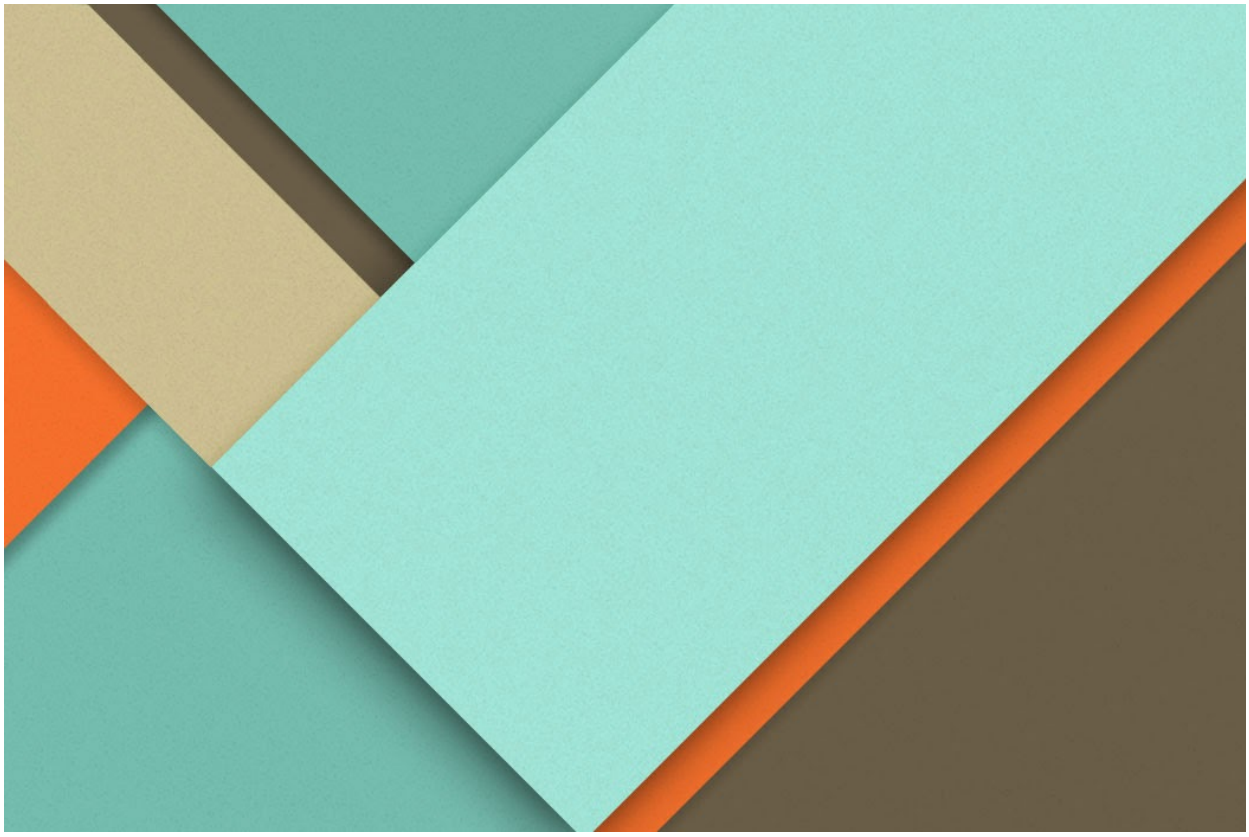

Smart Employee Management System [SEMS]

21.10.2020

Muhammad Arifur Rahman
Samuel On
Jaiveer Singh
Dong Jae Lee
Karmit Patel

Website: http://pentad-x5.unaux.com/

Github: https://github.com/arifrahmanca/Smart_Employee_Management_System

Overview

The primary goal of this document is to provide a comprehensive description of the high-level architecture and an accurate list of requirements for the Smart Employee Management System or SEMS for short. SEMS is a cloud-based software application designed to reproduce the traditional employee management system in a virtual environment. Requirements for each user group are defined and distinct non-functional requirements are described in detail. Along with the requirements, this document will cover system components such as Amazon Web Services (AWS) and their interactions with other components within the system. Different architecture styles and design patterns used for the system are mentioned and a comparison with other architecture patterns that are considered less suitable is made. Multiple diagrams will be used to illustrate the user and component interaction within the system.

User Groups

Smart Employee Management System or SEMS, has multiple types of users each with varying levels of privileges for segregation within the system. The four main types of users that exist within SEMS are regular employees, HR employees, the employer, and developers. Along with having different types of users, the system also maintains three different levels of permissions. Where higher permission levels also include the privileges granted to the lower ones.

Level 1 Users:

The level one users are the primary users of the system with the least amount of privileges. This level of the user includes regular employees and they have access to basic functionalities such as logging in, changing personal information, applying for leaves and holidays, and also submit timesheets and invoices for their work hours. Any information that isn't personal or related to the employee themselves will not be accessible to the employees to be viewed or rather modified.

