**Group B: Employee Performance Management System - Deliverable 4**

Project contact: danieln3@umbc.edu or tartelt1@umbc.edu

Tartela Tabassum - Project Manager

Daniel Nguyen - Full-Stack Developer

Amira Patel - UI/UX Designer

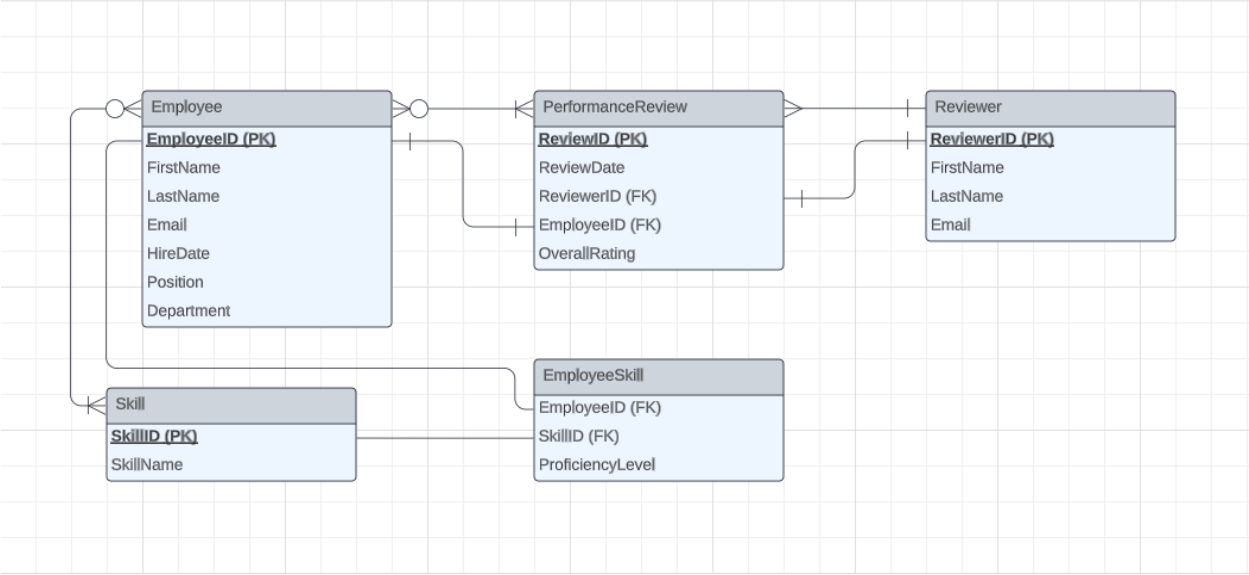
Sunny Ravulapalli - Quality Assurance (QA) Engineer

GitHub: https://github.com/Programmermandan/IS436-GroupB.git

Project plan board: https://github.com/users/Programmermandan/projects/4

(Time set aside to meet outside of class is included in the GitHub board)

## Data Model - ERD




**Entities and their attributes:**

Employee:

- Attributes: EmployeeID (Primary Key), FirstName, LastName, Email, HireDate, Position, Department

PerformanceReview:

- Attributes: ReviewID (Primary Key), ReviewDate, ReviewerID (Foreign Key to Reviewer), EmployeeID (Foreign Key to Employee), OverallRating

Reviewer:

- Attributes: ReviewerID (Primary Key), FirstName, LastName, Email

Skill:

- Attributes: SkillID (Primary Key), SkillName

EmployeeSkill:

- Attributes: EmployeeID (Foreign Key to Employee), SkillID (Foreign Key to Skill), ProficiencyLevel

**Relationships:**

- An Employee has zero or more performance reviews.
- Relationship Type: One-to-Many.
- Foreign Key: EmployeeID in PerformanceReview references Employee.
- A PerformanceReview has one Reviewer, and a Reviewer can conduct multiple PerformanceReviews.
- Relationship Type: Many-to-One.
- Foreign Key: ReviewerID in PerformanceReview references Reviewer.
- An Employee can have multiple Skills, and each Skill can be associated with multiple Employees.
- Relationship Type: Many-to-Many.
- Junction/Association Table: EmployeeSkill.
- Foreign Keys: EmployeeID in EmployeeSkill references Employee, SkillID in EmployeeSkill references Skill.

