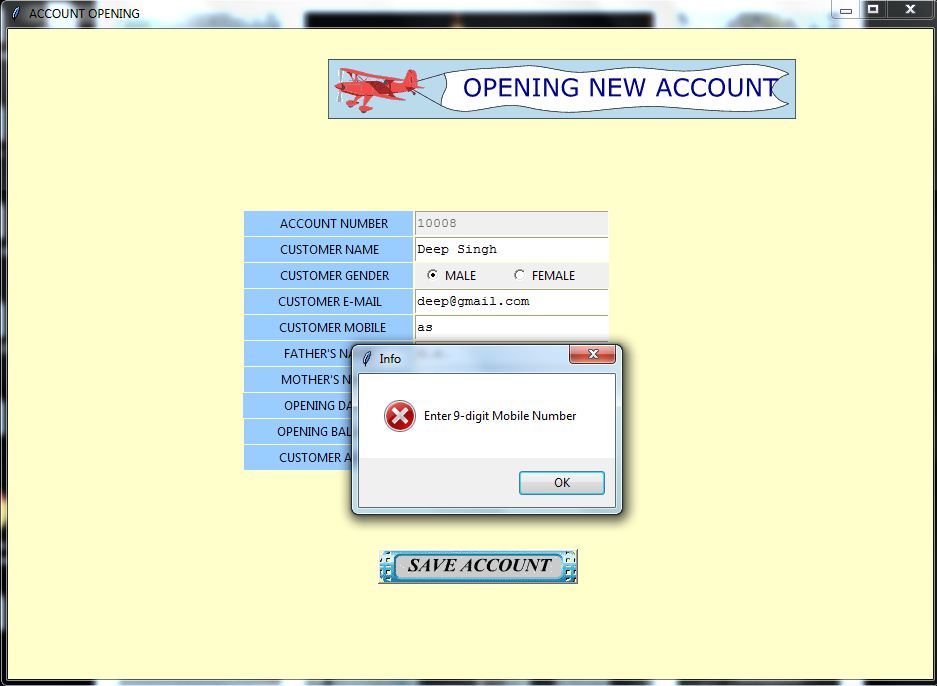
****

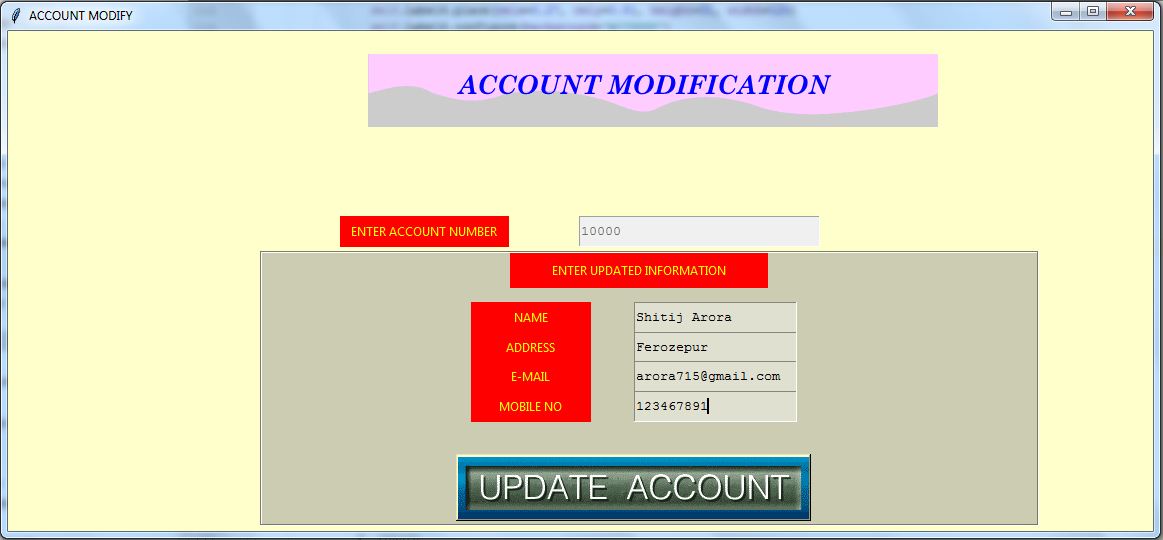
****

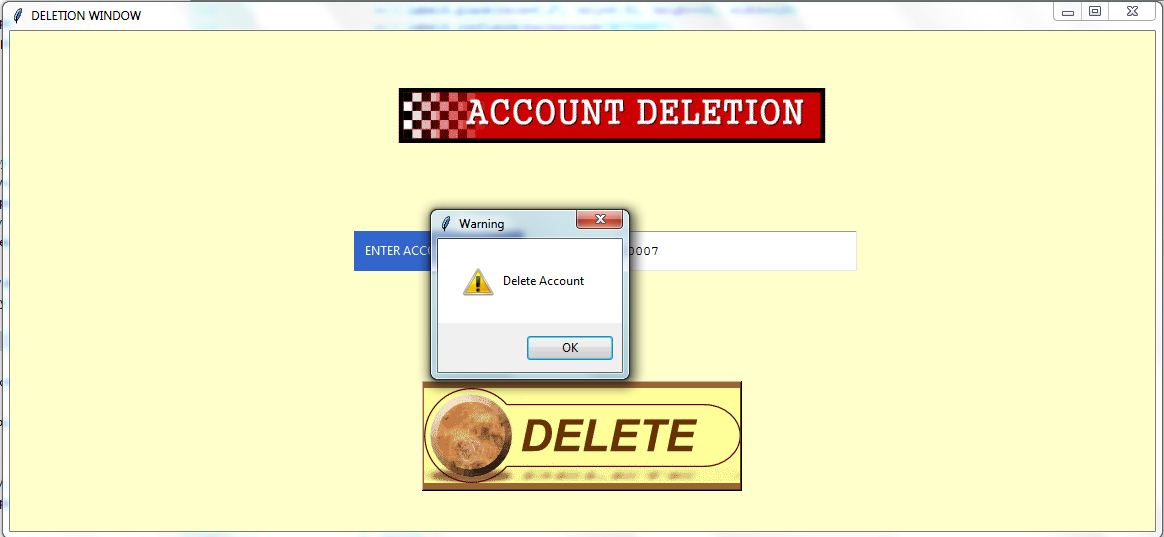
****

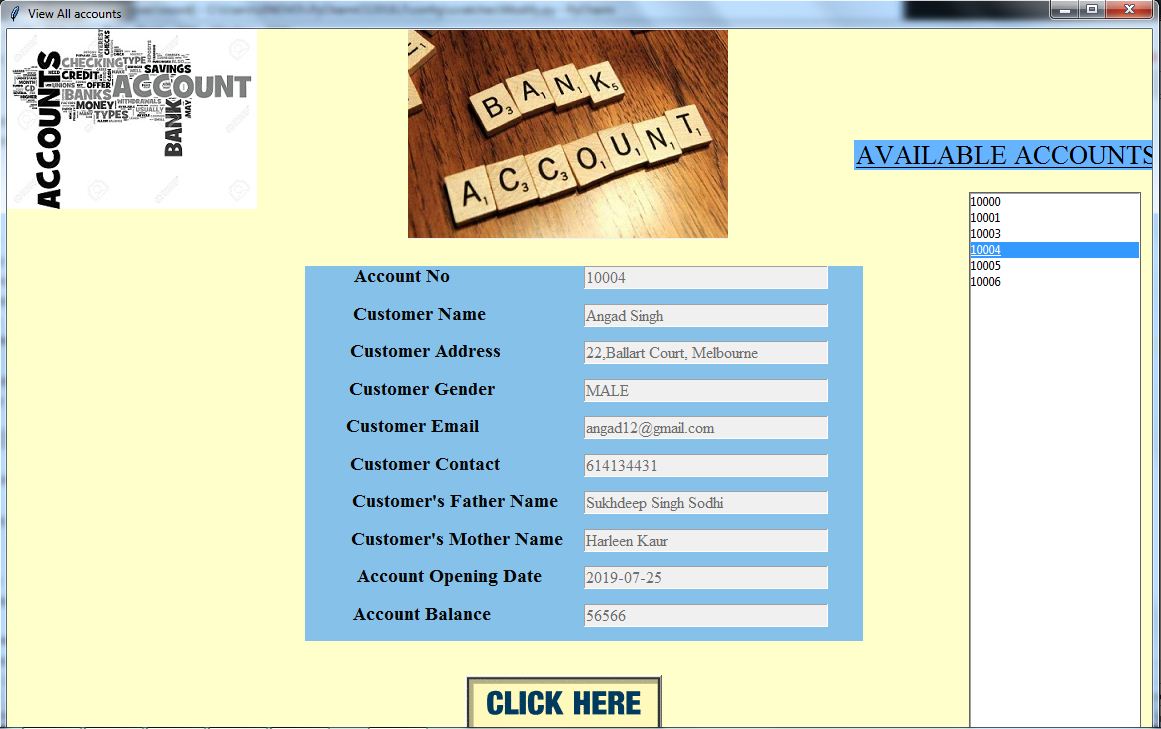
****

****

****

****

****

****

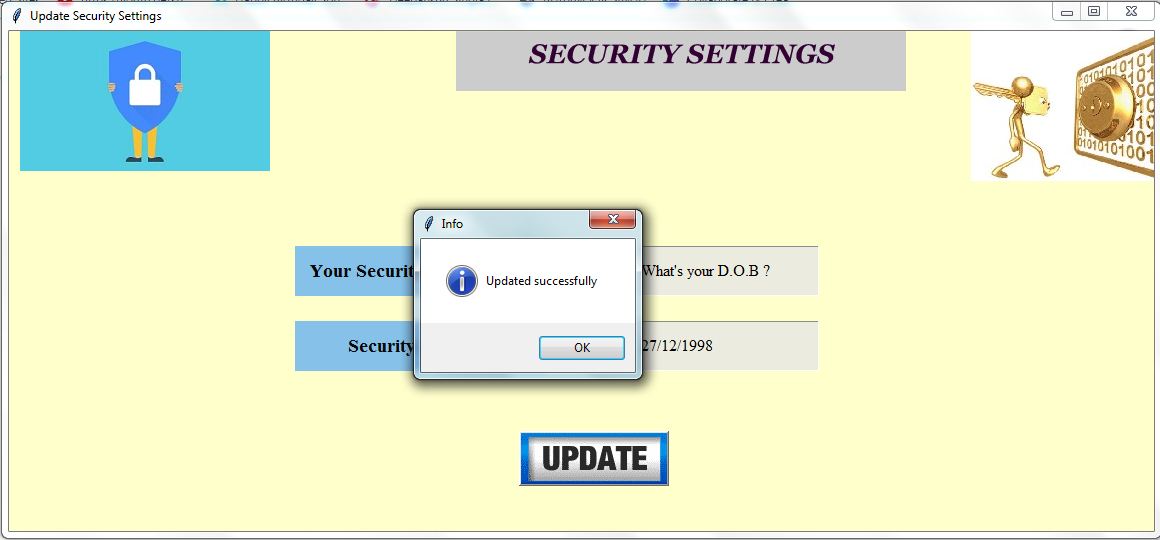
****

****

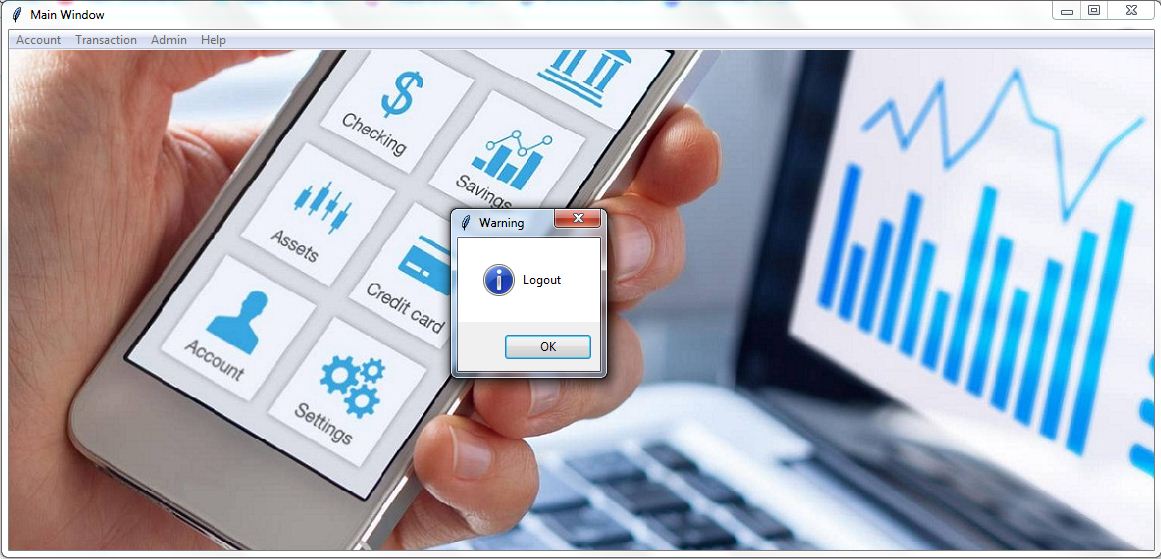
****

****

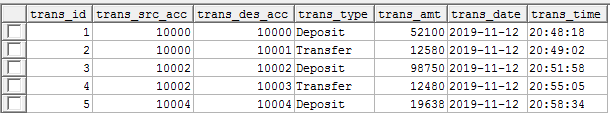
****

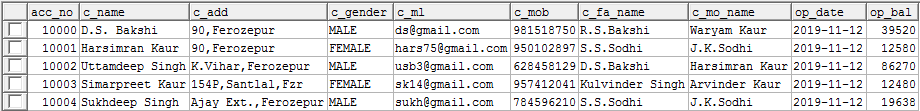
****

****

****

****

****

****

**APPENDIX –B**

**SOURCE CODE**

**Welcome Window:-**

from tkinter import \*

from PIL import Image,ImageTk

import Loading\_new

class welcome:

def \_\_init\_\_(self):

self.rt=Tk()

self.rt.title("YOUR BANKING SOLUTIONS")

self.rt.config(background="#ccffcc")

self.rt.geometry("1145x500+160+100")

self.l1=Label(self.rt)

self.l1.place(relx=0.25, rely=0.0, height=81, width=604)

img1=ImageTk.PhotoImage(Image.open("bank\_images/image1.gif"))#

self.l1.configure(image=img1)

self.l2=Label(self.rt)

limg=ImageTk.PhotoImage(Image.open("bank\_images/Branchless1.jpg"))#"

self.l2.configure(image=limg)

self.but=Button(self.rt,command=self.click)

img2 = ImageTk.PhotoImage(Image.open("bank\_images/cli.gif"))#

self.but.configure(image=img2)

self.rt.mainloop()

def click(self):

self.rt.destroy()

lo=Loading\_new.load\_class()

ob=welcome()

**Loading Window:-**

from tkinter import \*

from PIL import Image,ImageTk, from tkinter.ttk import Progressbar,import threading, Login\_new, time

class load\_class:

def \_\_init\_\_(self):

self.rt=Tk()

self.rt.title("YOUR BANKING SOLUTIONS")

self.rt.config(background="#ccffcc")

self.rt.geometry("1145x500+160+100")

self.l1=Label(self.rt)

