Building an Enterprise-Grade, Secure HR Platform

A Technical Deep Dive into the CMPE 272 HR Portal Project

Team: Code Coven

Repository: https://github.com/NMemane1/CMPE272-HR-PORTAL-ESP-TeamCodeCoven

Date: December 2025

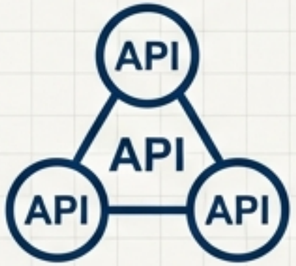
Key Achievements: Delivering a Production-Ready System

Enterprise-Grade Security:
Successfully implemented OAuth2/OIDC single sign-on (SSO) with comprehensive Role-Based Access Control (RBAC) enforcement.

Cloud-Native Deployment:
Deployed on a secure AWS infrastructure with a fully automated CI/CD pipeline using GitHub Actions.

Proven Reliability:
Achieved 100% API test coverage across 73 automated tests, validating functionality, security, and error handling.

Secure Data Management:
Ensured data integrity with encrypted database storage (at rest and in transit) and secure credential management via AWS Secrets Manager.



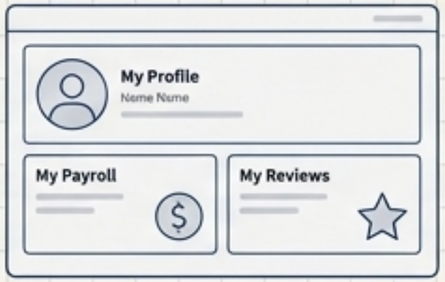
Robust API Layer:
Developed a comprehensive REST API in Spring Boot, fully documented with Swagger.

Defining the Mission: A Secure, Scalable, and User-Centric HR Portal

Project Objectives

- Manage employee profiles and organizational hierarchy.
- Streamline payroll processing and performance evaluations.
- Provide secure, role-based access to sensitive HR data.
- Create a scalable and maintainable enterprise platform.

Role-Based Feature Set



Employees

View/update profile, access payroll, view performance reviews.

