Smart Employee Management System (SEMS)

TEAM PENTAD X-5

Overview

User Groups

Functional Requirements

Non-Functional Requirements

Architecture

Flow of data and control using Sequence and Activity Diagrams

Concurrency within System

Lessons Learned

What is a Smart Employee Management System?

A database containing employee information.

Monitor Employee attendance.

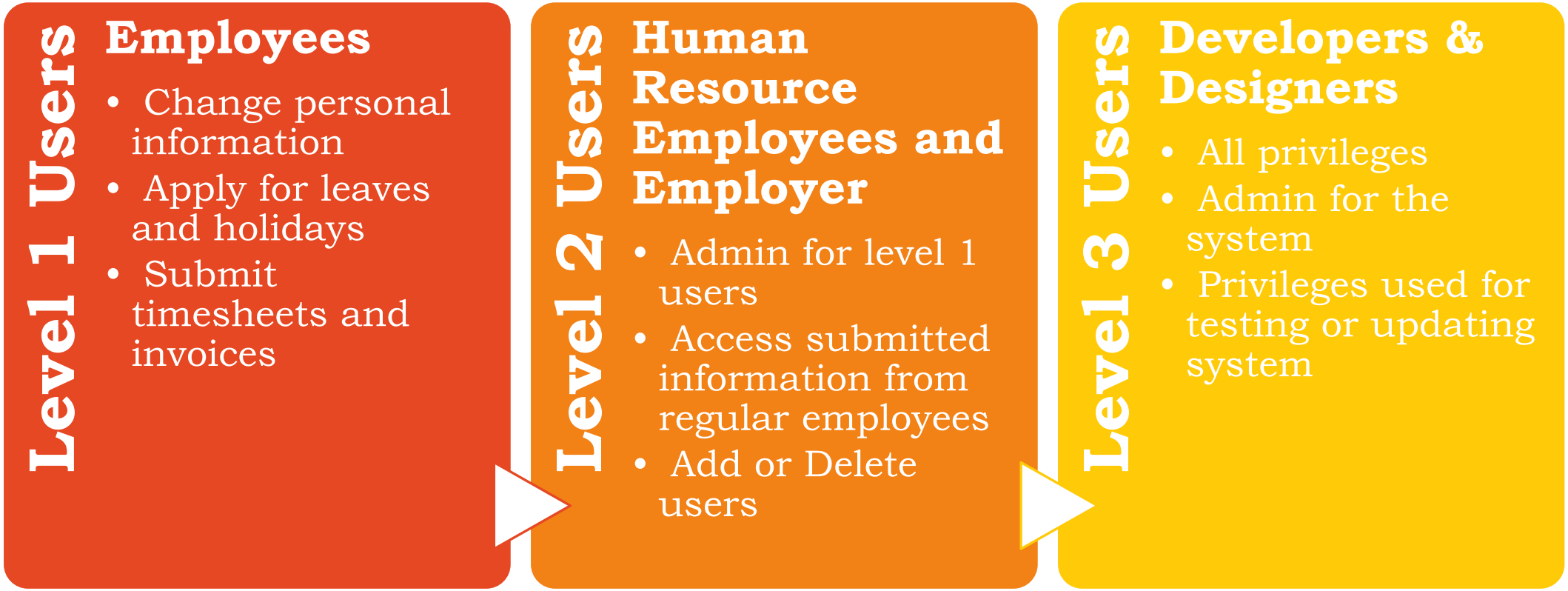
Track Work Hours

Payroll Processing

Add / Delete / Edit Timesheets

Create and Track Project Progress

User Groups


Level 1 Users
Employees
• Change personal information
• Apply for leaves and holidays
• Submit timesheets and invoices
Level 2 Users
Human Resource Employees and Employer
• Admin for level 1 users
• Access submitted information from regular employees
• Add or Delete users
Level 3 Users
Developers & Designers
• All privileges
• Admin for the system
• Privileges used for testing or updating system

Functional Requirements

Employee

- Update personal information
- Update timesheets and invoices.
- Apply for holidays and leaves.

Employer / HR Department

- Approve or decline timesheets.
- Retrieve employee's information
- Add or Delete employees.

