Smart Employee Management System (SEMS)

TEAM PENTAD X-5

Overview

- High Level Architecture
- SOA and Third-party Services / COTS
- Quality attributes
- Sub-system
- Use cases
- Diagrams
- Concurrency
- Lessons Learned
- Demo of SEMS

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

High Level Architecture

3 Tier

Presentation Layer (Client Tier)

Application Layer (Business Tier)

Database Layer (Data Tier)

Model View Controller

Model – Access to Database

View – Main Interface

Controller – Handle User Interaction and Model Update

Service Oriented Architecture

Google Re-captcha

Google O-Auth

Firebase Admin SDK

Firebase Database API

Firebase Hosting Service

Software Quality Attributes

Availability

Reliability

Correctness

Modifiability

Usability

Availability

SEMS shall achieve 99.5% uptime and SEMS shall inform users of any planned unavailability such as system maintenance.

These were both achieved through being hosted by the firebase server which in their Service Commitment agree says at least 99.95% uptime percentage [1].

Reliability

SEMS shall not corrupt or delete user’s data and any update of user records shall be saved effectively to the database.

This is achieved by using firebase’s database that has multiple servers that hosts our data.

Correctness

SEMS shall not allow the ability to run any defective source codes that can cause storing wrong data, performing wrong calculations, or initiating infinite loops causing a crash of the system.

The design tactic used to achieve this is MVC model that allows isolation of computing to the controller and single responsibility design principle for our services.

Modifiability

SEMS shall allow users to add new features or delete existing features without discontinuing the services.

For example, SEMS shall allow Employers to add new employees in the system or remove an existing employee without affecting the functionality of other users.

