RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI ENGINEERING COLLEGE

CS23333 OOPS Using Java

Laboratory Record Note Book

Name : Harrish L S

Year / Branch / Section: 2/CSE

Register No. : 240701178

College Roll No. :2116240701178

Semester : 3

Academic Year:2025-26

AJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institution

BONAFIDE CERTIFICATE

Name: …… Harrish L S…………………………………………………………

Academic Year: ……2025-26……… Semester: ……3…Branch:…CSE…………

Register No. **2116240701178**

Certified that this is the bonafide record of work done by the above student in the.

OOPS USING JAVA.. Laboratory

during the academic year 2025- 2026

Signature of Faculty in-charge

Submitted for the Practical Examination held on………………19.09.25……………

Internal Examiner External Examiner

INDEX

EX.NO	DATE	NAME OF THE EXPERIMENT	GITHUB QR
1		I/O, Data Types, Operators	
2		Control Structures	
3		Arrays	
4		Strings	
5		Classes & Objects	
6		Inheritance	
7		Interface	
8		Exceptions	
9		Collections	
10		Collections	
11		Project	
12		Lambda	

<u>ABSTRACT</u>

This project implements a robust **ATM Management System** as a standalone **desktop application**. Developed using **Java Swing** for the front-end and **MySQL with JDBC** for the backend, the system provides a user-centric interface for accessing and managing banking services securely. Users are able to perform core ATM transactions such as **balance inquiry, cash withdrawal, deposit, and view mini-statements in real time**, with all data persistently stored and synchronized through a structured relational database. The solution incorporates a carefully designed schema, including user, account, card, and transaction tables, to ensure data consistency and security. Each transaction is processed atomically to prevent errors like double withdrawal or overdraft, and a clear audit trail is established for accountability. The application features an intuitive pastel-themed UI, responsive feedback, and animated transitions to enhance user experience.Overall, the project demonstrates the integration of secure transaction control, real-time data validation, and **responsive desktop UI design** for modern banking systems.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson DR. **M. THANGAM MEGANATHAN** for their timely support and encouragement.We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance. No words of gratitude will suffice for the unquestioning support extended to us by our Head of The Department **Dr. E.M. MALATHY** and our Deputy Head of The Department **Dr. J. MANORANJINI for** being ever supporting force during our project work We also extend our sincere and hearty thanks to our internal guide **MRS.B.DEEPA,** for his valuable guidance and motivation during the completion of this project. Our sincere thanks to our family members, friends and other staff members of computer science engineering.

HARRISH L S – 240701178

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	**3**
1 **1.1** **1.2** **1.3** **1.4**	**INTRODUCTION** INTRODUCTIOPN SCOPE OF THE WEEK PROBLEM STATEMENT AIM AND OBJECTIVES OF THE PROJECT	**7** **8**
2 **2.1** **2.2**	SYSTEM SPECIFICATIONS HARDWARE SPECIFICATIONS SOFTWARE SPECIFICATIONS	**9**
3	MODULE DESCRIPTION	**11**
4	IMPLEMENTATION-CODING	**13**
5	SCREENSHOTS	**22**
6	DIAGRAMS	**26**
7	CONCLUSION AND FUTURE ENHANCEMENT	**27**
8	REFERENCES	**28**

LIST OF FIGURES

FIGURE NO	TITLE	PG NO
5.1	LOGIN PAGE	22
5.2	SIGNUP PAGE	22
5.3	OTP VERIFICATION	23
5.4	AADHAAR VERIFICATION	23
5.5	DATABASE SCHEMA	23
5.6	DEPOSIT WINDOW	24
5.7	WITHDRAWAL WINDOW	24
5.8	BALANCE WINDOW	25
5.9	MINI STATEMENT	25

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION :

The Automated Teller Machine (ATM) Management System stands as a comprehensive and practical solution for simulating essential banking operations in a desktop environment. By integrating Java Swing for the graphical user interface and MySQL with JDBC for backend data management, the system delivers a seamless and responsive user experience. It facilitates core banking functionalities such as balance enquiry, cash withdrawal, deposit, and transaction history viewing, all within a secure and intuitive framework.A key highlight of the system is its emphasis on real-time transaction processing and robust authentication mechanisms. Multi-level user verification ensures that sensitive financial data remains protected from unauthorized access, while backend operations are designed to maintain high data integrity and prevent issues such as overdrafts or transaction failures. The modular architecture not only promotes scalability but also simplifies future enhancements, allowing developers to introduce new features without compromising existing functionality. The user interface is thoughtfully designed to guide users through banking tasks with minimal effort, making the system accessible to individuals with varying levels of technical proficiency. Meanwhile, the backend logic is optimized for speed and reliability, ensuring that transactions are processed efficiently and accurately. This project serves as a valuable educational tool for students and aspiring developers, offering hands-on experience in integrating front-end and back-end technologies to build a real-world financial application. It demonstrates key software engineering principles such as modular design, secure coding practices, and database transaction management. Overall, the ATM Management System not only fulfills its intended purpose but also lays the groundwork for more advanced financial software development, making it a noteworthy contribution to the field of educational and practical software engineering.

1.2 SCOPE OF THE WORK

The scope of this project is the development of a functional, database-driven **ATM Management System desktop application** in Java.

Key deliverables include:

User Authentication Module: Enables secure login and account verification.

Real-Time Account Management: Allows users to check balance, deposit, and withdraw funds with immediate updates from the database.

Transactional Integrity: Ensures data accuracy through atomic operations, preventing double withdrawals or overdrafts.

Data Persistence Layer: Employs MySQL (account, transaction, user tables) managed via JDBC for all banking data.

Exclusions:

The system does not handle physical cash dispensing or integration with real ATM hardware.Advanced fraud detection, biometric authentication, and multi-bank connectivity are not included. External APIs, real-world currency conversions, and emergency lockout features are not implemented

1.3 PROBLEM STATEMENT

Traditional ATM systems confront critical issues such as transaction errors, risk of duplicate withdrawals, and security vulnerabilities, especially when multiple users perform operations concurrently. Manual or outdated systems may fail to maintain real-time account synchronization, risking balance inconsistencies and transaction delays. This project addresses these issues by developing a secure desktop application that utilizes database transactions for atomic updates, robust authentication, and a user-friendly Java-based interface.

