Group 1 | INFO 465 Architecture Assignment

**Architecture Document — Course Registration System**

**1. Introduction**

This document outlines the technical architecture for the Course Registration System. The architecture defines the cloud infrastructure, application design, database schema, network configuration, and security measures necessary to support performance, scalability, and reliability. It will serve as the foundation for subsequent builds and sprints.

**2. Cloud Provider and Services**

**Selected Provider:** Amazon Web Services (AWS)

**Justification:**

AWS allows scalability with a pay-as-you-go-model. Grants flexibility in scaling resources up or down without upfront investment.

Better services in regard to project requirements, including Database and Network versatility that Microsoft Azure and Google Cloud lack.

**Key Services:**

* **EC2:** Virtual servers to run the Node.js application.
* **RDS (MySQL):** Managed relational database with Multi-AZ support.
* **S3:** Storage for static assets, logs, and backups.
* **Elastic Load Balancer (ALB):** Distributes traffic across application servers.
* **Auto Scaling Groups:** Ensure the application scales based on demand.
* **VPC:** Isolated network with public and private subnets.
* **CloudWatch:** Logging, monitoring, and alarms.
* **IAM & Secrets Manager:** Identity, access, and credential management.

**3. Application Design**

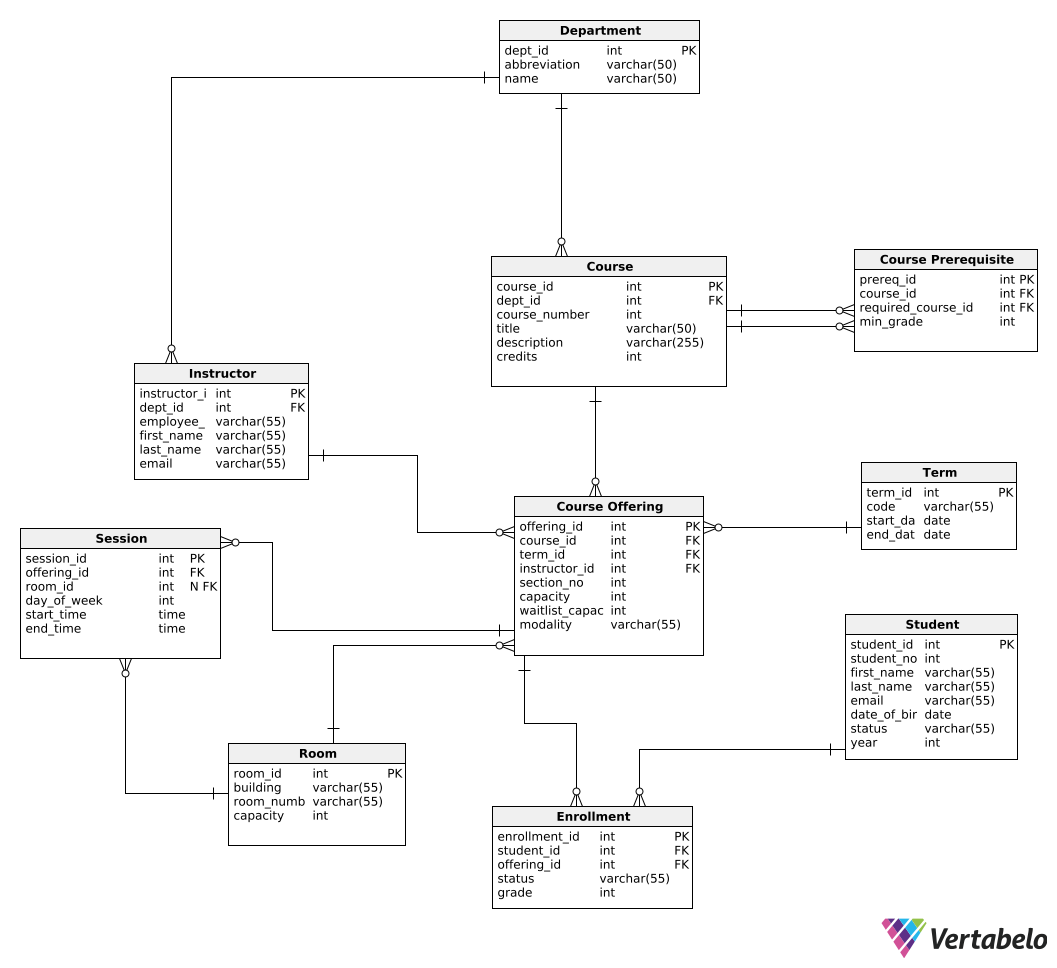
* **Programming Language:** JavaScript
* **Runtime Environment:** Node.js (v18+)
* **API:** RESTful API
* **Framework:** React (frontend), Express.js (backend middleware)

**4. Operating System and Virtual Servers**

**Operating System:** Amazon Linux 2 (lightweight, optimized for AWS, secure, cost-effective).

* **Application Servers:** t3.medium (2 vCPU, 4 GB RAM, 50 GB gp3 storage)
* **RDS Database:** db.t3.medium (100 GB gp3 storage, Multi-AZ enabled)
* **Optional Bastion Host:** t3.micro (if SSH access required).

