Employee Management System

Submitted By

Name	ID
Omar Khayyam	0242220005101911

LAB PROJECT REPORT

This Report Presented in Partial Fulfillment of the course: **CSE222**

OBJECT ORIENTED PROGRAMMING Lab

Computer Science and Engineering Department

DAFFODIL INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

December 17, 2024

DECLARATION

We hereby declare that this lab project has been done by us under the supervision of **Ms. Nasima Islam Bithi**, **Lecturer**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere as lab projects.

Submitted To:

Ms. Nasima Islam Bithi
Lecturer
Department of Computer Science and Engineering
Daffodil International University

Submitted by

Omar Khayyam
ID:0242220005101911
Dept.CSE,DIU

COURSE & PROGRAM OUTCOME

The following course have course outcomes as following:

Table 1: Course Outcome Statements

CO's	CO Statements
CO1	**Define** and **Relate** classes, objects, members of the class, and relationships among them needed for solving specific problems.
CO2	**Formulate** knowledge of object-oriented programming and Python in problem solving
CO3	**Analyze** Unified Modeling Language (UML) models to **Present** a specific problem
CO4	**Develop** solutions for real-world complex problems **applying** OOP concepts while evaluating their effectiveness based on industry standards.

Table 2: Mapping of CO, PO, Blooms, KP and CEP

CO	PO	Blooms	KP	CEP
CO1	PO1	C1, C2	KP3	EP1,EP3
CO2	PO2	C2	KP3	EP1, EP3
CO3	PO3	C4, A1	KP3	EP1, EP2
CO4	PO3	C3, C6, A3, P3	KP4	EP1, EP3

The mapping justification of this table is provided in section **4.3.1**, **4.3.2** and **4.3.3**

TABLE OF CONTENT

Chapter 1

Introduction

This chapter introduces the project, explaining the background, motivation, objectives, and the scope of work. It also highlights the feasibility of the project.

1.1 Introduction

This code is a comprehensive PYTHON program designed for managing Employee Management of employees in a company. It allows users to perform various operations such as adding new employees, searching for records by employee ID, removing existing records, and saving records. The program uses a class called employee to store information about each employee, including their ID, name, age, position, salary, and working years. The menu-driven interface makes it easy to navigate through different functionalities, ensuring efficient management of employee records. This project serves as a practical example of how data can be organized and manipulated using classes and file handling in Python, providing valuable insights into building robust and user-friendly applications.

1.2 Motivation

This Employee management system project is a fantastic way to apply our knowledge of Python programming and practice real-world object oriented skills. By working on this project, we gain hands-on experience with classes, file handling, and user interaction, which are essential concepts in software development. It's an opportunity to see how theoretical concepts translate into practical applications, providing valuable insights into developing efficient and user-friendly systems. The project enhances our object oriented skills and encourages clean, modular coding practices. This hands-on project is a great way to demonstrate our ability to build useful, interactive applications. Moreover, completing this project boosts our confidence and prepare ourselves for more complex challenges in the future.

1.3 Objectives

The primary objectives of this project are-

1 To improve coding practices, logical thinking, and object oriented abilities through practical application.
2 To gain insights into how Employee Management Systems operate, which can be applicable to larger software projects or systems.
3 To manage potential errors, such as salary deduction of employees or issues with file operations.
4 To understand and apply basic concepts of Python.
5 To design a user interface for easy navigation and interaction with the program.

1.4 Feasibility Study

1. Technical Feasibility:

- Language: Uses-python, a high level general-purpose programming language.
- Resources: Requires a pycharm compiler (e.g., GCC) and minimal hardware.
- Complexity: Involves basic to intermediate concepts, making it manageable for students.

2. Operational Feasibility:
   - Interface: User-friendly interface.
   - Training: Basic knowledge of command-line operations required.
   - Maintenance: Modular design, making it easy to update and maintain.

3. Economic Feasibility:
   - Development Cost: Minimal, leveraging free tools and resources.
   - Benefits: Enhances learning by applying programming concepts in a practical project.

1.5 Gap Analysis

1.Functional Gaps:
   - Deletion of Records: Use functionality to remove employees.
   - Validation: Lacks input validation to check for invalid or duplicate data entries.
   - Error Handling: Basic error handling, but could be improved for robustness, such as handling file I/O errors more gracefully.

2. User Experience:
   - User Interface: Text-based interface is simple but could be enhanced with a graphical interface for better usability.
   - Feedback Messages: Could include more detailed feedback messages to guide the user through the process.

