

IMPLEMENTATION OF SECURE WEB HOSTING USING HTTPS ON MICROSOFT AZURE

A PROJECT REPORT

Submitted by

SURESH KUMAR D	110521104049
VIMALESH C	110521104310
YOGESH BALAJI K	110521104058
RAMA KRISHNAN A	110521104032

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

**GOJAN SCHOOL OF BUSINESS AND TECHNOLOGY
EDAPALAYAM, REDHILLS, CH-52**

ANNA UNIVERSITY: CHENNAI 600 025

APRIL/MAY 2025

BONAFIDE CERTIFICATE

Certified that this project report “**IMPLEMENTATION OF SECURE WEB HOSTING USING HTTPS ON MICROSOFT AZURE**” is the Bonafide work of **SURESH KUMAR D (110521104049), VIMALESH C (110521104310), YOGESH BALAJI K (110521104058), RAMA KRISHNAN A (110521104032)** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported here is not a part of any other project work.

SIGNATURE

HEAD OF DEPARTMENT

Mrs. REVATHI. L, M.E.,
Department of Computer Science and
Engineering
Gojan School of Business and
Technology
80 Ft road, Edapalayam,
Redhills, Chennai-52

SIGNATURE

SUPERVISOR

Mrs. ANJALI. S, M.E.,
Department of Computer Science
and Engineering
Gojan School of Business and
Technology
80 Ft road, Edapalayam,
Redhills, Chennai-52

Submitted for the Project Viva Voce Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

SURESH KUMAR D

Department of Computer Science and Engineering

Gojan School of Business and Technology

DECLARATION

I hereby declare that the project entitled “**IMPLEMENTATION OF SECURE WEB HOSTING USING HTTPS ON MICROSOFT AZURE**” submitted by us to Anna University in May/June 2025 for the award of degree of Bachelor of Engineering in Computer Science and Engineering is the original and independent team work carried out by me under the guidance **Mrs. ANJALI. S, M.E.**, Assistant professor, and that it has not been formed the basis fully or partially for the award of any degree, diploma or other similar titles earlier and no part of above project work has been published or sent for the publication at the time of submission.

Place: - Chennai

Date: -

Signature

VIMALESH C

Department of Computer Science and Engineering

Gojan School of Business and Technology

DECLARATION

I hereby declare that the project entitled “**IMPLEMENTATION OF SECURE WEB HOSTING USING HTTPS ON MICROSOFT AZURE**” submitted by us to Anna University in May/June 2025 for the award of degree of Bachelor of Engineering in Computer Science and Engineering is the original and independent team work carried out by me under the guidance **Mrs. ANJALI. S, M.E.**, Assistant professor, and that it has not been formed the basis fully or partially for the award of any degree, diploma or other similar titles earlier and no part of above project work has been published or sent for the publication at the time of submission.

Place: - Chennai

Date: -

Signature

YOGESH BALAJI K

Department of Computer Science and Engineering

Gojan School of Business and Technology

DECLARATION

I hereby declare that the project entitled “**IMPLEMENTATION OF SECURE WEB HOSTING USING HTTPS ON MICROSOFT AZURE**” submitted by us to Anna University in May/June 2025 for the award of degree of Bachelor of Engineering in Computer Science and Engineering is the original and independent team work carried out by me under the guidance **Mrs. ANJALI. S, M.E.**, Assistant professor, and that it has not been formed the basis fully or partially for the award of any degree, diploma or other similar titles earlier and no part of above project work has been published or sent for the publication at the time of submission.

Place: - Chennai

Date: -

Signature

RAMA KRISHNAN A

Department of Computer Science and Engineering

Gojan School of Business and Technology

DECLARATION

I hereby declare that the project entitled “**IMPLEMENTATION OF SECURE WEB HOSTING USING HTTPS ON MICROSOFT AZURE**” submitted by us to Anna University in May/June 2025 for the award of degree of Bachelor of Engineering in Computer Science and Engineering is the original and independent team work carried out by me under the guidance **Mrs. ANJALI. S, M.E.**, Assistant professor, and that it has not been formed the basis fully or partially for the award of any degree, diploma or other similar titles earlier and no part of above project work has been published or sent for the publication at the time of submission.

Place: - Chennai

Date: -

Signature

ACKNOWLEDGEMENT

We express our deepest gratitude to our **Chairman Dr. G. Natarajan, Ph.D.**, and **Chairperson Mrs. Brindha Natarajan, B. Com**, for their valuable guidance and blessings.

We are deeply indebted to our beloved **Principal Dr. C. Selvakumar, Ph.D.**, Gojan School of Business and Technology, for providing us an excellent environment to carry out our course successfully.

We express our sincere thanks to our **Head of the Department Mrs. Revathi. L, M.E.**, Assistant Professor, who has been a constant source of inspiration and guidance in the course of the project.

We record our thanks to our **Supervisor Mrs. ANJALI. S, M.E.**, Assistant Professor, for being instrumental in the completion of our project with his exemplary guidance.

We thank all the **Staff Members** of our department for their valuable support and assistance at various stages of our project development.

Finally, we take this opportunity to extend our deep sense of gratitude and appreciation to our family and friends for all that they meant to us during the crucial times of the completion of our project.

ABSTRACT

This project delves into the Implementation of secure web hosting using HTTPS on Microsoft Azure, with a focus on leveraging the capabilities of Azure Application Gateway to ensure secure, scalable, and reliable access to web applications. In the current digital landscape, where cyber threats are increasingly sophisticated, the security of web applications has become a top priority for developers, businesses, and users. Ensuring that data transmitted between clients and servers is encrypted is essential to protect against attacks such as data breaches, man-in-the-middle attacks, and eavesdropping.

Microsoft Azure provides a powerful networking service called Azure Application Gateway, which operates at the application layer (OSI Layer 7). This service allows intelligent routing of client requests based on HTTP/HTTPS parameters, making it an ideal choice for hosting modern web applications. In this project, the Application Gateway is configured to serve as a reverse proxy that handles HTTPS requests through SSL/TLS termination, where Application gateway with Virtual Machine have some flaw consequently we migrate to Application gateway with Web Application.