1.4 AIM AND OBJECTIVES OF THE PROJECT

Aim

To develop a fully functional, transactional ATM Management System desktop application using Java Swing and MySQL that ensures secure transaction processing and high data integrity.

Objectives

Develop GUI: Build a user-friendly Java Swing interface for smooth banking operations such as login, withdrawal, deposit, and balance inquiry.

Establish Persistence: Design and implement a complete MySQL schema (users, accounts, cards, transactions).

Ensure Integrity: Use JDBC transactions (commit/rollback) to guarantee atomicity and accuracy for deposits, withdrawals, and all banking operations.

Provide Transaction Reporting: Enable users to access recent activity and balance history, generating comprehensive digital mini statements.

CHAPTER 2
SYSTEM SPECIFICATIONS

21 HARDWARE SPECIFICATIONS :

Component	**Minimum Specification**	**Recommended Specification**
Processor	Intel Core i3 (or AMD equivalent)	Intel Core i5 / i7 (or AMD equivalent)
RAM (Memory)	4 GB	8 GB or higher
Storage	10 GB Free Disk Space (SSD recommended)	250 GB Free Disk Space (SSD)
Monitor	1024 x 768 resolution	Full HD (1920 x 1080) resolution
Input	Standard Keyboard and Mouse	Standard Keyboard and Mouse

2.2 SOFTWARE SPECIFICATIONS (Updated for MySQL)

Component	**Specification**	**Purpose**
Operating System	Windows 10/11, macOS, or Linux	Platform for development and execution.
Programming Language	**Java SE (Standard Edition)**	Primary language for application logic.
Development Kit	**JDK (Java Development Kit) 17 or higher**	Required for compiling and running Java code.
GUI Library	**Java Swing** (javax.swing)	Used for building the interactive desktop interface.
Database Server	**MySQL Server v8.0 or higher**	Used for data storage and management.
Database Connector	**MySQL Connector/J (JDBC Driver)**	Allows the Java application to communicate with MySQL.
Database Tool	**MySQL Workbench** or similar**	For designing the schema and running test queries.

CHAPTER 3

MODULE DESCRIPTION

The system is structured into three logical modules to separate the presentation, utility, and core business logic.

3.1 User Interface (UI) Module (Java Swing)

This module manages all user interaction and visual presentation.

Account Actions Frame: Main screen where users can select actions (Deposit, Withdraw, Balance) using buttons or menu options.
Input Screen: Java Swing forms for entering the transaction amount, account number, and PIN.
Feedback Handling: Shows status messages (success, failure) and updated account balance after every transaction.
.

3.2 Database Utility Module (JDBC)

Handles secure database connectivity and basic data retrieval from MySQL.

DatabaseConnection Class: Utility class to manage JDBC driver loading and connections to the MySQL server.

Account Data Retrieval: Methods to fetch, update, and validate account information for operations like displaying balance or verifying credentials.

3.3 Transaction Processing Module (Core Logic)

This is the business logic layer that maintains data integrity during all ATM operations (Deposit, Withdraw, Balance Inquiry).

Deposit Logic (processDeposit):

Executes a database transaction to:

- Add the deposit amount to the account's current balance in the accounts table.
- Insert a new transaction record in the transactions table for audit and history.
- Goal: Ensures the balance is only updated if the transaction record is created successfully, enforcing transactional safety and preventing errors if interruptions occur.

Withdrawal Logic (processWithdrawal):

Executes a database transaction to:

Check if the account has sufficient funds.

- Deduct the withdrawal amount from the account's balance in the accounts table.
- Insert a withdrawal record in the transactions table.
- Goal: Guarantees withdrawals only proceed if funds exist; all changes rollback if any check fails, preventing overdraft.

Balance Inquiry Logic (processBalanceInquiry):

Retrieves and displays the current account balance directly from the accounts table for the logged-in user. Gathers information about account transactions by querying the transactions, accounts, and related tables (using SQL JOINs if needed) to generate mini-statements or full histories.

CHAPTER 4

IMPLEMENTATION -CODING

DBCONNECTION CODE:

```
CREATE DATABASE atm;
USE atm;

CREATE TABLE users (
id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(100) NOT NULL,
age INT,
city VARCHAR(50),
pin VARCHAR(10) NOT NULL,
balance DECIMAL(15,2) DEFAULT 0.00
);

ALTER TABLE users ADD COLUMN mobile VARCHAR(15);

ALTER TABLE users ADD COLUMN aadhar VARCHAR(20);

ALTER TABLE users ADD COLUMN aadhar_last4 VARCHAR(10);
```

DASHBOARD FRAME CODE:

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.Cursor;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GradientPaint;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.GridBagLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.Timer;
```

```
public class DashboardFrame extends JFrame {
   int userId;
   String name;

   public DashboardFrame(int userId, String name) {
      this.userId = userId;
      this.name = name;
      int frameW = 900;
      int frameH = 700;
      this.setTitle("ATM Dashboard");
      this.setSize(frameW, frameH);
      this.setDefaultCloseOperation(3);
      JPanel gradientPanel = new JPanel() {
         protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            Graphics2D g2d = (Graphics2D)g;
            GradientPaint gp = new GradientPaint(0.0F, 0.0F, new Color(230, 242, 255), 0.0F, (float)this.getHeight(), new
Color(255, 245, 235));
            g2d.setPaint(gp);
            g2d.fillRect(0, 0, this.getWidth(), this.getHeight());
         }
      };
      gradientPanel.setLayout(new BorderLayout());
      JLabel welcome = new JLabel("Welcome to the ATM!", 0);
      welcome.setFont(new Font("Segoe UI", 1, 44));
      welcome.setForeground(new Color(101, 84, 140));
      welcome.setBorder(BorderFactory.createEmptyBorder(36, 10, 28, 10));
      gradientPanel.add(welcome, "North");
      JPanel btnPanel = new JPanel();
      btnPanel.setLayout(new BoxLayout(btnPanel, 1));
      btnPanel.setOpaque(false);
      int panelH = frameH / 2 + 60;
      btnPanel.setPreferredSize(new Dimension(frameW, panelH));
      String[] labels = new String[]{"\ud83d\udcb0  Balance", "\ud83d\udcb5  Deposit", "\ud83d\udcb8  Withdraw",
"\ud83d\udcc4  Mini Statement"};
      Color[] btnColors = new Color[]{new Color(204, 232, 255), new Color(200, 251, 204), new Color(255, 233, 216),
new Color(244, 232, 255)};
      JButton[] buttons = new JButton[labels.length];
      Dimension btnSize = new Dimension((int)((double)frameW * (double)0.75F), 78);

