**Backend (Java Spring Boot):**

1. **Create a Spring Boot Project**:
   * Use Spring Initializr or Maven to create a new Spring Boot project.
   * Include dependencies like Spring Web, Spring Data JPA, etc., based on your requirements.
2. **Develop Backend Application**:
   * Define entities, repositories, services, controllers, and other necessary components for your application's functionality.
3. **Configure Database**:
   * Set up your database configuration in the application properties or YAML file (e.g., MySQL, PostgreSQL, MongoDB).
4. **Implement REST APIs**:
   * Create RESTful endpoints using **@RestController** and handle HTTP requests and responses.
5. **Security Implementation** (if required):
   * Implement Spring Security for authentication and authorization.
6. **Testing**:
   * Write unit tests and integration tests to ensure the correctness of your backend logic.
7. **Build Artifact**:
   * Build a JAR file or WAR file for your Spring Boot application.

**Frontend (Angular):**

1. **Create Angular Project**:
   * Use Angular CLI to create a new Angular project.
2. **Develop Frontend Application**:
   * Design UI components, services, modules, and other necessary elements for your frontend.
3. **Implement HTTP Services**:
   * Use Angular's HttpClient to communicate with backend REST APIs.
4. **Testing**:
   * Write tests for your Angular components and services.
5. **Build Artifact**:
   * Build the Angular project to generate the production-ready frontend assets.

**Deployment to AWS Cloud:**

1. **Setup AWS Account**:
   * Create an AWS account if you don’t have one.
2. **AWS Services for Deployment**:
   * Utilize services like Amazon EC2 (for backend deployment), Amazon S3 (for frontend assets), Amazon RDS or DynamoDB (for database), and possibly Amazon Elastic Beanstalk or ECS for containerized deployments.
3. **Backend Deployment**:
   * Set up an EC2 instance or use container services to deploy your Spring Boot application.
   * Configure security groups, IAM roles, and networking accordingly.
4. **Frontend Deployment**:
   * Upload the frontend build artifacts (HTML, CSS, JS files) to an Amazon S3 bucket or serve them through AWS Amplify, Elastic Beanstalk, or CloudFront.
5. **Database Setup**:
   * Configure your database instance on Amazon RDS or DynamoDB, and ensure it's accessible to your Spring Boot backend.
6. **Networking and Security**:
   * Configure security groups, IAM roles, VPC, subnets, and other network settings for secure communication between frontend, backend, and the database.
7. **Domain Configuration**:
   * Set up your domain using Route 53 or integrate with a third-party domain provider.
8. **Continuous Deployment**:
   * Implement CI/CD pipelines using AWS CodePipeline, GitHub Actions, or other similar tools for automated deployments.
9. **Monitoring and Scaling**:
   * Configure monitoring and scaling mechanisms using AWS CloudWatch, Auto Scaling, and other AWS services to ensure high availability and performance.

**Flowchart…**