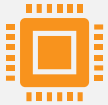
Database

- Update employee information.
- Register employees
- Update timesheets and invoices

Developer & Designer

- Change privilege level.
- Update different parts of the system.
- Access to all information in the database.

Non-Functional Requirements



Performance requirements

Response Time
Throughput
Scalability

Safety requirements

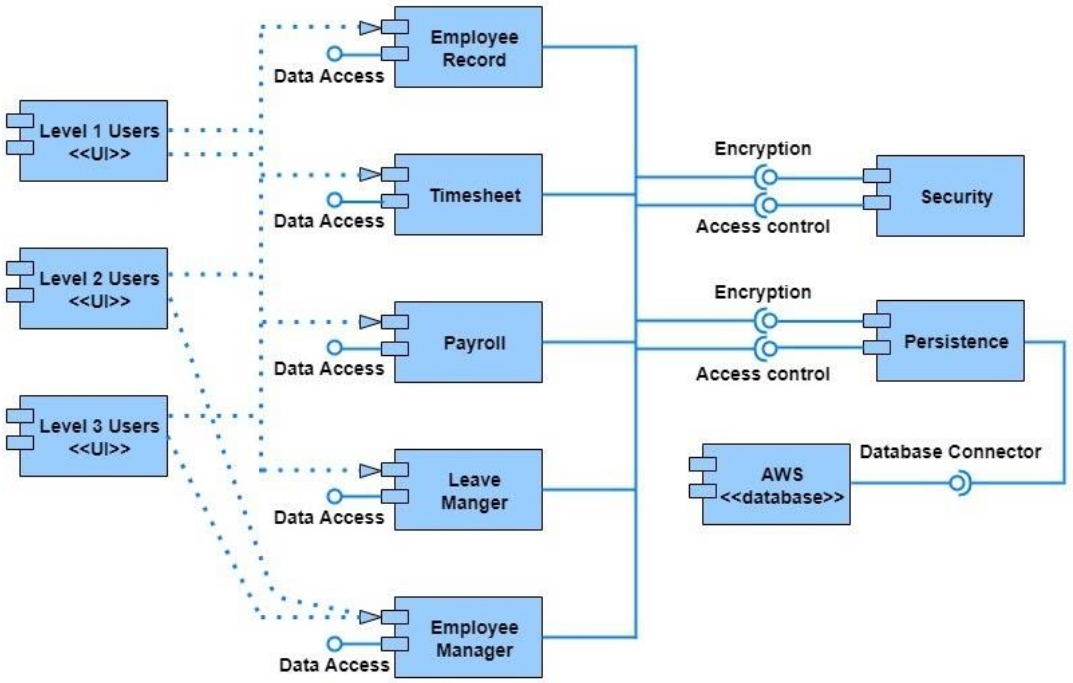
User safety
Data safety

Security requirements

reCAPTCHA verification
2-factor Authentication

Software Quality Requirements

Availability
Reliability
Correctness
Modifiability
Usability



Functionalities of the system (SEMS)

Successful authentication of the system

Maintaining / Monitoring timesheet log

Updating employee's records

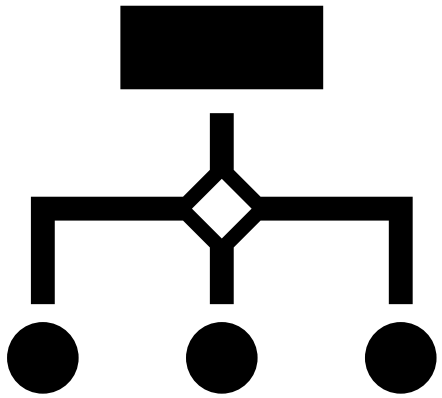
Applying / approving for leave request

Generating payroll

Adding / Removing employees

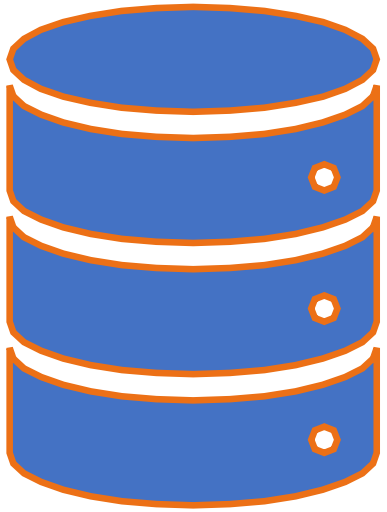
Sending user notifications

Resetting Password



Chosen Architecture & Comparison

Client-Server Model



- Efficient use of distributing power than file servers.
- No need for large data transfers.
- Easy to add new servers or upgrade existing servers.

