Final Group Project Report

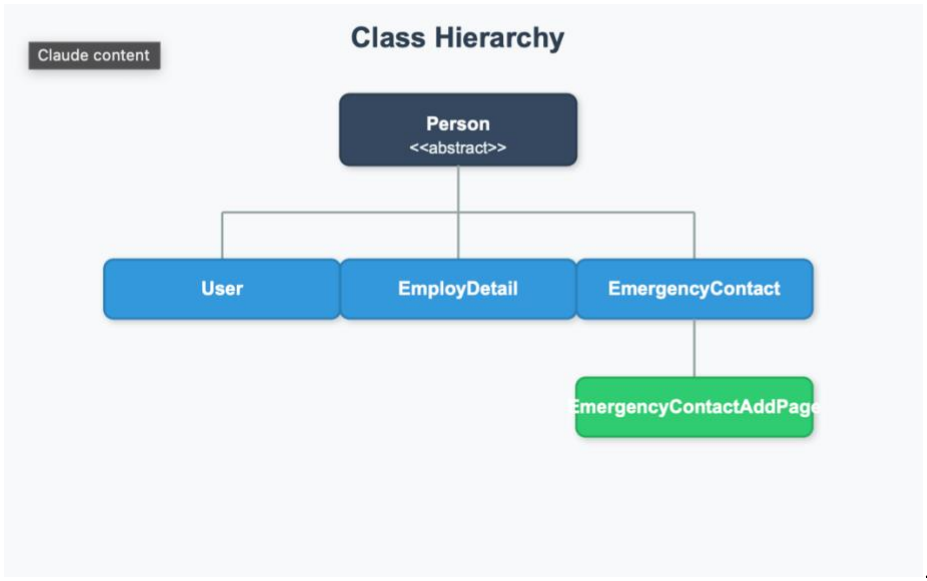
Employee Management System (EMS)

35

EC/2022/053	K.S.B.Galkotuwa
EC/2022/077	C.Niroshan
EC/2022/002	R.H.M.P.Rathmalage
EC/2022/078	G.Varushiny
EC/2022/023	A.Dhilukshy

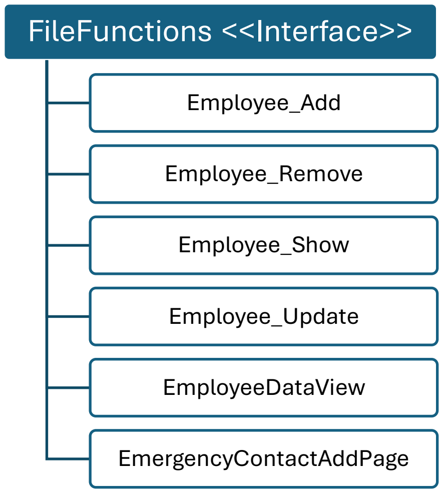
Table of Contents

Final Group Project Report 1
Employee Management System (EMS) 1
Introduction to the System 5
Architecture 7
• Core Classes Hierarchy 7



.... 4

• Interface Implementations 7
1. FileFunctions (Interface) 7



.... 7

2. Calcultions <<Interface>> 7
Key Components 8
1. Authentication System 8
• Implements user registration and login functionality 8
• Stores user credentials in a text file (userdata.txt) 8
• Provides basic security through username/password validation 8
2. Employee Management 8

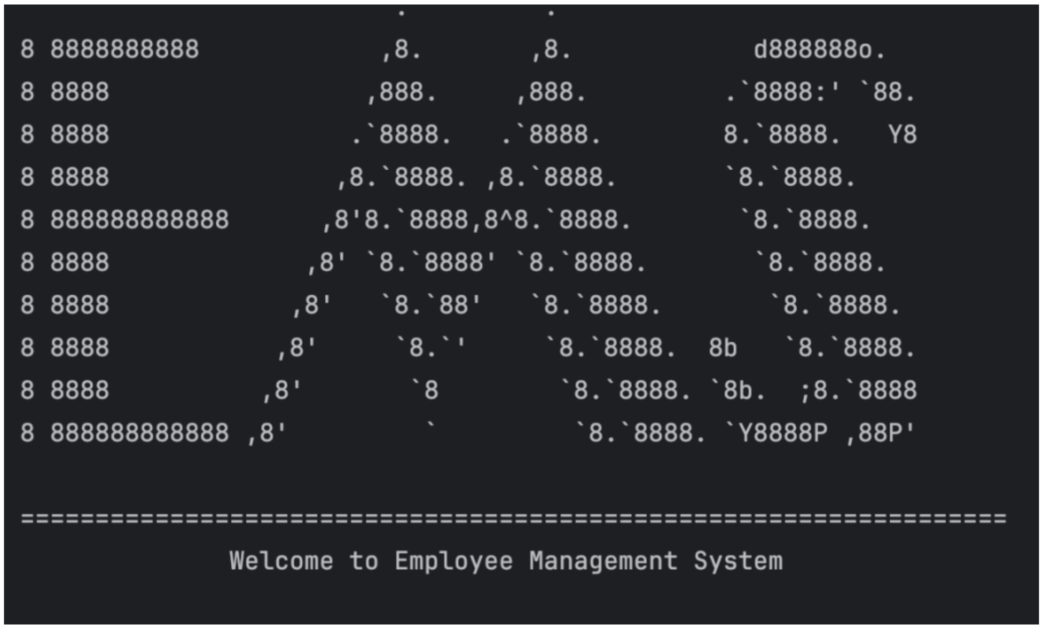
• Adding Employees (Employee_Add) 8
- Collects comprehensive employee details 8
- Generates HTML files for employee records 8
- Creates separate emergency contact records 8
- Calculates and stores salary information 8
• Viewing Employees (Employee_Show) 8
- Opens HTML files using desktop browser 8
- Provides formatted view of employee details 8
• Updating Employees (Employee_Update) 8
- Supports in-place updates of HTML files 8
- Maintains file structure during updates 8
• Removing Employees (Employee_Remove) 8
- Removes employee HTML files 8
- Clean deletion of records 8
• Salary Management 8
- Implements monthly and yearly salary calculations 8
- Accounts for varying days in months 8
- Generates detailed salary summaries in HTML format 8
Implementation Details 10
1. Data Validation 10
• The system implements robust validation across multiple areas: 10
- Email format validation using regex 10
- Contact number format validation (9 digits) 10
- Required field validation 10
- Salary validation (must be greater than zero) 10
2. HTML Generation 10
• The system creates three types of HTML files for each employee: 10
- Main employee details 10
- Emergency contact information 10
- Salary information 10
• Each HTML file includes: 10
- Consistent styling through external CSS 10
- Interactive elements via JavaScript 10
- Responsive design elements 10
- Cross-linking between related files 10
3. Exception Handling 10
• The system implements comprehensive error handling for: 10
- File operations 10
- Invalid user input 10
- Data validation 10
- Authentication failures 10
4. Directory Management 10

