IMPLEMENTING AND ARCHITECTING CLOUD WEB INFRASTRUCTURES

A Project Report

Submitted by

P.LAXMI NARAYANA(20331A1293)

P.S.V.SOUGANDHIKA(20331A1297)

MOHAMMAD ZAKIR AHMED(20331A1278)

In partial fulfilment for the award of the degree

Of

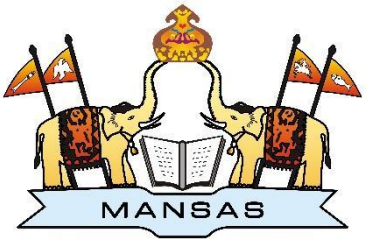
BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Under the esteemed Guidance of

Dr.V. NAGESH
Professor



DEPARTMENT OF INFORMATION TECHNOLOGY
MVGR COLLEGE OF ENGINEERING (A),
VIZIANAGRAM.

April 2024

DECLARATION

We hereby declare that the project entitled "**IMPLEMENTING AND ARCHITECTING CLOUD WEB INFRASTRUCTURES**" submitted for the partial fulfilment of B.Tech Degree is our original work and the project has not formed the basis for the award of any degree or any other similar titles.

Place: Vizianagaram

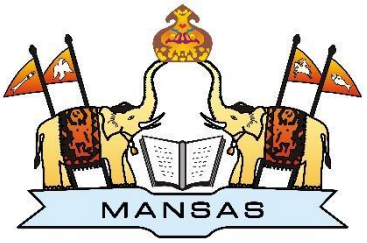
Date:

P.LAXMI NARAYANA(20331A1293)

P.S.V.SOUGANDHIKA(20331A1297)

MOHAMMAD ZAKIR AHMED(20331A1278)

CERTIFICATE



This is to certify that the project entitled "**IMPLEMENTING AND ARCHITECTING CLOUD WEB INFRASTRUCTURES**" is the bonafide work carried out by P. LAXMI NARAYANA (20331A1293), P.S.V.SOUGANDHIKA (20331A1297), MOHAMMAD ZAKIR AHMED(20331A1278), students of B.Tech Ⅷ semester, Information Technology, MVGR College of Engineering (Autonomous), Vizianagaram, during the year 2020-2024, in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology and that the project has not formed the basis for the award previously of any degree or any other similar title.

Signature of Project Guide

Dr.V. NAGESH,
Professor,
Department of IT,
MVGR College of Engineering (A).

Signature of Head of the department

Dr. P Srinivasa Rao,M.Tech., Ph.D.,
Professor & Head,
Department of IT,
MVGR College of Engineering (A).

IMPLEMENTING AND ARCHITECTING CLOUD WEB INFRASTRUCTURES

Team Name

Cloud Titans

Team

Department of Information Technology

Name: P.LAXMI NARAYANA

Email:lakshminarayana4848@gmail.com

Mobile: 6302138515

Name:

P.S.V.SOUGANDHIKA

Email:sougandhikapenmetsa@gmail.com

Mobile: 8309468479

Name: MOHAMMAD ZAKIR AHMED

Email:zakir4813@gmail.com

Mobile: 6302269690

Project Guide

Name: DR. V. NAGESH

Designation: Professor, IT

Email:**itsnageshv@gmail.com**

Objective

The main objective of the project is to design and implement 1-tier,2-tier and 3-tier architectures, along with load balancing to ensure application scalability through orchestration techniques.

Select the domain (s) where your solution can be implemented

Scalable Web Application Development

Describe how your solution is going to meet the program outcomes.

1.The project explores multi-tier architectures, leveraging AWS resources like EC2 instances, EFS and RDS to develop scalable web applications.

2.It begins with understanding the limitations of 1-tier and 2-tier and then progressing towards 3-tier architecture with load balancing and auto scaling for handling traffic spikes.

3.PHP is utilized for web application development, server-side processing and API development while EFS and RDS for storage and database.

4.High availability is ensures through redundancy in EC2 instances making this project a valuable guide for crafting robust, efficient and highly available web applications in multi-tier cloud environments.

Describe the engineering solution

Architecting scalable web apps on AWS for high availability

End Users / Stakeholders of the Solution

It provides developers, system adiminstrators and businesses with scalable and reliable web application infrastructure on AWS.

ACKNOWLEDGMENT

We wish to express our sincerest and most profound gratitude to our guide Dr.V.NAGESH, Professor, Department of Information Technology. We all thankful for his cooperation and guidance. He had given all the valuable instructions and support, even at odd hours and gave his precious time.

We thank our Head of the Department, Dr. P Srinivasa Rao, Professor, Department of Information Technology for providing us all the necessary infrastructure and support whenever necessary.

We also thank Dr. R Ramesh, Principal of MVGR college of Engineering, for extending his utmost support and cooperation in providing all provisions for successful completion of the project.

We sincerely thank the staff of Department of Information Technology for helping us out in all the ways we needed and they could have.

P.LAXMI NARAYANA(20331A1293)

P.S.V.SOUGANDHIKA(20331A1297)

MOHAMMAD ZAKIR AHMED(20331A1278)

ABSTRACT

The project explores multi-tier architectures, encompassing 1-tier, 2-tier, and 3-tier configurations, utilizing AWS resources including EC2 instances, Elastic File System (EFS), Relational Database Service (RDS), and PHP. We commence with 1-tier and 2-tier architectures to grasp limitations, then advance to 3-tier architecture, integrating load balancing and auto scaling for efficient handling of traffic spikes. EFS and RDS offer scalable storage and database services, while PHP facilitates web application development, server-side processing, and API development. High availability is ensured through redundancy in EC2 instances, RDS database replication, and EFS distributed file storage. This project serves as a valuable guide for crafting robust, efficient, and highly available web applications in multi-tier cloud environments.

Moreover, in our journey through different architectures, we pay special attention to security. As we transition from 1-tier to 2-tier and finally to 3-tier configurations, we ensure that each step maintains strong security measures. This includes controlling access to resources like EC2 instances, EFS, and RDS databases, setting up secure networks with VPC. By integrating security at every level, we not only build robust and scalable systems but also ensure that our web applications are safe from potential threats, no matter how much traffic they handle.

PROGRAM OUTCOMES

Engineering Graduates will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering

practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ProjectTitle	P.O Mapping
Implementing and Architecting Cloud Web Infrastructures	1,2,4,5,8,9,10,11,12

Table of Contents

S.no	Name	Page no.
1.	Cover page & Title page	1
2.	Declaration	2
3.	Certificate	3
4.	Acknowledgement	5
5.	Abstract	6
6.	Program Outcomes	7-8

CHAPTERS

1. INTRODUCTION

1.1	Project overview	12
1.2	Existing System	13
1.3	Proposed System	14

2. LITERATURE SURVEY

2.1	Multi-tier architectures	16
2.2	Load Balancing	16
2.3	Auto Scaling	17
2.4	Security Groups	17

3. SYSTEM STUDY AND ANALYSIS

3.1	Functional Requirements	19-21
3.2	Non-Functional Requirements	22
3.3	System Requirements	23
3.3.1	Hardware Requirements	23
3.3.2	Software Requirements	23

4 SYSTEM DESIGN

4.1 Activity Diagram 25

4.2 Network flow Diagram 26

4.3 Class diagram 27

5 THEORETICAL BACKGROUND

5.1 Architectures 29

5.2 PHP 29-30

5.3 Overall theoretical background 30

6 HARDWARE & SOFTWARE REQUIREMENTS

6.1 Software Requirements 32

6.2 Hardware Requirements 32-33

7 CODE MODULE

7.1 Code Module 35

7.2 Code Snippets 36-39

8 TESTING & DEPLOYMENT

8.1 Output Screen shots 41-45

9 CONCLUSION

9.1 Conclusion 47

9.2 Future Scope 48

10 REFERENCES 49-51

1. INTRODUCTION

1.1 PROJECT OVERVIEW

In the dynamic landscape of cloud-based architecture, our project delves deep into multi-tier configurations, spanning 1-tier, 2-tier, and 3-tier systems. We leverage cloud resources such as EC2 instances, Elastic File System (EFS), Relational Database Service (RDS), and PHP. Our journey begins by scrutinizing the limitations of 1-tier and 2-tier architectures, leading us to explore the intricacies of 3-tier structures.