Model View Controller

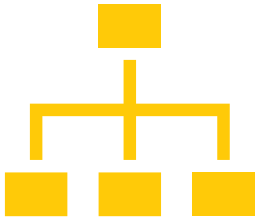
Model View Controller

Data manipulation independent of data representation.

Multiple formats of data presentation.

Easy collaboration.

Improves the response time for the user.



Layered Architecture

Each tier is managed by different machine.

Easy collaboration.

Easy to debug since each tier is independent of another

Service-oriented Architecture

Service Oriented Architecture

Language independent, allows for easier integration and maintenance

Developed system ready to be integrated

Reduction in developing time to create equivalent security measure

Fairly reliable service

Model View Controller

Slower development time

Difficult to reproduce similar results in a reasonable time frame

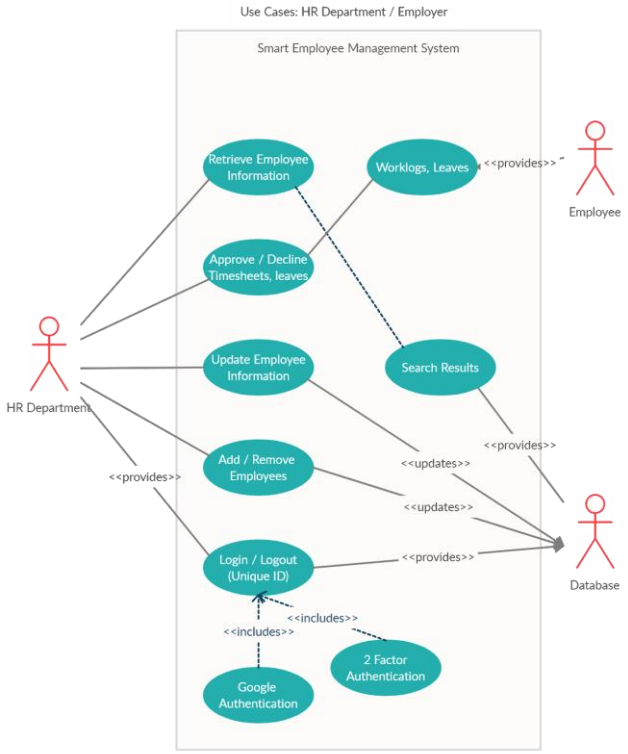
Use Cases:
Employer
HR Department

An Employer/HR Department can log in using unique credentials.

Employers/HR can add or remove employees from the database.

Employers/HR can approve/decline timesheets, invoices and leave requests.

Employers/HR have privileges to update employee information.



Use Cases: Employee

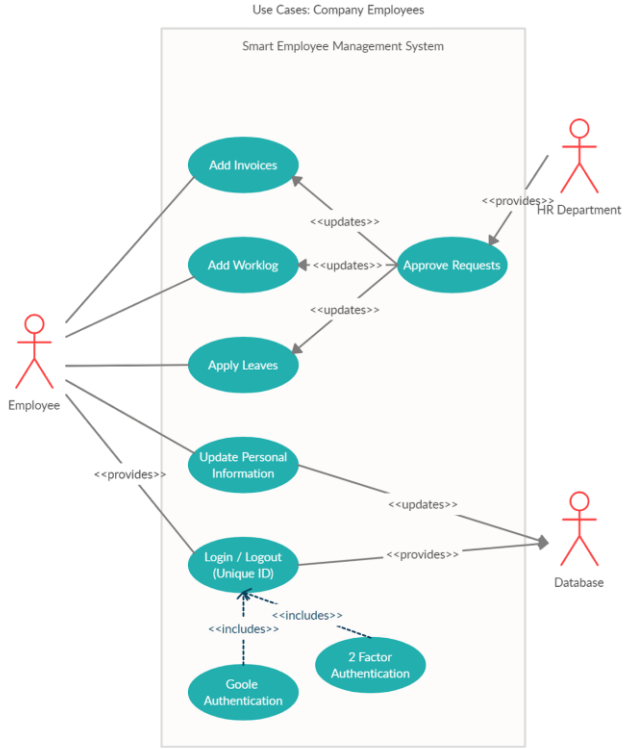
Employees can login to the system using unique credentials.