Addressing these gaps would enhance the functionality, user experience of the employee management system DIU's academic structure.

1.6 Project Outcome

1. **Hands-on Experience:** Gain practical skills in python programming, particularly in using class, object, inheritance, polymorphism, encapsulation, list, dictionary, numpy, abstraction etc.
2. **Functional Application:** Develop a functional employee management system that can add, display, search, and update employee records.
3. **Enhanced Object Oriented**: Improve object oriented and debugging skills through the development and testing phases.
4. **User Interaction:** Create a user-friendly interface that facilitates easy interaction with the application.
5. **Data Management:** Learn to efficiently manage and store data, including saving to and loading from files.
6. **Error Handling:** Implement and understand basic error handling techniques to manage unexpected inputs and file operatio

Chapter 2

Proposed Methodology/Architecture

This chapter outlines the system design, architecture, and project plan.

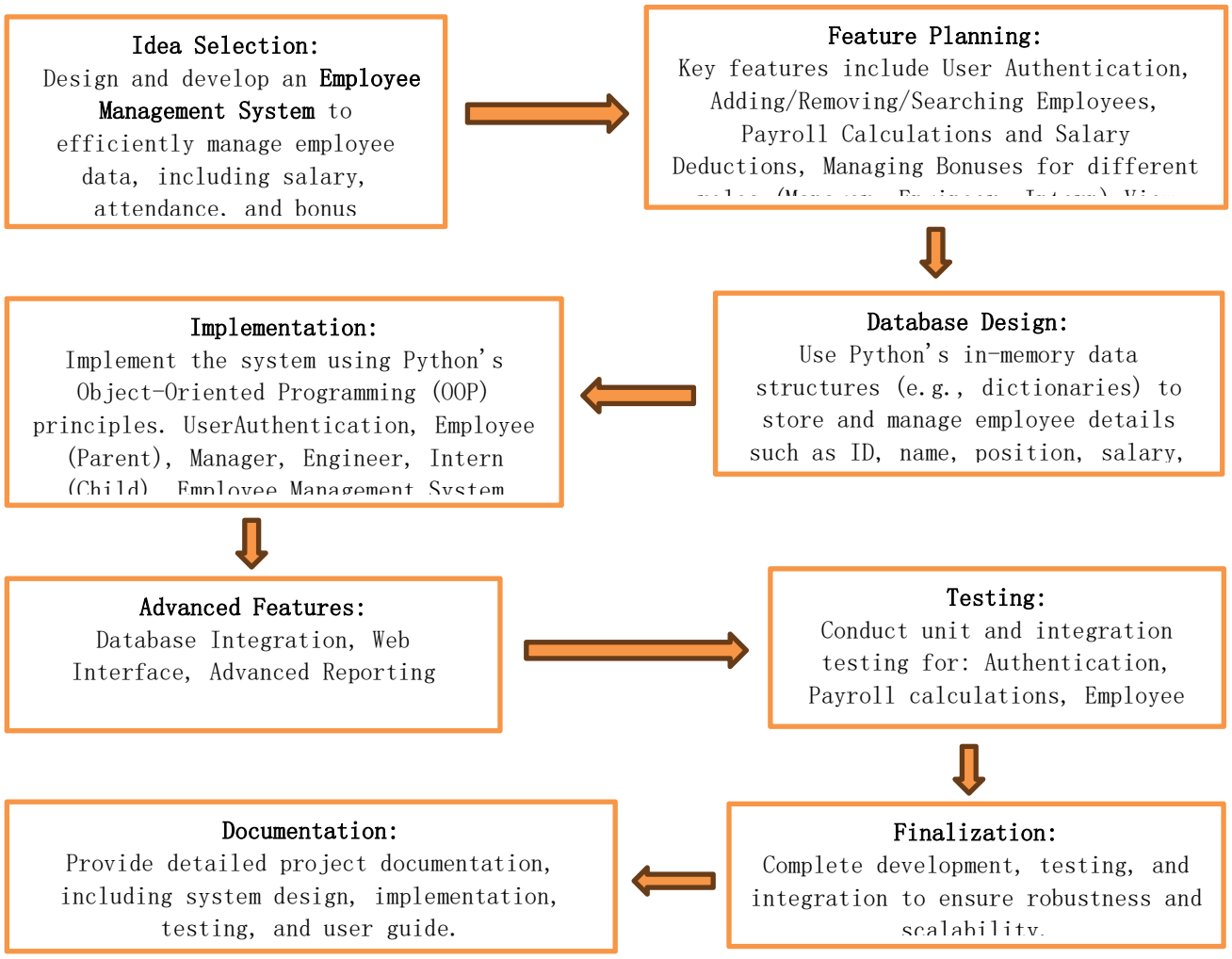
2.1 Requirement Analysis & Design Specification

The Employee Management System allows users to manage employee records efficiently. It supports adding, viewing, updating, and searching records by ID. A user interface simplifies user interaction.