- Automatic creation of required directories 10
- Structured file organization 10
- Consistent naming conventions 10

Classes, Attributes, and Methods 11
Use Cases of OOP Concepts 15
Demonstration of the System 20
Technical Strengths 21
1. Modularity: Clear separation of concerns through interface implementation 21
2. Extensibility: Abstract classes and interfaces allow for easy system expansion 21
3. Data Persistence: HTML-based storage provides both data storage and presentation 21
4. Input Validation: Comprehensive validation across all user inputs 21
5. Error Handling: Robust exception handling throughout the system 21

Areas for Improvement 21
1. Security: Basic text-file based authentication could be enhanced 21
2. Database Integration: Could benefit from proper database implementation 21
3. GUI Interface: Currently console-based, could be enhanced with a graphical interface 21
4. Data Encryption: Sensitive data is stored in plain text 21
5. Backup System: No built-in backup functionality 21

Modifications from the Initial Plan 22
Teamwork 23
Conclusion 25



```
                         .         .
8 8888888888            ,8.       ,8.            d888888o.
8 8888                 ,888.     ,888.         .`8888:' `88.
8 8888                .`8888.   .`8888.        8.`8888.   Y8
8 8888               ,8.`8888. ,8.`8888.       `8.`8888.
8 888888888888      ,8'8.`8888,8^8.`8888.       `8.`8888.
8 8888             ,8' `8.`8888' `8.`8888.       `8.`8888.
8 8888            ,8'   `8.`88'   `8.`8888.       `8.`8888.
8 8888           ,8'     `8.`'     `8.`8888.  8b   `8.`8888.
8 8888          ,8'       `8        `8.`8888. `8b.  ;8.`8888
8 888888888888 ,8'         `         `8.`8888. `Y8888P ,88P'

==================================================================
              Welcome to Employee Management System
```

Introduction to the System

The **Employee Management System (EMS)** is a user-friendly and efficient solution designed to help small and medium-sized businesses manage their employee records with ease. In today's fast-paced world, keeping track of employee information is crucial for any organization, but traditional methods like paper-based systems or plain-text files can be time-consuming, error-prone, and difficult to maintain. The EMS solves these problems by using modern technologies and **Object-Oriented Programming (OOP)** principles to create a simple yet powerful platform for managing employee data.

At its heart, the EMS uses **HTML-based file storage** to store employee records in a neat and organized way. This means that all the information is not only easy to access but also looks professional and can be viewed in any web browser. This makes it a great tool for businesses that want to modernize their HR processes without needing complex software. The system is built using **Java**, a popular programming language known for its reliability and flexibility, making it a perfect choice for creating a system that works across different platforms.

The EMS has a **menu-driven console interface**, which means users can easily navigate through the system using simple text commands. With this interface, users can add new employees, view their details, update information, or remove records when needed. The system also includes some advanced features like **emergency contact management** and **salary calculation**. For example, the salary calculation feature automatically calculates monthly and yearly salaries, taking into account the number of days in each month, and presents the information in a clear and easy-to-understand HTML format.

One of the best things about the EMS is that it follows **OOP principles**, which make the system modular, scalable, and easy to maintain. Each part of the system is designed to handle a specific task, such as adding employees or calculating salaries. This modular design not only makes the system easier to understand but also allows for future improvements without much hassle.

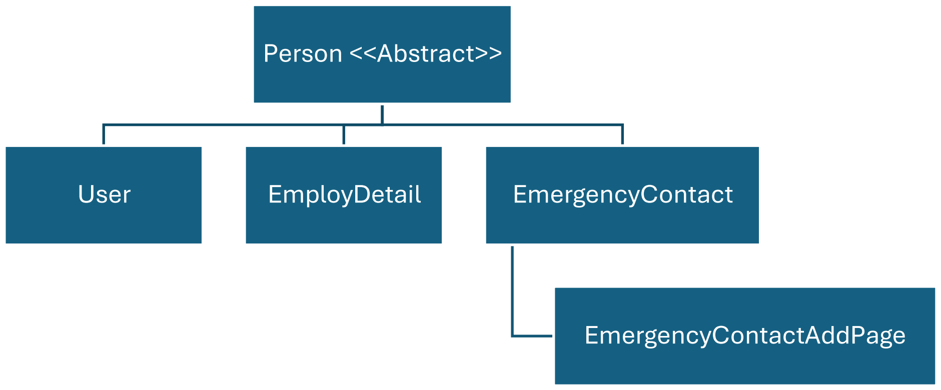
The EMS is especially useful for small businesses or startups that need a simple and cost-effective way to manage employee records without investing in expensive database systems. By storing data in HTML files, the system is both **portable** and **easy to use**, as the files can be opened on any device with a web browser. Additionally, the system includes strong **data validation** and **error handling** features to ensure that the information entered is accurate and that the system runs smoothly even if something goes wrong.

In conclusion, the **Employee Management System** is a modern, efficient, and practical solution for managing employee records. It combines the power of Java programming with the simplicity of HTML-based storage to create a platform that is both user-friendly and effective. Whether you're a small business owner or someone just starting to learn about computer science, the EMS is a great example of how technology can simplify everyday tasks and improve productivity. It's a project that not only solves real-world problems but also demonstrates the importance of good design and planning in software development.

We would like to extend our heartfelt thanks to **Dr. B.M.T. Kumarika** for her invaluable guidance and support throughout the development of this project. Her expertise and dedication in conducting the **BECS 12243 module** have been instrumental in helping us understand the core concepts of Object-Oriented Programming and software development. This project would not have been possible without her mentorship and encouragement. Thank you, Dr. Kumarika, for inspiring us to push our limits and create something meaningful!

