# A REPORT ON

INDUSTRIAL TRAINING ON “NFC based Employee Attendance management System”

*Taken at*

*“Groot Software”*

**In partial fulfillment of the requirement for the award of degree of Bachelor of Technology**

**In**

**Department of Information Technology**

**(Session 2024-25)**

**Duration: From 25th June, 2025 to 9th August, 2025**

**Submitted to:**

Mrs.Rama Bhardwaj

Department of IT**Submitted by:**

Vansh Sharma

V Semester 23EJCIT174

**Department of Department of Information Technology Jaipur Engineering College & Research Centre, Jaipur**

**Rajasthan Technical, University(RTU)**

1.

# CERTIFICATE

**A certificate of completion

AI-generated content may be incorrect.**

This is to certify that the report of the training submitted is the outcome of the practical training done at **“Groot Software”** in **“Java Core and Advanced”**, carried out by **Vansh Sharma** bearing **RTU Roll No.: 23EJCIT174** under the guidance and supervision of **Mrs. Rama Bhardwaj** for the award of Degree of **Bachelor of Technology (B.Tech.)** in **Department of Information Technology** from **Jaipur Engineering College & Research Centre, Jaipur (Raj.), India** affiliated to **Rajasthan Technical University, Kota** during the academic year **2025-26**.

To the best of my knowledge, the report:

i) Embodies the work of the candidate.  
 ii) Has duly been completed.  
 iii) Fulfills the requirement of the ordinance relating to the Bachelor of Technology degree of the

Rajasthan Technical University, Kota.  
 iv) Is up to the desired standard for the purpose for which it is submitted.

|  |  |  |
| --- | --- | --- |
| **Dr. Smita Agarwal** | **Mr. Piyush Gautam** | **Mrs. Rama Bhardwaj** |
| Head of Department | Assistant Prof. & TPO | Assistant Prof. |
| IT | IT | IT |
| JECRC, Jaipur | JECRC, Jaipur | JECRC, Jaipur |

# DECLARATION

I hereby declare that the report entitled **“ JAVA Development Industrial Training”** has been carried out and submitted by the undersigned to **Jaipur Engineering College & Research Centre, Jaipur (Rajasthan)** as an original work, conducted under the guidance and supervision of **Mrs. Rama Bhardwaj** of JECRC Jaipur.

The empirical findings in this report are based on the data and practical work, which has been collected and implemented by me during my training at **Groot Software**. I have not reproduced any material from any report of the University, neither of this year nor of any previous year.

I understand that any such reproduction of original work by another is liable to be punished in a way the University authorities deem fit.

**Place: Jaipur Vansh Sharma**

**Date:23/08/2025 23EJCIT174**

# PREFACE

Bachelor of Technology in Information Technology is a course of **Rajasthan Technical University** (Approved by AICTE) having a duration of four years. As a prerequisite of the syllabus, every student of this course has to undergo professional training from an industry related to the stream for **60 days** in order to successfully complete his/her studies. It is also required to submit a project report at the completion of the training.

The main objective of this training is to create awareness regarding the application of theoretical knowledge in the practical world of Information Technology and to provide students with real-world exposure.

I, therefore, submit this project report of my training, which was undertaken at Groot Software. It was a great experience to undergo training in a reputed organization in the field of **Java Development**.

I feel immense pleasure in presenting this project report. The report has been prepared on the basis of the study and development carried out during the training regarding the **NFC Based Employee Attendance System**, a dynamic and interactive Java and JDBC based application.

. “Any serious and lasting achievement or success can never be achieved without the help, guidance, and cooperation of the many people involved in the work.”

It is my pleasant duty to express my profound gratitude and sincere thanks to **Mr. Arpit Agrawal** and the team a Groot Software, who gave me the opportunity to undergo professional training in **Java Development.**.

I would like to express my deep gratitude to **Dr. Smita Agarwal**, Head of Department (Information Technology), **Jaipur Engineering College & Research Centre, Jaipur (Rajasthan)**, without whose permission this training would not have been possible. I am also thankful to **Mrs. Rama Bhardwaj**, Assistant Professor, **Mr. Piyush Gautam, TPO**, Department of IT, JECRC Jaipur, who recommended me for this training and provided valuable guidance throughout.

Last but not the least, I am grateful to all faculty members, mentors, and colleagues who directly or indirectly supported me in completing this project, encouraging me whenever I needed their help despite their busy schedules.

**Vansh Sharma (23EJCIT174)**

# COMPANY PROFILE

**Groot Software** is a leading technology training and development organization based in India, specializing in providing hands-on training and practical learning experiences for students and professionals. The company focuses on equipping learners with industry-relevant skills in web development, software engineering, data science, and other emerging technologies, enabling them to bridge the gap between academic learning and real-world professional experience.

## Mission

Groot Software’s mission is to empower students and aspiring professionals by providing comprehensive training programs that enhance their technical skills and career readiness. By delivering practical, project-based learning experiences, Groot Software ensures that learners are well-prepared to meet industry demands and excel in their chosen fields.

## Key Services and Offerings

1. **Hands-on Training Programs:**
   * Groot Software offers specialized training programs in areas such as **MERN Stack development, Full Stack Web Development, Data Science, AI, and Machine Learning**.
   * Courses are designed to provide **practical experience** through real-world projects, enabling students to build robust portfolios.
   * Upon completion, participants receive **certifications** that enhance employability and career prospects.
2. **Project-based Learning:**
   * Learners work on industry-relevant projects, such as dynamic web applications, dashboards, and full-stack platforms, gaining practical exposure to current technologies.
   * The training emphasizes **front-end and back-end integration, database management, and deployment strategies**.
3. **Career Support and Guidance:**
   * Groot Software provides mentorship, resume-building assistance, and career guidance to help students transition from training to professional roles.
   * Students are supported in securing internships and job opportunities in relevant industries.

## Key Features of Groot Software

* **Industry-oriented Curriculum:** Training programs are designed to meet current industry standards and demands.
* **Hands-on Projects:** Students gain practical skills through live projects and assignments.
* **Flexible Learning:** Programs accommodate both in-person and remote learners.
* **Career Enhancement:** Certification and guidance improve students’ employability and readiness for professional roles.

