Architecture Document

Cloud Service Provider:

Selected provider: AWS

Justification:

- We are choosing AWS because we are provided with a learner lab which is free to use and cost effective for us.
- AWS also can provide us with reliable service and availability.
- AWS also offers auto-scaling and load balancing.

Services being used:

- We will create and use a VPC to isolate networks.
- EC2 will also be implemented for backend applications.
- S3 will be used for files and backups.
- Subnets to split networks into public and private. (Used in VPC)
- Security groups for traffic control. (Used in VPC)

Application Design:

- **Programming Language:** We chose to work with JavaScript because it is widely used and it lets us develop both the front and back end of our application at an efficient rate.
- **Run-time Environment:** Node.js was chosen for the run-time environment because of the fast performance it provides, especially when it is required to handle multiple tasks at once, which is important for applications that are required to perform well when there is a lot of usage.
- **Application API:** Our application is going to use RESTful API and the reason being is because it is straightforward and works great on the web, which makes our application simple to maintain and scale.
- **Application Framework:** We decided to choose React as our application framework because it excels in the efficiency of managing and updating the applications user interface.

  Middleware: Express.js is what we picked to go with for the middleware because it is not only powerful, but also simple. This helps us in managing routes and requests proficiently without causing the application to slow down.

Operating System and Virtual Servers:

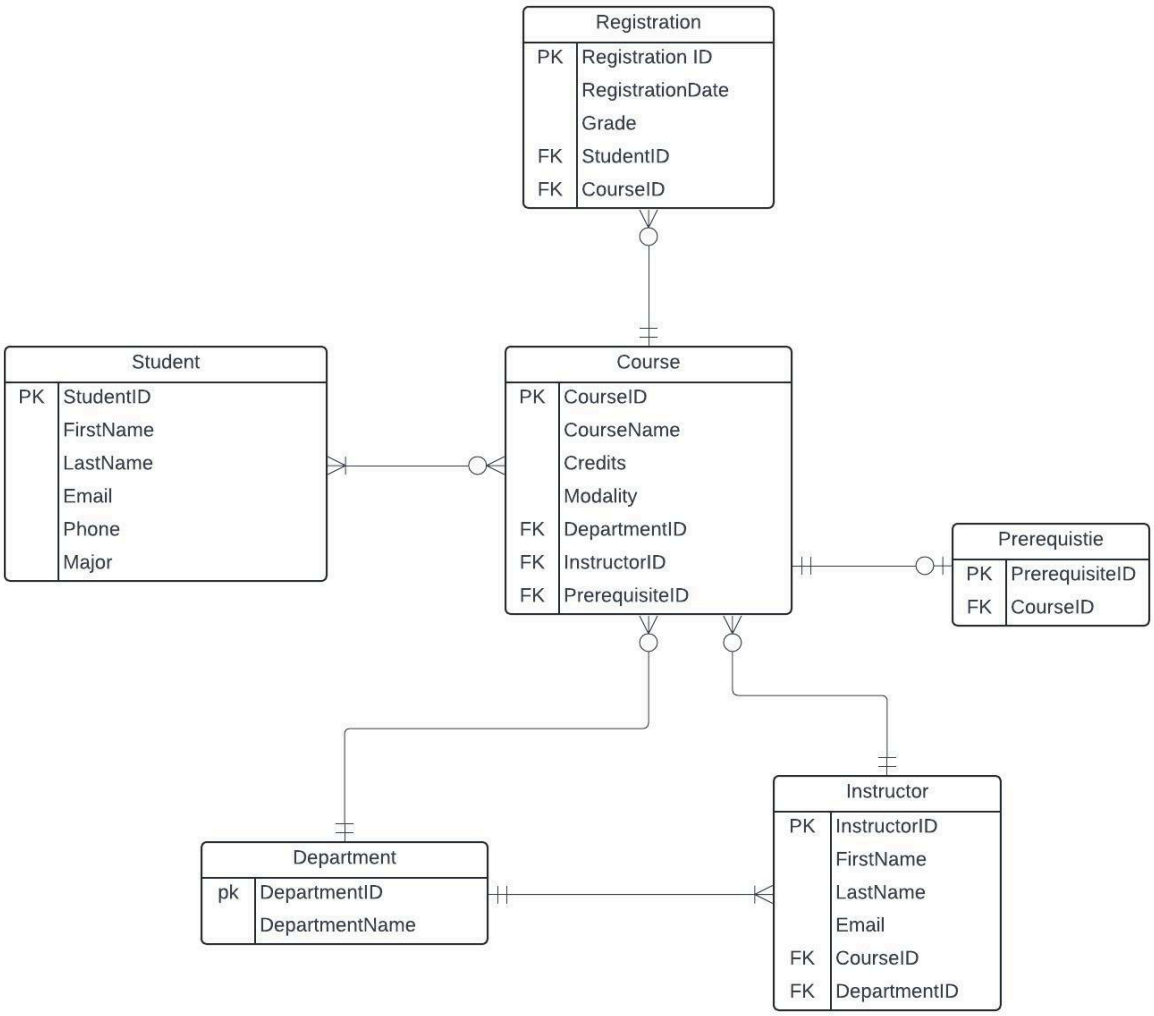
- The operating system that we decided to use was Linux for our virtual servers. Linux is overall preferred in this instance in our opinion for many different reasons such as it being widely supported, the great security features, and its efficiency in the way it handles server operations. It being open-source also helps reduce the overall project costs.