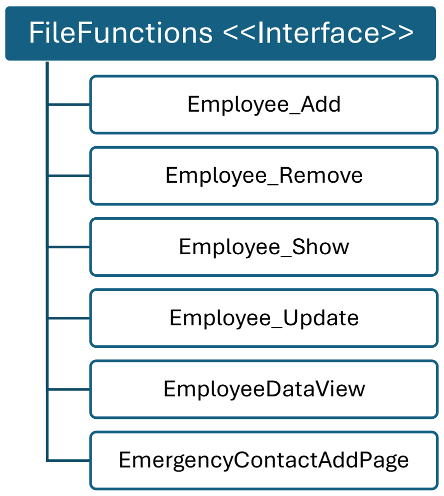
Architecture

- Core Classes Hierarchy

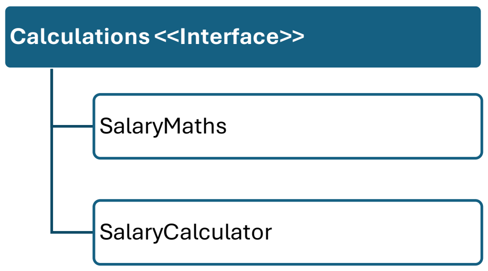


- Interface Implementations

1. FileFunctions (Interface)



2. Calculations <<Interface>>



Key Components

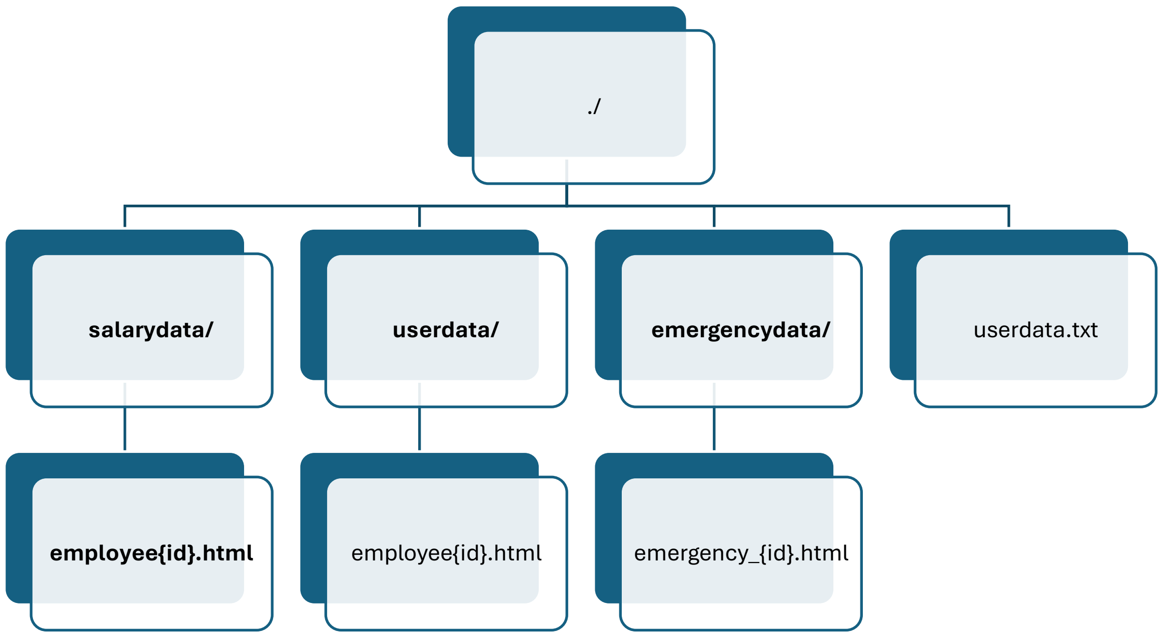
1. Authentication System

- Implements user registration and login functionality
- Stores user credentials in a text file (userdata.txt)
- Provides basic security through username/password validation

2. Employee Management

- Adding Employees (Employee_Add)
    - Collects comprehensive employee details
    - Generates HTML files for employee records
    - Creates separate emergency contact records
    - Calculates and stores salary information
- Viewing Employees (Employee_Show)
    - Opens HTML files using desktop browser
    - Provides formatted view of employee details
- Updating Employees (Employee_Update)
    - Supports in-place updates of HTML files
    - Maintains file structure during updates
- Removing Employees (Employee_Remove)
    - Removes employee HTML files
    - Clean deletion of records
- Salary Management
    - Implements monthly and yearly salary calculations
    - Accounts for varying days in months
    - Generates detailed salary summaries in HTML format

File Structure



userdata.txt – store username and password in comma separate format

Implementation Details

1. Data Validation

- The system implements robust validation across multiple areas:
    - Email format validation using regex
    - Contact number format validation (9 digits)
    - Required field validation
    - Salary validation (must be greater than zero)

2. HTML Generation

- The system creates three types of HTML files for each employee:
    - Main employee details
    - Emergency contact information
    - Salary information
- Each HTML file includes:
    - Consistent styling through external CSS
    - Interactive elements via JavaScript
    - Responsive design elements
    - Cross-linking between related files

3. Exception Handling

- The system implements comprehensive error handling for:
    - File operations
    - Invalid user input
    - Data validation
    - Authentication failures

4. Directory Management

    - Automatic creation of required directories
    - Structured file organization
    - Consistent naming conventions

Classes, Attributes, and Methods

The system is composed of several classes, each responsible for specific functionalities. Below is a breakdown of the key classes, their attributes, and methods

1. EntryPoint
   - Attributes:
     - USER_DATA_FILE
     - USER_DATA_DIR
     - EMERGENCY_DATA_DIR
     - SALARY_DATA_DIR
   - Methods:
     - main(): Entry point of the application, handles user registration and login.
     - createDirectories(): Creates necessary directories for storing employee data.
     - createUserDataFile(): Creates a file to store user credentials.
     - login(): Handles user login and navigation through the system.
     - register(): Handles user registration.
     - authenticateUser(): Validates user credentials during login.

2. MainMenu
   - Methods:
     - menu(): Displays the main menu of the system using ASCII art.

3. EmployDetail
   - Attributes:
     - Name
     - Email
     - Position
     - Address
     - employ_id
     - employ_salary
     - employ_contact
     - emergencyContact_name

        - emergencyContact_contact
        - emergencyContact_address
        - emergencyContact_email
    - Methods:
        - getInfo(): Collects employee details from the user.
        - toString(): Returns a string representation of the employee details.

4. Employee_Add
    - Methods:
        - createFileBaseHTML(): Creates an HTML file for the employee and writes their details into it.
        - listUserdataFiles(), removeFile(), viewFile(), updateFile(): Placeholder methods for file operations.

5. Employee_Show
    - Methods:
        - viewFile(): Opens and displays the HTML file of a specific employee.

6. Employee_Remove
    - Methods:
        - removeFile(): Deletes the HTML file of a specific employee.

7. Employee_Update
    - Methods:
        - updateFile(): Updates specific details in an employee's HTML file.

8. EmergencyContact
    - Attributes:
        - Inherits from Person class: name, contactNo, email, address
    - Methods:
        - toString(): Returns a string representation of the emergency contact details.

9. EmergencyContactAddPage
    - Methods:
        - createFileBaseHTML(): Creates an HTML file for the emergency contact details.

10. SalaryCalculator
    - Methods:
        - generateSalarySummary(): Generates a yearly salary summary in HTML format for an employee.
        - getMonthName(): Returns the name of the month based on the month number.

11. SalaryMaths
    - Methods:
        - daysInMonth(): Returns the number of days in a given month.

12. FileFunctions
    - Attributes:
        - USER_DATA_FILE
        - USER_DATA_DIR
        - EMERGENCY_DATA_DIR
        - SALARY_DATA_DIR
    - Methods:
        - createFileBaseHTML(), listUserdataFiles(), removeFile(), viewFile (), updateFile(): Interface methods for file operations.