**CHAPTER INDEX**

|  |  |  |
| --- | --- | --- |
| S.NO. | TITLE | PAGE NO. |
|  | CERTIFICATE | i |
|  | DECLARATION | ii |
|  | PREFACE | iii |
|  | ACKNOWLEDGEMENT | iv |
|  | COMPANY PROFILE | v |
| 1 | INTRODUCTION | 1 |
|  | 1.1 PURPOSE | 1 |
|  | 1.2 SCOPE | 2 |
|  | 1.3 FEATURES | 3 |
|  | 1.4 FIGURE 1 | 4 |
|  | 1.5 FIGURE 2 | 5 |
|  | 1.6 FIGURE 3-5 | 5-13 |
| 2 | FRONTENED DEVELOPMENT | 8 |
|  | 2.1 PURPOSE | 9 |
|  | 2.2 KEY FEATURES | 10 |
| 3 | BACKENED DEVELOPMENT | 11 |
|  | 3.1 KEY COMPONENT | 13 |
|  | 3.2 KEY FEATURES | 14 |
| 4 | AI IN WEB DEVELOPMENT | 15 |
| 5 | BOOTSTRAP | 18 |
|  | 5.1 PURPOSE AND BENIFITS | 20 |
| 6 | DBMS | 21 |
|  | 6.1 TYPE OF DBMS | 22 |
| 7 | CONCLUSION | 23 |
| 8 | FUTURE SCOPE | 24 |
| 9 | REFERENCES | 25 |

**CHAPTER 1 INDRODUCTION**

**Introduction to Software Development with Java**

Software Development is the process of designing, creating, testing, and maintaining software applications. For this project, the focus is on Desktop Application Development, which involves building robust, standalone applications that run directly on a computer's operating system. This encompasses everything from designing the user interface and writing the core application logic to managing databases and integrating with external hardware. A desktop application can be a simple utility tool or a complex, enterprise-level system like an NFC-based Employee Attendance Management System.

**Purpose of Desktop Application Development**

The primary purpose of desktop application development is to build powerful, reliable applications tailored for specific tasks within an organization or for individual users. Unlike web applications, they run locally on a machine, offering distinct advantages. The main goals include:

* Creating Functional Applications: Building applications that are stable, efficient, and perform their intended tasks flawlessly.
* Providing a Good User Experience (UX): Designing an intuitive and easy-to-navigate Graphical User Interface (GUI) so users can interact with the application seamlessly.
* Supporting Business Needs: Creating specialized software to solve specific business problems, such as automating employee attendance tracking to improve accuracy and save time.
* Direct Hardware Interaction: Enabling the software to communicate directly with hardware peripherals, a key requirement for the NFC reader in this project.

**Scope of Java Application Development**

The scope of developing an application like this is comprehensive, involving several key areas:

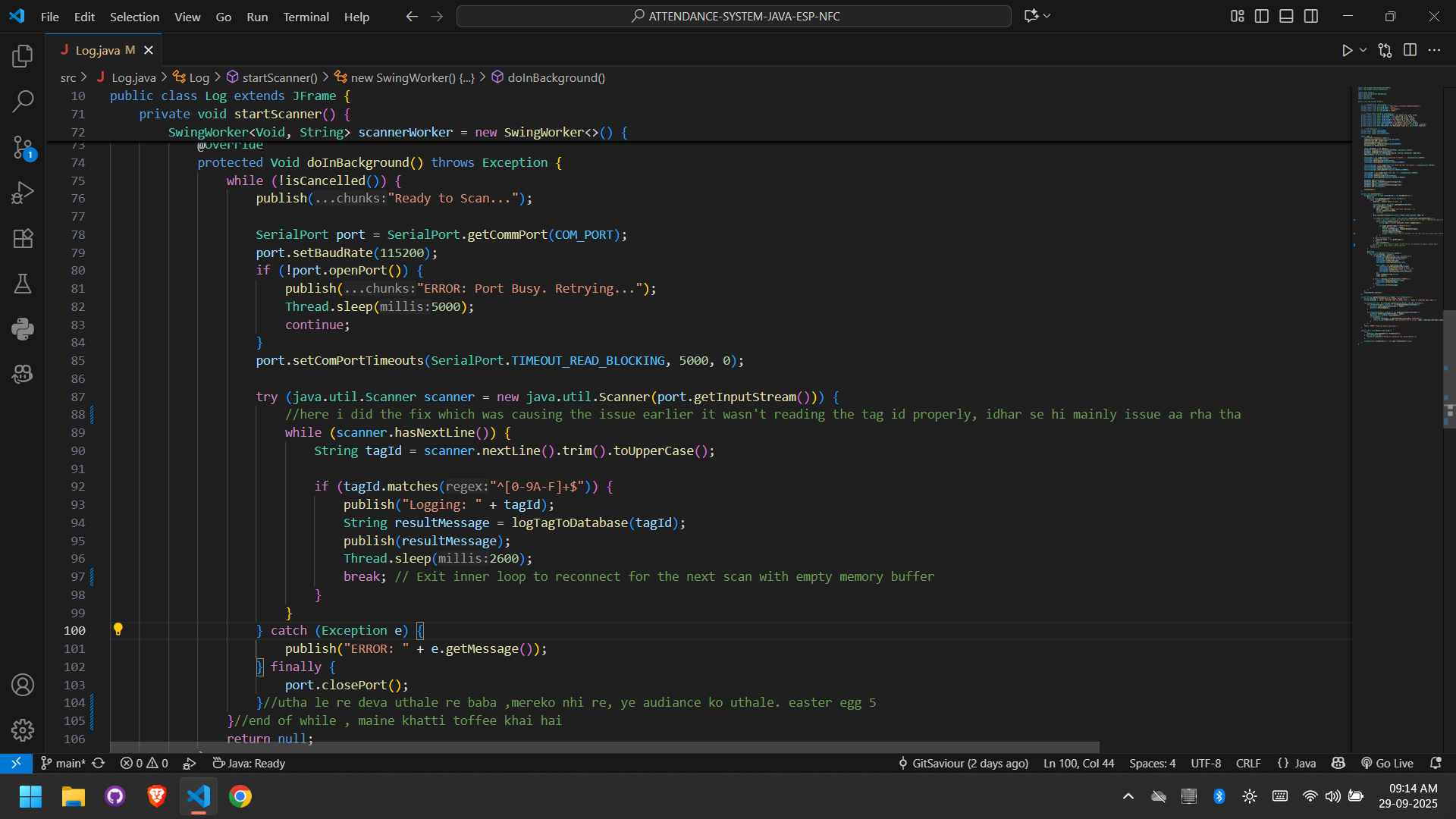
1. Frontend Development (GUI):
   * This focuses on what the user sees and interacts with—the application's windows, buttons, forms, and tables. It involves designing a user-friendly and visually clear layout.
   * Technologies Used: In this project, the GUI was built using Java Swing, a powerful toolkit for creating rich graphical user interfaces.
2. Backend Development (Core Logic):
   * This is the "engine" of the application. It focuses on the server-side logic that processes data, enforces business rules, and performs core computations. It's the part of the system the user doesn't see but that makes everything work.
   * Technologies Used: Core and Advanced Java concepts, including Object-Oriented Programming (OOP), data structures, and multithreading, were used to build the application's logic.
3. Database Management:
   * This component deals with storing, retrieving, and managing the application's data, such as employee details and attendance records. A persistent database ensures that information is saved securely even after the application is closed.
   * Technologies Used: JDBC (Java Database Connectivity) was used to connect the Java application to a relational database (like MySQL or Oracle) and execute SQL queries.
4. Hardware Integration:
   * This specialized area involves writing code that allows the software to communicate with and control external hardware devices. For this project, it meant interfacing with an NFC (Near Field Communication) reader to capture employee card data.
   * Technologies Used: Java libraries and APIs (like javax.smartcardio or third-party libraries) were used to establish communication between the application and the NFC hardware.

