REPORT

EMPLOYEE MANAGEMENT SYSTEM

PRESENTED BY

R190790

THULASI

EMPLOYEE MANAGEMENT SYSTEM

Abstract

The idea is that we perform different changes in our Employee Record by using different functions for example the Add Employee will insert a new row in our Employee, also, we will create a Remove Employee Function which will delete the record of any particular existing employee in our Employee table. This System works on the concepts of taking the information from the database making required changes in the fetched data and applying the changes in the record .

We also have the information about all the existing employees.

The main advantage of connecting our program to the database is that the information becomes lossless even after closing our program a number of times.

Hey there in this project first of all ill make two file in python first one file for the front end named as gui.py , and second one file is for the backend of the our project name Database.py

Then first working on the building front for the project and than we start making frame as like format – main frame{top frame{topleft{topright}} {downframe{downleftframe{downrightframe}}

Then we making widget section of the window – in this we are try to making a labels and entry fields of the gui front end . also creating global variable and local Variable .

Then making buttons on the gui window for adding data , delete , reset, exit, search operations.

Now we making Backend for maintain data from the front end buttons , we making first of all diff-diff function to make database operation – addEmployeerec - View data, - deleterec , - search data , - data update .

Now Fetching data from the backend file to front file . import Employeedatabase then importing all funcutions of the backend file . then making funcution like api to get data from backend funcutions into gui file funcution .

Then calling the all over functions in the button of the front end gui frame . and now making calculation for the - PayRef and – MonthlySalary funcutions to calculate -M_pay , MTax ,MPension , Deduct , Grosspay , NI_Payment ,

NetPayafter all the operation results we are counting for the RECEIPT on output GUI Frame List box showing .

EMPLOYEE MANAGEMENT SYSTEM

Abstract

The idea is that we perform different changes in our Employee Record by using different functions for example the Add Employee will insert a new row in our Employee, also, we will create a Remove Employee Function which will delete the record of any particular existing employee in our Employee table. This System works on the concepts of taking the information from the database making required changes in the fetched data and applying the changes in the record .

We also have the information about all the existing employees.

The main advantage of connecting our program to the database is that the information becomes lossless even after closing our program a number of times.

Hey there in this project first of all ill make two file in python first one file for the front end named as gui.py , and second one file is for the backend of the our project name Database.py

Then first working on the building front for the project and than we start making frame as like format – main frame{top frame{topleft{topright}} {downframe{downleftframe{downrightframe}}

Then we making widget section of the window – in this we are try to making a labels and entry fields of the gui front end . also creating global variable and local Variable .

Then making buttons on the gui window for adding data , delete , reset, exit, search operations.

Now we making Backend for maintain data from the front end buttons , we making first of all diff-diff function to make database operation – addEmployeerec - View data, - deleterec , - search data , - data update .

Now Fetching data from the backend file to front file . import Employeedatabase then importing all funcutions of the backend file . then making funcution like api to get data from backend funcutions into gui file funcution .

Then calling the all over functions in the button of the front end gui frame . and now making calculation for the - PayRef and – MonthlySalary funcutions to calculate -M_pay , MTax ,MPension , Deduct , Grosspay , NI_Payment ,

NetPayafter all the operation results we are counting for the RECEIPT on output GUI Frame List box showing .

1. INTRODUCTION

Employee Management Portal Project In Python is a comprehensive system designed to manage employee information and records.
It is a user-friendly system that allows users to easily store, update, and retrieve employee information.

The task is to create a Database-driven Employee Management System in Python that will store the information in the MySQL Database. The script will contain the following operations:
Add Employee
Remove Employee
Update Employee
Clear Employees
This System works on the concepts of taking the information from the database making required changes in the fetched data and applying the changes in the record.

1.1. Background

1. **Define requirements:** Identify organization needs and functionalities required in the Employee Management System.
2. **Database design:** Choose a suitable database (e.g., SQLite, MySQL) and design the schema for storing employee data.
3. **UI design:** Plan the layout using tools like Tkinter or frameworks like Flask/Django for web-based applications.

1.2. Objectives

1. **Efficiency:** Streamline and automate HR processes to improve overall efficiency in managing employee data,other related tasks.
2. **Accuracy:** Reduce errors and data inconsistencies by centralizing employee information in a structured database, minimizing manual data entry.
3. **Time Savings:** Save time for HR personnel and employees by automating routine tasks, allowing them to focus on more strategic and value-added activities.

4. **Cost Reduction:** Streamline HR processes and reduce manual paperwork, leading to cost savings in terms of time, resources, and potential errors.