To further the project involves the complete deployment and configuration of a sample web application hosted in Azure, connected to the Application Gateway. It includes steps such as provisioning resources, uploading and binding SSL certificates, configuring backend pools, setting up listeners and rules, and enabling diagnostic logging for monitoring and troubleshooting. The outcome is a secure, HTTPS-enabled web application infrastructure that meets modern standards for security, reliability, and scalability, making it suitable for real-world enterprise and cloud-native applications.

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO
1.1	Azure Application Gateway with VM diagram	9
3.1.1	Diagram of Vnet	21
3.1.2	Diagram of Subnet	21
3.1.3	Virtual Machine Diagram	21
3.2	IIS Installation in VM	22
3.4.1	Adding VM in application gateway backend	26
3.4.2	Adding backend setting in Application gateway	26
3.4.3	Adding Rule in Application Gateway	27
3.4.4	Application Gateway with VM Output	27
3.5	Web Application provision	28
3.6.1	Sequence Diagram	27
3.6.2	Class Diagram	28
3.6.3	Use Case Diagram	28
3.6.4	Activity Diagram	29
3.6.5	Collaboration Diagram	30
3.7.1	Web Application Output	36
3.8	AppGateway web app backend	35
3.8.1	DDOS Attack	36
3.8.4	Three-way hand shake	38
3.10	Activity diagram for WAF Authentication	49
3.12	AppGateway Rules Architecture	55

3.13.1	Diagram of Backend Health Probe	56
3.13.2	AppGateway WAF Data Flow Diagram	57
4.1	Web Application Firewall (WAF)	59
5.1	Custom Domain Output	62

LIST OF ABBREVIATIONS

VM	-	Virtual Machine
WebApp	-	Web Application
IIS	-	Internet Information Services
APPGateway	-	Application Gateway
VNET	-	Virtual Network
SUBNET	-	Virtual Subnet
WAF	-	Web Application Firewall

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	Viii
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	x
1.	INTRODUCTION	1
	1.1 Problem Definition	2
	1.2 Existing System	3
	1.3 Proposed System	4
	1.4 Introduction to Azure AppGateway	6
2.	LITERATURE REVIEW	10
3.	THEORETICAL BACKGROUND	20
	3.1 Implementation Environment	20
	3.2 IIS Installation using PowerShell	22
	3.3 IIS Installation using Server Manager	23
	3.4 Connecting Virtual Machine to AppGateway	24
	3.5 Web Application Provision	26
	3.6 Module Design	27
	3.6.1 Sequence Diagram	27
	3.6.2 Class Diagram	28
	3.6.3 Use Case Diagram	28
	3.6.4 Activity Diagram	29
	3.6.5 Collaboration Diagram	30

CHAPTER NO.	TITLE	PAGE NO.
3.7	CSS – Index.html file	31
	3.7.1 Web Application Output	35
3.8	App Gateway with Web Application Backend pool	37
	3.8.1 DDOS Attack	38
	3.8.2 THREE-WAY Handshake	38
	3.8.3 THREE-WAY Handshake in Azure Application Gateway	39
	3.8.4 Step By Step Flow	39
	3.8.5 Mapping Azure Application Gateway Public Ip into GoDaddy Domain	40
3.9	Json Script of Application Gateway	41
3.10	App Gateway secure webhosting enable with WAF	49
3.11	Web Hosting with SSL	50
	3.11.1 Web Hosting with Custom Domain Binding	50
	3.11.2 SSL Binding with Custom Domain	51
	3.11.3 Application Gateway Backend Target	52
	3.11.4 Custom listeners	53
3.12	Application Gateway Rules	54
3.13	Health Probe	55
	3.13.1 Backend – Health Probe Diagram	55
	3.13.2 Application Gateway WAF Dataflow Diagram	56

CHAPTER NO.	TITLE	PAGE NO.
4.	WEB APPLICATION FIREWALL	58
4.1	WAF Description	58
4.2	OWASP Top 10 CRS	59
5.	RESULTS & DISCUSSION	62
5.1	Output	62
6.	CONCLUSION AND FUTURE WORK	63
6.1	Conclusion	63
6.2	Future Work	63
	REFERENCES	65

CHAPTER 1

INTRODUCTION

The rise of cloud computing has fundamentally transformed how web applications are developed, deployed, and scaled. Businesses and developers are increasingly leveraging cloud platforms to build applications that are more flexible, efficient, and accessible. However, this digital shift has also introduced new challenges particularly in the areas of web security, traffic management, and performance optimization. In this context, secure web hosting has become a top priority, especially for applications that handle sensitive user data, financial transactions, or personal information.

One of the most effective and widely adopted methods for securing web traffic is the implementation of HTTPS (Hypertext Transfer Protocol Secure), which ensures that communication between the user and the server is encrypted using SSL/TLS protocols. HTTPS not only safeguards against threats such as data interception, man-in-the-middle attacks, and eavesdropping, but also enhances trust and credibility with users. As security becomes a baseline expectation rather than an optional feature, hosting web applications over HTTPS is now considered an industry best practice.

Microsoft Azure offer advanced networking services such as the Azure Application Gateway. This cloud-native service operates at the application layer (OSI Layer 7) and provides features such as SSL termination, Web Application Firewall (WAF), autoscaling, and intelligent traffic routing. These capabilities make it possible to create highly available, secure, and scalable hosting environments for web applications.

This project focuses on implementing a secure web hosting architecture using HTTPS and Microsoft Azure Application Gateway. It demonstrates how to deploy a web application behind the gateway, configure SSL certificates, enable WAF protection, and manage routing rules for optimized performance. By leveraging Azure's built-in tools and best practices, this project serves as a model for deploying secure, cloud-based applications that can adapt to modern security threats and usage patterns.

