SWE30010 - Managing IT Projects

TASK 14: Software Design

HUMAN RESOURCE MANAGEMENT WITH
ATTENDANCE SYSTEM

Group 2

GROUP 2 INFORMATION

Name	ID	Tutor	Class
Le Hoang Hai	103542974	Thomas Hang Nsam@swin.edu.au	Saturday 7:00 AM
Nguyen Dinh Nhat Minh	103802490		
Nguyen Nhat Huy	103802911		
Nguyen Ngoc Minh Thy	103802791		

A. Sprint 1 Software Design

With the completion of sprint one, we are able to create an activity UML diagram that captures our website's functionality as well as its design.

Since we have two different kinds of users, employees and managers, their activities differ dramatically. While employees can only view their own attendance, and personal information, managers can do that while also add more employees as well as updating their pay (via the payroll manager) as well.

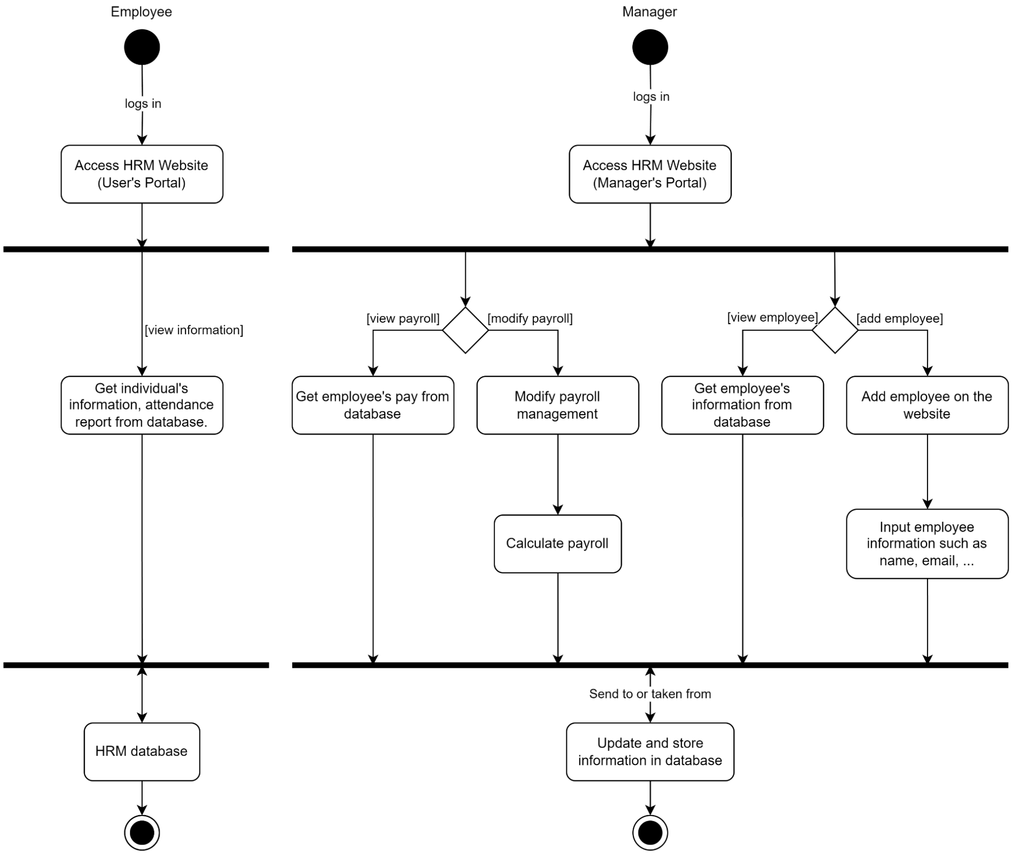
Employee
Manager
logs in
logs in
Access HRM Website (User's Portal)
Access HRM Website (Manager's Portal)
[view information]
[view payroll]
[modify payroll]
[view employee]
[add employee]
Get individual's information, attendance report from database.
Get employee's pay from database
Modify payroll management
Get employee's information from database
Add employee on the website
Calculate payroll
Input employee information such as name, email, ...
Send to or taken from
HRM database
Update and store information in database

Figure A: Software design after Sprint 1

B. Justification

1. MVC

We followed the Model-View-Controller (MVC) design pattern when creating the post-sprint 1 activity UML design. In this design, the controller (website's backend) coordinates interactions between the model and view, the view (website's frontend) manages the display layer, and the model (website's backend and database) represents the data and business logic. By adhering to the MVC pattern, we can ensure that modifications made to one component don't unduly impact the others, encouraging maintainability and simplifying future development.

2. Security

We have implemented various API security measures to protect user data and prevent unauthorised access. Our methods include:

Salting: Each password is salted with a random value before hashing.

Cryptographic Hashing: Salted passwords are hashed using the bcrypt algorithm, making them resistant to brute-force attacks.

Limited Database Access: Access to data in the database is restricted to authorised personnel only (developer and overseer), reducing the risk of data breaches.

Role-Based Access Control: Our system incorporates a role-based access control mechanism with two roles: "admin" and "user." This ensures that different users have appropriate levels of access privileges and view features appropriate to their role.

JSON Web Tokens (JWT): JWTs are utilised for token generation and management. Upon successful authentication, a unique token is generated and issued to the user. These tokens expire after a set period, mitigating the risk associated with stolen or intercepted tokens.

Middleware Validation: The backend API employs middleware to verify the authenticity of incoming requests. Only authenticated users with valid tokens are permitted to make requests, enhancing overall system security.

In this first sprint, our concerns are only about the website's basic functionalities, hence, advanced security has yet to be developed; still, this critical layer of security safeguards our platform against unauthorised access and potential security breaches, reinforcing user trust and confidence in our commitment to protecting their data and privacy.