**5. Database Design**

**Database Engine:** MySQL

**Normalization:** Schema adheres to **Third Normal Form (3NF)** to eliminate redundancy and ensure data integrity.

**6. Network Architecture and Security**

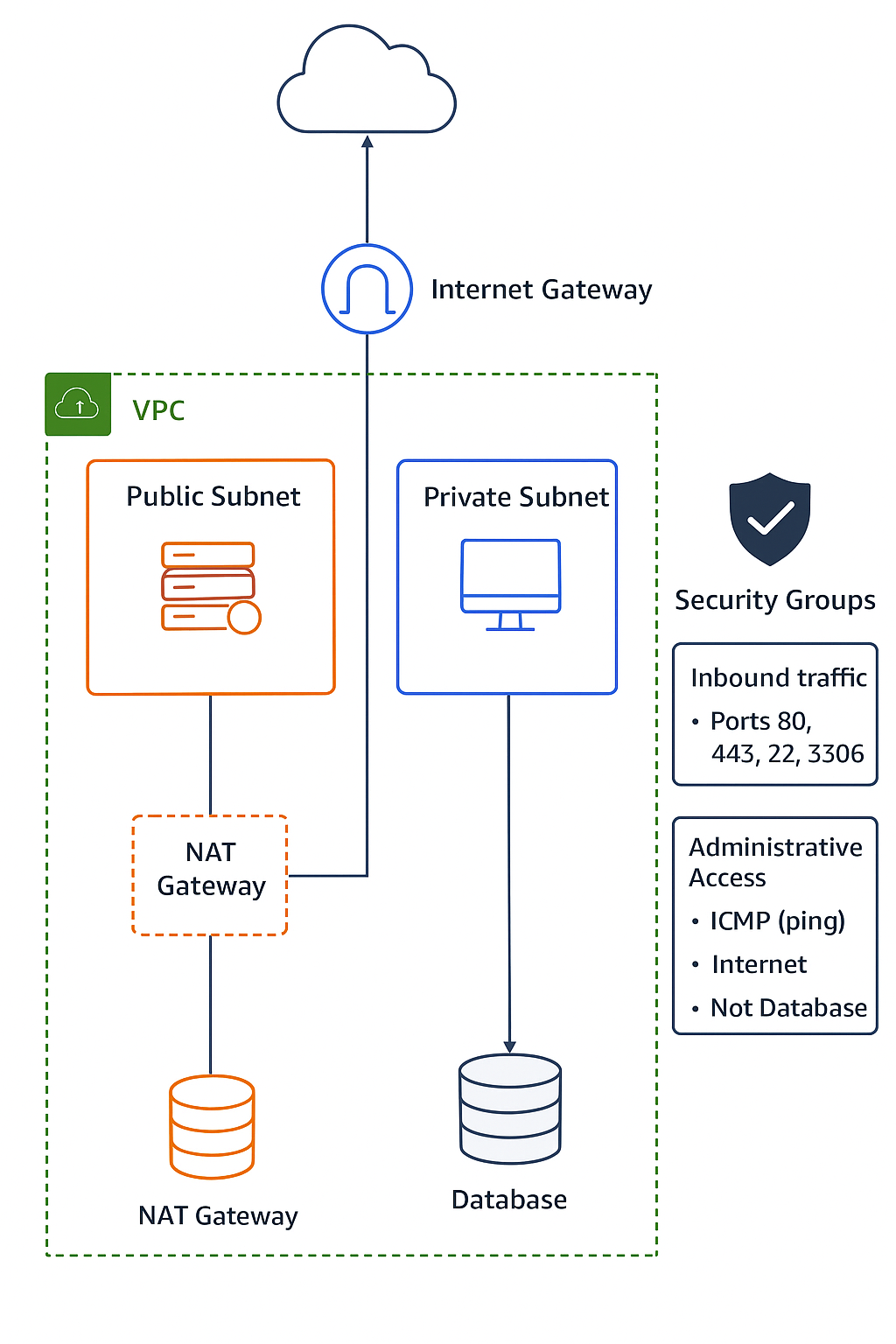
**VPC Design:**

* **Two Availability Zones** for resiliency. US-east-1a, US-east-1b
* **Subnets:**
  + **Public Subnets:** Contain ALB and NAT Gateway. 2 Public subnets, one in each availability zone.
  + **Private Subnets:** Contain app servers and RDS database. 2 Private subnets, one in each availability zone.