**Alternative Matrix**

| Criteria | Weight (W) | Custom Java App |
|---|---|---|
| Technical Feasibility | 0.3 | 9 |
| Economic Feasibility | 0.4 | 8 |
| Organizational Feasibility | 0.3 | 7 |

| | | |
|---|---|---|
| Total | 1.0 | 8.0 |

Justification of chosen alternative matrix:

- Technological Feasibility: Since, the custom Java application is written in Java, a widely used and adaptable programming language suited for a wide range of enterprise applications, it earns a high technological feasibility score (9).
- Economic Feasibility: A score of 8 indicates that the development and maintenance expenditures of the custom Java application are fair and provide good value.
- Organizational Feasibility: The custom Java application receives a score of 7, indicating that it is reasonably well aligned with the organization's existing processes and culture.
- Weights: To achieve correct normalization, the weights (W) for each criterion stay the same as in the preceding example, with a total weight of 1.0.
- Overall: The overall score for the custom Java application is calculated by multiplying each criterion score by its corresponding weight and then summing up the results. The custom Java program received an overall score of 8.0.

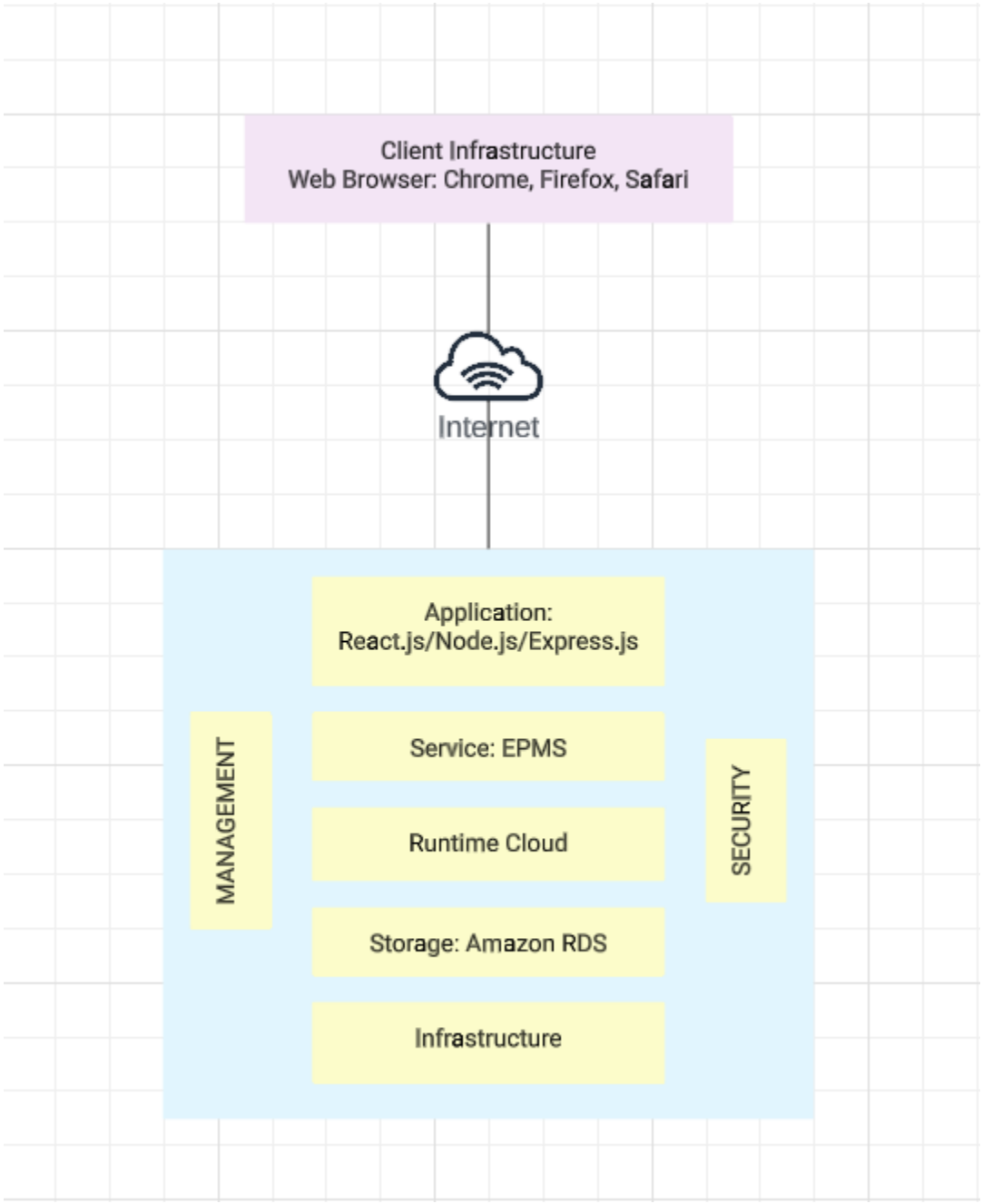
The correct calculation for the total score is as follows:

Total Score = (Technical Feasibility Score * Technical Feasibility Weight) + (Economic Feasibility Score * Economic Feasibility Weight) + (Organizational Feasibility Score * Organizational Feasibility Weight) Total Score = (9 * 0.3) + (8 * 0.4) + (7 * 0.3) Total Score = 2.7 + 3.2 + 2.1 Total Score = 8.0

Based on its technical, economic, and organizational feasibility, this matrix indicates that the custom Java program is a promising alternative for an Employee Performance Management

System. However, like with any selection, additional aspects such as user requirements, scalability, security, and support must be considered before making a final decision for the alternative matrix.

**Architecture Diagram**



**Matrix:**

| Non-Functional Requirement | Server-Based | Client-Based | Thin-Client Server | Thick Client Server | Cloud Computing |
|---|---|---|---|---|---|
| Scalability | ✔ | ✘ | ✔ | ✔ | ✔ |
| Accessibility | ✔ | ✘ | ✔ | ✔ | ✔ |
| Performance | ✔ | ✘ | ✔ | ✔ | ✔ |
| Security | ✔ | ✘ | ✔ | ✔ | ✔ |
| Maintenance | ✘ | ✔ | ✔ | ✔ | ✔ |
| Cost | ✘ | ✘ | ✔ | ✘ | ✔ |

**Narrative:**

- Scalability: Cloud computing is a good fit for scalability due to its ability to dynamically allocate resources based on demand. Thin-client server architecture also supports scalability by centralizing application logic and data.
- Accessibility: All architectures except client-based ensure easy accessibility, with cloud computing offering the advantage of remote access.

- Performance: Server-based, thin-client server, and cloud computing architectures can distribute processing load, enhancing performance.
- Security: All architectures can implement security measures, but cloud computing may raise concerns about data privacy.
- Maintenance: Client-based, thin-client server, and cloud computing architectures allow centralized updates, reducing maintenance efforts.
- Cost: Thin-client server and cloud computing architectures offer cost-effective solutions, while client-based may incur higher maintenance costs.

**System Architecture Justification:**

Considering the matrix and narrative, a Cloud Computing Architecture is chosen for the Employee Performance Management System. This decision is based on the strong support for scalability, accessibility, performance, security, and maintenance. While there are concerns about data privacy and potential costs, the benefits outweigh these drawbacks for the chosen system.

**Hardware and Software Specifications:**

Cloud Server Configuration:

- Cloud Provider: Amazon Web Services (AWS)
- Instance Type: m5.large (2 vCPUs, 8GB RAM)
- Storage: Amazon RDS (Relational Database Service)
- Load Balancer: Elastic Load Balancing (ELB)

Client Configuration:

- Web Browser: Chrome, Firefox, Safari

Software Stack:

- Frontend: React.js
- Backend: Node.js with Express.js
- Database: MySQL
- Authentication: JSON Web Tokens (JWT)
- Cloud Services: AWS Lambda, AWS S3 (for file storage), AWS CloudWatch (monitoring)