Throughout this exploration, we employ load balancing and auto-scaling techniques to adeptly manage traffic spikes. Scalable storage and database services are achieved through EFS and RDS, while PHP powers the deployment of the Apache web server. At the core of our project lies the commitment to unwavering high availability, achieved through meticulous redundancy across EC2 instances, RDS database replication, and EFS distributed file storage. This project stands as an invaluable reference, empowering individuals to craft resilient and high-performance web applications in complex, multi-tier cloud environments. Welcome to a world where cloud excellence converges with the art of application architecture.

1.2 EXISTING SYSTEM

The existing system comprises a combination of 1-tier, 2-tier, and 3-tier architectures, each deployed without the incorporation of load balancers and auto scaling mechanisms. In the 1-tier architecture, all components, including the presentation, application logic, and data storage layers, are housed within a single server instance. Similarly, in the 2-tier configuration, the application logic and data storage layers are separated, typically with the application deployed on one server and the database on another. Transitioning into the 3-tier architecture, an additional layer is introduced for separating the presentation layer from the application logic and data storage layers.

Despite the inclusion of multiple tiers, the absence of load balancers and auto scaling poses challenges in efficiently distributing incoming traffic and dynamically adjusting resources to meet demand fluctuations. It leaves the system vulnerable to downtime in the event of server failures. Thus, while the existing architecture provides a basic framework for application deployment, its limitations necessitate the implementation of load balancing and auto scaling to enhance scalability, reliability, and performance.

1.3 PROPOSED SYSTEM

In our proposed system, we begin by comprehensively understanding the limitations of traditional 1-tier, 2-tier, and 3-tier architectures. Acknowledging the scalability and reliability challenges they present, we leverage Amazon Web Services (AWS) to implement dynamic load balancing and autoscaling mechanisms, effectively overcoming these constraints. Concurrently, we integrate PHP into our architecture, utilizing its flexibility and robustness to develop scalable and responsive web applications. By combining AWS's cloud infrastructure with PHP's versatility, we create a resilient and agile system capable of adapting to fluctuating workloads seamlessly.

Moreover, our disaster recovery mechanisms, facilitated by AWS's multi-region capabilities, ensure uninterrupted service by dynamically redirecting traffic in case of system failures. Through proactive health checks, we continuously monitor the system's condition, enabling swift responses to maintain high availability and reliability. This holistic approach enables us to build a resilient infrastructure that meets the demands of modern applications while mitigating potential risks effectively.

2.LITERATURE SURVEY

2.1 Multi-tier architectures

In their survey on "Quality of Service in Multi-Tier Web Applications," **Mohamed Ghetas, Chan Huah Yong, and Putra Sumari** address challenges in managing performance amidst dynamic workloads. They emphasize the importance of QoS assurance and propose self-adaptive resource provisioning strategies. The authors evaluate existing rule- and model-based approaches for dynamic resource management, identifying benefits and drawbacks. They suggest new research directions to enhance resource management efficiency, contributing insights to improving QoS in multi-tier web environments.

2.2 Load Balancing

Garima Rastogi, a Research Scholar at DIT University, and Dr. Rama Sushil, Head of the Department of Information Technology at the same institution, present a comprehensive analysis of load balancing techniques. Load balancing is crucial for distributing processing loads evenly and dynamically across all nodes within a system. Effective load balancing optimizes resource utilization, minimizing consumption and enhancing user satisfaction. Additionally, it plays a pivotal role in conserving energy, contributing to a cleaner and greener environment. The efficient distribution of workload not only optimizes resource usage but also reduces energy consumption, aligning with sustainability objectives. Thus, load balancing emerges as an essential component in computing environments, addressing both performance optimization and environmental concerns.

2.3 Auto-Scaling

In their research paper titled "Research on Auto-Scaling of Web Applications in Cloud: Survey, Trends and Future Directions," **Parminder Singh, Pooja Gupta, Kiran Jyoti, and Anand Nayya** explore the challenges and techniques associated with auto-scaling in cloud environments. Auto-scaling dynamically adjusts resources for elastic applications based on incoming workloads, aiming to meet service level agreements (SLAs), ensure quality of service (QoS), and minimize scaling costs. The authors outline the components of the MAPE loop (Monitoring, Analysis, Planning, Execution) in auto-scaling and discuss the challenges such as under-provisioning, over-provisioning, and oscillation. They delve into the various techniques and approaches used in auto-scaling, including reactive and proactive strategies, threshold-based rules, machine learning, and control theory. The paper highlights the importance of efficient resource allocation and management in ensuring optimal performance and reliability of web applications in cloud environments.

2.4 Security groups

In their work on "Computing Security: Threats and Mitigation Strategies," **Bader Alouffi, Muhammad Hasnain, Abdullah Alharbi, Wael Alosaimi, Hashem Alyami, and Muhammad Ayaz** underscore the importance of security in promoting the acceptance of cloud computing services. They highlight literature on security solutions, propose new perspectives on cloud security, and address challenges in cloud computing. Their study introduces solutions like the Autonomous Cloud Intrusion Response System (ACIRS) to mitigate risks. They emphasize investing in cloud computing-associated device security and explore research efforts in evaluating cloud computing security.

3. SYSTEM STUDY AND ANALYSIS

3.1 Functional Requirements

This project can be divided into 4 modules:

- **Registration Module**

  ➔ **Input:** This registration page accepts user input with the following:
    - Username
    - Email address
    - Password

  ➔ **Processing:**
    - The user enters his data.
    - Upon clicking register the SQL query is instructed to store the data in the database.

  ➔ **Output:** If the registration is successful, a confirmation message is displayed with the link to login page, otherwise an error message is displayed prompting the user to fill the missing fields and retry the registration.

➔ **Login Module**

  ➔ **Input:** The login page expects the user to provide the following details to login:
    - Username
    - Password

  ➔ **Processing:**
    - The user enters his data.
    - Upon clicking login, the website checks whether the details provided by the user matches with what's stored in the database.

  ➔ **Output:**
    - If the login is successful, the user is redirected to the next page. If the login fails due to incorrect username or password, an error message prompts the user to log in again.

- Additionally another link is provided to new users prompting if the user is new and do not have an account asking them to register.

➔ **User login process in multi-tier architectures**

➔ **Input:**User provided login details which includes username and password

➔ **Processing:**

- PHP script verifies the user's credentials against stored data.
- Dynamically distributing incoming traffic across multiple instances.
- Scaling the resources up or down depending on the requirement using AWS Auto Scaling.

➔ **Output:**

- If the credentials mentioned by the user are successful then the images are retrieved from the database.
- Accessing different pages and performing actions are handled by the architecture layers.

➔ **Load Balancing and Auto Scaling Module**

➔ **Input:**System resource utilization includes the following;

- CPU usage
- Incoming traffic

➔ **Processing:**

- Monitoring system load using the AWS services ELB (Elastic Load Balancing).
- Dynamically distributing incoming traffic across multiple instances.
- Scaling the resources up or down depending on the
- requirement using AWS Auto Scaling.

➔ **Output:**

- Distribution of incoming traffic to maintain system stability.
- Automatic scaling of resources to handle work load efficiently.

➔ **Disaster Recovery Module**

➔ **Input:**System utilization includes the following;

- System health status
- Server availability

➔ **Processing:**

- Utilization of AWS multi-region capabilities to replicate data and services across geographically distributed locations.
- Monitoring the system health continuously.
- Automatic rerouting of traffic depending on the system health status.