2.1.1 Overview

This Employee Management System is a python program designed to manage employee records. It allows users to add, view, update, and search records by employee ID . Using a user friendly interface. It provides an efficient solution for maintaining employee records in a structured manner.

2.1.2 Proposed Methodology/ System Design



2.1.3 UI Design

Employee Management System
UML Diagram

2.2 Overall Project Plan

This employee Management System is an object oriented Python program that allows users to manage employee management. It supports adding, displaying, searching, and updating records stored. The program ensures ease of use through a user-friendly interface and robust error handling. It is designed for scalability and efficient management of employee data in small organizations.

Chapter 3

Implementation and Results

This chapter presents the implementation details of the Employee management Management System, evaluates the system's performance, and discusses the results obtained during the testing phase.

3.1 Implementation

Functionalities Implemented

1. User Authentication System

Features:

- **Signup**: Allows new users to register by creating a User ID and password.
- **Login**: Validates existing user credentials to grant access.

Implementation Details:

- The user authentication class stores user credentials in a dictionary (users).
- Passwords are matched during login to ensure security.

The core functionality revolves around managing employee records and operations.

2. Employee Management

Classes:

- **Employee** (Base Class): Represents generic employee attributes and methods.
- **Manager, Engineer, Intern**: Subclasses of Employee with specific salary structures and bonus calculation methods.

Features:

- **Add Employee**: Adds new employees to the system with role-specific details.
- **Remove Employee**: Deletes an employee record based on Employee ID.
- **Search Employee by ID**: Retrieves details, working years, and future salary for an employee.
- **Search by Position**: Lists employees based on their role.
- **View All Employees**: Displays all registered employees and their details.
- **Apply Yearly Increment**: Increases salaries by 10% annually.
- **Salary Deduction for Absence**: Deducts salary proportionally for absent days.

3. Payroll and Bonus Management

Features:

- **Bonus Calculation**:
  - Manager: 20% of the base salary.
  - Engineer: 15% of the base salary.
  - Intern: Flat bonus of $500.
- **Payroll Summary**:
  - Calculates total and average salaries of employees.

Implementation:

- Leveraged NumPy for efficient payroll calculations (np.sum and np.mean).
- Bonus calculations are overridden in the subclasses to reflect role-specific policies.

4. Employee Search and Recent Hires

- Provides functionality to search employees by ID or position.
- Lists the most recently added employees, up to a user-defined count.

Code Structure and Design

Object-Oriented Approach

- **Encapsulation**: Classes encapsulate attributes (e.g., name, position, salary) and methods (e.g., calculate_bonus, yearly_increment).
- **Inheritance**: Subclasses (Manager, Engineer, Intern) inherit common properties and methods from the Employee class, reducing redundancy.
- **Polymorphism**: Bonus calculation is implemented differently in each subclass, demonstrating polymorphism.

Modular Design

The system is modular, allowing for future extensions such as:

- Adding new employee roles with unique features.
- Expanding user authentication to include roles.

Error Handling and Validation

The system incorporates error-handling mechanisms to ensure robustness:

- **Validation**:
  - Matching passwords during signup.
  - Checking for duplicate Employee IDs during registration.
  - Validating absent days (e.g., ensuring non-negative values).
- **Exception Handling**: Prevents runtime crashes through try-except blocks to manage unexpected errors.

Testing

Test Scenarios:

1. **Authentication**:
    - Verify signup and login functionality with valid and invalid credentials.
2. **Employee Operations**:
    - Test adding, removing, and searching employees with various inputs.
3. **Salary and Bonus Calculations**:
    - Validate bonus and salary deduction computations.
4. **Payroll Summary**:
    - Confirm total and average payroll calculations for diverse employee sets.

Tools and Libraries

1. **Programming Language**: Python
2. **Libraries**:
    - NumPy: For payroll calculations.
    - datetime: To calculate working years based on the start year.