      for(int i = 0; i < labels.length; ++i) {
         buttons[i] = new JButton(labels[i]);
         this.styleFlatPastelButton(buttons[i], btnColors[i], btnColors[i].brighter());
         buttons[i].setFont(new Font("Segoe UI", 1, 30));
         buttons[i].setPreferredSize(btnSize);
         buttons[i].setMaximumSize(btnSize);
         buttons[i].setAlignmentX(0.5F);
         btnPanel.add(Box.createVerticalStrut(20));
         btnPanel.add(buttons[i]);
      }

      btnPanel.add(Box.createVerticalGlue());
      JPanel centerHolder = new JPanel(new GridBagLayout());
      centerHolder.setOpaque(false);
      centerHolder.add(btnPanel);
      gradientPanel.add(centerHolder, "Center");
      buttons[0].addActionListener((e) -> this.showBalance());
      buttons[1].addActionListener((e) -> this.deposit());
      buttons[2].addActionListener((e) -> this.withdraw());
      buttons[3].addActionListener((e) -> this.miniStatement());
      this.setContentPane(gradientPanel);
      this.setOpacity(0.0F);
      this.setLocationRelativeTo((Component)null);
      this.setVisible(true);
      (new Timer(16, new ActionListener() {
            float opacity = 0.0F;

            public void actionPerformed(ActionEvent e) {
                this.opacity += 0.04F;
                if (this.opacity >= 1.0F) {
                    this.opacity = 1.0F;
                    DashboardFrame.this.setOpacity(this.opacity);
                    ((Timer)e.getSource()).stop();
                } else {
                    DashboardFrame.this.setOpacity(this.opacity);
                }

            }
        })).start();
    }

    private void styleFlatPastelButton(final JButton btn, final Color base, final Color hover) {
        btn.setBackground(base);
        btn.setForeground(new Color(34, 41, 47));
        btn.setFocusPainted(false);
        btn.setContentAreaFilled(true);
        btn.setOpaque(true);
        btn.setCursor(Cursor.getPredefinedCursor(12));
        btn.setBorder(BorderFactory.createLineBorder(base.darker(), 3, true));
        btn.setBorder(BorderFactory.createCompoundBorder(btn.getBorder(), BorderFactory.createEmptyBorder(9, 28, 9,
28)));
        btn.addMouseListener(new MouseAdapter() {
            public void mouseEntered(MouseEvent evt) {
                btn.setBackground(hover);
            }

            public void mouseExited(MouseEvent evt) {
                btn.setBackground(base);
            }
        });
    }

    void showBalance() {
        try (
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/atm", "root", "nanandan1*");
            PreparedStatement stmt = conn.prepareStatement("SELECT balance FROM users WHERE id = ?");
        ) {
            stmt.setInt(1, this.userId);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                JOptionPane.showMessageDialog(this, "Balance: ₹" + rs.getDouble("balance"));
            }
        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "DB Error!");
        }

    }

    void deposit() {
        String amtStr = JOptionPane.showInputDialog(this, "Enter deposit amount:");
        if (amtStr != null) {
            try {
                double amt = Double.parseDouble(amtStr);

                try (
                    Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/atm", "root", "nanandan1*");
                    PreparedStatement stmt = conn.prepareStatement("UPDATE users SET balance = balance + ? WHERE id =
?");
                ) {
                    stmt.setDouble(1, amt);
                    stmt.setInt(2, this.userId);
                    stmt.executeUpdate();
                    PreparedStatement log = conn.prepareStatement("INSERT INTO transactions (user_id, type, amount)
VALUES (?, 'deposit', ?)");
                    log.setInt(1, this.userId);
                    log.setDouble(2, amt);
                    log.executeUpdate();
                    JOptionPane.showMessageDialog(this, "Deposited ₹" + amt);
                }
            } catch (Exception ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(this, "Error!");
            }

        }
    }

    void withdraw() {
        String amtStr = JOptionPane.showInputDialog(this, "Enter withdrawal amount:");
        if (amtStr != null) {
            try {
                double amt = Double.parseDouble(amtStr);

                try (
                    Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/atm", "root", "nanandan1*");
                    PreparedStatement balStmt = conn.prepareStatement("SELECT balance FROM users WHERE id = ?");
                ) {
                    balStmt.setInt(1, this.userId);
                    ResultSet rs = balStmt.executeQuery();
                    if (rs.next() && rs.getDouble("balance") >= amt) {
                        PreparedStatement stmt = conn.prepareStatement("UPDATE users SET balance = balance - ? WHERE id
= ?");
                        stmt.setDouble(1, amt);
                        stmt.setInt(2, this.userId);
                        stmt.executeUpdate();
                        PreparedStatement log = conn.prepareStatement("INSERT INTO transactions (user_id, type, amount)
VALUES (?, 'withdraw', ?)");
                        log.setInt(1, this.userId);
                        log.setDouble(2, amt);
                        log.executeUpdate();
                        JOptionPane.showMessageDialog(this, "Withdrawn ₹" + amt);
                    } else {
                        JOptionPane.showMessageDialog(this, "Insufficient balance!");
                    }
                }
            } catch (Exception ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(this, "Error!");
            }

        }
    }

    void miniStatement() {
        try (
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/atm", "root", "nanandan1*");
            PreparedStatement stmt = conn.prepareStatement("SELECT type, amount, timestamp FROM transactions
WHERE user_id = ? ORDER BY timestamp DESC LIMIT 5");
        ) {
            stmt.setInt(1, this.userId);
            ResultSet rs = stmt.executeQuery();
            StringBuilder sb = new StringBuilder("<html><b>Recent Transactions:</b><br>");

            boolean any;
            for(any = false; rs.next(); any = true) {
                String type = rs.getString("type");
                if (type.equals("deposit")) {
                    type = "deposited";
                }

                if (type.equals("withdraw")) {
                    type = "withdrawn";
                }

                sb.append("<span style='color:#414F6B'>").append(type).append("</span>
₹").append(rs.getDouble("amount")).append(" on <i>").append(rs.getTimestamp("timestamp")).append("</i><br>");
            }

            if (!any) {
                sb.append("<i>No recent transactions.</i>");
            }

            sb.append("</html>");
            JOptionPane.showMessageDialog(this, sb.toString());
        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "DB Error!");
        }