Level 2 Users:

Employees that are a part of the human resources department will be classified as level two users of the system. Level two users will be granted more access on top of the privileges that are already granted to level one users. They will be capable of accessing any information submitted by the level one user and also have access to add or delete users in the database. Users with level two privileges will act as an admin for regular employees by monitoring information submitted and approving or declining them.

Level 3 Users:

Above the level two users, the system will have users with level three privileges that act as an admin for the whole system. The types of users that are granted these privileges will be the developers and designers of the system. Although developers of the system will have

access to all privileges of level two and level one user, in most scenarios those privileges will only be utilized in testing or updating the system.

Functional Requirements

Employee (Level 1 Users)

- Any employee should be able to login and logout using their provided credentials.
- Any employee should be able to update their personal information such as date of birth, name, etc.
- Employees can update their timesheet and invoice anytime before submission.
- Employees can submit an invoice with a timesheet of logged hours for the week.
- Employees can apply for holidays and leaves.
- Any employee should be able to reset their password used for login.
- Employees can view their previous timesheets and invoices dating back to a year prior.

HR Department and Employer (Level 2 Users)

- Any Human Resources (HR) employee should be able to login and logout using their provided credentials
- HR employees can retrieve information such as an employee's timesheet, payroll, and username from the database.
- HR employees can approve or decline timesheets submitted by regular employees.
- HR employees can approve or decline holidays and leave applications from employees.
- HR employees can add employees to the database.
- HR employees can delete registered employees from the database.
- Any HR employee should be able to reset their password used for login.

Designer & Developer (Level 3 Users)

- Developers should be able to login to the system using credentials for a test account.
- Developers should be able to change the level of privileges while logged onto the test account.
- Developers should be able to update parts of the system or database without affecting other parts of the system.
- Developers should be able to modify different views available on the system.
- Developers should be able to see all information in the database.
- Developers should be able to modify any information within the database.

Database

- The database should be able to update employee information.
- The database should be able to add and register employees onto the database.
- The database should be able to remove registered employees off the database.
- The database should be able to classify employees based on their departments.

- The database should be able to classify employees based on their privileges in the database
- The database should be able to update timesheets and invoices.
- The database should be able to process leave requests.
- The database should be able to automatically generate payrolls based on the submitted invoices and timesheets.
- The database should be able to allow access to data depending on the given user's privilege.
- The database should be able to reset passwords set by any user.

Properties

- The login system should utilize the two-factor authentication system provided by Google Authenticator.
- The login system should utilize the reCAPTCHA system provided by Google.
- Every registered user will have a unique identification number within the database.

Non-Functional Requirements

Performance requirements:

- Response Time: On average, SEMS shall be able to process a request within 2 seconds or less in each 5-minute period. 95% of all response times shall be less than 5 seconds.
- Throughput: SEMS shall be able to process 10,000 requests per hour.
- Scalability: SEMS shall be able to handle requests from 1,000 concurrent users at any point of time.

Safety requirements:

- While accessing SEMS over the web browser or mobile app, SEMS should not cause any harm to the user's device/computer such as downloading viruses or malware and causing malfunctioning of the device.
- SEMS shall ensure that all data is stored safely in the database and take precautions to avoid any kind of accidental data loss.

Security requirements:

<u>reCAPTCHA verification:</u>

- SEMS will require the users to enter a decipher test obtained and verified by Google reCAPTCHA API before they can gain access to their accounts to provide protection against automated attacks.

<u>Two-factor authentication:</u>

- SEMS will allow users to activate two-factor authentication to verify themselves to prevent unauthorized access to their accounts. After submitting the login details

(User ID and Password), they will be asked to provide another piece of information, for example, “one-time code” sent to Email or cell phone number.

Software quality attributes:

Availability:

- SEMS shall be available for use 24x7x365 and shall achieve 99.5% uptime.
- SEMS shall inform users of any planned unavailability such as system maintenance.

Reliability:

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

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.
- SEMS shall provide real-time data to the end-users.

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.
- SEMS shall be easy to use and easy to learn for the users.

Functionalities of the system:

1. Successful authentication of the system: Upon entering the accurate login credentials and reCAPTCHA to the user interface, the UI shall communicate with the Database to authenticate a user login and Google reCAPTCHA API (Security component) shall return successful validation before a user can log in into the system. In scenarios where the user chooses 2-factor authentication (another Security component), the users will be required to provide a “one-time passcode” for logging.
2. Maintaining / Monitoring timesheet log: Here, the User component interacts with the Timesheet component to access and edit the timesheet logs. The Timesheet component interacts with the Database component to display and save timesheet logs of users.
3. Modifying employee records: The User interacts with the Employee Record component to view and edit employee’s personal data and finally, Employee Record interacts with the Database to save the data.