1.1 PROBLEM DEFINITION

As the dependency on web applications continues to grow, so does the need for secure and reliable hosting environments. A significant number of web applications today are deployed without adequate security configurations, leaving them vulnerable to cyber threats such as data interception, cross-site scripting (XSS), SQL injection, and man-in-the-middle (MITM) attacks. One of the primary causes of such vulnerabilities is the lack of HTTPS implementation and proper traffic management practices.

Moreover, traditional web hosting solutions often struggle to meet the performance, scalability, and security demands of modern applications, especially under dynamic workloads. Inadequate handling of SSL certificates, lack of centralized security policies, and poor scalability can lead to system downtime, user data breaches, and performance bottlenecks.

Cloud-native tools such as Microsoft Azure Application Gateway offer solutions to these problems through features like SSL/TLS termination, Web Application Firewall (WAF), autoscaling, and URL-based routing. However, many developers and organizations face challenges in correctly configuring and deploying these features due to a lack of technical understanding or standardized implementation guidelines.

This project addresses the core problem of insecure and inefficient web hosting practices by designing and implementing a secure web hosting architecture using HTTPS and Microsoft Azure Application Gateway. It aims to demonstrate a practical and scalable solution that not only encrypts web traffic but also intelligently manages requests and defends against common web application attacks, thereby ensuring confidentiality, availability, and integrity of web services in a cloud environment.

1.2 EXISTING SYSTEM

In many traditional web hosting environments, applications are hosted on single servers or virtual machines without robust mechanisms for traffic management, load balancing, or advanced security. These setups typically rely on self-managed HTTPS configurations, where SSL certificates are manually installed and renewed on individual servers, it introduces several limitations when applied to enterprise-level or high-traffic web applications.

Furthermore, existing systems often use basic load balancers or network-level firewalls, which provide only limited security and routing capabilities. These systems lack application-layer intelligence, making them unable to route traffic based on content or to defend against threats that target the application layer (e.g., SQL injection, XSS, etc.).

Some of the key drawbacks of traditional or existing web hosting systems include:

- **Manual SSL/TLS management:** Certificates must be installed and renewed on each server, increasing the risk of expiration and misconfiguration.
- **Limited scalability:** As traffic grows, manual intervention is often required to add or configure more servers.
- **Poor security posture:** Without integrated firewalls or security monitoring, applications remain vulnerable to a wide range of attacks.
- **No intelligent routing:** Traditional systems route traffic based on IP or port, with no ability to make content-based routing decisions.
- **Lack of centralized control:** Security, performance, and monitoring tools are often spread across multiple platforms and services, making management inefficient.

Due to these limitations, traditional systems are not ideal for hosting modern, dynamic, and security-critical web applications, especially in a cloud-first world. There is a clear need for an automated, secure, and cloud-native solution that can manage web traffic efficiently while ensuring robust protection against threats.

1.3 PROPOSED SYSTEM

To address the limitations of traditional web hosting and provide a more secure, scalable, and manageable solution, this project proposes the deployment of a **web Application hosted on an Azure Virtual Machine (VM)**, protected and managed by an **Azure Application Gateway**. This setup leverages Azure's cloud-native services to enhance **security, performance, and availability** for modern web applications.

In the proposed system, the **web application** runs on a Virtual Machine inside an Azure **Virtual Network (VNet)**. The **Azure Application Gateway** is placed in front of the VM, acting as an **application-layer (OSI Layer 7) load balancer and reverse proxy**. It receives all incoming HTTP/HTTPS requests and intelligently routes them to the backend VM hosting the web application. It also terminates SSL connections to ensure **end-to-end encryption** and **centralized certificate management**.

Key Features of the Proposed System:

- **Secure HTTPS Hosting:**

SSL/TLS termination at the Application Gateway enables encrypted communication between clients and the server.

Centralized certificate management simplifies renewal and reduces the risk of misconfiguration.

- **Web Application Firewall (WAF):**

Integrated WAF protects the application from common web attacks such as SQL injection, cross-site scripting (XSS), and other OWASP Top 10 vulnerabilities.

Security policies can be customized and updated regularly.

- **Intelligent Traffic Management:**

URL-based routing allows specific paths (e.g., /api, /admin) to be routed to different backends if needed.

Session affinity and health probes ensure reliable user sessions and application uptime.

- **Scalability and High Availability:**

The Application Gateway supports autoscaling based on demand, ensuring consistent performance during traffic spikes.

The VM can be part of an availability set or scale set for further resilience.

- **Monitoring and Diagnostics:**

Azure Monitor and Application Gateway diagnostics provide real-time visibility into traffic patterns, security logs, and performance metrics.

- **Isolation and Control via Virtual Network (VNet):**

The entire setup is deployed within a secure Azure VNet, ensuring isolation from public access except through the gateway.

- Benefits of the Proposed System.
- Centralized security and traffic control.
- Simplified SSL certificate handling.
- Enhanced protection from application-layer threats.
- Better scalability and availability compared to traditional hosting.
- Full integration with Azure monitoring and automation tools.

This proposed system effectively combines **infrastructure-level control (via VMs)** with **application-layer security and performance optimization (via Azure Application Gateway)**. It provides a robust and future-ready architecture for securely hosting web applications in the cloud.

1.4 INTRODUCTION TO AZURE APPLICATION GATEWAY

In the era of digital transformation, the way organizations host and manage web applications has evolved drastically. Traditional hosting methods often struggle to meet the performance, scalability, and security demands of modern applications, especially those that handle sensitive user data, process complex transactions, or experience varying traffic loads. With this growing complexity, cloud-native solutions like Microsoft Azure have emerged as powerful alternatives, offering a suite of services designed to ensure web applications are secure, scalable, and high-performing.

One of the key services offered by Microsoft Azure to address these challenges is the Azure Application Gateway, which acts as an intelligent Layer 7 (application layer) load balancer and traffic management service. Unlike traditional load balancers, the Azure Application Gateway is designed to manage traffic specifically for web applications, making it an ideal solution for businesses that need to ensure secure, optimized, and efficient traffic delivery to their hosted services.