➔ **Output:**

- Detection of timely system failures to make ensure high availability.
- Uninterrupted services by redirecting the traffic to alternate regions without data loss.

3.2 Non-Functional Requirements

➔ **High Availability:**

- Ensuring that the system remains accessible even in the situation of failures or other disruptions automatic enrouting to available AWS multiple zones.

➔ **High Scalability:**

- AWS dynamic infrastructure allows the system to effortlessly accommodate to increased traffic ensuring smooth performance.

➔ **Load Balancing:**

- Load balancing helps in distributing the incoming requests across multiple instances in the application leading to optimal resource utilization.

➔ **High Security:**

- With the help of security groups, we can control the traffic of the system ensuring that only authorized connections are allowed providing the security.

➔ **Disaster Recovery:**

- Utilizing AWS multi-region capabilities, we redirect traffic to healthy regions during system failures, ensuring uninterrupted service and minimal downtime to users.

➔ **Maintenance:**

- It is easy to use, maintain and add new features to it.

3.3 SYSTEM SPECIFICATIONS

3.3.1 HARDWARE SPECIFICATIONS

Processor: i5

Ram : 8GB

Hard Disk : 128GB

3.3.2 SOFTWARE SPECIFICATION

Operating System : UBUNTU(Version - 20.04)

Code : PHP,HTML

Software installed : Maven, Java, Ubunty(O.S)