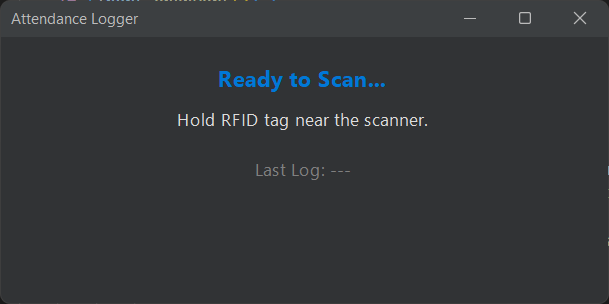
**Features of a Java Desktop Application**

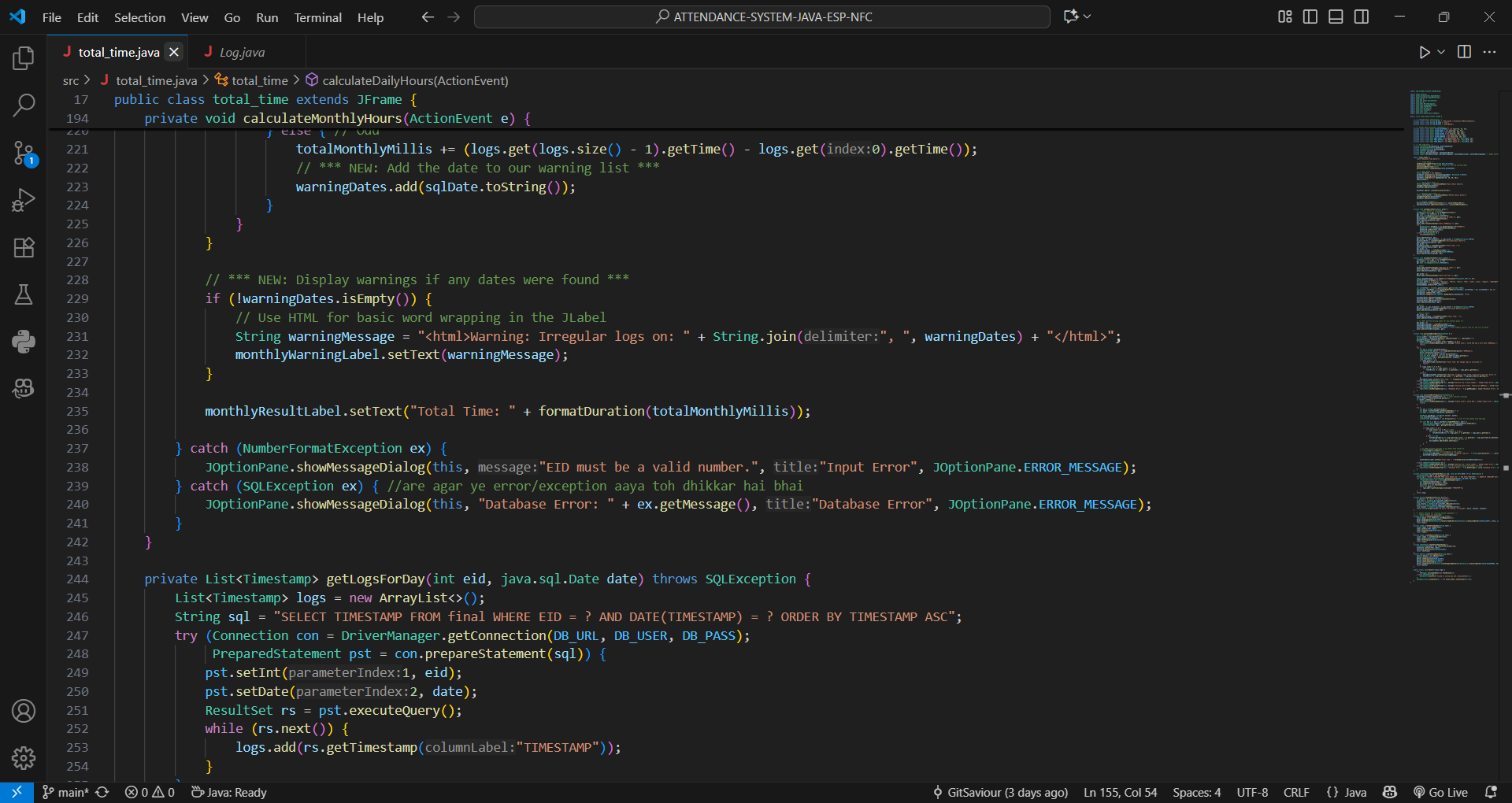
**Key features demonstrated in this project include:**

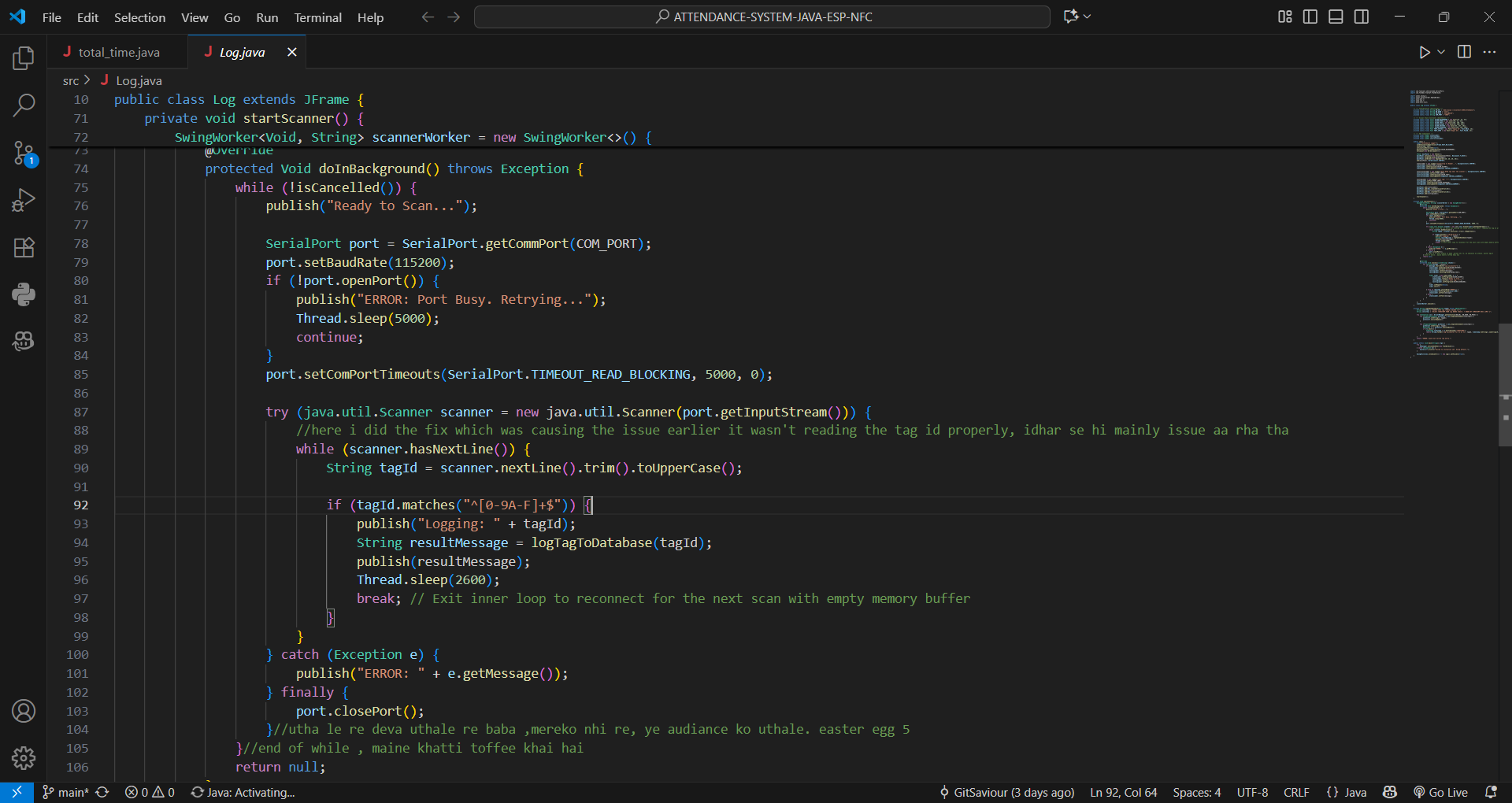
1. Platform Independence:
   * Built with Java, the application can run on any operating system (Windows, macOS, Linux) that has the Java Virtual Machine (JVM) installed. This is the "Write Once, Run Anywhere" principle.
2. Rich User Interface (Interactivity):
   * Using Java Swing, the application provides a responsive and interactive GUI, allowing administrators to manage employees, view attendance logs, and generate reports through buttons, forms, and data tables.
3. Security:
   * The application is designed to be secure, managing sensitive employee data. This is handled through proper data validation and controlled access to the database via JDBC, preventing unauthorized data manipulation.
4. Performance and Efficiency:
   * As a compiled language running directly on the OS, Java offers high performance, essential for quick data processing and real-time responses from the NFC reader. Optimized database queries using JDBC further enhance speed.
5. Data Persistence:
   * All employee and attendance data is stored permanently in a database. JDBC ensures that data is reliably written to and read from the database, maintaining data integrity.
6. Robustness:
   * Java's strong memory management and exception-handling features make the application robust and less prone to crashes, which is critical for a system used for daily operations.