4. Applying / approving for leave request: The Level 1 User interacts with the Leave Manager component to request for leave and Leave Manager interacts with the Database to save the request. Similarly, Level 2 User interacts with the Leave Manager to view and approve the employee's leave request and save it to the Database.
5. Preparing payroll: The system will automatically generate payroll for employees and employees shall be able to view their pay stubs. Level 2 User interacts with the Payroll component to approve employee's pay stubs upon successful user verification from the Database. Level 1 User can also interact with the Payroll to view their pay stubs.
6. Adding / Removing employees: Employers/HR department shall be able to add new employees in the system and save it to the database as well as remove an existing employee from the system. Here, Level 2 User interacts with the Employee Manager component which further interacts with the Database.
7. Sending notifications to users: The system shall notify users for attempting access from unrecognized devices or unusual locations and shall deliver email and text messages for any change in their profile including password. The UI interacts with the Security and the Database components to check and verify user login info.
8. Resetting Password: The system shall allow users to reset passwords if any user forgets his/her password. The UI verifies user identity (e.g. username, security questions) from Database and interacts with the Security (2-factor authentication, "one-time passcode") to acquire and reset the password and finally interacts with the Database to save the password.

Comparison to Existing Solutions

Compared to many of the products out there such as Sap,15five, Gusto, WordDay, KissFlow, Saba that incorporate many different purposes for the application, it also increases the complexity. A few examples are:

Sap involves multiple different tools such as employee surveys, role-based dashboards and reporting, customizable impact reports, event-based triggers, and employee rewards programs. 15five provides engaging work culture through weekly check-ins, dedicated 1-on-1s, self-review tools, and a recognition system. Saba focuses on ways to decrease turnover, optimize the workforce organization, prepare succession planning, and build the employer brand in a way that attracts top-quality talent.

For our system we have the following 2-factor authentication, timesheet maintenance, payroll processing, and time request, employee HR records. In our case, we keep our system simple and easy to use. It reduces the learning curve and frustration with the system. By focusing on a few critical features for HR services, this allows us to ensure they are functioning at all times and maintainable for a smaller group of developers and the pricing of our system will reflect this.

Architecture

The system will be using a database and a domain supplied by AWS to provide database services to the system through the Client-Server Model. This model is chosen for the database because it will be supplying many clients throughout the system. The advantages of this model is that query processing is done by the database, no need for large data transfers, more efficient use of distributing power than file servers, and it's easy to add new servers and upgrade existing. These are a few ideal benefits since SEMS will require storing records of employees while requiring SQL power and 24/7 availability.

The system will also be using a Model View Controller architecture to provide the interface for modifying data that connects to our database. This model is chosen since there will be many users viewing the data, and the requirements for future interactions with data are unknown. The model allows us to organize the operations into different levels: model, view and controller. The advantages are that the development of the applications is faster, collaboration is made easier, the experience for the user is faster, allowing data to change independently of its representation and vice versa.

The system will also be utilizing third-party multi-factor authentication and ReCaptcha service using Google API and other REST APIs via Service Oriented Architecture. (SOA) to help secure the system. The advantage of this is that it allows for easy integration and maintenance while reducing the amount of time to develop an equivalent security measure.

Analysis and Comparison of Alternative Architectures

When choosing the database server, there were no alternative architectures considered as the client-server architecture did exactly the purpose we wanted. It allows us to effectively query information of our clients and return the information quickly. The downside is that because it's distributed, it is prone to Denial of Service Attack, however, this can be mitigated with security measures. The pros outweigh the cons.

When choosing the model-view-controller to drive our user experience, the alternative architecture considered a similar version of MVC such as n-tier architecture. N-architecture presents n-tiers of dividing the application, such as logic tier, presentation tier, and data tier. Where there is a middle layer or logic tier that provides all communication between each tier, while the MVC architecture uses the control later to access the model and view layers. MVC architecture is stateful and so the control layer creates the model from the requirements and pushes the model to the view layer. The MVC architecture is more than enough to drive our user experience without involving more machines to handle the different tiers of the n-tier architecture.

When choosing the SOA architecture, the consideration that took place was whether to implement it ourselves or have another entity use it. Our alternative architecture is MVC since it allows us to have a middle man between the data and the user through the view

interface. We decided that it was much easier to go with SOA architecture as it allowed us to integrate it easier and be able to incorporate a well-made product that reduces the amount of time to develop.