Open SSH

4. SYSTEM DESIGN

4.1 Activity diagram

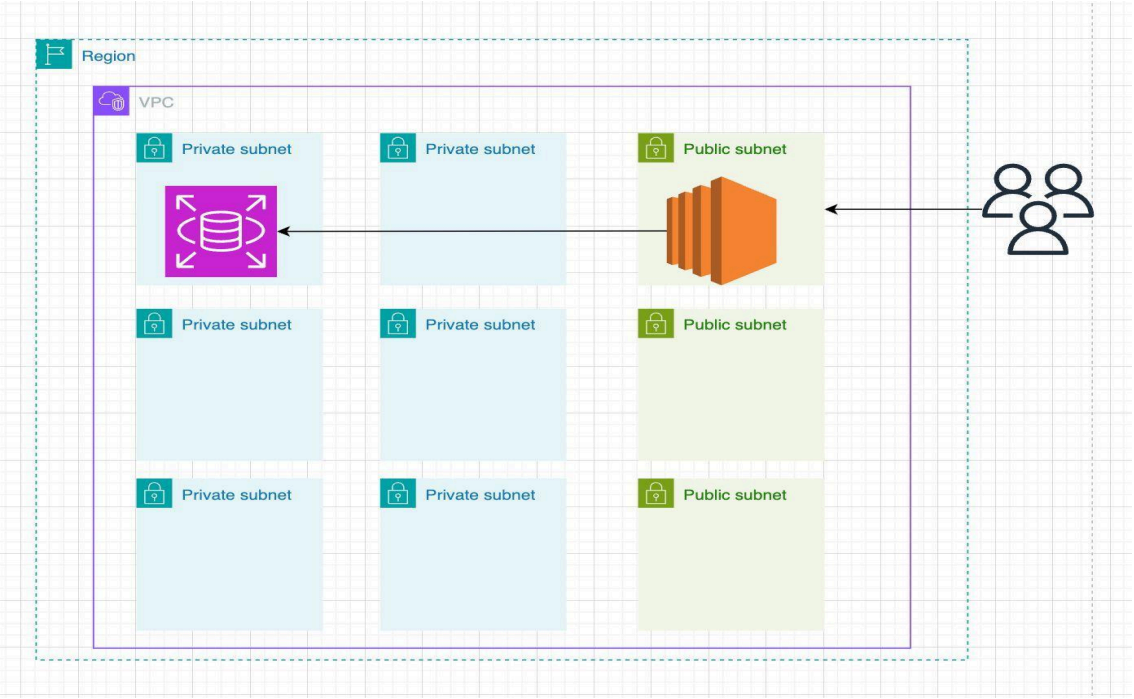


FIG 1

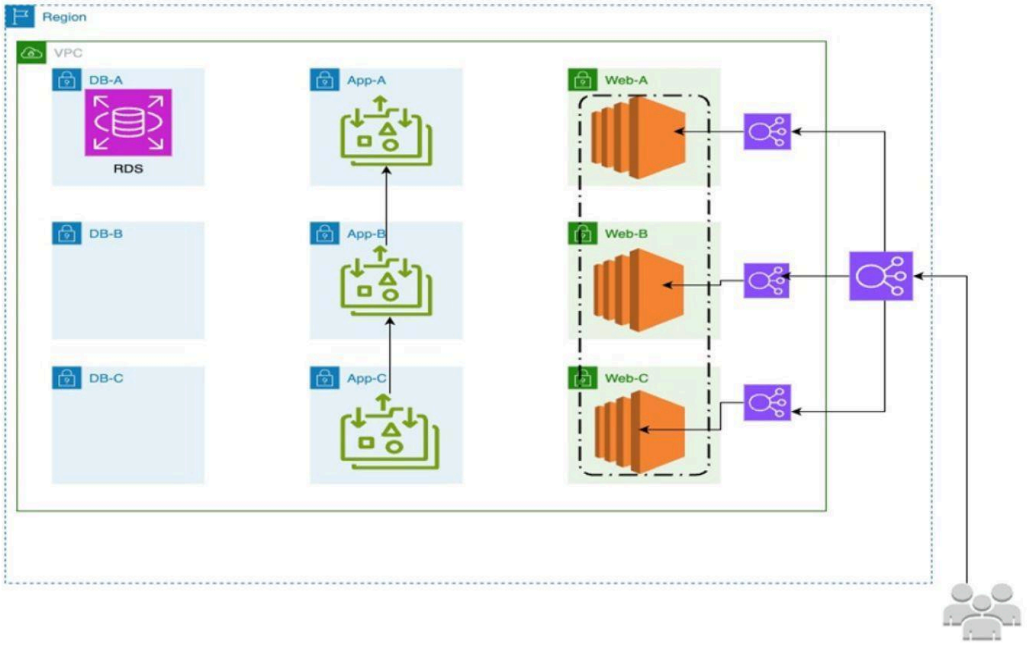


FIG 2

4.2 Network Flow Diagram

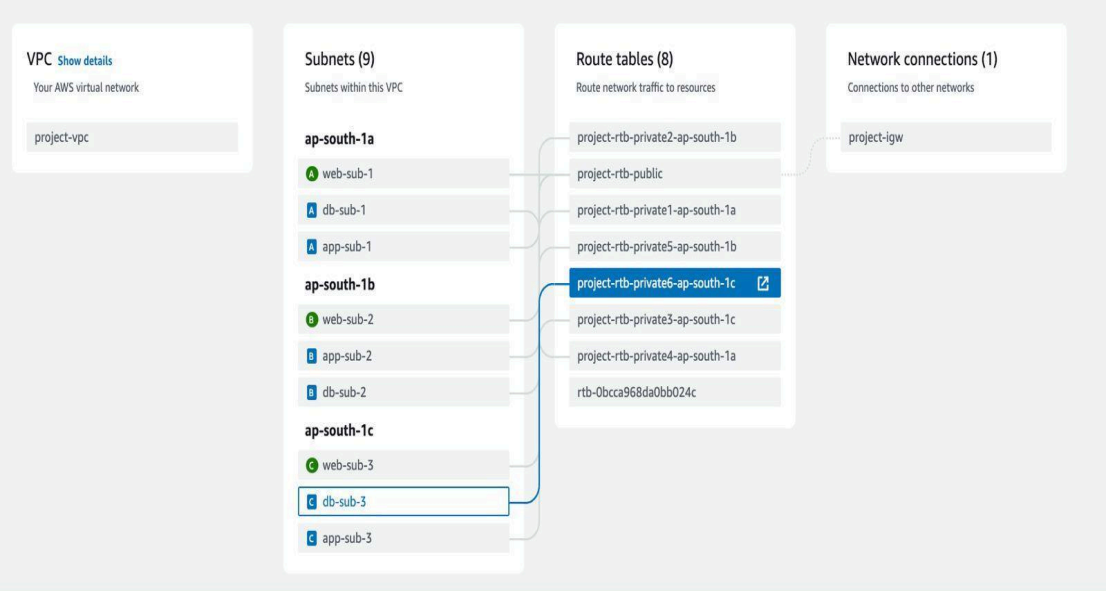
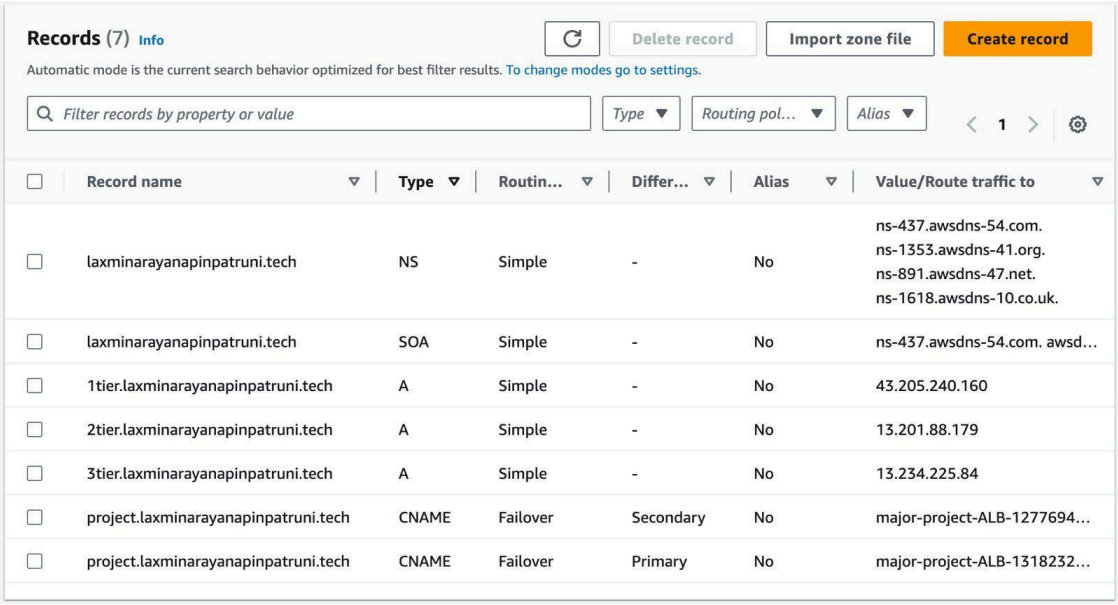


FIG 3



Record name	Type	Routin...	Differ...	Alias	Value/Route traffic to
laxminarayanapinpatruni.tech	NS	Simple	-	No	ns-437.awsdns-54.com. ns-1353.awsdns-41.org. ns-891.awsdns-47.net. ns-1618.awsdns-10.co.uk.
laxminarayanapinpatruni.tech	SOA	Simple	-	No	ns-437.awsdns-54.com. awsd...
1tier.laxminarayanapinpatruni.tech	A	Simple	-	No	43.205.240.160
2tier.laxminarayanapinpatruni.tech	A	Simple	-	No	13.201.88.179
3tier.laxminarayanapinpatruni.tech	A	Simple	-	No	13.234.225.84
project.laxminarayanapinpatruni.tech	CNAME	Failover	Secondary	No	major-project-ALB-1277694...
project.laxminarayanapinpatruni.tech	CNAME	Failover	Primary	No	major-project-ALB-1318232...

FIG 4

4.3 Work flow diagram

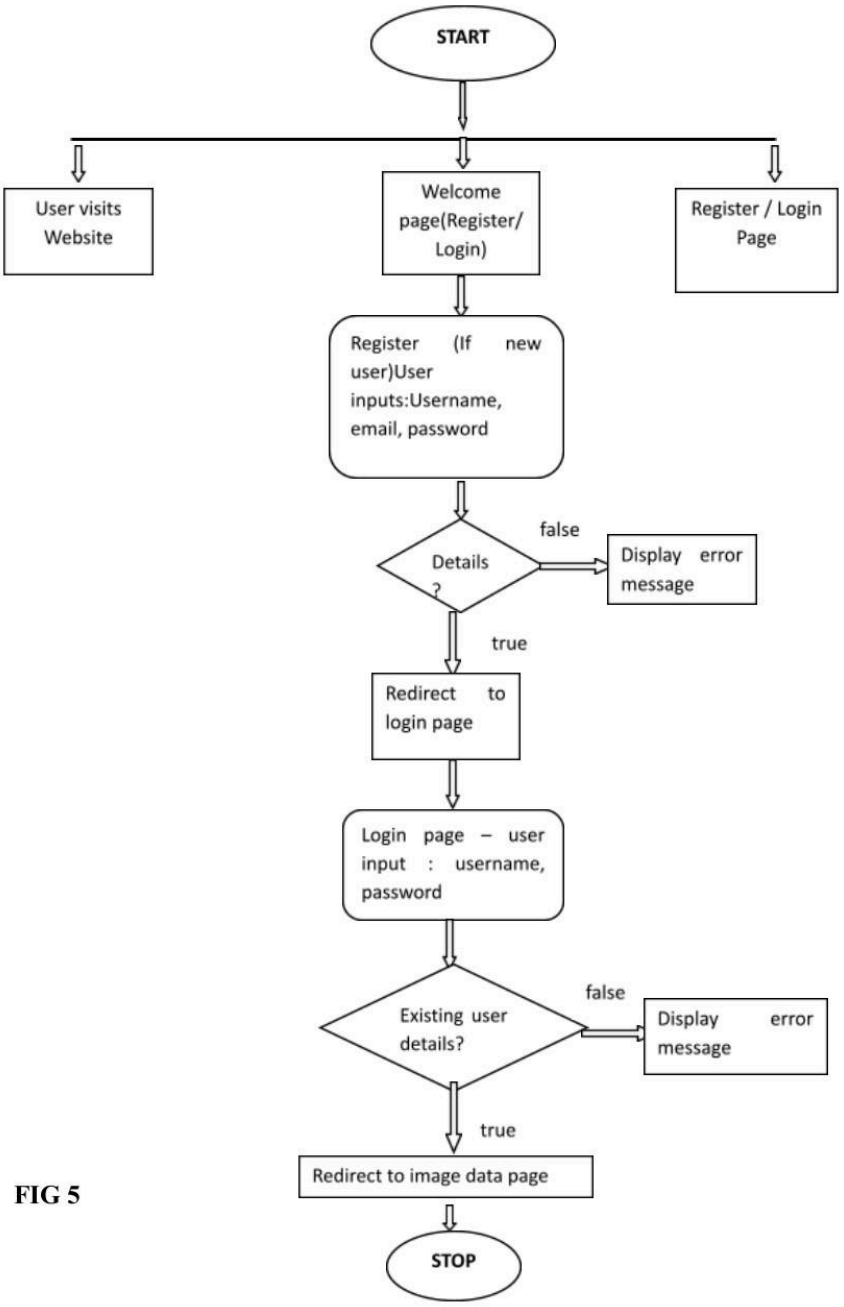


FIG 5

5.THEORETICAL BACKGROUND

5.1 ARCHITECTURES

In 1-tier architecture, also known as monolithic architecture, the entire application is deployed on a single server. This includes the user interface, application logic, and data storage layers, all tightly coupled within the same software bundle. While simple and easy to develop, 1-tier architectures suffer from scalability and maintenance issues due to the lack of separation of concerns.