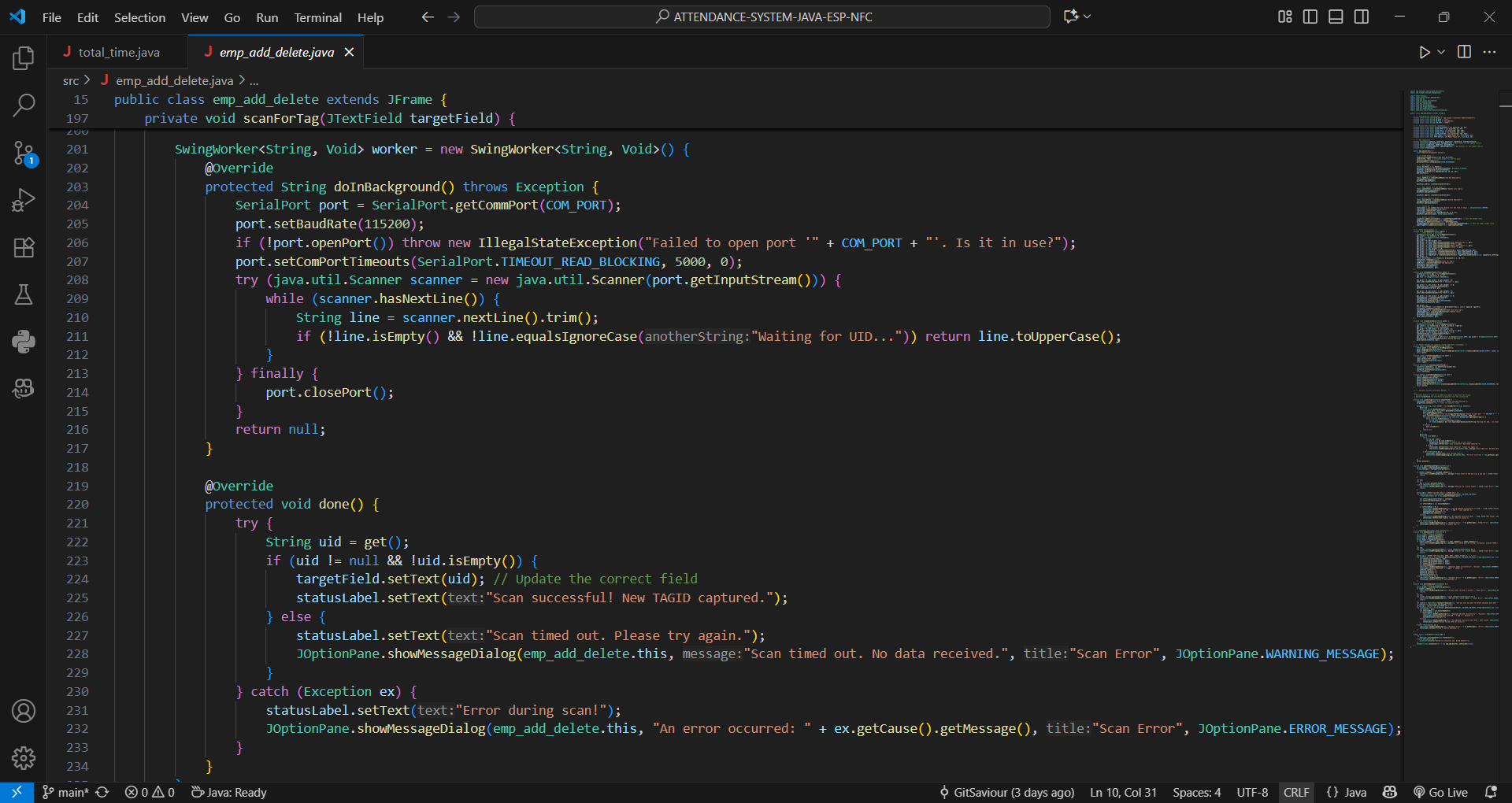
**PROJECT CODE FIGURES**

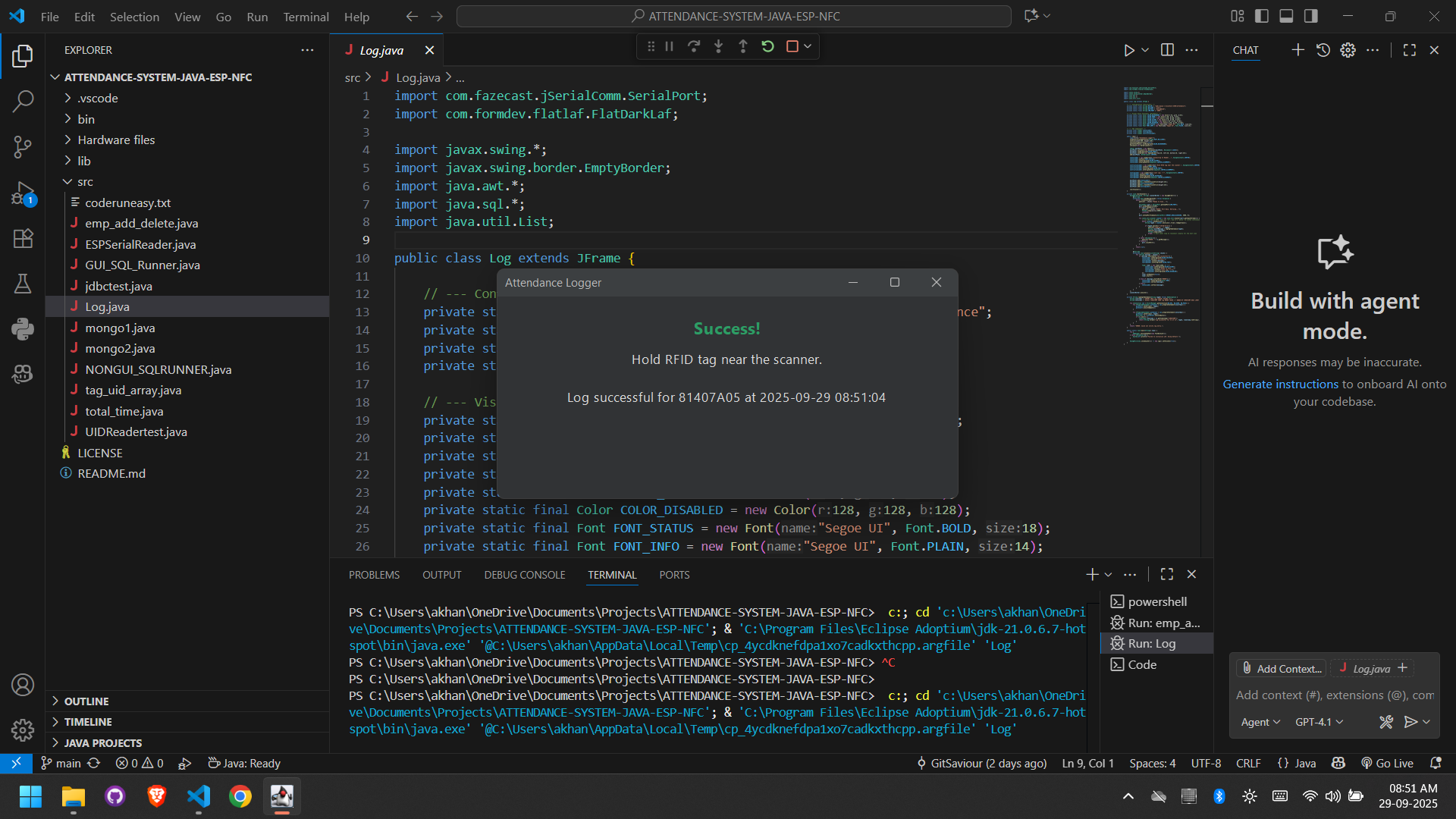
****

****

****

****

****

****

**CHAPTER 2: GUI (GRAPHICAL USER INTERFACE) DEVELOPMENT**

**GUI Development**: A Brief Overview

GUI (Graphical User Interface) development for a desktop application focuses on creating the visual elements that users directly see and interact with. This includes designing the layout of windows, placing components like buttons and text fields, and defining the application's behavior in response to user actions. The goal is to create an interface that is intuitive, functional, and user-friendly for the NFC-based Employee Attendance Management System.

1. Purpose of GUI Development

The main purpose of GUI development is to provide a seamless and efficient experience for the end-user, who in this case, would be an administrator managing the attendance system. A well-designed GUI ensures that the application is:

* Visually Clear and Easy to Navigate: The layout is structured logically, allowing users to easily find functions like adding an employee or viewing reports.
* Responsive to User Input: The application provides immediate feedback for actions, such as confirming that an employee's attendance has been successfully logged.
* Interactive: It offers dynamic components like data tables that update in real-time, forms for data entry, and dialog boxes for alerts and messages.

2. Core Technologies and Libraries

In Java, GUI development is primarily handled by two core libraries:

1. AWT (Abstract Window Toolkit): This is the original, foundational library for creating GUIs in Java. AWT components rely on the native operating system's UI elements, which can lead to a different look and feel on different platforms.
2. Java Swing: Used for this project, Swing is a more advanced and versatile GUI toolkit. It is built on top of AWT but provides a more extensive set of components. Swing components are "lightweight," meaning they are written entirely in Java and paint themselves, ensuring a consistent look and feel across all operating systems.
3. Event Handling: This is the mechanism that makes a GUI interactive. It involves listening for user actions (like a button click or mouse movement) and executing specific code in response. Java uses EventListeners (e.g., ActionListener, MouseListener) to manage this interactivity.

3. Key Components and Tools

To build the application's interface, several Swing components and tools were utilized:

* Containers: These are components that hold and organize other components.
