High Level Design

Employee Management System

TEAM: 4

HIGH LEVEL DESIGN

Contents

1. Introduction
- 1.1. Why this High Level Design Document
- 1.2. Scope
- 1.3. Definition
- 1.4. Overview

2. General Description
- 2.1. System Perspective
- 2.2. Tools used
- 2.3. General Constraints
- 2.4. Assumptions

3. Design Details
- 3.1. Main Design Features
- 3.2. Application Architecture
- 3.3. Database Design
- 3.4. User Interface
- 3.5. Error Handling
- 3.6. Performance
- 3.7. Reliability & Availability
- 3.8. Reusability
- 3.9. Application compatibility
- 3.10. Help
- 3.11. Resource utilization
- 3.12. Portability
- 3.13. Maintainability
- 3.14. Security
- 3.15. Interface

Change Record

Revision	Date	Author	Changes
1.0		Team 4	
2.0		Team 4	
3.0		Team 4	

HIGH LEVEL DESIGN
Revision:3

1. Introduction

1.1. Why this High Level Design Document?

The purpose of this High Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

1.2. Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1.3. Definitions

- C++ - A programming language.
- Object Oriented Programming concepts were added.
- File managements are used in program
- Files are used as the Data base
- Visual studio 2019 is use for coding

1.4. Overview The

HLD will:

- Present all of the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces.
- Describe the performance of requirements.
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:
    - Reliability
    - Maintainability
    - Availability
    - Reusability
    - Portability
    - Application compatibility

HIGH LEVEL DESIGN
Revision:3

2. General Description

2.1. System Perspective

Employee Management System is a software that is designed to manage Employee information in a more operative manner that will help admin to manage employee data effectively. It provides the functionality to view, organize and manage the employee information that are available in the database(files). The admin can add details of new employees in the organization into the database and also can able to delete the details of employee from the database.
The data should be found easily if needed. So, this SRS report is a case study for system analysis of the Employee Management System, and the study is a clear comprehensive view about this system, and how we can design a system for this record management.

2.2. Tools used

- Windows Operating System
- Visual Studio 2019 for developing the source code.
- C++ language.
- TXT files as database.
- Web http://draw.io/

2.1. General Constraints

The Employee Management System must be user friendly. The system shall be built using a standard Visual Studio, Windows operating system, console for loading the output, Files used as a database and web draw.io used for designing the diagram for documentation.

2.2. Assumptions

This project is based on the idea of storing employee details in an organization. In doing so, many documents are created, and it is assumed that design flaws will be found early on. It is also assumed that all aspects of this project have the ability to work together in the way the designer is expecting. Another assumption is that the current intended documentation will suffice to make this project count towards the Software Engineering Subtract. There is also an assumption that none of the work or hardware will be stolen or sabotaged.
.

