A Real Time Project Report on

" Bank Management System"

*Submitted to the*

# GURU NANAK INSTITUTIONS TECHNICAL CAMPUS (AUTONOMOUS)

*In partial fulfilment of the requirement for the award of the degree of*

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

BY

SANDIP MANDAL(22WJ1A05U2)

Under the Esteemed Guidance Of

Mr. Devdas Saraswat

Associate Professor of CSE, GNITC



Department Of Computer Science Engineering

GURU NANAK INSTITUTIONS TECHNICAL CAMPUS (AUTONOMOUS)

School of Engineering and

Technology, Ibrahimpatnam, R.R District 501506, 2023-2024

**I**



**Department of Computer Science & Engineering**

**CERTIFICATE**

This is to certify that this Real Time project entitled **"BANK MANAGEMENT SYSTEM"** being submitted by **SANDIP MANDAL (22WJ1A05U2),** in partial fulfilment for the award of the Degree of **Bachelor of Technology** in **Computer Science** & **Engineering** of the **GuruNanak Institution Technical Campus, Hyderabad** during the academic year 2023 - 2024, is a record of bonafide work carried out under our guidance and supervision at **Guru Nanak Institutions Technical Campus (Autonomous)**

**Mr. Devdas Saraswat Mr. V. Devasekhar Mrs.Dr . Geeta Tripathi**

RTP PROJECT COORDINATORACADEMIC COORDINATOR HOD CSE

**II**

**PROJECT COMPLETION CERTIFICATE**

This is to certify that the following students of Second year B. Tech, Department Of **Computer Science and Engineering**- Guru Nanak Institutions Technical Campus (GNITC) have completed their training and project at GNITC successfully.

## STUDENT NAME

1. **SANDIP MANDAL**
2. ROLL NO **22WJ1A05U2**

The project titled **"BANK MANAGMENT SYSTEM"** in **june 2024.** The project has been completed in all aspects.

# SIGNATURE

**III**

**ACKNOWLEDGEMENT**

I wish to express my sincere thanks to **Dr. Rishi Sayal, Professor & Associate Director, GNITC** for providing me with the conductive environment for carrying through my academic schedules and project with ease.

I have been truly blessed to have a wonderful adviser **Dr. Geeta Tripathi** & **HOD of CSE, GNITC** for guiding me to explore the ramification of my work and I express my sincere gratitude towards him for leading me throughout the completion of the project.

I would like to say my sincere thanks to **Mr. Devdas Saraswat, Associate Professor, Department of CSE, RTP** Project Coordinator, for providing seamless support and right suggestions are given in the completion of the project.

I specially thank my internal guide **Mr. V. Devasekhar, Associate Professor** & **Academic coordinator of CSE, GNITC** for his suggestions and constant guidance in every stage of the project.

Finally, I would like to thank my family members for their moral support and encouragement to achieve goals

**SANDIP MANDAL**           **(22WJ1A05U2)**

**IV**

# ABSTRACT

The bank management system project is a program that keeps track of a client’s bank account. This project demonstrates the operation of a banking account system and covers the essential functions of management software. It develops a project for resolving a customer’s financial applications in a banking environment to meet the needs of an end banking user by providing multiple ways to complete banking chores.Additionally, this project is to provide additional features to the user’s .Work space that are not available in traditional banking project. The bank management system is built on cutting-edge technologies. This project’s main goal is to create software for a bank account management system. This project was designed to make it simple and quick to complete previously impossible processes with manual systems which are now possible with this software.This project is developed using Python, a popular programming language known for its simplicity and ease of use. With the Bank Management System project in Python, you can manage customer accounts, process transactions,generate reports, and much more.This project is not only useful for banks but it can be used for learners to build a bank management college project. The system provides a user-friendly interface that enables users to perform transactions easily and quickly.

Overall, this Bank Management System project in Python is a practical and valuable tool for anyone looking to streamline their financial operations.

**V**

## 

## TABLE OF CONTENT



|  |  |  |
| --- | --- | --- |
| **CHAPTER**  **NO.** | **TITLE** | **PAGE NO** |
|  | **INDEX**  **CERTIFICATE**  **ACKNOWLEDGEMENT**  **ABSTRACT**  **TABLE OF CONTENT**  **LIST OF FIGURES** | I  II  III  IV  VI  VII |
| 1. | **CHAPTER 1: INTRODUCTION**   * 1. Advantages   2. Disadvantages   3. Application | **1-2** |
| 2. | **CHAPTER 2: ANALYSIS**  2.1 Requirement Analysis  2.2 Feasibility Study  2.3 Risk Analysis | **3** |
| 3. | **CHAPTER 3: DESIGN ENGINEERING**  3.1System Architecture  3.2 Database Design  3.3 User Interface Design  3.4Security Design | 4 |
| 4. | **CHAPTER 4: IMPLEMENTATION**  4 Implementation using python  4.1 Django/Flask  4.2 Source Code  4.3 SQL  4.4 Pandas /Numpy  4.5 PyCrypto  4.6 Coding | **5-9** |

**VI**

|  |  |  |
| --- | --- | --- |
| 5. | **CHAPTER 5: SOFTWARE TESTING**  5 Testing Methodologies  5.1 Unit Testing  5.2 Intergrating Testing  5.3 Performance Testing  5.4 Security Testing | **10** |
| 6 | **CHAPTER 6: POST IMPLEMENTATION**  6.1 Monitoring and maintance  6.2 Security Audits  6.3 Performance and Optimization  6.4 Backup and Disaster Recovery | **11** |
| 7. | **CHAPTER 7:RESULT**  7.1 Result Analysis | **12-14**  **15** |
| 8 | **CHAPTER 8:CONCLUSION**  8.1 Conclusion |  |

### LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 5.1 | SYSTEM ARCHITECTURE | 4 |
| 5.2 | USER CASE DIAGRAM | 10 |
| 5.3 | ACTIVITY DIAGRAM | 10 |
| 5.4 | CLASS DIAGRAM | 17 |
| 5.5 | DATA-FLOW DIAGRAM |  |

**VII**

## CHAPTER 1

## INTRODUCTION



* 1. **General**

A Bank Management System (BMS) is a software application designed to handle the operations of a bank. This system facilitates the management of accounts, transactions, customer details, loans, and other financial activities. Implementing a BMS using Python leverages the language's simplicity and extensive libraries, enabling the development of robust and scalable applications.

* + 1. **Advantages**

· **Efficiency and Speed**:

* Automates repetitive tasks, reducing manual workload.
* Processes transactions and updates records in real-time, improving service speed.

· **Accuracy**:

* Minimizes human errors in calculations and data entry.
* Ensures consistent data management and reporting.

· **Security**:

* Implements robust security measures to protect sensitive customer information.
* Uses encryption and authentication to safeguard transactions.

· **Customer Service**:

* ·Enhances customer experience with quick and accurate service.
* Provides features like online banking, fund transfers, and account management.

· **Data Management**:

* Centralizes data storage, making information retrieval easy and efficient.
* Facilitates data analysis and reporting for informed decision-making.
  + 1. **Disadvantages**

· **Implementation Cost**:

* Initial setup and integration with existing systems can be expensive.
* Requires investment in hardware, software, and training.

**1**

· **Maintenance**:

* Regular updates and maintenance are needed to ensure security and functionality.
* Downtime during maintenance can disrupt services.

· **Cybersecurity Risks**:

* Despite security measures, systems are still vulnerable to cyber attacks.
* Requires continuous monitoring and updating of security protocols.

· **Technical Issues**:

* Software bugs or system failures can impact operations.
* Dependency on technology requires skilled personnel for troubleshooting.

**1.1.3 Application**

· **Customer Account Management**:

* Opening and closing accounts.
* Managing account details and balances.

· **Transaction Management**:

* Processing deposits, withdrawals, and transfers.
* Recording and tracking transactions.

· **Loan Management**:

* Managing loan applications, approvals, and repayments.
* Calculating interest and maintaining loan records.

· **Customer Service**:

* Providing customer support and query resolution.
* Enabling online banking and mobile banking features.

· **Financial Reporting**:

* Generating reports on transactions, balances, and financial performance.
* Analyzing data for strategic planning and compliance.

**2**

1

## CHAPTER -2 ANALYSIS



**1.** **Requirement Analysis**:

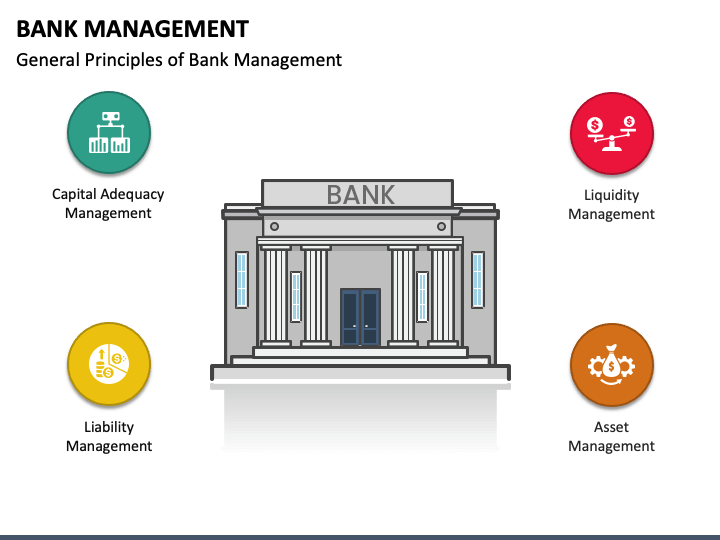
* Identify the needs of the bank and its customers.
* Define system functionalities and features.

**· 2.** **Feasibility Study**:

* Assess the technical, financial, and operational feasibility of the system.
* Determine the resources required for implementation.

**·3**. **Risk Analysis**:

* Identify potential risks such as data breaches, system failures, and compliance issues.
* Develop mitigation strategies.



**3**

**CHAPTER-3 DESIGN**



**3.**

**3.1 System Architecture**:

* Design a multi-tier architecture with a user interface, business logic layer, and database layer.
* Ensure scalability and flexibility to accommodate future growth.

**3.2** **Database Design**:

* Create a relational database schema to manage accounts, transactions, customers, and loans.
* Ensure data integrity and security.

**3.3** **User Interface Design**:

* Develop an intuitive and user-friendly interface for both staff and customers.
* Implement responsive design for accessibility on various devices.

**3.4 Security Design**:

* Integrate authentication, authorization, and encryption mechanisms.
* Implement secure coding practices to prevent vulnerabilities.

**4**

**CHAPTER -4**

**IMPLEMENTATION & CODING**



**4.**Implementation Using Python

#### Libraries and Frameworks

#### ****4.1 Django/Flask**:**

Frameworks for developing web applications with robust backend support.

Facilitates rapid development and scalability.

