DATABASE MANAGEMENT SYSTEM
PROJECT REPORT

Submitted To: Dinesh Patel Sir

Submitted By: Abhay Choudhary
Navni Solanki
Gourav Singh Panwar

Bank Management System

- **Problem statement –** XYZ Bank's online banking system struggles to balance security, data inconsistencies, and customer engagement, leading to frustrated customers and financial losses.

- **Motivation & Objective –**

1. **Protect Customer Data:** To safeguard sensitive customer information and prevent unauthorized access, theft, or misuse.

2. **Maintain Trust and Reputation:** To ensure that customers trust XYZ Bank with their financial information and maintain a positive reputation in the industry.

3. **Improve Customer Confidence**: To provide customers with a secure and reliable online banking experience, increasing their confidence in the bank's ability to protect their data.

- **Objective -** To design and implement a secure online banking database management system that ensures the confidentiality, integrity, and availability of customer data, while meeting regulatory compliance requirements.

❑ Overview of the Implementation –

The Bank Management System was built to make handling banking operations easier and more organized. It helps manage customer accounts, track transactions, debit/credit etc. The system runs on a database that follows a normalized structure, ensuring the data is well-organized and efficient. Key features include:

1.Managing accounts
2.Transactions
3.Managing accounts balance
4.Customer management

❑ Solution Statement –

Design and implement a secure online banking database system for XYZ Bank using a relational database management system (RDBMS) that ensures data encryption, access control etc. The solution will utilize a database schema designed to minimize data redundancy and improve data integrity, with indexes and constraints to ensure data consistency. The system will be developed using a RDBMS such as MySQL with a focus on scalability, performance, and security.

❑ Challenges Faced And How We Solved Them –

Learning a new concept of making any management system through python is slightly difficult yet interesting. learning new concept and implementing it through sources leads to the final conclusion.

1. **Designing the Database -** Initially, it was hard to design the database, To fix this, we used normalization to fix the table properly.

2. **Handling Transactions –** It was tricky to manage multiple transactions happening at the same time without messing up the data.

3. **Web Application's alignment –** At first it was hard to fix the designing and pictures according to the sequences but later judged and with trial we came up with the conclusion.

4. **Code Sequence –** Proper arrangement of code from which the the alignment works properly.

SOLUTION (PHASE – 01)