self.l1.place(relx=0.28, rely=0.01, height=320, width=500)

self.l1.config(background="#ccffcc")

img1=ImageTk.PhotoImage(Image.open("bank\_images/in.gif"))

self.l1.configure(image=img1)

self.v1=IntVar() #has the current value of progressbar

self.l2=Label(self.rt)

self.l2.place(relx=0.06, rely=0.68, height=40, width=380)

self.l2.configure(background="#ccffcc")

self.l2.configure(width=380)

self.pb=Progressbar(self.rt,orient=HORIZONTAL)

self.pb.configure(mode='determinate',maximum=100)

self.pb.configure(variable=self.v1)

self.pb.place(relx=0.16, rely=0.80, relwidth=0.70, relheight=0.0, height=49)

self.pb.configure(length="650")

tup = (101,)

self.t1=threading.Thread(target=self.move,args=tup,name='first')

self.t1.start()

self.rt.after(500,self.check)

self.rt.mainloop()

def move(self,a):

for i in range(a):

self.v1.set(i)

self.l2.configure(text="Loading "+str(self.v1.get()) +"%")

time.sleep(.01)

def check(self):

if self.v1.get() != 100:

self.rt.after(500,self.check)

else:

self.rt.destroy()

object=Login\_new.Login\_Class()

**Login Window:-**

from tkinter import \*, from tkinter import messagebox

from PIL import Image, ImageTk

import pymysql.cursors, Forgot\_new, After\_new

class Login\_Class:

def \_\_init\_\_(self):

self.rt = Tk()

self.rt.geometry("1145x500+160+100")

self.rt.title("YOUR BANKING SOLUTIONS")

self.rt.configure(background="#ccffcc")

self.Label1 = Label(self.rt)

self.Label1.place(relx=0.27, rely=0.0, height=80, width=550)

self.Label1.configure(background="#ccffcc")

img1 = ImageTk.PhotoImage(Image.open("bank\_images/login.gif"))

self.Label1.configure(image=img1)