3.2 Performance Analysis

Data Structures:

- The employees dictionary ensures O(1) complexity for add, remove, and search operations by ID.
- Iterative operations like payroll calculation (O(n)) are reasonable for the expected scale.

Scalability:

- The system is limited by memory since all data is stored in dictionaries. For large datasets, the lack of persistent storage or indexing mechanisms could hinder performance.

Error Handling:

- Custom error handling with try-except blocks improves program robustness.
- However, error messages could provide more detailed guidance for users.

3.3 Results and Discussion

Results

1. **System Functionality**:
    a. The project successfully implements core features of employee management:
        - Adding, removing, and searching employees.
        - Calculating payroll, bonus, and yearly increments.
        - Deducting salary for absenteeism.
    b. User authentication works as intended, ensuring only authorized users can access the system.
2. **Performance**:
    a. **Efficiency**:
        - Key operations like searching, adding, and removing employees perform efficiently using a dictionary for storage.

b. **Scalability**:
    - The current implementation handles small datasets effectively, but lacks persistent storage, limiting scalability for larger applications.

3. **Accuracy**:
    a. Correct calculations for bonuses, payroll, and yearly increments were observed during testing.
    b. Employee-specific details (e.g., working years, position-based bonuses) are computed accurately.
4. **Error Handling**:
    a. The system provides basic error handling for common issues, such as duplicate employee IDs and invalid user inputs.

Discussion

1. **Strengths**:
    - **Modular Design**:
        - The use of distinct classes (e.g., Employee, Manager, Intern) enhances code reusability and maintainability.
    - **Customizability**:
        - The system is flexible for extending functionality, such as adding new employee roles or features.
    - **Ease of Use**:
        - A simple menu-driven interface guides users effectively.
2. **Challenges**:
    - **Data Persistence**:
        - The lack of storage mechanisms results in data loss upon program termination, limiting real-world applicability.
    - **Input Validation**:
        - Insufficient validation for inputs like age, emp_id, and absent_days caused runtime exceptions during edge cases.
3. **Performance Bottlenecks**:
    - As the number of employees increases, iterative operations (e.g., payroll calculation) could become time-consuming without optimization or threading.
4. **Usability**:
    - While functional, the text-based interface lacks modern design features, making it less appealing for end-users in a real-world setting.
5. **Potential Issues**:
    - **Hardcoded Salaries**:
        - Fixed salary values for roles limit flexibility. Implementing a salary management system would enhance customization.
    - **Security Concerns**:
        - User credentials are stored in memory without encryption, posing security risks.

Chapter 4

Engineering Standards and Mapping

This chapter maps the project implementation to engineering standards, program outcomes, and complex problem-solving categories. It also evaluates the societal, environmental, and ethical impacts of the IoT-Based Smart Classroom and highlights sustainability and project management aspects.

4.1 Impact on Society, Environment and Sustainability

A Employee Management System (EMS) enhances efficiency, accuracy, and fairness in employee compensation, promoting transparency and trust. It reduces environmental impact by minimizing paper use and lowering carbon emissions. The system also supports sustainability by optimizing resources and driving cost savings. Additionally, it encourages ethical practices and ensures compliance with labor laws and data privacy standards.

4.1.1 Impact on Life

The employee Management System significantly improves organizational efficiency by automating processes, reducing manual errors, and saving time. This enhances employee satisfaction and ensures accurate and timely salary disbursement, positively impacting workplace productivity.

4.1.2 Impact on Society & Environment

Society:
PRMS promotes fair compensation, transparency, and trust in workplaces, fostering social equity. It ensures legal compliance and financial security, benefiting both employees and employers.

Environment:
By reducing paper use and streamlining processes, PRMS lowers waste and energy consumption.

The system's digital nature supports sustainable practices, reducing carbon footprints and operational costs.

4.1.3 Ethical Aspects

Ethically, an employee Management System ensures fair compensation, transparency, and compliance with labor laws, promoting equal treatment for all employees. It safeguards sensitive financial data, adhering to privacy regulations to protect employee information. Additionally, it supports accountability by reducing errors and fraud in processes, fostering trust between employers and employees

4.1.4 Sustainability Plan

- Paperless Operations: Transition to a fully digital payroll system to eliminate paper usage, reducing waste and supporting a sustainable, eco-friendly work environment.

- Energy Efficiency: Utilize cloud-based payroll systems hosted on energy-efficient data centers, minimizing carbon footprints and optimizing resource consumption.