ER – MODEL

- **Entities –**

1. Customer
2. Loan
3. Account
4. Branch
5. Banker
6. Transaction

- **Attributes –**

1. Customer –

- Customer_id
- Customer_name
- Customer_contact
- Customer_age

2. Loan –

- Loan_id
- Loan_Type
- Amount

3. Account –

- Account_number
- Account_Type
- Balance

4. Branch –

- Branch_id
- Location
- Branc_contact

5. Banker –

- Position
- Banker_name
- Banker_id

6. Transaction –

- Transaction_id
- Amount
- Date
- Transaction_status
- Time
- Transaction_type

- **Key Attributes –**

1. Customer ID
2. Loan ID
3. Account Number
4. Banker ID
5. Branch ID
6. Transaction Id

- **Derived Attributes –**

1. Customer Age
2. Loan Amount
3. Account Balance

- **Multivalued Attributes –**

1. Customer Contact Number
2. Branch Contact Number

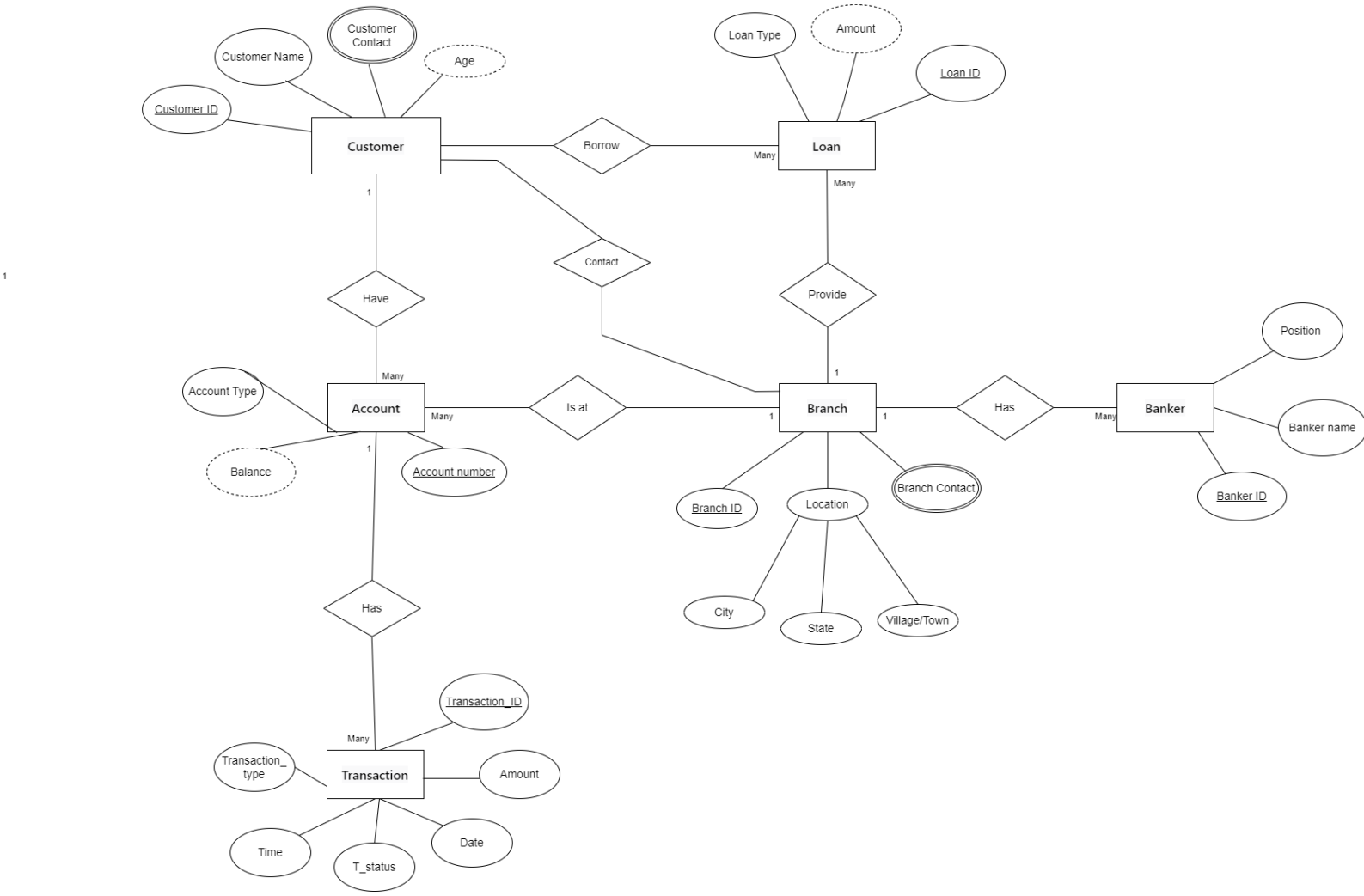
- **Relationships -**

1. A Customer Have Accounts
2. A Customer Borrows Loans
3. Loans provided by Branch
4. Branch Has Banker
5. Accounts is at Branch
6. A Customer Contact with Branch
7. Account has Transaction

- **Cardinality Constraints -**

1. A Customer can Have Many Accounts. (One : Many)
2. A Customer Can Borrows Many Loans. (One : Many)
3. A Branch can provide many Loans. (One : Many)
4. Bankers are work in a Branch. (Many : One)
5. In a Branch there are so much accounts. (One : Many)
6. A Customer Contact can contact with any Branch. (One : Many)
7. An Account can have multiple Transaction. (One : Many)

ER – MODEL DIAGRAM



ER Model Drive Link – https://drive.google.com/file/d/156ZTORlaGuS3JYgoKX1xdWHVpnVfxaQe/view?usp=drive_link

SOLUTION (PHASE – 02)

CREATING DATABASE

- The tables in our database are as follows: accounts, branch, customer, employee, employeeinfo and transaction.

```
mysql> show tables;
+-------------------------+
| Tables_in_assignment_3 |
+-------------------------+
| accounts                |
| branch                  |
| customer                |
| employee                |
| employeeinfo            |
| transaction             |
+-------------------------+
6 rows in set (0.02 sec)
```

SOLUTION (PHASE – 03)

FINAL IMPLEMENTATION WITH PYTHON

❑ CODE FOR LOGIN PAGE –

- **PURPOSE –** This code is a graphical user interface (GUI) for a Bank Database Management system, created using the Tkinter library in Python. It provides a login screen where users can enter their username and password, and upon successful authentication, it welcomes the user and presumably proceeds to the main database management system. The code also includes error handling for empty or incorrect login credentials.

- **CODE –**

```
from tkinter import *
from PIL import ImageTk,Image
from tkinter import messagebox
def logfun():
   if user.get()=='' or password.get()=='':
      messagebox.showerror('Error', 'Please Enter User Name and Password.....!')
   elif user.get()=='Abhay' and password.get()=='au23b1003':
      messagebox.showinfo('Success',f'Welcome {user.get()} in Bank Management System...!')
      home.destroy()
      import maindb
   else:
      messagebox.showerror('Failure','Check your User Name and Password.....!')
home = Tk()
home.title('BANK DATABASE MANAGEMENT')
home.geometry('1566x790')
home.resizable(True,True)
```

```
#backgroundimage
img=Image.open("bgbank.png")
img=img.resize((1566,790))
bg= ImageTk.PhotoImage(img)
Label(home,image=bg,bg='white').place(x=0,y=0)
#frame
frame=Frame(home,width=600,height=400,bg="#095e7d")
frame.place(x=100,y=120)
heading=Label(frame,text='Login',fg='white',bg='#095e7d',font=('Poppi
ns',19,'bold'))
heading.place(x=170,y=5)
#id password
#ID
ulogo= PhotoImage(file='user1.png')
Label(frame,image=ulogo,width=50,height=50,bg='#095e7d',fg='white')
.place(x=38,y=59)
Label(frame,text='User
Id',fg='white',bg='#095e7d',font=('Poppins',14)).place(x=70,y=90)
user                                                                =
Entry(frame,width=45,fg='black',border=2,bg="white",font=('Poppins',1
1))
user.place(x=30,y=120)
#PASSWORD
passwordlogo= PhotoImage(file='password1.png')
Label(frame,image=passwordlogo,width=50,height=50,bg='#095e7d',fg
='white').place(x=39,y=159)
Label(frame,text='Password',fg='white',bg='#095e7d',font=('Poppins')).
place(x=70,y=190)
password                                                            =
Entry(frame,width=45,show='*',fg='black',border=2,bg="white",font=('P
oppins',11))
password.place(x=30,y=220)
#button
Button(frame,width=20,pady=7,text='Login',bg='#57a1f8',fg='white',bor
der=0,command=logfun).place(x=140 ,y=280)
home.mainloop()
#home screen
```