This project focuses on utilizing Azure Application Gateway in conjunction with Azure Virtual Machines (VMs) to host a secure, high-performance web application. The architecture combines the flexibility and control provided by VMs with the advanced traffic management and security features of the Application Gateway, creating a scalable and reliable web hosting environment.

Key Components of the Architecture:

➤ Azure Virtual Machine (VM) as Backend:

The Virtual Machine in Azure serves as the primary host for the web application. This VM could be running a variety of web technologies (e.g., Apache, Nginx, Node.js, etc.), allowing for full control over the application and its environment.

VMs can be part of an availability set or scale set to ensure redundancy and high availability, reducing the risk of downtime and improving

performance under varying traffic conditions.

➤ **Azure Application Gateway:**

The Application Gateway sits in front of the VM and acts as the entry point for all incoming HTTP/HTTPS traffic. This service is a Layer 7 load balancer that not only handles routing of requests but also offers advanced traffic management features like URL-based routing, host-based routing, and session affinity.

➤ **SSL/TLS termination:**

At the Application Gateway ensures that secure communication is handled efficiently, freeing the backend VM from the overhead of decrypting traffic.

As a reverse proxy, it forwards client requests to the appropriate backend server while protecting the infrastructure by hiding the internal network from direct internet exposure.

➤ **Web Application Firewall (WAF):**

Integrated WAF adds a critical layer of security by protecting the application from common vulnerabilities and threats such as SQL injection, Cross-Site Scripting (XSS), and Distributed Denial-of-Service (DDoS) attacks. The WAF follows industry standards (e.g., OWASP Top 10) to mitigate attacks and safeguard sensitive data.

With the WAF enabled, the Application Gateway can automatically filter out malicious requests before they even reach the backend VM, ensuring that the web application remains secure even in the face of evolving attack methods.

➤ **Autoscaling and Performance Optimization:**

The Azure Application Gateway can automatically **scale up or down** based on incoming traffic. This feature helps ensure that the system remains highly responsive during periods of high demand, without manual intervention.

Health probes monitor the health of backend VMs and ensure that traffic is only routed to healthy instances, ensuring high availability and minimal downtime for users.

➤ **Centralized Certificate Management:**

The Application Gateway provides SSL/TLS termination capabilities, meaning it handles the encryption and decryption of HTTPS traffic. It is easier to maintain and update certificates across multiple servers or services without needing to configure them on each individual VM.

➤ **Virtual Network (VNet) and Network Security:**

The Azure Virtual Network (VNet) provides network isolation for all Azure resources, including the VM and the Application Gateway. This isolation enhances security, ensuring that only traffic routed through the Application Gateway can reach the backend VM.

➤ **Network Security Groups (NSGs)**

It can be used to define strict access control policies, further protecting the backend infrastructure and controlling who can access the application.

Benefits of the Proposed Architecture:

➤ **Improved Security:**

The integration of Web Application Firewall (WAF) ensures proactive defense against web-based threats such as SQL injection, cross-site scripting (XSS), and other common vulnerabilities. Furthermore, SSL termination at the gateway ensures secure communication between clients and the application.

➤ **Scalability:**

The architecture is designed to automatically scale based on traffic patterns. Both the Application Gateway and the backend VMs can dynamically adjust to accommodate increases or decreases in traffic, ensuring optimal performance during varying usage scenarios.

➤ **Performance Optimization:**

With URL-based routing, session affinity, and autoscaling features, the architecture ensures that web traffic is intelligently managed, reducing response times and improving the user experience.

➤ **Centralized Management:**

By centralizing traffic management, SSL/TLS termination, and security policies through the Azure Application Gateway, the system reduces operational overhead and simplifies the management of the web application.

➤ **High Availability:**

The use of availability sets or scale sets ensures that the system can tolerate individual component failures without disrupting the service, providing resilience and reliability.

➤ **Cost-Effective:**

The proposed system leverages the pay-as-you-go model of Azure, meaning that you only pay for the resources used. The autoscaling feature helps to keep costs under control by only scaling up resources when necessary.

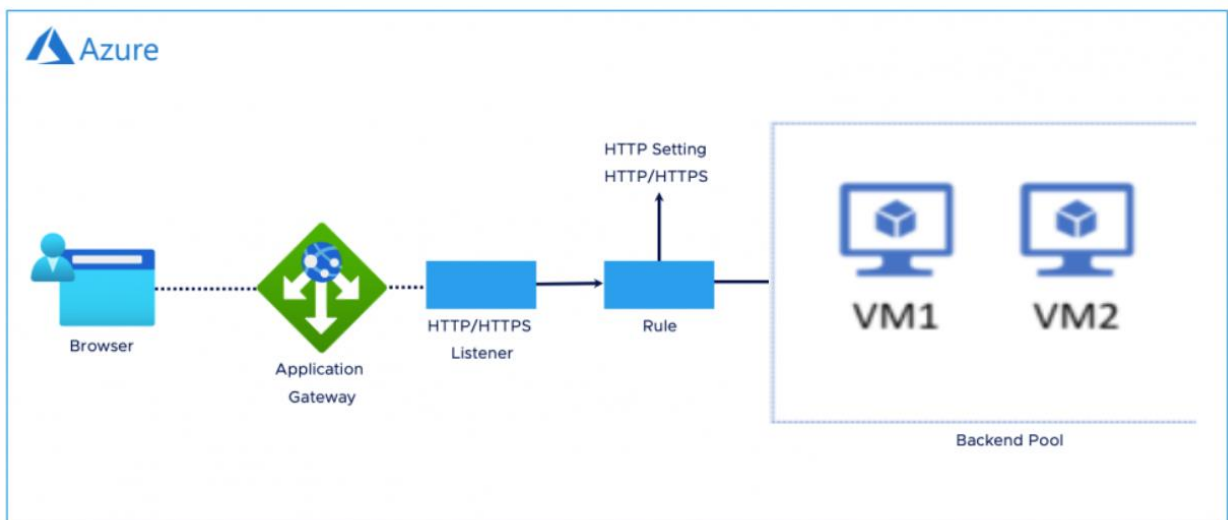


Fig 1.1 Azure Application Gateway with VM diagram

CHAPTER 2

LITERATURE REVIEW

The emergence of cloud computing has transformed the deployment and management of web applications, emphasizing the need for robust, scalable, and secure hosting environments. As web applications increasingly handle sensitive data and critical business operations, web security and performance **optimization** have become vital concerns. This literature review explores the key themes related to **secure web hosting**, focusing on the use of **HTTPS**, **SSL/TLS termination**, **load balancing**, and **application-layer traffic management** using **Microsoft Azure Application Gateway**.

