PAYROLL MANAGEMENT SYSTEM

SAMEER AHMED
(Roll No.: 24GCA021)

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF SCIENCES

JAMIA MILLIA ISLAMIA

MAY, 2025

Payroll Management System

By

Sameer Ahmed

(Enrollment No.: 24-01580)

Submitted

in partial fulfillment of the requirements of the degree of

Post Graduate Diploma in Computer Application (PGDCA)

to the

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF SCIENCES

JAMIA MILLIA ISLAMIA

MAY, 2025

Declaration

I, Sameer Ahmed, student of PGDCA hereby declare that the project report entitled **“ Payroll Management System ”** which is submitted by me to the Department of Computer Science, Jamia Millia Islamia, New Delhi, in partial fulfillment of the requirement of the degree of **Post Graduate Diploma in Computer Applications (PGDCA)**, is a record of the original work carried out by me and have not been submitted in part or full to any other university or institute for the award of any degree or diploma.

Sameer Ahmed

Roll No.: 24GCA021

Enrollment No.: 24-01580

Certificate

On the basis of declaration made by the student **Sameer Ahmed**, I hereby certify that the project report entitled **“ Payroll Management System ”** submitted by **Sameer Ahmed** to the Department of Computer Science, Jamia Millia Islamia, New Delhi, for the partial fulfillment of the requirements of the degree of **Post Graduate Diploma in Computer Applications (PGDCA)** is carried out by him under my/our guidance and supervision. The report has reached the requisite standards for submission.

Prof. M. Nazir
(Supervisor’s Signature)

Acknowledgement

I would like to express my sincere gratitude to my respected guide **Prof. M. Nazir**

for their invaluable guidance, encouragement, and continuous support throughout the development of this project. Their constructive feedback and insightful suggestions played a significant role in shaping this work into its present form.

I extend my heartfelt thanks to my parents, friends, and peers for their constant motivation, support, and understanding during the course of this work.

I am also thankful to the faculty members of the **Department of Computer Science**, and the administration of **Jamia Millia Islamia** for providing me with the necessary resources and academic environment to successfully complete this project.

Sameer Ahmed

Roll No.: 24GCA021

Enrollment No.: 24-01580

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This project, titled *Payroll Management System*, is designed to automate employee salary management using Python and SQLite. It eliminates manual errors in payroll processing by automating salary calculations, deductions, and receipt generation. The system provides secure role-based access for administrators and employees, allowing them to manage and view salary records efficiently. Built using Python's Tkinter library with ttkbootstrap styling, the application features modules for login authentication, employee management, payroll computation, salary slip generation in PDF/TXT formats, and a built-in calculator tool. By integrating database operations with a user-friendly GUI, this project aims to simplify payroll tasks for small to medium-sized organizations, ensuring accuracy, transparency, and efficiency in employee compensation management.

Chapter 1: Introduction

1.1 Background

Payroll processing forms the backbone of every organization, big or small. Employees expect timely and accurate compensation for their work, and organizations are responsible for ensuring compliance with tax regulations, deductions, and benefits. Traditionally, many companies relied on manual systems where salary calculations were performed using spreadsheets or even pen-and-paper. These methods, though simple, are highly prone to human errors, time-consuming, and difficult to maintain as the company grows. Mistakes in payroll can lead to employee dissatisfaction, legal troubles, and financial mismanagement. An automated Payroll Management System addresses these issues by providing a centralized platform to store employee data, compute salaries, generate payslips, and maintain transaction records. Such systems not only improve accuracy but also enhance transparency and operational efficiency. Thanks to the rise of open-source technologies like Python and SQLite, it has become feasible for even small and medium enterprises (SMEs) to adopt customized payroll solutions that fit their unique requirements.

1.2 Purpose & Scope

The purpose of this Payroll Management System is to automate every aspect of employee salary management. From registering an employee to computing their monthly salary with allowances and deductions, and finally generating a professional salary slip, this system aims to handle the complete payroll cycle with minimal manual intervention.
The scope of the project covers maintaining a comprehensive employee database, securely processing payroll calculations, applying tax and insurance deductions, and storing salary records for future reference. The system includes a Graphical User Interface (GUI) built using Python's Tkinter library and styled with ttkbootstrap, making it intuitive even for users with little technical knowledge. Additionally, the application features role-based access, where administrators have full control over employee and salary management, while employees are granted limited access to view their own payslips and details. The inclusion of a built-in calculator and receipt generation in PDF and TXT formats adds further value, ensuring the system's practical applicability for real-world businesses.

1.3 Objectives

The primary objective of this project is to automate the entire payroll process, thereby eliminating manual errors and inefficiencies. Specifically, the system is designed to:

- Provide a centralized and secure database for storing employee and salary records.
- Perform accurate and automated salary computations, factoring in allowances (DA, HRA, MA) and deductions (PF, tax, insurance).
- Generate professional, formatted salary slips in both PDF and TXT formats.
- Offer an intuitive and user-friendly interface using modern styling through ttkbootstrap.
- Implement a role-based login system, providing secure access for admins and employees.
- Include utility features such as a built-in calculator for quick computations.
- Facilitate CRUD (Create, Read, Update, Delete) operations for managing employee and salary records.

Ultimately, the system aims to enhance the efficiency, accuracy, and scalability of payroll processing for small to medium organizations.

1.4 Technologies Used

The system leverages several modern technologies. The core logic is developed using **Python[1] 3.x**, known for its readability and robustness. The GUI is built with **Tkinter[2]**, Python's standard interface for creating desktop applications, enhanced with **ttkbootstrap[5]** themes for a modern look.

SQLite[3] serves as the database engine, offering lightweight yet powerful data storage without requiring a separate server. For generating PDF salary slips, the project employs the **ReportLab[4]** library, a widely used tool for creating PDF documents programmatically. Other libraries like **hashlib** ensure secure password storage via SHA-256 hashing, while datetime, os, and tkinter.ttk are used for date handling, file management, and styling. The use of these open-source technologies ensures that the system remains cost-effective, efficient, and adaptable to different user requirements.

Chapter 2: System Analysis

2.1 Problem Definition

In many organizations, payroll processing is still done manually using spreadsheets or outdated software that requires significant human involvement. This manual process is error-prone, leading to mistakes in salary calculations, incorrect tax deductions, or missed benefits, which can negatively affect employee morale and violate compliance regulations. Moreover, generating payslips, maintaining salary records, and retrieving historical data become cumbersome tasks when handled manually.

