Architecture Design Document – Team 2

Team Members: Sahithi, Tobey, Eryx, Jorge

Date: 06/30/2025

1. Cloud Service Provider Selection and Justification

Chosen Provider: AWS (Amazon Web Services)

Justification:

- **Cost:** AWS offers a flexible pay-as-you-go model that reduces upfront infrastructure costs. With options like EC2 Spot Instances and the AWS Free Tier, it's budget-friendly for scalable application development.
- **Scalability:** AWS provides auto-scaling, elastic load balancing, and global availability zones, ensuring the application can scale seamlessly with user demand and traffic spikes.
- **Services Used:**
    - **EC2 (Elastic Compute Cloud):** To host our backend Node.js server in a scalable and secure environment.
    - **S3 (Simple Storage Service):** To store static assets like images, documents, and backups.
    - **RDS (Relational Database Service):** For hosting a MySQL database to manage structured application data.
    - **VPC (Virtual Private Cloud):** To configure network-level security with public/private subnets.
    - **CloudWatch:** For monitoring application health and logging.
    - **IAM (Identity and Access Management):** To control access to AWS resources.

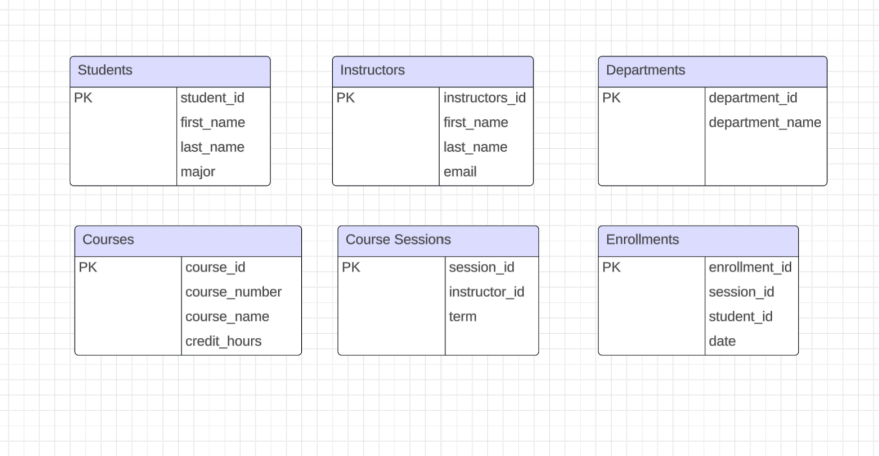
2. Application Design

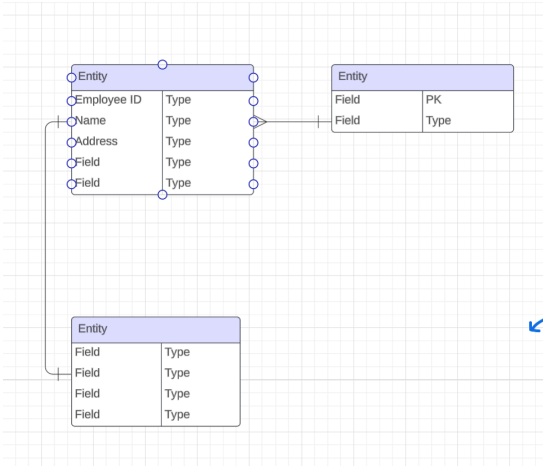
- **Programming Language:** JavaScript

- **Runtime Environment:** Node.js
- **Application API:** RESTful
- **Framework:** React
- **Middleware:** Express
- **Brief Description:**
  JavaScript enables both frontend and backend development using a unified language, streamlining development. Node.js provides a lightweight, event-driven runtime ideal for scalable applications. Express simplifies routing and server-side logic, while React offers a dynamic and responsive user interface. These technologies work together to support a modular and maintainable web application.