[1]Web Security and the Role of HTTPS

The use of **HTTPS (Hypertext Transfer Protocol Secure)** is foundational to securing web communications. HTTPS utilizes **SSL/TLS (Secure Sockets Layer / Transport Layer Security)** protocols to encrypt data in transit, ensuring that information exchanged between users and servers remains private and tamper-proof. Research by Abdulkareem et al. (2018) illustrates how the adoption of HTTPS significantly mitigates risks related to data interception, credential theft, and identity spoofing.

Organizations like the **Electronic Frontier Foundation (EFF)** and **Google** have pushed for widespread adoption of HTTPS by flagging non-secure websites and advocating for automatic encryption. The integration of HTTPS has also become a ranking signal in search engine algorithms, as reported in studies by Moz (2019), highlighting not just the security but also the **SEO advantages** of secure hosting.

[2]SSL/TLS Termination and Load Balancer Integration

Terminating SSL at the **load balancer or application gateway level** is an established architectural best practice. According to Microsoft's official Azure documentation and third-party assessments (e.g., Rajendran et al., 2020), **SSL termination at the Azure Application Gateway** simplifies certificate management and offloads encryption processing from backend services. This allows for better performance, easier scaling, and more consistent policy enforcement.

Studies by Fang et al. (2017) further reinforce the importance of central SSL termination in distributed systems, pointing out that central management reduces the risk of expired or misconfigured certificates and allows consistent enforcement of secure cipher suites.

[3]Web Application Firewalls (WAF) in Cloud Environments

Azure Application Gateway supports integration with a **Web Application Firewall (WAF)**, which protects applications from common threats defined in the **OWASP Top 10** (e.g., XSS, SQL injection, insecure deserialization). Research by Alqahtani et al. (2021) confirms that WAFs, when properly configured, can significantly reduce vulnerabilities in cloud-hosted web applications. Azure's WAF is built on **ModSecurity rules**, which are customizable and continuously updated to address new threats.

This integration is especially important in **multi-tenant cloud environments**, where applications face a broader range of attack vectors. Studies have shown that the combination of HTTPS with a WAF provides a **defense-in-depth strategy**, aligning with cybersecurity frameworks such as **NIST SP 800-53** and **ISO/IEC 27001**.

[4]Application Gateway Features and Intelligent Traffic Management

Azure Application Gateway provides advanced **Layer 7 routing capabilities**, which include path-based routing, multi-site hosting, and session affinity using cookies. Research by Microsoft and third-party evaluations (e.g., Forrester Wave Report, 2021) emphasize the benefit of **context-aware routing**, which allows web traffic to be routed based on application logic rather than static IP addresses.

This feature supports **microservices architectures**, where different backend pools can handle different parts of the application based on URL paths. The intelligent routing capabilities not only improve performance and user experience but also facilitate **better resource utilization and fault isolation** — concepts supported in studies on service-oriented architecture (SOA) and microservices by Dragoni et al. (2018).

[5]Scalability, Monitoring, and Automation in Azure

Microsoft Azure supports **Autoscaling and diagnostics** for Application Gateway, enabling dynamic resource allocation and performance optimization. Research by Patel & Shah (2019) explains how autoscaling reduces downtime during traffic surges, contributing to improved Service Level Agreements (SLAs). Azure Monitor and Log Analytics further provide visibility into traffic patterns, latency, and security threats, enabling proactive tuning of the infrastructure.

These features align with **DevOps and Site Reliability Engineering (SRE)** practices that emphasize continuous monitoring and rapid iteration. As Gartner (2022) notes in its cloud trends analysis, observability and automation are key enablers of **resilient and secure cloud architectures**.

[6]Gaps and Opportunities

Despite the maturity of Azure's services, many organizations and developers fail to implement security best practices effectively. Studies by OWASP (2023) show that misconfigured HTTPS settings, expired certificates, or disabled WAF rules still contribute to a large number of breaches. This project aims to address these gaps by demonstrating a **complete, correctly configured setup** for secure web hosting using **Azure Application Gateway**, HTTPS with SSL termination, and WAF protection — following best practices and real-world deployment strategies.

[7]Conclusion of Literature Review

The reviewed literature confirms that **secure web hosting using HTTPS and cloud-native services like Azure Application Gateway** is both a practical and necessary approach for modern application deployment. By incorporating centralized SSL termination, intelligent routing, WAF protection, and scalability features, Azure provides a comprehensive solution for hosting secure and high-performing web applications. This project contributes to the existing body of work by implementing a hands-on, end-to-end solution that aligns with academic research, industry standards, and cloud architecture best practices. focusing on ensuring Internet security through the utilization of Virtual Private Networks (VPNs).

With the widespread adoption of the Internet, it has become an indispensable tool for working, learning, and conducting various activities anytime and anywhere. While the Internet offers unparalleled convenience, it also poses significant risks to personal data security due to its open nature. The main function of VPNs is to establish a private network within the public network and encrypt communication to enhance security. VPNs achieve this by creating a

network tunnel in the public network using encryption technology, thereby ensuring safe data transmission and preventing unauthorized access. **"Research on network security of VPN technology," 2020 International Conference on Information Science and Education (ICISE- IE), Sanya, China, 2020, pp. 539-542, doi: 10.1109/ICISE51755.2020.00121.**

Advantages: One prominent type of VPN is IPsec VPN, which utilizes the IPsec protocol to enable remote access. It establishes a secure tunnel over a public network, allowing two private networks to transmit data securely. IPsec VPN offers advantages such as multi-level networking, making it suitable for inter- institutional networking. Additionally, it operates in a fixed networking mode, providing transparent access to users without requiring them to log in.