self.Label2 = Label(self.rt)

self.Label2.place(relx=0.263, rely=0.32, height=41, width=280)

self.Label2.configure(background="#cc99ff")

self.Label2.configure(foreground="#000000")

self.Label2.configure(text='''ENTER ADMIN ID''')

self.Label3.configure(text='''ENTER PASSWORD''')

self.Label3.configure(width=164, font=('Helvetica', 20))

self.Entry1 = Entry(self.rt)

self.Entry2.insert(0, "\*\*\*\*\*")

self.Entry2.configure(state="readonly")

self.Entry2.configure(state="normal", font=('Helvetica', 20))

self.Entry2.configure(show="\*")

self.Button1 = Button(self.rt, command=self.login)

self.Button1.place(relx=0.55, rely=0.7, height=70, width=340)

imga = ImageTk.PhotoImage(Image.open("bank\_images/LOG.gif"))

self.Button1.configure(image=imga)#,background="#ccffcc"

self.Button2 = Button(self.rt, command=self.forgot)

self.Button2.place(relx=0.03, rely=0.71, height=70, width=399)

imge = ImageTk.PhotoImage(Image.open("bank\_images/FOR.gif"))

self.Button2.configure(image=imge)#,background="#ccffcc"

self.rt.mainloop()

def login(self):

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

un = self.Entry1.get()

pwd = self.Entry2.get()

cursor.execute("select \* from tbadmin where admin\_id=%s and admin\_pass=%s", (un, pwd))

con.commit()

rows = cursor.rowcount

if rows > 0:

messagebox.showinfo('Info', 'Welcome Admin')

self.Entry1.delete(0, END)

self.Entry2.delete(0, END)

self.rt.destroy()

object=After\_new.Menu\_Class(un)

else:

messagebox.showinfo('Info', 'Invalid Name or Password')

self.Entry1.delete(0, END)

self.Entry2.delete(0, END)

return

def forgot(self):

self.rt.destroy()

f = Forgot\_new.Forgot\_Class()