❑ CODE FOR ADD ACCOUNT –

- **PURPOSE –** This code is a graphical user interface (GUI) for adding a new bank account to a database management system. It provides a window with input fields for customer information (name, address, contact, email) and account details (account type, initial balance). When the "Add Account" button is clicked, the code inserts the customer and account information into the database, retrieves the newly created account details, and updates a data frame (presumably a GUI component) with the new account information.

- **CODE –**

```
def addaccount():
wwindow = Tk()
wwindow.title("Add Account")
wwindow.geometry("400x450")
cnamelabel = Label(wwindow, text="Enter Customer Name:")
cnamelabel.pack()
cnameentry = Entry(wwindow, width=30)
cnameentry.pack()
caddressslabel = Label(wwindow, text="Enter Customer Address:")
caddressslabel.pack()
caddresssentry = Entry(wwindow, width=30)
caddresssentry.pack()
ccontactlabel = Label(wwindow, text="Enter Customer Contact:")
ccontactlabel.pack()
ccontactentry = Entry(wwindow, width=30)
ccontactentry.pack()
cemaillabel = Label(wwindow, text="Enter Customer Email
(optional):")
```

```
cemaillabel.pack()
cemailentry = Entry(wwindow, width=30)
cemailentry.pack()
atypelabel = Label(wwindow, text="Enter Account Type (Current, Saving, FD):")
atypelabel.pack()
atypeentry = Entry(wwindow, width=30)
atypeentry.pack()
blabel = Label(wwindow, text="Enter Initial Balance:")
blabel.pack()
bentry = Entry(wwindow, width=30)
bentry.pack()
def add_account():
customer_name = cnameentry.get()
customer_address = caddresssentry.get()
customer_contact = ccontactentry.get()
customer_email = cemailentry.get()
account_type = atypeentry.get()
initial_balance = int(bentry.get())
cur.execute("INSERT INTO customer (Name, Address, Contact, Email) VALUES (%s, %s, %s, %s)", (customer_name, customer_address, customer_contact, customer_email))
conn.commit()
cur.execute("SELECT CId FROM customer WHERE Name = %s AND Address = %s AND Contact = %s AND Email = %s", (customer_name, customer_address, customer_contact, customer_email))
customer_id = cur.fetchone()[0]
cur.execute("INSERT INTO accounts (CId, AType, Balance) VALUES (%s, %s, %s)", (customer_id, account_type, initial_balance))
conn.commit()
cur.execute("SELECT a.AId, c.Name, a.AType, a.Balance, c.Address, c.Contact, c.Email FROM accounts a JOIN customer c ON a.CId = c.CId WHERE a.AId = (SELECT MAX(AId) FROM accounts)")
rows = cur.fetchall()
for row in rows:
```

```
dframe.insert('', 'end', values=row)
wwindow.destroy()
mainbank.update_idletasks()
add_button = Button(wwindow, text="Add Account",
command=add_account)
add_button.pack()
wwindow.mainloop()
```

