****

**Architecture Build**

1. **Cloud Service Provider**

**Chosen Cloud Provider: AWS**

AWS is chosen because:

* **Free tier option/$50 budget-** Many services(EC2, S3, RDS) offer free usage for the first 12 months.
* **Scalability-** AWS allows upgrading resources only when needed.
* **Managed Services -** AWS automatically handles, backups, security, and scaling.

1. **AWS Services Used ( Cost-Optimized)**

| **AWS Service** | **Purpose** | **Estimated Cost** |
| --- | --- | --- |
| EC2 (t3.micro) | Runs backend (Node.js & Express) | Free Tier |
| S3 | Stores frontend | Free Tier |
| RDS (MySQL, db.t3.Micro) | Stores user app data | Free Tier |
| API Gateway | Manages API traffic | $0.01 per 1m Calls |
| IAM | Secure access control | Free |
| CloudWatch (basic) | Monitor servers health | Free Tier |

**\**

1. **Basic Architecture Setup**

* **Frontend (React App):** Hosted on S3.
* **Backend (Node.js & Express):** Runs on EC2 Instance (t3.micro, Free Tier).
* **Database:** AWS RDS MySQL (Free Tier, 1 instance).
* **Security:** Only necessary ports are open for public access.

1. **Network and Security**

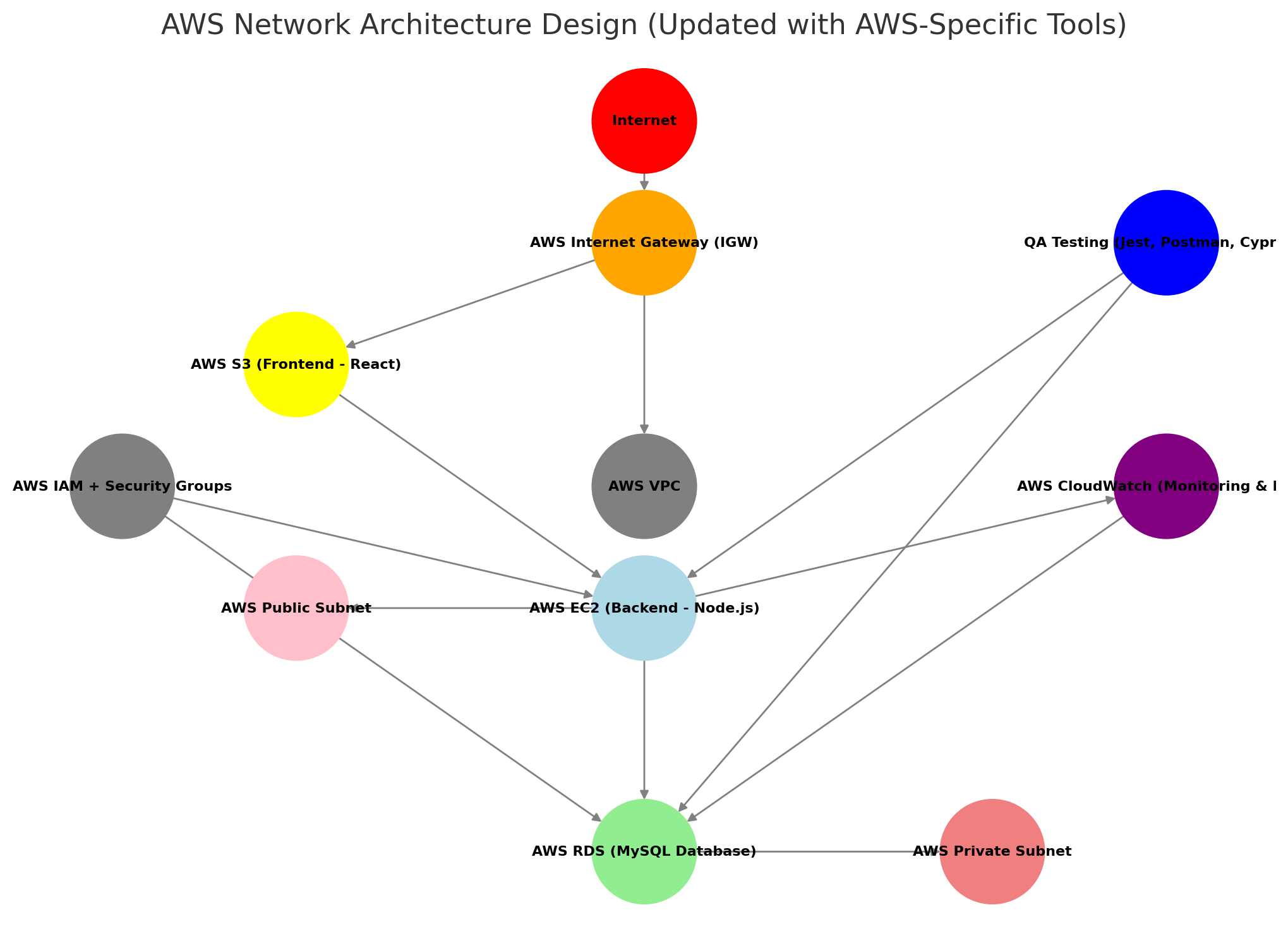
VPC with Two Subnets:

* **Public Subnet:** For EC2 (Backend)
* **Private Subnet:** For RDS (MySQL)

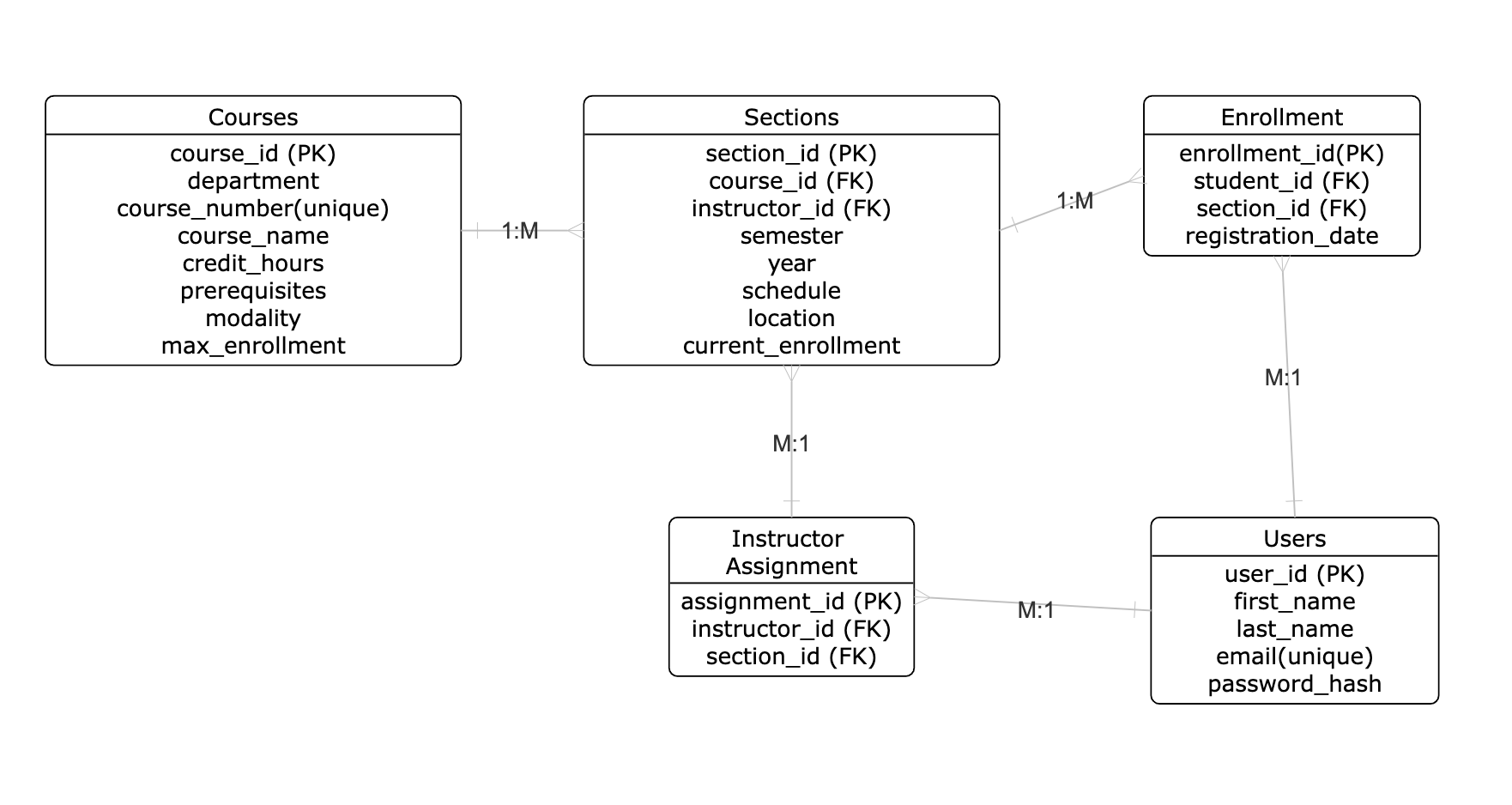
**Security Groups**

| **Service** | **Port** | **Source** |
| --- | --- | --- |
| React (S3) | 443 (HTTPS) | Public |
| Backend API (EC2) | 443 (HTTPS) | Public |
| Database (RDS MySQL) | 3306 | Private Only |
| SSH (Admin Access) | 22 | Admin Only |

1. **Application Design** 
   1. **Programming Language: JavaScript**
      1. Chosen for its asynchronous capabilities, rich ecosystem, and frontend-backend compatibility.
   2. **Run-time Environment: Node.js**
      1. Provides a lightweight and efficient environment, suitable for microservices and real-time applications.
   3. **Application API: RESTful API**
      1. Ensures interoperability, scalability, and ease of integration with other systems.
   4. **Application Framework: React.js (for frontend)**
      1. Offers a component-based architecture, optimized performance, and a rich UI/UX experience.
   5. **Middleware: Express.js**
      1. Enables efficient request handling, robust routing, and simplified API development.

****

**Online Course Registration System ER Diagram**

****

**Role Summary**

**Application Design- Mohamed Elnafe**

As the application designer, my responsibility was to create a user-friendly interface for our project. I developed wireframes and prototypes to ensure that everything was visually appealing and functioned smoothly. One challenge I encountered was ensuring that the design not only looked attractive but also integrated seamlessly with other components of the project, such as the cloud setup and data visualizations. To address this, I gathered feedback from my teammates and made adjustments to enhance usability. Collaborating closely with the team enabled us to ensure that the design aligned well with the overall project, ultimately improving the user experience.