**Forgot Window:-**

from tkinter import \*, from PIL import Image, ImageTk

import pymysql.cursors, Login\_new, from tkinter import messagebox

class Forgot\_Class:

def \_\_init\_\_(self):

self.top = Tk()

self.top.geometry("1145x500+160+100")

self.top.title("YOUR BANKING SOLUTIONS")

self.top.configure(background="#ccffcc")

self.Label1 = Label(self.top)

self.Label1.place(relx=0.25, rely=0.0, height=66, width=650)

self.Label1.configure(background="#ccffcc")

img1 = ImageTk.PhotoImage(Image.open("bank\_images/Forgot.gif"))

self.Label1.configure(image=img1)

self.Label2 = Label(self.top)

self.Button3 = Button(self.Frame2, command=self.reset)

self.Button3.place(relx=0.3, rely=0.64, height=45, width=200)

imlng2 = ImageTk.PhotoImage(Image.open("bank\_images/re.gif"))

self.Button3.configure(image=imlng2)#,background="#cccccc"

self.Button2 = Button(self.Frame1, command=self.verify)

self.Button2.place(relx=0.39, rely=0.65, height=55.35, width=157.2)

ime = ImageTk.PhotoImage(Image.open("bank\_images/ver1.gif"))

self.Button2.configure(image=ime)

self.top.mainloop()

def verify(self):

usrans = self.Entry2.get()

if (usrans == self.secans):

self.Frame2.place(relx=0.06, rely=0.39, relheight=0.59, relwidth=0.9)

self.Entry2.config(state='disabled')

self.Button2.destroy()

else:

messagebox.showinfo('Info', 'Invalid Security Answer')

def reset(self):

pwd = self.Entry3.get()

cpwd = self.Entry4.get()

if (pwd == cpwd):

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

aid = self.Entry1.get()

cursor.execute("update tbadmin set admin\_pass=%s where admin\_id=%s", (pwd, aid))

con.commit()

messagebox.showinfo('Info', 'Password updated successfully')

self.top.destroy()

obj1 = Login\_new.Login\_Class()

else:

messagebox.showinfo('Info', 'Password and Confirm Password should match')

self.Entry3.delete(0, END)

self.Entry4.delete(0, END)

def check(self):

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

aid = self.Entry1.get()

cursor.execute("select adsecques, adsecans from tbadmin where admin\_id=%s", (aid))

con.commit()

rows = cursor.rowcount

if rows > 0:

row = cursor.fetchone()

self.Label4.configure(text=row[0])

self.secans = row[1]

self.Entry1.config(state='disabled')

self.Frame1.place(relx=0.22, rely=0.3, relheight=0.65, relwidth=0.65)

else:

messagebox.showinfo('Info', 'Invalid ID')

self.Entry1.delete(0, END)

**Main Window:-**

from tkinter import \*

from PIL import Image,ImageTk, from tkinter import messagebox

import Open, View\_new, Search, Delete, Login\_new, Modify, Withdraw, Deposit, Transfer, Mini, About\_us, Edit\_prof, Password\_ch, Ad\_sec

class Menu\_Class:

def \_\_init\_\_(self,id):

self.rt=Tk()

self.rt.title("Main Window")

self.admid=id

self.mb=Menu(self.rt)

self.rt.configure(menu=self.mb)

self.rt.option\_add("\*tearOff",False)

self.rt.geometry("1145x500+160+100")

self.lab = Label(self.rt)

self.lab.place(relx=0.0, rely=0.0, relheight=1.0, relwidth=1.0)# )

img12 = ImageTk.PhotoImage(Image.open("bank\_images/main2.jpg"))

self.lab.configure(image=img12)

self.m1=Menu(self.mb)

self.m2=Menu(self.mb)

self.m3=Menu(self.mb)

self.m4=Menu(self.mb)

self.mb.add\_cascade(menu=self.m1,label="Account")

self.mb.add\_cascade(menu=self.m2, label="Transaction")

self.mb.add\_cascade(menu=self.m3, label="Admin")

self.mb.add\_cascade(menu=self.m4, label="Help")

self.m2.add\_command(label="Withdraw",command=self.withdraw)#

self.m2.add\_command(label="Deposit",command=self.deposit)#

self.m2.add\_command(label="Transfer",command=self.transfer)#

self.m2.add\_command(label="Mini Statement",command=self.mini)#

self.m3.add\_command(label="Edit Profile",command=self.prof)

self.m3.add\_command(label="Edit Password",command=self.passd)

self.m3.add\_command(label="Edit Security Settings",command=self.sec)

self.m3.add\_separator()

self.m3.add\_command(label="Logout",command=self.log\_out)

self.m4.add\_command(label="About Us",command=self.aboutus)

self.rt.mainloop()

def open(self):

self.rt.destroy()

o1=Open.openi\_acc(self.admid)

def modify(self):

self.rt.destroy()

o2 = Modify.mod\_acc(self.admid)

def delete(self):

self.rt.destroy()

o3=Delete.del\_acc(self.admid)

def view(self):

self.rt.destroy()

o4=View\_new.ViewAllAccount(self.admid)

def search(self):

self.rt.destroy()

o5=Search.search\_acc(self.admid)

def withdraw(self):

self.rt.destroy()

o6=Withdraw.with\_acc(self.admid)

def deposit(self):

self.rt.destroy()

o7=Deposit.depo\_acc(self.admid)

def transfer(self):

self.rt.destroy()

o8=Transfer.trans\_acc(self.admid)

def mini(self):

self.rt.destroy()

o9=Mini.mini\_stat(self.admid)

def log\_out(self):

messagebox.showinfo("Warning","Logout")

self.rt.destroy()

o10 = Login\_new.Login\_Class()

def aboutus(self):

self.rt.destroy()

o11=About\_us.about\_us(self.admid)

def prof(self):

self.rt.destroy()

o12=Edit\_prof.EditPrf(self.admid)

def passd(self):

self.rt.destroy()

o13=Password\_ch.Password(self.admid)

def sec(self):

self.rt.destroy()

o14=Ad\_sec.Security(self.admid)