❑ GIT-HUB LINK FOR THE ENTIRE PYTHON CODE –

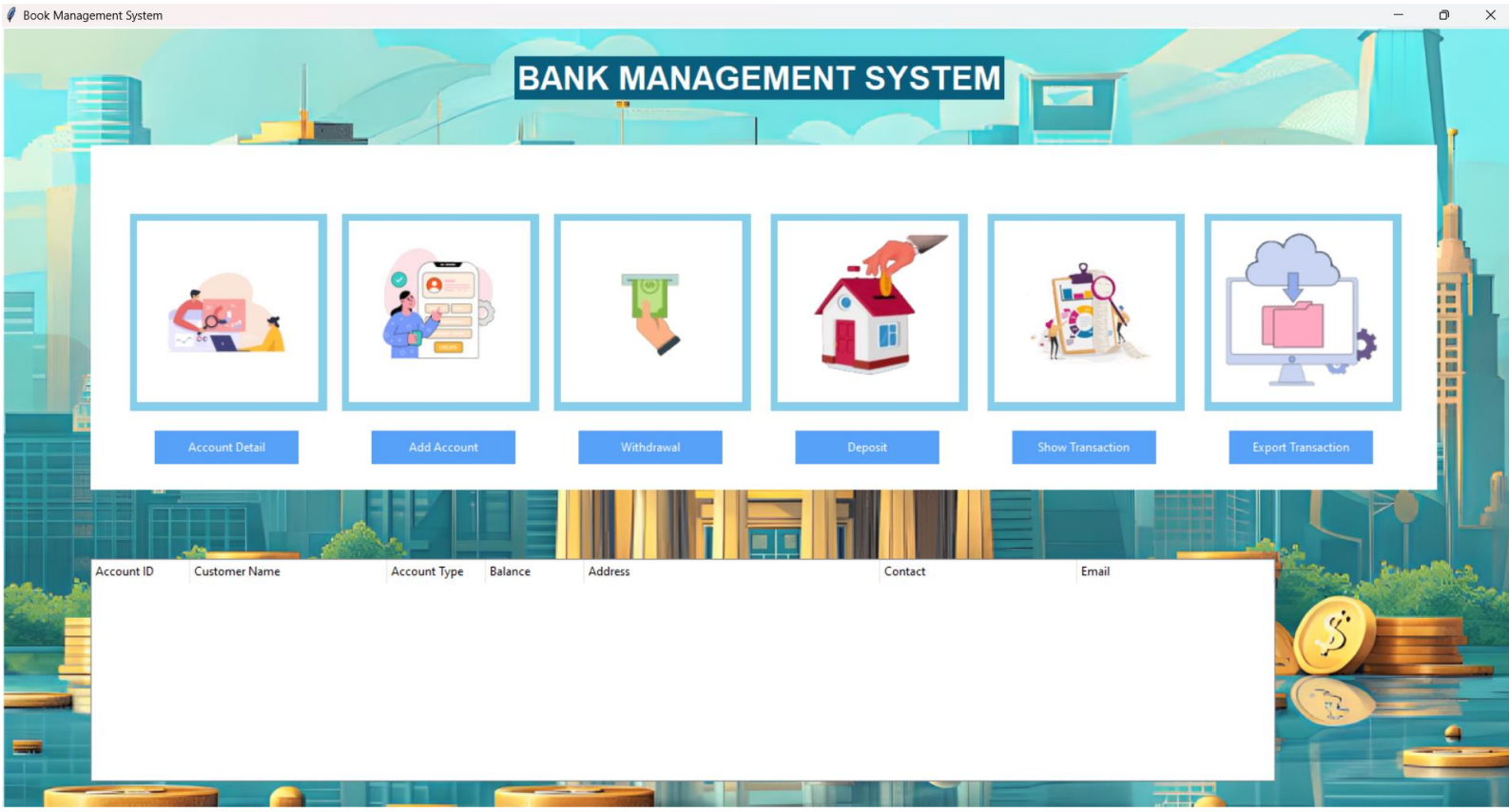
https://github.com/abhaychoudhary11/Bank-Database-management-System

REMEMBER – In the git hub link we have two files named as – login.py for the login page and maindb.py for the home page and all the functionalities of our system.