Employees can apply for leaves, absences to the Employer/HR.

Employees can view, update their worklog and payroll information.

Employees can update their personal information and login credentials.

Employees can raise invoices to Employer/HR.



Use Cases:
Data Base System

Database has access to all the user and system information.

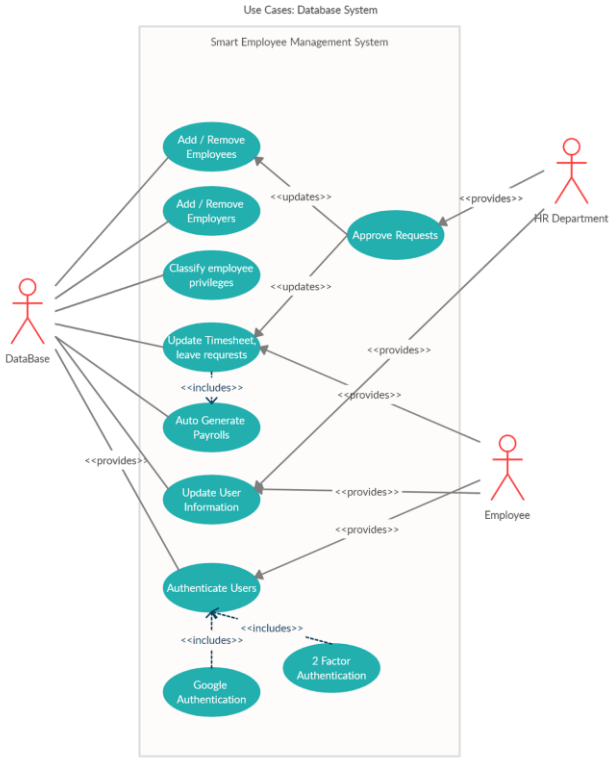
Database can add/remove/update employee information.

Database can add/remove/update Employer/HR information.

Database can automatically calculate and generate employee payrolls.

Database can add/remove/update leaves, timesheets from the employees.

Database can update user privileges.



Use Cases: Developer & Designer

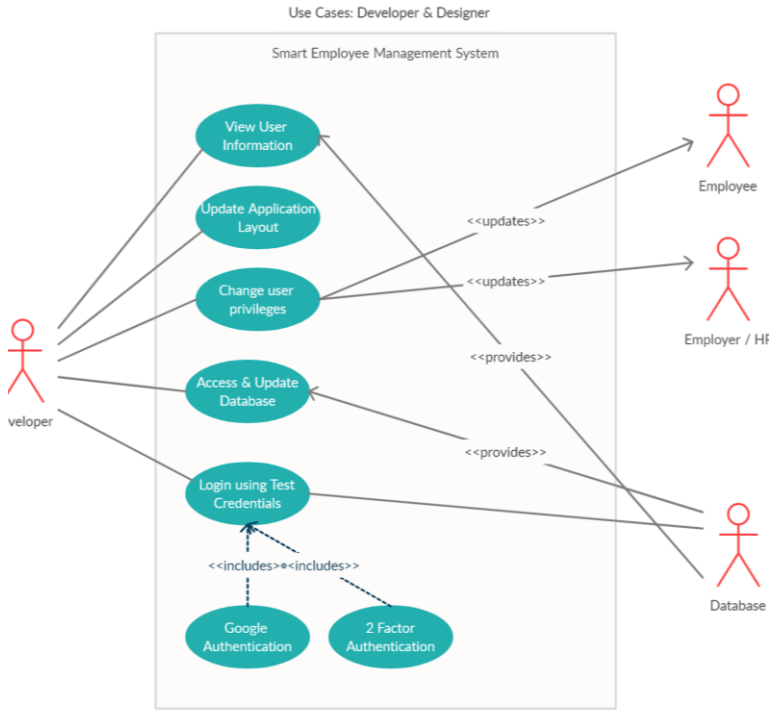
Developers can view user information and system status.