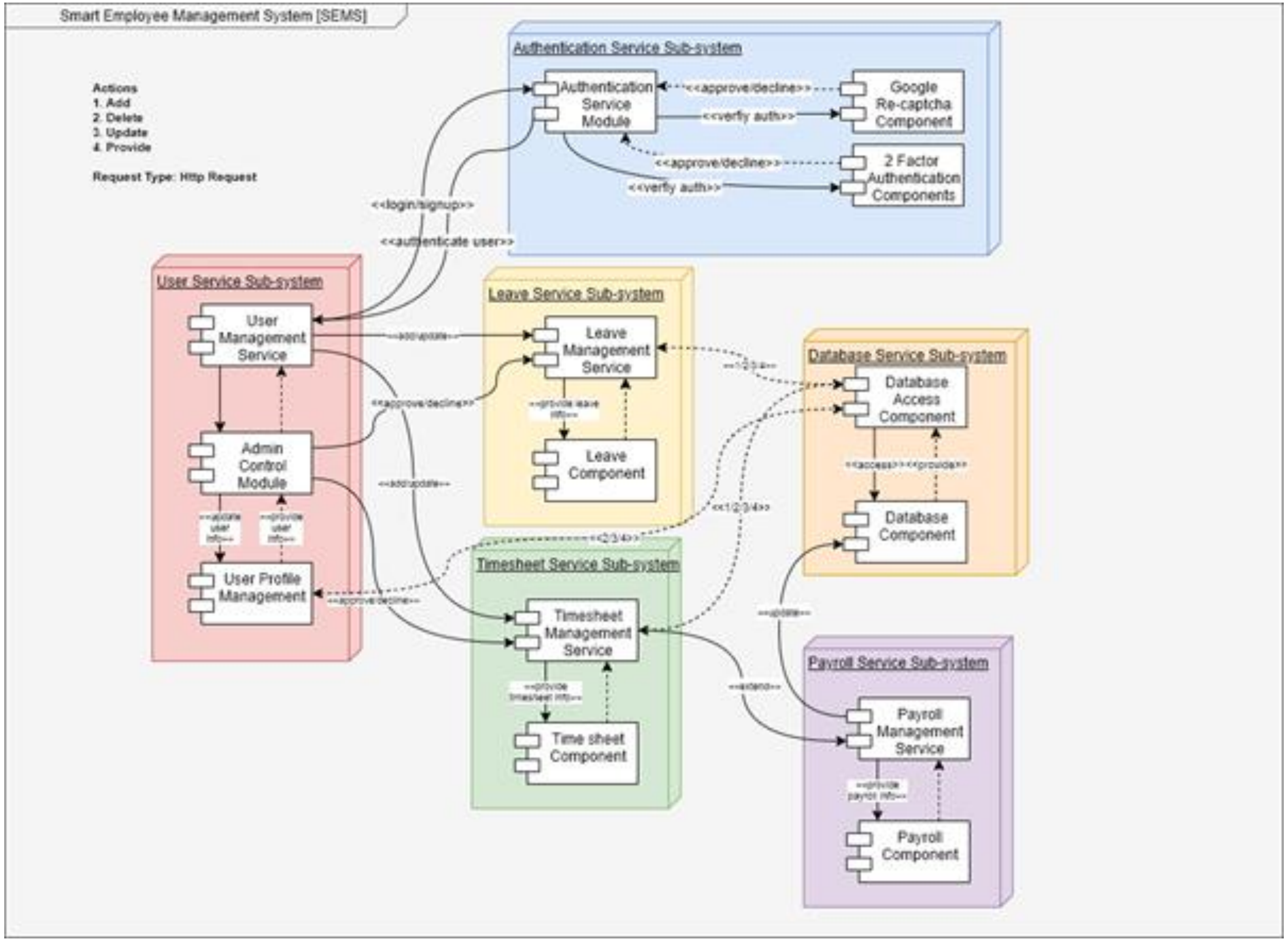
Usability

SEMS shall support operations invariably in different operating systems or web browsers as well as in multiple platforms including Computer or Mobile devices and shall be accessible remotely. The design tactic chosen was to create a website that is supported by the internet.

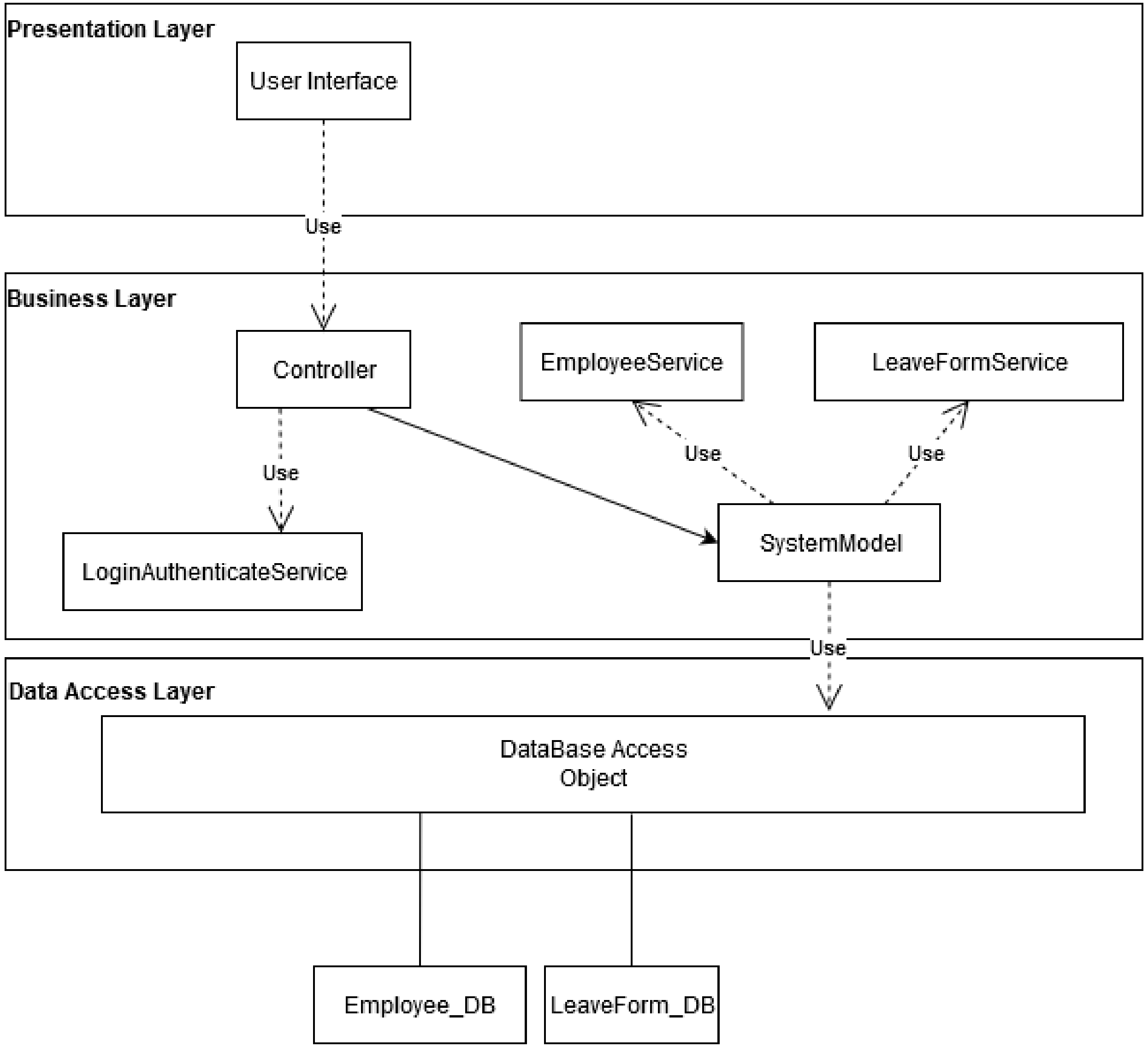
SEMS shall be easy to use and easy to learn for the users. The design tactic used was to focus on the functionality and button visibility such as timesheet and leave applications.