**4.2 SQLAlchemy/Django ORM**:

ORM libraries for database interaction.

Simplifies database queries and management.

**4.3 Pandas/Numpy**:

Libraries for data manipulation and analysis.

Useful for generating financial reports and analytics.

**4.4 PyCrypto/Fernet**:

Libraries for encryption and security.

Ensures data confidentiality and integrity.

**CODING:**

import tkinter as tk

from tkinter import messagebox

class Account:

def \_\_init\_\_(self, account\_number, name, initial\_deposit):

self.account\_number = account\_number

self.name = name

**5**

self.balance = initial\_deposit

self.transactions = [("Initial deposit", initial\_deposit)]

def deposit(self, amount):

if amount > 0:

self.balance += amount

self.transactions.append(("Deposit", amount))

return True

else:

messagebox.showerror("Error", "Deposit amount must be positive.")

return False

def withdraw(self, amount):

if 0 < amount <= self.balance:

self.balance -= amount

self.transactions.append(("Withdrawal", amount))

return True

else:

messagebox.showerror("Error", "Invalid withdrawal amount.")

return False

def get\_balance(self):

return self.balance

def get\_statement(self):

return self.transactions

def get\_account\_info(self):

return {

"account\_number": self.account\_number,

"name": self.name,

"balance": self.balance

}

class Bank:

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

self.accounts = {}

def create\_account(self, account\_number, name, initial\_deposit):

if account\_number not in self.accounts:

self.accounts[account\_number] = Account(account\_number, name, initial\_deposit)

messagebox.showinfo("Success", f"Account {account\_number} created successfully.")

else:

messagebox.showerror("Error", "Account number already exists.")

def delete\_account(self, account\_number):

if account\_number in self.accounts:

del self.accounts[account\_number]

messagebox.showinfo("Success", f"Account {account\_number} deleted successfully.")

**6**

else:

messagebox.showerror("Error", "Account number not found.")

def get\_account(self, account\_number):

if account\_number in self.accounts:

return self.accounts[account\_number]

else:

messagebox.showerror("Error", "Account number not found.")

return None

class BankManagementGUI:

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

self.bank = Bank()

self.root = root

self.root.title("Bank Management System")

self.create\_widgets()

def create\_widgets(self):

# Create labels and entries

self.label\_account\_number = tk.Label(self.root, text="Account Number:")

self.label\_account\_number.grid(row=0, column=0, padx=10, pady=5)

self.entry\_account\_number = tk.Entry(self.root)

self.entry\_account\_number.grid(row=0, column=1, padx=10, pady=5)

self.label\_name = tk.Label(self.root, text="Name:")

self.label\_name.grid(row=1, column=0, padx=10, pady=5)

self.entry\_name = tk.Entry(self.root)

self.entry\_name.grid(row=1, column=1, padx=10, pady=5)

self.label\_amount = tk.Label(self.root, text="Amount:")

self.label\_amount.grid(row=2, column=0, padx=10, pady=5)

self.entry\_amount = tk.Entry(self.root)

self.entry\_amount.grid(row=2, column=1, padx=10, pady=5)

# Create buttons for operations

self.button\_create\_account = tk.Button(self.root, text="Create Account", command=self.create\_account)

self.button\_create\_account.grid(row=3, column=0, columnspan=2, padx=10, pady=5, sticky=tk.W+tk.E)

self.button\_delete\_account = tk.Button(self.root, text="Delete Account", command=self.delete\_account)

self.button\_delete\_account.grid(row=4, column=0, columnspan=2, padx=10, pady=5, sticky=tk.W+tk.E)

self.button\_deposit = tk.Button(self.root, text="Deposit", command=self.deposit)

self.button\_deposit.grid(row=5, column=0, columnspan=2, padx=10, pady=5, sticky=tk.W+tk.E)

**7**

self.button\_withdraw = tk.Button(self.root, text="Withdraw", command=self.withdraw)

self.button\_withdraw.grid(row=6, column=0, columnspan=2, padx=10, pady=5, sticky=tk.W+tk.E)

self.button\_balance = tk.Button(self.root, text="Check Balance", command=self.check\_balance)

self.button\_balance.grid(row=7, column=0, columnspan=2, padx=10, pady=5, sticky=tk.W+tk.E)

self.button\_statement = tk.Button(self.root, text="Account Statement", command=self.account\_statement)

self.button\_statement.grid(row=8, column=0, columnspan=2, padx=10, pady=5, sticky=tk.W+tk.E)

def create\_account(self):

account\_number = self.entry\_account\_number.get()

name = self.entry\_name.get()

amount = float(self.entry\_amount.get())

self.bank.create\_account(account\_number, name, amount)

def delete\_account(self):

account\_number = self.entry\_account\_number.get()

self.bank.delete\_account(account\_number)

def deposit(self):

account\_number = self.entry\_account\_number.get()

amount = float(self.entry\_amount.get())

account = self.bank.get\_account(account\_number)

if account:

if account.deposit(amount):

messagebox.showinfo("Success", "Deposit successful.")

def withdraw(self):

account\_number = self.entry\_account\_number.get()

amount = float(self.entry\_amount.get())

account = self.bank.get\_account(account\_number)

if account:

if account.withdraw(amount):

messagebox.showinfo("Success", "Withdrawal successful.")

def check\_balance(self):

account\_number = self.entry\_account\_number.get()

account = self.bank.get\_account(account\_number)

if account:

**8**

balance = account.get\_balance()

messagebox.showinfo("Balance", f"Current balance: {balance}")

def account\_statement(self):

account\_number = self.entry\_account\_number.get()

account = self.bank.get\_account(account\_number)

if account:

statement = account.get\_statement()

statement\_str = "\n".join([f"{transaction[0]}: {transaction[1]}" for transaction in statement])

messagebox.showinfo("Account Statement", statement\_str)

def main():

root = tk.Tk()

app = BankManagementGUI(root)

root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

main()

## i

## 

## 9

## CHAPTER-5

## **TESTING**

## 

Testing is an essential phase in the development lifecycle of a Bank Management System (BMS). It ensures that the system is reliable, secure, and performs as expected under various conditions. In this chapter, we will delve into the details of different types of testing, the process of setting up a testing environment in Django, and examples of unit tests, integration tests, performance tests, and security tests.