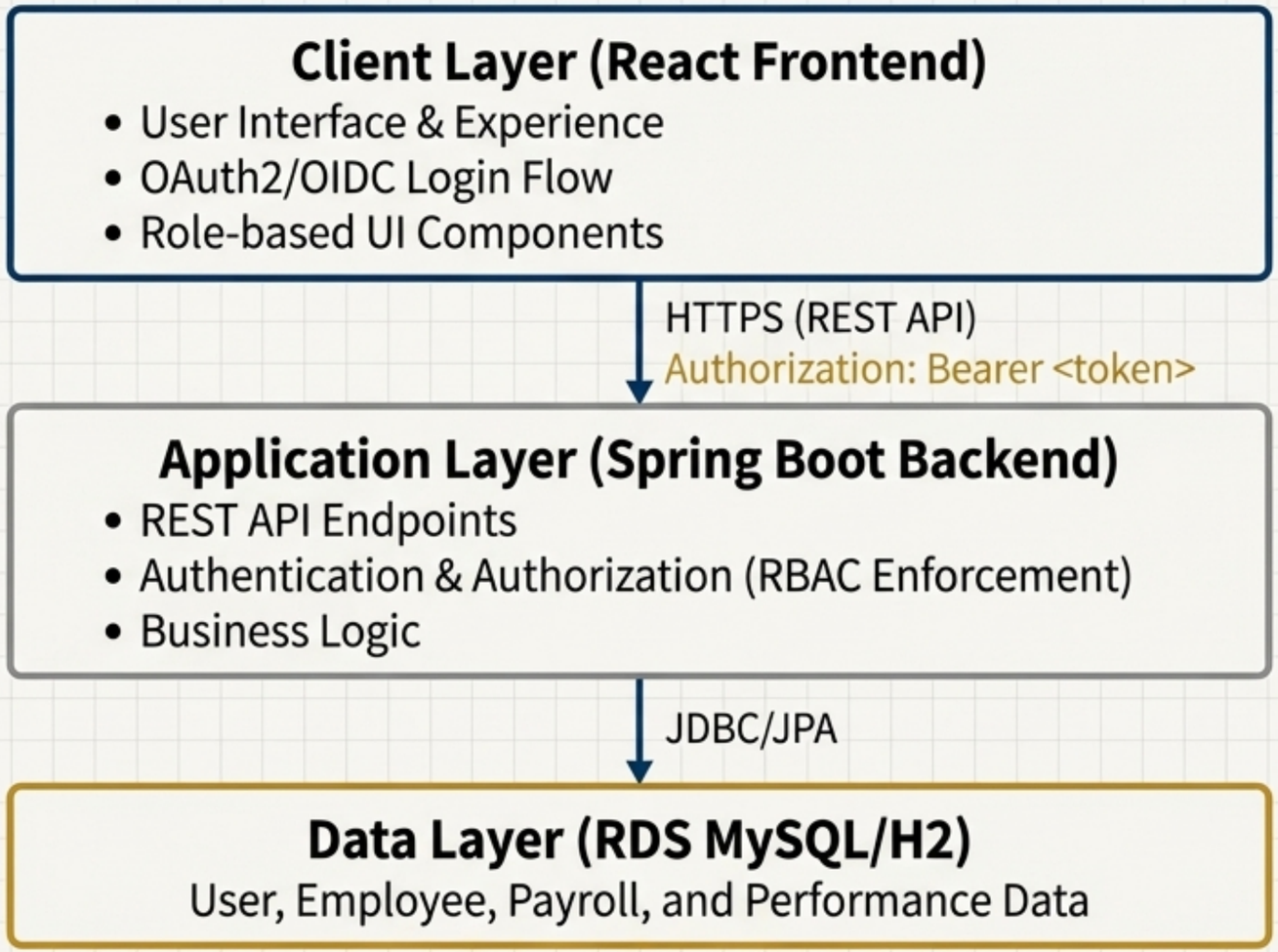
Managers

All Employee permissions, plus manage team profiles, access team data, conduct reviews.