The primary problem this project aims to solve is the inefficiency and inaccuracy of manual payroll management. By providing an automated system, it reduces human errors, saves time, and improves the overall payroll workflow.

2.2 Existing System Vs Proposed System

The existing manual system in many companies involves HR staff manually entering employee data, calculating salaries using Excel formulas, and generating payslips using Word or email. This method is prone to miscalculations, data duplication, and security breaches. In contrast, the proposed Payroll Management System automates the entire process. Employee details and salary records are stored in an SQLite database, salaries are computed automatically using programmed logic, and payslips are generated instantly in PDF/TXT format. Role-based login ensures that sensitive data is only accessible to authorized personnel. Overall, the proposed system offers improved accuracy, security, and operational efficiency compared to the manual method.

2.3 Feasibility Study

- **Technical Feasibility:** The system is built using open-source technologies like Python, SQLite, and ReportLab, which are widely supported and require minimal hardware resources. It runs on standard Windows and Linux systems without the need for special infrastructure, making it technically feasible for most organizations.

- **Operational Feasibility:** The system features an intuitive GUI styled with ttkbootstrap, making it easy for HR staff and employees to use without requiring advanced technical skills. Role-based access ensures that admins have full control, while employees can securely view their own records.
- **Economic Feasibility:** Since the system is developed using free, open-source tools, there are no licensing costs. The hardware requirements are minimal, and the cost of deployment is limited to basic system setup, making it economically viable for small and medium enterprises.

2.4 Software & Hardware Requirements

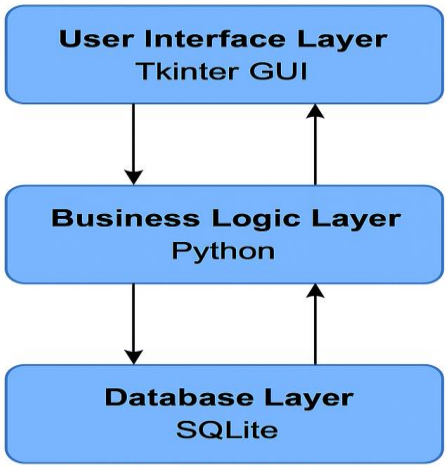
- **Software Requirements:**
  - Python 3.x (recommended version 3.8 or higher)
  - Tkinter (comes built-in with Python)
  - SQLite3 (bundled with Python)
  - ReportLab library for PDF generation
  - ttkbootstrap library for modern GUI styling
  - hashlib library for secure password storage
- **Hardware Requirements:**
  - A personal computer or laptop with at least 2GB RAM (4GB recommended)
  - Processor: Intel i3 or equivalent (i5 or higher recommended)
  - Storage: 100MB free disk space for application and database
  - Display resolution: Minimum 1024x768 pixels

---XXX---

Chapter 3: System Design

3.1 System Architecture

Figure 3.1: System Architecture



The Payroll Management System follows a modular and layered architecture to ensure scalability, maintainability, and clear separation of concerns. The application is organized into three primary layers:

- **User Interface Layer:** This is the front-end of the system, built using Python's Tkinter library. It provides graphical forms through which users can interact with the application. Users input data such as employee details, salary components, and login credentials via this interface.
- **Business Logic Layer:** This is the core of the application, responsible for handling logic-based operations. It performs functions such as login authentication, salary calculations, data validation, and integration of GUI actions with database queries. All processing logic is handled here, using Python scripts.
- **Database Layer:** This layer is powered by SQLite, a lightweight and efficient relational database system. It securely stores structured data such as employee

records, salary entries, and login credentials. Each form in the GUI interacts with this layer through SQL commands handled by Python's database APIs.

3.2 Data Flow Diagrams (DFD)

To visualize how data moves through the system, Data Flow Diagrams (DFDs) are used at different levels:

- **DFD Level 0 (Context Diagram):**

  At this level, the system is viewed as a single process. It receives external inputs like login credentials, employee information, and salary details from users. These inputs are processed internally, and the system produces outputs such as calculated net salaries, payslips, and confirmation messages.

Figure 3.2: DFD Level 0

- **DFD Level 1 (Process Breakdown):**

  This level breaks down the major system processes:

  - The **Login Module** validates user credentials by checking them against the admins or employees tables in the database.
  - The **Employee Management Module** handles CRUD operations for employee records.
  - The **Salary Calculation Module** computes gross and net salary by processing allowances (DA, HRA, MA) and deductions (PF, tax, insurance).

- The **Receipt Generation Module** fetches salary records and formats them into professional documents (PDF or TXT) using ReportLab. Each module interacts with the SQLite database to retrieve or store data as needed.

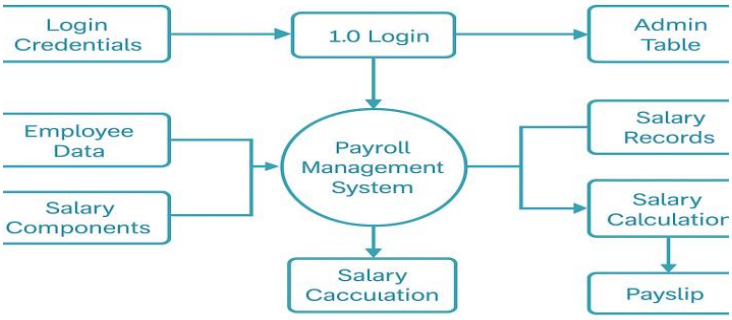


Figure 3.3: DFD Level 1

3.3 ER Diagram

The Entity-Relationship (ER) diagram outlines the data structure of the Payroll Management System. The system features the following core entities:

- **Employee:**

  Stores personal and professional details of employees.

  Attributes: emp_id, name, email, phone, address, gender, department, designation, doj, password

- **Salary:**

  Maintains detailed records of monthly salaries for each employee.

  Attributes: id, emp_id, basic_salary, da, hra, ma, pf, insurance, tax, net_salary, date

- **Admin:**

  Stores admin login information.

  Attributes: id, username, password

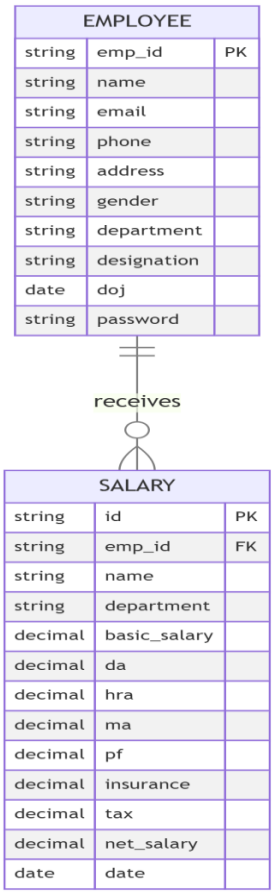


Figure 3.4: ER Diagram

The **relationship** between Employee and Salary is **one-to-many**, meaning one employee can have multiple salary records corresponding to different time periods. This relationship is enforced through the emp_id foreign key in the salaries table.