1.3. Purpose

The purpose of an Employee Management System is to optimize HR processes, improve data accuracy, ensure compliance, empower employees, and contribute to strategic decision-making for the overall betterment of the organization. it helps achieve several specific objectives:

1. **Efficient Workforce Administration:** Streamline HR tasks for improved administrative efficiency.
2. **Accurate Employee Information:** Centralize data to ensure accuracy and reliability for HR functions.
3. **Compliance Management:** Automate processes to adhere to labor laws and policies, minimizing legal risks.
4. **Enhanced Decision-Making:** Provide management with reliable information for strategic decisions.
5. **Employee Engagement and Empowerment:** Improve satisfaction through self-service features and participation in HR processes.
6. **Cost Reduction:** Streamline processes, reduce paperwork, and minimize errors for cost savings.
7. **Strategic Workforce Planning:** Offer insights into workforce trends and skill gaps for long-term planning.
8. **Performance Management:** Facilitate employee performance tracking and development identification.
9. **Adaptability and Scalability:** Design a system to evolve with HR practices and accommodate organizational growth.

1.4. Scope

1. **Comprehensive employee data management.**
2. **Attendance tracking and leave management.**

3. **Payroll processing and financial transactions.**
4. **Performance evaluation and goal tracking.**
5. **Training, development, and certification management.**
6. **Recruitment support and seamless onboarding.**
7. **Employee self-service capabilities.**
8. **Compliance with legal requirements.**
9. **Reporting and analytics for workforce insights.**
10. **Communication, collaboration, and engagement features.**
11. **Succession planning and talent development.**
12. **Robust data security and privacy measures.**

1.5. Applicability

1. **Streamlining HR processes and administrative tasks.**
2. **Improving accuracy in employee data management.**
3. **Ensuring compliance with labor laws and regulations.**
4. **Enhancing decision-making through data insights.**
5. **Fostering employee engagement and self-service.**
6. **Managing payroll efficiently and accurately.**
7. **Supporting recruitment and onboarding processes.**
8. **Facilitating performance evaluation and training.**
9. **Enhancing communication and collaboration.**
10. **Providing a secure and mobile-accessible platform.**
11. **Adapting to organizational growth and changes.**
12. **Integrating with other systems for seamless operations.**

2. PROJECT PLAN

2.1. Problem Statement

Current manual methods of entering and managing employee data in our organization lead to inefficiencies, data inaccuracies, and compliance risks. The absence of a centralized system results in time-consuming administrative tasks, hindering timely decision-making and impacting overall workforce management. There is a need for an automated Employee Data Entry System to streamline processes, improve data accuracy, and ensure compliance with labor laws and internal policies.

2.2. Requirement Specification

1. **Database Design (SQLite):**

   Tables for employee information, attendance records, and leave applications.
   Appropriate relationships between tables for data consistency.
   Indexing for efficient data retrieval.

2. **Data Storage and Retrieval:**

   Implement CRUD operations for employee data.
   Store attendance and leave data in the SQLite database.
   Retrieve and display employee information dynamically in the UI.

3. PROPOSED SYSTEM AND METHODOLOGY

3.1. System Architecture

Programming Language: Python 3.6
Environment (Libraries and Technologies):
Presentation Layer: Tkinter GUI,
Data Access Layer: SQLite Database:

3.2. Implementation

In this project, the prescribed sequence is:

- Environment Setup:
  Install Python and Tkinter on your system.