Sub-System

Major Subsystem Interaction



Sub-System of 2 use cases



Use Cases: Employer HR Department Update Employee Information

Precondition:

- The HR Department has logged into the system using a valid username and password.

Success Post condition:

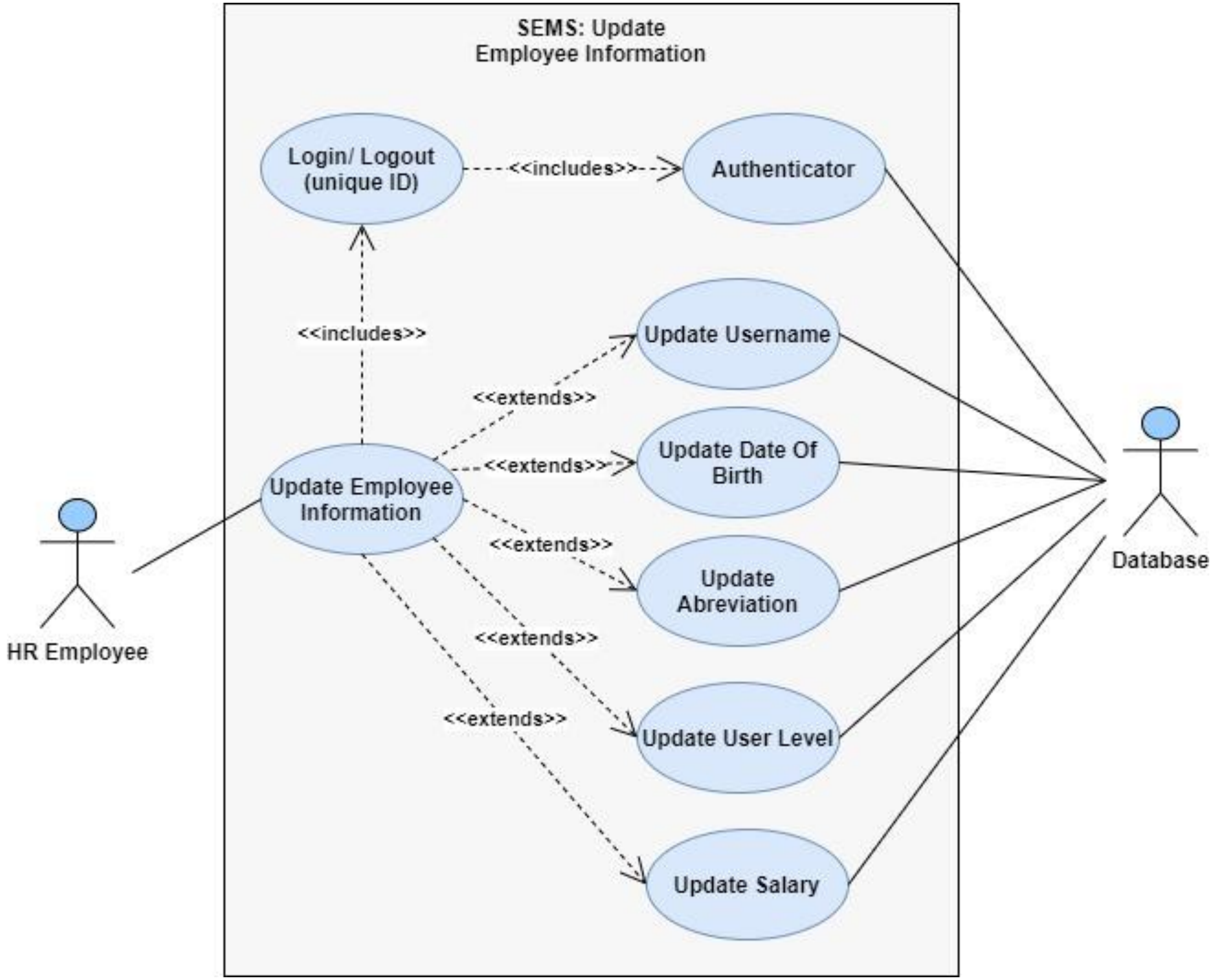
- An updated employee profile.

Main Success Scenario:

- The HR Department selects “Employee” from the main menu.

Alternative Flow:

- 1a. The HR Department updates Username.
- 1b. The HR Department updates Address.
- 1c. The HR Department updates Profile Picture.



