SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE – 412115

(A CONSTITUENT OF SYMBIOSIS INTERNATIONAL

(DEEMED UNIVERSITY))

2024-25

A MINI-PROJECT ON

APPLICATIONS OF CLOUD COMPUTING USING AWS

BACHELOR OF TECHNOLOGY

IN

Artificial Intelligence and Machine Learning

SUBMITTED BY

ANANYA SACHAN	PRN: 22070126010
APARNA IYER	PRN: 22070126017
HEVARDHAN S.	PRN: 22070126046
RIYA SHUKLA	PRN: 22070126090

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Sr. No.	Abbreviated Word	Expansion
1.	AWS	Amazon Web Services
2.	EC2	Amazon Elastic Compute Cloud
3.	S3	Amazon Simple Storage Service
4.	RDS	Amazon Relational Database Service
5.	IAM	Identity and Access Management
6.	ALB	Application Load Balancer

CHAPTER I: INTRODUCTION

1.1. Project Theme:

1.1.1. Overview: The core theme of this project is to create a robust, high-performance infrastructure for deploying a web application on AWS. The application is hosted using key cloud services that support automatic scaling, load balancing, and optimal resource management, which together ensure the application's responsiveness and reliability.

1.1.2 Technical Approach: The project uses AWS's EC2 (Elastic Compute Cloud) for virtual server instances, Auto Scaling to adjust the number of instances based on traffic demand, and Elastic Load Balancing to distribute incoming requests evenly. These components work together to handle traffic fluctuations and maintain service availability, which is critical for applications with varying user loads.

1.2. Motivation:

1.2.1. Project Rationale: As web applications experience inconsistent demand, scaling infrastructure manually can be challenging, time-consuming, and costly. This project was inspired by the need to develop a scalable solution that automatically adjusts to demand in real-time, ensuring optimal performance without manual intervention.

1.2.2. Learning Objectives: This project is an opportunity to apply cloud computing knowledge to real-world deployment scenarios. Working with AWS provides hands-on experience with cloud services like EC2 and Auto Scaling, which are foundational to cloud-native application development and are essential skills in the modern tech landscape.

1.3. Objectives:

1.3.1. Successful Deployment: Deploy a fully functional web application on AWS, accessible to users at all times, with services configured for scalability and resilience.

1.3.2. Automated Scaling: Use EC2 Auto Scaling to automatically adjust the number of instances to meet demand. This helps manage resources efficiently, reducing costs during low traffic and ensuring availability during high traffic.

1.3.3. Traffic Distribution: Implement Elastic Load Balancing to evenly distribute incoming requests, preventing any single instance from being overwhelmed and enhancing the application's reliability.

1.3.4. Cost and Resource Optimization: By automating the management of instances based on traffic, the project aims to demonstrate an efficient way to manage infrastructure costs while ensuring a stable, responsive application.

1.4. Importance of using Cloud Services for Deployment:

1.4.1. Scalability and Flexibility: Cloud platforms like AWS offer on-demand resources that can scale with traffic. Unlike traditional data centres, cloud infrastructure adapts to needs in real time, preventing over-provisioning and excessive costs.

1.4.2. Cost-Effectiveness: Instead of purchasing and maintaining physical hardware, cloud deployment only incurs costs for the resources used. This pay-as-you-go model minimises expenses, especially for applications with fluctuating demand.

1.4.3.Reliability and Redundancy: Cloud providers like AWS have a robust infrastructure with built-in redundancy. This means services are replicated across multiple data centres, reducing the risk of downtime and providing a stable environment for mission-critical applications.

1.5. Advantages of AWS:

1.5.1. Service Diversity: AWS offers a broad range of services beyond just compute power, including databases, analytics, machine learning, and content delivery networks (CDNs). This flexibility allows for seamless integration of different functionalities.

1.5.2. Global Reach: AWS has data centres worldwide, enabling applications to be deployed closer to end-users, reducing latency and improving user experience.

1.5.3. Built-In Security: AWS provides a variety of security features, such as IAM (Identity and Access Management), security groups, and encryption options. These features help ensure that applications are protected from unauthorised access and data breaches.

