Project Title: Architecting a Modern Serverless Application with AWS

Project Description:

The goal is to design a scalable, secure, and performant architecture for a modern web and mobile application that will leverage AWS services to deliver a seamless user experience. The architecture should be serverless, ensuring cost-efficiency and automatic scaling.

Deliverables:

1. **Architecture Diagram:** Create a detailed and well-labeled diagram illustrating the proposed architecture. The diagram should clearly show:
    - **Networking:** How the application is structured across VPCs, subnets, and security groups.
    - **Components:** The main components of the system (web app, mobile app, backend services, databases, etc.) and their interactions.
    - **Data Flows:** How data moves between components, including data ingestion, processing, storage, and retrieval.
    - Example diagram https://d1.awsstatic.com/architecture-diagrams/ArchitectureDiagrams/mobile-web-serverless-RA.pdf?did=wp_card&trk=wp_card
2. **Component Design (High-Level):** Describe the responsibilities of each major component:
    - **Web/Mobile App:** Outline the frontend architecture and how it communicates with the backend.
    - **Backend Services:** Describe the functions and responsibilities of the serverless backend services (using AWS Lambda or Fargate).
    - **Event-Driven Components:** Explain how events are used to trigger actions and communicate between components.
    - **Data Storage:** Describe the types of data stored in each database (RDS, DynamoDB, Elasticsearch) and the rationale for using each.
    - **API:** Explain the API structure (REST or GraphQL) and how it facilitates communication between the frontend and backend.
3. **Security Considerations:** Address security aspects such as:
    - **Authentication and Authorization:** How user access will be managed and secured.
    - **Data Protection:** Strategies for encrypting data at rest and in transit.
    - **Network Security:** How the VPC, subnets, and security groups will be configured to protect resources.
4. **Scalability and High Availability:** Explain how the architecture can:
    - Scale horizontally to handle increased traffic or workload.
    - Ensure high availability and resilience to failures.
    - Automatically scale resources based on demand.
5. **DevOps Considerations:** Describe how you would implement:

    - **Infrastructure as Code (IaC):** Tools and practices to automate infrastructure provisioning and management.
    - **CI/CD Pipelines:** Processes for continuous integration and deployment to ensure smooth releases and rapid feedback loops.
    - **Monitoring and Logging:** Mechanisms for collecting logs, monitoring system health, and alerting on issues.
6. **Cost Optimization:** Discuss potential strategies for optimizing costs, such as:
    - Right-sizing resources based on actual usage patterns.
    - Using spot instances for non-critical workloads.
    - Leveraging reserved instances for predictable workloads.
7. **Documentation:** Provide clear and concise documentation within the `README.md` file, including:
    - Introduction and overview of the architecture
    - Explanation of the design choices and rationale
    - Instructions on how to set up and deploy the infrastructure (if applicable)

Evaluation Criteria:

1. **Architectural Soundness:** Is the proposed architecture scalable, secure, and appropriate for a modern web/mobile application?
2. **Technical Depth:** Does the candidate demonstrate a deep understanding of AWS services, serverless architecture, and event-driven design principles?
3. **Clarity and Communication:** Is the architecture diagram clear and well-organized? Is the documentation thorough and easy to understand?
4. **Feasibility and Practicality:** Is the proposed solution realistic and achievable?
5. **DevOps Maturity:** Does the candidate demonstrate an understanding of DevOps practices and how they apply to the proposed architecture?

Submission: The candidate should submit their solution in a public GitHub repository. The repository should include the architecture diagram, component design documents, and the detailed `README.md` file.