**Open Account Window:-**

from tkinter import \*, import pymysql.cursors, datetime, After\_new

from tkinter import messagebox, from PIL import Image, ImageTk

class openi\_acc:

def \_\_init\_\_(self,uname):

self.rt=Tk()

self.rt.title("ACCOUNT OPENING")

self.rt.configure(background="#ffffcc")

self.rt.geometry("925x650+200+10")

self.l1=Label(self.rt)

self.l1.place(relx=0.22, rely=0.0, height=120, width=700)

img1 = ImageTk.PhotoImage(Image.open("bank\_images/open2.gif"))

self.l1.configure(image=img1,background="#ffffcc")

self.id = uname

def save(self):

if(self.Entry2.get()=="" or self.Entry3.get()=="" or self.Entry5.get()=="" or self.Entry7.get()=="" or self.Entry8.get()=="" ):

messagebox.showerror("Info","Fields Cannot be Empty")

elif(len(self.Entry6.get()) < 9 or len(self.Entry6.get()) > 9 ):

messagebox.showerror("Info","Enter 9-digit Mobile Number")

else:

try:

accno = int(self.Entry1.get())

cmob = int(self.Entry6.get())

copb = int(self.Entry10.get())

except ValueError as e:

messagebox.showerror("Info","Account No, Mobile No. and Opening Balance can be Integers Only")

return

cname = self.Entry2.get()

cname1=cname.title()

cadd = self.Entry3.get()

cadd1=cadd.title()

cgen = self.s1.get()

cmail = self.Entry5.get()

f\_name = self.Entry7.get()

f\_name1=f\_name.title()

m\_name = self.Entry8.get()

m\_name1=m\_name.title()

copd = self.a1

try:

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

cursor.execute("insert into tbaccount values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",(accno, cname1, cadd1, cgen, cmail, cmob, f\_name1, m\_name1, copd, copb))

con.commit()

rows = cursor.rowcount

def back(self):

self.rt.destroy()

f = After\_new.Menu\_Class(self.id)

**Deletion Window:-**

from tkinter import \*, from PIL import Image, ImageTk

import pymysql.cursors, After\_new, from tkinter import messagebox

class del\_acc:

def \_\_init\_\_(self,uname):

self.rt=Tk()

self.rt.title("DELETION WINDOW")

self.rt.configure(background="#ffffcc")

self.rt.geometry("1145x500+160+100")

self.l1 = Label(self.rt)

self.l1.place(relx=0.22, rely=0.0, height=170, width=700)

img1 = ImageTk.PhotoImage(Image.open("bank\_images/del1.gif"))

self.l1.configure(image=img1,background="#ffffcc")

self.id = uname

self.Label2 = Label(self.rt)

self.Label2.place(relx=0.3, rely=0.40, height=40, width=169)

self.Label2.configure(background="#3366cc")

self.Label2.configure(foreground="#ffffff")

self.Label2.configure(text='''ENTER ACCOUNT NUMBER''')

self.Entry1 = Entry(self.rt)

self.Entry1.place(relx=0.53, rely=0.40, height=40, relwidth=0.21)

self.Button1 = Button(self.rt, command=self.view)

self.Button1.place(relx=0.36, rely=0.70, height=110, width=320)

img3 = ImageTk.PhotoImage(Image.open("bank\_images/deln.gif"))

self.Button1.configure(background="#996633",image=img3)

self.rt.mainloop()

def view(self):#correct this ftn

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

aid = int(self.Entry1.get())

cursor.execute("select \* from tbaccount where acc\_no=%s", (aid))

con.commit()

rows = cursor.rowcount

if rows > 0:

messagebox.showwarning("Warning", "Delete Account")

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

aid = int(self.Entry1.get())

cursor.execute("delete from tbaccount where acc\_no=%s", (aid))

con.commit()

messagebox.showinfo('Info', 'Account Deleted')

self.rt.destroy()

obs = After\_new.Menu\_Class(self.id)

else:

messagebox.showinfo('Info', 'Invalid Account No.')

self.Entry1.delete(0, END)

**View Accounts Window:-**

from tkinter import \*, from tkinter import messagebox

from PIL import Image, ImageTk, import pymysql.cursors, After\_new

class ViewAllAccount:

def \_\_init\_\_(self,uname):

self.root = Tk()

self.root.title("View All accounts")

self.root.geometry("1145x800+160+0")

self.root.config(background='#ffffcc')

self.conn = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

self.cursor = self.conn.cursor()

i1 = ImageTk.PhotoImage(Image.open("bank\_images/va.jpg"))

self.Label1 = Label(self.root, image=i1)

self.Label1.place(relx=0.0, rely=0.0, width=250, height=180)

self.id = uname

i2=ImageTk.PhotoImage(Image.open("bank\_images/ba.jpg"))

self.Label2 = Label(self.root, image=i2)

self.Label2.place(relx=0.35, rely=0.0, width=320, height=210)

self.l=Label(self.root)

self.l.place(relx=0.74, rely=0.15, width=320, height=30)

self.t4.configure(state='normal')

self.t5.configure(state='normal')

self.t6.configure(state='normal')

self.t7.configure(state='normal')

self.t8.configure(state='normal')

self.t9.configure(state='normal')

self.t10.configure(state='normal')

def other(self):

self.root.destroy()

obja=After\_new.Menu\_Class(self.id)

**Withdrawal Window:-**

from tkinter import \*, import time, pymysql.cursors, from tkinter import messagebox

from PIL import Image, ImageTk, import datetime, import After\_new

class with\_acc:

def \_\_init\_\_(self,uname):

self.rt=Tk()

self.rt.title("WITHDRAWAL WINDOW")

self.rt.configure(background="#ffffcc")

self.rt.geometry("1145x500+160+100")

self.rt.mainloop()

def check(self):

aid=self.Entry1.get()

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

cursor.execute("select op\_bal from tbaccount where acc\_no=%s", (aid))

con.commit()

rows = cursor.rowcount

def other(self):

aid = self.Entry1.get()

if (self.Entry3.get() == ""):

messagebox.showerror("Info", "Can't be Empty")

else:

try:

abc = int(self.Entry3.get())

except ValueError as e:

messagebox.showerror("Info","Can be Integers Only")

self.Entry3.delete(0, END)

return

self.Entry3.configure(state="disabled")

abc1=int(abc)

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

cursor.execute("select op\_bal from tbaccount where acc\_no=%s", (aid))

con.commit()

rows = cursor.rowcount

if rows > 0:

row = cursor.fetchone()

row1=int(row[0])

if(abc1 > row1):

messagebox.showerror("Info","Insufficient Balance")

self.Entry3.configure(state="normal")

elif(abc1 <=0):

messagebox.showerror("Info","Enter Amount Greater than Zero")

self.Entry3.configure(state="normal")

else:

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

aid = self.Entry1.get()

avalue=self.Entry3.get()

abvalue=self.Entry2.get()

sub=int(abvalue) - int(avalue)

cursor.execute("update tbaccount set op\_bal=%s where acc\_no=%s", (sub,aid))

con.commit()

rows = cursor.rowcount

if rows > 0:

row = cursor.fetchone()

messagebox.showinfo("Info","Successful")

a1 = datetime.date.today()

a2 = time.strftime("%H:%M:%S")

a3 = int(self.Entry3.get())

aid = int(self.Entry1.get())

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

cursor.execute("select ifnull(max(trans\_id),0) from tbtransacc")

con.commit()

row = cursor.fetchone()

ano = row[0]

ano = ano + 1

tp = "Withdrawal"

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

cursor.execute("insert into tbtransacc values(%s,%s,%s,%s,%s,%s,%s)",

(ano, aid, aid, tp, a3, a1, a2))

con.commit()

else:

messagebox.showerror("Info","Transaction Failed")

self.rt.destroy()

of=After\_new.Menu\_Class(self.id)

**Deposit Window:-**

from tkinter import \*, import pymysql.cursors, import After\_new

from PIL import Image, ImageTk, import datetime, import time, from tkinter import messagebox

class depo\_acc:

def \_\_init\_\_(self,uname):

self.rt=Tk()

self.rt.title("DEPOSIT WINDOW")

self.rt.configure(background="#ffffcc")

self.rt.geometry("1145x500+160+100")

self.id = uname

self.l1 = Label(self.rt)

self.l1.place(relx=0.31, rely=0.0, height=66, width=480)

img1 = ImageTk.PhotoImage(Image.open("bank\_images/depos.gif"))

self.l1.configure(image=img1)

self.Label3 = Label(self.Frame1)

self.Label3.place(relx=0.01, rely=0.03, height=31, width=178)

self.Label3.configure(background="#ff66ff")

self.Label3.configure(foreground="#000000")

self.Label3.configure(text='''CURRENT BALANCE''')

self.Entry2 = Entry(self.Frame1)

self.Entry2.place(relx=0.5, rely=0.03, height=31, width=178)

self.Entry2.configure(font="TkFixedFont")

self.Label4 = Label(self.Frame1)

self.Label4.place(relx=0.01, rely=0.28, height=31, width=178)

self.Label4.configure(background="#ff66ff")

self.Label4.configure(foreground="#000000")

self.Label4.configure(text='''ENTER DEPOSIT AMOUNT''')

self.rt.mainloop()

def check(self):

aid=self.Entry1.get()

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

cursor.execute("select op\_bal from tbaccount where acc\_no=%s", (aid))

con.commit()

rows = cursor.rowcount

if rows > 0:

row = cursor.fetchone()

print(row)

self.Entry1.configure(state="disabled")

self.Frame1.place(relx=0.26, rely=0.3, relheight=0.45, relwidth=0.43)

self.Entry2.insert(0, row)

self.Entry2.configure(state="disabled")

else:

messagebox.showerror("Info","Enter Correct Account Number")

def other(self):

aid = self.Entry1.get()

if (self.Entry3.get() == ""):

messagebox.showerror("Info", "Can't be Empty")

else:

try:

abc = int(self.Entry3.get())

except ValueError as e:

messagebox.showerror("Info","Can be Integers Only")

self.Entry3.delete(0, END)

return

self.Entry3.configure(state="disabled")

abc1=int(abc)

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

cursor.execute("select op\_bal from tbaccount where acc\_no=%s", (aid))

con.commit()

rows = cursor.rowcount

self.rt.destroy()

of = After\_new.Menu\_Class(self.id)

else:

messagebox.showerror("Info","Transaction Failed")

self.Entry3.configure(state="normal")

o1=After\_new.Menu\_Class(self.id)

**Transfer Window:-**

from tkinter import \*, from tkinter import messagebox

from PIL import Image, ImageTk, import pymysql.cursors, datetime, After\_new, time

class trans\_acc:

def \_\_init\_\_(self,uname):

self.rt = Tk()

self.rt.title("TRANSFER WINDOW")

self.rt.geometry("1145x500+160+100")

self.rt.configure(background="#ffffcc")

self.l1 = Label(self.rt)

self.l1.place(relx=0.01, rely=0.0, height=180, width=350)

img1 = ImageTk.PhotoImage(Image.open("bank\_images/transfer.png"))

self.l1.configure(image=img1)

self.id = uname

self.l23 = Label(self.rt)

self.l23.place(relx=0.4, rely=0.0, height=60, width=350)

img0 = ImageTk.PhotoImage(Image.open("bank\_images/t.gif"))

self.l23.configure(image=img0)

self.l24 = Label(self.rt)

self.l24.place(relx=0.8, rely=0.0, height=200, width=250)

img20 = ImageTk.PhotoImage(Image.open("bank\_images/trans1.png"))

self.l24.configure(image=img20)

self.al=Label(self.rt)

self.sutto = Button(self.Frame1, command=self.other) #

self.sutto.place(relx=0.22, rely=0.6, height=58, width=270)

img89 = ImageTk.PhotoImage(Image.open("bank\_images/trans2.gif"))

self.sutto.configure(image=img89)

self.Frame2 = Frame(self.Frame1)

self.Frame2.place(relx=0.03, rely=0.3, relheight=0.55, relwidth=0.68)

self.Frame2.configure(relief=FLAT)

self.Frame2.configure(borderwidth="2")

self.Frame2.configure(relief=GROOVE)

self.Frame2.configure(background="#ebebe0")

self.Frame2.configure(width=530)

self.Frame2.place\_forget()

self.rt.mainloop()

con.commit()

rows=cursor.rowcount

if rows > 0:

messagebox.showinfo("Info","Successful")

a1 = datetime.date.today()

a2 = time.strftime("%H:%M:%S")

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

cursor.execute("select ifnull(max(trans\_id),0) from tbtransacc")

con.commit()

row = cursor.fetchone()

ano = row[0]

ano = ano + 1

tp = "Transfer"

a3=int(self.Entry4.get())

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

cursor.execute("insert into tbtransacc values(%s,%s,%s,%s,%s,%s,%s)",

(ano, aid1, aid2, tp, a3, a1, a2))

con.commit()

self.rt.destroy()

o6=After\_new.Menu\_Class(self.id)

else:

messagebox.showerror("Info","Failure")

self.rt.destroy()

o11 = After\_new.Menu\_Class(self.id)

else:

messagebox.showerror("Info","Transaction Failed")

self.rt.destroy()

o10=After\_new.Menu\_Class(self.id)