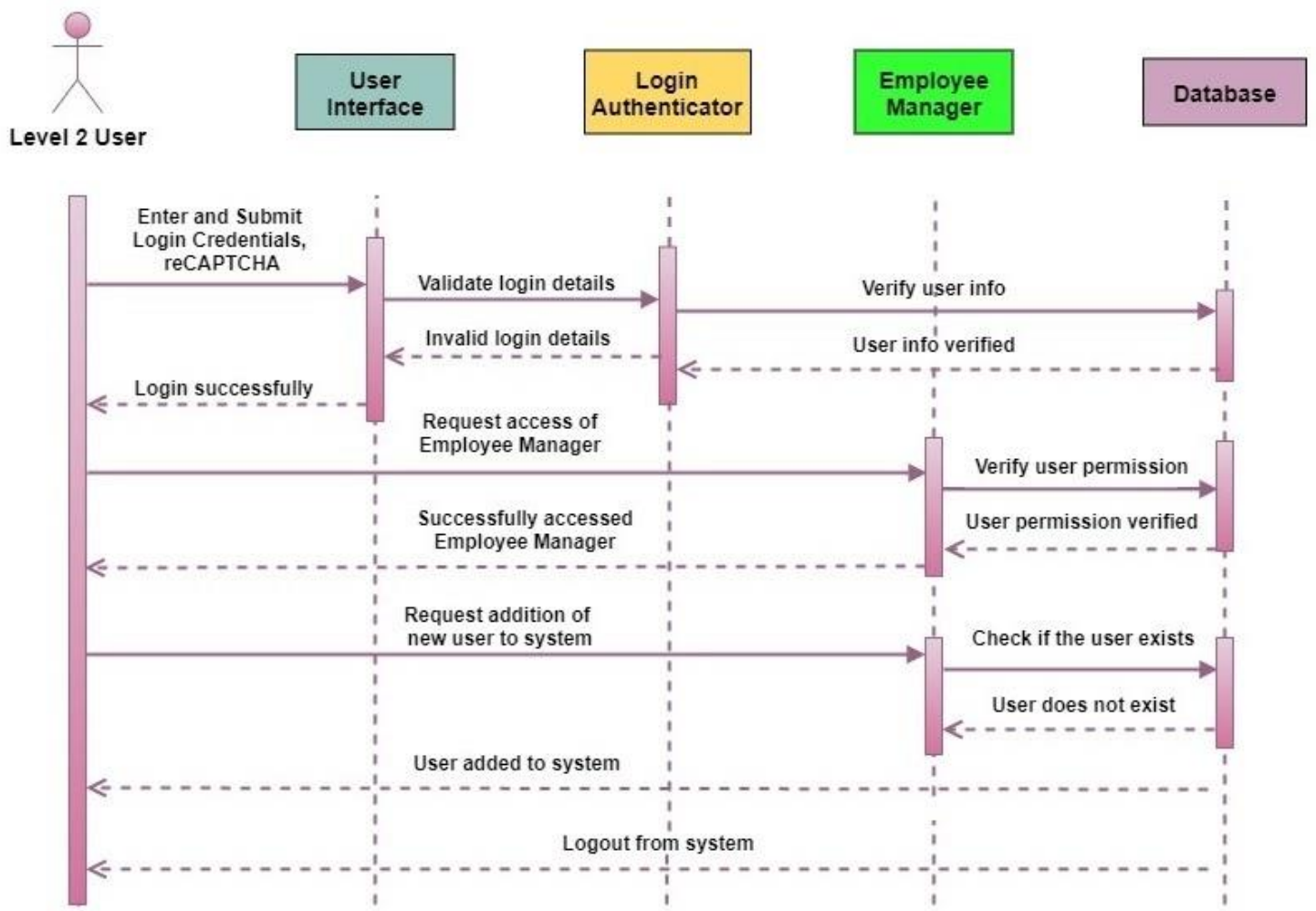
Developers can update system layout, working and properties without breaking the current system.