2-tier architecture separates the application into two layers: the client or presentation layer and the server or database layer. This separation allows for better scalability and maintenance compared to 1-tier architectures, as the client and server layers can be scaled independently. However, 2-tier architectures still face challenges in handling large numbers of concurrent users and managing complex business logic.

3-tier architecture further divides the application into three layers: the presentation layer, the application or business logic layer, and the data storage layer. This architectural style enhances scalability, maintainability, and reusability by separating concerns and decoupling components. Each layer can be scaled independently, and modifications to one layer do not affect the others, leading to easier maintenance and better performance.

5.2 PHP

PHP serves as a cornerstone in web development, offering powerful capabilities, seamless HTML integration, and robust server-side scripting. At its core, PHP operates on the server-side, executing scripts prior to sending resulting HTML to users' browsers. This functionality empowers developers to efficiently handle dynamic content, user authentication, and database interactions, among other critical tasks.

Moreover, PHP's open-source nature and cross-platform compatibility underscore its widespread adoption and versatility. As an open-source language, PHP benefits from a thriving community of developers continuously enriching its ecosystem with new libraries, frameworks, and resources.

5.3 OVERALL THEORETICAL BACKGROUND

Moreover, the project incorporates disaster recovery mechanisms facilitated by AWS's multi-region capabilities. This ensures uninterrupted service by dynamically redirecting traffic in the event of system failures, thereby enhancing reliability and mitigating potential risks. Proactive health checks enable continuous monitoring of the system's condition, facilitating swift responses to maintain high availability and reliability.

By combining AWS's cloud infrastructure with PHP's flexibility and robustness, the project adopts a holistic approach to build a resilient infrastructure that meets the demands of modern applications while effectively mitigating potential risks. This theoretical background lays the foundation for the project's development, emphasizing scalability, reliability, and adaptability as key pillars of the system architecture.

6. HARDWARE & SOFTWARE REQUIREMENTS

6.1 Softwares used:

- **PHP:** PHP serves as the backbone of our web application development, offering flexibility and robustness crucial for creating scalable and responsive interfaces.

- **MARIA DB:** MariaDB, our chosen relational database management system, ensures efficient storage and retrieval of application data.

- **MYSQL Client:** The MYSQL client is an indispensible tool for managing and interacting with our MariaDB database. It facilitates seamless communication between our PHP-based application and the database, enabling the execution of SQL queries, data manipulation operations, and database administration tasks.

- **Apache Server:** Apache server acts as the cornerstone of our web hosting infrastructure, serving as the platform for hosting and delivering our PHP-based web application to users.

6.2 Hardwares used:

- **EC2(Elastic Compute Cloud):** EC2 instances area our scalable virtual sservers in AWS, ensuring optimal performance for our PHP-based applications, They allow us to dynamically adjust resources to meet fluctuating workloads, providing flexibility and cost-efficiency.

- **EFS(Elastic File system):** EFS provides shared file storage accessible from multiple EC2 instances, centralizing our application data storage. Its elastic scalability and high availability ensure reliable data access and durability.

- **RDS(Relational Database Service):** RDS offers managed database instances, ensuring data integrity and availability for our application data storage. With automated backups and maintenance, it simplifies database administrations tasks.

- **Load Balancer:** Load balancers distribute traffic across multiple EC2 instances, optimizing performance and availability for our applications. By dynamically adjusting resources, they enhance resilience and scalability.

- **System with i3 Core Processor and 4GB RAM:** The system with and i3 Core processor and 4GB RAM serves as a development and testing environment for our application allowing the developers to prototype, test and debug code.

7. CODE MODULE

7.1 Code Modules

- **Database Connection Module:** This module establishes a connection to the MYSQL database present on the Amazon RDS (Relational Database Service). It uses the PHP function to connect to the specified host, using the provided username and password. If the connection fails, it terminates the script by displaying the error message.

- **User Registration Module:** In this module, users can register. It consists of an HTML form which include sthe fields username, email address and password. Upon submission, PHP code processes the form data and inserts it into the users table in the database. If registration is successful a message saying successful is displayed along with a link to login. Else if any required fields are missing and error message prompts the user to register again.

- **User Login Module:** This module consists of user authentication. It presents users with a login form containing fields for username and password. After submission, PHP code retrieves the entered credentials from the form and verifies with the login details. If credentials match the existing user, the user is redirected to the next page, else an error message is displayed prompting the user to login again.

- **Main page Module:** This module serves as the landing page for users who visit the application. It displays a simple greeting and provides buttons for users to navigate to the login and registration pages. This apge serves as a centre for user interaction, directing them to the appropriate actions based on the requirement.

- **Image Gallery Module:** This module showcases a gallery of nature images. It utilizies HTML and CSS to style the gallery, displaying the images fetched from the specified image sources. This module enhances the user by presenting the content and contributing to overall user experience of the application.

7.2 CODE SNIPPETS

```
<?php

require('dbs.php');

session_start();


if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['username'], $_POST['password'])) {

   $username = mysqli_real_escape_string($con, $_POST['username']);

   $password = mysqli_real_escape_string($con, $_POST['password']);


      $query = "SELECT username, password FROM users WHERE username = ? AND password = ?";

   $stmt = mysqli_prepare($con, $query);


   mysqli_stmt_bind_param($stmt, "ss", $username, $password);

   mysqli_stmt_execute($stmt);

   mysqli_stmt_store_result($stmt);


   if (mysqli_stmt_num_rows($stmt) > 0) {

      $_SESSION['username'] = $username;

      // Redirect to another page

         header("Location: page2.html"); // Adjust the redirection page as needed

      exit();
```

```
    } else {
                    echo "<div class='form'><h3>Incorrect username or password.</h3><a href='logins.php'>Login again</a></div>";
    }

    mysqli_stmt_close($stmt);
} else {
?>
    <form class="form" action="" method="post">
        <h1 class="lt">Login</h1>
                        <input type="text" class="li" name="username" placeholder="Username" required>
                    <input type="password" class="li" name="password" placeholder="Password" required>
        <input type="submit" value="Login" name="submit" class="lb">
                        <p class="link">Don't have an account? <a href="registrations.php">Register now</a></p>
    </form>
<?php
<!DOCTYPE html>
<html>
<head>
    <title>Login/Register</title>
    <style>
        body {
```

```
    font-family: Arial, sans-serif;
    background: #f9f9f9;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}
.card {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
    text-align: center;
    width: 300px; /* You can adjust this if needed */
}
.button {
    display: block;
    width: auto; /* Adjust width to content */
    padding: 8px 16px; /* Reduced padding */
    margin: 5px auto; /* Center button and reduce margin */
    font-size: 14px; /* Adjust font size */
    color: white;
```

```
            background: #007bff;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            text-decoration: none;
            transition: background 0.3s;
        }
        .button:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <div class="card">
        <h2>Welcome</h2>
        <a href="logins.php" class="button">Login</a>
        <a href="registrations.php" class="button">Register</a>
    </div>
</body>
</html>
```