Use Cases: Employee Submit Leave

Precondition:

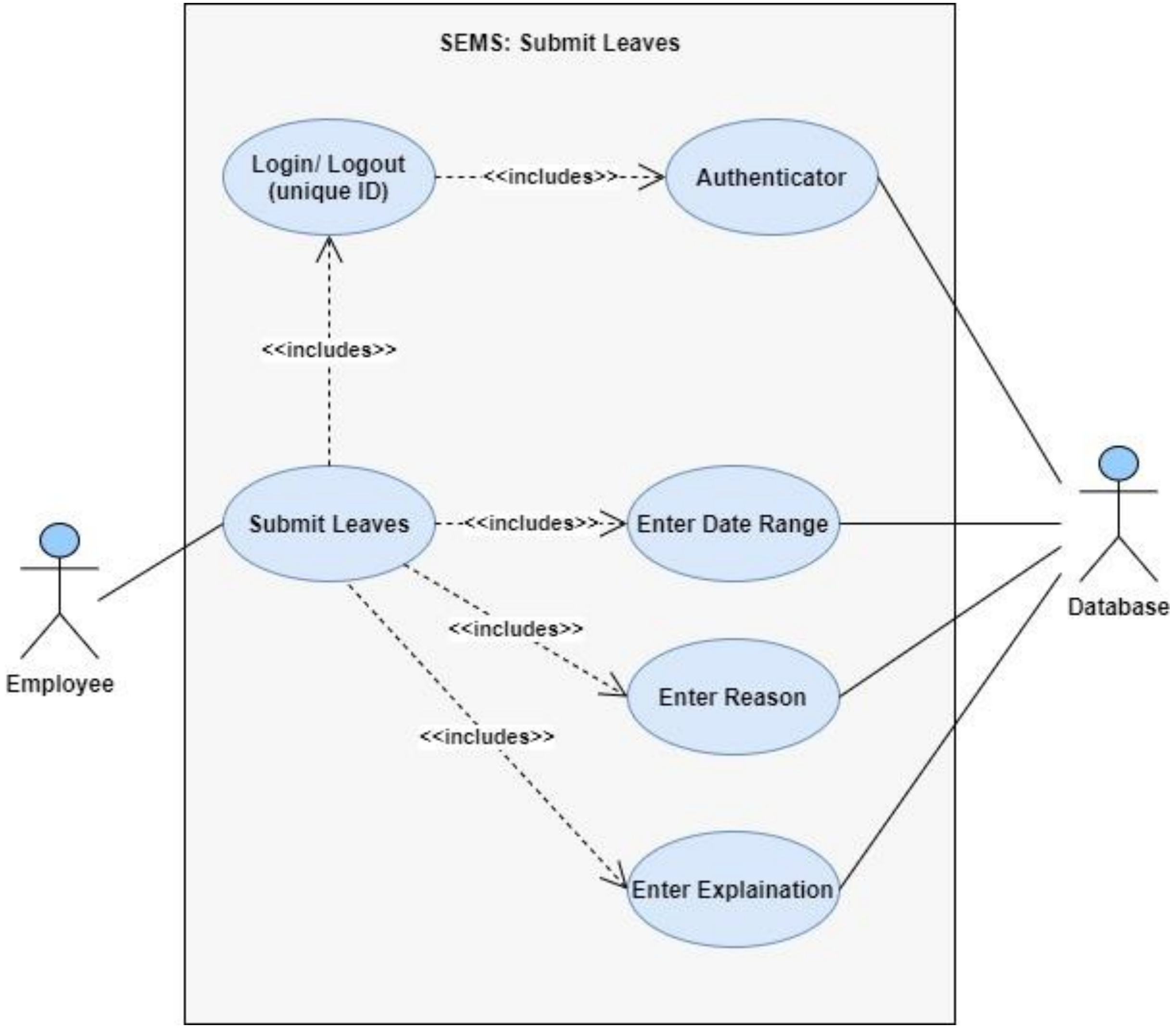
- The employee has logged into the system using a valid username and password.

Success Post condition:

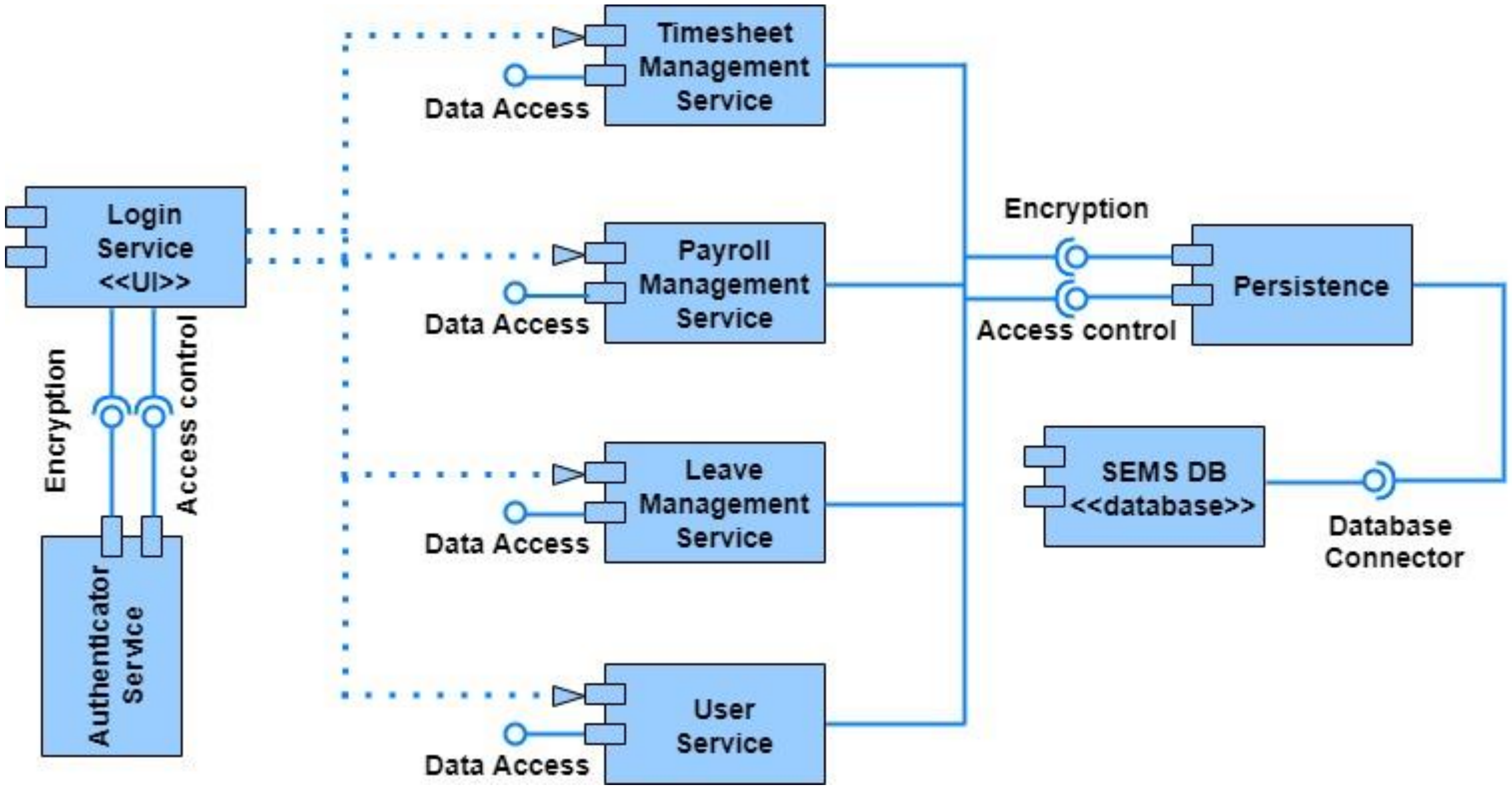
- A successfully submitted leave application.

Main Success Scenario:

- The Employee selects “Leaves Tab” from the main menu.
- The Employee enters the data range.
- The Employee enters the reason.
- The Employee enters the explanation.