Developers has full access to the database system.

Developer/Designer can login to the system as a test user.

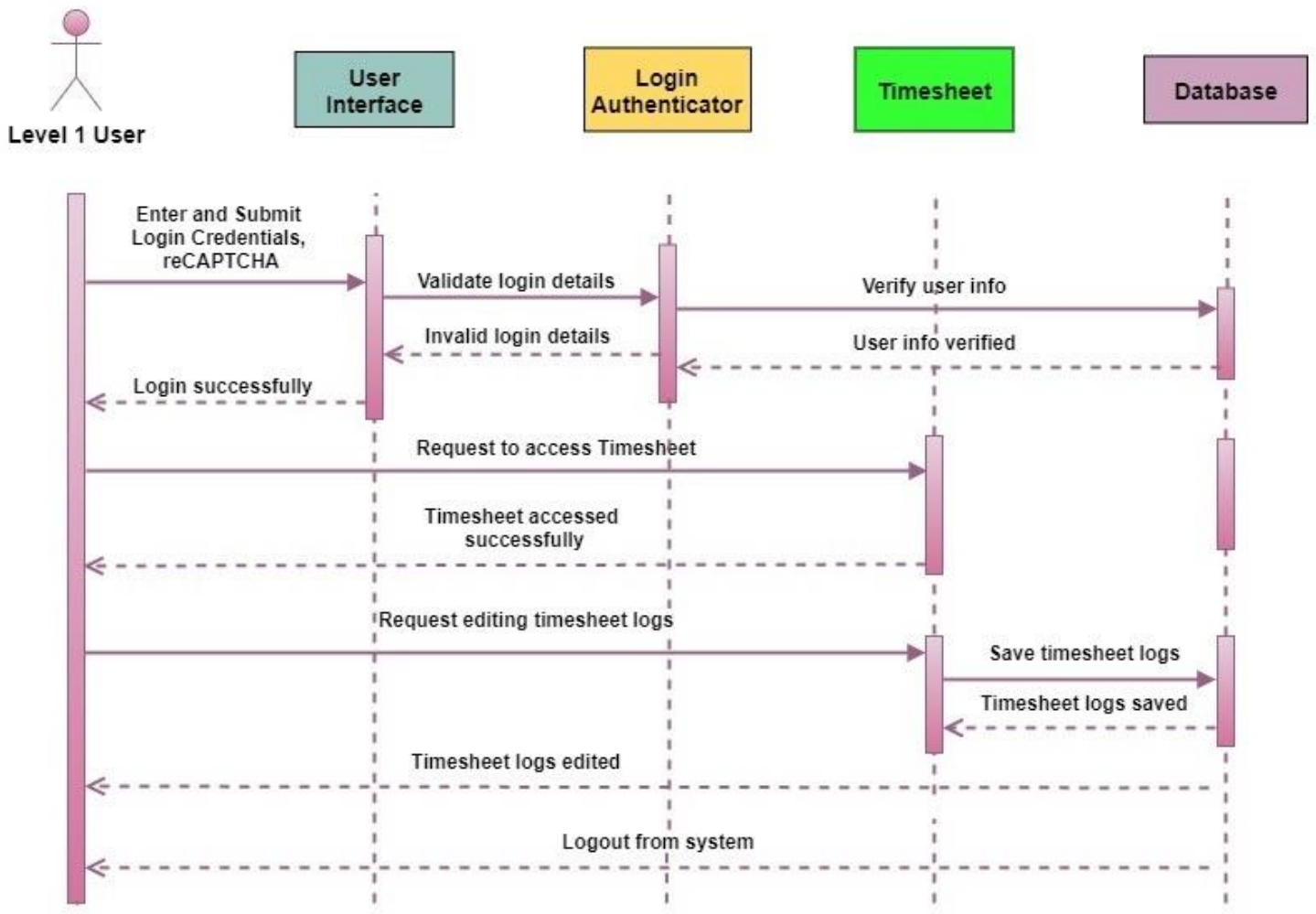
Developers can update user privileges.





Sequence Diagram

Creating new user.
Login / Log out.