- Long-term Cost Savings: Automate payroll processes to reduce administrative costs, enabling reinvestment in sustainable business practices, employee welfare, and green initiatives.

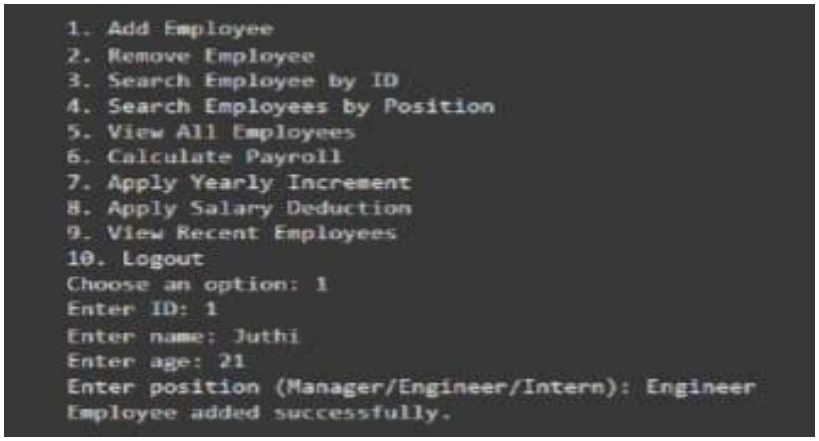
Project Management

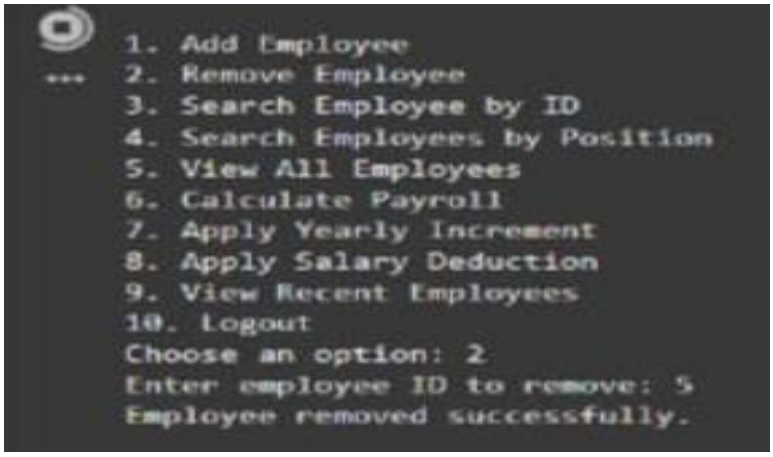
The project was managed through clear task distribution, where each team member handled specific aspects like system design, coding, testing, and documentation. Resources were utilized efficiently, with tools like GitHub for version control and communication platforms for collaboration. Progress was monitored through regular meetings, milestone tracking, and updates to ensure timely completion and alignment with project goals. This approach facilitated smooth teamwork and successful project delivery.