**Firewalls and Security Groups:**

* **Internet Gateway:** Provides internet access to public subnets.
* Application Security Groups:
  + Allow port 80 and port 443 (HTTP and HTTPS)
  + Allow inbound traffic from the load balancer to the application servers on desired custom ports ( for Node.js / Express)
  + Inbound traffic:
    - Ports 80,443,22,3306 (HTTP, HTTPS, SSH, MySQL)
  + Outbound traffic:
    - Anything to the internet just not anything to the databases, as those are sensitive.
  + Administrative Access:
    - ICMP (ping) is enabled for troubleshooting within the VPC.
* Security Considerations
  + Defense in Depth: Layered security, using VPC isolation, subnets, and encryption
  + Least Privilege: Access controls limit communication between tiers (web, app, then the database)
  + Database encryption thanks to AWS KMS
* NAT Gateway**:** Enables private subnets to access the internet securely.

**Network Architecture diagram:**

****

**Security Groups:**

* **ALB SG:** Allow HTTP(80)/HTTPS(443) from anywhere → forward to app servers.

**Required Ports:**

* HTTP (80), HTTPS (443), App (3000), MySQL (3306), SSH (22), RDP (3389 if Windows), ICMP (ping).

**7. Data Visualization Tool (Vincent)**

**Selected Tool:** Power BI

### **Data Visualization Tool Justification: Power BI**

Our team selected Power BI as the data visualization tool for this project based on its strong balance of functionality, integration, and cost-effectiveness:

* **Functionality and Features**
  + Power BI provides robust tools for transforming, visualizing, and sharing data.
  + Dashboards allow stakeholders to interpret trends and make data-driven decisions.
  + Deployment pipelines and version controls give tighter management of business logic compared to alternatives like Tableau or Looker Studio.
* **AI and Data Integration Capabilities**
  + Better integration of AWS, such as visualizations and analytics.
  + AI features such as anomaly detection and predictive analytics
* **Team Familiarity and Ease of Use**
  + Power BI has an intuitive, user-friendly interface, lowering the learning curve for our team.
  + Many team members already have experience with Microsoft products, making adoption faster and smoother.
* **Cost and Scalability Considerations**
  + Power BI is more cost-effective than its competitors
  + Better cost within varied scalability

**8. Testing and Quality Assurance**

**Unit Testing:**

* Tools: Jest (backend), React Testing Library (frontend).
* Validates individual functions and modules. Going into each repo branch and validating each line.
* Process: We will use Jest and React to validate individual modules and functions. A coverage target of 80% or higher will be required before merging to GitHub, ensuring early detection of bugs and reliable components.

**Integration Testing:**

* Tools: Supertest, Postman/Newman.
* Ensures services (API, DB) work together.
* Strategy: Smooth interaction between the application’s services (APIs, database, and external components).

**End-to-End Testing:**

* Tools: Cypress or Playwright.
* Validates entire user workflows (e.g., registration).
* Strategy: Simulate user behavior to verify the entire system works as intended from a user’s perspective. Simulate user behavior with mock data ranging from small amounts of data to large amounts. This is done in order to make sure scalability works for all types of users.

**CI/CD Pipeline:**

* GitHub Actions runs pushing, pulling, linting, unit tests, builds, and deploys.
* GitHub will be used for the pipeline over GitLab, as GitHub satisfies the scope of our needs.
* Our GitHub repository will also have integrated branches set up, so every group member has their own branch. This way, we have better organization in our repository
* Automated tests run on pull requests and staging environments.
* Strategy: Automate testing, code quality checks, and deployments for continuous feedback.

**9. Authentication and Authorization**

**Access Model:**

* Anyone can access via a student role however, we will be utilizing the principle of least privilege. For student access, we will only give privileges based on what is needed, but only members of group 1 get administrative roles / privileged roles.
* Open-Access since the authorization process is out of scope for development.

**10. Team Responsibilities**

* **Caleb Heino / Kevon Mahban - Cloud Architect:** AWS services selection, VPC design, Auto Scaling, IAM.
* **Vincent Arcuri / Kevon Mahban - Application Developer:** API design, runtime setup, port configuration.
* **Eddie Chen / Vincent Arcuri - Database Architect:** ER diagram, schema design, normalization.
* **Kevon Mahban / Eddie Chen - Network Engineer:** Subnets, firewalls, security groups.
* **Vincent Arcuri / Eddie Chen - QA Analyst:** Test plan, automated test scripts, CI integration.
* **Caleb Heino / Kevon Mahban - Project Manager:** Oversees tasks, ensures GitHub deliverables.

**11. Collaborative GitHub**

[*https://github.com/Alfredio5/Team-1-Repo.git*](https://github.com/Alfredio5/Team-1-Repo.git)