  Server Configuration:
- The instance type that we will use is Amazon EC2 instances for the backend application servers. The specific instance type will be chosen based on the requirements for the performance in regards to the CPU, memory, and storage needs.

- Two CPUs and 4GB RAM was what was decided to be a good starting point for our project. Having two CPUs and 4GB RAM is sufficient enough for dealing with moderate traffic but can also be adjusted based on the real-time demand.
- EBS volumes are going to be attached to each instance for constant storage of data. We are going to start with 50GB and if we need more, we will increase it.

Database Design:

- **Entity Relationship Diagram:**



-

Database Management System (DBMS)

- We have chosen to use MySQL as our database management system because of its performance and efficiency. MySQL also allows for replications which allows the database to scale out efficiently over multiple servers. MySQL is user-friendly, cost effective, and is also compatible for web-based applications which makes it superior for us.

Network Architecture and Design:

NETWORK ARCHITECTURE OVERVIEW

- **Virtual Private Cloud (VPC) – Isolates resources for security & scalability.**
- **Subnets:**
    - **Public Subnet: Hosts application servers, load balancer, and web-facing services.**
    - **Private Subnet: Hosts the database, restricting external access.**
- **Firewalls (Security Groups & Network ACLs):**
    - **Control inbound/outbound traffic to prevent unauthorized access.**

---

2. SECURITY GROUP CONFIGURATION

Application Security Group (Public Subnet)

- **Ingress Rules (Incoming Traffic):**
    - **Port 80 (HTTP): Allows standard web traffic.**
    - **Port 443 (HTTPS): Allows encrypted communication.**
    - **Port 22 (SSH): Restricted to a trusted admin IP for maintenance.**
- **Egress Rules (Outgoing Traffic):**
    - **Allows outgoing requests to the database on Port 3306 (MySQL) or 1433 (SQL Server).**
    - **Allows outbound internet access for API calls & updates.**

Database Security Group (Private Subnet)

- **Ingress Rules (Incoming Traffic):**
    - **Only allows connections from the Application Server on database ports.**
    - **Blocks direct internet access for security.**
- **Egress Rules (Outgoing Traffic):**
    - **Allows responses only to the application servers (no external access).**

---

3. PORT CONFIGURATIONS

Application Server (Public Subnet)

- **Port 80 (HTTP) – Web traffic.**
- **Port 443 (HTTPS) – Secure connections.**
- **Port 5000/8080 (Custom API Port) – If needed for backend API communication.**

Database Server (Private Subnet)