### 5.1 Types of Testing

**5.1 Unit Testing**:

Tests individual components or functions of the system

Ensures that each part of the system works correctly in isolation

**5.2 Integration Testing**:

Tests the interaction between different components.

Ensures that the integrated components work together as expected.

**5.3 Functional Testing**

Tests the system against the functional requirements/specifications.

Ensures that the system performs the intended functions correctly.

**5.4 Performance Testing**:

Tests the system’s performance under various conditions.

Measures the responsiveness, throughput, and scalability of the system.

**5.5 Security Testing**:

Tests the system’s security features.

Ensures that the system is protected against threats and vulnerabilities.

**10**

## CHAPTER-6

**POST IMPLEMENTATION**

### Post-Implementation Activities

**6.1 Monitoring and Maintenance**

* **System Monitoring**: Continuously monitor the performance and health of the BMS. This includes tracking key performance indicators (KPIs) such as response times, transaction processing speeds, system uptime, and error rates.
* **Regular Maintenance**: Perform routine maintenance tasks to ensure the system remains secure and efficient. This includes applying software updates, patches, and performing database optimizations.

**6.2 Security Audits**

* **Regular Audits**: Conduct regular security audits to identify and address potential vulnerabilities in the system. This includes reviewing access controls, encryption methods, and security protocols.
* **Compliance**: Ensure the system complies with relevant regulations and standards, such as GDPR (General Data Protection Regulation) and PCI-DSS (Payment Card Industry Data Security Standard)

.6.3 **Performance Optimization**

* **Load Testing**: Perform periodic load testing to ensure the system can handle increased traffic and transaction volumes, especially during peak times.
* **Scalability**: Continuously assess and improve the system's scalability to accommodate future growth and additional features.

**6.4 Backup and Disaster Recovery**

* **Data Backup**: Implement a robust backup strategy to regularly back up critical data and ensure it can be quickly restored in case of data loss or corruption.
* **Disaster Recovery Plan**: Develop and maintain a disaster recovery plan to ensure business continuity in the event of a system failure, cyberattack, or natural disaster.

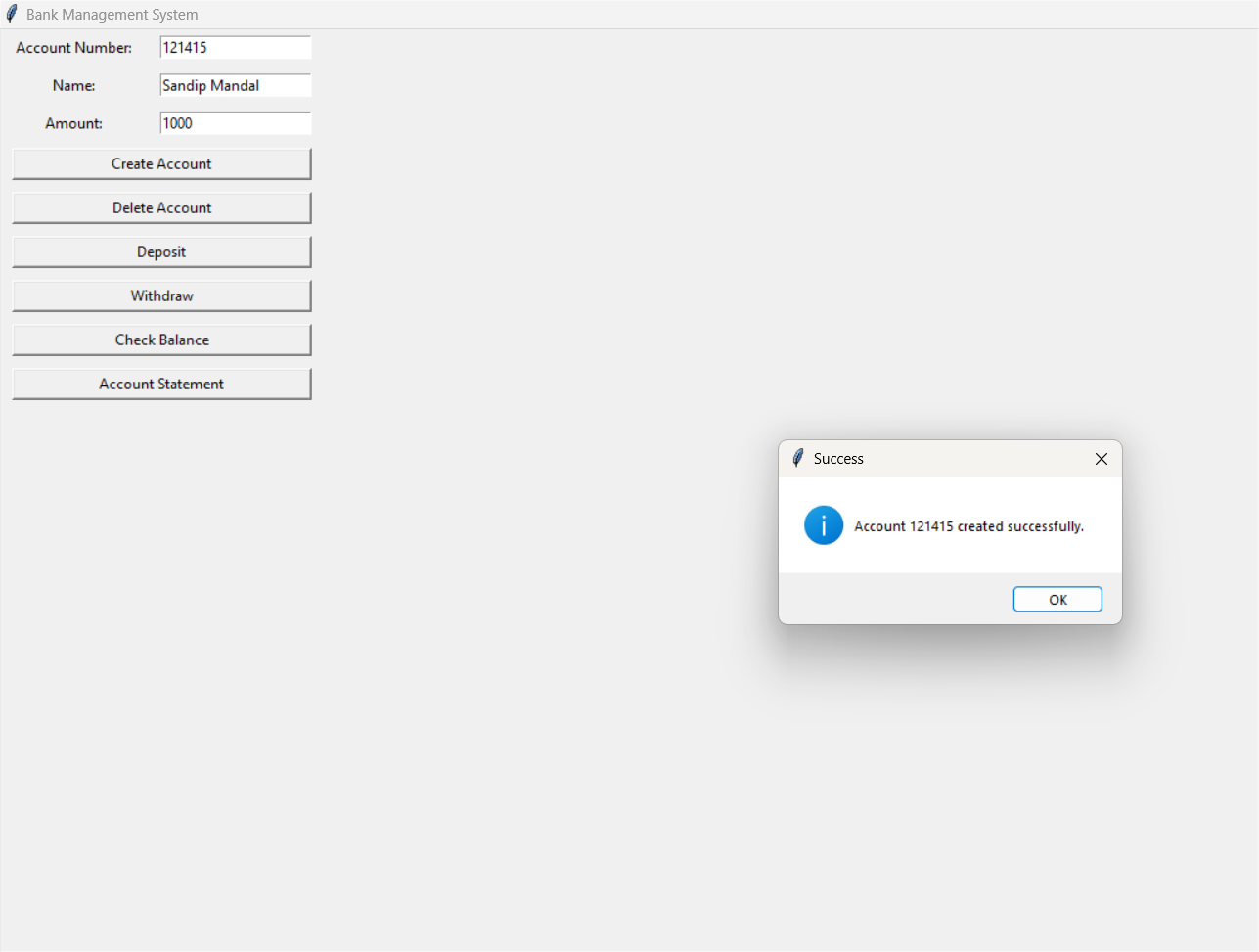
.