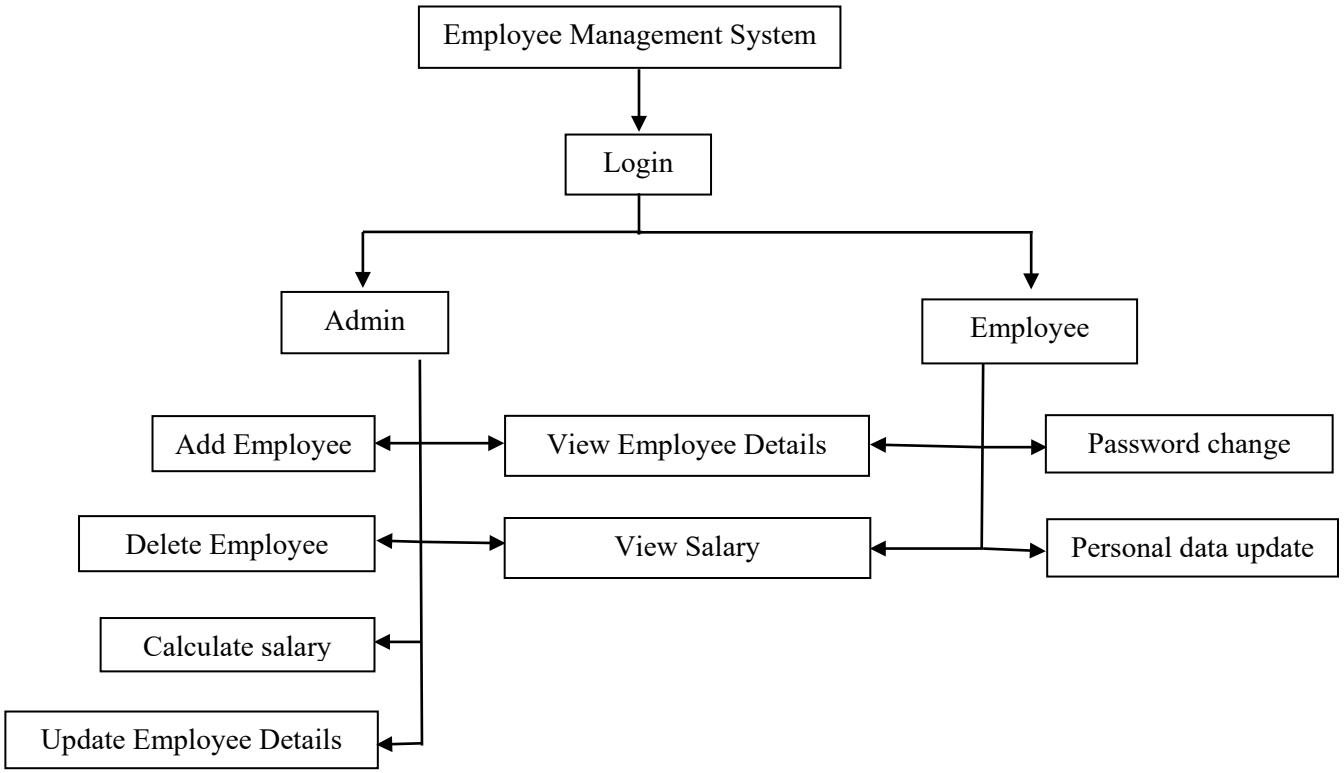
HIGH LEVEL DESIGN
Revision:3

3. Design Details

3.1. Main Design Features

The main design features include three components: the architecture, the user interface and the database. There are four databases in this application: Employee Data, Job Title ,Leave Record, Salary Table. There are two modules in this system: admin and employee. The main functions of admin are to add employee details ,view details, update details and delete employee details .In addition, the functions of employee include view details, update details and changing password .In order to make these designs easier to understand, the design has been illustrated in attached diagrams (Use case diagram, Sequence diagram, Class diagram). …….

3.2. Application Architecture



3.3. Database Design

The DBMS is used as the database function which creates and manipulates a hashed database for the application, which stores key/data pairs in a data file.

3.4. User Interface

The user interface is a very simple plain layout with little to no graphics. It will display information very clearly for the user and will primarily output information to the admin and employee through pages according to admin and employee login credential.

3.5. Error Handling

Should errors be encountered, an explanation will be displayed as to what went wrong. An error will be defined as anything that falls outside the normal and intended usage.

3.6. Performance

Performance is going to be very important for this project. The performance shall depend upon hardware components of the client/Employee.

3.7. Reliability & Availability

In order to ensure reliability, this system is being designed using software that is established to be stable and easy to use.

This system is designed to run 24/7 and be readily available to the user

3.8. Reusability

The code written and the components used should have the ability to be reused with no problems. Should time allow, and detailed instructions are written on how to create this project, everything will be completely reusable to anyone.

3.9. Application compatibility

The main portion of this project will be using C++. Each component will have its own task to perform, and it is the job of the code to ensure proper transfer of information. C++ is a powerful language. In C++ we can write structured programs and Oops also. C++ is a superset of C and therefore most constructs of C are legal in C++ with their meaning unchanged. However, there are some exceptions and additions.

3.10. Help

Help will come in the form of all the documentation created prior to coding, which explains the intended uses. Should time allow, detailed instructions will be written on how to create and implement the system with the intention of publishing as an Open Source solution.

3.11. Resource utilization

When any task is performed, it will likely use all the processing power available until that function is finished.

3.12. Portability

This system should have the ability that, once it is together, the entire system should be able to be physically moved to any location. all components should be compiled from source.

3.13. Maintainability

Very little maintenance should be required for this setup. An initial configuration will be the only system required interaction after the system is put together. The only other user maintenance would be any changes to settings after setup, and any specified special cases where user settings or history need to be changed. Upgrades of hardware and software should have little effect on this project.

3.14. Security

Because security is not the prime focus of this project, only the minimal aspects of security will be implemented. A username and password will be required to log into the system. For now, all data will be sent in plain text.

3.15. Interfaces

There are two main interfaces for this project. First the interface between employee/Admin as well as Terminal, the second is the Terminal and the database(files).