**Cloud Architect - Mohamed Salman**

As the Cloud Architect, my primary responsibility was to design the AWS Cloud infrastructure for the application ensuring cost-effectiveness, scalability, security, and high availability while staying within the $50 budget. One of the main challenges I faced was balancing cost constraints while ensuring scalability and security. Given the $50 budget, selecting AWS services that provided optimal performance without exceeding costs was difficult. I would've liked to use Cloudfront and Auto-scaling but those services would have exceeded our cost.

**Network Engineer & QA Analyst- Robert Freeman**

As the Network Engineer, I designed a secure and scalable AWS network architecture, configuring VPCs, subnets, and security groups to ensure safe communication between the frontend (S3), backend (EC2), and database (RDS MySQL) while restricting unnecessary access. I placed RDS in a private subnet, allowing access only from EC2, and fine-tuned IAM roles and security groups to balance security and accessibility for API requests within AWS Free Tier limits.

As the QA Analyst, I developed a testing strategy to ensure smooth frontend, backend, and database interactions. Using Jest, Postman, and Cypress, I conducted unit, integration, and end-to-end testing to validate system functionality and security. A key challenge was optimizing API calls and database queries while maintaining performance. Through rigorous testing, I ensured a reliable, efficient, and scalable system with a seamless user experience.

**Database Architect - Andrew Slobodnik**

As the Database Architect, I was responsible for designing the relational database schema for the Online Course Registration system, making sure that it follows Third Normal Form (3NF). The schema has five key tables: Users, Courses, Sections, Enrollment, and Instructor Assignment. The database management system is MySQL.

Tableau is the optimal choice for our data visualization component of this project because of its powerful analytics capabilities, ability to integrate with AWS, and provision of interactive dashboards with real-time monitoring. With its ability to directly connect to AWS RDS (MySQL) and integrate with AWS CloudWatch, Tableau will significantly enhance our decision-making process. Tableau is also an ideal choice because of my prior experience working with the tool, which will streamline the setup process and ensure that dashboards are developed efficiently.