Subsystems

Sub-system interaction

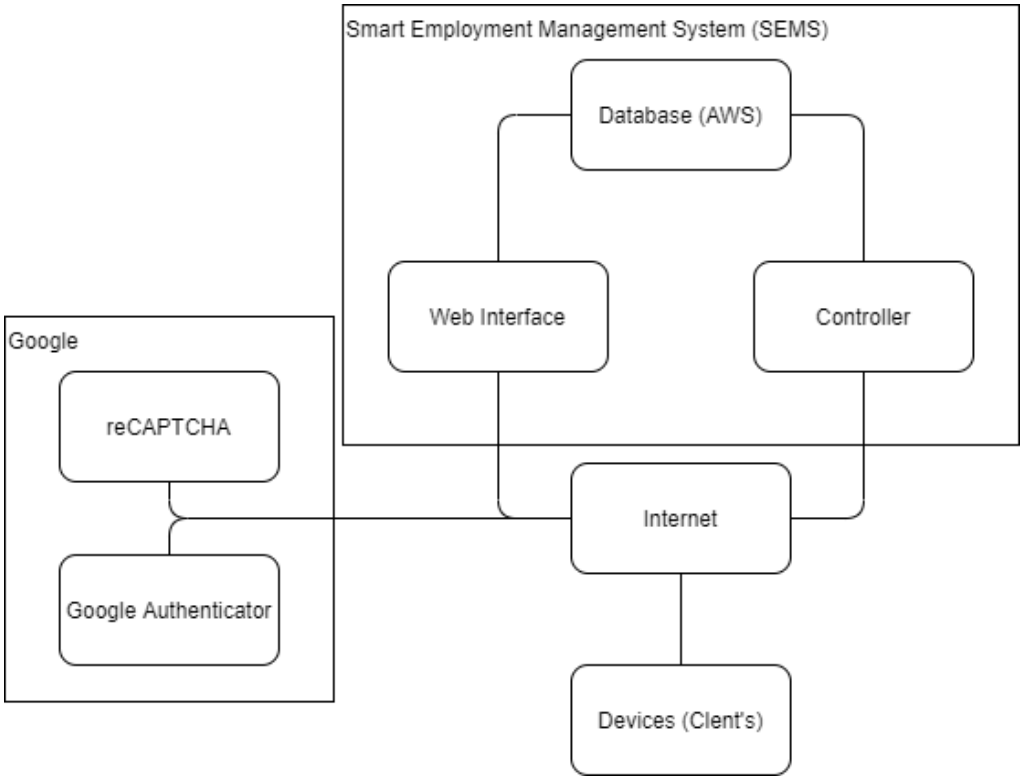

Figure – 1: Diagrammatic representation of sub-system interactions of SEMS.

Clients will use the necessary devices such as computers or a mobile device to connect to the internet to access SEMS. By establishing a connection with the internet, the clients will be able to authenticate themselves using reCAPTCHA and Google Authenticator. Afterwards, the clients are faced with the web interface allowing them to view the data from the database and controller to edit the data from Amazon Web Services (AWS). The connection established with the database represents the client-server architecture and three components (web interface (view), controller, database (model)) represents the model view controller.