**Mini Statement Window:-**

from tkinter import \*, from tkinter.ttk import Treeview, from tkinter import messagebox, from PIL import Image, ImageTk, import pymysql.cursors, After\_new

class mini\_stat:

def \_\_init\_\_(self,uname):

self.rt = Tk()

self.rt.title("STATEMENT WINDOW")

self.rt.geometry("1445x400+05+30")

self.rt.configure(background="#ffffcc")

self.l1 = Label(self.rt)

self.l1.place(relx=0.01, rely=0.0, height=220, width=220)

img1 = ImageTk.PhotoImage(Image.open("bank\_images/stat.jpg"))

self.l1.configure(image=img1)

self.id = uname

self.l12 = Label(self.rt)

self.l12.place(relx=0.35, rely=0.0, height=57, width=479)

img12 = ImageTk.PhotoImage(Image.open("bank\_images/acst.gif"))

self.l12.configure(image=img12)

def check(self):

con = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

cursor = con.cursor()

aid = self.Entry1.get()

cursor.execute("select \* from tbtransacc where trans\_src\_acc=%s", (aid))

con.commit()

rows = cursor.rowcount

if rows > 0:

self.Entry1.config(state='disabled')

self.fr.place(relx=0.22, rely=0.38, relheight=0.6, relwidth=0.65)

msg = ""

for i in cursor:

self.ta.insert("1.0", "\n")

msg = msg +" "+str(i[0]) + '\t' +" "+ str(i[1]) +" " + '\t' + " " + str(i[2])+""+ '\t'+" " + str(i[3])+"" +'\t'+" "+ str(i[4])+" "+ '\t'+" "+str(i[5])+" "+ '\t'+" "+str(i[6]) + '\n'+'\n'

self.ta.insert("3.0", msg)

self.ta.configure(state="disabled")

else:

messagebox.showinfo('Info', 'Invalid Account Number')

self.Entry1.delete(0, END)

def close(self):

self.rt.destroy()

o1=After\_new.Menu\_Class(self.id)

**Edit Profile Window:-**

from tkinter import \*, from tkinter import messagebox

from PIL import Image, ImageTk, import pymysql.cursors, After\_new

class EditPrf:

def \_\_init\_\_(self, uname):

self.root = Tk()

self.root.geometry("1145x500+160+100")

self.root.title("Edit Admin's Profile")

self.root.configure(background="#ffffcc")

self.id = uname

self.conn = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

self.cursor = self.conn.cursor()

i1 = ImageTk.PhotoImage(Image.open("bank\_images/admod1.png"))

self.Label1 = Label(self.root,image = i1)

self.Label5.place(relx=0.26, rely=0.55, width=180, height=40)

self.Entry2 = Entry(self.root, font=('Times New Roman', 12))

self.Entry2.place(relx=0.5, rely=0.55, width=180, height=40)

self.Entry2.configure(background="#ebebe0")

self.Label6 = Label(self.root, text="Email ID: ", font=("Times New Roman", '14', 'bold'))

self.Label6.config(foreground='#070707', background='#85C1E9')

self.Label6.place(relx=0.26, rely=0.65, width=180, height=40)

self.Entry3 = Entry(self.root, font=('Times New Roman', 12))

self.Entry3.place(relx=0.5, rely=0.65, width=180, height=40)

self.Entry3.configure (background="#ebebe0")

self.Button2 = Button(self.root)

i5 = ImageTk.PhotoImage(Image.open("bank\_images/admod.gif"))

self.Button2.config(image=i5, command=self.modify)

self.Button2.place(relx=0.39, rely=0.83, width=150, height=60)

self.cursor.execute("select admin\_name,admin\_add,admin\_el,admin\_ph from tbadmin where admin\_id = %s", (self.id))

row = self.cursor.fetchone()

self.Entry1.insert(0, row[0])

self.Text1.insert('1.0', row[1])

self.Entry2.insert(0, row[3])

self.Entry3.insert(0, row[2])

self.conn.commit()

self.root.mainloop()

def modify(self):

nm = self.Entry1.get()

add = self.Text1.get('1.0', END)

mob = self.Entry2.get()

email = self.Entry3.get()

if nm == "" or len(add) == 0 or email == "" or mob == "":

messagebox.showerror('Info', "Can't be empty")

else:

try:

mob1 = int(mob)

except ValueError as e:

messagebox.showerror('Info', 'Contact no. can only be an integer value!!')

return

if len(mob) != 9:

messagebox.showerror('Info', 'Contact no. can only be of 9-digits!!')

return

elif not ('@gmail.com' in email):

messagebox.showerror('Info', 'Invalid Email ID!!')

return

else:

self.cursor.execute('update tbadmin set admin\_name = %s,admin\_add = %s,admin\_el = %s,admin\_ph = %s where admin\_id = %s',(nm, add, email, mob, self.id))

self.conn.commit()

messagebox.showinfo('Info', 'Updated successfully!!')

self.root.destroy()

obj2 = After\_new.Menu\_Class(self.id)

**Update Password Window:-**

from tkinter import \*, from tkinter import messagebox

from PIL import Image, ImageTk, import pymysql.cursors, After\_new

class Password:

def \_\_init\_\_(self, uname):

self.root = Tk()

self.root.geometry("1145x500+160+100")

self.root.title('Update Password')

self.root.configure(background="#ffffcc")

self.id = uname

self.conn = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

self.cursor = self.conn.cursor()

i2 = ImageTk.PhotoImage(Image.open("bank\_images/PU.gif"))

self.Label2 = Label(self.root,image = i2)

self.Label2.place(relx = 0.37,rely = 0.027, width = 380,height = 60)

i21 = ImageTk.PhotoImage(Image.open("bank\_images/pup.jpg"))

self.Label56 = Label(self.root, image=i21)

self.Label56.place(relx=0.01, rely=0.0, width=200, height=180)

i22 = ImageTk.PhotoImage(Image.open("bank\_images/pup2.jpg"))

self.Label200 = Label(self.root, image=i22)

self.Label200.place(relx=0.81, rely=0.0, width=250, height=160)

self.Label3 = Label(self.root, text="Enter New Password: ", font=('Times New Roman', '14', 'bold'))

self.root.mainloop()

def update(self):

pwd = self.Entry1.get()

cpwd = self.Entry2.get()

if pwd == "" or cpwd == "":

messagebox.showerror('Info', 'Cannot be empty')

else:

if pwd == cpwd:

self.cursor.execute('update tbadmin set admin\_pass = %s where admin\_id = %s', (pwd, self.id))

self.conn.commit()

messagebox.showinfo('Info', 'Updated successfully')

self.root.destroy()

aj=After\_new.Menu\_Class(self.id)

else:

messagebox.showerror('Info', 'Password and Confirm Password should match!!')

