Ankur Tandan
Full Stack 2
Final Reflection
Presentation link: https://youtu.be/X6cthEEM7K8

## Experiences and Strengths

How this course helps in reaching professional goals:

This course has taught me a lot about developing and deploying web applications in the cloud. I now understand how to use cloud services like AWS to build and manage full stack applications. These skills are important for my career because many companies are moving their applications to the cloud. Being able to develop cloud-based applications makes me a more valuable candidate in the job market.

Skills learned, developed, or mastered:

During this course, I learned how to work with Docker, create APIs using AWS Lambda, and manage data with DynamoDB. I also improved my skills in setting up cloud-based environments and testing applications. These skills are essential for modern software development, especially as more companies adopt cloud technologies.

Strengths as a software developer:

As a software developer, my strengths include problem-solving, attention to detail, and the ability to learn new technologies quickly. I am also strong in designing and building scalable applications that can handle growth over time. I am confident in working with both frontend and backend technologies, making me a versatile developer.

Roles I am prepared to assume:

With the skills I've gained, I am ready to take on roles such as a Cloud Developer, Full Stack Developer, or DevOps Engineer. I am comfortable with both development and deployment processes, which makes me a good fit for these positions.

## Planning for Growth

Using cloud services to plan for growth:

Planning for the future growth of my web application means thinking about how it will handle more users and more data. Cloud services like AWS make it easier to scale an application because they offer tools that automatically adjust resources based on demand. For example, I could use microservices or serverless architectures to break my

application into smaller parts that can be managed separately. This makes it easier to update, scale, and handle errors without affecting the whole system.

Handling scale and error handling:

To handle scale, I would use services like AWS Lambda and API Gateway, which automatically scale up when there is more traffic. For error handling, I would implement monitoring tools like CloudWatch to track issues in real time and set up automated alerts to respond quickly to problems.

Predicting the cost:

Predicting costs involves looking at how much each part of the application uses resources like computing power and storage. Serverless options, like AWS Lambda, usually offer more predictable costs because you only pay when your code is running. Containers might be more cost-effective if the application runs continuously because they use reserved resources that could be cheaper in the long run.

Pros and cons for expansion:

Pros of microservices/serverless: Easier to scale, more flexible, and you only pay for what you use.

Cons: Can be more complex to manage and debug, and there might be latency issues if not set up correctly.

Pros of containers: Consistent environments and potentially lower costs for long-running applications.

Cons: Require more management and might not scale as efficiently as serverless.

Elasticity and pay-for-service in decision-making:

Elasticity is crucial because it ensures that the application can handle changes in demand automatically without needing manual intervention. Pay-for-service is also important because it allows me to control costs by only paying for the resources I actually use. When planning for future growth, I would choose services that offer both elasticity and a clear pay-for-service model, which will help manage costs while ensuring the application remains responsive and reliable as it grows.