Disadvantages: Despite its benefits, utilizing VPNs, including IPsec VPN, presents both advantages and disadvantages, shaping its suitability for various network security needs. Further exploration and understanding of VPN technologies are crucial for effectively safeguarding network resources and ensuring the privacy and integrity of data transmitted over the Internet.

[8] Tim Lackorzynski; Stefan Köpsell; Thorsten Strufe proposed the investigation and comparison of various Virtual Private Network (VPN) solutions to address the evolving needs of future industrial networks. Unlike traditional approaches, future industrial networks will grow from existing installations without displacing legacy components, emphasizing the secure integration of these components as a crucial building block for Industry 4.0 realization. Secure and high-performance VPNs are identified as essential tools for achieving this integration seamlessly. The investigation involved testing the performance of different VPN solutions on multiple hardware platforms, ranging from resource-

constrained to powerful systems. Non-functional aspects such as security, manageability, and ease of use were thoroughly discussed to assess their suitability for future industrial use cases. **"A Comparative Study on Virtual Private Networks for Future Industrial Communication Systems," 2019 15th IEEE International Workshop on Factory Communication Systems (WFCS), Sundsvall, Sweden, 2019, pp. 1-8, doi: 10.1109/WFCS.2019.8758010.**

Advantages: The investigation and comparison of various VPN solutions provide industry stakeholders with valuable insights into selecting the most suitable options for future industrial setups. By testing the performance of VPN solutions on multiple hardware platforms, stakeholders gain an understanding of their scalability and compatibility with different infrastructure requirements.

Disadvantages: The process of investigating and comparing VPN solutions may require significant time and resources, particularly when evaluating performance across various hardware platforms. Identifying and testing VPN solutions for future industrial setups necessitates a comprehensive understanding of industry-specific requirements and challenges, which may pose complexities during the evaluation process.

[9] Yinhai Xie; Changliang Zhang; Xingting He; Juan Tian proposed the study and implementation of VPN access technology and remote secure desktops to meet the growing demand for mobile office remote access in meteorological departments. This paper aims to integrate existing remote access methods and optimize the mobile office access platform to enhance efficiency and security. **"Application research of meteorological virtual private network security remote access technology," 2023 IEEE 6th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 2023, pp. 423-426, doi: 10.1109/ITNEC56291.2023.10082333.**

Advantages: The implementation of VPN access technology and remote secure desktops enhances the accessibility and flexibility of meteorological department resources, enabling mobile office remote access regardless of location. Integrating existing remote access methods and optimizing the mobile office access platform streamlines operations within meteorological departments, leading to increased efficiency and productivity. By providing a secure virtual working environment through secure desktops based on SSL VPN, sensitive data within meteorological departments is protected, mitigating risks associated with data leakage, transmission, and unauthorized access.

Disadvantages: The implementation of VPN access technology and remote secure desktops requires thorough analysis of specific needs and network structures within meteorological services, potentially leads to complexities and challenges in tailoring the solution to meet organizational requirements. Establishing a special security zone for meteorology and dividing rights and privileges necessitates meticulous planning and execution, which may involve significant time and resource investments. While VPN access technology and remote secure desktops offer enhanced security measures, there may still be vulnerabilities and risks associated with remote access, requiring continuous monitoring and updates to mitigate potential threats.

[10] **S. Tongkaw; A. Tongkaw** delve into the realm of Multi-VLAN design and implementation, particularly within the context of two campuses of Songkhla Rajabhat University. This research underscores the significance of Multi-VLANs in providing secure and efficient network management, especially as diverse applications such as VoIP, Wireless Access Points, LAN, Server farms, and IoT devices are increasingly integrated into networks. The study highlights three key benefits of Multi-VLAN design: cost-effectiveness in deploying various applications, heightened protection and

security measures, and a reduction in network administrators' workload in maintaining and managing the network efficiency, even across distant campuses. **"Multi-VLAN Design over IPSec VPN for Campus Network," 2018 IEEE Conference on Wireless Sensors (ICWiSe), Langkawi, Malaysia, 2018, pp. 66-71, doi: 10.1109/ICWISE.2018.8633293.**

Advantages: The emphasis on increased protection and security demonstrates how Multi-VLANs can enhance data confidentiality and integrity, crucial in environments handling sensitive information. The reduction in network administrators' workload is another significant advantage, illustrating how Multi-VLANs streamline network management across geographically dispersed locations, ensuring data transmission efficiency even in distant campuses. This research thus provides a compelling case for institutions seeking robust and efficient network infrastructure solutions.

Disadvantages: While the study presents compelling advantages, it may not address the potential challenges or limitations of Multi-VLAN implementations. Organizations should carefully consider their specific requirements and infrastructure before adopting Multi-VLAN designs. Depending on the scale and complexity of the network, implementing Multi-VLANs could introduce complexities in configuration and maintenance, requiring specialized knowledge and training for network administrators.

CHAPTER 3

THEORETICAL BACKGROUND

3.1 IMPLEMENTATION ENVIRONMENT

- Virtual Network
- Subnets
- Virtual Machine
- Application Gateway
- Web App
- IIS Installation in VM
- SNI-SSL Certificate
- Web Application Firewall.

3.1.1 Azure Virtual Network

Azure Virtual Network (VNet) is a logically isolated network in the Microsoft Azure cloud. It enables you to securely connect Azure resources (like virtual machines, databases, and applications) to each other, the internet, and your on-premises networks.

