  
  
  
  
 **Cloud Computing**

***Serverless Registration WebApp***

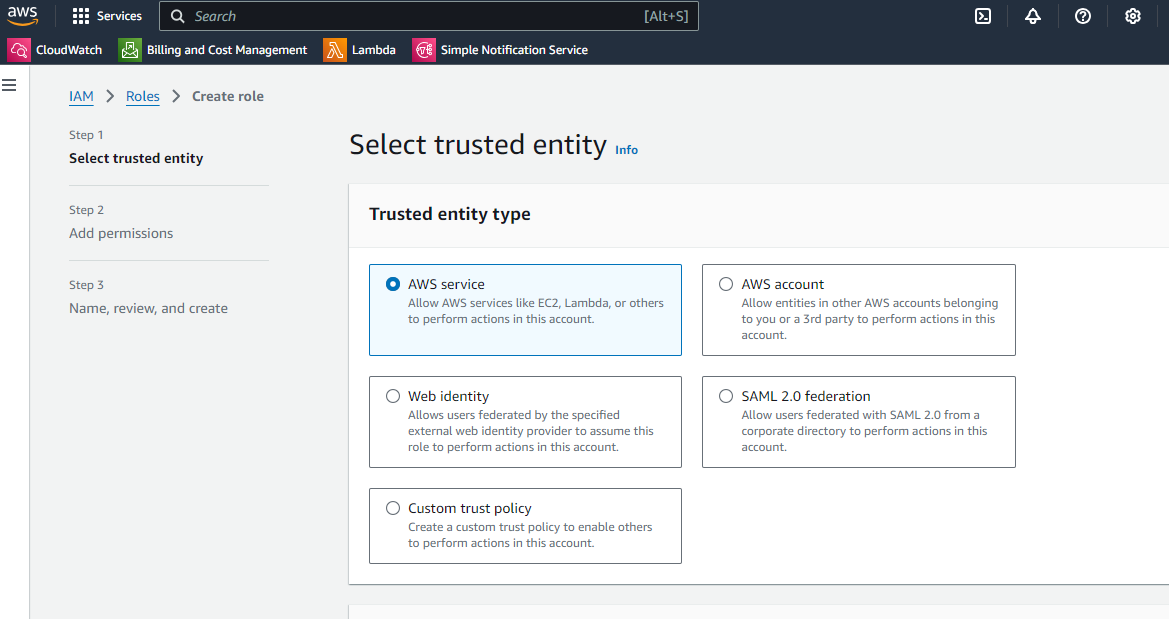
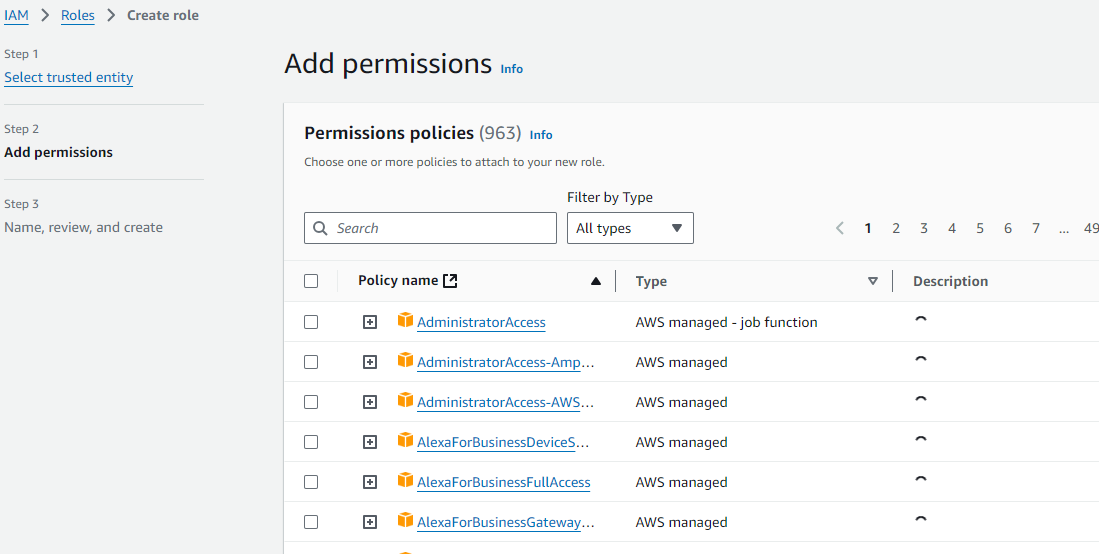
**Under the Guidance of**