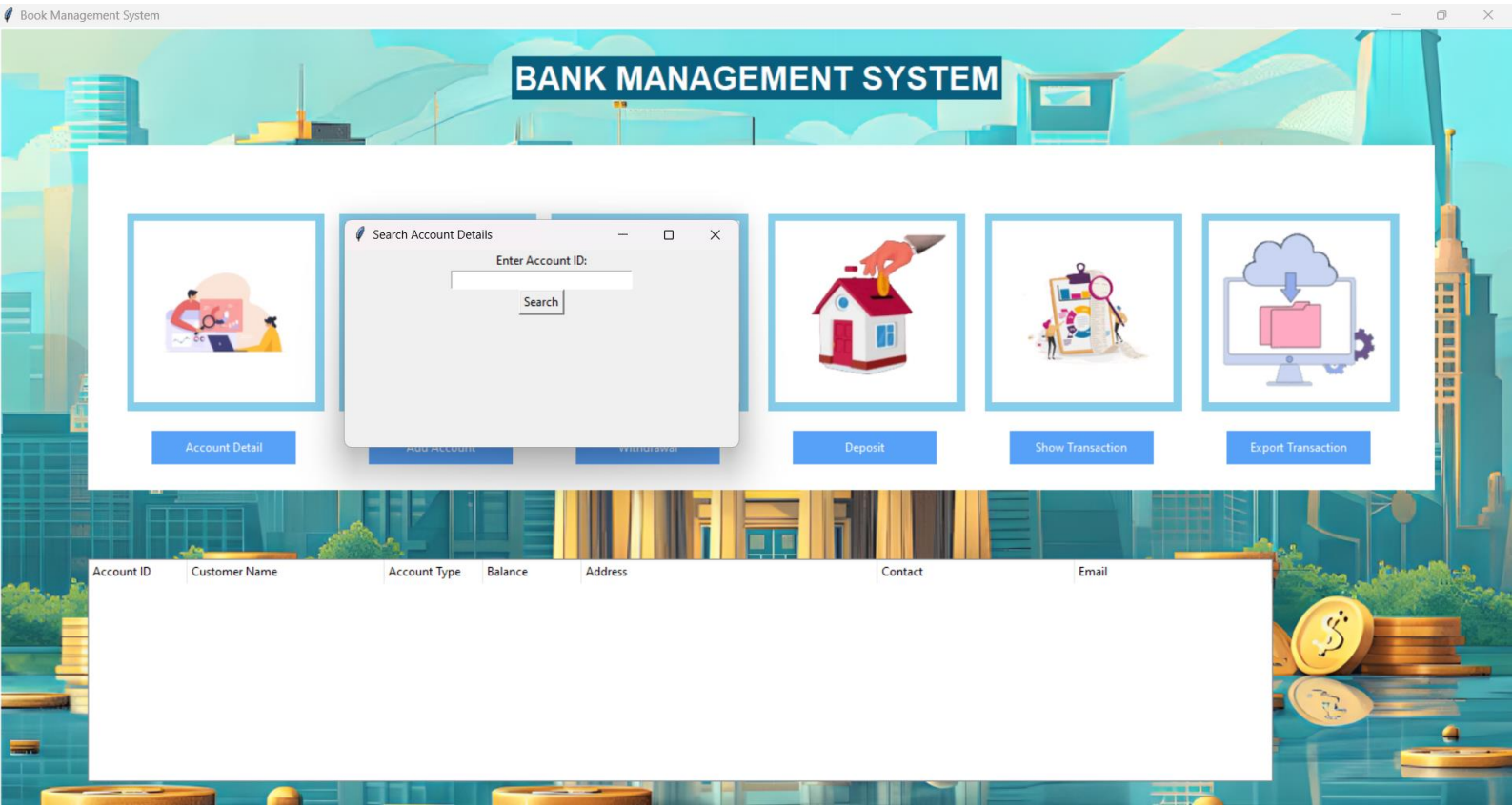
❑ USER FLOW

Step 1 – Login page from where a user can login to our bank management system with the user id and password.

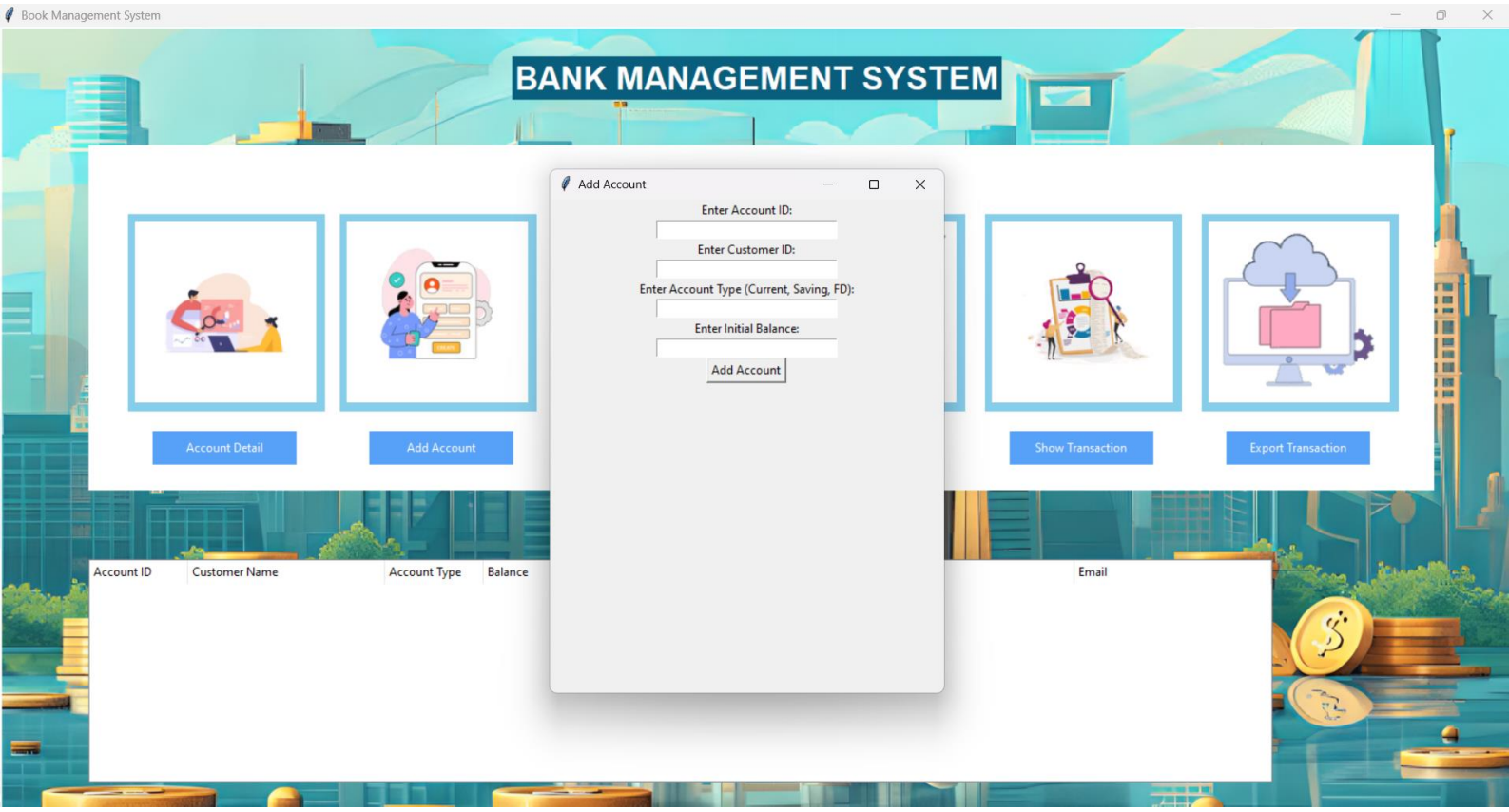
Step 2 – Home Screen where our bank management system has six functionalities.



