JAVANGERS ASSEMBLE EMPLOYEE MANAGEMENT SYSTEM

BY JACOB CANDELARIA

INTRODUCTION

- Key Features of the Project:
  - Employee Time Management – Punch In/Out timestamps
  - Employee Management
    - Standard Users
      - ✓ Update Select Personal Information Fields
    - Admin Users
      - ✓ Create New Employees
      - ✓ Update All Personal Information Fields for *all* Employees
      - ✓ View All Employee Time Entries
      - ✓ Modify Employee Time Entries

PROJECT DEMO

EMPLOYEE LOGIN SCREEN

NEW SKILLS/TECHNOLOGIES

- Dagger – further increased understanding of dependency injection via Dagger.
- Swagger – learned how to design APIs, create documentation, and import into AWS API Gateway for a faster setup.
- AWS DynamoDB – further increased understanding of DynamoDB and mapping to Java objects.
- AWS Lambda – learned how to create custom Lambda requests and responses.
- AWS API Gateway – learned how to create API endpoints, setup CORS, and deploy to various stages to connect different parts of the project.
- BouncyCastle Argon2 – learned how to implement the Argon2 algorithm for generating and verifying password hashes.
    - Argon2 was the winner of the Password Hashing Competition (PHC) in 2015.
    - Standards: Included in many modern security standards and recommendations for password hashing, including those by OWASP (Open Web Application Security Project).

CHALLENGES

- Going into the Project
  - Limited Knowledge/Experience implementing AWS Lambda Functions and Dagger from scratch.
- Unexpected Challenges
  - Initially, I was going to use AWS Cognito to manage the user credentials, validation, and secure the API endpoints. Due to limitations on the AWS account, I was unable to implement this and had to fallback to the Argon2 algorithm for manual verification.

OUTCOMES

- I got to experience what it was like to be part of the development of a product from start to finish.
- I learned how to design and develop my own APIs, which are crucial for connecting the backend with the frontend for web-based programs.
- I further increased my understanding of secure password storage and manual credential validation, which is important in case of data breaches.
- Future consideration:
    - For API Development, it's better to have error codes in both the documentation and integrate them in the backend for easier handling on the frontend.