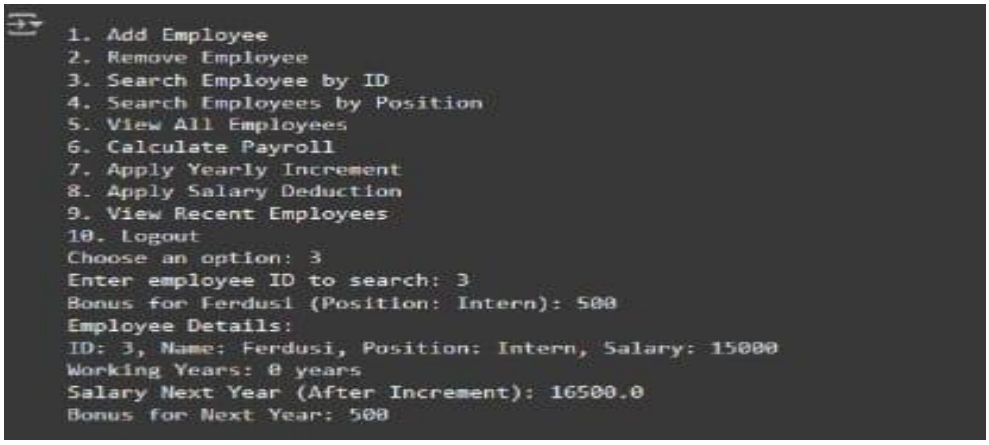
Omar khayyyam

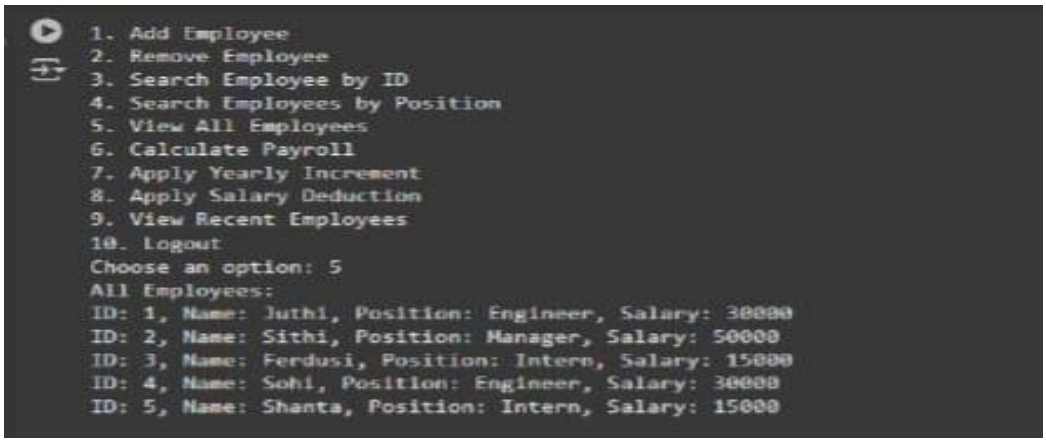
Id:0242220005101911

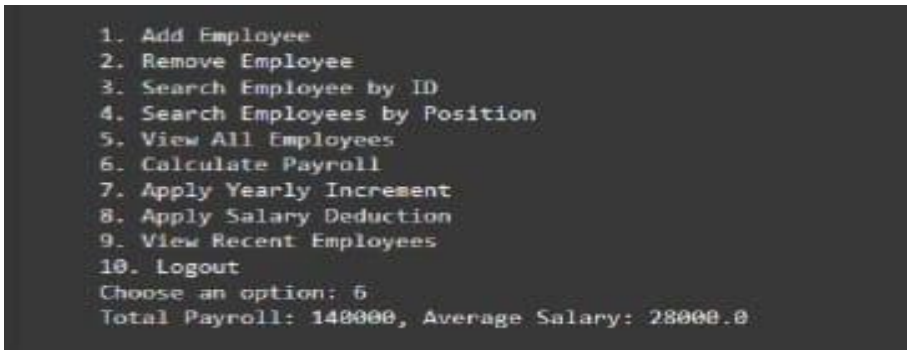
- **Developed core functionalities of the Employee Management System, including adding and removing employees, ensuring accurate and efficient employee management.**
- **Added viewing options for all employees and recent employees, allowing users to access detailed and filtered employee records efficiently.**
- **Created the project report.**