1.5.4. Reliability and Service-Level Agreements: AWS provides high availability with its robust infrastructure and offers SLAs that guarantee uptime, making it suitable for critical applications

1.6. Feasibility:

1.6.1.Technical Feasibility: The project is achievable within the scope of AWS's services. AWS documentation and a range of available online resources offer substantial guidance on implementing EC2, Auto Scaling, and Load Balancing effectively.

1.6.2.Cost Feasibility: With AWS's pay-as-you-go pricing model, costs can be controlled by scaling down resources during low-traffic periods. The project plan includes monitoring expenses to ensure that cloud usage stays within the allocated budget.

1.6.3.Time Feasibility: Given AWS's user-friendly console and available SDKs, deploying and configuring the necessary services is feasible within a short development cycle. This makes the project suitable for the given timeframe, with room for iterative testing and improvements.

1.7. Constraints:

1.7.1. Financial Constraints: Although AWS offers cost-effective solutions, costs can escalate if Auto Scaling frequently provisions new instances during traffic peaks. It's important to monitor usage closely to avoid unexpected charges.

1.7.2. Technical Limitations: Certain AWS services and instance types may not be available in all regions, which could limit deployment options. Additionally, configuring Auto Scaling and Load Balancing requires technical know-how to prevent misconfigurations that could affect performance or cost.

1.7.3. Security and Compliance Constraints: Ensuring security at every level—such as by configuring security groups, IAM roles, and monitoring access logs—adds complexity to the project. Compliance with organisational or regulatory requirements might also impose additional configuration and monitoring needs.

1.7.4. Performance Constraints: Balancing load across instances may introduce latency if services are not well-configured or if traffic spikes are unpredictable. Elastic Load Balancing mitigates this to some extent but requires careful tuning for optimal performance.

CHAPTER II: ARCHITECTURE

The **Career Ally chatbot** is an AI-driven solution designed to provide personalised career guidance, offering tailored recommendations based on individual skills and career goals. Built on AWS, the architecture ensures that the chatbot can efficiently handle large volumes of user queries while maintaining high performance, availability, and security. By leveraging Amazon EC2 for backend processing, Elastic Load Balancer (ELB) for traffic distribution, and Auto Scaling for dynamic resource management, Career Ally adapts to user demand in real time. The use of Amazon CloudWatch enables robust monitoring, ensuring proactive management of performance and potential issues, while AWS IAM enforces strict access control for secure operation. This architecture delivers a scalable, cost-effective, and secure platform, making Career Ally a reliable tool for users seeking actionable career insights.