- **Port 3306 (MySQL) – Restricted to application servers.**
- **Port 1433 (SQL Server) – If using Microsoft SQL.**
- **Port 5432 (PostgreSQL) – Optional custom database port for security.**

---

4. ARCHITECTURE OVERVIEW

VPC Structure:

- **Public Subnet: Load balancer + application servers.**
- **Private Subnet: Database servers.**

Traffic Flow:

- **User → Load Balancer → App Server (Public Subnet) → Database (Private Subnet).**

Firewall & Security Controls:

- **Restrict access to only necessary ports.**
- **Block external access to the database.**
- **Secure web traffic with HTTPS.**

SUMMARY

VPC with Public & Private Subnets – Secure & scalable.
Strict Security Group Rules – Prevents unauthorized access.
Proper Port Configuration – Ensures safe communication.
Load Balancer & Firewalls – Improves performance & protection.

-

Data Visualization Tool:

- **Functionality & Features:**
  - Interactive dashboards for real-time tracking.
  - Drag-and-drop interface for easy report creation.
  - Data blending & filters for in-depth analysis.
- **AI & Data Integration:**
  - AI-driven insights to predict course demand.

    - Connects to multiple data sources (databases, Excel, cloud).
    - Supports real-time data updates.
- **Ease of Use & Familiarity:**
    - User-friendly, no extensive coding required.
    - Plenty of tutorials & training resources.
    - Suitable for both technical & non-technical users.
- **Cost & Scalability:**
    - Flexible pricing for different needs (individual, team, enterprise).
    - Scalable for small or large implementations.
    - Available as cloud-based or on-premise solution.
- **Other Benefits:**
    - Customizable reports for students, advisors, and admins.
    - Easy collaboration & sharing of insights.
    - Strong security with role-based access control.

Conclusion:

Tableau is **powerful, user-friendly, scalable, and AI-driven**, making it a great choice for optimizing course registration data analysis and decision-making.

-

Testing and QA Process:

- For testing we opted to use JUnit tests as they are easy to configure and give way to test for specific methods, classes, and functions that would be found in our code or work. We also have experience with this, JUnit test requires us to write our own tests for each module before we upload them allowing us to check our own work.
- For quality assurance we have opted to use students in VCU as they would be the primary users of our platform and ensuring that it is both easy to use and intuitive to students is a priority for seamless class selection & management.

Authentications and Authorization Process:

- We have chosen to allow anyone affiliated with VCU and who own a VCU email account to be able to access our system, we have chosen this so that not only instructors and students would be able to use our platform, but also advisors & administrators.
- For our authorization process we have chosen to use DUO as it is already used within our school and would offer a smooth transition to a new system.

Github Screenshot:

Team Contribution Summary

Team Responsibilities/Contributions

- Sushanth took care of justifying our cloud service provider as well as defining and selecting the services that we will be using in our architecture. Sushanth also created the database design ERD provided above and specified and justified the database management system that we will be using (MySQL).
- Nihad worked on our application design which included defining our applications programming language, run-time environment, application API, application framework, and middleware. Nihad also worked on selecting our operating system and virtual server. We were deceived on using Windows and specified our server configuration above in the OS and VS section.
- Shakir worked on our blueprints for our testing and QA process as well as our authentication and authorization process. Testing and QA section specified and defined our unit testing, integration testing, and end-to-end testing processes. Our authentication and authorization section defined our user access, our roles within the application and our authentication process
- Hasan worked on our data visualization tool section which consisted of selecting which tool we will be using (Tableau) and justifying why we selected it. Hasan also completed our Network Architecture and design which illustrated our network architecture, and specifies otis port and security group configuration.

Team Challenges

- We faced several challenges while trying to design our application and lay out blueprints for future reference. The biggest problem was defining our requirements and what is necessary for our application. We had differing ideas about the scope and functionality of our project which led to refinement and more through discussions. Another challenge we faced was balancing ease of use, scalability, and compatibility with our project goals. This ultimately led to more research and deeper understanding of what our expectations are.