HR Administrators

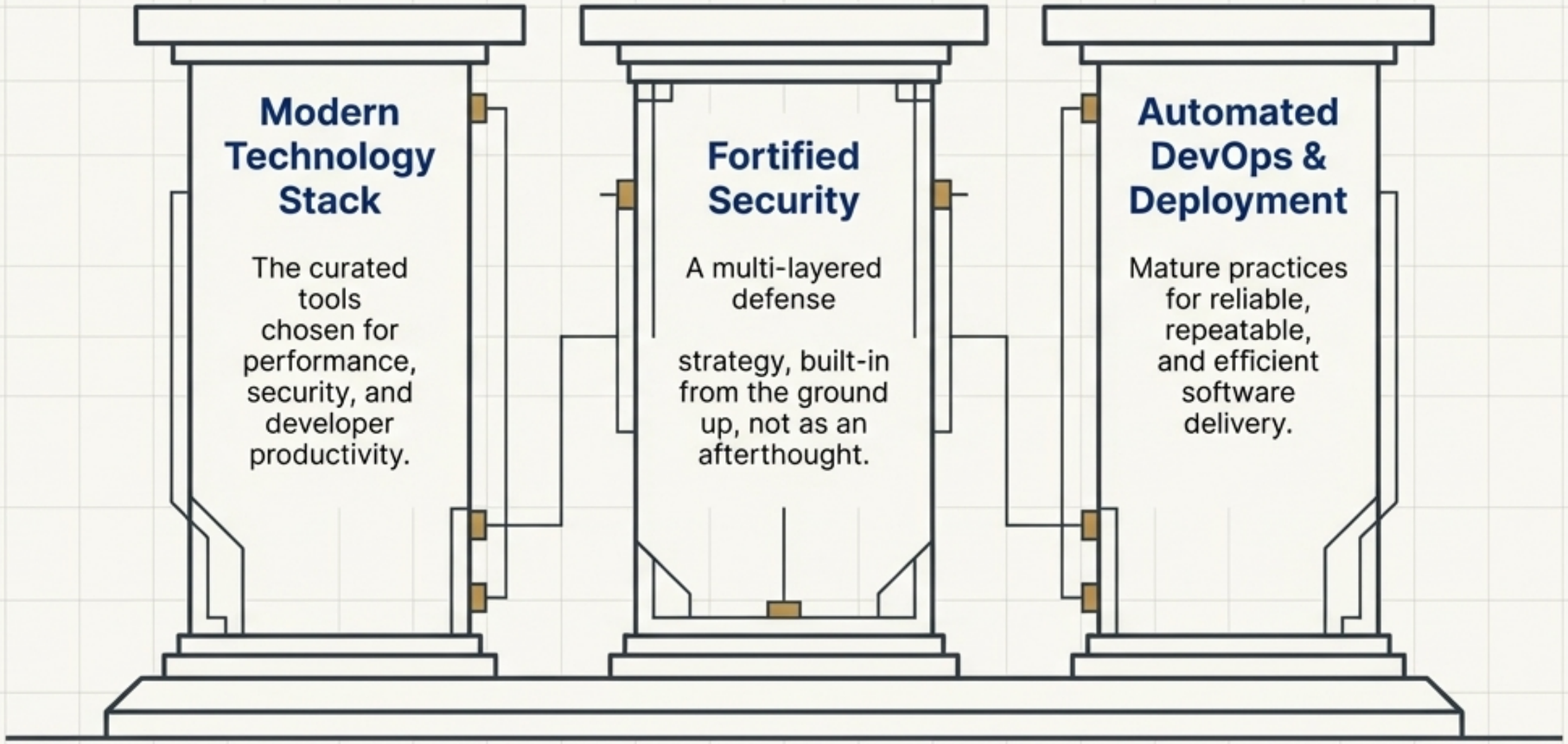
Full system access, create/manage all records, process organization-wide payroll, manage user roles.

The System Blueprint: A Resilient Three-Tier Architecture



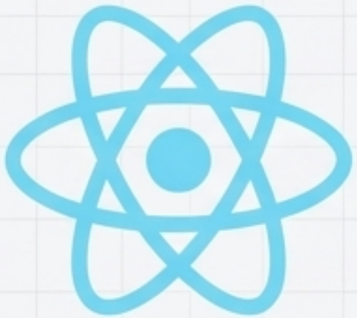
This decoupled architecture ensures scalability, maintainability, and clear separation of concerns, forming the foundation for a robust enterprise application.

Architectural Blueprint
The Three Pillars of an Enterprise-Grade Application



Our approach was built on three foundational pillars, ensuring the final product is not only functional but also secure, scalable, and maintainable.

Pillar 1: A Modern, Purpose-Driven Technology Stack



Frontend

- **React 18.x**: UI Framework
- **Vite 4.x**: Build Tool & Dev Server

- **React Router 6.x**: Client-side Routing
- **Axios 1.x**: HTTP Client
- **Tailwind CSS 3.x**: Styling Framework

Backend

- **Spring Boot 3.x**: Application Framework
- **Spring Security 6.x**: Authentication & Authentication & Authorization
- **Spring Data JPA 3.x**: Data Access Layer
- **MySQL 8.x / H2 2.x**: Production / Development Database