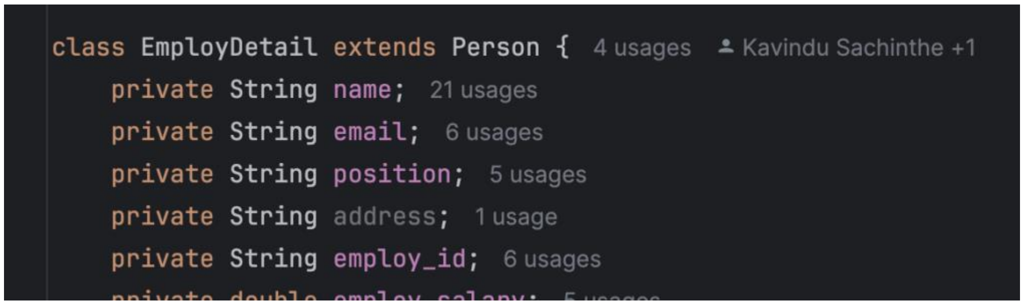
13. Person
    - Attributes:
        - Name
        - contactNo
        - email
        - address
    - Methods:
        - getInfo(), getAddress(), setAddress(), getContactNo(), setContactNo(), getEmail(), setEmail(), getName(), setName()

14. User
    - Attributes:
        - Name
        - password
    - Methods:
        - getName(), getPassword(), toString()

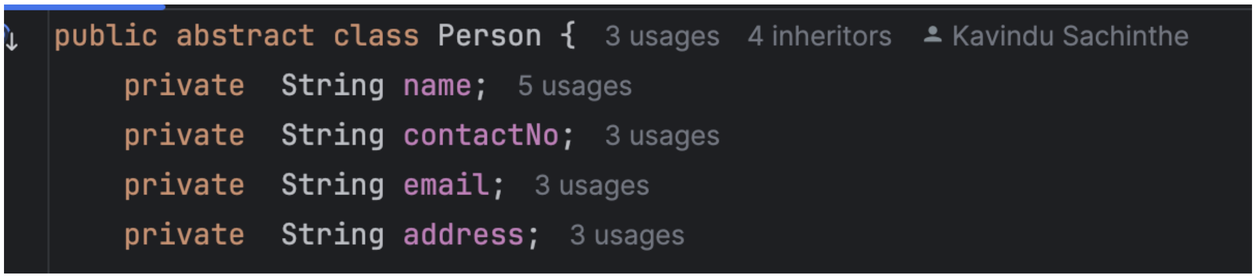
15. CodeExit
    - Methods:
        - out(): Displays an exit message and terminates the program.

Use Cases of OOP Concepts

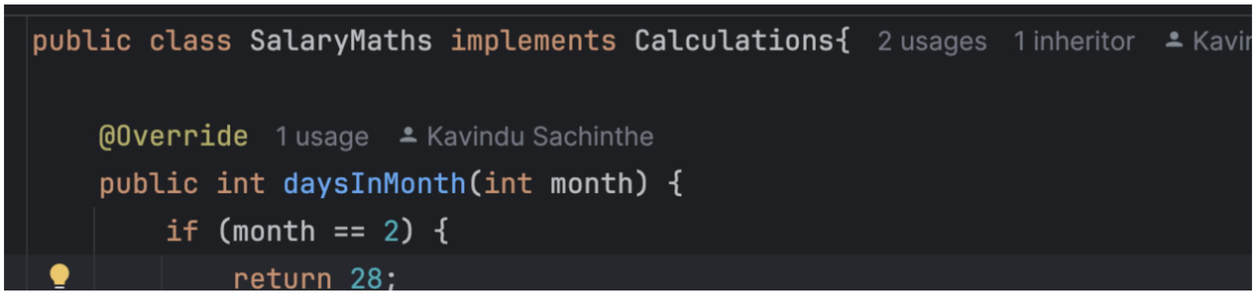
1. Inheritance:
   - The EmergencyContact class inherits from the Person class, reusing attributes like name, contactNo, email, and address.
   - The SalaryCalculator class inherits from the SalaryMaths class, reusing the daysInMonth() method.



```
class EmployDetail extends Person {  4 usages  Kavindu Sachinthe +1
    private String name;  21 usages
    private String email;  6 usages
    private String position;  5 usages
    private String address;  1 usage
    private String employ_id;  6 usages
    private double employ_salary;  5 usages
```

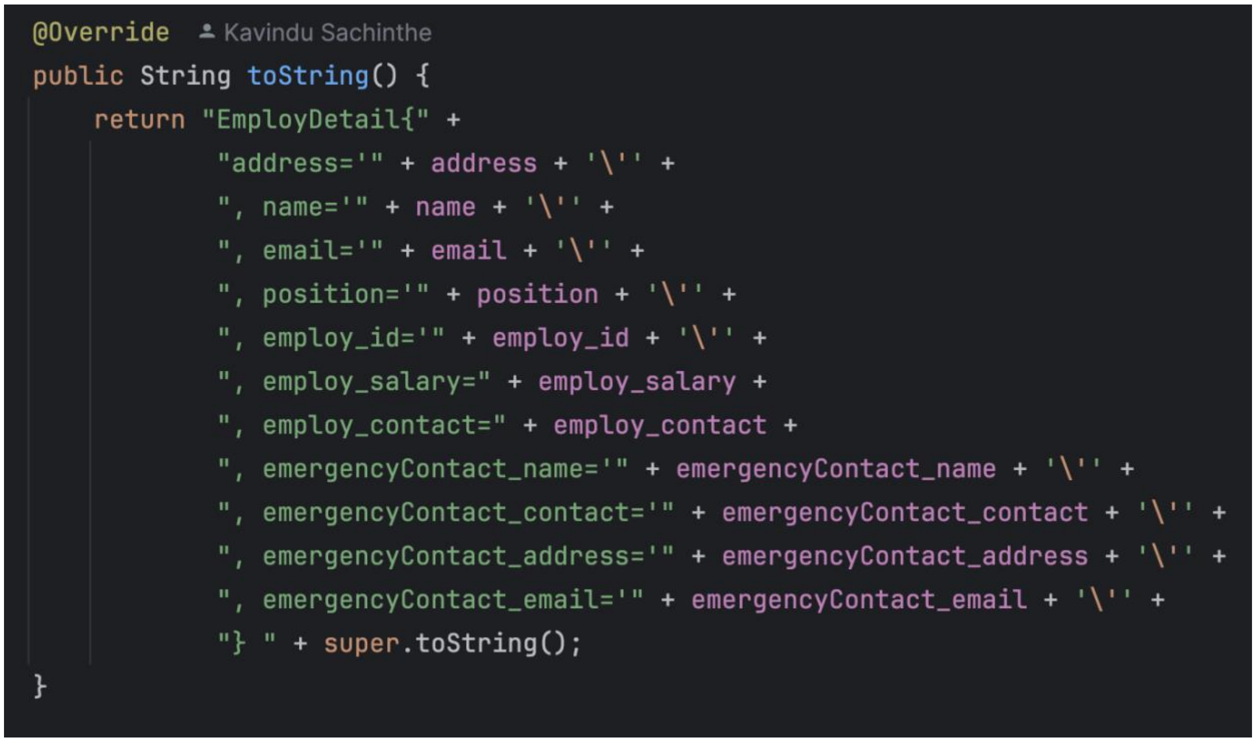


```
public abstract class Person {  3 usages  4 inheritors  Kavindu Sachinthe
    private  String name;  5 usages
    private  String contactNo;  3 usages
    private  String email;  3 usages
    private  String address;  3 usages
```