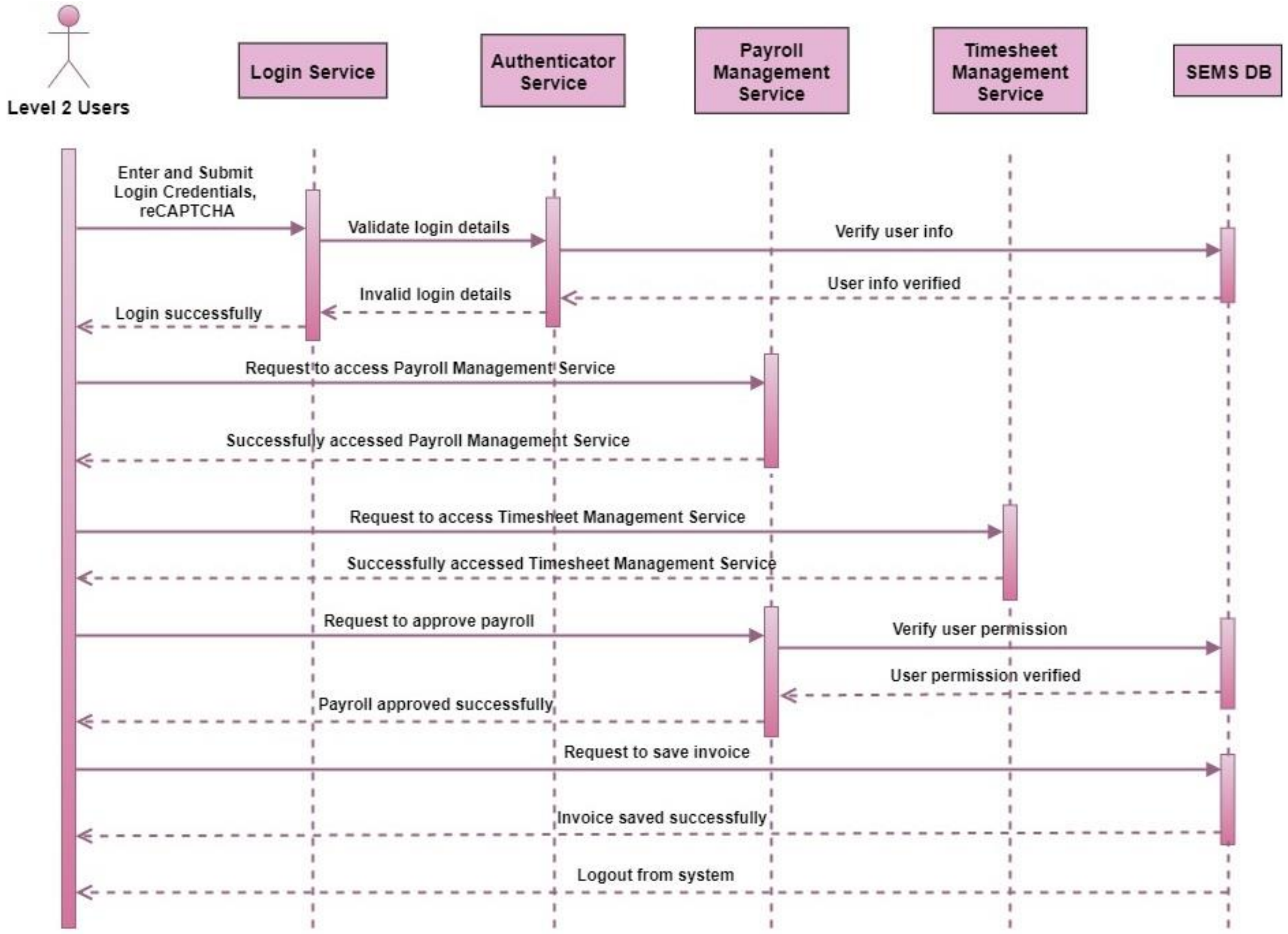


Component Diagram:

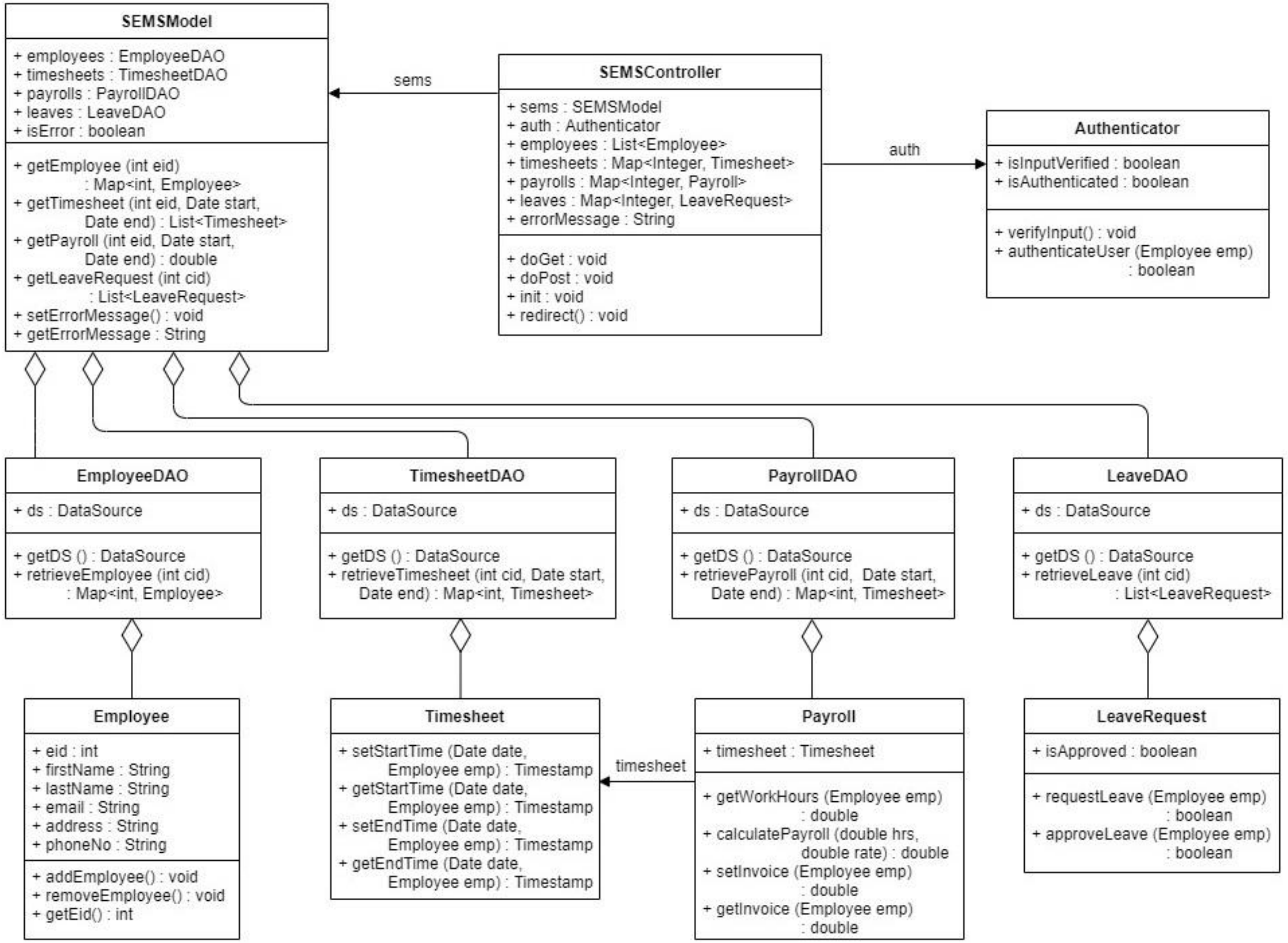


Sequence Diagram:

Level 2 user (HR employee) approving payroll


Level 2 Users
Login Service
Authenticator Service
Payroll Management Service
Timesheet Management Service
SEMS DB
Enter and Submit Login Credentials, reCAPTCHA
Validate login details
Verify user info
User info verified
Invalid login details
Login successfully
Request to access Payroll Management Service
Successfully accessed Payroll Management Service
Request to access Timesheet Management Service
Successfully accessed Timesheet Management Service
Request to approve payroll
Verify user permission
User permission verified
Payroll approved successfully
Request to save invoice
Invoice saved successfully
Logout from system

Class Diagram:



SEMSModel

+ employees : EmployeeDAO
+ timesheets : TimesheetDAO
+ payrolls : PayrollDAO
+ leaves : LeaveDAO
+ isError : boolean

+ getEmployee (int eid) : Map<int, Employee>
+ getTimesheet (int eid, Date start, Date end) : List<Timesheet>
+ getPayroll (int eid, Date start, Date end) : double
+ getLeaveRequest (int cid) : List<LeaveRequest>
+ setErrorMessage() : void
+ getErrorMessage : String

SEMSController

+ sems : SEMSModel
+ auth : Authenticator
+ employees : List<Employee>
+ timesheets : Map<Integer, Timesheet>
+ payrolls : Map<Integer, Payroll>
+ leaves : Map<Integer, LeaveRequest>
+ errorMessage : String

+ doGet : void
+ doPost : void
+ init : void
+ redirect() : void

Authenticator

+ isInputVerified : boolean
+ isAuthenticated : boolean

+ verifyInput() : void
+ authenticateUser (Employee emp) : boolean

EmployeeDAO

+ ds : DataSource

+ getDS () : DataSource
+ retrieveEmployee (int cid) : Map<int, Employee>

TimesheetDAO

+ ds : DataSource

+ getDS () : DataSource
+ retrieveTimesheet (int cid, Date start, Date end) : Map<int, Timesheet>

PayrollDAO

+ ds : DataSource

+ getDS () : DataSource
+ retrievePayroll (int cid, Date start, Date end) : Map<int, Timesheet>

LeaveDAO

+ ds : DataSource

+ getDS () : DataSource
+ retrieveLeave (int cid) : List<LeaveRequest>

Employee

+ eid : int
+ firstName : String
+ lastName : String
+ email : String
+ address : String
+ phoneNo : String

+ addEmployee() : void
+ removeEmployee() : void
+ getEid() : int

Timesheet

+ setStartTime (Date date, Employee emp) : Timestamp
+ getStartTime (Date date, Employee emp) : Timestamp
+ setEndTime (Date date, Employee emp) : Timestamp
+ getEndTime (Date date, Employee emp) : Timestamp