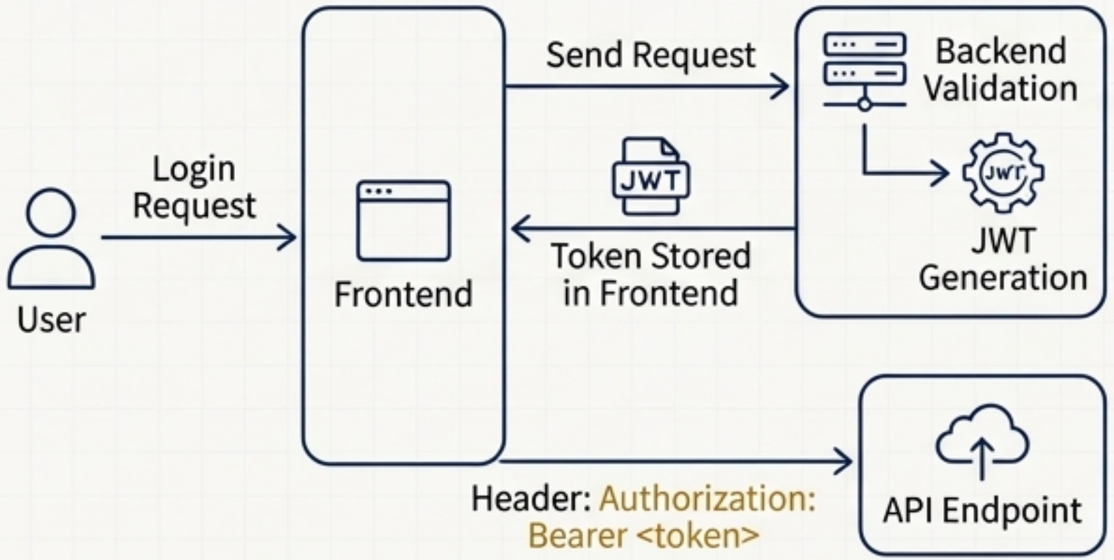
DevOps & Infrastructure

- **Docker / Docker Compose**
- **Docker / Docker Compose**: Containerization
- **AWS (EC2, ECS, RDS, VPC)**: Cloud Infrastructure
- **GitHub Actions**: CI/CD Pipeline
- **Terraform**: Infrastructure as Code

Pillar 2: Fortifying the Gates with Authentication & RBAC

Industry-Standard Authentication with OAuth2/OIDC

- Utilizes OAuth2 framework and OIDC for identity.
- Employs stateless JWTs (JSON Web Tokens) for authentication.
- Uses BCrypt for secure password hashing.



Granular Authorization via a Three-Tier Role Hierarchy

Role	Permissions	Access Level
EMPLOYEE	View own profile, payroll, and reviews	Self-access only
MANAGER	All EMPLOYEE permissions + team-level data access	Team-level access
HR_ADMIN	All MANAGER permissions + full system configuration	Organization-wide access

Pillar 2 (cont.): A Defense-in-Depth Security Posture

Protecting Data at Every Stage

Data at Rest: AWS RDS encryption.

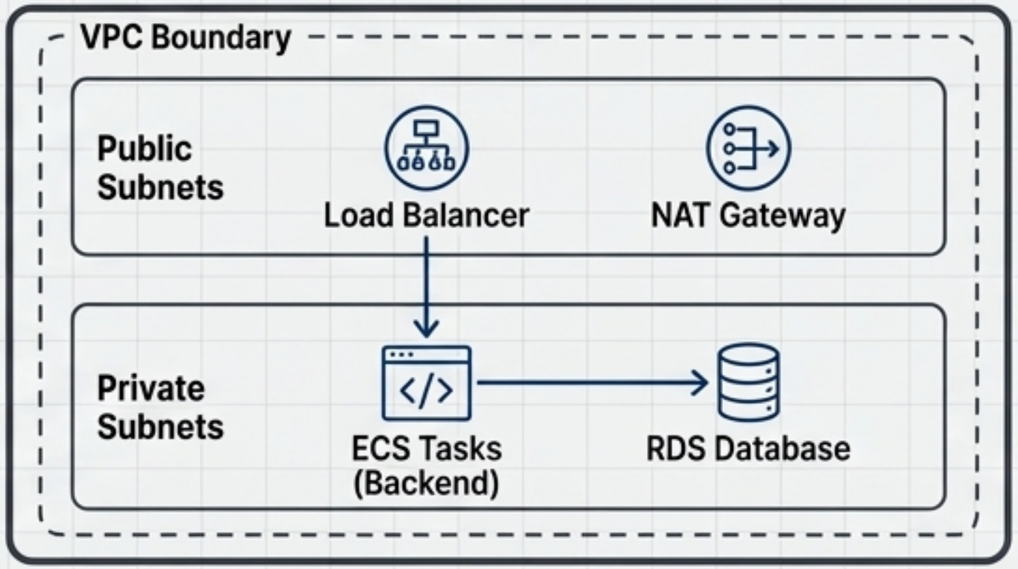
Data in Transit: TLS/SSL encryption for all communication.