3.4 Database Design (Tables & Fields)

The database of the Employee Payroll Management System is structured using **SQLite**, a lightweight relational database system that stores data locally in a .db file. The schema is designed to normalize employee and salary data efficiently, avoid redundancy, and support scalable payroll operations. There are three primary tables in the database: employees, salaries, and admins.

- **Employees Table**

  This table stores detailed personal and professional information of each employee. Every record is uniquely identified by emp_id.

Field Name	Data Type	Description
emp_id (PK)	TEXT	Primary Key, unique identifier for employee
name	TEXT	Full name of the employee
email	TEXT	Email address
phone	TEXT	Contact number
address	TEXT	Residential address
gender	TEXT	Gender (Male/Female/Other)
department	TEXT	Department name (e.g., HR, IT, Finance)
designation	TEXT	Job title or role
date_of_joining	TEXT	Joining date of the employee
password	TEXT	Encrypted password (SHA-256 hashed)

- **Salaries Table**

  This table records monthly salary data for employees. Each entry is linked to an employee through a foreign key emp_id.

Field Name	Data Type	Description
id (PK)	INTEGER	Auto-incremented primary key
emp_id (FK)	TEXT	Foreign Key referencing `employees.emp_id`
name	TEXT	Redundant for quick access and display
department	TEXT	Department of the employee
basic_salary	REAL	Base salary value
da	REAL	Dearness Allowance
hra	REAL	House Rent Allowance
ma	REAL	Medical Allowance
pf	REAL	Provident Fund deduction
insurance	REAL	Health/accident insurance deduction
tax	REAL	Income Tax deduction
net_salary	REAL	Final salary after deductions
date	TEXT	Date of salary generation

3.5 GUI Design (Screenshots)

The graphical user interface (GUI) of the Employee Payroll Management System is designed using **Tkinter** and styled with **ttkbootstrap** to ensure a modern, intuitive, and user-friendly experience. Each screen in the application is dedicated to a specific function, structured clearly with proper labels, entry fields, buttons, and tables. The design prioritizes usability, ensuring that both administrators and employees can interact with the system effortlessly.

Login Form

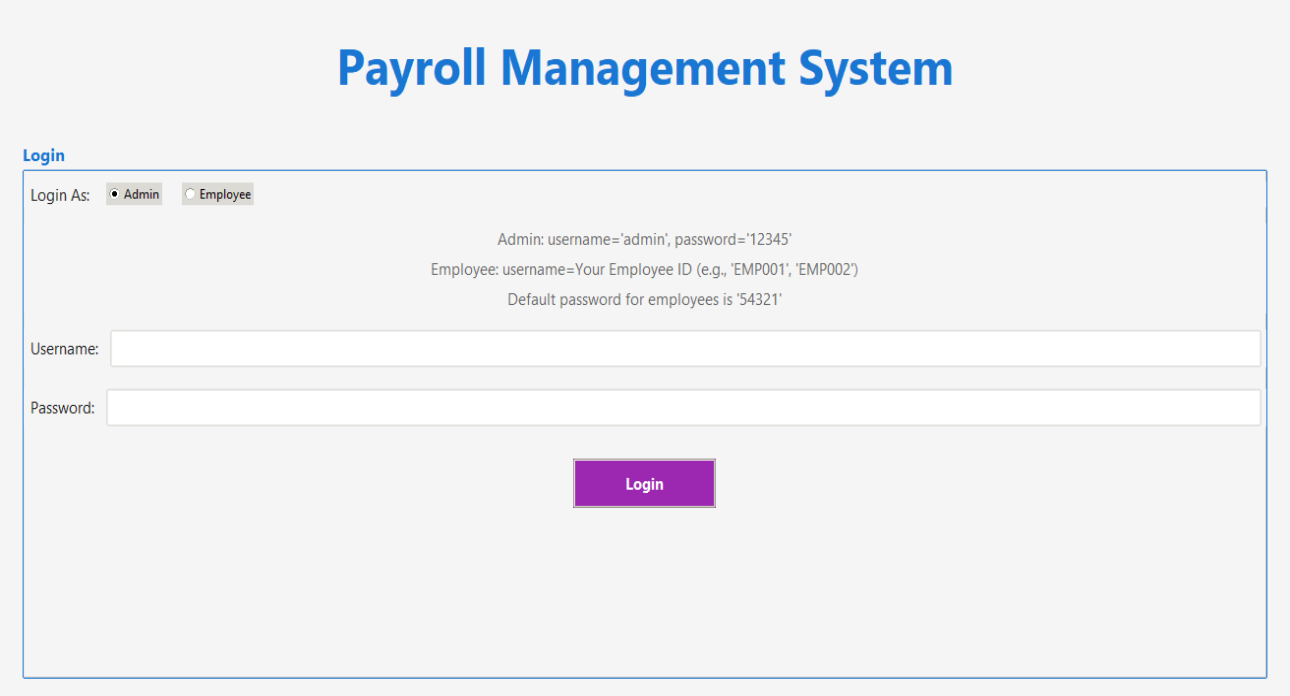


Figure 3.5: Login Form Screenshot

The login form serves as the entry point to the system. It features input fields for the username and password, and includes role-based authentication logic. Based on the credentials entered, the system identifies the user as an **Admin** or **Employee** and redirects them to their respective dashboards. Error handling is implemented to alert users in case of invalid login attempts.

Employee Management Screen

This screen is accessible to admin users only. It allows them to:

- **Add** new employees
- **Update** existing employee details
- **Delete** outdated records
- **Search** employees based on various filters (e.g., ID, name, department) Input fields are available for entering personal and professional details such as name, email, phone number, department, designation, etc. A live table widget displays all employee records in real time.