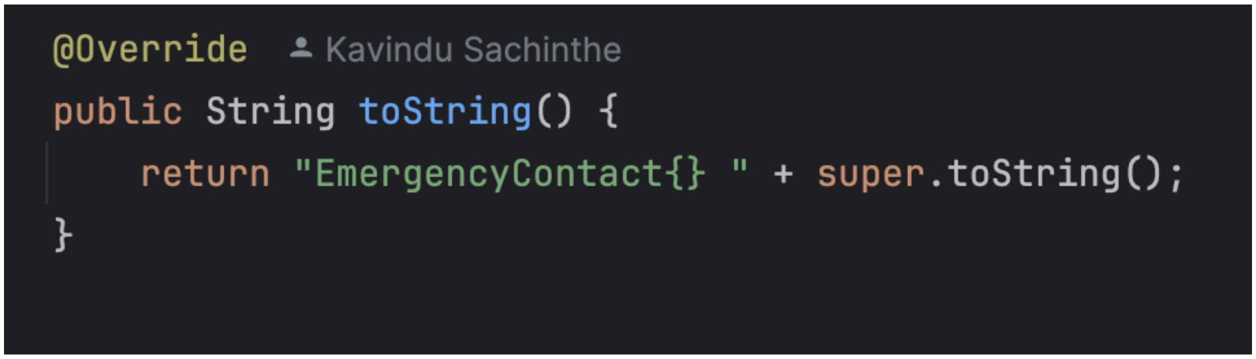


```
public class SalaryMaths implements Calculations{  2 usages  1 inheritor  Kavi

    @Override  1 usage  Kavindu Sachinthe
    public int daysInMonth(int month) {
        if (month == 2) {
            return 28;
```

2. Polymorphism:
    - The toString() method is overridden in multiple classes (EmployDetail, EmergencyContact, User) to provide customized string representations of objects.

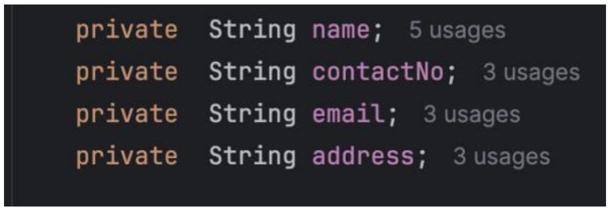


```java
@Override  Kavindu Sachinthe
public String toString() {
    return "EmployDetail{" +
            "address='" + address + '\'' +
            ", name='" + name + '\'' +
            ", email='" + email + '\'' +
            ", position='" + position + '\'' +
            ", employ_id='" + employ_id + '\'' +
            ", employ_salary=" + employ_salary +
            ", employ_contact=" + employ_contact +
            ", emergencyContact_name='" + emergencyContact_name + '\'' +
            ", emergencyContact_contact='" + emergencyContact_contact + '\'' +
            ", emergencyContact_address='" + emergencyContact_address + '\'' +
            ", emergencyContact_email='" + emergencyContact_email + '\'' +
            "} " + super.toString();
}
```

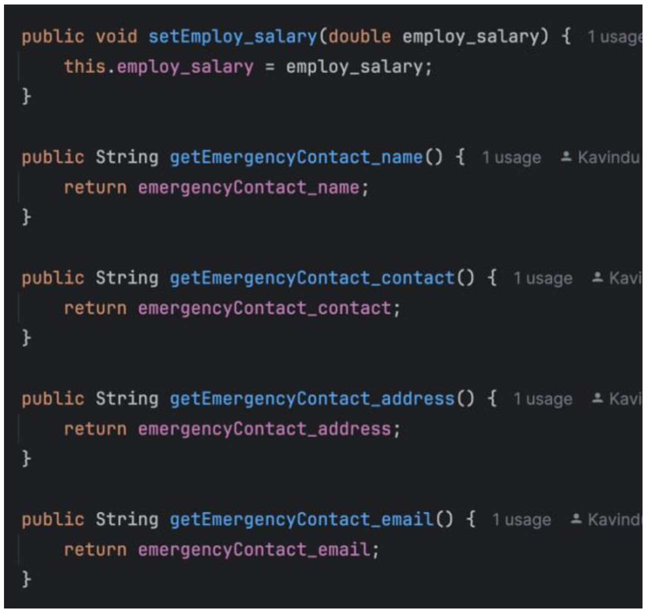


```java
@Override  Kavindu Sachinthe
public String toString() {
    return "EmergencyContact{} " + super.toString();
}
```

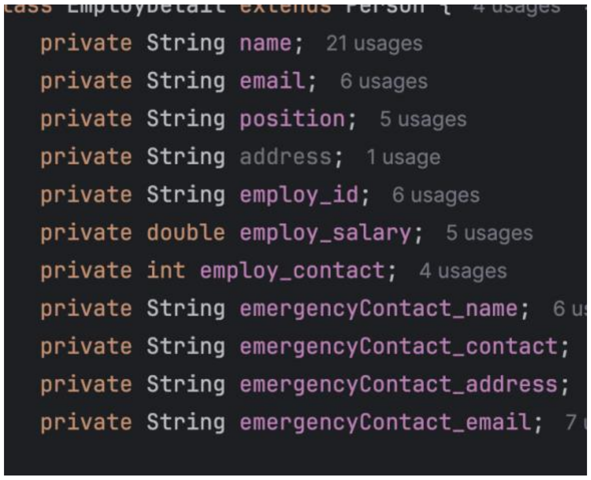
3. Encapsulation:
    - All attributes in the Person class are private, and access to them is controlled through public getter and setter methods.