Credentials: Secure credential management with AWS Secrets Manager.

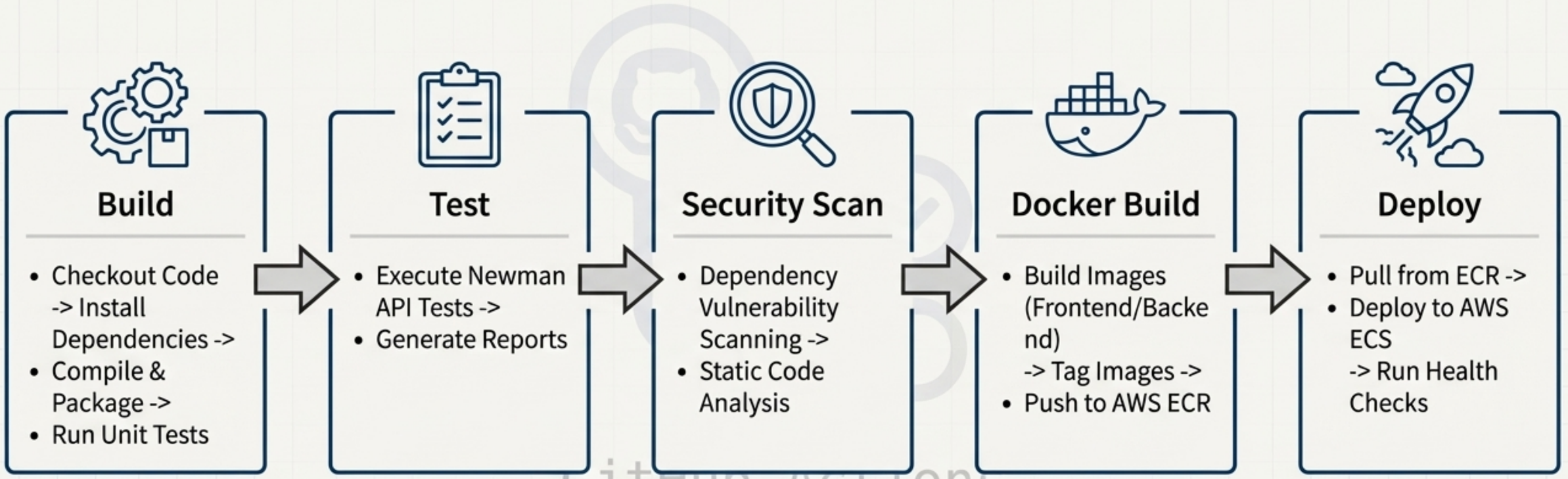
API Hardening: SQL injection prevention, XSS protection, and rate limiting.

Isolating Infrastructure with AWS VPC

- **VPC:** Provides complete network isolation.
- **Security Groups:** Act as firewalls, enforcing the principle of least privilege.
- **Private Subnets:** Ensure the application and database are not directly exposed to the internet.



Pillar 3: Automated Delivery with a Mature CI/CD Pipeline



Key Takeaway: This fully automated pipeline ensures that every code change is rigorously tested, scanned, and deployed reliably, minimizing manual errors and accelerating delivery.

Rigorous Validation: Achieving 100% Pass Rate Across 73 Automated Tests

Test Category	Total Tests	Passed	Pass Rate
Authentication	15	15	**100%**
Authorization (RBAC)	18	18	**100%**
Employee CRUD	12	12	**100%**
Payroll Operations	10	10	**100%**
Performance Reviews	8	8	**100%**
Error Handling	10	10	**100%**
Total	**73**	**73**	**100%**

Key Findings

- ✅ All authentication and RBAC rules function as designed.
- ✅ Unauthorized API calls correctly return 401 Unauthorized or 403 Forbidden.
- ✅ Excellent average API response time of 113ms.
- ✅ Graceful handling of all tested error scenarios.