  + JFrame: The main window of the application.
  + JPanel: A general-purpose container used to group related components together within a frame.
  + JDialog: A pop-up window used for showing messages or asking for user input.
* Components (Widgets): These are the interactive elements within the containers.
  + JButton: A clickable button to trigger actions.
  + JLabel: A non-editable text or icon display.
  + JTextField: A box for single-line text input.
  + JTable: A powerful component for displaying data in a grid of rows and columns, essential for showing employee lists and attendance logs.
* Layout Managers: These control the positioning and sizing of components within a container (e.g., BorderLayout, FlowLayout, GridLayout).
* IDE (Integrated Development Environment): Tools like Eclipse or IntelliJ IDEA were used to design, code, and debug the GUI, often providing visual designers that simplify the layout process.

4. Key Features

1. Intuitive Design: The layout of the attendance system is designed to be self-explanatory, ensuring that an administrator can use it with minimal training.
2. Event-Driven Interactivity: ActionListeners are attached to buttons like "Add Employee" or "Generate Report" to execute the corresponding backend logic and database operations.
3. Platform Independence: Because it is built with Java Swing, the application runs consistently across different operating systems like Windows, macOS, and Linux, requiring only the Java Runtime Environment (JRE).
4. Data Visualization: A critical feature is the use of JTable to display large sets of employee and attendance data in a structured, readable format.
5. Real-Time Feedback: The GUI provides immediate feedback to the user, such as displaying "Access Granted" or "Access Denied" after an NFC card is scanned

.

5. GUI Development Process

1. Planning & Design: Create simple sketches (wireframes) of the application's screens to map out the user workflow.
2. Coding: Translate the designs into functional code using Java Swing components, containers, and layout managers. Implement the event-handling logic to connect the GUI to the backend.
3. Testing: Thoroughly test the user interface to find and fix bugs, ensuring all buttons work and the layout is not distorted on different screen sizes.
4. Packaging: Bundle the application into an executable JAR (Java Archive) file for easy distribution and deployment.
5. Maintenance: Make regular updates to improve the user experience or add new features based on feedback.