8.TESTING AND DEPLOYEMENT

8.1 OUTPUTS

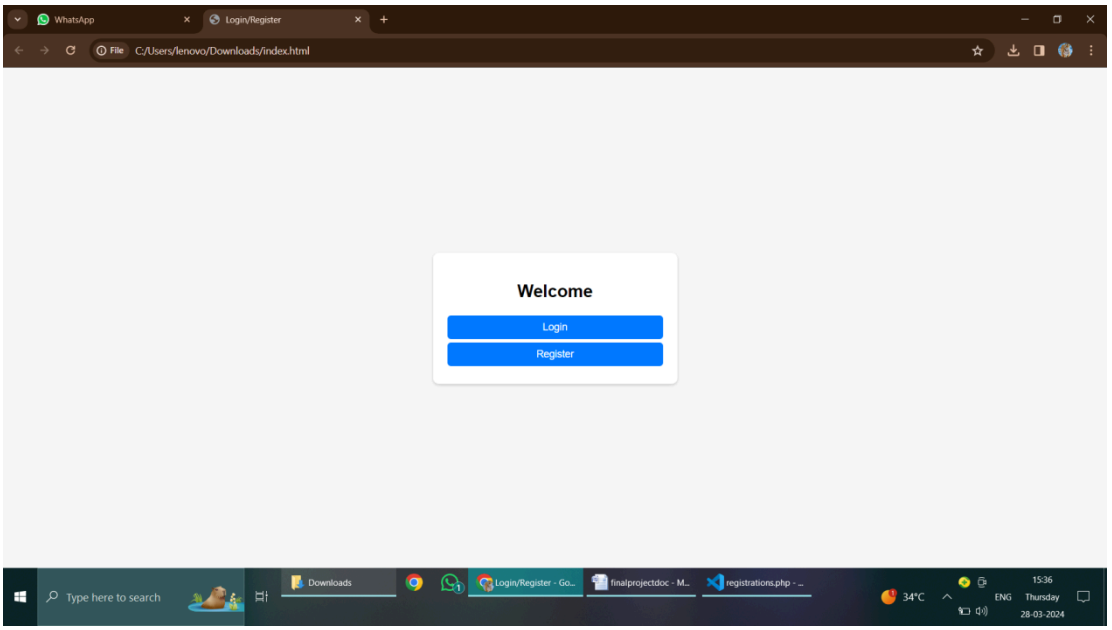


FIG 6

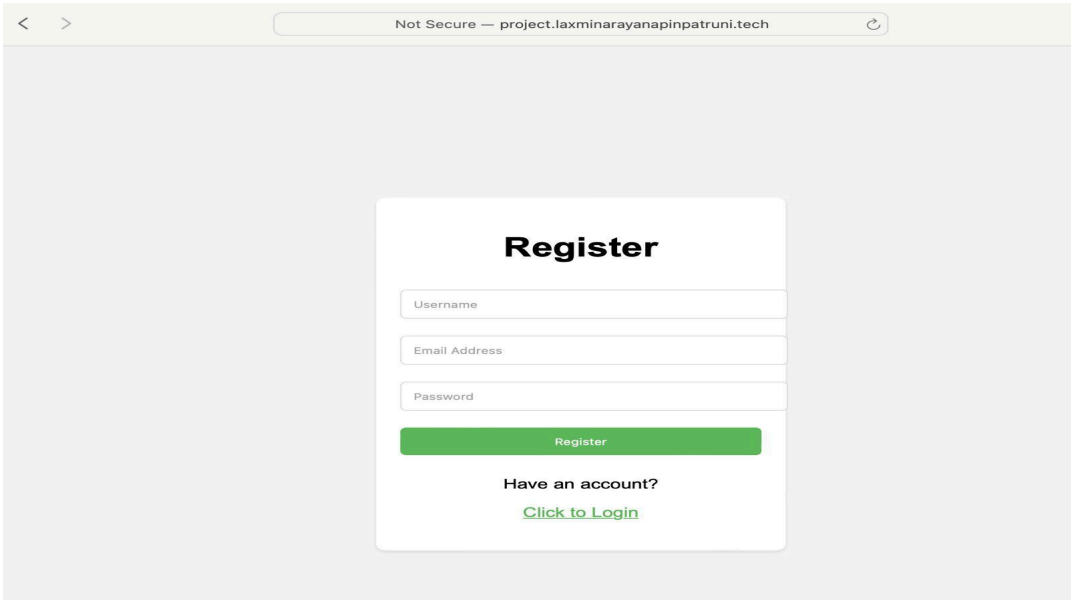


FIG 7

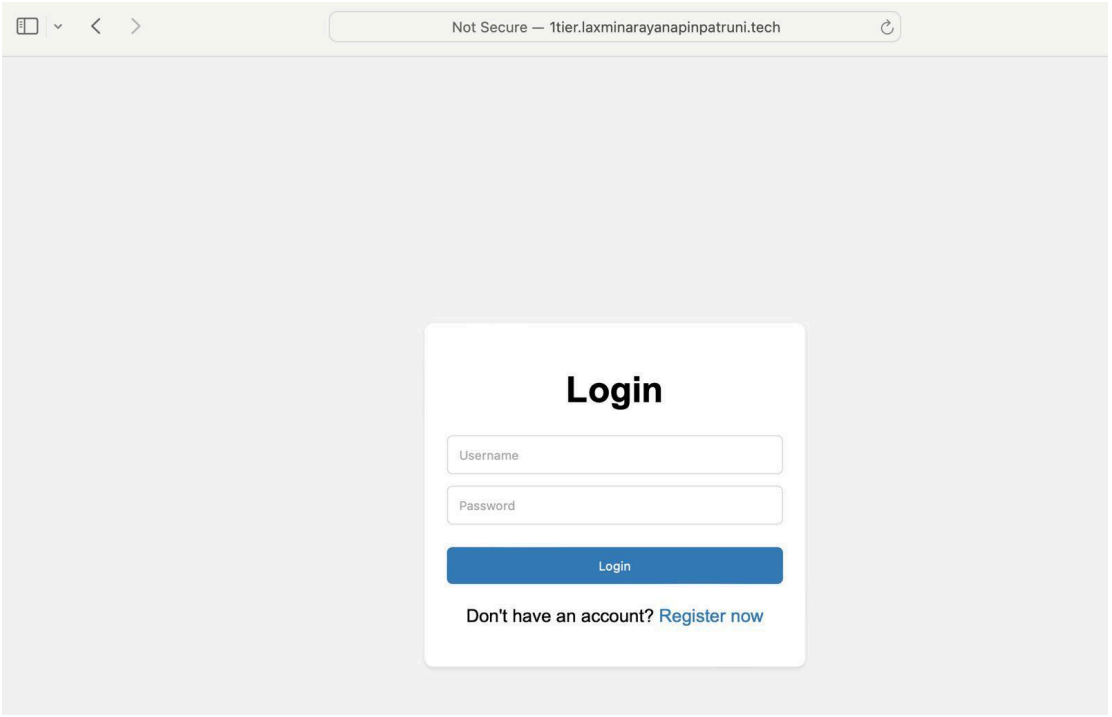

Not Secure — 1tier.laxminarayanapinpatruni.tech
Login
Username
Password
Login
Don't have an account? Register now