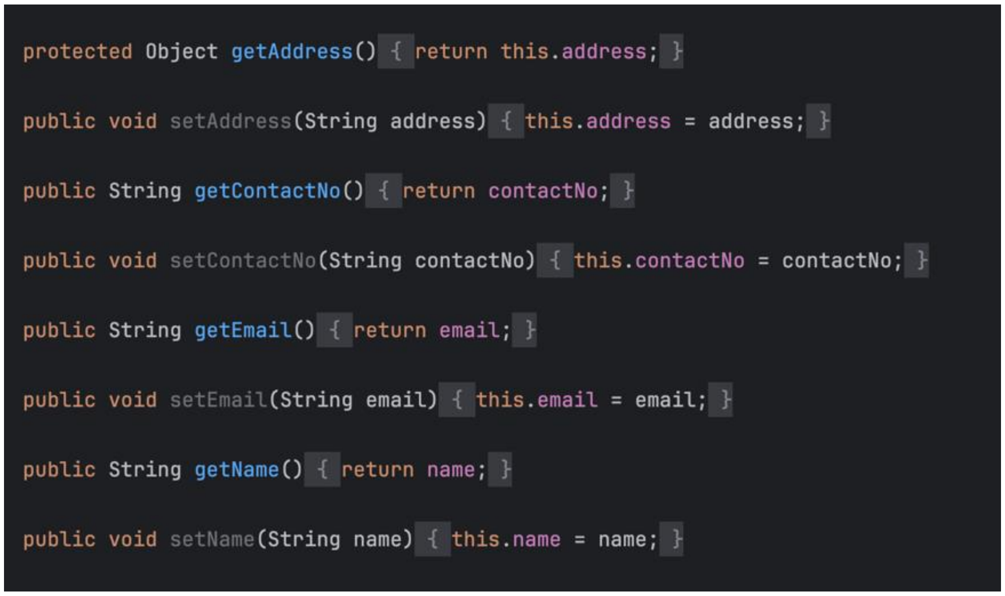
```
private  String name;  5 usages
private  String contactNo;  3 usages
private  String email;  3 usages
private  String address;  3 usages
```



```
public void setEmploy_salary(double employ_salary) {  1 usag
    this.employ_salary = employ_salary;
}

public String getEmergencyContact_name() {  1 usage  Kavindu
    return emergencyContact_name;
}

public String getEmergencyContact_contact() {  1 usage  Kavi
    return emergencyContact_contact;
}

public String getEmergencyContact_address() {  1 usage  Kavi
    return emergencyContact_address;
}

public String getEmergencyContact_email() {  1 usage  Kavind
    return emergencyContact_email;
}
```

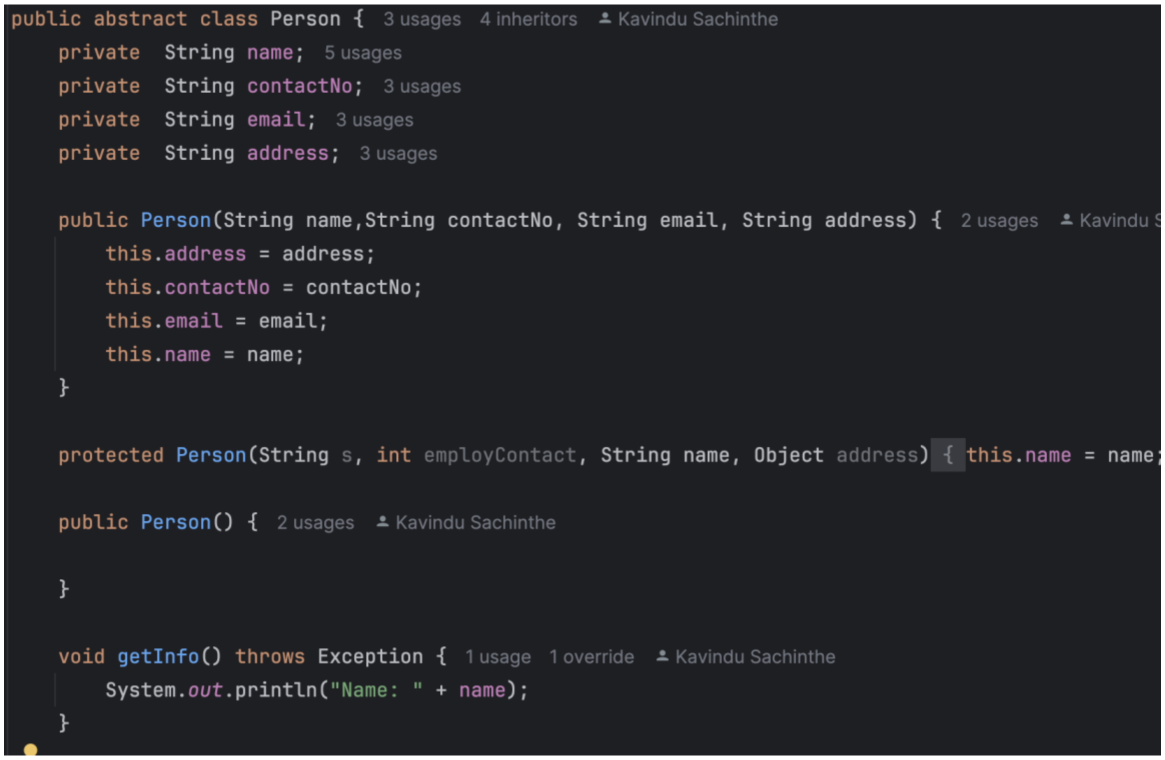


```
lass EmployDetail extends Person {  4 usages
    private String name;  21 usages
    private String email;  6 usages
    private String position;  5 usages
    private String address;  1 usage
    private String employ_id;  6 usages
    private double employ_salary;  5 usages
    private int employ_contact;  4 usages
    private String emergencyContact_name;  6 u
    private String emergencyContact_contact;
    private String emergencyContact_address;
    private String emergencyContact_email;  7
```

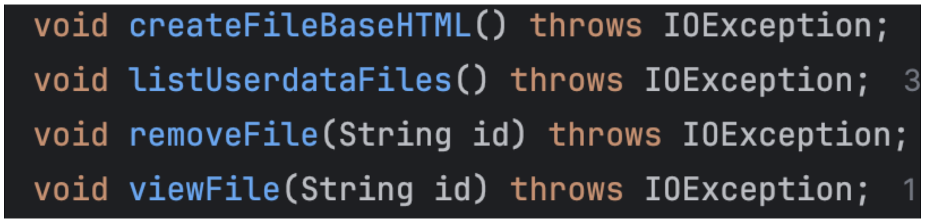


```
protected Object getAddress() { return this.address; }

public void setAddress(String address) { this.address = address; }

public String getContactNo() { return contactNo; }

public void setContactNo(String contactNo) { this.contactNo = contactNo; }

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }
```

4. Abstraction:
    - The FileFunctions interface defines a contract for file operations, which is implemented by classes like Employee_Add, Employee_Remove, and Employee_Show.

