Andy Jang, Kevin Chavez, Grant Derdevanis, and Ameha Getahun
Dr. McGarry
INFO 465-001
July 1, 2025

Architecture Build Assignment

Cloud Service Provider

Our cloud services will be provided by AWS. AWS is the best provider for our needs due to multiple reasons. Firstly, AWS provides elastic infrastructure that can easily and automatically scale with our system demands. We can easily and efficiently scale up when summer enrollment is high or down when summer enrollment is low. This also provides cost efficiency for our system as well, as we are only paying for what we need. Secondly, AWS has a wide variety of availability zones that allow us to operate at all times, providing backup options in case of server error. AWS offers services we will use for our databases (RDS), virtual machines (EC2), network security and private cloud (VPC).

We will use the VPC, EC2 and RDS services of AWS.

Operating System and Virtual Servers

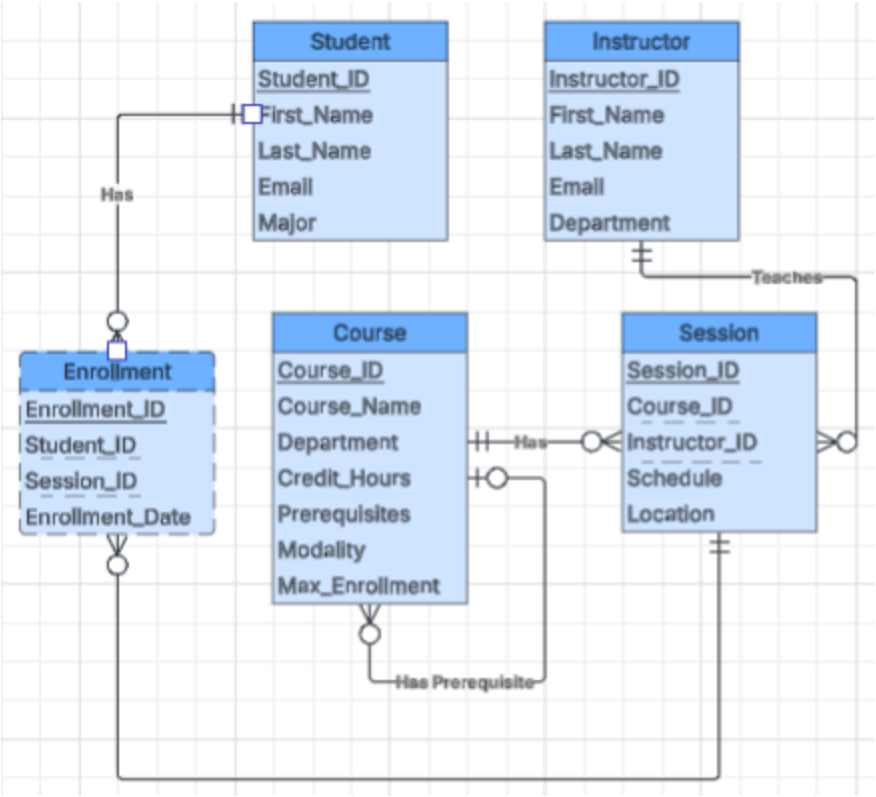
We will be using Windows for our virtual servers. We selected Windows because our team's general familiarity with Windows will make development more efficient, as well as integration with existing Windows-based development tools.

Our server configuration will be as follows:

Windows Server Base 2022
T2.medium
2 vCPU
4 GiB Memory
Storage: 1 volume(s) - 30 GiB

Database Design

ER Diagram:



Database Management System:

MySQL:

- Relational database support
- Compatibility with deployment in AWS RDS
- Scalability for future application growth

Data Visualization Tool

Tableau:

Functionality & Features-

- Industry-standard tool for building interactive dashboards
- Ideal for visualizing complex data relationships
- Supports filtering and real-time dashboards
- Easy-to-use interface with customizable charts, maps, and reports

AI & Data Integration-

- Tableau integrates well with MySQL and AWS RDS
- Basic predictive and trend analytics are built in
- Strong support for forecasting and clustering

Team Familiarity & Ease of Use-

- All members of Team 3 have used Tableau in previous courses
- Intuitive for users with less technical knowledge

- Online resources to learn how to use this application are widely available

Cost & Scalability-

- Tableau has a free-to-use version
- Scales well for enterprise use

Other Relevant Factors-

- Easy to prototype dashboards quickly during the development process
- Visually very professional and clean

Testing and Quality Assurance Process

Unit Testing

Plan:
Unit testing will focus on validating individual components or modules of the application in isolation. Each function, class, or module will be tested independently to ensure it performs as expected.