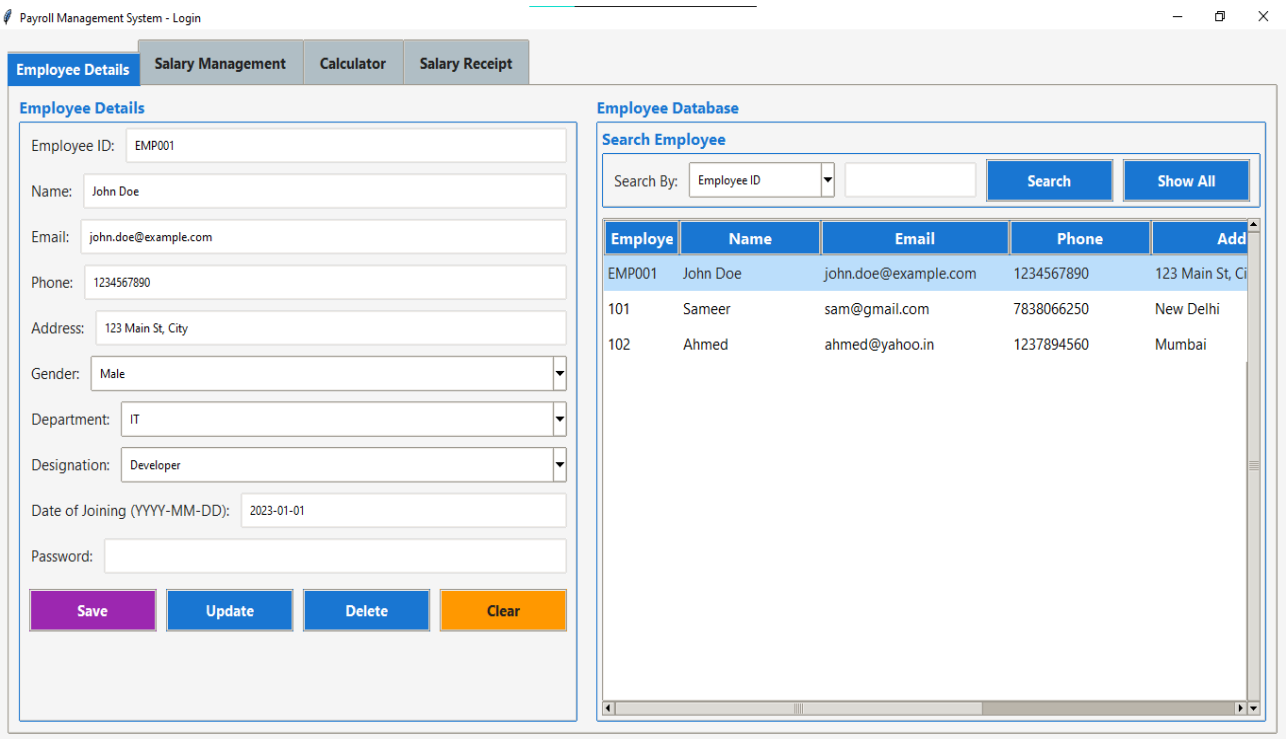


Figure 3.6: Employee Management Screen

Salary Calculation Form

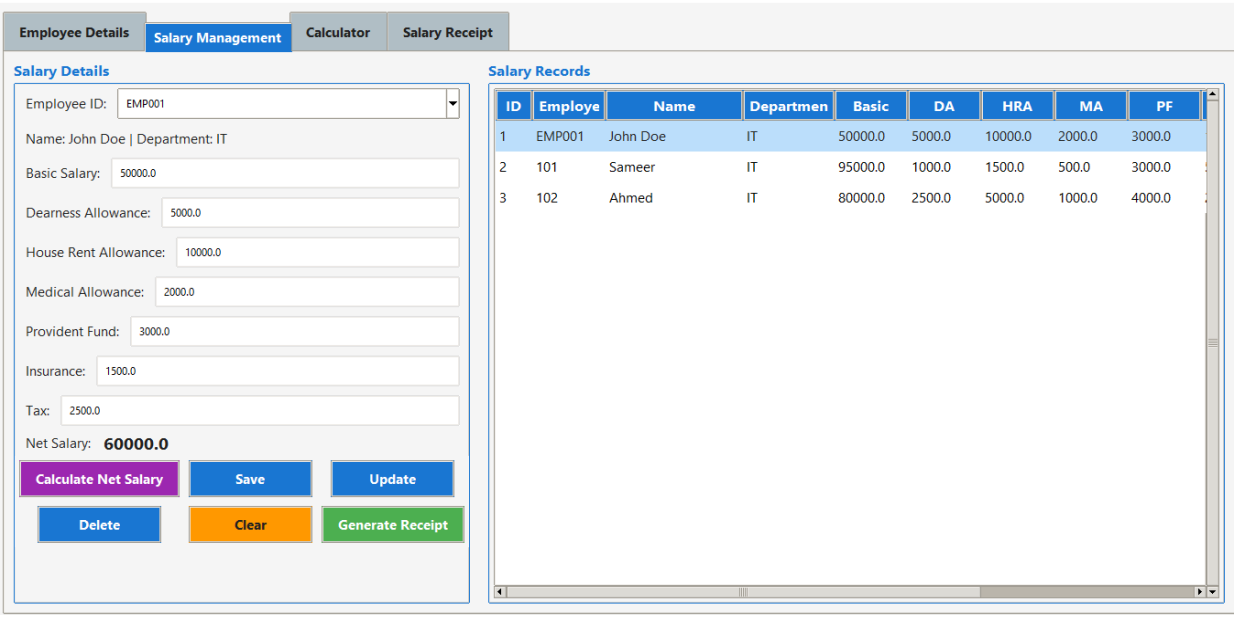


Figure 3.7: Salary Calculation Form

This module enables admins to compute salaries using allowances and deductions such as DA (Dearness Allowance), HRA (House Rent Allowance), MA (Medical Allowance), PF (Provident Fund), Insurance, and Tax. Once the components are entered, the system calculates the **net salary** and stores the result in the database. Buttons are provided to **Save**, **Update**, and **Delete** salary records.

Receipt Generation Screen

This screen provides the functionality to generate **salary receipts** for employees. By integrating the **ReportLab** library, the application can output receipts in both **PDF** and **TXT** formats. The receipts display a detailed salary breakdown and include all necessary identifiers such as employee ID, department, date, and net salary. Each generated receipt can be previewed or saved locally.