6. Conclusion

GUI development with Java Swing is essential for creating a functional and user-friendly desktop application like the attendance management system. By using Swing's comprehensive set of components and its robust event-handling model, developers can build powerful, interactive, and platform-independent applications that effectively meet user needs.

**Basic Java Swing Code Example**

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class BasicSwingApp {

public static void main(String[] args) {

// Create the main window (frame)

JFrame frame = new JFrame("Basic Swing Application");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(300, 200);

// Create a button

JButton button = new JButton("Click Me");

// Add an ActionListener to the button to handle clicks

button.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// This code runs when the button is clicked

System.out.println("Button was clicked!");

JOptionPane.showMessageDialog(frame, "You clicked the button!");

}

});

// Add the button to the frame's content pane

frame.getContentPane().add(button);

// Make **the frame visible**

**frame.setVisible(true);**

**}**

**}**

# CHAPTER 3 BACKENED DEVELOPMENT

The "backend" refers to the core processing engine that operates behind the Graphical User Interface (GUI). It encompasses all the non-visual components, including the application's core logic, business rules, database interactions, and hardware communication. While the Swing GUI handles user interaction, the backend does all the heavy lifting to ensure the application functions correctly, securely, and efficiently.

Purpose of the Backend

The primary purpose of the application's backend logic is to:

* Handle Data Processing, Storage, and Retrieval: Manage all employee and attendance data stored in the database.
* Implement Business Logic: Enforce the rules of the attendance system, such as validating an employee's NFC scan, preventing duplicate entries, and calculating work hours.
* Manage Access Control: If applicable, handle administrator logins and permissions within the application.
* Ensure Performance and Reliability: Optimize data operations and handle potential errors gracefully to keep the application stable.
* Connect All Components: Serve as the bridge connecting the GUI, the database, and the external NFC reader hardware.

Key Components of the Backend

1. Core Programming Language (Java):
   * Java serves as the foundation for the entire backend. Its Object-Oriented Programming (OOP) principles were used to model the system's entities. For instance, Employee and AttendanceRecord classes were created to represent real-world data, making the code organized, reusable, and easy to maintain.
2. Database:
   * The database is where all the application's persistent data is stored, such as employee details (ID, name, NFC card ID) and their attendance logs (date, time-in, time-out).
   * For this project, a Relational Database (SQL-based) like MySQL was chosen because the data is highly structured and relational in nature.
3. Database Connectivity (JDBC - Java Database Connectivity):
   * JDBC is the critical API that enables the Java application to communicate with the database. It acts as the bridge for sending and receiving data. The process involves:
     + Establishing a connection to the database.
     + Creating and executing SQL statements (e.g., INSERT, SELECT, UPDATE).
     + Processing the results returned by the database and mapping them to Java objects.
4. Hardware Integration Layer:
   * A unique and essential part of this project's backend is the code dedicated to communicating with the NFC reader. This layer uses specific Java libraries (e.g., javax.smartcardio) to:
     + Listen for a connected NFC reader.
     + Detect when an NFC card is tapped.
     + Read the unique identifier (UID) from the card.
     + Pass this UID to the business logic layer for processing.
5. Authentication and Authorization:
   * This component manages user access. For this system, it could be a simple login screen requiring an administrator's username and password. The backend logic verifies these credentials against data securely stored in the database to grant access to the application's management functions.

Key Features of the Backend

1. Data Management (CRUD Operations):
   * The backend manages the complete lifecycle of data. Using JDBC, it performs CRUD (Create, Read, Update, Delete) operations. For example, it creates new employee records, reads attendance logs for reports, updates employee details, and deletes records when necessary.
2. Business Logic:
   * This is the set of rules that define how the attendance system works. Key logic implemented in the backend includes:
     + When an NFC card is scanned, the backend searches the database for the corresponding employee.
     + It determines if the scan is a "check-in" or "check-out" based on previous records for that day.
     + It records the event with a precise timestamp.
     + It handles logic for generating daily, weekly, or monthly attendance reports.
3. Security:
   * Protecting employee data is crucial. The backend ensures security through several measures:
     + Using PreparedStatements in JDBC to prevent SQL injection attacks, which is a common vulnerability.
     + Securely handling database credentials within the application code.
     + Validating all data received from the GUI before processing it.
4. Efficiency and Reliability:
   * The backend is designed to be efficient, with optimized database queries to ensure fast responses, especially during a real-time NFC scan.
   * Robust error handling using Java's try-catch-finally blocks ensures that the application can gracefully manage issues like database connection failures or hardware errors without crashing.

**CHAPTER 4: SYSTEM IMPLEMENTATION AND KEY MODULES**

This chapter details the technical implementation of the NFC-based Employee Attendance Management System. The application was developed entirely in Java, leveraging several key technologies to create a robust, responsive, and functional desktop solution.

The system is architected into three primary modules, each corresponding to a core piece of functionality:

1. **Employee Management Module**: For administering employee records.
2. **Real-time Attendance Logging Module**: The core terminal for capturing punches.
3. **Reporting and Time Calculation Module**: For analyzing attendance data.

Across all modules, the following core technologies were consistently used:

* GUI: Java Swing was used to build the graphical user interface, with the FlatLaf library providing a modern dark theme.
* Database Connectivity: JDBC (Java Database Connectivity) was used to interface with a MySQL relational database.
* Hardware Integration: The jSerialComm library was used to establish serial communication with the external NFC/RFID reader.
* Concurrency: SwingWorker was implemented to handle long-running tasks like hardware scanning and database queries in background threads, ensuring the user interface remains responsive at all times.

Module 1: Employee Management (emp\_add\_delete.java)

This module serves as the administrative control panel for the system. It allows a system administrator to perform essential CRUD (Create, Read, Update, Delete) operations on employee records.

GUI Implementation:

The user interface is built using a JFrame that contains three distinct JPanels, each dedicated to a specific function: "Add New Employee," "Update Lost Tag," and "Delete Employee." A GridBagLayout is used within each panel to organize labels (JLabel) and input fields (JTextField) in a clean, structured grid. Helper methods like createStyledPanel() and createStyledButton() were used to ensure a consistent and modern visual style throughout the module.

Backend and Hardware Logic:

The true functionality is driven by the backend logic, which seamlessly integrates hardware and database operations.

1. NFC Tag Scanning: The scanForTag() method is responsible for interfacing with the NFC reader. When the "Scan" button is clicked, a SwingWorker is initiated.
   * The doInBackground() method opens a connection to the NFC reader via the specified serial port (COM3) using the jSerialComm library.
   * It waits to receive the unique ID (UID) from a scanned tag. This is a blocking I/O operation, and running it in a background thread is critical to prevent the GUI from freezing.
   * The done() method is executed on the GUI thread after the scan is complete (or times out). It updates the appropriate JTextField with the captured Tag ID and provides feedback to the user via a JLabel.
2. Database Operations:
   * Add Employee: The addEmployee() method retrieves data from the input fields, performs basic validation (checking for empty fields, parsing numbers), and constructs a SQL INSERT statement.
   * Update Tag: The updateEmployeeTag() method allows an administrator to assign a new NFC tag to an existing employee, executing a SQL UPDATE statement.
   * Delete Employee: The deleteEmployee() method executes a SQL DELETE statement after a confirmation dialog, preventing accidental data loss.

