Architecture

Leave Management System Web Application

Revision Number : 1.0
Last Date of Revision : 20/07/2022

Billa Mounika

Contents

Document Version Control

Date Issued	Version	Description	Author
12/03/2022	1	Initial Architecture-Ver1.0	Billa Mounika

Abstract

Every organization has a challenge to manage their employees. If it is not done properly, some problems may happen to deliver projects/work. That's why a leave management system is necessary for an organization.
Here employees can apply for leaves and supervisors can manager those leaves easily.

1 Introduction

1.1 Why this Architecture

The purpose of this Architecture 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.

The main objective of this project is to make a system where admin can manage employees like adding, removing, editing the info of employees and approve or reject leaves of the employees. And employees can apply for leave and claim the money .

The Architecture will:

- Present all of the design aspects and define them in a detail
- Describe the user interface being implemented
- Includes design features and architecture of the project
- List and describe the non functional attribute like:
    - Reliability
    - Maintainability
    - Portability
    - Reusability
    - Application compatibility
    - Serviceability

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 administrator of the system.

1.3 Constraints

The Leave Management System must be user-friendly, as automated as possible, and the user should not be required to know any other work.

2.Technical specifications

2.1 Logging

The system should log every event so that the user will know what process is running internally.

Initial Step-By-Step Description

1. The system identifies at what step logging required
2. The system should be able to log each and every system flow
3. Developers can choose a logging method. You can choose database logging or file logging as well
4. System should not hang even after so many loggins. Logging just because we can debug issues, so logging is mandatory.

2.2 Database

System needs to store each and every record in the database, employee info as well as logging. And it has to send data to the user on request. Here we have used a NoSQL database, Cassandra.

3. Deployment

4.Technology stack

Front End	HTML/CSS/JS/Bootstrap
Backend	Python/ Flask
Database	Cassandra

Deployment	Heroku
version control	GitHub

5. Proposed Solution

The solution proposed here is a leave management system that can be implemented to perform the above-mentioned use cases. Here employees can apply for leave and then notification will be sent to supervisors, he/she can take action on that. Also supervisors can edit, delete, and update the details of the employees. There will be an option for employees where they can claim the equivalent money if they have more than 22 leave left.

6. 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.

7. KPIS (Key Performance Indicators)

1. Employees can apply for leave easily.
2. Supervisors can manage the employees leaves properly.
3. Details of each employee is stored securely.
4. Delete, edit of employees's information can be done quickly.
5. Everyone can claim the money equivalent to leave left securely.

8. Conclusion

This system optimizes the employee management in an organization. It helps both employees and the organization to fulfill their needs. This way an organization can save both time and cost.