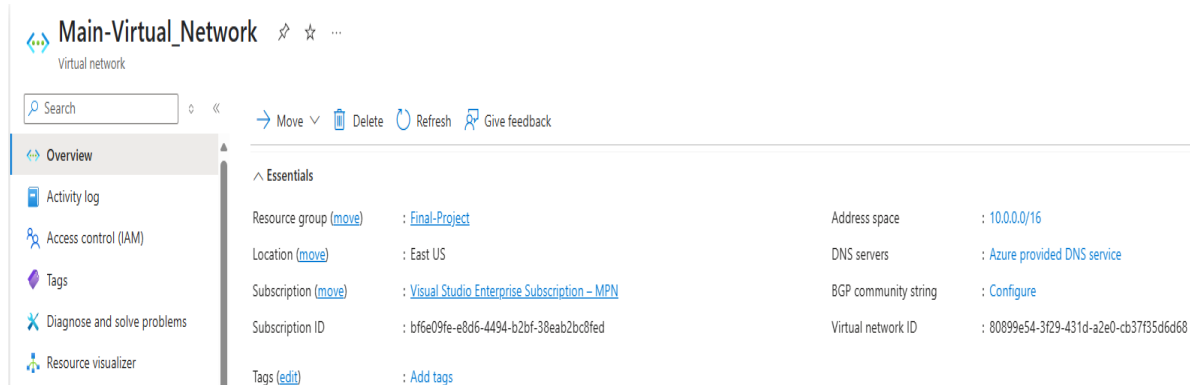


Fig 3.1.1 Diagram of Vnet

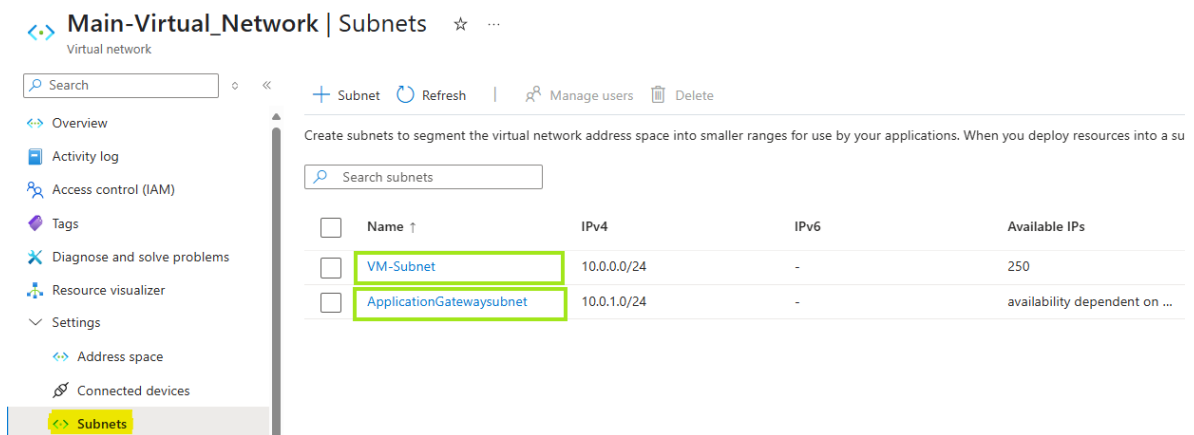


Fig 3.1.2 Diagram of Subnets

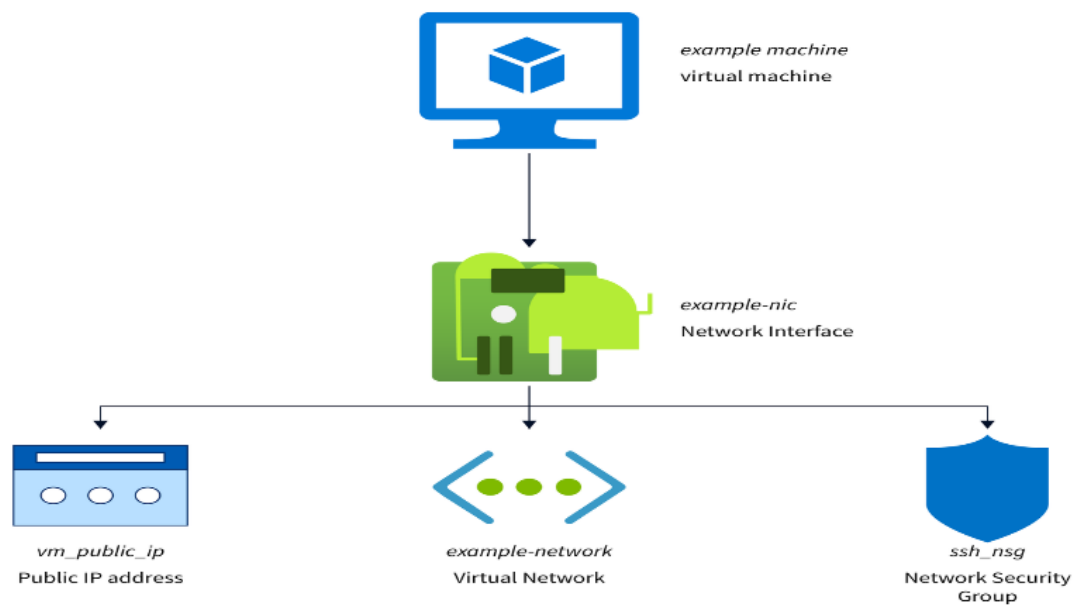


Fig 3.1.3 Virtual Machine Diagram

3.2 IIS INSTALLATION USING POWERSHELL

To install **IIS (Internet Information Services)** on a **Windows Virtual Machine (VM)** using **PowerShell**, these commands can be run directly on the VM.

In Virtual machine *overview* – *Operations* - *Run Command* – *RunPowershellscript*.

```
“ import-module servermanager  
add-windowsfeature web-server -includeallsubfeature”
```

WEB-VM-1 | Run command

Virtual machine

run

Now it's possible to execute multiple scripts at the same time, manage their progress and persist execution state, now available through Azure CLI and PowerShell.

Run Command uses the VM agent to let you run a script inside this virtual machine. This can be helpful for tasks like installing software, configuring the system, or running diagnostics. Select a command below to see details.