self.Entry1.delete(0, END)

self.Entry2.delete(0, END)

**Update Security Window:-**

from tkinter import \*, from tkinter import messagebox

from PIL import Image, ImageTk, import pymysql.cursors, After\_new

class Security:

def \_\_init\_\_(self, uname):

self.root = Tk()

self.root.geometry('1145x500+160+100')

self.root.title('Update Security Settings')

self.root.configure(background="#ffffcc")

self.id = uname

self.conn = pymysql.connect(host='localhost', user='root', password='noor#123', db='dbbank')

self.cursor = self.conn.cursor()

i0 = ImageTk.PhotoImage(Image.open("bank\_images/setsec.gif"))

self.Label1 = Label(self.root, image=i0)

self.Label1.place(relx=0.39, rely=0.0, width=450, height=60)

i1 = ImageTk.PhotoImage(Image.open("bank\_images/secu.jpg"))

self.Label1 = Label(self.root,image = i1)

self.Label1.place(relx = 0.01,rely = 0.0,width = 250,height = 140)

i2 = ImageTk.PhotoImage(Image.open("bank\_images/secu1.jpg"))

self.Label2 = Label(self.root,image = i2)

self.Label2.place(relx = 0.84,rely = 0.00, width = 200,height = 150)

self.Label3 = Label(self.root, text="Your Security Question is: ", font=('Times New Roman', '14', 'bold'))

self.root.mainloop()

def update(self):

secques = self.Entry1.get()

secans = self.Entry2.get()

if secques == "" or secans == "":

messagebox.showerror('Info', 'Cannot be empty')

else:

self.cursor.execute('update tbadmin set adsecques = %s , adsecans = %s where admin\_id = %s”,(secques, secans, self.id))

self.conn.commit()

messagebox.showinfo('Info', 'Updated successfully')

self.root.destroy()

u=After\_new.Menu\_Class(self.id)

**About Us Window:-**

from tkinter import \*, from PIL import Image, ImageTk, import After\_new

class about\_us:

def \_\_init\_\_(self,uname):

self.root1 = Tk()

self.root1.geometry("1145x900+160+5")

self.root1.title('Banking Solutions')

self.root1.configure(background="#ffffcc")

self.id = uname

i1 = ImageTk.PhotoImage(Image.open("bank\_images/crb.jpg"))

self.Label1 = Label(self.root1, image=i1)

self.Label1.place(relx=0.01, rely=0.0, width=310, height=160)

i2 =ImageTk.PhotoImage(Image.open("bank\_images/auh.jpg"))

self.Label2 = Label(self.root1,image = i2)

self.Label2.place(relx = 0.79,rely = 0.0, width = 270, height = 162)

i3 = ImageTk.PhotoImage(Image.open("bank\_images/cbh2.png"))

self.Label3 = Label(self.root1, image=i3)

self.Label3.place(relx=0.34, rely=0.0, width=470, height=265)

self.Text1 = Text(self.root1, width=100, height=15, wrap='word')

self.Text1.place(relx=0.3, rely=0.44) #replace file

self.Text1.insert('1.0', '''This project has been developed by BHAVNOOR SINGH under the supreme guidance of Er. PUNEET AGGARWAL.

Core (centralized online real-time exchange) banking is a banking service provided by a group of networked bank branches

where customers may access their bank account and perform basic transactions from any of the member branch offices.

A Core banking system(CBS) is defined as a back-end system that processes daily banking transactions, and posts updates

to accounts and other financial records. It is called core because it is a central to the business of the bank.

A core banking system, when implemented well, ensures accurate and error-free delivery

of financial services to customers regardless of where he maintains his account, thus adding to the

banks' efficiency and performance. Thus CBS is a step towards enhancing customer convenience through anywhere and anytime Banking.

Minimum Software Requirements:

1. Operation System : Windows

2. Language : PYTHON 3.6

3. Front End: PyCharm IDE

4. Back End: MySQL Server 5.0''')

self.Text1.config(state='disabled', font=('Times New Roman', '12', 'italic'), background="#d2a679")

self.Text2 = Text(self.root1, width=40, height=15, wrap='word')

self.Text2.place(relx=0.01, rely=0.44) # replace file

self.Text2.insert('1.0', '''CONTACT DETAILS:-

1.MOBILE NO.- +919877308355

2.E-MAIL- bhav27noor@gmail.com

3.ADDRESS- 90, Waryam Villa,Ferozepur

4.POSTAL ADDRESS- 152001''')

self.Text2.config(state='disabled', font=('Times New Roman', '12', 'italic'), background="#d2a679")

self.b = Button(self.root1, command=self.back)

self.b.place(relx=0.47, rely=0.88, width=80, height=68)

i4 = ImageTk.PhotoImage(Image.open("bank\_images/close.gif"))

self.b.configure(image=i4)

self.root1.mainloop()

def back(self):

self.root1.destroy()

zx=After\_new.Menu\_Class(self.id)

**Queries Used:-**

•create database dbbank

•create table tbadmin(admin\_id int primary key, admin\_pass varchar(50), adsecques varchar(50), adsecans varchar(50), admin\_name varchar(50), admin\_el varchar(50), admin\_ph varchar(50))

•create table tbaccount(acc\_no int(50) primary key, c\_name varchar(50), c\_add varchar(50), c\_gender varchar(50), c\_ml varchar(50), c\_mob int(50), c\_fa\_name varchar(50), c\_mo\_name varchar(50), op\_date varchar(50), op\_bal int(50)

•create table tbtransacc(trans\_id int(50) primary key, trans\_src\_acc int(50), trans\_des\_acc int(50), trans\_type varchar(50), trans\_amt int(50), trans\_date varchar(50), trans\_time varchar(50))

•alter table tbtransacc add constraint fk foreign key (trans\_src\_acc) references tbaccount(acc\_no)

•alter table tbtransacc add constraint fk1 foreign key (trans\_des\_acc) references tbaccount(acc\_no)

• alter table tbadmin add admin\_add varchar(50)

•insert into tbadmin values(100,"noor#123","Your

D.O.B?","27/12/1998","Bhavnoor","noor2gmail.com",9876544897,"Fzr")