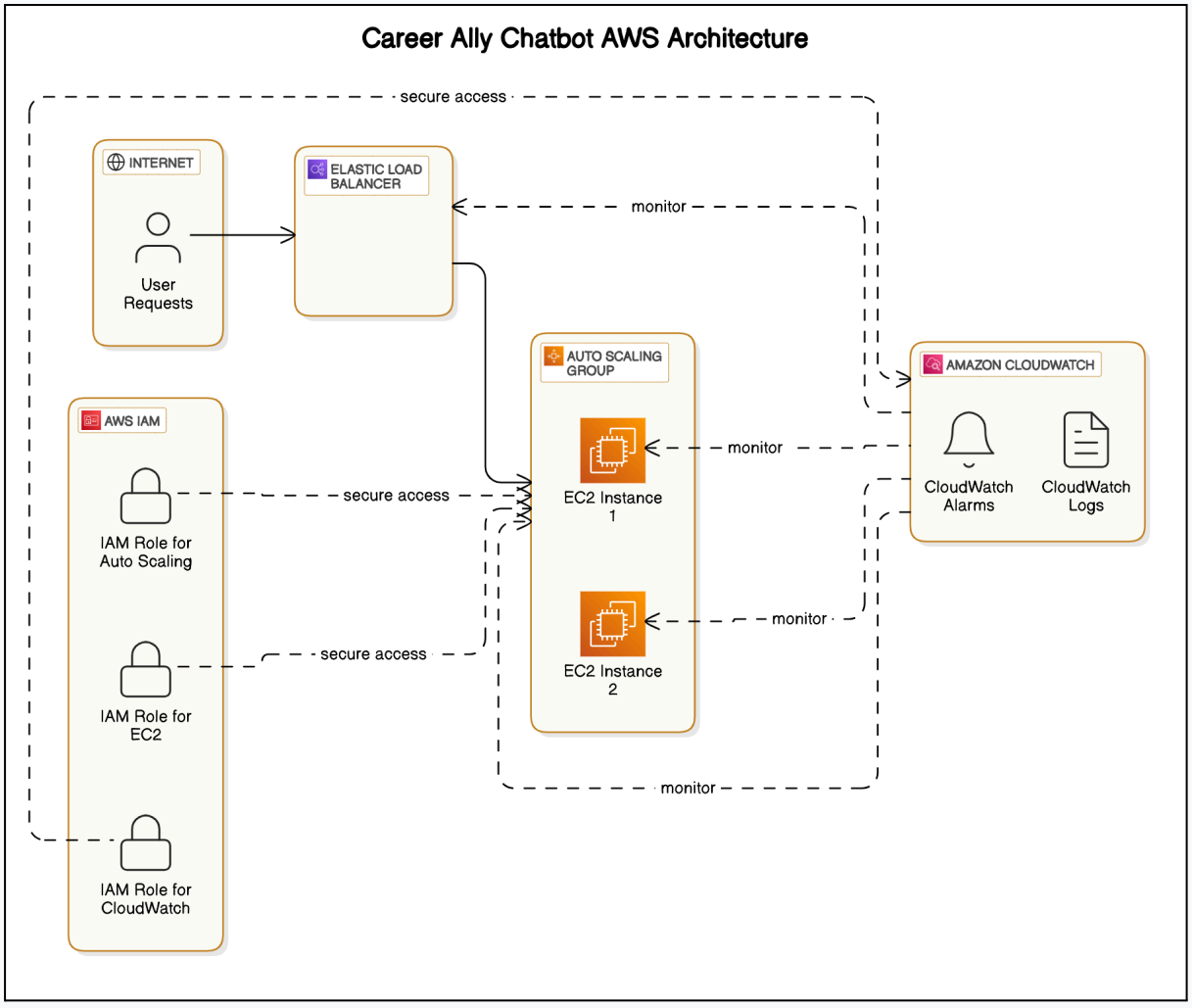

Fig. 2.1.: Career Ally Chatbot AWS Architecture

2.1. Architecture Overview:

The Career Ally chatbot is designed to provide personalised career guidance by analysing user inputs, assessing skills, and delivering AI-driven recommendations. The architecture leverages key AWS services to ensure scalability, high availability, performance, and security.

2.2. Components of the Architecture:

2.2.1. Amazon EC2

- Hosts the chatbot's backend, handling AI-driven logic, user interactions, and API services.
- Provides scalable compute resources for processing user queries in real-time.

2.2.2. Elastic Load Balancer (ELB)

- Distributes incoming user traffic across multiple EC2 instances.
- Ensures high availability and fault tolerance by routing traffic to healthy instances.

2.2.3. Auto Scaling Group

- Dynamically adjusts the number of EC2 instances based on demand.
- Scales out during high traffic and scales in during low traffic to optimize cost.
- Monitors performance using CloudWatch metrics to trigger scaling actions.

2.2.4. Amazon CloudWatch

- Monitors the health and performance of EC2 instances, ELB, and Auto Scaling activities.
- Provides real-time metrics, logs, and alarms for proactive monitoring and troubleshooting.
- Enables performance dashboards and alerts for system health.