Evidence from the Field: Validating Critical Security and Business Logic

Test Case 1: Authentication Success

Scenario: `POST /api/auth/login` with valid employee credentials.

Result:
Returns `200 OK` with user data, role (`EMPLOYEE`), and JWT.

200 OK

```
{
  "user": "...",
  "role": "ENPLOYEE",
  "token": "jwt..."
}
```

Test Case 2: RBAC Enforcement

Scenario: An authenticated **Employee** attempts to access an HR_ADMIN-only endpoint like `DELETE /api/employees/3`.

Result:
API correctly returns `403 Forbidden`.

403 Forbidden

Test Case 3: Core Business Logic

Scenario: An authorized user sends a `PUT /api/payroll/1` request to update a salary.

Result:
Returns `200 OK` with the updated payroll record, confirming the new salary value.

```
PUT /api/payroll/1
{
  "salary": 130000
}
```

→

200 OK

```
{
  "id": 1,
  "salary": 130000,
  "..."
}
```

Ingenuity in Action: Resolving a Critical Spring Security Conflict

The Challenge

Problem

The `/api/auth/login` endpoint was incorrectly returning `403 Forbidden`, blocking all authentication attempts.

Root Cause

Spring Security's default filter chain was intercepting the request before it could reach the custom JWT authentication logic.

The Solution

Action

Reconfigured the `SecurityFilterChain` to explicitly permit unauthenticated access to the auth endpoint while securing all others.

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http
        .csrf().disable()
        .authorizeHttpRequests(auth -> auth
            .requestMatchers("/api/auth/**").permitAll() // <-- The critical fix
            .anyRequest().authenticated()
        )
        .sessionManagement()
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS);
    return http.build();
}
```

Outcome

This precise configuration resolved the conflict, enabling the JWT authentication to function correctly and securing all other API endpoints as intended.

The Strategic Roadmap: Evolving the HR Platform

Phase 1 (Short-term): Foundational Enhancements

UI/UX

- Dashboard with analytics, real-time notifications, dark mode.

Features

- ✓ Time-off request workflow, document management, audit logging.

Phase 2 (Medium-term): Integration & Advanced Security

Security

- ✓ Multi-factor authentication (MFA), session management with refresh tokens.

Integrations

- ✓ Calendar (Google/Outlook), Slack/Teams notifications, SSO with enterprise identity providers.

Phase 3 (Long-term): Intelligence & Scalability

AI/ML Features

- ✓ Predictive analytics for attrition, salary recommendation engine.

Architecture

- ✓ Caching layer (Redis), database read replicas, transition to a microservices architecture.

A Showcase of Enterprise Development Mastery

Technical Skills Applied

- **Full-Stack Development:** Spring Boot backend and React frontend.
- **Cloud Engineering:** AWS infrastructure design, deployment (VPC, ECS, RDS), and Infrastructure as Code (Terraform).
- **DevOps & Automation**: CI/CD pipeline implementation with GitHub Actions and automated API testing with Postman/Newman.
- **Enterprise Security**: Deep implementation of Spring Security, OAuth2/OIDC, RBAC, and network security principles.

Collaboration & Project Management

- **Agile Methodology**: Sprint planning and retrospectives.
- **Version Control**: Git branching strategies with pull requests and code reviews.
- **Process**: Clear role definition, task distribution, and continuous documentation.

The Team: Code Coven

Team Member	Primary Role	Key Responsibilities
Katherine	Frontend Development	React UI, OAuth2/OIDC integration, Protected routes, Frontend Dockerfile
Yuling	Database & AWS DevOps	DB schema, RDS setup, VPC/ECS configuration, CI/CD pipeline, CloudWatch
Nikita	Backend & Security	REST API development (Spring Boot), OAuth2/OIDC SSO, RBAC middleware
Shilpa	Testing & Documentation	Postman test automation, UI/Backend testing, Architecture diagrams