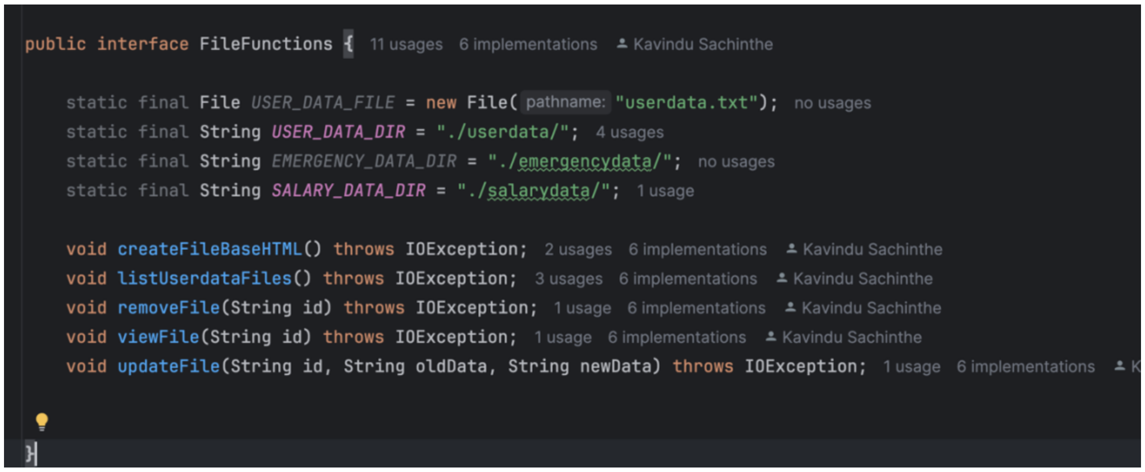


```
public abstract class Person {  3 usages  4 inheritors  Kavindu Sachinthe
    private  String name;  5 usages
    private  String contactNo;  3 usages
    private  String email;  3 usages
    private  String address;  3 usages

    public Person(String name,String contactNo, String email, String address) {  2 usages  Kavindu S
        this.address = address;
        this.contactNo = contactNo;
        this.email = email;
        this.name = name;
    }

    protected Person(String s, int employContact, String name, Object address) { this.name = name;

    public Person() {  2 usages  Kavindu Sachinthe

    }

    void getInfo() throws Exception {  1 usage  1 override  Kavindu Sachinthe
        System.out.println("Name: " + name);
    }
```

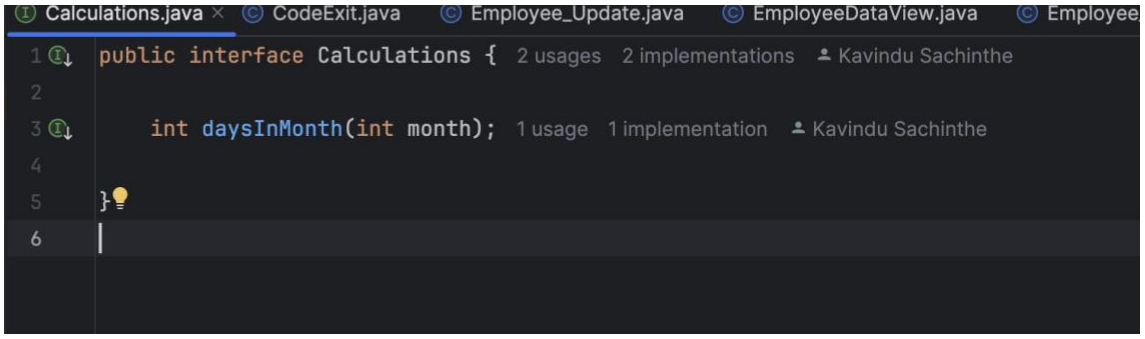


```
void createFileBaseHTML() throws IOException;
void listUserdataFiles() throws IOException;  3
void removeFile(String id) throws IOException;
void viewFile(String id) throws IOException;  1
```

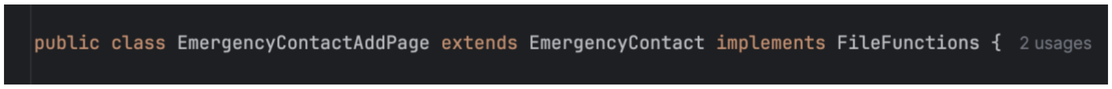
5. Interface:
    - The FileFunctions interface is implemented by multiple classes to ensure consistent file handling across the system.
    - The Calculations interface is implemented by multiple classes to calculate calculate months in the system.



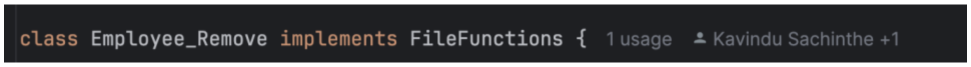
```
public interface FileFunctions {  11 usages  6 implementations  Kavindu Sachinthe

    static final File USER_DATA_FILE = new File( pathname: "userdata.txt");  no usages
    static final String USER_DATA_DIR = "./userdata/";  4 usages
    static final String EMERGENCY_DATA_DIR = "./emergencydata/";  no usages
    static final String SALARY_DATA_DIR = "./salarydata/";  1 usage

    void createFileBaseHTML() throws IOException;  2 usages  6 implementations  Kavindu Sachinthe
    void listUserdataFiles() throws IOException;  3 usages  6 implementations  Kavindu Sachinthe
    void removeFile(String id) throws IOException;  1 usage  6 implementations  Kavindu Sachinthe
    void viewFile(String id) throws IOException;  1 usage  6 implementations  Kavindu Sachinthe
    void updateFile(String id, String oldData, String newData) throws IOException;  1 usage  6 implementations  K

}
```



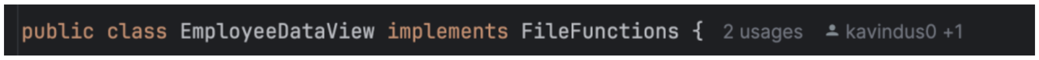
```
public interface Calculations {  2 usages  2 implementations  Kavindu Sachinthe

    int daysInMonth(int month);  1 usage  1 implementation  Kavindu Sachinthe

}
```



```
public class EmergencyContactAddPage extends EmergencyContact implements FileFunctions {  2 usages
```



```
class Employee_Remove implements FileFunctions {  1 usage  Kavindu Sachinthe +1
```