Crucially, all database queries are executed using PreparedStatement. This is a vital security practice that parameterizes the SQL queries, effectively preventing SQL injection attacks and ensuring data integrity.

Module 2: Real-time Attendance Logging (Log.java)

This is the core operational module of the system, designed to run continuously at an entry/exit point to log employee attendance in real-time.

GUI Implementation:

The interface is intentionally minimalist, consisting of a single JFrame with a large status label (JLabel). Its purpose is to provide immediate and clear feedback to the employee upon scanning their card, displaying messages like "Ready to Scan...", "Success!", or "ERROR".

Backend and Hardware Logic:

This module's effectiveness relies on its ability to run a persistent background process.

1. Continuous Scanning Loop: The startScanner() method launches a SwingWorker that contains an infinite loop (while (!isCancelled())). This ensures that the application is always listening for an NFC tag without requiring any user interaction.
2. Hardware Communication: Inside the loop, the jSerialComm library is used to open the serial port and listen for data from the NFC reader. The loop includes error handling to retry the connection if the port is busy.
3. Safe GUI Updates: As events occur in the background thread (e.g., a tag is scanned, an error occurs), the publish() method is called. This sends messages to the process() method, which safely updates the statusLabel on the GUI thread. This publish/process mechanism is the correct way to handle real-time UI updates from a background task in Swing.
4. Database Logging: Upon receiving a valid Tag ID, the logTagToDatabase() method is called. It connects to the database via JDBC and executes an INSERT statement into the log table. This action creates an immutable record of every scan, with the timestamp generated by the database server itself for maximum accuracy.

Module 3: Reporting and Time Calculation (total\_time.java)

This module provides the analytical capabilities of the system, allowing an administrator to query the attendance database and calculate the total work hours for employees.

GUI Implementation:

The interface is divided into two main panels: one for "Daily Hours Query" and another for "Monthly Hours Query." It utilizes a combination of Swing components to create an intuitive user experience:

* JFormattedTextField with a MaskFormatter is used for date input, ensuring the format is always correct.
* JComboBox and JSpinner are used for easy selection of the month and year for monthly reports.

Backend and Business Logic:

This module contains the most complex business logic in the entire application.

1. Data Retrieval: The getLogsForDay() method is a helper function that queries the database for all attendance logs for a specific Employee ID on a specific date, ordered by time. This provides the raw data needed for calculation.
2. Daily Hours Calculation: The calculateDailyHours() method fetches the logs for a day and processes them. The algorithm pairs the timestamps (first with second, third with fourth, etc.) to represent check-in/check-out events. The time difference for each pair is calculated and summed up to get the total duration.
3. Monthly Hours Calculation: The calculateMonthlyHours() method iterates through every day of the selected month. For each day, it calls the daily calculation logic and aggregates the results to produce a total for the entire month.
4. Handling Irregularities: The system is robustly designed to handle edge cases. If an employee has an odd number of logs for a day (e.g., they forgot to scan out), the system flags this as an irregularity. It displays a warning message to the administrator and, as a fallback, calculates the duration between the very first and very last scan of that day. This ensures that some data is still provided while also highlighting a potential issue for manual review.

CHAPTER 5: KEY LIBRARIES AND FRAMEWORKS

In modern software development, building an application from scratch is inefficient. Instead, developers leverage powerful, pre-existing libraries and frameworks to accelerate development, improve reliability, and add sophisticated features that would otherwise be time-consuming to create. For the NFC-based Employee Attendance Management System, two critical third-party libraries were instrumental:

1. **FlatLaf**: A modern "Look and Feel" library for styling the user interface.
2. **jSerialComm**: A hardware communication library for interfacing with the NFC reader.

These libraries serve a similar purpose to frameworks like Bootstrap in web development: they provide a robust foundation of pre-built functionality and styling, allowing the developer to focus on the core business logic of the application.

Purpose and Benefits of Using Libraries

The primary purpose of integrating these libraries was to streamline the development process and deliver a more professional and functional application.

The key benefits included:

* Speed and Efficiency: Using FlatLaf eliminated the need to manually design and code a custom theme for the application's UI. Similarly, jSerialComm provided a ready-to-use solution for serial port communication, saving dozens of hours of complex, low-level coding.
* Enhanced User Experience: The default Java Swing appearance can look dated. FlatLaf immediately provided a clean, modern, and visually appealing dark theme (FlatDarkLaf), significantly improving the user experience and giving the application a professional finish.
* Reliability and Platform Independence: These libraries are well-maintained and tested. jSerialComm abstracts away the operating system-specific complexities of serial port handling, allowing the application to run on Windows, macOS, or Linux without code changes.
* Focus on Core Logic: By entrusting the UI aesthetics to FlatLaf and hardware communication to jSerialComm, the development effort was concentrated on what mattered most: the business logic for managing employees, logging attendance, and calculating work hours.

Key Library Feature: FlatLaf Look and Feel

FlatLaf is a modern, open-source Look and Feel (LaF) library for Java Swing applications. Its role is directly analogous to a CSS framework like Bootstrap—it completely re-styles the application's visual components.

Implementation:

As seen in the main method of each module, integrating FlatLaf was remarkably simple:

Java

try {

UIManager.setLookAndFeel(new FlatDarkLaf());

} catch (Exception ex) {

System.err.println("Failed to initialize LaF. Using default.");

}

With this single line of code, the entire application's appearance was transformed.

Key Features Utilized:

* Modern Dark Theme: The project specifically used the FlatDarkLaf theme to provide a sleek, modern interface that is easy on the eyes, a popular choice in today's software.
* Automatic Component Styling: Once the Look and Feel was set, FlatLaf automatically restyled all standard Swing components (JFrame, JButton, JTextField, JPanel, etc.) to ensure a consistent and professional design across all windows and dialogs.
* High-Resolution Display Support: FlatLaf is designed for modern hardware, providing excellent scaling on HiDPI (High Dots Per Inch) displays, ensuring the UI remains crisp and clear on any monitor.

Key Library Feature: jSerialComm Hardware Communication

jSerialComm is a platform-independent Java library for accessing and communicating with serial port devices like the NFC reader used in this project. It is analogous to Bootstrap's JavaScript plugins, providing complex, pre-built functionality that makes it easy to add interactive features.

Implementation:

The library was used extensively in the emp\_add\_delete and Log modules to read data from the NFC reader. The core logic for establishing a connection is straightforward:

Java

// Get the specific communication port

SerialPort port = SerialPort.getCommPort("COM3");

// Configure port settings

port.setBaudRate(115200);

// Open the port to begin communication

port.openPort();

// Read data from the port's input stream

java.util.Scanner scanner = new java.util.Scanner(port.getInputStream());

Key Features Utilized:

* Platform Independence: This is the most critical feature of jSerialComm. It handles the low-level, OS-specific code required to access serial ports, meaning the project's Java code can run on any platform without modification.
* Simple Port Management: The library provides an easy-to-use API for discovering, selecting, and configuring serial ports, including setting parameters like baud rate and timeouts.
* Data Stream Abstraction: jSerialComm abstracts the serial port into a standard Java InputStream. This allows the application to use familiar Java classes like java.util.Scanner to read data, dramatically simplifying the code required to capture the Tag ID from the reader. This abstraction is visible in the scanForTag() and startScanner() methods of the project.