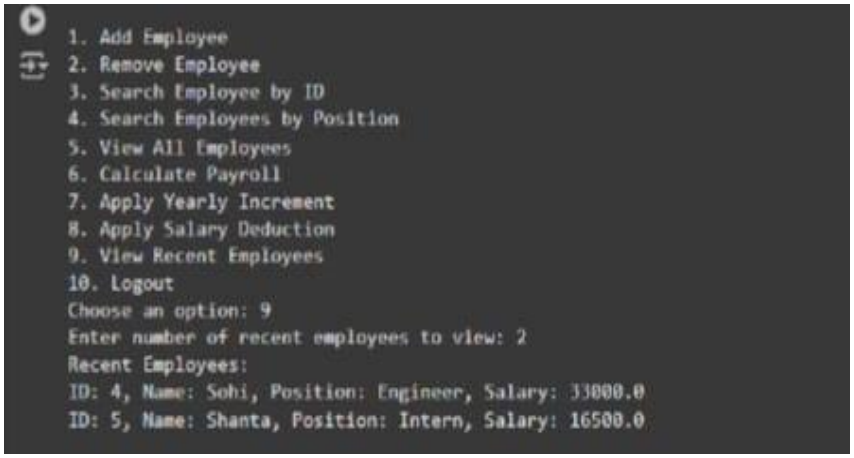












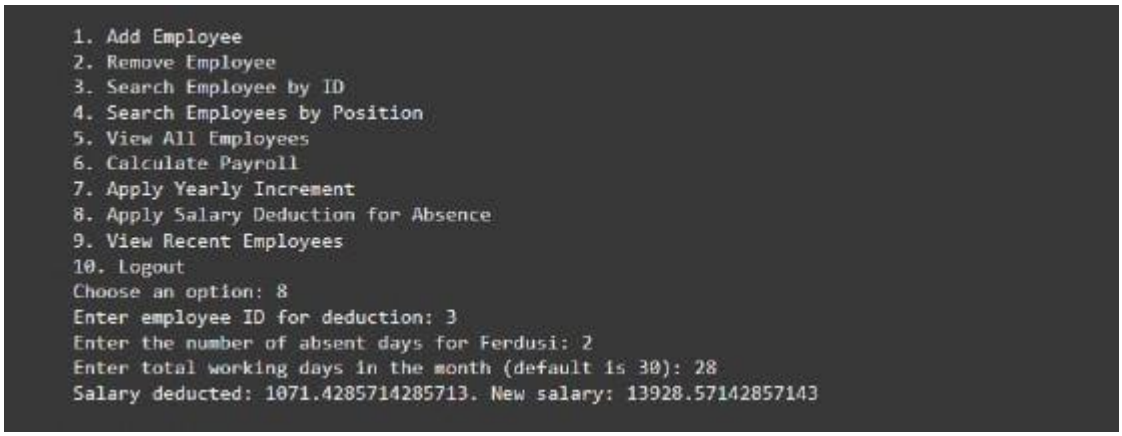
- **Developed user management features such as signup and login.**
- **Implemented search functionality to search employees by position.**
- **Handled salary deductions for absences with proper calculations.**
- **Developed payroll calculation features, including yearly increments and salary deductions.**
- **Worked on logout features to ensure secure system exits.**
- **Created the project report.**

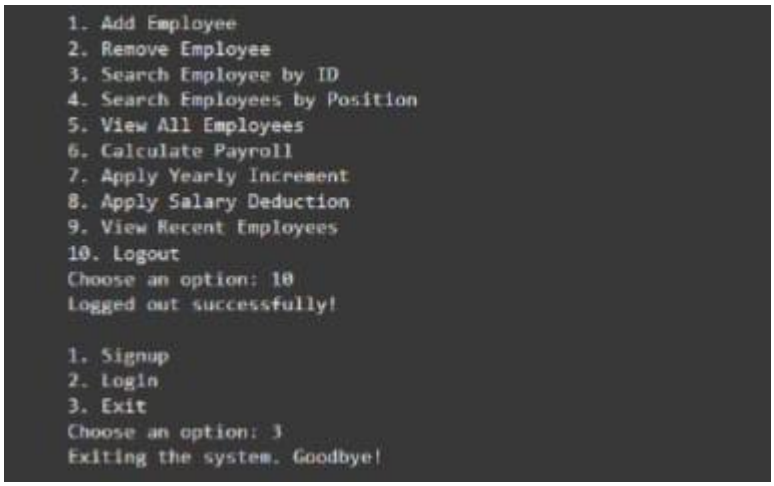
```
Welcome to the Employee Management System!

1. Signup
2. Login
3. Exit
Choose an option: 1
Enter User ID: 111
Enter Password: jj
Confirm Password: jj
Signup successful! You can now log in.
```

```
1. Signup
2. Login
3. Exit
Choose an option: 2
Enter User ID: 111
Enter Password: jj
Login successful! Welcome!
```

```
1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position
5. View All Employees
6. Calculate Payroll
7. Apply Yearly Increment
8. Apply Salary Deduction
9. View Recent Employees
10. Logout
Choose an option: 4
Enter position to search (Manager/Engineer/Intern): Engineer
Employees in Engineer position (Salary: 30000):
Juthi (ID: 1)
Sohi (ID: 4)
```

```
1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position
5. View All Employees
6. Calculate Payroll
7. Apply Yearly Increment
8. Apply Salary Deduction
9. View Recent Employees
10. Logout
Choose an option: 7
Yearly increment applied to all employees.
```





(Exception handling)

- **It identifies and handles invalid operations, such as duplicate employee IDs, invalid login attempts, or trying to update a non-existent employee record.**
- **It improves the user experience by providing clear error messages instead of abrupt terminations or crashes.**

4.2 Complex Engineering Problem

The employee Management System solves complex engineering problems by ensuring efficient data handling, secure storage, and quick retrieval of employees using structured file operations. It addresses scalability, accuracy, and automation challenges in managing l for diverse organizational needs.