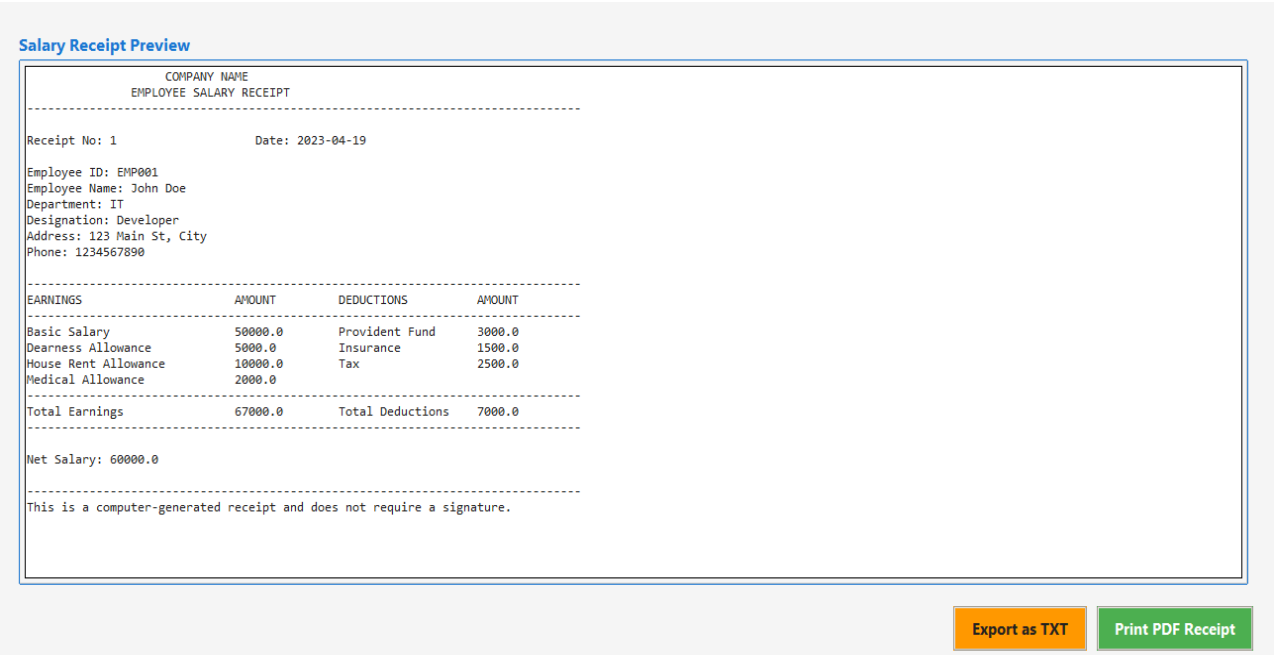


Figure 3.8: Payslip Receipt Preview

Calculator

A built-in calculator enhances the system's usability by offering real-time arithmetic operations during payroll computation. Built using Tkinter and styled using ttkbootstrap, the calculator performs basic mathematical functions and is helpful for quick on-the-spot calculations without leaving the application interface.

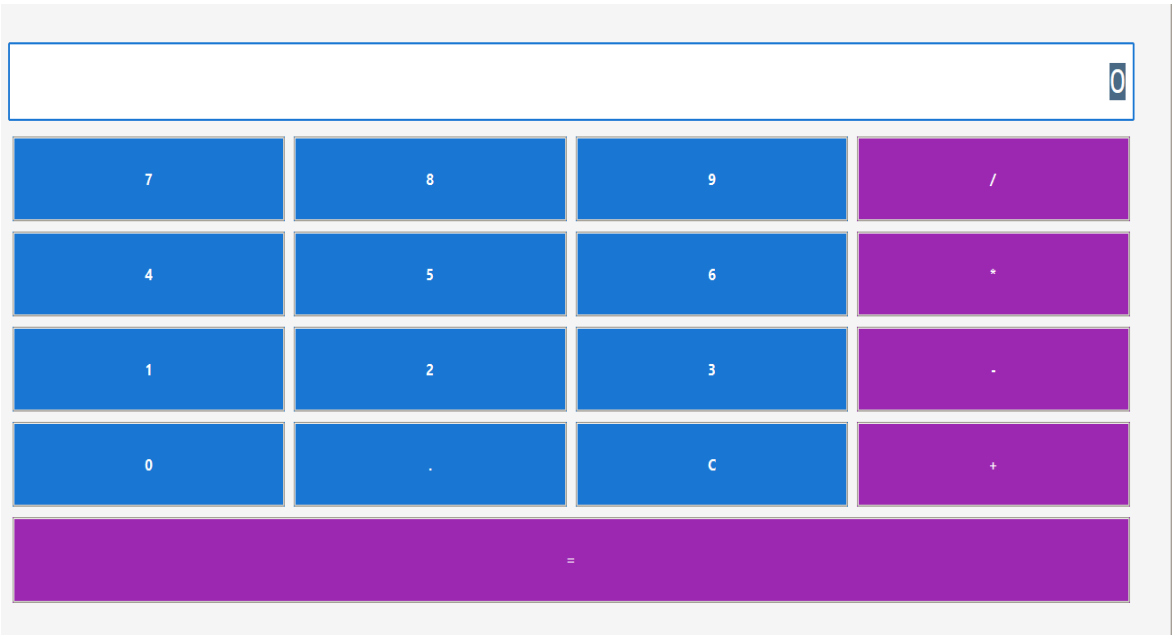


Figure 3.9: Build-in Calculator Screenshot

Chapter 4: Implementation

4.1 Module Descriptions

The Payroll Management System is organized into multiple functional modules, each responsible for a specific aspect of the application. These modules include login authentication, employee record management, salary computation, receipt generation, a calculator utility, and UI styling. Together, they work seamlessly to ensure accurate payroll processing, secure access, and an intuitive user experience.

4.1.1 Login Module (login.py)

The login module serves as the entry point to the system and is critical for maintaining data security. Upon startup, users are prompted to enter their username and password. The system cross-references these credentials with stored records in the admins and employees tables. Passwords are never stored in plain text; instead, they are hashed using SHA-256, ensuring secure authentication.

The login module intelligently differentiates between Admins and Employees. Admin users are granted full system privileges, such as managing employee records, calculating salaries, and generating payslips. In contrast, Employees have restricted access, allowing them to view only their own salary records and receipts. This role-based access is implemented to maintain data privacy and ensure that sensitive payroll information is handled by authorized personnel only. Furthermore, the module provides informative error messages for invalid credentials, enhancing user experience.

4.1.2 Employee Management (database.py)

This module is the backbone of the system's record-keeping functionality. Admin users can perform CRUD (Create, Read, Update, Delete) operations on employee records. The system collects vital details like employee ID, name, email, phone number, department, designation, and date of joining. Each time a new employee is added, their password is securely hashed and stored in the employees table.