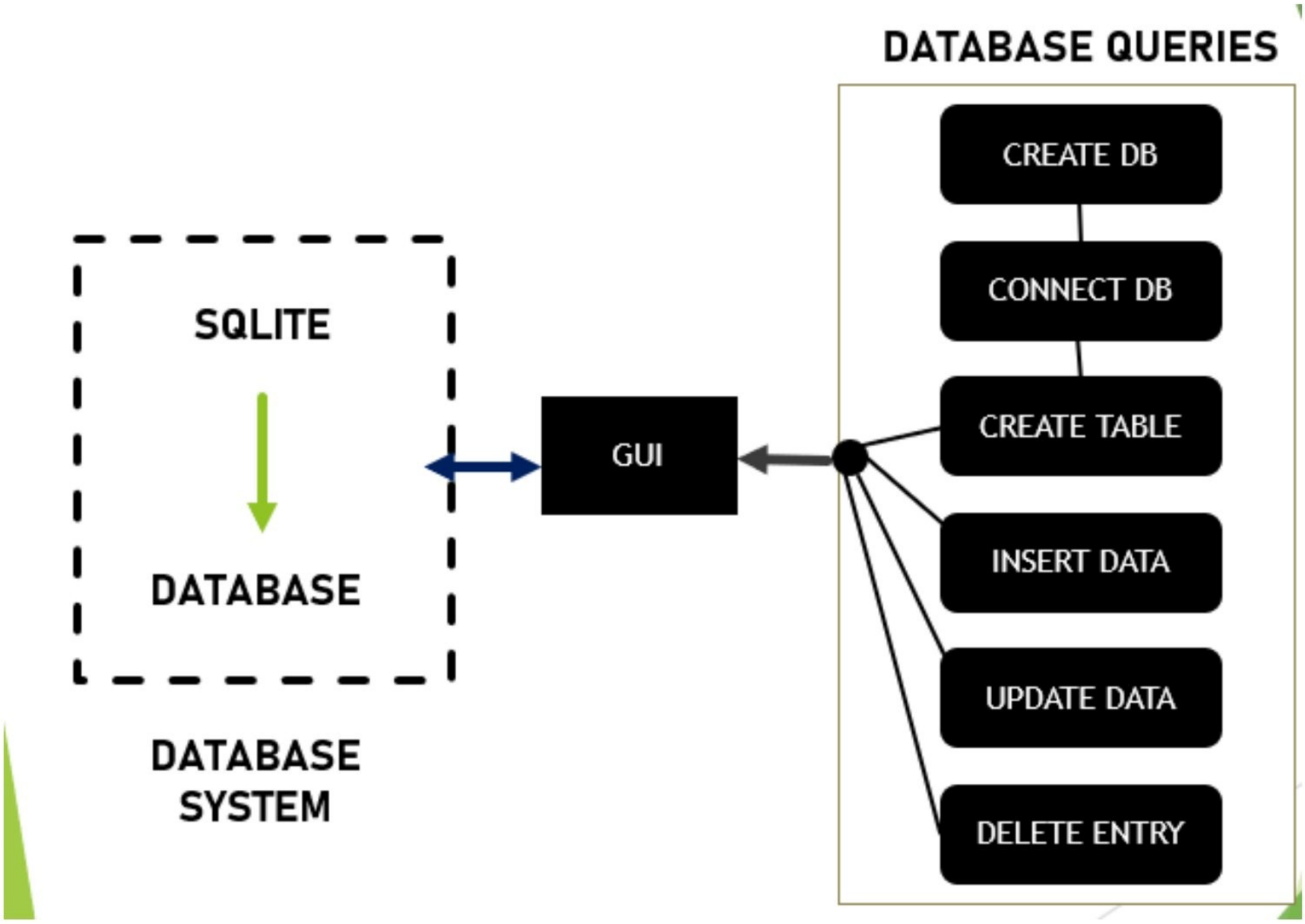
- Database Design:
  Design the SQLite database schema for employee information records,
  Create the necessary tables and establish relationships.

- Tkinter GUI Design:
  Create Tkinter window and frames for different forms (Employee data entry)
  Design and place widgets such as labels, entry fields, buttons, and on the forms.

- Controller and Business Logic:
  Implement a controller module to handle user inputs and events from the GUI.
  Develop business logic to manage CRUD operations for employee data, attendance, leave, and report generation.

- Data Access Layer (SQLite):
  Connect to the SQLite database using Python's SQLite module.
  Implement functions to perform database operations like insertion, retrieval, and updates.

3.2.1. Flow Chart Diagram



4. RESULTS AND EXPLANATION

4.1. Implementation Approaches

Step 1: Environment Setup

Make sure you have Python and Tkinter installed on your system. You can install the required modules using:

pip install tk

Step 2: Creating a database in SQLite “employee”

Step 3: Creating a Table

Step 4: Tkinter GUI Design

Step 5: Run the Application

Step 6: Document and Report

4.2. Pseudo Code

Step 1 : Importing essential libraries

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from db import Database
import sqlite3
```

Step 2: Database connection:

```
class Database:
def init(self, db):
self.con = sqlite3.connect(db)
self.cur = self.con.cursor()
```

Step 3: Create Table: sql = ””””

```
CREATE TABLE IF NOT EXISTS employees(
id Integer Primary Key,
```

```
name text,
age text,
doj text,
email text,
gender text,
contact text,
address text
)
”””
self.cur.execute(sql)
self.con.commit()
```

Step 4:CRUD OPERATIONS :

Insert Function