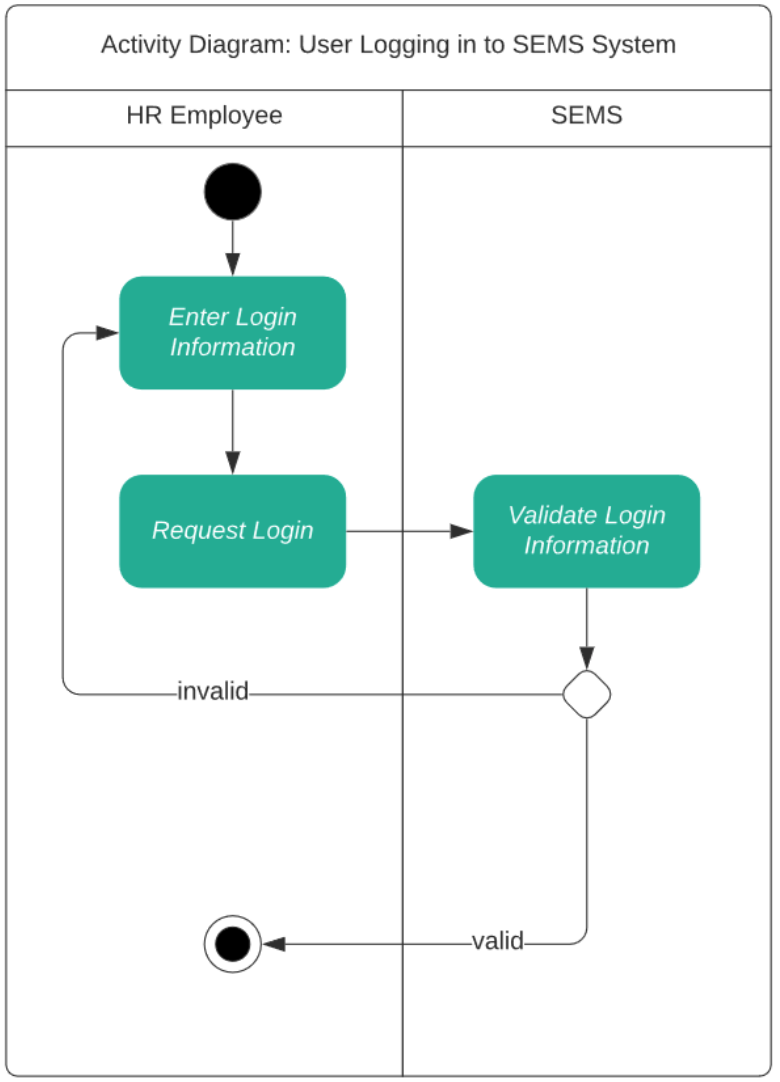


Sequence Diagram

Employee adding/updating timesheets.