2.2.5. AWS IAM (Identity and Access Management)

- Manages secure access between AWS services using roles and policies.
- Ensures least-privilege access to EC2, Auto Scaling, and CloudWatch.
- Protects sensitive data and controls access to system configurations

2.3. Data Flow in the AWS Architecture:

1. **User Interaction**:
    - Users interact with the Career Ally chatbot through a web or mobile interface, sending requests over the internet.
2. **Traffic Distribution**:
    - The **Elastic Load Balancer (ELB)** receives incoming user requests and distributes them across available EC2 instances in the Auto Scaling Group.
3. **Backend Processing**:
    - **Amazon EC2** instances handle the processing of user requests, including AI/ML algorithms, data retrieval, and response generation.
4. **Auto Scaling**:

    - The **Auto Scaling Group** monitors the load on EC2 instances using **CloudWatch** metrics and adjusts the number of running instances based on demand.
5. **Monitoring and Logging**:
    - **Amazon CloudWatch** collects performance metrics, logs, and alarms to ensure the smooth operation of the system.
    - Any anomalies or threshold breaches trigger CloudWatch Alarms for proactive issue management.
6. **Security Management**:
    - **AWS IAM** secures interactions between AWS resources and ensures only authorised access to the system components.

2.4. Key Benefits of the Chosen Architecture:

- **Scalability**: Automatically scales with user demand, ensuring a responsive user experience even during peak traffic.
- **High Availability**: The combination of ELB and Auto Scaling ensures that the chatbot remains available and reliable.
- **Cost Efficiency**: Auto Scaling optimises resource usage, helping to minimise costs during low-traffic periods.
- **Robust Monitoring**: CloudWatch provides real-time visibility into system performance, enabling proactive troubleshooting.
- **Secure Access Control**: IAM enforces strict access policies, safeguarding the application and user data.

This architecture is designed to deliver a highly scalable, secure, and efficient solution for deploying the Career Ally chatbot on AWS.

CHAPTER III: AWS SERVICES USED

In this section, we outline our approach to deploying **Career Ally**—an NLP-driven career guidance chatbot—on Amazon Web Services (AWS) using the services provided by AWS.

Career Ally leverages natural language processing to provide personalised career guidance, requiring reliable, scalable, and responsive cloud infrastructure to ensure a smooth user experience.

To deploy this application efficiently, we rely on several essential components within Amazon EC2, including **EC2 Instances**, **Auto Scaling Groups**, and **Load Balancers**.

These AWS services enable us to meet the computational demands of Career Ally, while optimising for cost-effectiveness and resilience.

1. **EC2 Instances**:
    - EC2 instances are virtual servers that provide the foundational compute power necessary for running Career Ally's NLP models and processing user queries in real-time. We can select instance types optimised for high CPU or memory usage, based on the application's requirements, ensuring that Career Ally operates efficiently with sufficient resources.
    - **Role in Application Deployment**: EC2 instances serve as the primary processing environment for Career Ally, handling everything from NLP computations to API requests. With Amazon EC2, we can adjust instance specifications or add new instances as needed, providing Career Ally with the flexibility to adapt to varying usage patterns and ensuring that resources are utilised effectively.

2. **Auto Scaling Groups**:
    - Auto Scaling Groups, part of the Amazon EC2 Auto Scaling service, ensure that Career Ally remains responsive during peak traffic periods by automatically scaling the number of EC2 instances up or down based on real-time demand. This capability is essential for handling fluctuations in user activity, especially during high-usage periods.

- **Role in Application Deployment**: For Career Ally, Auto Scaling Groups help maintain application availability and performance without over-provisioning resources. During high-traffic times, such as career fair seasons or exam periods, Auto Scaling Groups can automatically increase the number of instances. During low-traffic times, they scale back, controlling costs while maintaining consistent performance. This elasticity allows Career Ally to provide a seamless experience to users regardless of demand.

3. **Load Balancers**:
    - AWS Elastic Load Balancing (ELB) distributes incoming traffic evenly across the EC2 instances that power Career Ally, ensuring that no single instance becomes overloaded. ELB offers several types of load balancers, such as the Application Load Balancer (ALB) for HTTP and HTTPS traffic, which is suitable for Career Ally's web and mobile interface.
    - **Role in Application Deployment**: By routing user requests across multiple instances, Load Balancers improve both the reliability and scalability of Career Ally. They prevent any one instance from becoming a single point of failure, helping to maintain uptime and responsiveness even under high load. This traffic management ensures that users always have fast and reliable access to Career Ally's services.