    }
}
```

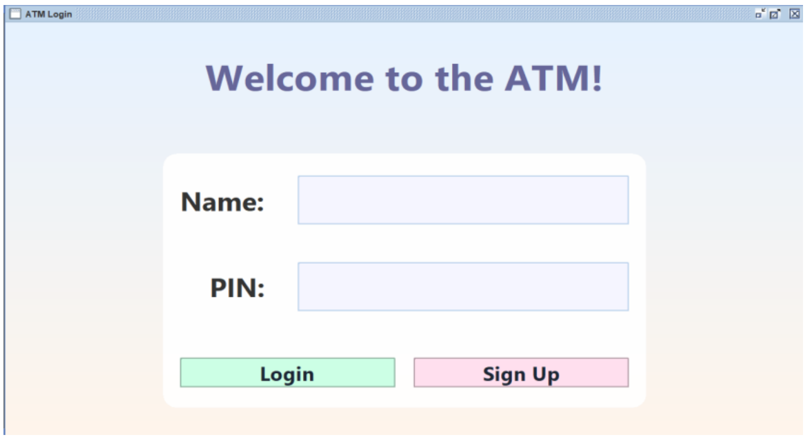
LOGIN FRAME CODE:

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GradientPaint;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;
import javax.swing.Timer;

public class LoginFrame extends JFrame {
    JTextField nameField;
    JPasswordField pinField;

    public LoginFrame() {
        this.setUndecorated(true);
        this.setTitle("ATM Login");
        this.setSize(1050, 600);
        this.setDefaultCloseOperation(3);
        JPanel gradientPanel = new JPanel() {
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                Graphics2D g2d = (Graphics2D)g;
                GradientPaint gp = new GradientPaint(0.0F, 0.0F, new Color(230, 242, 255), 0.0F, (float)this.getHeight(), new
Color(255, 245, 235));
                g2d.setPaint(gp);
                g2d.fillRect(0, 0, this.getWidth(), this.getHeight());
            }
        };
        gradientPanel.setLayout(new BorderLayout());
        JLabel titleLabel = new JLabel("Welcome to the ATM!", 0);
        titleLabel.setFont(new Font("Segoe UI", 1, 50));
        titleLabel.setForeground(new Color(102, 102, 153));
        titleLabel.setBorder(BorderFactory.createEmptyBorder(40, 12, 35, 12));
        gradientPanel.add(titleLabel, "North");
        JPanel card = new JPanel(new GridBagLayout()) {
            public void paintComponent(Graphics g) {
                super.paintComponent(g);
                Graphics2D g2 = (Graphics2D)g;
                g2.setColor(new Color(255, 255, 255, 233));
                g2.fillRoundRect(0, 0, this.getWidth(), this.getHeight(), 36, 36);
            }
        };


        card.setPreferredSize(new Dimension(630, 350));
        card.setOpaque(false);
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(26, 22, 26, 22);
        gbc.anchor = 13;
        JLabel nameLbl = new JLabel("Name:");
        nameLbl.setFont(new Font("Segoe UI", 1, 36));
        gbc.gridx = 0;
        gbc.gridy = 0;
        card.add(nameLbl, gbc);
        this.nameField = new JTextField(24);
        this.nameField.setFont(new Font("Segoe UI", 0, 32));
        this.styleField(this.nameField);
        gbc.gridx = 1;
        gbc.gridy = 0;
        gbc.fill = 2;
        gbc.weightx = (double)1.0F;
        card.add(this.nameField, gbc);
        JLabel pinLbl = new JLabel("PIN:");
        pinLbl.setFont(new Font("Segoe UI", 1, 36));
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.fill = 0;
        gbc.weightx = (double)0.0F;
        card.add(pinLbl, gbc);
        this.pinField = new JPasswordField(24);
        this.pinField.setFont(new Font("Segoe UI", 0, 32));
        this.styleField(this.pinField);
        gbc.gridx = 1;
        gbc.gridy = 1;
        gbc.fill = 2;
        gbc.weightx = (double)1.0F;
        card.add(this.pinField, gbc);
        JPanel btnPanel = new JPanel(new GridLayout(1, 2, 24, 0));
        btnPanel.setOpaque(false);
        JButton loginBtn = new JButton("Login");
        JButton signupBtn = new JButton("Sign Up");
        this.styleButton(loginBtn, new Color(204, 255, 229), new Color(218, 255, 238));
        this.styleButton(signupBtn, new Color(255, 223, 238), new Color(255, 239, 244));
        loginBtn.setFont(new Font("Segoe UI", 1, 27));
        signupBtn.setFont(new Font("Segoe UI", 1, 27));
        Dimension btnSize = new Dimension(240, 58);
        loginBtn.setPreferredSize(btnSize);
        signupBtn.setPreferredSize(btnSize);
        btnPanel.add(loginBtn);
        btnPanel.add(signupBtn);
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.gridwidth = 2;
        gbc.anchor = 10;
        gbc.insets = new Insets(38, 22, 22, 22);
        card.add(btnPanel, gbc);
        JPanel cardHolder = new JPanel(new GridBagLayout());
        cardHolder.setOpaque(false);
        cardHolder.add(card);
        gradientPanel.add(cardHolder, "Center");
        this.setContentPane(gradientPanel);
        loginBtn.addActionListener((e) -> this.login());
        signupBtn.addActionListener((e) -> this.signup());
        this.setLocationRelativeTo((Component)null);
        this.setOpacity(0.0F);
        this.setVisible(true);
        (new Timer(16, new ActionListener() {
            float opacity = 0.0F;

            public void actionPerformed(ActionEvent e) {
                this.opacity += 0.04F;
                if (this.opacity >= 1.0F) {
                    this.opacity = 1.0F;
                    LoginFrame.this.setOpacity(this.opacity);
                    ((Timer)e.getSource()).stop();
                } else {
                    LoginFrame.this.setOpacity(this.opacity);
                }

            }
        })).start();
    }

    private void styleField(JTextField field) {
        field.setBorder(BorderFactory.createCompoundBorder(BorderFactory.createLineBorder(new Color(187, 210, 236),
2, true), BorderFactory.createEmptyBorder(10, 16, 10, 16)));
        field.setBackground(new Color(245, 245, 255));
    }

    private void styleButton(final JButton btn, final Color base, final Color hover) {
        btn.setBackground(base);
        btn.setForeground(new Color(28, 38, 52));
        btn.setFocusPainted(false);
        btn.setBorder(BorderFactory.createLineBorder(base.darker(), 2, true));
        btn.addMouseListener(new MouseAdapter() {
            public void mouseEntered(MouseEvent evt) {
                btn.setBackground(hover);
            }

            public void mouseExited(MouseEvent evt) {
                btn.setBackground(base);
            }
        });
    }

    void login() {
        String name = this.nameField.getText().trim();
        String pin = new String(this.pinField.getPassword());

        try {
            try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/atm", "root", "nanandan1*"))
{
                try (PreparedStatement stmt = conn.prepareStatement("SELECT * FROM users WHERE name = ? AND pin =
?")) {
                    stmt.setString(1, name);
                    stmt.setString(2, pin);
                    ResultSet rs = stmt.executeQuery();
                    if (!rs.next()) {
                        JOptionPane.showMessageDialog(this, "Invalid credentials!");
                        return;
                    }

                    String otp = "123456";
                    String userOtp = JOptionPane.showInputDialog(this, "Enter 6-digit OTP sent to your mobile:");
                    if (otp.equals(userOtp)) {
                        String aadhaarDB = rs.getString("aadhaar_last4");
                        String userAadhaar = JOptionPane.showInputDialog(this, "Enter last 4 digits of your Aadhaar for
verification:");
                        if (!aadhaarDB.equals(userAadhaar)) {
                            JOptionPane.showMessageDialog(this, "Aadhaar verification failed!");
                            return;
                        }

                        int userId = rs.getInt("id");
                        this.setVisible(false);
                        (new DashboardFrame(userId, name)).setVisible(true);
                        return;
                    }

                    JOptionPane.showMessageDialog(this, "Incorrect OTP!");
                }
            }

        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "DB Error!");
        }
    }

    void signup() {
        JTextField nameF = new JTextField(18);
        JTextField ageF = new JTextField(18);
        JTextField cityF = new JTextField(18);
        JTextField mobileF = new JTextField(18);
        JTextField aadhaarF = new JTextField(18);
        JTextField pinF = new JTextField(18);
        Object[] fields = new Object[]{"Name:", nameF, "Age:", ageF, "City:", cityF, "Mobile Number:", mobileF,
"Aadhaar Last 4 Digits:", aadhaarF, "PIN:", pinF};
        int ok = JOptionPane.showConfirmDialog(this, fields, "Sign Up", 2, -1);
        if (ok == 0) {
            try (
                Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/atm", "root", "nanandan1*");
                PreparedStatement stmt = conn.prepareStatement("INSERT INTO users (name, age, city, mobile,
aadhaar_last4, pin, balance) VALUES (?, ?, ?, ?, ?, ?, 0)");
            ) {
                stmt.setString(1, nameF.getText().trim());
                stmt.setInt(2, Integer.parseInt(ageF.getText().trim()));
  stmt.setString(3, cityF.getText().trim());
                stmt.setString(4, mobileF.getText().trim());
                stmt.setString(5, aadhaarF.getText().trim());
                stmt.setString(6, pinF.getText().trim());
                stmt.executeUpdate();
                JOptionPane.showMessageDialog(this, "Account created! Please log in.");
            } catch (Exception ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(this, "DB Error!");
            }
        }

    }

    public static void main(String[] args) {
        JFrame.setDefaultLookAndFeelDecorated(true);
        SwingUtilities.invokeLater(() -> new LoginFrame());
```