3. Database Schema and Design

- **Chosen Database Management System: MySQL.** We decided MySQL would be the ideal choice for this assignment due to its reliability and its ease of use. It offers scalability when managing structured data in the real world making it suitable for any small or large projects. MySQL is also supported across many platforms and programming languages which allows for easy integration with front end applications and web frameworks.
- **Schema Description**: The relationship between entities and relationships allows the database to organize and effectively manage information relevant to the operation. It provides a detailed representation of the primary entities, attributes, and their relationship that exist between them. This allows the decision maker to make smart decisions for the future of the company without putting stakeholders at risk.
- **ER Diagram:** Below I attached an example of an ER diagram that I will be using throughout this project. I will be using LucidChart to draw my diagram and properly put it into the third normalization form. My primary key entities that I will be using are *Students, Instructors, Courses, Course Sessions, Enrollments*, and *Departments.*





4. Network Architecture and Security

Key Components:

- **VPC (Virtual Private Cloud)**: CIDR block 10.0.0.0/16
- **Public Subnets**: For IIS/Node.js web and app servers
- **Private Subnets**: For MySQL database, not internet-accessible
- **NAT Gateway**: Allows private servers to get updates without public access
- **Internet Gateway**: Enables public-facing servers to communicate online

Redundancy:

- Two public and two private subnets across Availability Zones to support 240 hosts per subnet

Security Controls:

- **Web Server (EC2)**: Allow ports 80, 443, ICMP in; all traffic out
- **RDS (MySQL)**: Allow port 3306 from EC2 only; block other traffic
- **Network ACLs**: Applied at subnet-level for defense-in-depth
- **DMZ Zone**: Public-facing; **Internal Zone**: Secure services like DB

Ports Used:

- 80 (HTTP), 443 (HTTPS), 3306 (MySQL), ICMP (ping)

5. Data Visualization Tool Selection and Justification

- **Chosen Tool:** Power BI

- **Justification:** I've used Power BI in the past, so I was already comfortable working with it. It lets me quickly create clean, professional dashboards without needing to code a bunch of visuals from scratch.
- **Features and Functionality:** It has drag-and-drop capabilities, support for slicers and filters, and can easily display things like course registration counts, student distribution, and schedule breakdowns.
- **AI/Data Integration:** Power BI includes smart features like built-in AI visuals, trends, and forecasting tools. It also supports DAX and Power Query, which helped me work with more complex data sets.
- **Ease of Use:** Since it's part of Microsoft's ecosystem, it has a familiar interface and was easy for our team to pick up. That helped speed up our dashboard development.
- **Cost and Scalability:** The Desktop version is free, which is perfect for student projects. And if we ever wanted to scale up to a web-based deployment or enterprise use, the Pro version is affordable.

6. Testing and Quality Assurance Process

- **Unit Testing:** We used Jest to test individual backend functions like validating course availability and checking input fields. This matches the shift-left testing concept from class, where we aim to catch issues early in development.
- **Integration Testing:** For testing how different parts of the app connect, we used Supertest to verify that our API endpoints correctly interact with the MySQL database on AWS RDS. This supports the layered MVC structure we talked about in INFO 465.
- **End-to-End Testing:** We used Postman to simulate full workflows like registering a student or dropping a class. it was straightforward to learn, and it helped us test the whole system from the user's perspective.
- We plan on using Jest for unit testing, Supertest for integration testing, and Postman to simulate real user actions. These tools help us test the app in layers, following the QA process we learned in class. While we didn't build a full CI/CD pipeline, our workflow follows DevOps practices using Git and regular testing during development.

7. Authentication and Authorization

Access Control:
The application is restricted to registered users only. Users must log in to access any features. No public or guest access is allowed.

Defined Roles:

- **User** – Can access and manage their own data

- **Admin** – Can manage all users and view system settings

Authentication:
Users will log in using a username and password. Passwords will be securely stored using hashing (bcrypt).

Authorization:
User roles are checked after login to control what each person can see and do. This applies to both the interface and the backend routes.

8. Team Contribution Summary

- - Eryx Bautista – Cloud Architect and App Developer contributed to the worksheet by outlining the cloud infrastructure and application design choices to ensure our system is scalable, cost-effective, and aligned with the project requirements.
- - Sahithi Mudiganti – Network Engineer: Designed and documented the VPC network infrastructure using AWS best practices
- - Jorge – Database Architect: Designing the overall database schema to model the application's data entities and their relationships accurately and ensuring the schema adhered to third form norm by removing redundancy and using foreign keys to the diagram.
- - Tobey Peralta – QA Analyst & Project Lead: Wrote the QA/testing strategy, used Postman and Jest for testing, and helped keep the team organized.

Challenges & Resolutions:
Our team had trouble meeting at the same time because of different working schedules. We worked around this by using group chats and sharing updates regularly to stay on track.

9. GitHub Screenshot