An intelligent search function allows admins to filter records based on various criteria such as ID, name, department, or designation. All records are displayed in a tabular format within the GUI, making it easy for users to view, sort, and manage employee data. The use of SQLite ensures that data is stored in a relational manner, allowing for efficient querying and future scalability.

Error handling is built-in to prevent duplicate entries, invalid email formats, or incorrect phone numbers, thus maintaining data integrity throughout.

4.1.3 Payroll Calculation (main.py)

At the core of this system is the salary computation module. Admin users input values for basic salary, allowances (DA, HRA, MA), and deductions (PF, insurance, tax). The system automatically performs arithmetic operations to calculate the net salary using the formula: (Net Salary = Basic + DA + HRA + MA - PF - Insurance – Tax) This logic ensures that salaries are computed accurately every time, reducing the risk of human errors. Once calculated, the salary record is stored in the salaries table, linked to the respective employee by their emp_id.

This module also offers options to update or delete salary records as needed, and automatically timestamps each salary entry with the current date, providing a clear historical trail of salary payments.

4.1.4 Receipt Generation (receipt.py)

The receipt generation module leverages the powerful **ReportLab** library to produce professional-grade salary slips. These receipts summarize the employee's salary details, including basic salary, allowances, deductions, and net pay. Admin users can generate these slips in both **PDF** and **TXT** formats.

The PDF generator uses ReportLab’s canvas system to design structured, readable payslips with headers, tables, and formatted text. It even dynamically inserts the employee’s name, ID, department, and salary components into the document. The system ensures that each receipt is saved with a unique filename based on the employee ID and date, preventing file overwriting.

Additionally, employees can view and download their own salary receipts, making this module highly functional and practical for real-world payroll operations.

4.1.5 Calculator Utility (calculator.py)

This module adds a handy feature to the system—a built-in calculator. Developed using Tkinter, this tool offers basic arithmetic functions that aid admins during salary calculations. It supports operations like addition, subtraction, multiplication, and division. The calculator is styled using ttkbootstrap themes, making it visually consistent with the rest of the application. This utility, while simple, enhances the system's usability by providing real-time computation capabilities within the payroll environment, eliminating the need to switch between different applications.

4.1.6 Styling (styles.py)

The system's modern look is achieved through the styles.py module, which applies ttkbootstrap themes to standard Tkinter widgets. This module defines custom styles for buttons, entries, tables, and tabs, giving the application a professional, polished appearance. Colors, padding, font sizes, and hover effects are meticulously configured to enhance user experience. Importantly, the styles are made platform-compatible by handling exceptions for macOS systems, ensuring that the application runs smoothly on both Windows and Linux environments without crashing.

4.2 Code Snippets (Key Logic Only)

```
net_salary = ( basic_salary + da + hra + ma ) - ( pf + insurance + tax )
```

da = Dearness Allowance

hra = House Rent Allowance

ma = Monthly Allowance

pf = Provident Fund

Chapter 5: Testing

5.1 Testing Approach

A comprehensive testing methodology was employed to ensure the correctness and robustness of the system. The testing process covered:

- **Unit Testing:** Each individual module, such as login, employee CRUD operations, salary computation, and receipt generation, was tested separately to confirm correct behavior in isolation.
- **System Testing:** The entire system was tested as a whole to verify that modules interact correctly, ensuring smooth workflows from login to payslip generation.
- **User Acceptance Testing (UAT):** Test users (faculty and peers) were invited to interact with the application to provide feedback on usability and performance.

All modules were subjected to both valid and invalid inputs to test their ability to handle edge cases, errors, and user mistakes.

5.2 Sample Test Cases

Test Case	Input	Expected Output	Pass/Fail
Login with valid admin credentials	Username: admin, Password: 12345	Access granted, admin dashboard loads	Pass
Login with invalid password	Username: admin, Password: xyz	Access denied, error message shown	Pass
Add new employee	Emp ID: E001, Name: John Doe	Employee record saved and displayed	Pass
Generate salary receipt	Basic: 30000, DA: 5000, Deductions: 2000	PDF receipt generated correctly	Pass