*Dr. Aditi Sharma*

| **Manav Barik** | **21070122092** |
| --- | --- |

**Department of Computer Science**

***Introduction***

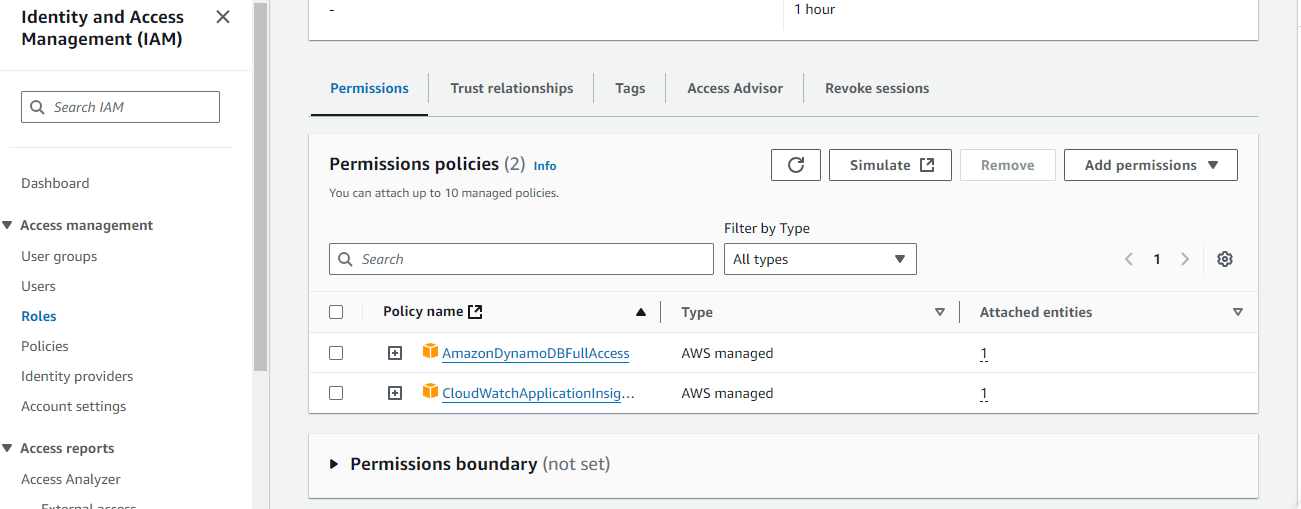
**Serverless architecture has revolutionized web development by allowing developers to build and run applications without managing servers. This presentation will focus on creating a serverless registration form using AWS services.  
  
  
*What is Serverless Architecture?***

**Serverless architecture allows developers to build and run applications without managing servers. It abstracts away the infrastructure, enabling developers to focus solely on writing code. This architecture is highly scalable and cost-effective, as you only pay for the compute time you consume.  
  