Payroll

+ timesheet : Timesheet

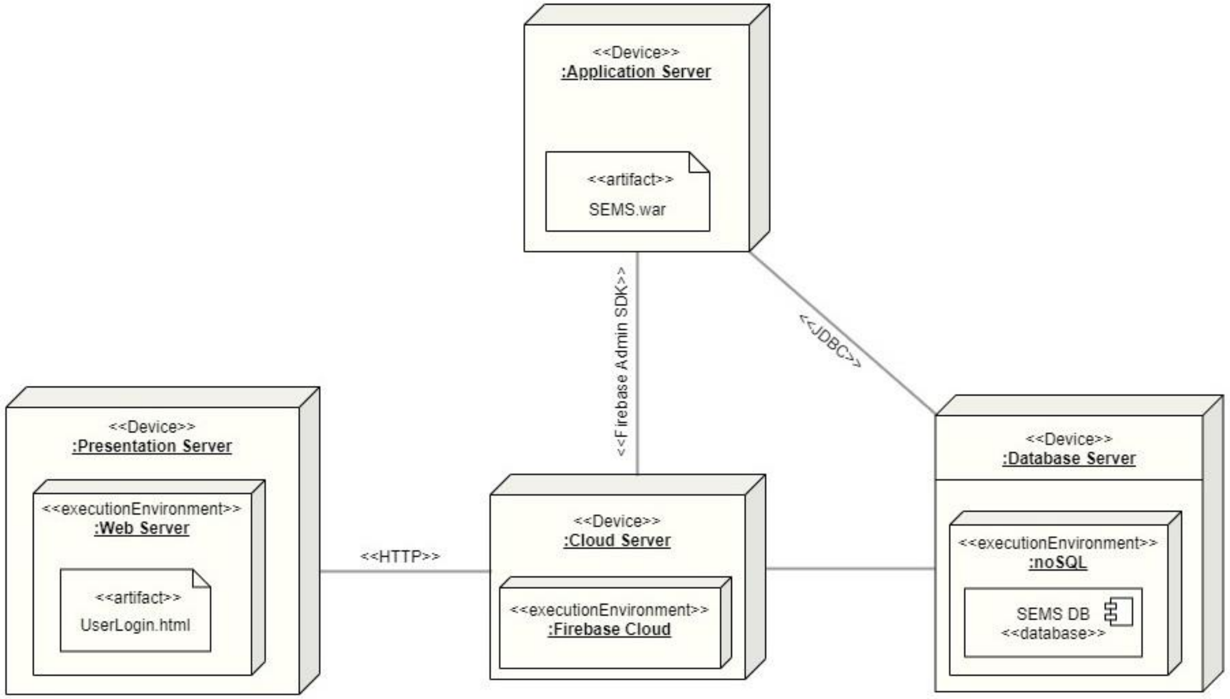
+ getWorkHours (Employee emp) : double
+ calculatePayroll (double hrs, double rate) : double
+ setInvoice (Employee emp) : double
+ getInvoice (Employee emp) : double

LeaveRequest

+ isApproved : boolean

+ requestLeave (Employee emp) : boolean
+ approveLeave (Employee emp) : boolean

Deployment Diagram:



Concurrency in SEMS

User Concurrency

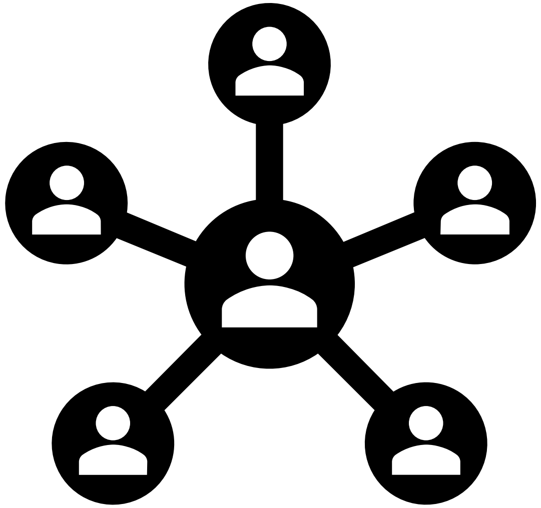
- Serving multiple users at the same time

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



Updates to Conceptual Architecture

Switched from 2-tier client server to 3-tier client server

Changed how subsystems interact with each other to better depict their relationships

Concrete VS Conceptual

Concrete

Actual implementation of what was conceived in conceptual architecture

See how the sub-systems actually interact with each other and what dependencies exist

Conceptual

Ignored implementations details to focus on functions of the system

Doesn't depict how sub-system actually interacts with each other

Learned Lessons

Importance of Conceptual Architecture

Importance of Concrete Architecture

How the two work together to better help understand the overall architecture of the system

Violations can occur between the two as such changes can be made to make them more aligned