5.3 Screenshots (Placeholder)

Figure 5.1 Login form with successful login message

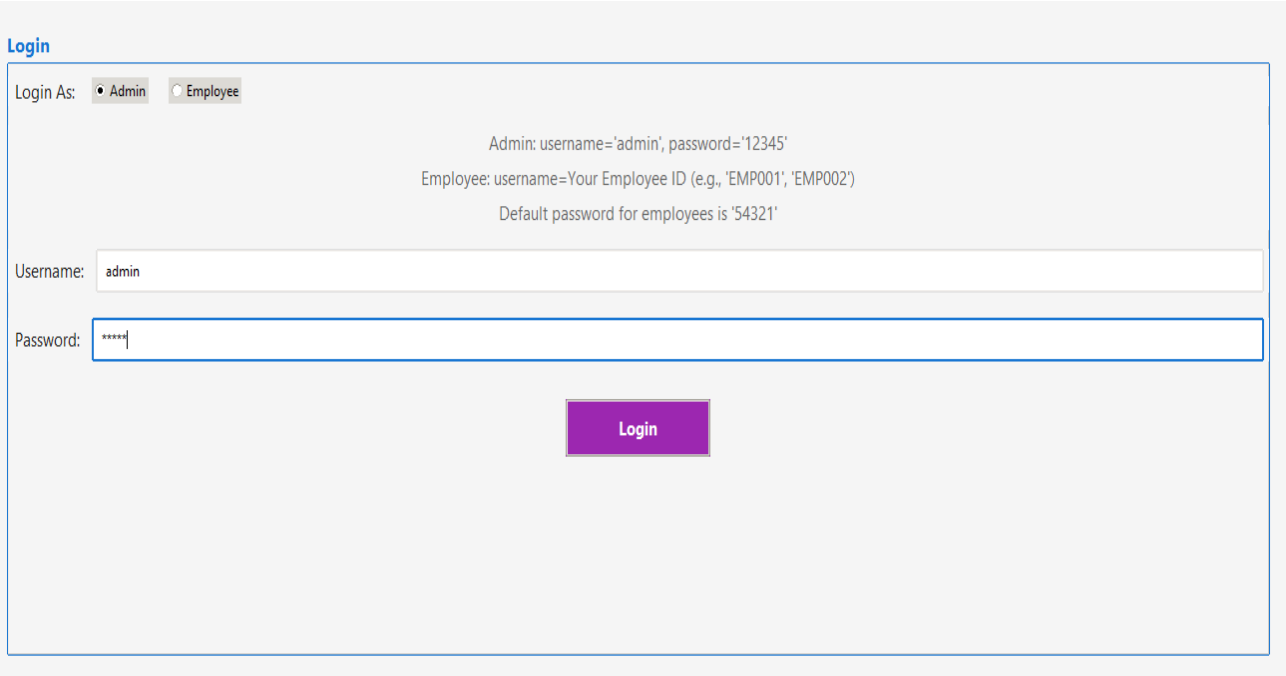


Figure 5.2 Employee record added and displayed in table

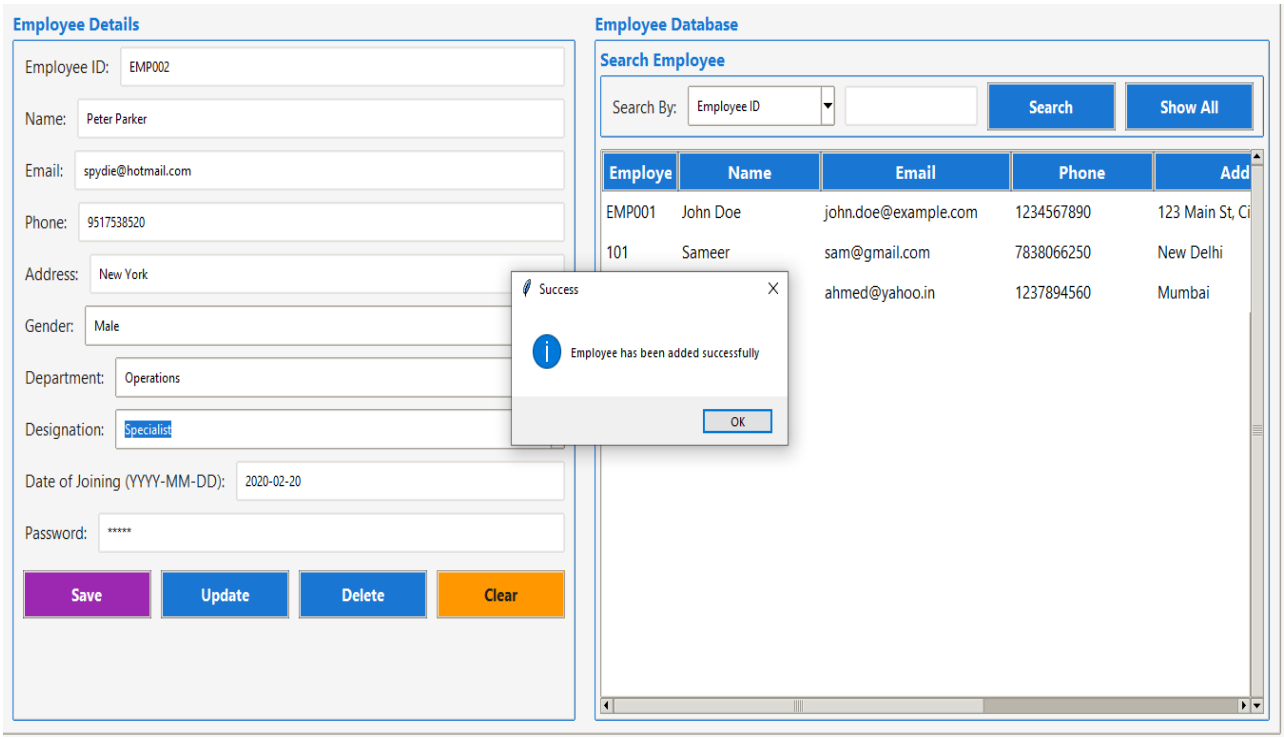


Figure 5.3 Salary form with net salary computed

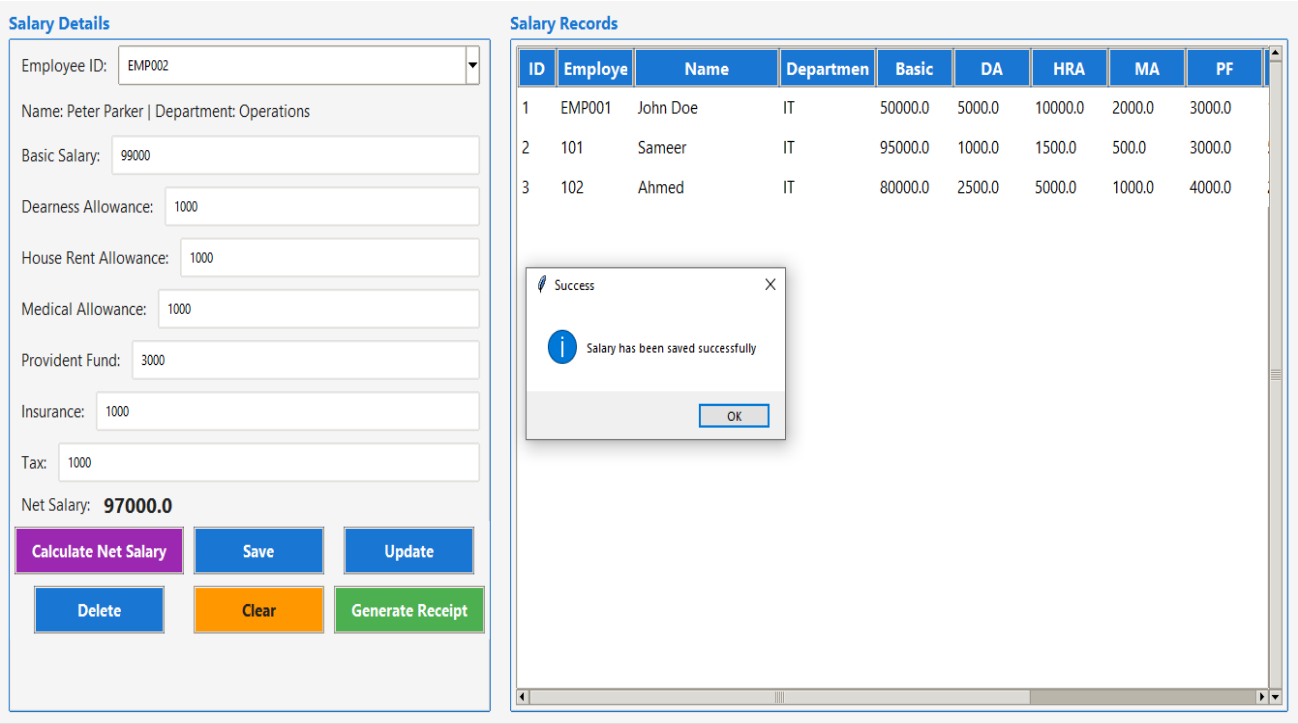


Figure 5.4 Payslip generated in PDF format

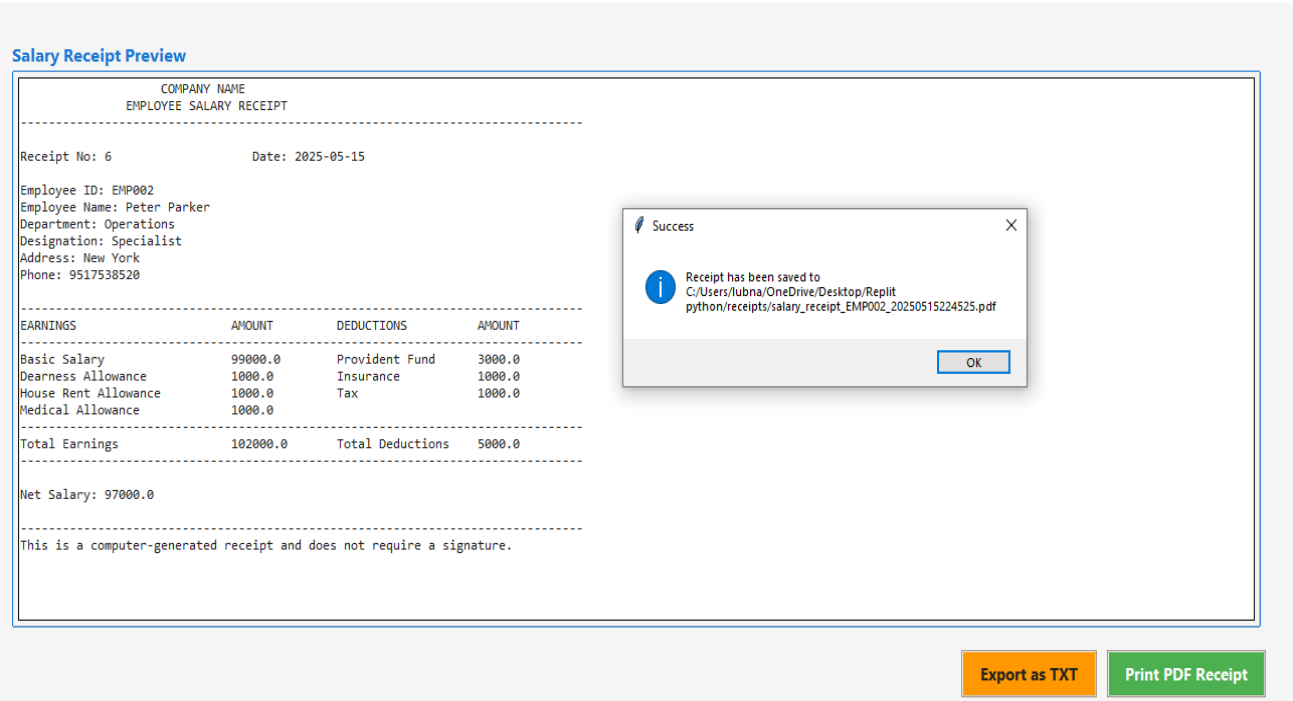


Figure 5.5 Built-in calculator performing operations

---XXX---

Chapter 6: Conclusion and Future Scope

6.1 Conclusion

The *Employee Payroll Management System* successfully fulfills its core objective of automating the payroll process while maintaining high standards of accuracy, efficiency, and usability. The system brings together critical functionalities—such as employee record management, salary computation, salary slip generation, and even a built-in calculator—into a unified platform that is both robust and user-friendly.

By leveraging open-source technologies like **Python**, **Tkinter**, and **SQLite**, the application maintains a lightweight footprint while still offering powerful capabilities. The modular architecture ensures ease of maintenance and future scalability. Furthermore, the implementation of **role-based login** enhances data security, ensuring that only authorized personnel have access to sensitive payroll data. The GUI, styled using **ttkbootstrap**, enhances user experience through a clean and modern interface.

One of the system's standout features is its ability to generate professional salary slips in both **PDF** and **TXT** formats using the **ReportLab** library. This not only provides documentation for employees but also supports HR in maintaining formal payroll records. The calculator utility, while simple, adds real-world usability to the system by allowing quick financial computations during payroll processing.

In summary, this project demonstrates the practical application of programming, database design, and user interface development to solve real-world problems. It showcases how a well-designed software solution can replace error-prone manual processes with an efficient, secure, and accurate digital alternative.