CHAPTER 5

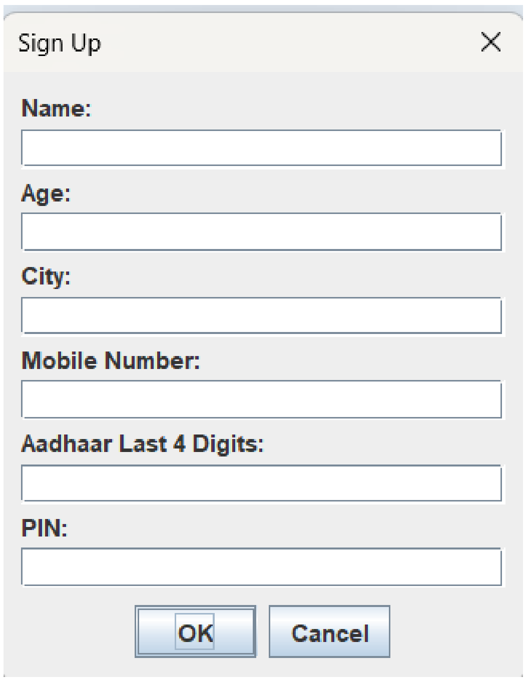
SCREEN SHOTS

Fig 5.1 LOGIN PAGE



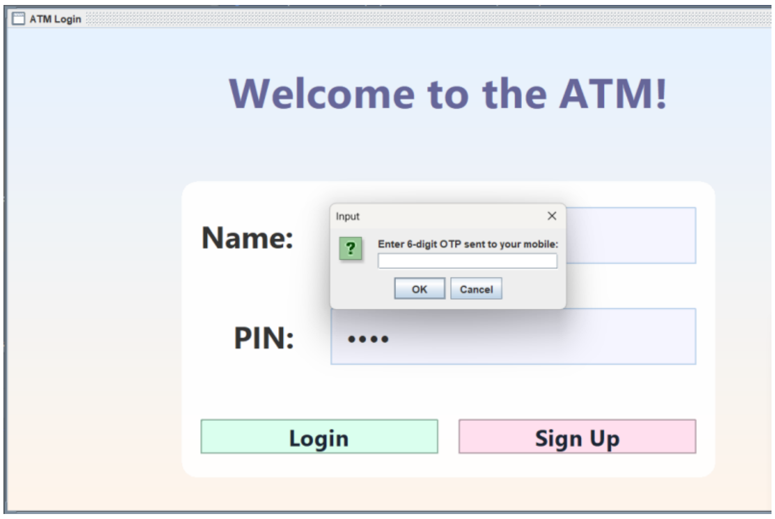
The ATM login interface features a welcome message, fields for name and PIN entry, and buttons for login or sign-up, offering a simple, user-friendly design for secure user authentication.

