Architecture Document

Cloud Service Provider (Jed)

We will be using AWS as our cloud service provider, some reasons are its cost effective for students, its the biggest cloud provider and it gives us all the pieces we need in one place

Services:

- EC2/Fargate
- RDS (MySQL)
- S3
- Load Balancer
- VPC

Application Design (Jed)

Programing Language: JavaScript

Runtime Environment: Node.js

Application API: RESTful API

Application Framework: React

Middleware: Express

Ports: The app listens on port 8080, but users will reach it through 80 (HTTP) or 443 (HTTPS)

Operating System (Jed)

For our operating system we will run it on Linux servers (Amazon Linux 2023) because it's cheaper, lighter and faster, and easy for cloud apps to run on.

Server Configuration:

- Instance Size: t4g.small
- CPU: 2 vCPUs

- Memory: 2 GB RAM
- Storage: 20 GB gp3 EBS

Database Schema and Design (Joseph)

The database management system selected is MySQL, hosted through Amazon RDS. The following ERD attempts to follow the third normal form (3NF) to eliminate redundancy and maintain referential integrity. Key entities include:

- **Users**: UserID, Name, Email, Role
- **Courses**: CourseID, Title, Description, InstructorID
- **Enrollments**: EnrollmentID, UserID, CourseID, DateEnrolled
- **Assessments**: AssessmentID, CourseID, Title, MaxScore
- **Submissions**: SubmissionID, AssessmentID, UserID, Score, DateSubmitted

The relationships are structured as:

- One-to-many between **Users** and **Enrollments**
- One-to-many between **Courses** and **Assessments**
- One-to-many between **Assessments** and **Submissions**

This design supports scalability by isolating entities into clean, relational structures, ensuring easy expansion for future features (e.g., analytics, additional user roles).

Network Architecture and Security (Joseph)

We will be using **AWS Virtual Private Cloud** (VPC).

Core Components

- **VPC:** A dedicated isolated environment with logically separated resources.
- **Subnets:**
    - **Public Subnets:** For load balancers and bastion hosts.

    - **Private Subnets:** For application servers and databases, ensuring sensitive resources remain inaccessible from the public internet.
    - Each subnet is **redundant across multiple Availability Zones** for high availability.
- **Internet Gateway:** Provides public access to the load balancer and application tier where appropriate.
- **NAT Gateway:** Allows private subnet resources to initiate outbound connections (e.g., for updates) without exposing them to inbound traffic.

Security Measures

- **Security Groups:** Configured with strict ingress and egress rules.
    - **Application Servers:** Allow traffic on port **80 (HTTP)** and **443 (HTTPS)**, along with restricted administrative access (e.g., SSH on port 22).
    - **Database Servers:** Allow inbound traffic only from application servers on port **3306 (MySQL)**.
    - **Diagnostic Tools:** ICMP (ping) is allowed internally for troubleshooting, while RDP/SSH access is restricted to the bastion host.

Data Visualization Tool (Joseph)

For this project, the selected data visualization tool is **Tableau**.

- **Functionality and Features**: Tableau provides powerful interactive dashboards, advanced visualizations, and seamless integration with MySQL and AWS data sources. Its drag-and-drop interface allows users to create complex visualizations without heavy coding.
- **AI and Data Integration**: Tableau includes predictive analytics, trend analysis, and AI-driven features like “Explain Data,” which help uncover insights automatically. It also integrates well with external data science platforms for advanced modeling.
- **Ease of Use**: Tableau is intuitive for both technical and non-technical users. Its visual nature makes it easier for stakeholders to interpret data quickly, while still offering depth for more advanced analysis.
- **Cost and Scalability**: Tableau offers flexible licensing models (Tableau Desktop, Server, and Online) that scale with organizational needs. It is well suited for both small teams and enterprise environments.

Testing and Quality Assurance Process (Aidan)

To ensure code quality and system reliability during development, our team will implement multiple levels of testing:

- **Unit Testing**: Each module or component will be tested in isolation. Jest will be the primary tool for JavaScript-based testing. The goal is to validate input/output correctness for each function or method.

- **Integration Testing**: After individual modules pass unit tests, we will verify their interaction. API endpoints will be tested with Postman to confirm correct data exchange between the application and database.

- **End-to-End Testing**: Full workflows will be tested from the user's perspective. For example, registering a student and submitting data will be tested through the UI to ensure proper backend execution. Cypress will be considered for automated end-to-end scenarios.

- **Bug Tracking and Retesting**: Issues will be tracked in GitHub Issues. Each bug will be assigned, fixed, and retested before closing.

- **Sprint Reviews**: At the end of each sprint, the QA plan requires re-running core test cases to validate production readiness.

Authentication and Authorization Process (Aidan)

For this system, open access will be assumed, meaning any user can reach the application without login credentials. Roles will still be defined within the design for clarity:

- **Student**: Can access and submit content.

- **Administrator**: Can review submissions, manage data, and monitor system activity.

No formal authentication service (e.g., LDAP, OAuth) is required, and authorization mechanisms are out of scope, per assignment requirements. The assumption of an open system simplifies testing and deployment.

Team Responsibilities and Contributions/Summary (Aidan)

- **Aidan (Project Manager, QA Analyst, Repo Owner)**: Coordinated the team's activities, ensured timely progress, and wrote the Testing/QA, Authentication/Authorization, and Team Responsibilities sections. Created and maintained the GitHub repository, posting the final deliverable and providing proof.

- **Jed (Application Developer, Cloud Architect)**: Defined the Cloud Service Provider (AWS) and justified its selection. Designed the Application layer, including programming language, runtime, framework, middleware, OS, server configuration, and port usage.

- **Joseph (Database Architect, Network Engineer)**: Built the ER diagram and database schema in third normal form. Designed the Network Architecture, including VPC, subnets, firewalls, and security groups. Selected and justified the data visualization tool.

- **Collaboration**: Work was divided according to each member's role. Challenges primarily involved coordinating diagram creation across tools. This was resolved by assigning Joseph sole responsibility for diagram completion, while others provided review feedback.