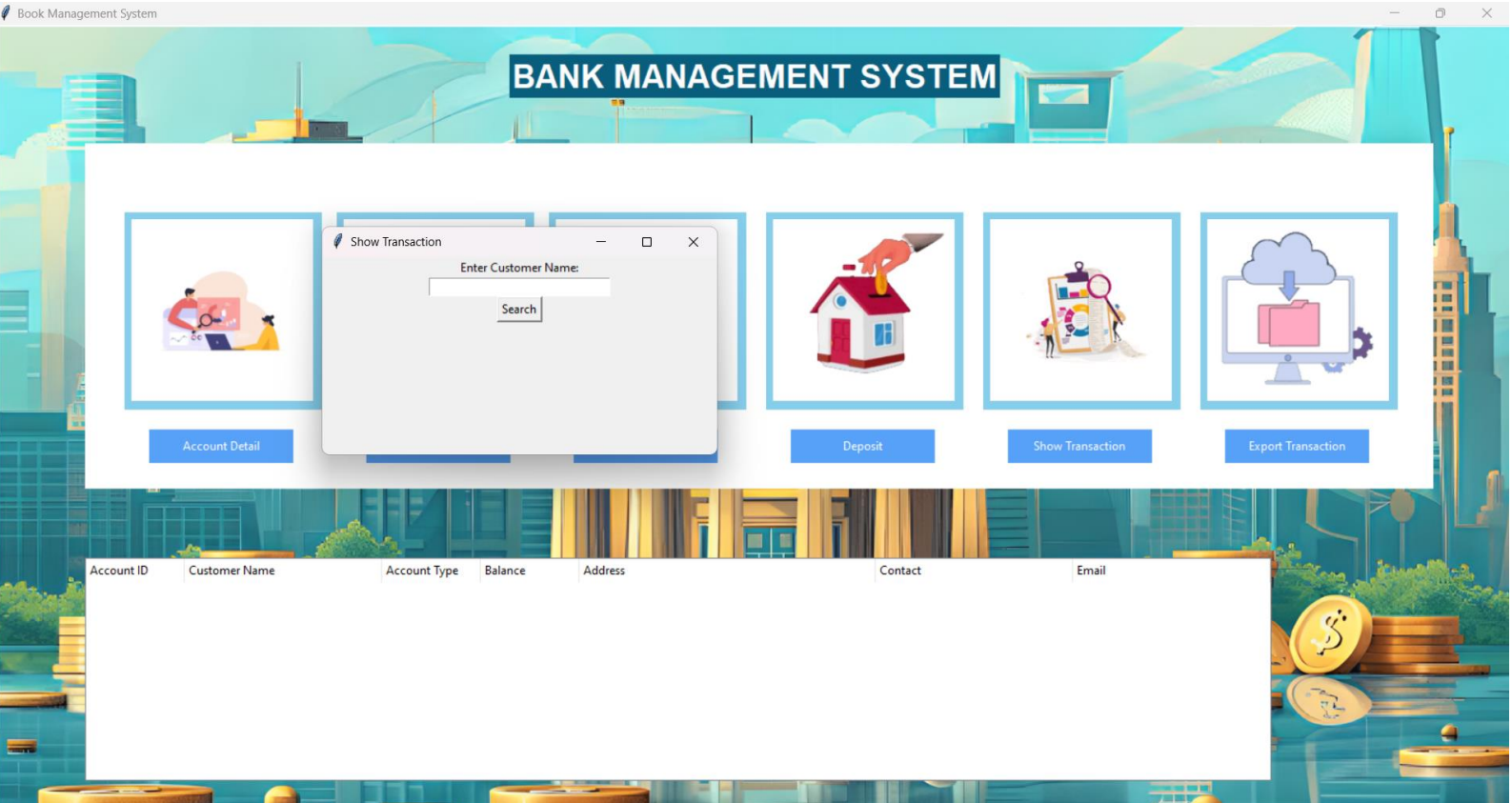
Functionality 1 – <u>Account Detail</u> from where a user can check his account details.