# CHAPTER 6 DBMS

**DBMS** stands for **Database Management System**. It is software that helps to store, manage, and retrieve data efficiently in a structured way. In web development, DBMS is crucial because websites and web applications often need to handle large amounts of data, such as user information, posts, products, and transactions. A DBMS helps developers manage this data effectively and securely.

## Purpose of DBMS in Web Development

In web development, the purpose of a DBMS is to:

1. **Store Data**: It provides a structured way to store data such as text, numbers, images, and other media.
2. **Retrieve Data**: Allows efficient retrieval of data when needed, such as when displaying user profiles or product listings on a website.
3. **Manage Data Relationships**: Helps organize and maintain relationships between different types of data, such as linking users to their posts or orders.
4. **Ensure Data Security**: A DBMS ensures data is protected, providing features like access control, encryption, and data backups.

## Types of DBMS Used in Web Development

1. **Relational DBMS (RDBMS)**:
   * In an RDBMS, data is stored in tables with rows and columns, like a spreadsheet. The data in different tables can be linked using **keys**.
   * Example: **MySQL**, **PostgreSQL**, **SQLite**, and **Microsoft SQL Server**.
   * Used for structured data that fits into a clear format, such as user profiles or orders.
   * **SQL (Structured Query Language)** is used to interact with the database, including adding, updating, and retrieving data.

## NoSQL DBMS:

* + NoSQL databases are used when the data is unstructured or semi-structured, such as documents, key-value pairs, or graph-based data.
  + Example: **MongoDB**, **Firebase**, **Cassandra**.
  + More flexible than RDBMS and useful for handling large-scale data with variable formats, like social media posts or IoT sensor data.
  + NoSQL databases use a different query language (e.g., MongoDB uses its own query format).

## Advantages of Using DBMS in Web Development

1. **Data Integrity**: Ensures that data is accurate and consistent across the system. For example, when you update a user’s email address, the DBMS ensures the change is applied correctly everywhere it’s needed.
2. **Security**: DBMS systems provide features such as access control, encryption, and backup capabilities to ensure data is protected against unauthorized access and loss.
3. **Scalability**: Modern DBMS like **MongoDB** and **PostgreSQL** can handle large volumes of data and allow web applications to scale as the number of users or data increases.
4. **Efficiency**: DBMS systems are optimized to quickly retrieve and process data, ensuring fast response times for web applications.
5. **Data Relationships**: Relational databases allow developers to create relationships between different tables, such as linking users to their orders, which helps organize and manage complex data.

# CONCLUSION

# This training project culminated in the successful development of a complete NFC-based Employee Attendance Management System, serving as a comprehensive application of the principles learned in Java and Advanced Java. The project successfully demonstrates that a powerful and practical solution can be built by integrating distinct technological components into a single, cohesive system.

# The key takeaway from this endeavor is the understanding of how different layers of an application are intertwined. The project required more than just knowledge of a single technology; it necessitated the seamless integration of:

# A Graphical User Interface (GUI) built with Java Swing for user interaction.

# A robust backend using Core Java to implement the essential business logic.

# A persistent data layer managed through JDBC for secure and reliable communication with a MySQL database.

# A hardware integration layer to communicate directly with the external NFC reader.

# The development process underscored the importance of concepts like concurrency, where SwingWorker was used to ensure the application remained responsive during hardware and database operations. It also highlighted the practical application of secure coding practices, such as using PreparedStatement to prevent SQL injection.

# Ultimately, this project achieved its goal of creating a functional, end-to-end system that automates the attendance process, thereby increasing efficiency and accuracy. It provided invaluable hands-on experience, bridging the gap between theoretical knowledge and real-world application development. The skills acquired in managing a full software lifecycle—from UI design and hardware interfacing to backend logic and database management—form a solid foundation for future challenges in the field of software engineering.

# FUTURE SCOPE

The current desktop application serves as a strong foundation for significant future enhancements that can evolve it into a modern, enterprise-grade solution.

Key advancements would focus on a **web-based architecture**, featuring a remote management dashboard built with a framework like Spring Boot and a centralized cloud database for universal access. The PC-based logging terminals could also be replaced with dedicated, low-cost **IoT devices** for more scalable and efficient deployment across multiple locations.

To enhance system intelligence and security, **Artificial Intelligence** could be integrated to perform predictive analysis on attendance patterns and detect anomalies. For maximum accuracy, **biometric authentication**, such as fingerprint scanning, could be added as a second verification factor alongside the NFC scan to eliminate proxy attendance.

Finally, user accessibility could be improved by developing a **companion mobile application** for employees to view their records, supported by an automated notification system to keep staff and management informed. These upgrades would transform the foundational application into a comprehensive and intelligent workforce management platform.

# REFERENCES

1. **Telusko (YouTube Channel)**
   * **URL:** [**https://www.youtube.com/c/Telusko**](https://www.youtube.com/c/Telusko)
   * **Description: An educational YouTube channel that provides in-depth video tutorials on Core Java, Advanced Java topics like JDBC, and database connectivity. This resource was fundamental for building the backend logic and data management layers of the project.**
2. **Oracle Java Documentation**
   * **URL:** [**https://docs.oracle.com/en/java/**](https://docs.oracle.com/en/java/)
   * **Description: The official and most authoritative source for the Java programming language. The API documentation for Java Swing (for the GUI) and JDBC (for database interaction) was an essential reference for implementing the project's core features.**
3. **W3Schools (SQL Tutorials)**
   * **URL:** [**https://www.w3schools.com/sql/**](https://www.w3schools.com/sql/)
   * **Description: A popular online learning platform. Its clear and concise tutorials on SQL (Structured Query Language) were used to write and optimize the database queries that the Java application executes via JDBC.**
4. **Stack Overflow**
   * **URL:** [**https://stackoverflow.com/**](https://stackoverflow.com/)
   * **Description: A large online community for programmers. This platform was an invaluable resource for troubleshooting specific technical challenges and finding solutions to common errors encountered during development with Java Swing, JDBC, and hardware integration.**
5. **Google Gemini**
   * **URL:** [**https://gemini.google.com/**](https://gemini.google.com/)
   * **Description: An AI-powered assistant from Google. It was used throughout the development cycle for generating code snippets, debugging complex logic, optimizing algorithms, and refining technical documentation for this report.**
6. **jSerialComm (Library Documentation)**
   * **URL:** [**https://github.com/Fazecast/jSerialComm**](https://github.com/Fazecast/jSerialComm)
   * **Description: The official GitHub repository for the jSerialComm library. The documentation and examples provided here were crucial for successfully implementing the communication between the Java application and the external NFC serial reader.**