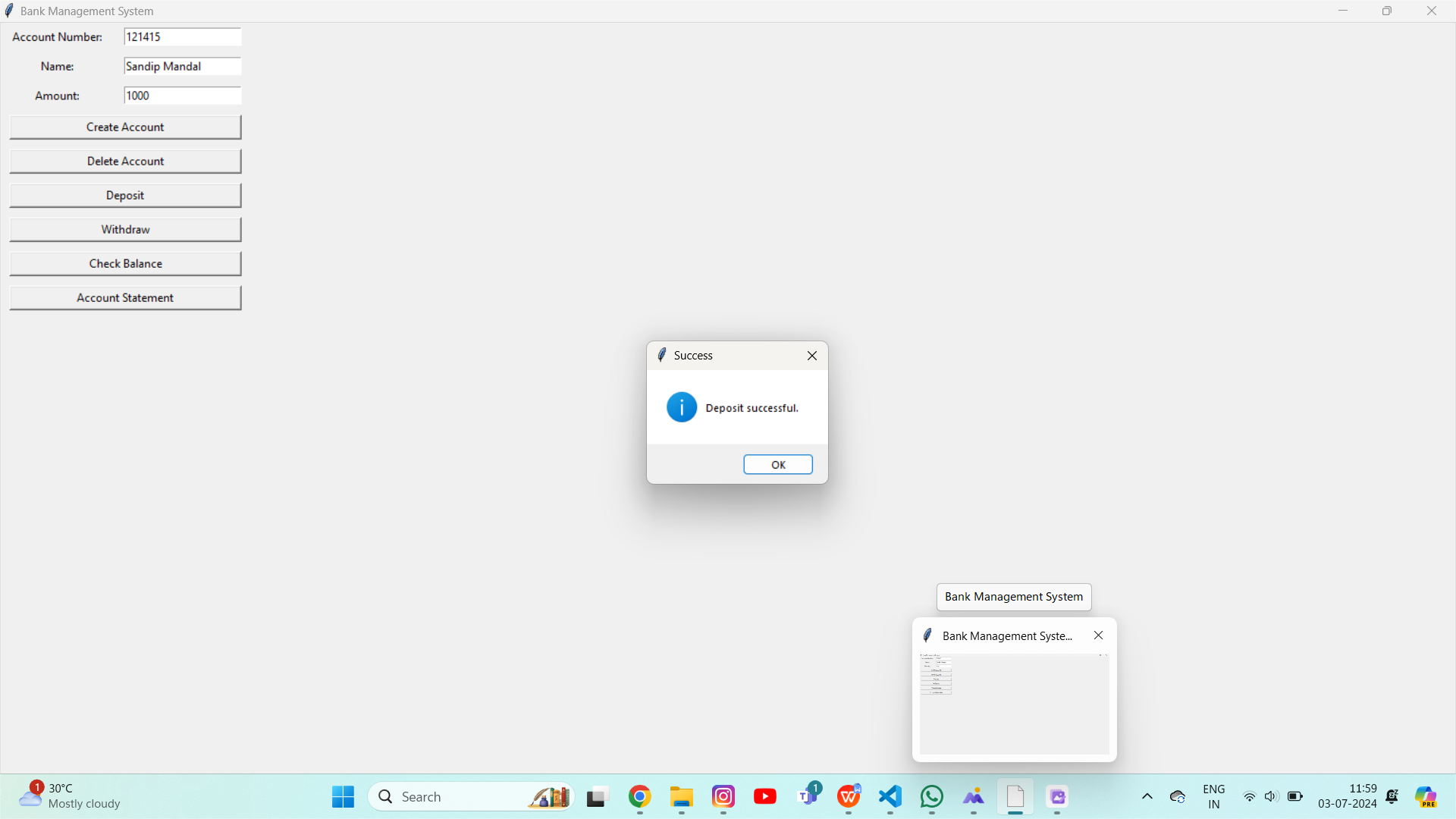
**11**

**CHAPTER-7**

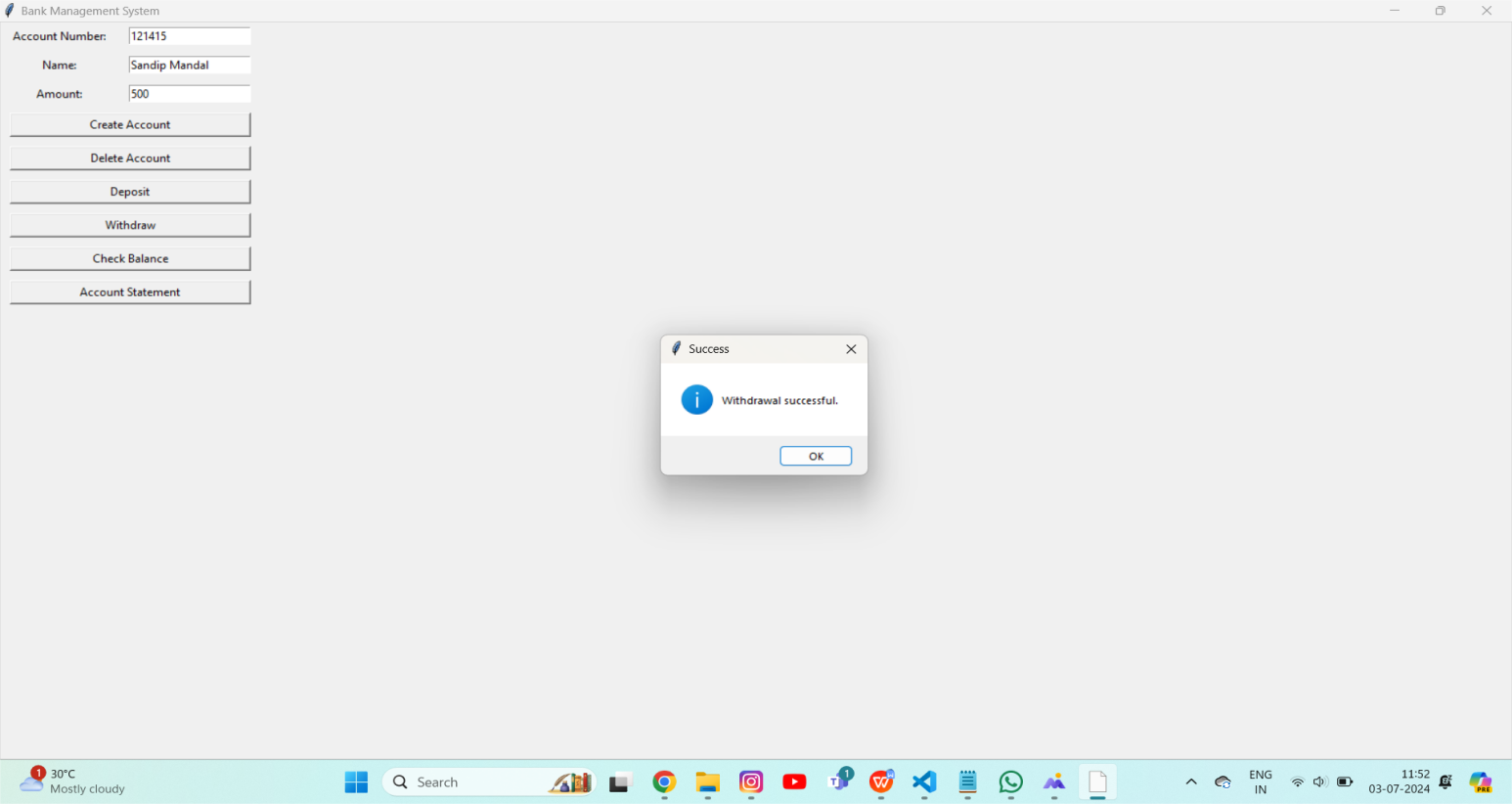
**RESULTS ANALYSIS**

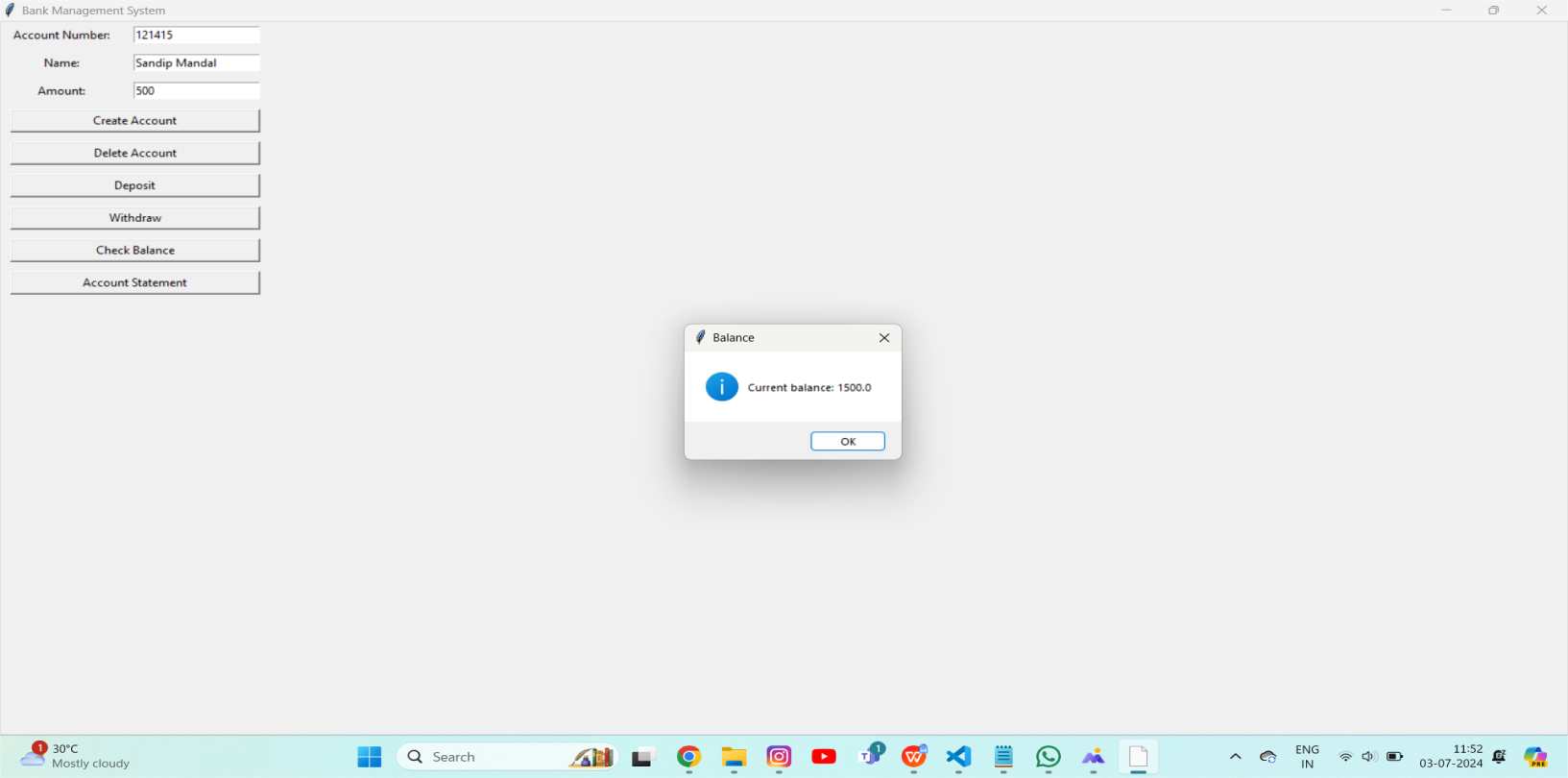


****

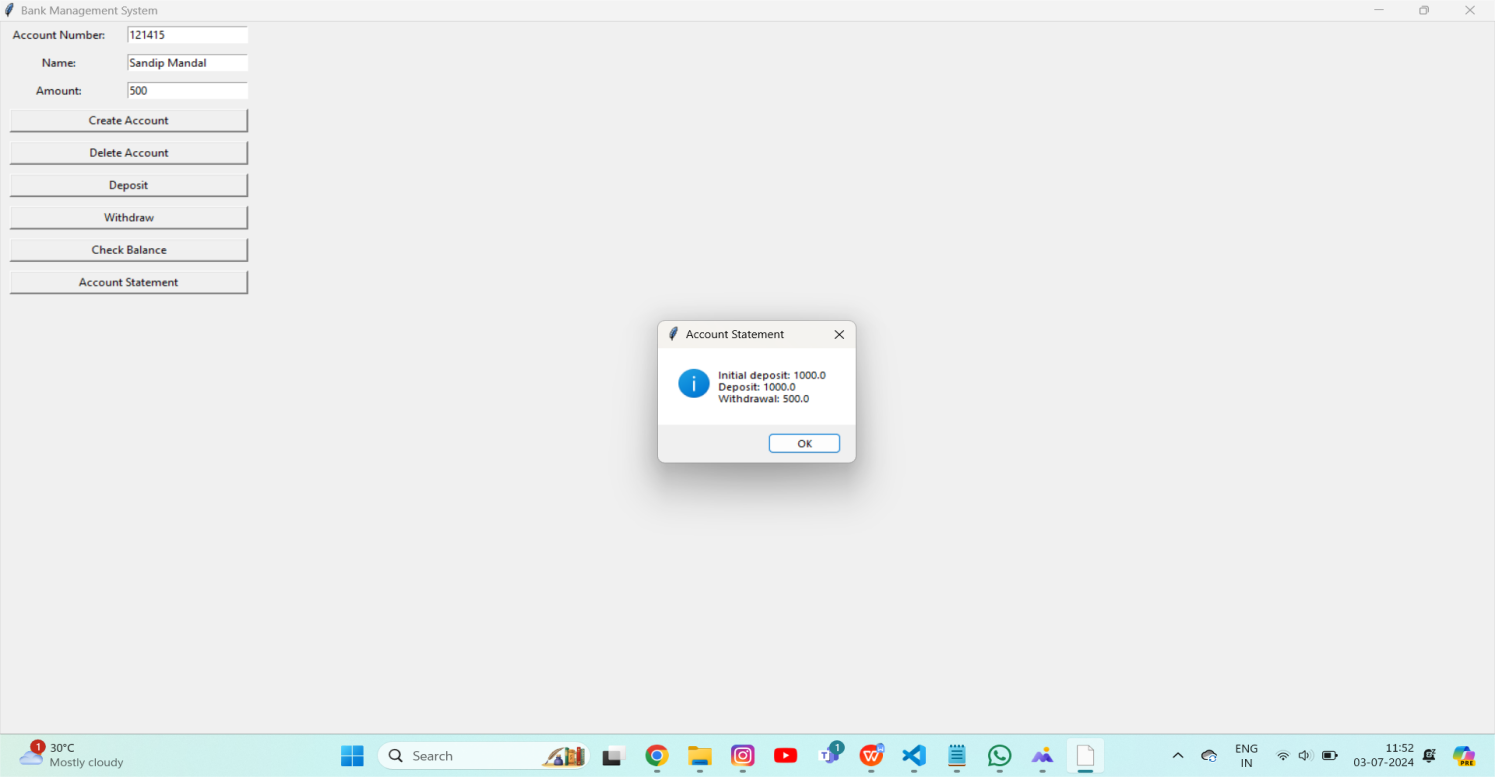
****

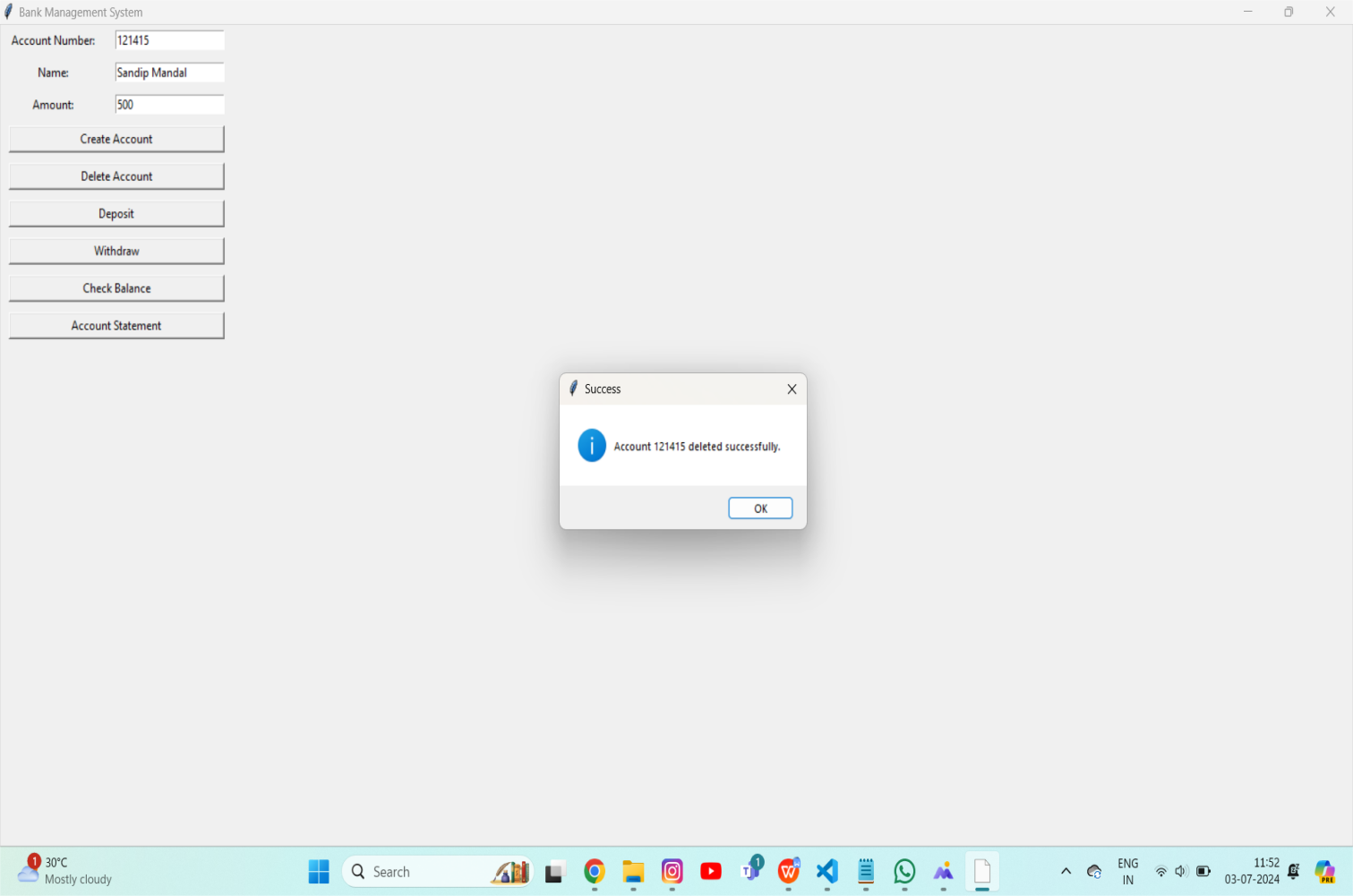
**12**

****

****

**13**

****

****

**14**

## **CHAPTER-8**

**CONCLUSION**

## Conclusion on Bank Management System

A Bank Management System is integral to the efficient and effective operation of modern financial institutions. It encompasses a wide range of functionalities, including account management, transaction processing, customer relationship management, loan and credit management, and compliance with regulatory requirements. Implementing a robust bank management system offers numerous benefits:

**Efficiency and Automation**

One of the primary advantages of a bank management system is the automation of routine tasks. This reduces the need for manual intervention, minimizes human error, and speeds up processes, allowing bank employees to focus on more complex and value-added activities.

### Improved Customer Service

With a centralized database and streamlined processes, banks can provide better customer service. Customers can access their accounts, perform transactions, and get assistance quickly and efficiently through various channels, including online banking, mobile apps, and ATMs.

### Enhanced Security and Compliance

### Bank management systems are designed with advanced security features to protect sensitive financial data and transactions. They also help banks comply with regulatory requirements by providing accurate and timely reports, tracking all transactions, and maintaining detailed records.

**15**