THREE TIER USER MANAGEMENT APPLICATION USE CASE

 Aws offers a comprehensive set of networking services that enables organizations to build and manage robust and scalable network architecture.

In my previous employment where I worked as a solutions architect, I was tasked in designing and setting up a three tier architecture for the hosting of one of the java based applications that was being developed.

In this software architecture pattern the application was broken down into three logical tiers: -

-The Presentation layer (Client Layer)

-The business logic layer (Application layer)

-The Data storage layer (Database Server).

This web application had a frontend, backend and a database.

The reason I went with the three-tier architecture was because of the fact that: -

The three tier allows for different development teams to each work on their own areas of expertise.

You are able to scale the application up and out.

It gives you the ability to update the technology stack of one tier without impacting other areas of the application.

 In the course of designing my architecture I took into consideration the six pillars of a well architected framework which are operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability.

What I did was, at the front end I configured route 53 for DNS (Domain Name System) resolution.

(When configuring my route 53 I Map it to the elastic IP of the LB, so as to direct a certain percentage of traffic to different end points with predefined weights)

I then set up WAF (Web Application Firewall) (to prevent DDOS (Distributed denial of service) attacks. This sat at the entry point of the application.

In the 1st layer resided my public subnet, which contained my NAT gateway and my load balancer.

I then used Amazon certificate manager to get the SSL (Secure Socket Layer) certificate to secure the traffic coming into my environment from HTTP - HTTPS

I achieved this by enforcing SSL at the level of my Load Balancer to convert   all http traffic to https (Https is more secured traffic compared to Http)

(When configuring my SG (Security Group) for my apps and web servers, I specified the source to be the elastic IP of the LoadBalancer sitting in the public subnet)

I kept my webservers, appServers and Databases in separate private subnets and for their security groups I specified the source to be the elastic IP of the load balancer.

At the next Tier I had the webservers. Here  I decided to use Nginx server because of its reverse proxy capabilities, i.e. it intercepts and forwards requests from clients and returns the server’s response to the clients.

At the level of application layer I used Apache Tomcat since it was a Java based application. There I  specified the necessary authentication for our database, by specifying the identifiers name and password.

I decided to go with Amazon's Relational Database Service (RDS) since  it was a managed database service provided by Amazon. While creating the RDS database I created a read-only replica in one availability zone and another standby in another region, in case of a fail over.

I ensured that AWS best practices was enforced by configuring Auto scaling for high availability and multi AZ deployments. I also ensured that security was enforced at all levels.

This was realized using Infrastructure as code (IAC), terraform to be specific.