CHAPTER IV: IMPLEMENTATION

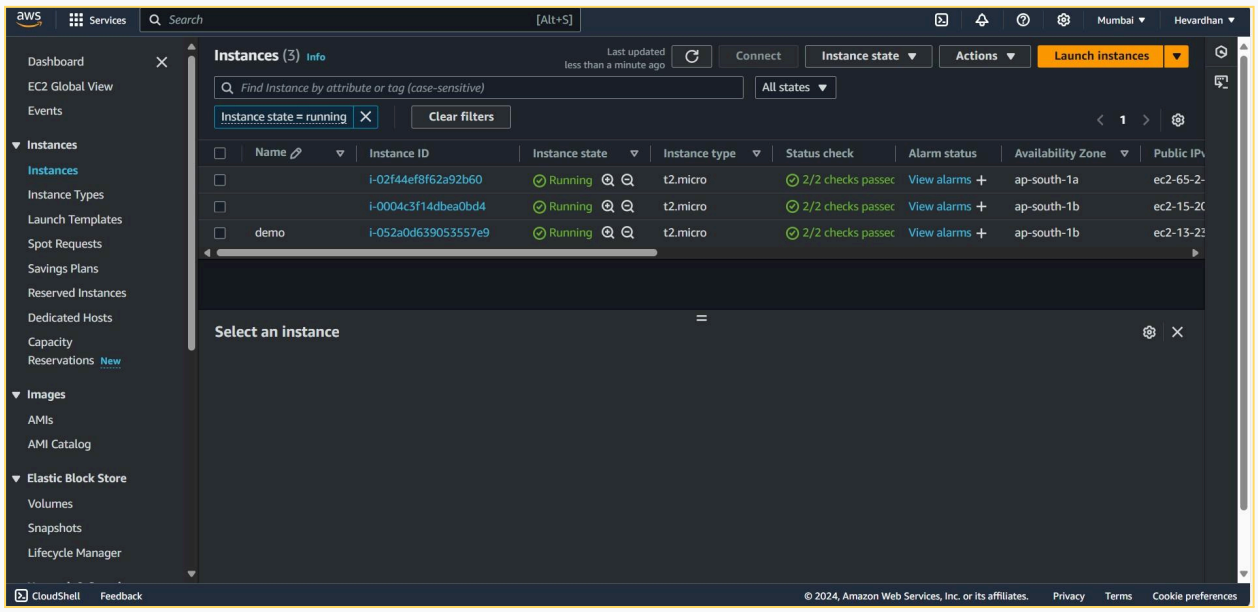

Fig. 4.1.: EC2 Instances created for Deployment

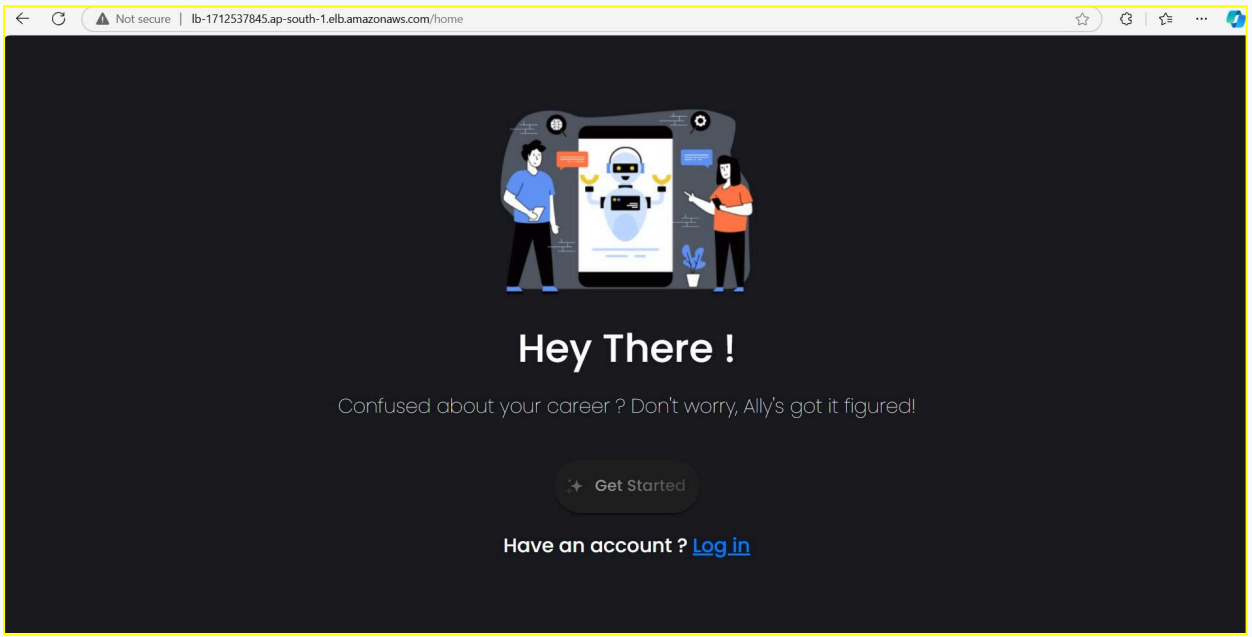

Fig. 4.2.: Career Ally Deployment Page using AWS

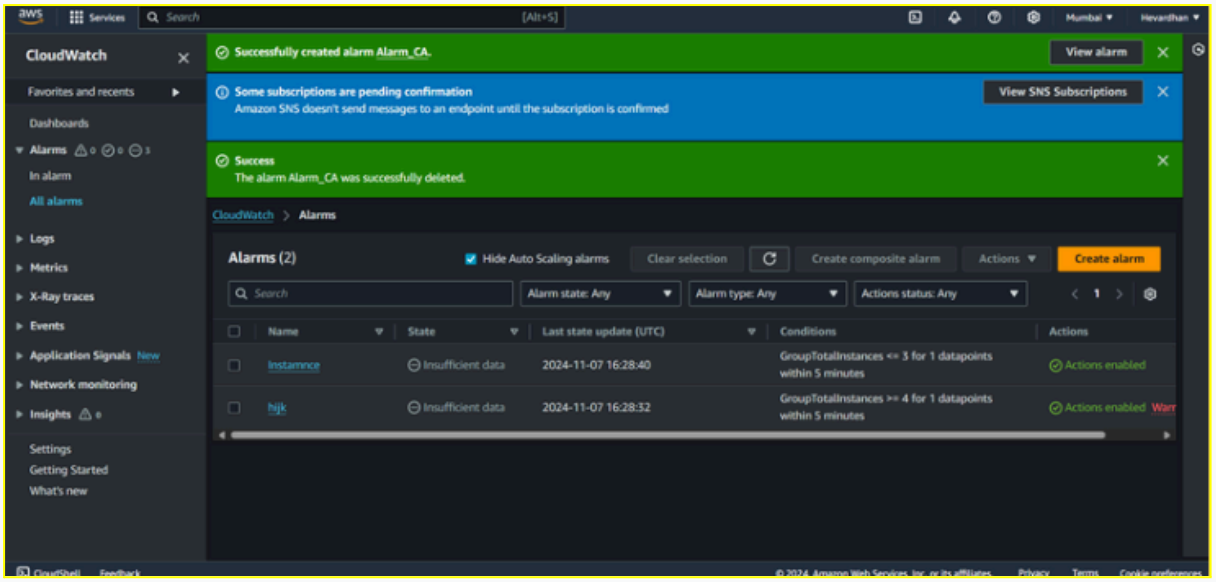

Fig. 4. 3.: CloudWatch Configuration

Steps for Implementation:

1. Set up the Launch Template:

● Create an EC2 launch template in AWS that specifies the configuration for instances in the autoscaling group. Include the AMI(Ubuntu), instance type(micro), security groups(HTTP, SSH), and key pair.

● Add the Docker installation and static website setup in the User Data field to automate the setup on each instance launch:

Code:

```
#!/bin/bash

apt update -y

apt install -y npm

apt install -y docker.io

git clone https://github.com/hevardhan/CareerAlly.git

cd CareerAlly

docker build -t payas:v1 .

docker run -d -p 80:80 payas:v1
```

2. Create an Auto Scaling Group (ASG):

● Define the ASG with minimum(1), maximum(5), and desired(2) instance counts, link it to the launch template.