4.2.1 Mapping of Program Outcome

The program aligns with outcomes like **PO1 (Engineering Knowledge)** by applying structured programming for employee management and **PO2 (Problem Analysis)** by solving data storage, retrieval, and processing challenges efficiently.

Table 4.1: Justification of Program Outcomes

PO's	**Justification**
PO1	Demonstrate a comprehensive understanding of fundamental database management concepts, including the relational data model, normalization techniques, and SQL basics Design, implement and optimize relational databases, incorporating advanced SQL queries, indexing techniques and query optimization strategies.
PO2	Design, implement and optimize relational databases, incorporating advanced SQL queries, indexing techniques and query optimization strategies.
PO3	Design and development: Design solutions for complex engineering problems and design system components or processes that meet specified needs.

4.2.2 Complex Problem Solving

The Employee Management System addresses complex problem-solving by automating the management, storage, and retrieval of payroll data, ensuring accuracy, scalability, and efficient processing.

EP1 **Dept of Knowledge**	The project uses core knowledge of programming (python), data structures , and file handling to solve employee management problems.
EP2 **Range of Conflicting Requirements**	Balancing between a user-friendly text interface and the need for complex data storage, retrieval, and updates through file operations.
EP3 **Depth of Analysis**	The project deeply analyzes the process of employee-management, addressing challenges such as accurate data entry, search functionality, and data persistence.
EP4 **Familiarity of Issues**	The system is designed to address common issues in employee management, such as manual record-keeping errors, scalability, and data retrieval.
EP5 **Extent of Applicable Codes**	The code applies to various employee management contexts, supporting scalability, secure file operations, and flexible data management.
EP6 **Extent Of Stakeholder Involvement**	Stakeholders like HR, administrators, and employees are considered in the design, ensuring the system meets practical and user requirements
EP7 **Inter- dependence**	The system integrates various modules (e.g., data entry, updates, search, file I/O) that depend on each other to work cohesively and efficiently.

4.3.3 Complex Engineering Activities:

EA1 **Range of resources**	The system uses a variety of resources including Python programming, file handling, and user input/output, along with system memory to handle employee data efficiently.
EA2 **Level of Interaction**	The system allows interaction through a simple command-line interface, enabling users to add, display, search, and update payroll records with real-time processing.
EA3 **Innovation**	The project applies a modular approach to solving the employee management problem, combining core programming techniques with file I/O to persist data efficiently.
EA4 **Consequences for society and environment**	The system helps organizations streamline processing, reducing manual errors and saving time, indirectly improving employee satisfaction and organizational efficiency.
EA5 **Familiarity**	The system uses familiar programming concepts like structs, file handling, and basic input/output that are commonly used in data management and Employee management systems.

Chapter 5

Conclusion

The Employee Management System efficiently handles employee payroll operations with features like adding, searching, updating, and persisting records. It demonstrates a robust and scalable solution for managing data in a structured and user-friendly manner.

5.1 Summary

This Employee Management System is a python application designed to manage employee records efficiently. The system supports adding, searching, updating, and displaying employee data while ensuring secure and persistent storage through binary file operations. It provides a user-friendly, menu-driven interface for seamless interaction. The program ensures accuracy in Employee management and offers solutions for key challenges such as data organization, retrieval, and updating. With its modular design and robust features, the system is a practical tool for small to medium-sized organizations to streamline their processes.

5.2 Limitation

1. **Password Storage**:
    - Passwords are stored in plain text, which poses a security risk.
2. **Static Salary Structure**:
    - Role-based salaries are hardcoded, limiting flexibility.

5.3 Future Work

1. **Enhance Data Persistence**:
    - Integrate persistent storage using a database (e.g., SQLite) or file storage (e.g., JSON/CSV).
2. **Improve Security**:
    - Implement encryption for user credentials.
    - Add password recovery features.
3. **Optimize Performance**:
    - Use threading or asynchronous operations for scalability in handling large datasets.
4. **Upgrade User Interface**:
    - Develop a graphical user interface (GUI) using frameworks like tkinter or a web interface using Flask.
5. **Enhance Error Handling**:

- Implement comprehensive input validation and exception handling.

6. **Expand Functionality**:
   - Add new features such as leave management, employee performance tracking, and department-level reporting.

References

- Lutz, M. (2009). *Learning Python*. O'Reilly Media, Inc.
- Python Documentation: https://docs.python.org/3/