Fig 5.2 SIGN UP PAGE



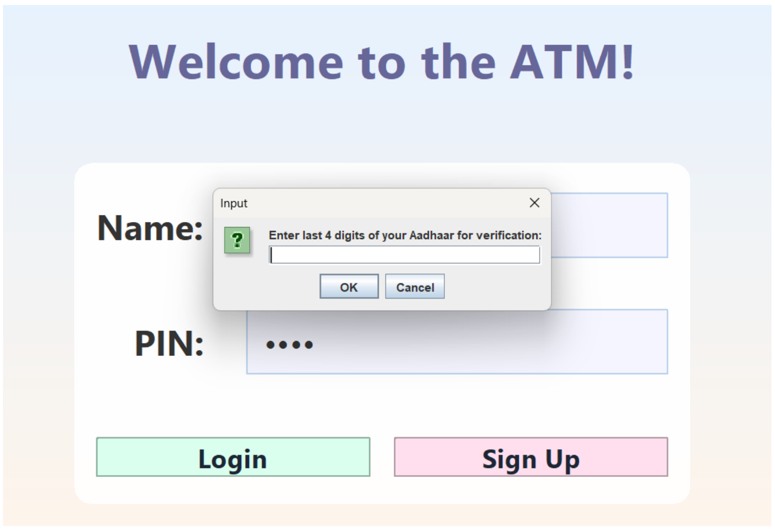
The Sign Up page interface allows users to register by entering personal details like name, age, city, mobile number, Aadhaar last 4 digits, and PIN. It includes "OK" and "Cancel" buttons for submission or exit

Fig 5.3 OTP VERIFICATION



The OTP Verification page enhances ATM security by prompting users to enter a 4-digit OTP sent to their mobile after inputting their name and PIN. It includes Login and Sign Up options.

Fig 5.4 AADHAAR VERIFICATION



The Aadhaar Verification page adds an extra layer of ATM security by prompting users to enter the last 6 digits of their Aadhaar number after providing their name and PIN.

Fig 5.5 DATABASE SCHEMA

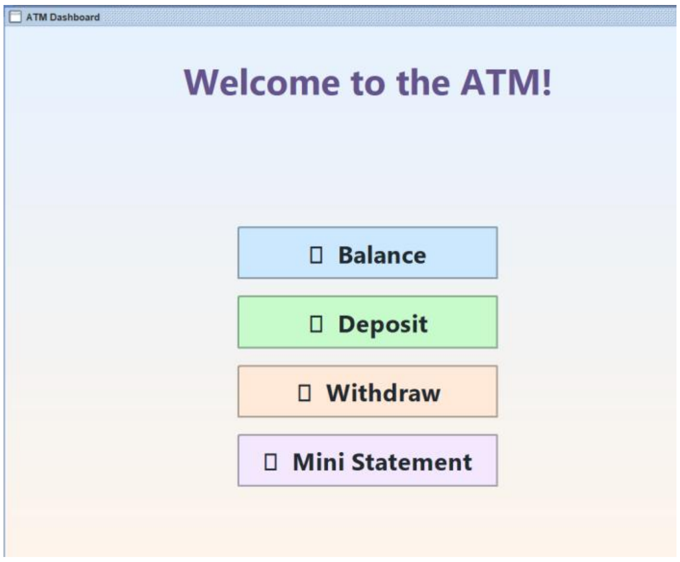


Fig 5.6 DEPOSIT WINDOW

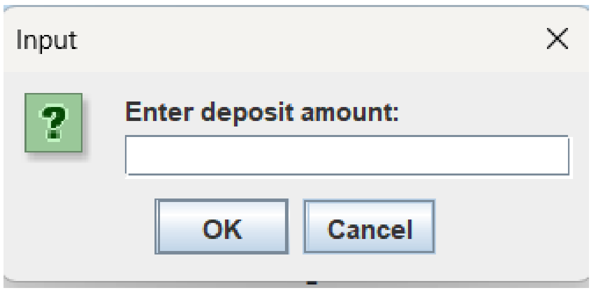


Fig 5.7 WITHDRAWAL WINDOW

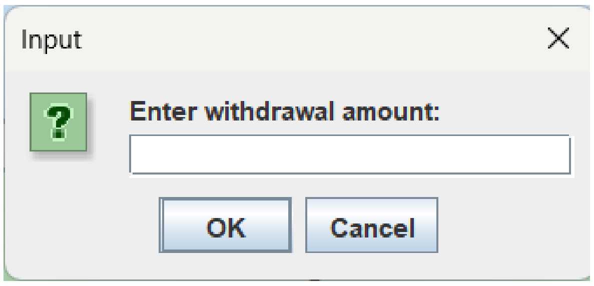
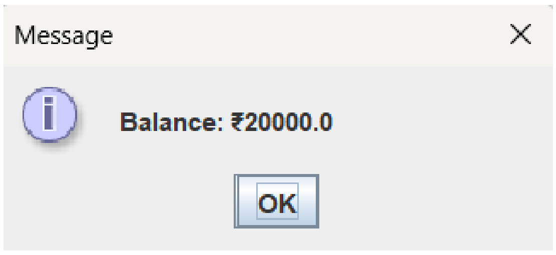
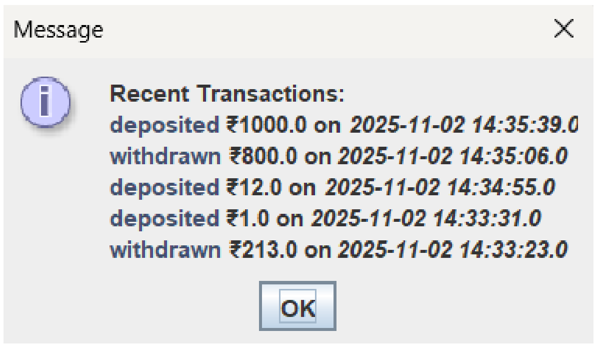