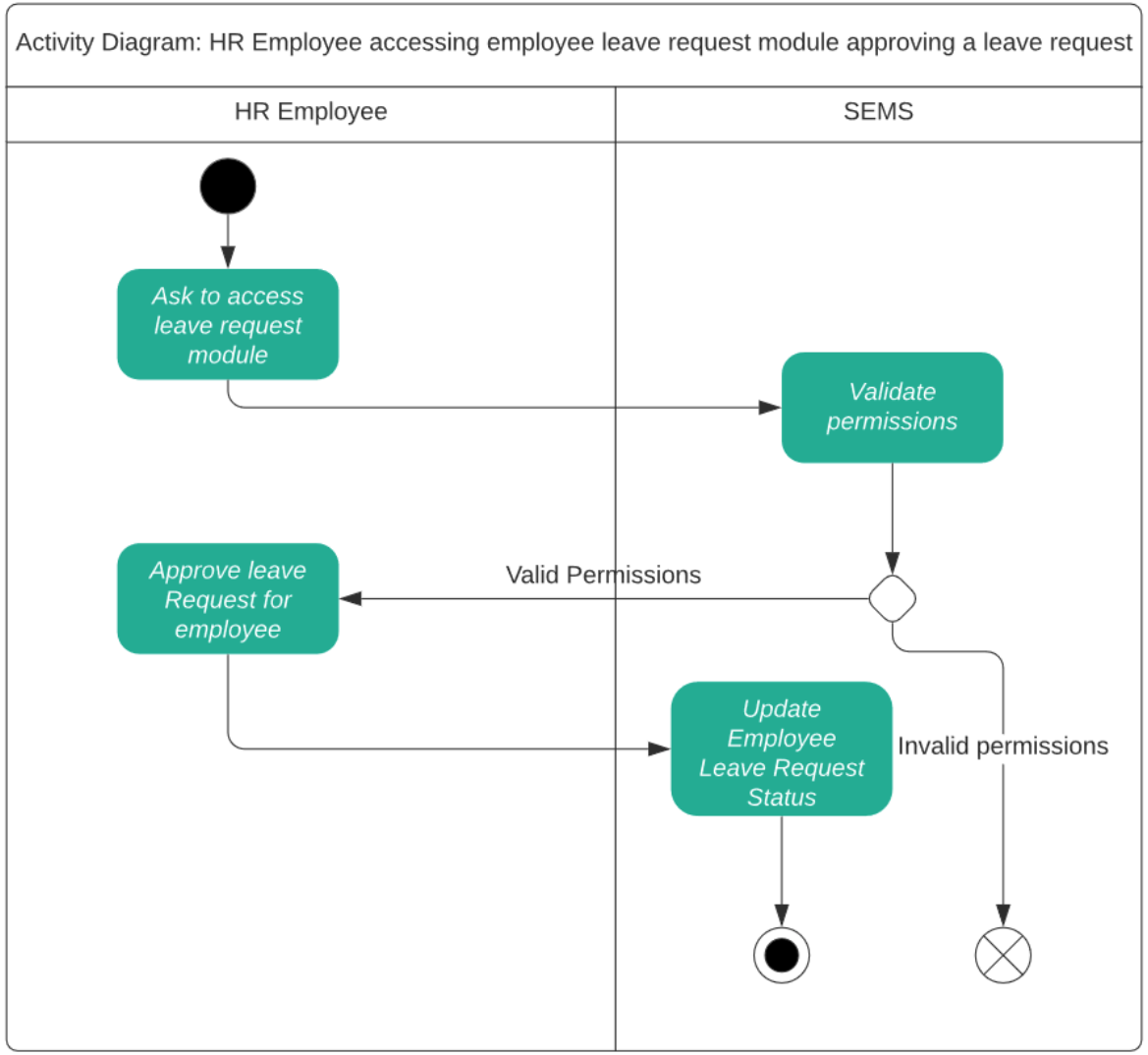
Activity Diagram

Employer / HR
Logging in the system.



Activity Diagram

Employer / HR accessing employee leave request module and update leave requests.



Concurrency in SEMS

User Concurrency

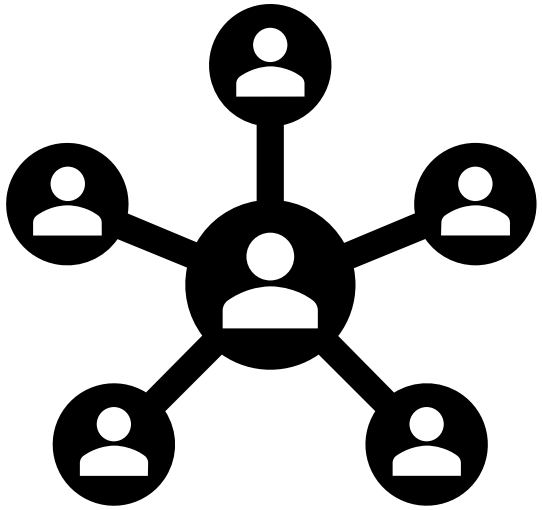
- Serving multiple users at the same time

Component Concurrency

- A component can be used by many users simultaneously

Example:

- Multiple users should be able to login to the system simultaneously and access the different components of the system while others are also using them



Learned Lessons

Architecture can get complex when layering it with other architectures

Software systems have many users for which you need to examine the different ways in which they interact with the system

Understanding the flow of data when examining use cases is an important part of understanding how the system functions

Different diagrams can help understand how data flows within a system