System Information

Use Cases

I. Employer/HR Department - Level 2 User

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

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

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

D. Employers/HR have privileges to update employee information.

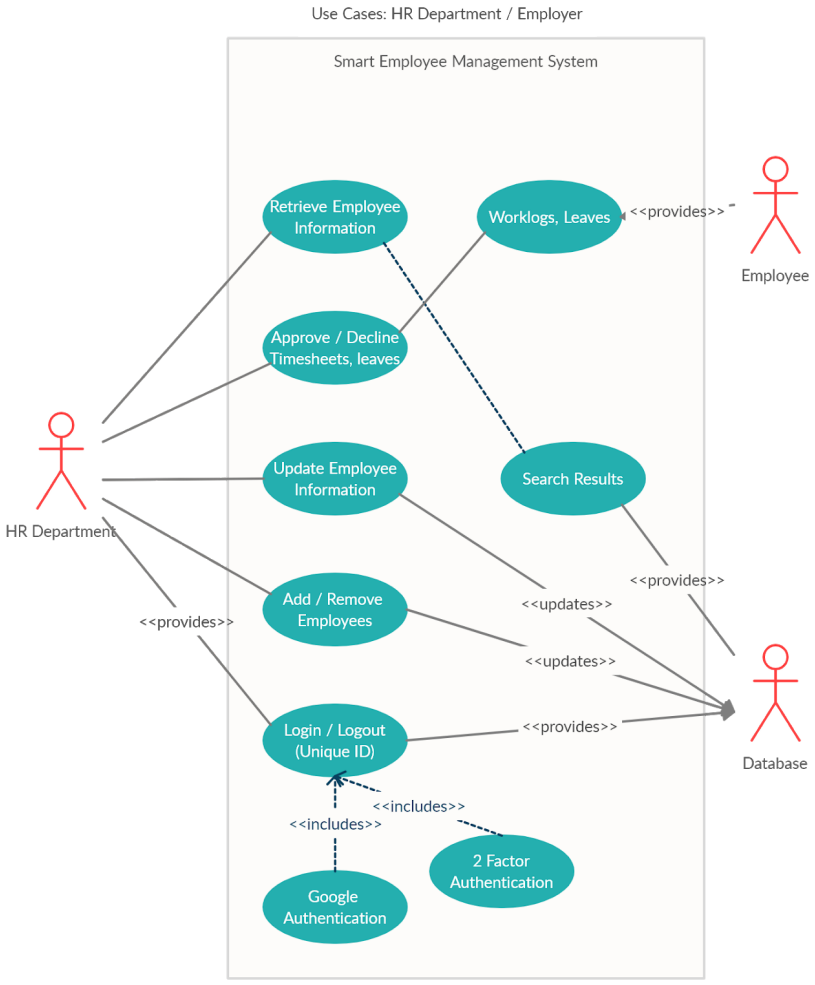

Figure – 2: Employer / HR Use Case Diagram

II. Employee - Level 1 User

A. Employees can log in to the system using unique credentials.

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

C. Employees can view, update their work log and payroll information.

D. Employees can update their personal information and login credentials.

E. Employees have the ability to raise invoices to Employer/HR.

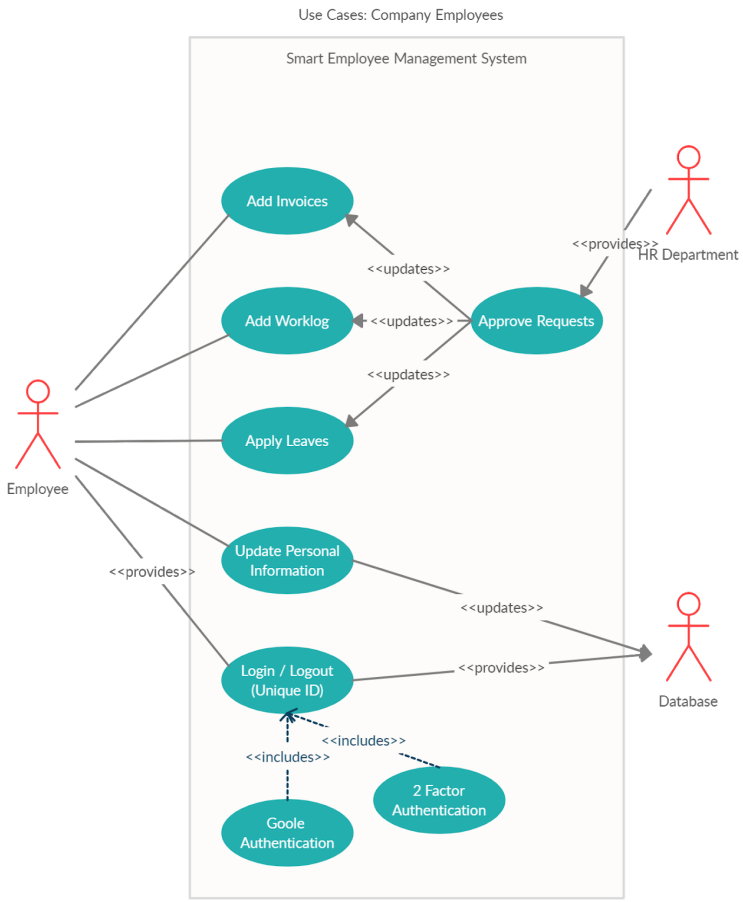

Figure – 3: Employee Use Case Diagram

III. Database System - Level 3

A. The database has access to all the user and system information.

B. The database can add/remove/update employee information.