Fig 5.8 BALANCE WINDOW



Balance: ₹20000.0

Fig 5.9 MINI STATEMENT



```
Recent Transactions:
deposited ₹1000.0 on 2025-11-02 14:35:39.0
withdrawn ₹800.0 on 2025-11-02 14:35:06.0
deposited ₹12.0 on 2025-11-02 14:34:55.0
deposited ₹1.0 on 2025-11-02 14:33:31.0
withdrawn ₹213.0 on 2025-11-02 14:33:23.0
```

These ATM interface windows show key banking functions:

Withdrawal Window: Prompts user to enter withdrawal amount with OK/Cancel options.

Balance Window: Displays current account balance with an OK button.

Mini Statement: Lists recent transactions including deposits and withdrawals with timestamps.

They illustrate essential ATM operations for user interaction and financial tracking.

CHAPTER 6

ARCHITECTURE

6.1: ER DIAGRAM

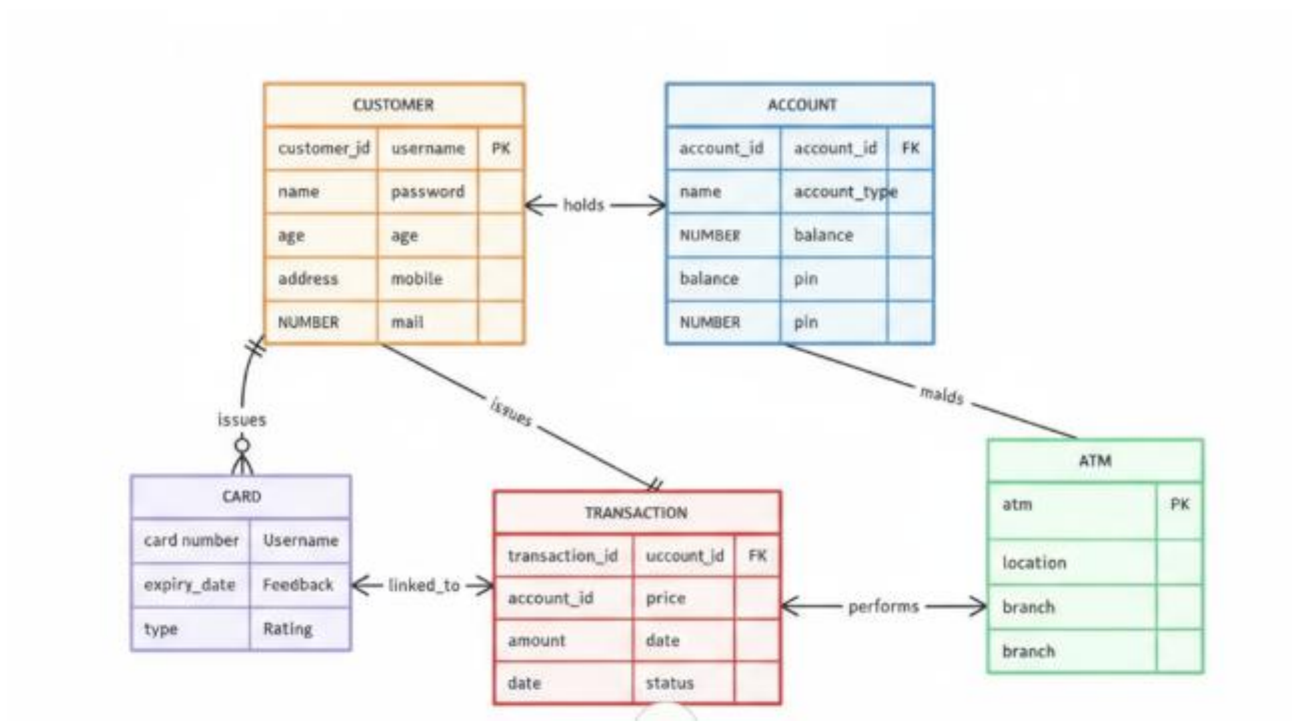


FIG NO.6.1: ENTITY RELATIONSHIP DIAGRAM

This ER diagram (fig 1) represents a banking system with five main entities: CUSTOMER, ACCOUNT, CARD, TRANSACTION, and ATM. Each entity includes relevant attributes, such as customer details, account information, card data, and transaction records. Relationships define how these entities interact—CUSTOMERs hold ACCOUNTs and issue CARDs, ACCOUNTs perform TRANSACTIONs, and TRANSACTIONs are linked to CARDs and executed at ATMs. The diagram also shows communication flows, like ACCOUNTs mailing CUSTOMERs and CARDs receiving feedback. This structure ensures efficient data management, supports secure financial operations, and provides a clear blueprint for designing and implementing a robust banking database system.