*Benefits of Serverless Architecture***

**Scalability: Automatically scales to handle varying loads.**

**Cost-Effective: Pay only for the compute time you use.**

**Maintenance-Free: No need to manage servers, reducing operational overhead.**

**Rapid Deployment: Quick and easy deployment of applications.  
  
  
CORS:**

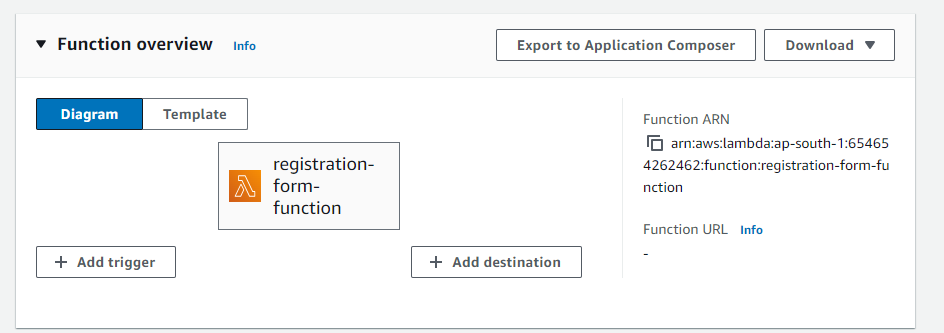
**Cross-Origin Resource Sharing (CORS) in AWS allows client web applications loaded in one domain to interact with resources in a different domain.**

***Components of a Serverless Registration Form***

**Frontend: The user interface where users fill out the registration form.**

**Backend: The serverless functions that process the form data.**

**Database: To store the submitted form data securely.  
  
Building a Serverless Registration Form**

****

**Step 1: Set Up the Frontend**

**HTML/CSS/JavaScript: Create the registration form using HTML, CSS, and JavaScript.**

**Static Website Hosting: Use services like AWS S3 to host the static website.**

**Step 2: Set Up the Backend**

**AWS Lambda: Write serverless functions to handle form submissions.**

**API Gateway: Expose the Lambda functions as RESTful APIs.**

**DynamoDB: Store the form data securely in a NoSQL database.**

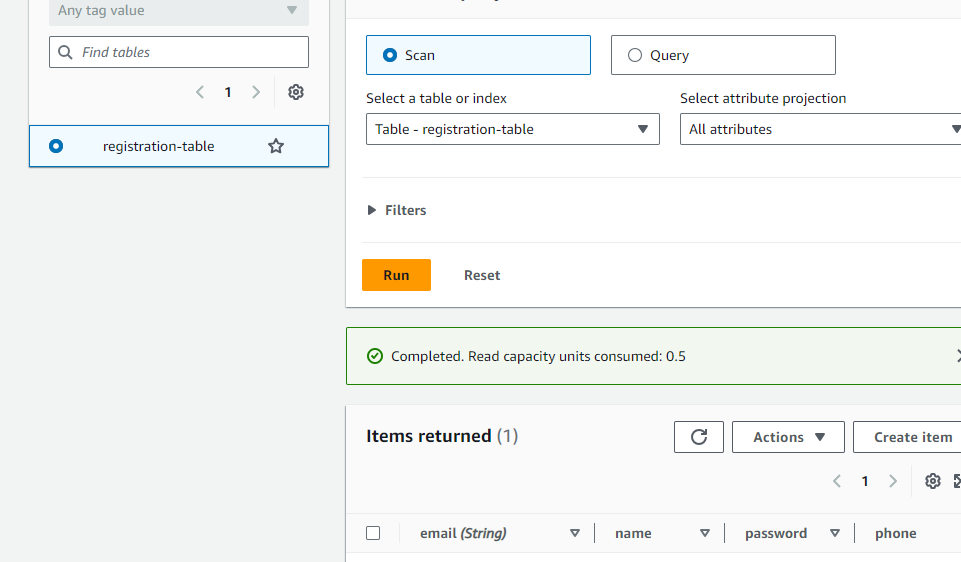
**Step 3: Integrate Frontend and Backend**