Tools:

- For JavaScript/Node.js, we will use Jest or Mocha as the testing framework, combined with Chai for assertions.
- Tests will mock dependencies (e.g., databases or APIs) using libraries like Sinon to isolate the unit under test.

Process:

- Developers will write unit tests alongside feature development (Test-Driven Development approach encouraged).
- Tests will be automated and run as part of the CI/CD pipeline (e.g., GitHub Actions) to ensure no regressions are introduced.

Integration Testing

Approach:

Integration testing will verify how different modules or services interact with each other. This includes testing API endpoints, database interactions, and communication between microservices (if applicable).

Tools:

- Supertest for testing RESTful APIs in Node.js.
- Postman for manual and automated API testing.

Process:

- Test scenarios will cover successful and error cases (e.g., invalid inputs, failed dependencies).
- Realistic test data will be used to simulate production behavior.

End-to-End Testing

Strategy:
End-to-end (E2E) testing will validate the entire application workflow from a user's perspective, ensuring all integrated components function together as expected.
Tools:

- Cypress or Selenium for browser automation.
- Puppeteer for headless testing.

Process:

- Critical user journeys (e.g., login, data submission, dashboard interactions) will be scripted and automated.
- Tests will run in a staging environment mirroring production.

Authentication and Authorization Process

Access and Roles

1. System Accessibility:

    - The application will be open to anyone (no authentication required) as per the assignment scope.

    - This simplifies initial development and testing but is not suitable for production environments with sensitive data.

2. Roles (Placeholder for Future Implementation):

    - **Administrator**: Manages system settings, user access, and data.

    - **Student/User**: Interacts with the application's core features (e.g., submitting data, viewing dashboards).

Justification for Omission of Authentication

- **Scope Limitation**: The assignment explicitly states that authentication/authorization is out of scope.

- **Focus on Core Architecture**: Resources are prioritized for designing scalable infrastructure, database schema, and testing processes.

- **Future Considerations**: In production, tools like AWS Cognito, OAuth 2.0, or LDAP would be implemented for secure access control.

Application Design

- **Programming Language**: JavaScript

- **Runtime Environment**: Node.js

- **Application API**: RESTful API using Express.js

- **Application Framework**: None (pure Node/Express); however, a front-end framework like React could be added in future development.

- **Middleware**: Express.js is used to manage API routing, error handling, and middleware functions such as body parsing and CORS headers.

Justification:
Using Node.js with Express allows for rapid API development with a lightweight, modular setup. JavaScript is familiar to the team and allows for both front-end and back-end work to share a language, improving productivity and reducing learning overhead.

Network Architecture and Design

We will design a secure and scalable network architecture using AWS Virtual Private Cloud (VPC).

- **VPC**: One primary VPC configured to isolate resources securely.

- **Subnets**:

    - **Public Subnet**: For EC2 instances running the application, spread across two availability zones for high availability.

    - **Private Subnet**: For the RDS MySQL database, also spread across two AZs.

- **Internet Gateway**: Attached to the VPC, providing internet access to instances in public subnets.
- **NAT Gateway**: Allows private subnets to access the internet without exposing the database to public IPs.

Security Groups:

- **EC2 Application Security Group**
        - Port 80 (HTTP) — open to the world
        - Port 443 (HTTPS) — open to the world
        - Port 3389 (RDP) — restricted to admin IPs for troubleshooting
        - Port 22 (SSH) — restricted to admin IPs
    - Allow all outbound
- **RDS MySQL Security Group**
        - Port 3306 — allow only from EC2 app security group
            - Allow all outbound

Team Contribution

Name	Role	Contribution
Grant Derdevanis	Cloud Infrastructure	- Cloud Service Provider - Operating System & Virtual Servers
Andy Jang	Project Manager, Database Architect	- Database Design - Data Visualization Tool - Project Management
Ameha Getahun	QA Analyst	- Testing and Quality Assurance Process - Authentication and Authorization Process
Kevin Chavez	Application Developer, Network Engineer	- Application Design - Express API, port and network security group setup, subnet/VPC diagram,

Challenges

One of the primary challenges our group encountered was coordinating around conflicting schedules, especially given the tight timeframe for this project. To overcome this, we divided responsibilities strategically—allowing one half of the group to begin work on the project while the other half picked up the next phase as their schedules permitted. This approach ensured steady progress while accommodating each team member's availability, ultimately allowing us to collaborate effectively despite time constraints.