● Enable health checks to monitor instance health regularly and Cloud Watch is enabled to monitor the health and performance of the EC2 instances and the load balancer.

3. Set up the Load Balancer (LB):

● Create an Application Load Balancer (ALB) in AWS, which distributes traffic across the instances.

● Configure Availability Zones and Security Groups with two inbound rules to enable HTTP and SSH access.

● Attach the target group with health checks to ensure instances are running properly and route traffic only to healthy instances.

4. Define Target Group:

● Configure Target type(Instances), Protocol:Port(HTTP:80) and anything else leave with the default option enabled.

● Set health check parameters to monitor instance health regularly.

5. CloudWatch:

● Set an Alarm to notify when the number of instances running exceeds a particular value.

6. Deploy and Test:

● Launch the ASG to automatically create instances, and verify the static website is accessible through the load balancer’s DNS.

● Test auto scaling by generating load and observing if new instances are added or removed.

CHAPTER V: CHALLENGES

During the deployment of Career Ally on AWS, we encountered several challenges that required careful attention and strategic solutions:

5.1. EC2 Management:

- Selecting the right instance type was critical to balance performance and costs.
- Ensuring instance security and regular updates required ongoing maintenance.
- Implementing automated recovery was necessary to handle unexpected instance failures.

5.2. Auto Scaling:

- Defining optimal scaling policies was challenging; overly aggressive scaling wasted resources, while conservative settings led to performance issues.
- The scaling response was occasionally delayed, affecting performance during traffic spikes.

5.3. Load Balancer:

- Configuring ELB health checks and optimising traffic distribution required fine-tuning to prevent latency and ensure high availability.

5.4. CloudWatch Monitoring:

- Setting up accurate metrics and alerts was complex, needing adjustments to avoid false positives and missed alerts.
- Managing log storage costs became essential due to rapid data growth.

5.5. IAM Security:

- Balancing security and functionality through precise IAM policies was challenging, especially for secure service communication.

5.6. Cost Management:

- We encountered unexpected costs from over-provisioning, necessitating continuous resource optimization.

5.7. Networking:

- Configuring security groups and VPC settings for seamless connectivity required careful planning.
- Reducing latency was essential to maintain a responsive user experience during peak traffic.

Addressing these challenges involved planning, automation, and leveraging AWS best practices to ensure a reliable, cost-effective, and secure deployment of Career Ally.

CHAPTER VI: CONCLUSION

- **Project Summary**: This project successfully demonstrates the deployment of a web application on AWS, utilising EC2 instances, Auto Scaling, and Elastic Load Balancing to build a robust, scalable, and high-performing infrastructure. By setting up a flexible environment that adjusts resources based on traffic demands, we ensure that the application can handle variable user loads while maintaining consistent performance and availability.

- **Security and Access Control**: Security is a crucial aspect of any cloud deployment, and in this project, AWS Identity and Access Management (IAM) was used to control access to cloud resources. IAM allowed us to define and enforce precise permissions, ensuring that only authorised users and services could interact with the application's infrastructure. This access control helps maintain the integrity and security of the application, safeguarding it from unauthorised actions or potential breaches.

- **Scalability and Cost-Efficiency**: The integration of Auto Scaling and Elastic Load Balancing enabled our infrastructure to respond automatically to changing traffic patterns, ensuring both scalability and cost-efficiency. By scaling resources up during peak times and down during low-traffic periods, we optimised resource usage and minimised unnecessary expenses.

- **Reliability and Future Potential**: Through this AWS-based deployment, we created an environment that is highly reliable, thanks to the redundancy and failover mechanisms built into AWS services. The use of IAM for access control, along with monitoring and logging tools (e.g., CloudWatch), positions this infrastructure as a strong foundation for future enhancements. Adding additional AWS services, such as

S3 for storage or RDS for managed databases, could further enhance the application's capabilities, providing a scalable path for future development.

Overall, this project highlights the advantages of using AWS for deploying scalable web applications. The cloud-native approach not only simplifies infrastructure management but also enhances application security, availability, and performance. By leveraging AWS's extensive toolset, we were able to achieve an efficient and secure deployment, laying the groundwork for continuous improvement and scalability.