FIG 8

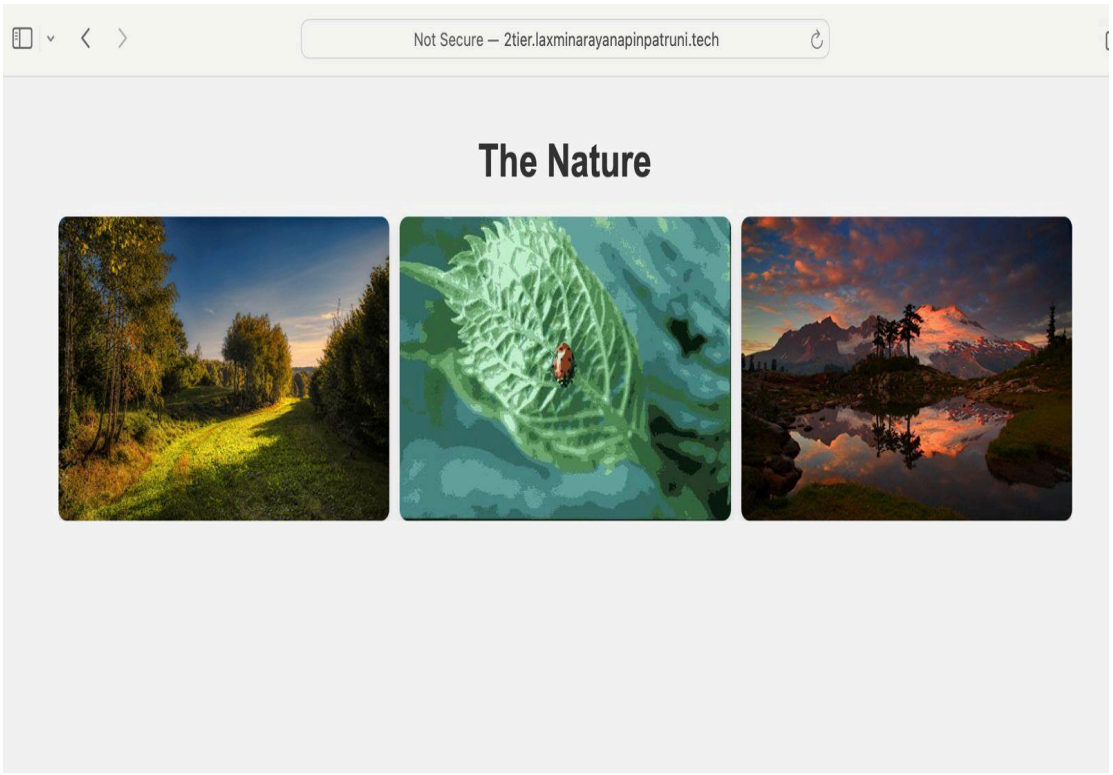

Not Secure — 2tier.laxminarayanapinpatruni.tech
The Nature

FIG 9

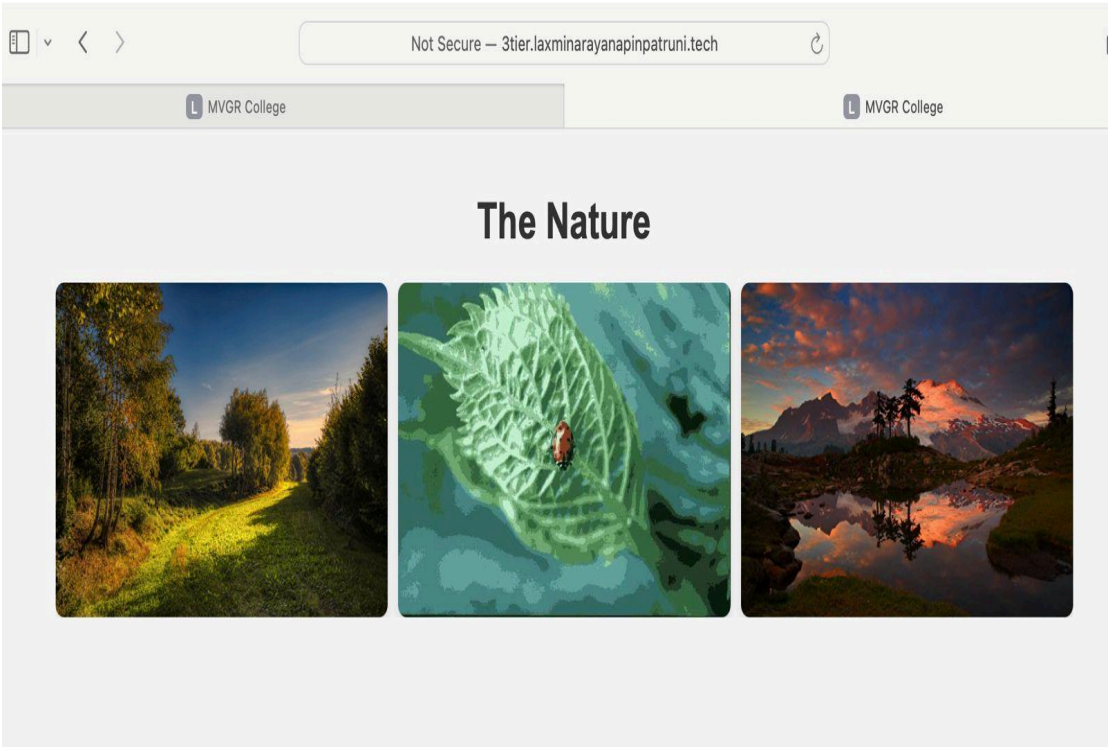

Not Secure — 3tier.laxminarayanapinpatruni.tech
MVGR College
MVGR College
The Nature

FIG 10

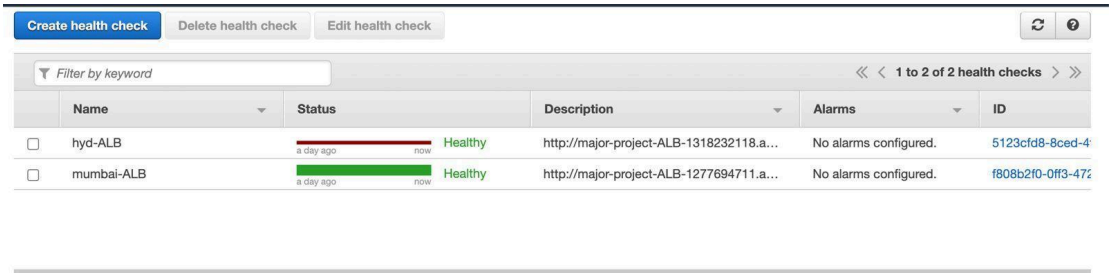

Create health check
Delete health check
Edit health check
Filter by keyword
1 to 2 of 2 health checks
Name
Status
Description
Alarms
ID
hyd-ALB
a day ago
now
Healthy
http://major-project-ALB-1318232118.a...
No alarms configured.
5123cfd8-8ced-4
mumbai-ALB
a day ago
now
Healthy
http://major-project-ALB-1277694711.a...
No alarms configured.
f808b2f0-0ff3-472

FIG 11

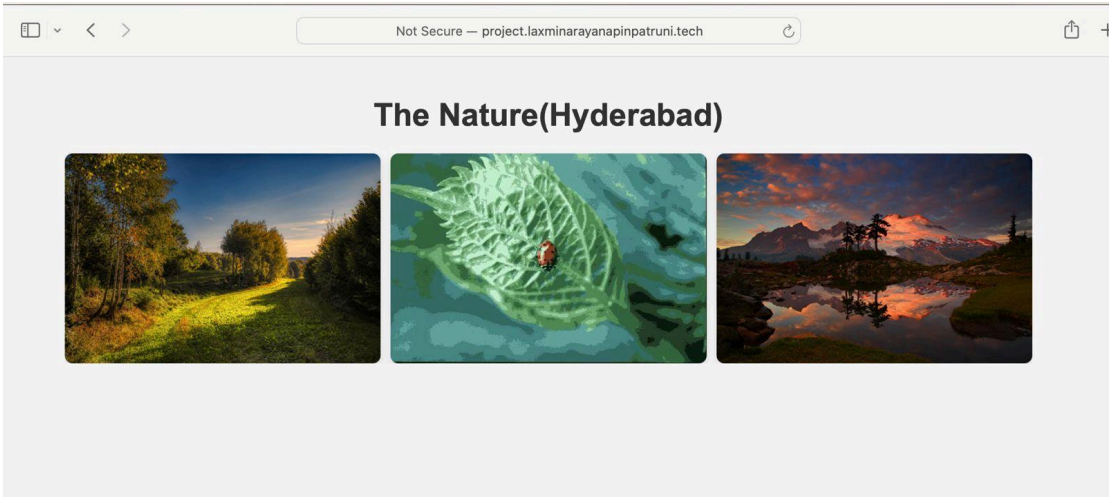

Not Secure — project.laxminarayanapinpatruni.tech
The Nature(Hyderabad)