6.2: ARCHITECHTURE DIAGRAM

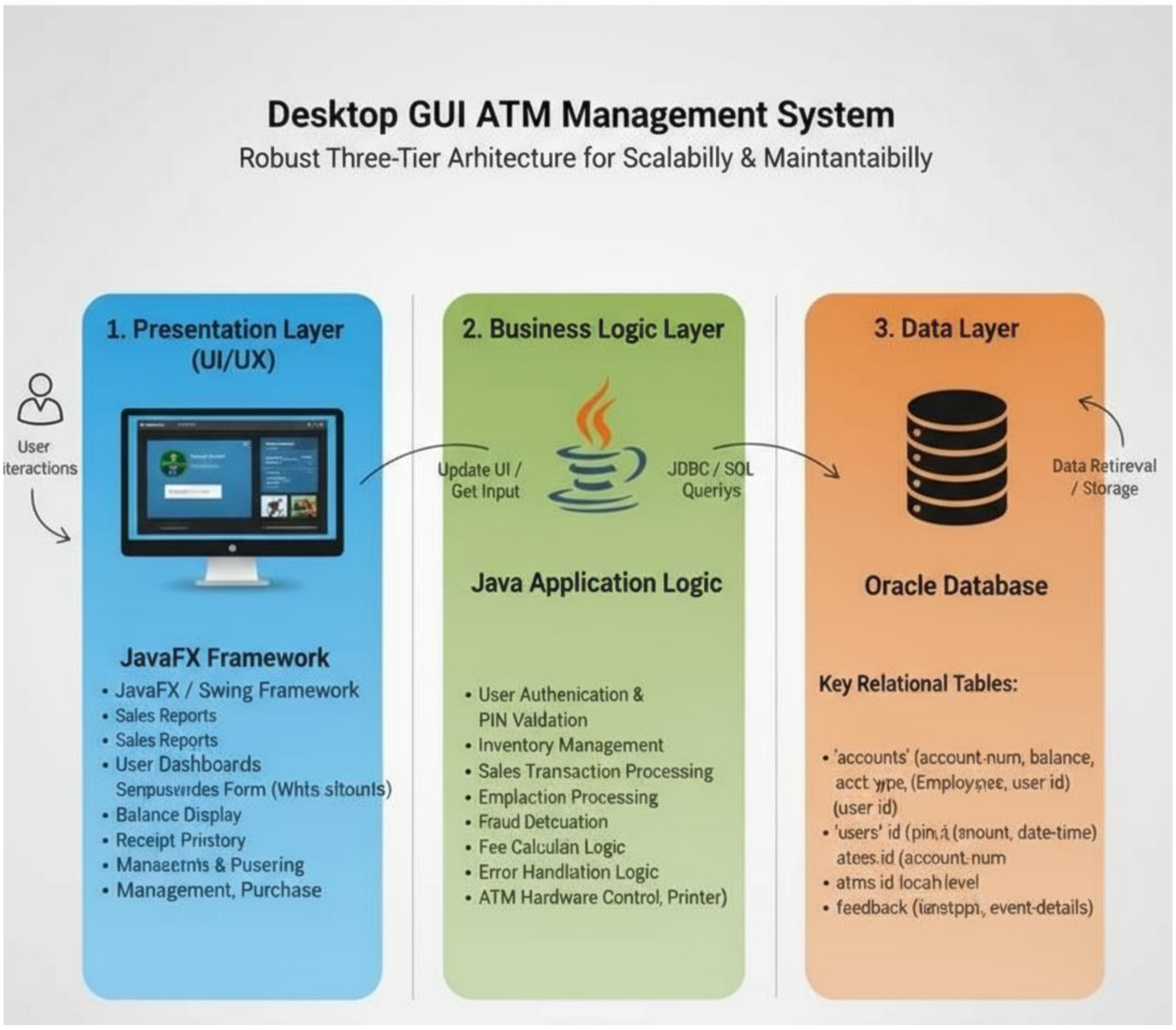

Desktop GUI ATM Management System
Robust Three-Tier Arhitecture for Scalabilly & Maintantaibilly
1. Presentation Layer (UI/UX)
User iteractions
JavaFX Framework
• JavaFX / Swing Framework
• Sales Reports
• Sales Reports
• User Dashboards
Senpuswrdes Form (Whts sitounts)
• Balance Display
• Receipt Pristory
• Manaeetmis & Pusering
• Management, Purchase
2. Business Logic Layer
Update UI / Get Input
JDBC / SQL Queriys
Java Application Logic
• User Authenication & PIN Valdation
• Inventory Management
• Sales Transaction Processing
• Emplaction Processing
• Fraud Detcuation
• Fee Calculan Logic
• Error Handlation Logic
• ATM Hardware Control, Printer)
3. Data Layer
Data Retireval / Storage
Oracle Database
Key Relational Tables:
• 'accounts' (account-num, balance, acct ype, (Employsee, user id) (user id)
• 'users' id (pin, (anount, date-time) atees id (account-num
• atms id locah level
• feedback (ianstppi, event-details)

FIG NO. 6.2: ATM ARCHITECHTURE DIAGRAM

This diagram (fig 2) explains a Desktop GUI ATM Management System using a three tier architecture. The Presentation Layer (UI/UX) handles user interactions via JavaFX or Swing, displaying dashboards and processing user commands. The Business Logic Layer contains Java application logic, including authentication, transactions, and calculations, connecting the UI and database using JDBC/SQL. The Data Layer consists of an Oracle Database with relational tables for accounts, users, transactions, and feedback, managing all data storage and retrieval. This layered structure ensures scalability, easier maintenance, and a clear separation between user interface, processing, and data handling.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The successful development and deployment of the ATM Management System marks a significant achievement in delivering a secure, efficient, and user-centric banking solution. By leveraging Java Swing for the front-end interface and MySQL for robust back-end data management, the system ensures seamless integration between user actions and database operations. The core strength of the application lies in its implementation of reliable transactional mechanisms, which uphold data integrity and consistency across all financial operations—be it deposits, withdrawals, or balance inquiries. In essence, the ATM Management System not only meets its initial objectives but also sets a strong foundation for continued innovation in digital banking solutions. It exemplifies how thoughtful design, rigorous backend logic, and user-friendly interfaces can converge to create a powerful tool that streamlines essential financial services. This project stands as a testament to effective software engineering practices and offers valuable insights for future developments in the fintech domain.

FUTURE ENHANCEMENT

To make the system a commercial-grade application, focus on these expansions:

Payment Gateway Integration: Add secure connectivity to online payment platforms to enable direct transfers and utility payments in addition to basic ATM operations.

Admin Module: Create a password-protected management interface for staff to perform CRUD operations on user accounts, generate audits, and manage account limits.

User Profiles: Implement authentication for login/registration, allowing users to track and manage their account history and personal details securely.

Web/Mobile Expansion: Refactor the core logic to support web and mobile deployments for access flexibility and wider reach..

REFERENCES

1. https://docs.oracle.com/javase/8/docs/api/javax/swing/packagesummary.html
2. https://docs.oracle.com/javase/tutorial/jdbc/index.html
3. https://dev.mysql.com/doc/refman/8.0/en/
4. https://dev.mysql.com/doc/connector-j/8.0/en/
5. https://www.oracle.com/java/technologies/data-access-object.html