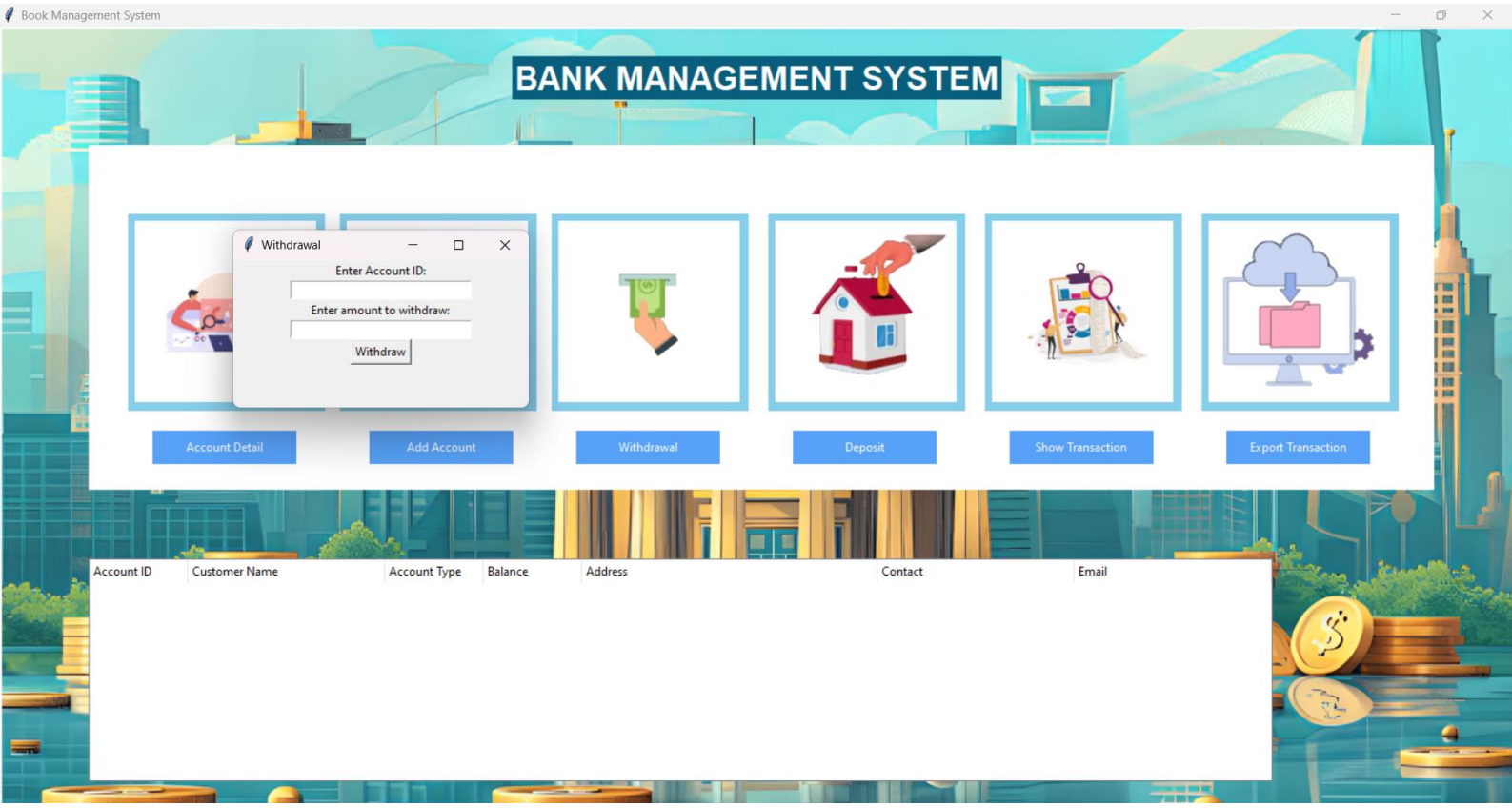
Functionality 2 – Add Account from where a new account can be added to the database.



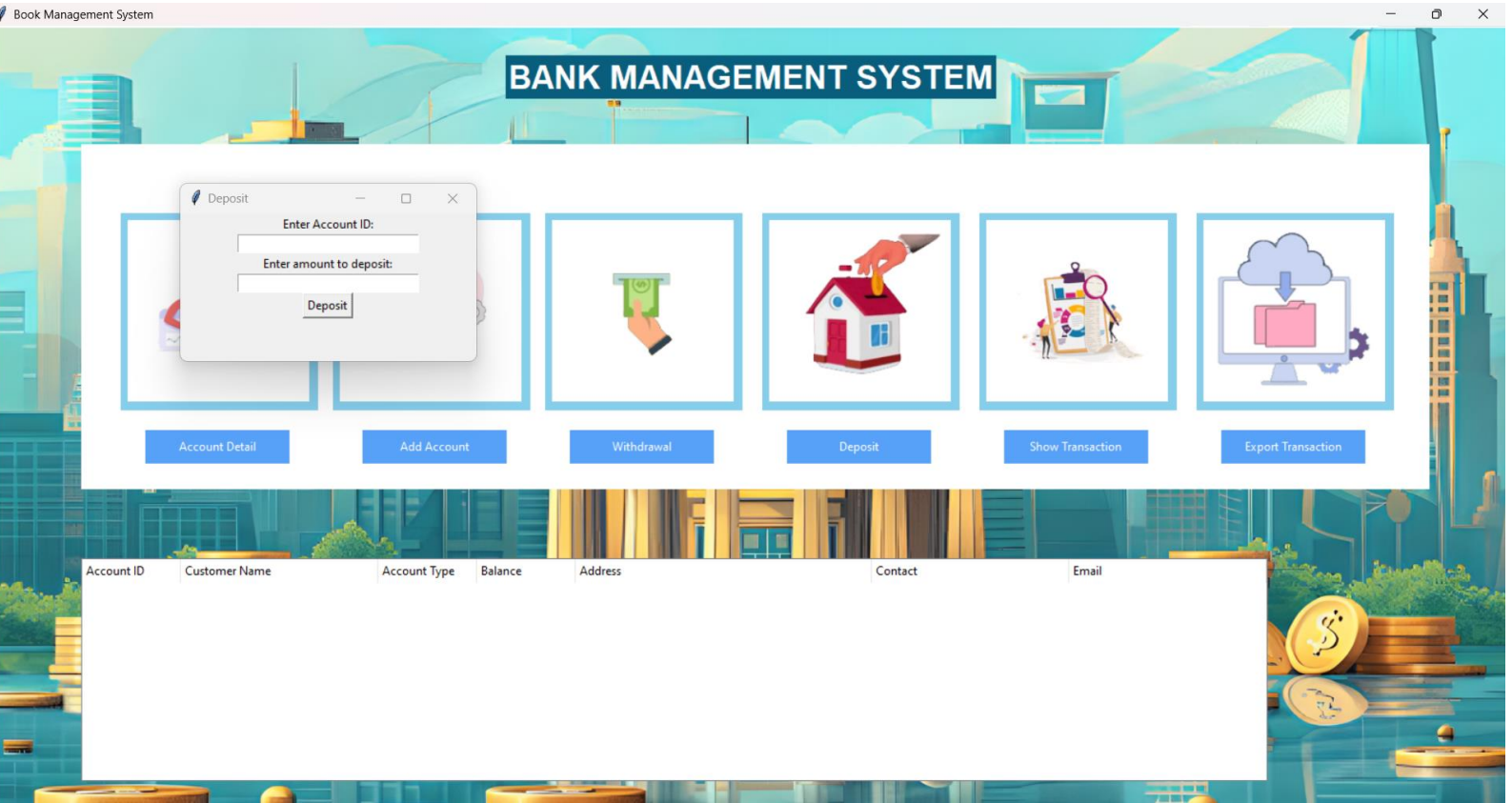
Functionality 3 – Show Transaction from where all the transaction history can be viewed easily.