**Form Submission: Use JavaScript to send form data to the API Gateway.**

**Data Validation: Implement validation in the Lambda function to ensure data integrity.**

**Data Storage: Store validated data in DynamoDB.**

**Example: Building a Serverless Registration App with AWS Amplify**

**AWS Amplify provides a set of tools and services to build full-stack applications quickly.   
**

**Here’s a brief overview of how to build a serverless registration app using AWS Amplify:**

**Initialize Amplify: Set up your project with Amplify CLI.**

**Add Authentication: Use Amplify to add user authentication.**

**Add API: Create a GraphQL API to handle form submissions.**

**Add Storage: Use DynamoDB to store form data.**

**Deploy: Deploy the application to AWS.**

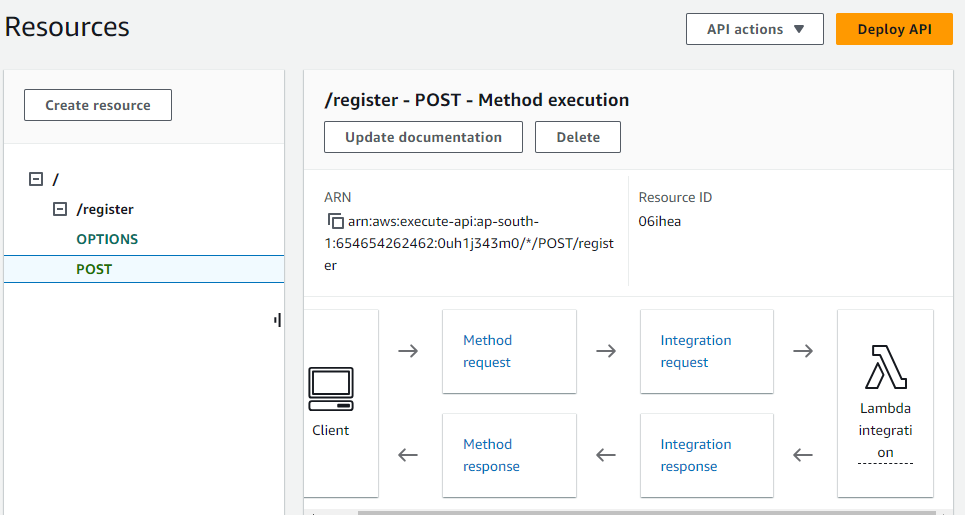
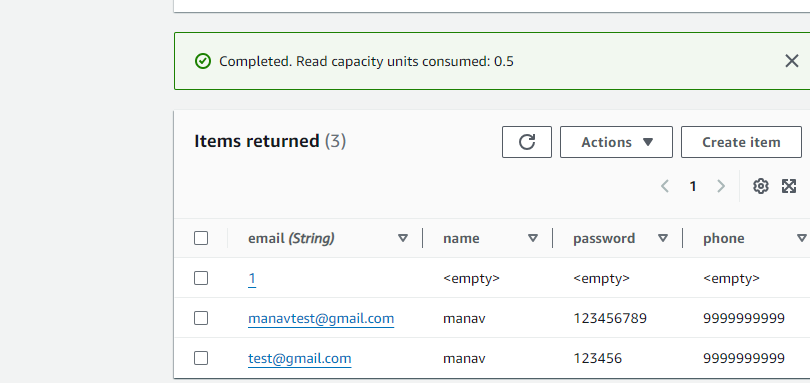
**Best Practices**

**Security: Implement proper security measures, such as input validation and encryption.**

**Error Handling: Handle errors gracefully to provide a better user experience.**

**Monitoring: Use AWS CloudWatch to monitor the application and set up alerts.**

**Conclusion**

**Serverless architecture offers a powerful and efficient way to build registration forms. By leveraging AWS services, developers can create scalable, cost-effective, and maintainable applications. Whether you are building a simple contact form or a complex registration system, serverless architecture provides the flexibility and performance needed to meet your requirements.  
  
  
  
  
**

**References**

**Serverless Registration Form using AWS Services: LinkedIn**

**Build a Serverless Full-Stack Registration App in minutes using AWS Amplify: AWS Blog**

**Building an Event Registration Page with Composition API and Serverless Functions: Netlify Blog**

**Creating Serverless Contact Forms with AWS: Medium**

**How to build a serverless contact form on AWS: Pluralsight**