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

D. The database can automatically calculate and generate employee payrolls.

E. The database can add/remove/update leaves, timesheets from the employees.

F. The database can update user privileges.

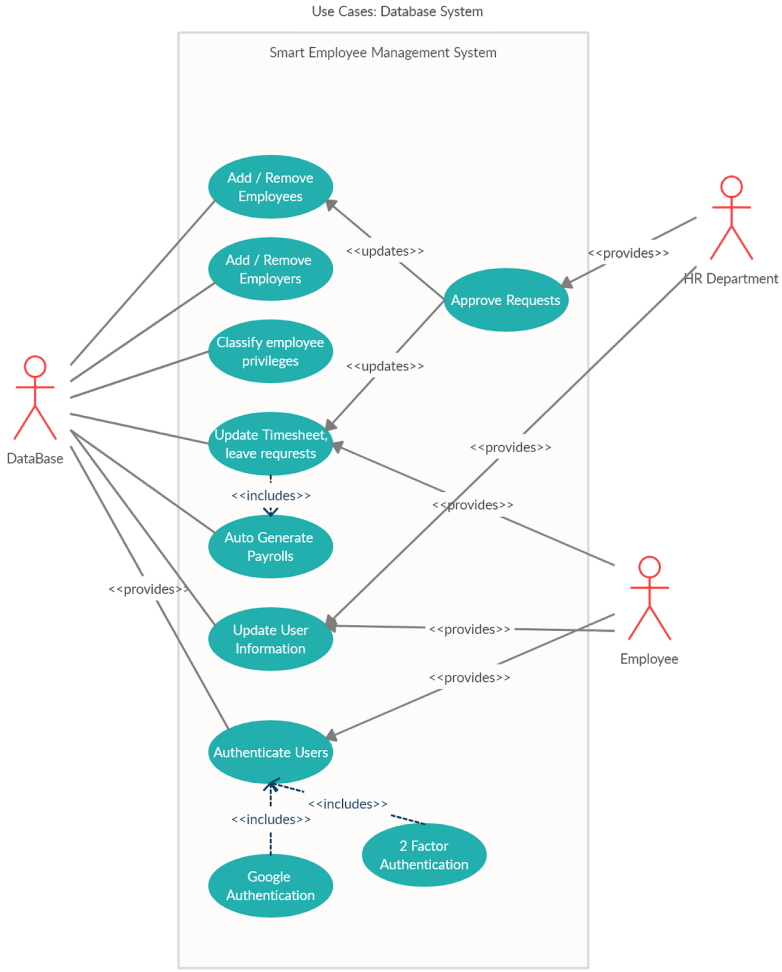

Figure – 4: Database System Use Case Diagram

IV. Developer & Designer - Level 3 User

A. Developers can view user information and system status.

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

C. Developers have full access to the database system.

D. Developer/Designer can log in to the system as a test user.

E. Developers can update user privileges.

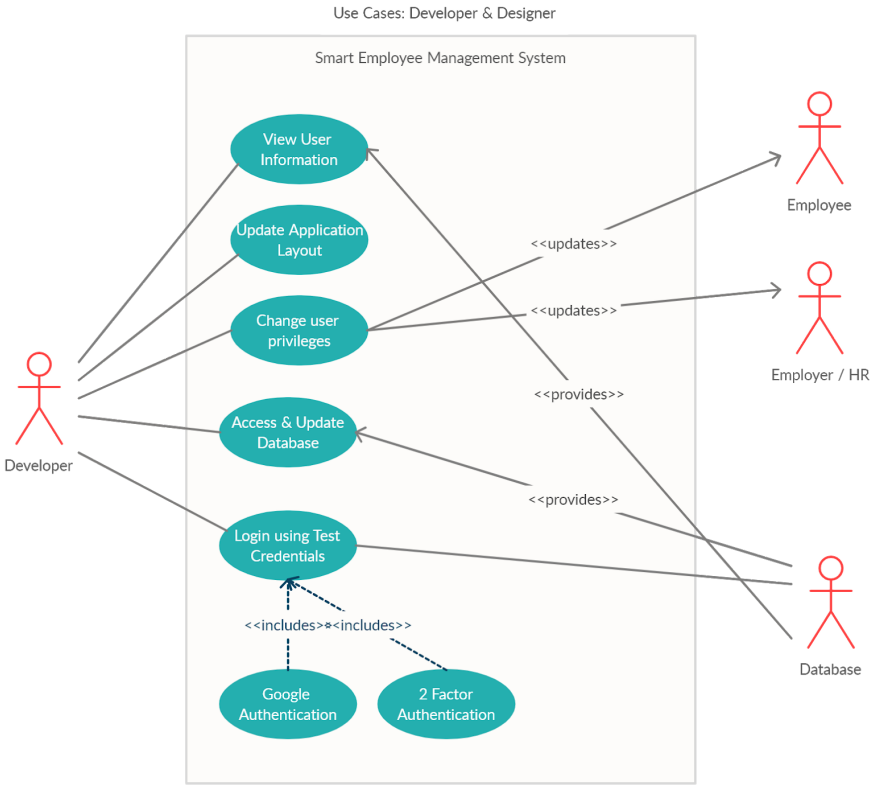

Figure – 5: Developer & Designer Use Case Diagram

Activity Diagrams

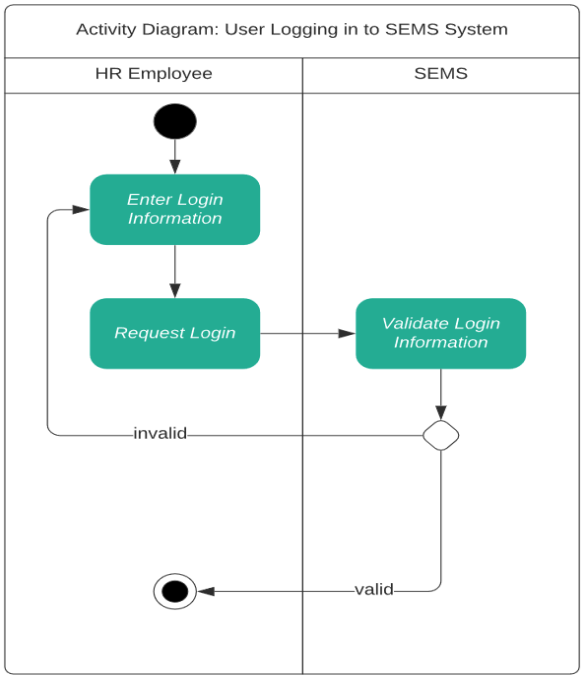

Figure – 6: Activity diagram of Level 2 user (HR employee) login into SEMS.

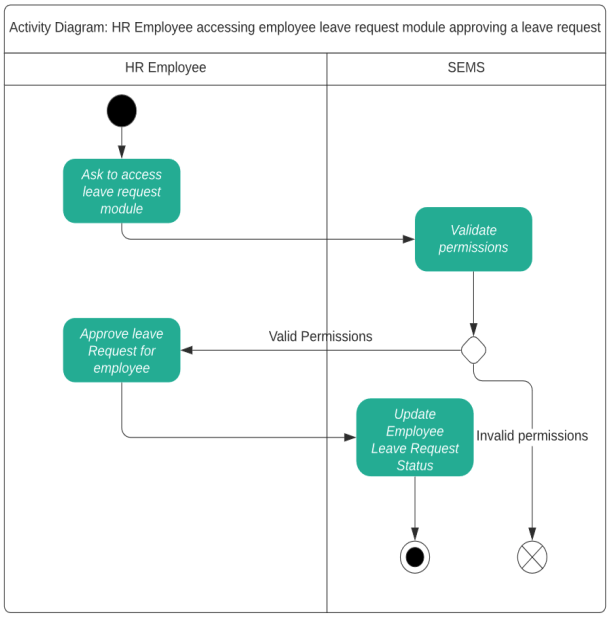

Figure – 7: Activity diagram of Level 2 user (HR employee) approving leave request of an employee.