```
def insert(self, name, age, doj, email, gender, contact, address):
self.cur.execute(”insert into employees values (NULL,?,?,?,?,?,?,?)”,
(name, age, doj, email, gender, contact, address))
self.con.commit()
```

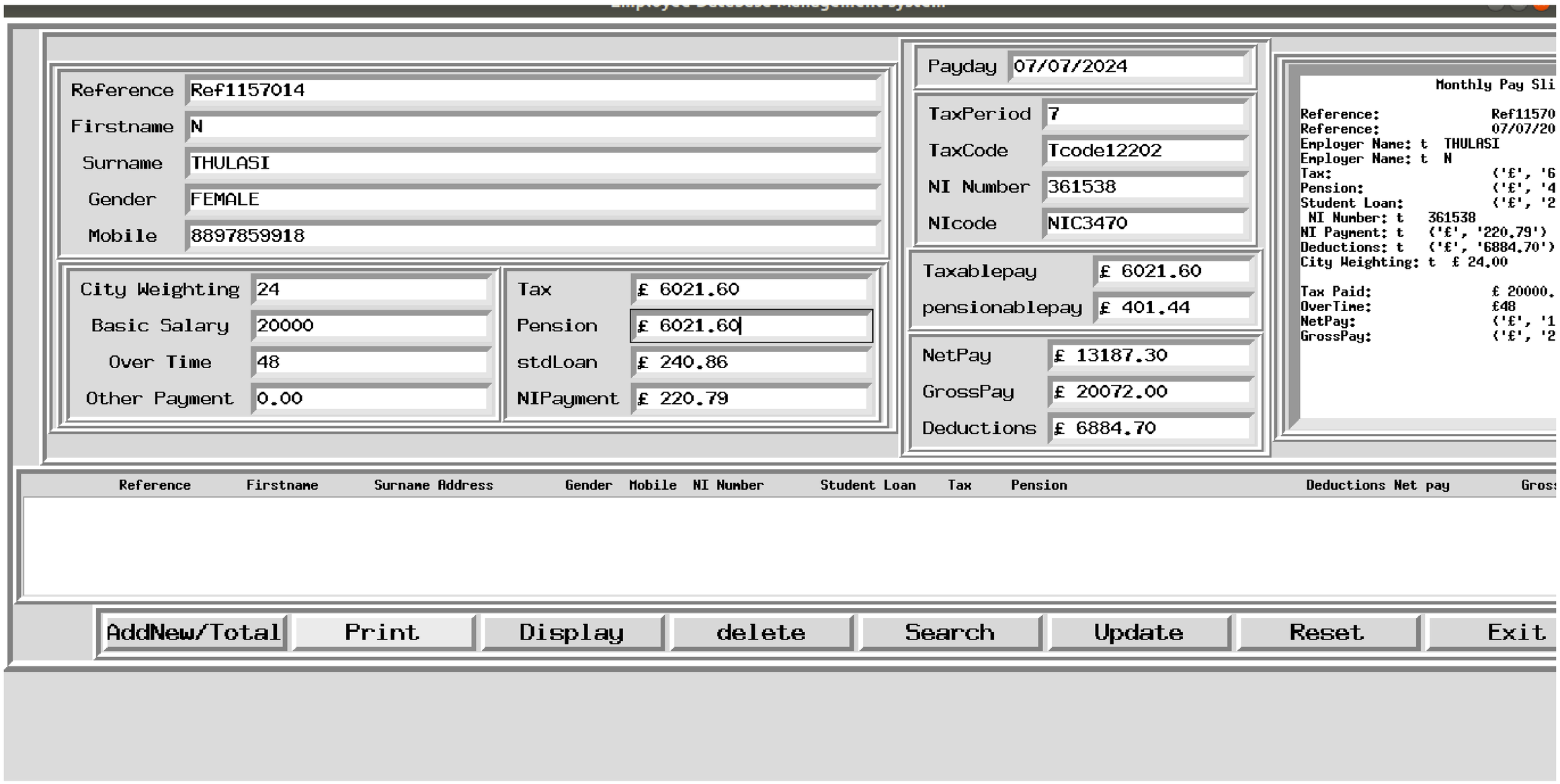
Fetch All Data from DB

```
def fetch(self):
self.cur.execute(”SELECT * from employees”)
rows = self.cur.fetchall()
print(rows)
return rows
```

Delete a Record in DB

```
def remove(self, id):
self.cur.execute(”delete from employees where id=?”, (id,))
self.con.commit()
```

OUTPUT:



5. CONCLUSION

In conclusion, developing an Employee management portal using Python, particularly with the Tkinter library and SQLite database, offers a practical and versatile solution for managing workforce information.

This system provides a user-friendly interface for efficient data entry, ensuring accuracy and compliance with organizational and legal requirements.

By following the step wise implementation and incorporating additional features as needed, organizations can benefit from improved HR processes, and enhanced decision-making capabilities.

The system's scalability, adaptability, and security measures contribute to its effectiveness in meeting the evolving needs of the organization.

Continuous maintenance, user training, and responsiveness to user feedback are essential for sustaining the system's functionality and ensuring its long-term success in facilitating efficient employee data management.