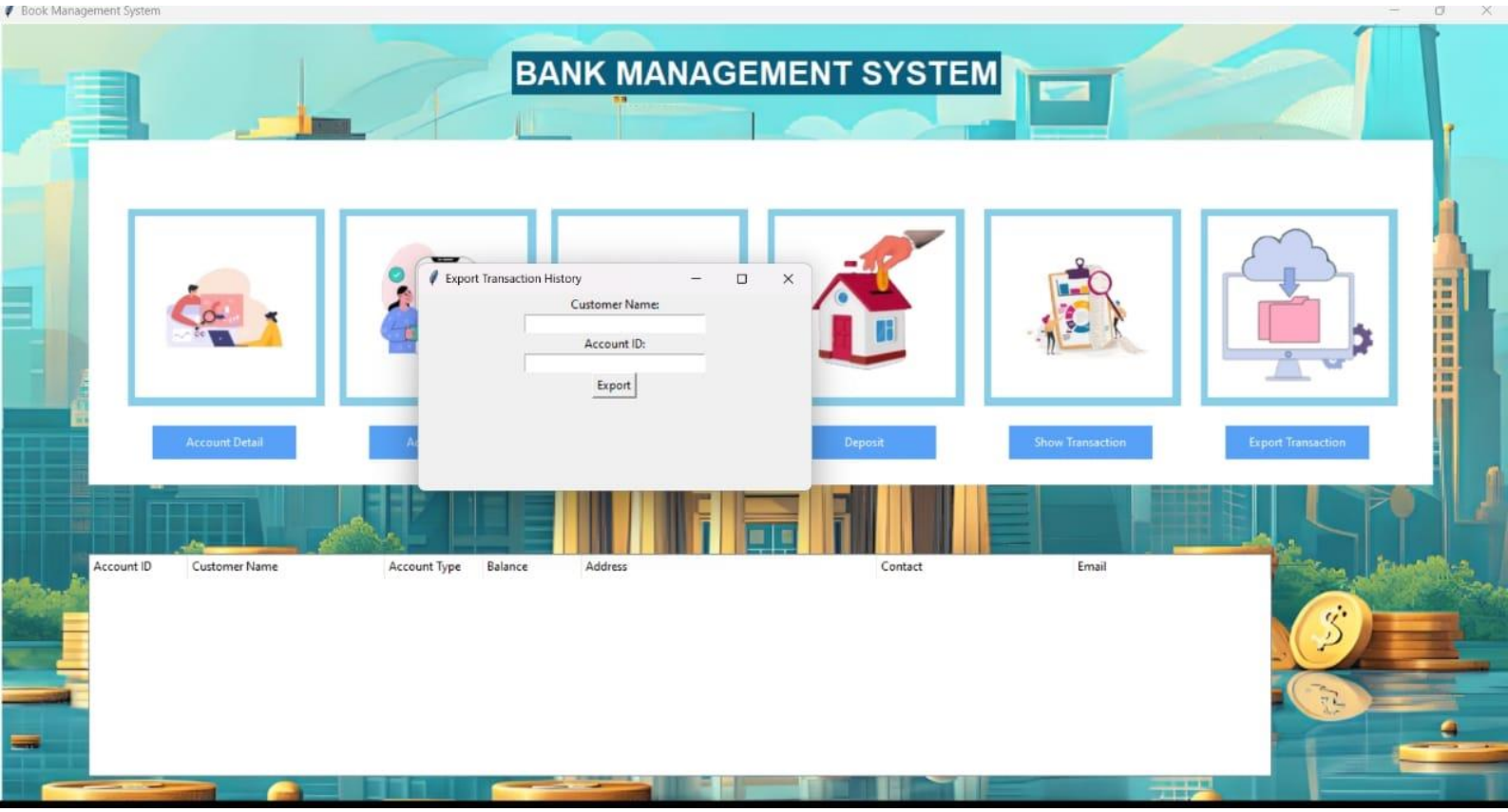
Functionality 4 – Withdraw from where amount can be withdraw only if the account has sufficient balance other wise it will show no sufficient balance.



Functionality 5 – Deposit from where we can add some amount to our account/database.



Functionality 6 – Export Transaction from where we can download the transaction file to our system in one click.



THANK YOU !!!