Sequence Diagrams:

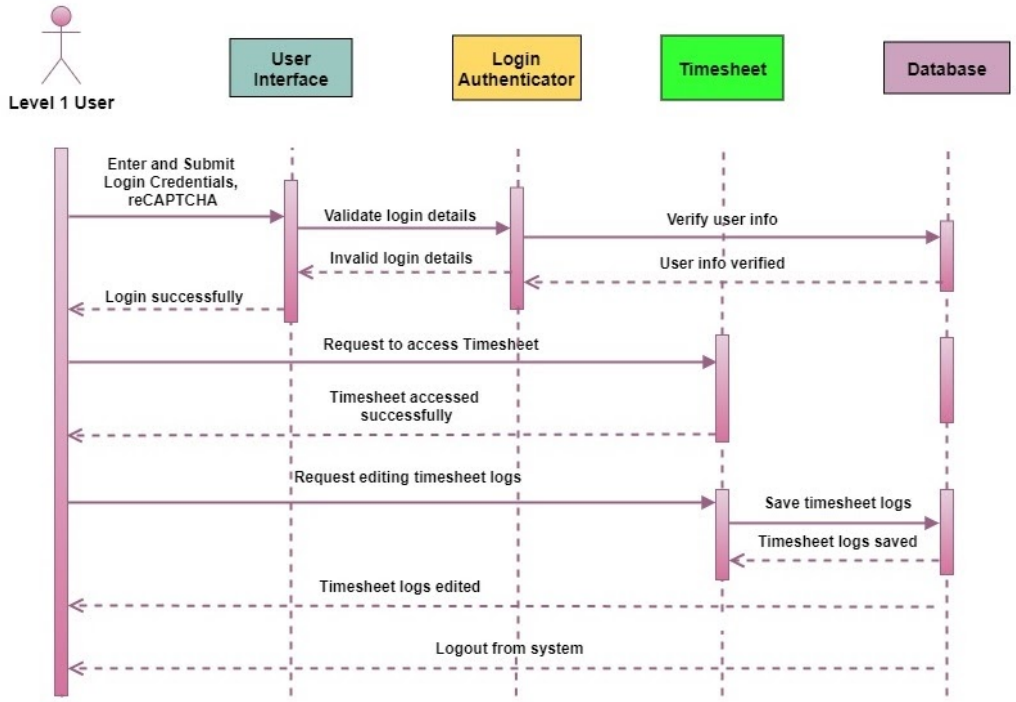

Figure – 8: Sequence Diagram of Level 1 user (Employee) editing timesheet logs.

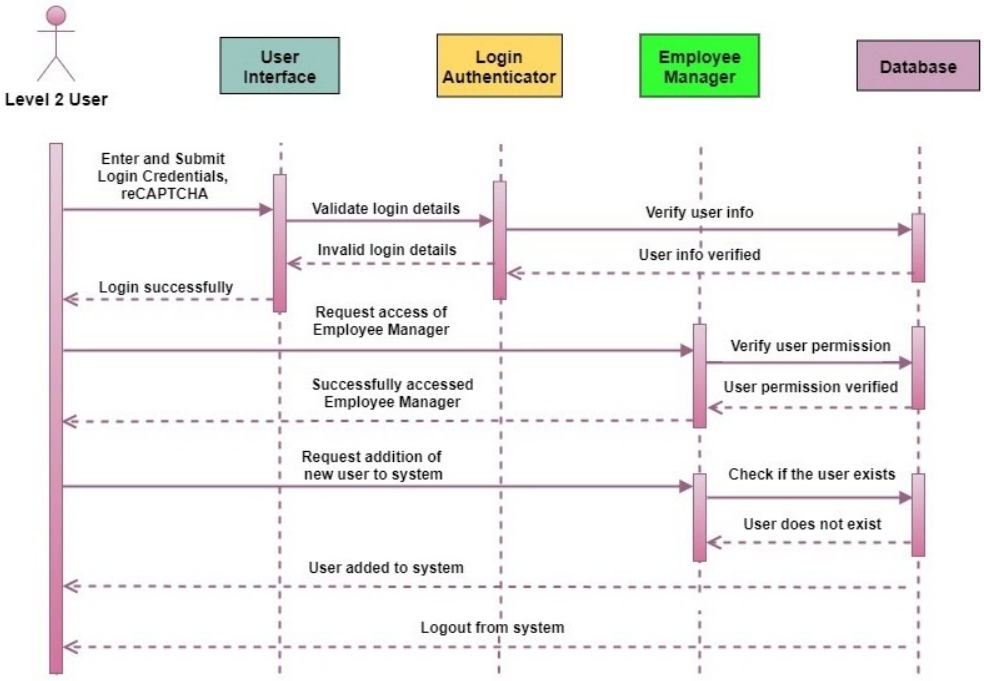

Figure – 9: Sequence Diagram of Level 2 user adding a new employee in the system.