FIG 12


Filter by keyword
1 to 2 of 2 health checks
Name
Status
Description
Alarms
ID
hyd-ALB
Unhealthy
a day ago
now
http://major-project-ALB-1318232118.a...
No alarms configured.
5123cfd8-8ced-4
mumbai-ALB
Healthy
a day ago
now
http://major-project-ALB-1277694711.a...
No alarms configured.
f808b2f0-0ff3-472

FIG 13

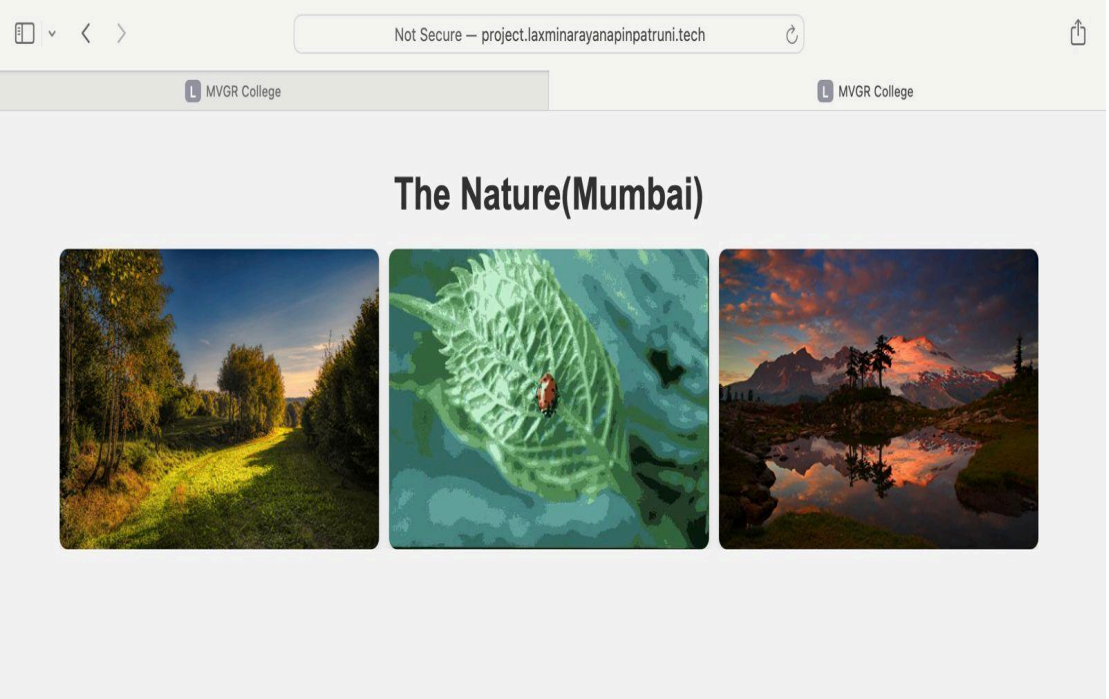

Not Secure — project.laxminarayanapinpatruni.tech
MVGR College
MVGR College
The Nature(Mumbai)

FIG 14

9.CONCLUSION AND FUTURE SCOPE

9.1 CONCLUSION

In the journey from 1-tier to 3-tier architectures using AWS resources such as EC2 instances, EFS, RDS, and PHP, we've explored the evolution of web application architecture to achieve scalability, reliability, and efficiency. Starting with 1-tier architecture, we observed its limitations in handling scalability and fault tolerance. Moving to 2-tier architecture provided some improvements in separation of concerns but still lacked the scalability and resilience needed for modern web applications. Finally, transitioning to a 3-tier architecture with load balancing and auto-scaling capabilities enabled us to efficiently handle traffic spikes and ensure high availability.

Utilizing AWS services like EC2 instances, EFS, and RDS allowed us to build scalable storage and database solutions while leveraging PHP for server-side processing and API development. The incorporation of load balancing and auto-scaling mechanisms enhanced the system's ability to dynamically adapt to varying workloads, ensuring consistent performance and availability.

Our project exemplifies the transformative power of transitioning from traditional architectures to cloud-native solutions, facilitated by AWS resources and PHP. This evolution not only addresses the limitations of legacy architectures but also sets the stage for building resilient, agile, and scalable systems capable of meeting the evolving demands of modern web applications. Through strategic utilization of cloud infrastructure and programming languages like PHP, we've laid the foundation for a robust and future-ready architecture capable of thriving in dynamic and competitive digital environments.

9.2 FUTURE SCOPE

- **Container Orchestration:** Explore Kubernetes as a container orchestration solution to automate deployment, scaling, and management of containerized applications, enhancing agility and scalability.

- **Serverless Data Processing:** Implement AWS services like AWS Glue and Amazon Athena for serverless data processing, enabling efficient extraction, transformation, and analysis of large datasets without the need for managing infrastructure.

- **Real-time Monitoring and Analytics:** Integrate AWS CloudWatch and Amazon CloudWatch Logs for real-time monitoring and analytics, enabling proactive identification of performance bottlenecks and system anomalies for immediate resolution.

- **Immutable Infrastructure:** Embrace the concept of immutable infrastructure using AWS CloudFormation or AWS CDK to automatically provision and manage infrastructure as code, ensuring consistency and reproducibility across environments.

- **Integration with AWS AI/ML Services:** Utilize AWS AI/ML services such as Amazon Rekognition and Amazon Comprehend to incorporate advanced AI-driven features like image recognition and natural language processing into the system, enhancing user engagement and experience.

10.REFERENCES

[1] E. a=Al-Payis and H. Kurdi, “Performance Analysis of Load Balancing Architectures in Cloud Computing,”2013 European Modelling Symposium, Manchester, UK, 2013, pp. 520-524, doi: 10.1109/EMS.2013.10.
http://ieeexplore.ieee.org/document/6779898

[2] M. Pokharel, S. Lee and J. S. Park, “Disaster Recovery for System Architecture Using Cloud Computing,” 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet, Seoul, Korea (South), 2010, pp.304-307, doi: 10.1109/SAINT.2010.23.
https://ieeexplore.ieee.org/document/5598055

[3] Parminder Singh, Pooja Gupta, Kiran Jyoti, Anand Nayyar, “Research on Auto-Scaling of Web Applications in Cloud: Survey, Trends and Future Directions,” 2019 Auto Scaling techniques for cloud application to estimate the required resources to process the input requests. doi: 10.12694/scpe.v20i2.1537
https://www.researchgate.net/publication/332828653_Research_on_Auto-Scaling_of_Web_Applications_in_Cloud_Survey_Trends_and_Future_Directions

[4] Jun Tie, Jia Jin and Xiaorong Wang, “Study on application model of three-tiered architecture,” 2011 Second International Conference on Mechanic Automation and Control Engineering, Inner Mongolia, China, 2011, pp.7715-7718, doi: 10.1109/MACE.2011.5900038.
http://ieeexplore.ieee.org/document/5988838

[5] Mohamed Ghetas, Huah-Yong Chan and Sumari Putra, “A survey of quality of service in multi-tier web applications,” 2016 The infrastructure of modern web applications employs multi-tier architecture to offer flexibility, a modular approach, scalability an reliability in deploying web services. doi:10.3837/tiis.2016.01.014
https://www.researchgate.net/publication/297047540_A_survey_of_quality_of_service_in_multi-tier_web_applications

[6] W. -H. Liao, S. -C. Kauai and Y. -R. Leau, “Auto-scaling Strategy for Amazon Web Services in Cloud Computing,” 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), Chengdu, China, 2015, pp.1059-1064, doi: 10.1109/SmartCity.2015.209.
https://ieeexplore.ieee.org/document/7463864

[7] P. A. Abdalla and A. Varol, “Advantages to Disadvantages of Cloud Computing for Small-sized Business,” 2019 7th International Symposium on Digital Forensics and Security (ISDFS), Barcelos, Portugal, 2019, pp. 1-6, doi: 10.1109/ISDFS.2019.8757549.
https://ieeexplore.ieee.org/document/8757549