6.2 Future Scope

While the current system efficiently handles payroll requirements for small to medium organizations, there is considerable scope for future enhancements to broaden its utility and reach:

- **Role-Based Access Control (RBAC):** Introduce more granular roles such as HR Manager, Finance Officer, and Employee, each with customized permissions and access levels.
- **Online/Web Version:** Migrate the application from a desktop environment to a web-based platform using technologies like Django, Flask, or Node.js, allowing access from any device with internet connectivity.
- **Tax Report Generation:** Add support for annual tax summaries, automated tax calculations, and generation of tax documents such as Form-16 in compliance with local regulations.
- **Employee Attendance Integration:** Link the system to attendance tracking modules or biometric systems so that salaries are calculated based on actual days worked, overtime, or leaves.
- **Database Migration:** Move from SQLite to **MySQL**, **PostgreSQL**, or a cloud-based solution like **Firebase** or **AWS RDS** to support concurrent users and high-volume data storage.
- **Mobile App Extension:** Develop a mobile version (Android/iOS) that allows employees to log in, view payslips, update personal details, and get notifications regarding salary status or tax filings.

By implementing these future enhancements, the Payroll Management System can evolve into a fully integrated **Human Resource Management (HRM)** platform, capable of supporting the broader administrative needs of larger organizations.

---XXX---

REFERENCES

- [1] Python Official Documentation — https://docs.python.org/3/

- [2] Tkinter GUI Programming Tutorials — https://tkdocs.com/tutorial/

-[3] SQLite3 Database Guide — https://www.sqlitetutorial.net/

- [4] ReportLab PDF Generation Docs — https://www.reportlab.com/docs/reportlab-userguide.pdf

-[5] ttkbootstrap Theme Docs — https://ttkbootstrap.readthedocs.io/en/latest/

- StackOverflow Python Discussions

- GeeksforGeeks Python GUI Tutorials