Component Diagram

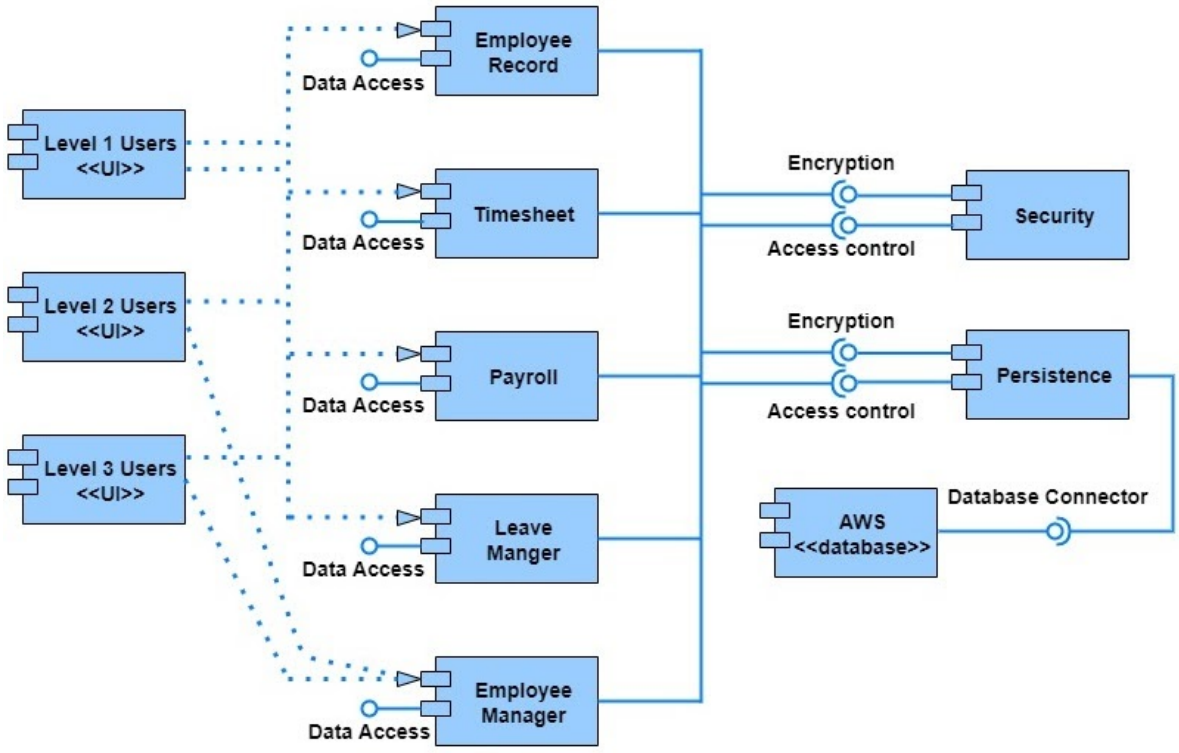

Figure – 10: Component Diagram of Smart Employee Management System (SEMS).

Concurrency

Concurrency in our software system is vital to its function as the software system needs to provide concurrency of users and concurrency among the different subsystems of the system. However, before we discuss why concurrency is needed we must define concurrency which is the ability for multiple things to be done at the same time or rather simultaneously. This naturally leads to its inclusion in our software system which is intended to serve multiple users at the same time. Furthermore, each of the users on the system should be allowed to access the different components of the system at the same time as others. As such each of these components should be independent of each other and function in a concurrent manner. An example would be that two or more employees should be able to access the employee management system at the same time and utilize the different components of the system such as updating timesheets and viewing previous pay stubs at the same time.

Division Of Responsibilities

The division of responsibilities among the developers of the software system is directly impacted by the software architectures and software development process chosen. As such our chosen architecture of utilizing an MVC architecture within a client-server architecture will determine how responsibilities are divided among developers. Initially, the client-server architecture splits the client-side and the server-side. Further, the MVC resides on the server side where the front-end developers work on the view and controller while the backend developers work on the model. Moreover, the model interacts with the database which further divides the responsibilities of those who manage the database. Finally, the software process chosen will further manage responsibilities within those individual categories to best provide a solution to the requirements provided.

Software Process

The preferred software development process for this project would be an iterative methodology like scrum. Although plan-driven methodologies like the waterfall method can be useful it is usually preferred when the user requirements are constant. However, we believe that since our project is intended to be used by many users and must constantly evolve to best cater to their growing requirements a more incremental methodology would be optimal. Not only that but most plan-driven projects tend to be delivered late and tend to be over budget versus an iterative methodology that seems to be historically more successful for projects of this size. A Scrum methodology would in this case be more practical as we can incrementally add features to the software system and quickly adapt to the ever-changing needs of the clients of the system. In a scrum methodology, incremental builds of the system are made every 2-3 weeks and as requirements of the system change, we can adjust the system to best provide the needs of the users. With this methodology, we can provide a high-quality product that is created to provide as much value to the end-users as possible.