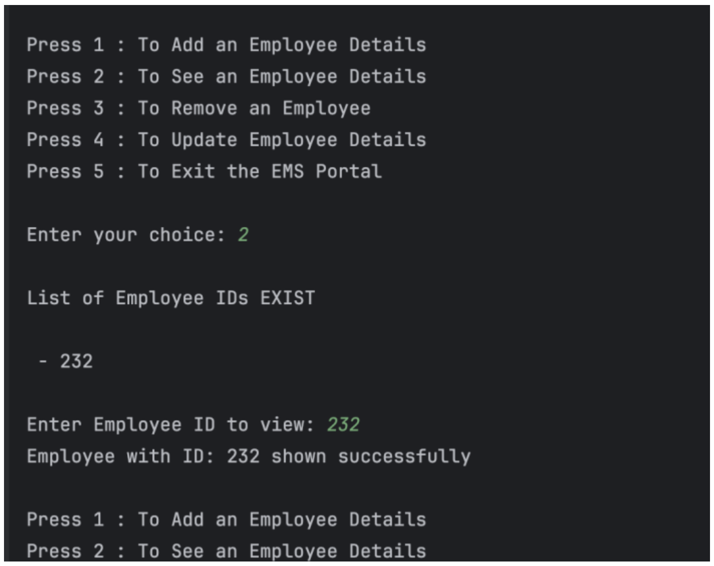


```
public class EmployeeDataView implements FileFunctions {  2 usages  kavindus0 +1
```

Demonstration of the System

https://vimeo.com/1049384123?share=copy

1. User Registration and Login:
    - Users can register by providing a username and password. The system checks for existing users and stores the credentials in a file.
    - Upon successful login, users are presented with the main menu.
2. Adding an Employee:
    - Users can add an employee by entering details such as name, ID, email, position, contact information, and emergency contact details.
    - The system creates an HTML file for the employee and stores their details in a structured format.
3. Viewing Employee Details:
    - Users can view an employee's details by entering their ID. The system opens the corresponding HTML file in a web browser.
4. Updating Employee Details:
    - Users can update specific details of an employee by providing the employee ID, the old data, and the new data. The system modifies the HTML file accordingly.
5. Removing an Employee:
    - Users can remove an employee by entering their ID. The system deletes the corresponding HTML file.
6. Salary Calculation:
    - The system calculates the yearly salary summary for an employee, including daily and monthly salary breakdowns, and stores it in an HTML file.



```
Press 1 : To Add an Employee Details
Press 2 : To See an Employee Details
Press 3 : To Remove an Employee
Press 4 : To Update Employee Details
Press 5 : To Exit the EMS Portal

Enter your choice: 2

List of Employee IDs EXIST

 - 232

Enter Employee ID to view: 232
Employee with ID: 232 shown successfully

Press 1 : To Add an Employee Details
Press 2 : To See an Employee Details
```

Technical Strengths

1. Modularity: Clear separation of concerns through interface implementation

2. Extensibility: Abstract classes and interfaces allow for easy system expansion

3. Data Persistence: HTML-based storage provides both data storage and presentation

4. Input Validation: Comprehensive validation across all user inputs

5. Error Handling: Robust exception handling throughout the system

Areas for Improvement

1. Security: Basic text-file based authentication could be enhanced

2. Database Integration: Could benefit from proper database implementation

3. GUI Interface: Currently console-based, could be enhanced with a graphical interface

4. Data Encryption: Sensitive data is stored in plain text

5. Backup System: No built-in backup functionality

Modifications from the Initial Plan

1. Addition of Emergency Contact Management:
    - Initially, the system only handled basic employee details. The emergency contact feature was added to provide a more comprehensive employee management solution.
2. Salary Calculation Module:
    - The salary calculation feature was not part of the initial plan but was added to enhance the system's functionality.
3. HTML-Based Storage:
    - The initial plan considered plain-text storage, but the team decided to use HTML files for better readability and integration with web applications.
4. Login Menu:
    - Initial Plan is to implement Open Access System, But due to security of information, Login Menu added, Wish to Add MD5 Encryption but due to Complexity and Time Feasibility It Avoided
5. Register Menu:
    - Initial Plan is to implement Open Access System, But due to security of information, Register Menu added for Adding Limited Users

Teamwork

The project was developed collaboratively by the team members, with each member responsible for specific modules:

- EC/2022/053 (K.S.B. Galkotuwa):
    - Implemented the MainMenu and EntryPoint classes.
    - Implemented Interfaces Calculations
    - Implemented FileFunction Interface
    - Handled system coordination and integration.
    - HTML file created and handled errors and handle Exceptions
    - EmployeeDataView class created
    - Intergration Testing Implemented
    - System Testing Implemented

- EC/2022/023 (A. Dhilukshy):
    - Implemented the EmployDetail and Employee_Add classes.
    - User Class created
    - Implemented Person super class.
    - Ensured proper validation and handling of duplicate entries.
    - Java Script File created
    - EntryPoint Created using switch-case expression
    - Unit Test Created for EmployDetail and Employee_Add classes

- EC/2022/078 (G. Varushiny):
    - Implemented the Employee_Show class.
    - Login Menu Added
    - Handled exceptions for missing or corrupted files.
    - Documentation Added for the code
    - User Authentication methods Added for Registration
    - Unit Test Created for Employ_show and Login classes

- EC/2022/002 (R.H.M.P. Rathmalage):
  - Implemented the Employee_Remove class.
  - Register menu Added
  - Validate User Inputs
  - Ensured error handling for non-existing files.
  - Unit Test Created for Employee_Remove and RegisterMenu
  - Static Arts Created

- EC/2022/077 (C. Niroshan):
  - Implemented the Employee_Update and CodeExit classes.
  - Handled Errors using try-catch block
  - Ensured the integrity of HTML formatting during updates.
  - CSS Stylesheet created
  - Salary Calculator HTML File Created

The team used GitHub for version control and collaboration. The repository can be accessed here.

https://github.com/kavindus0/EmployeeManagementSystemJava

Conclusion

The **Employee Management System (EMS)** is a practical and efficient solution for managing employee records, designed with simplicity and usability in mind. By leveraging **HTML-based storage** and **Java programming**, the system provides a modern, cost-effective alternative to traditional record-keeping methods. Its **menu-driven interface** makes it accessible to users with minimal technical knowledge, while its advanced features, such as **salary calculation** and **emergency contact management**, add significant value for businesses.

The project also serves as an excellent example of how **Object-Oriented Programming (OOP)**principles can be applied to create modular, scalable, and maintainable software. The system's design ensures that it can be easily extended or modified in the future, making it a versatile tool for businesses of all sizes.

While the EMS is already a robust solution, there is always room for improvement. Future enhancements could include the integration of a **graphical user interface (GUI)**, **database support** for better data management, and **data encryption** for improved security. These additions would further enhance the system's functionality and make it even more user-friendly.

In conclusion, the EMS is a testament to the power of combining **technology** and **good design** to solve real-world problems. It not only simplifies employee management but also serves as a valuable learning tool for students and aspiring developers. This project has been a rewarding experience, and we are proud of what we have accomplished.