Name	Description
RunPowerShellScript	Executes a PowerShell script
DisableNLA	Disable Network Location Awareness
DisableWindowsUpdate	Disable Windows Update
EnableAdminAccount	Enable administrator account
EnableEMS	Enable Event Management System
EnableRemotePS	Enable remote PowerShell
EnableWindowsUpdate	Enable Windows Update
IPConfig	List IP configurations
RDPSettings	Verify RDP List
ResetRDPAuth	Reset RDP Authentication
SetRDPPort	Set Remote Desktop Port

RunPowerShellScript

Script execution complete

PowerShell Script

```
1 import-module servermanager  
2 add-windowsfeature web-server -includeallsubfeature
```

Run

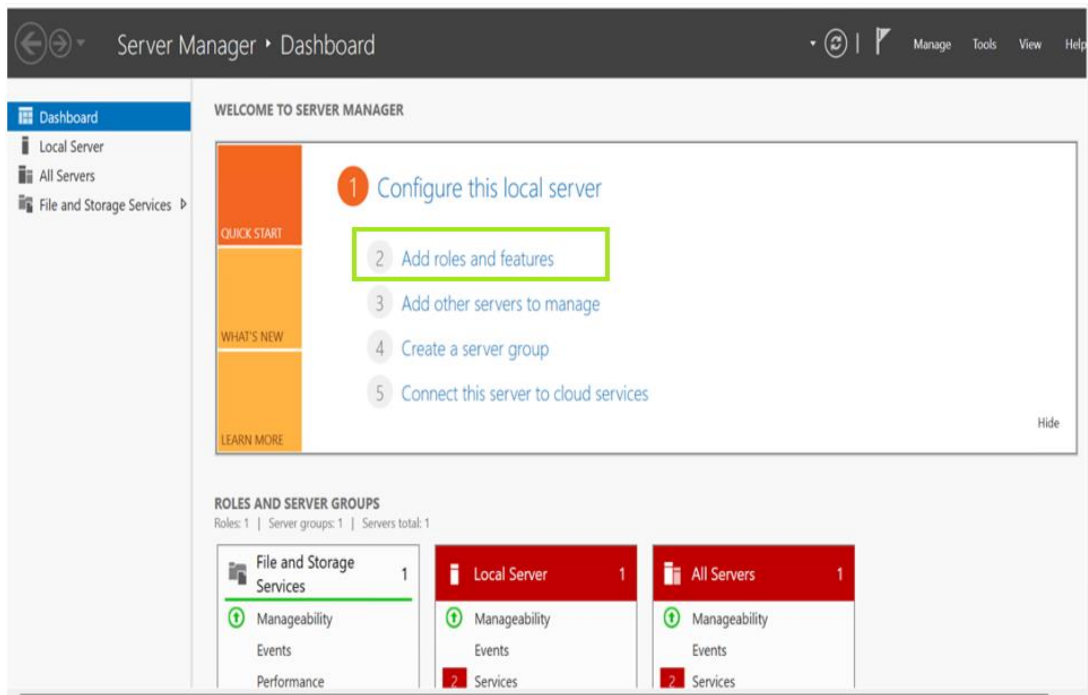
Output

Success	Restart Needed	Exit Code	Feature Result
True	No	Success	(ASP.NET 4.7, .NET Framework 3.5 (includes...)

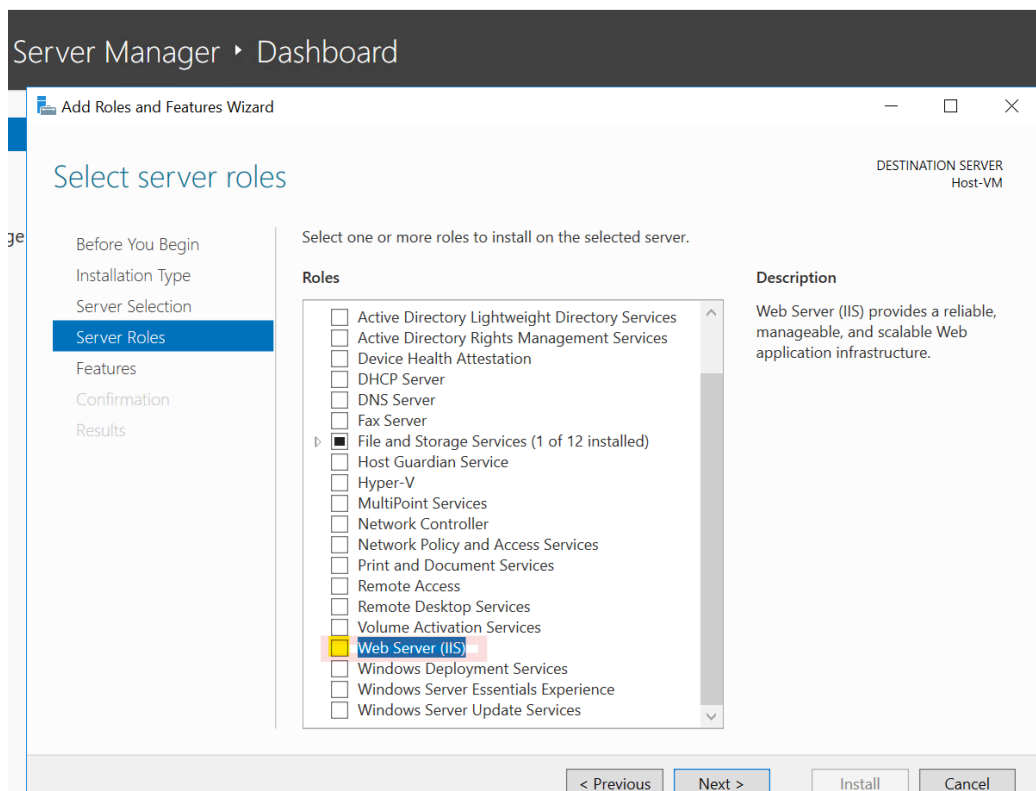
Fig 3.2 IIS Installation in VM

3.3 IIS INSTALLATION USING SERVER MANAGER

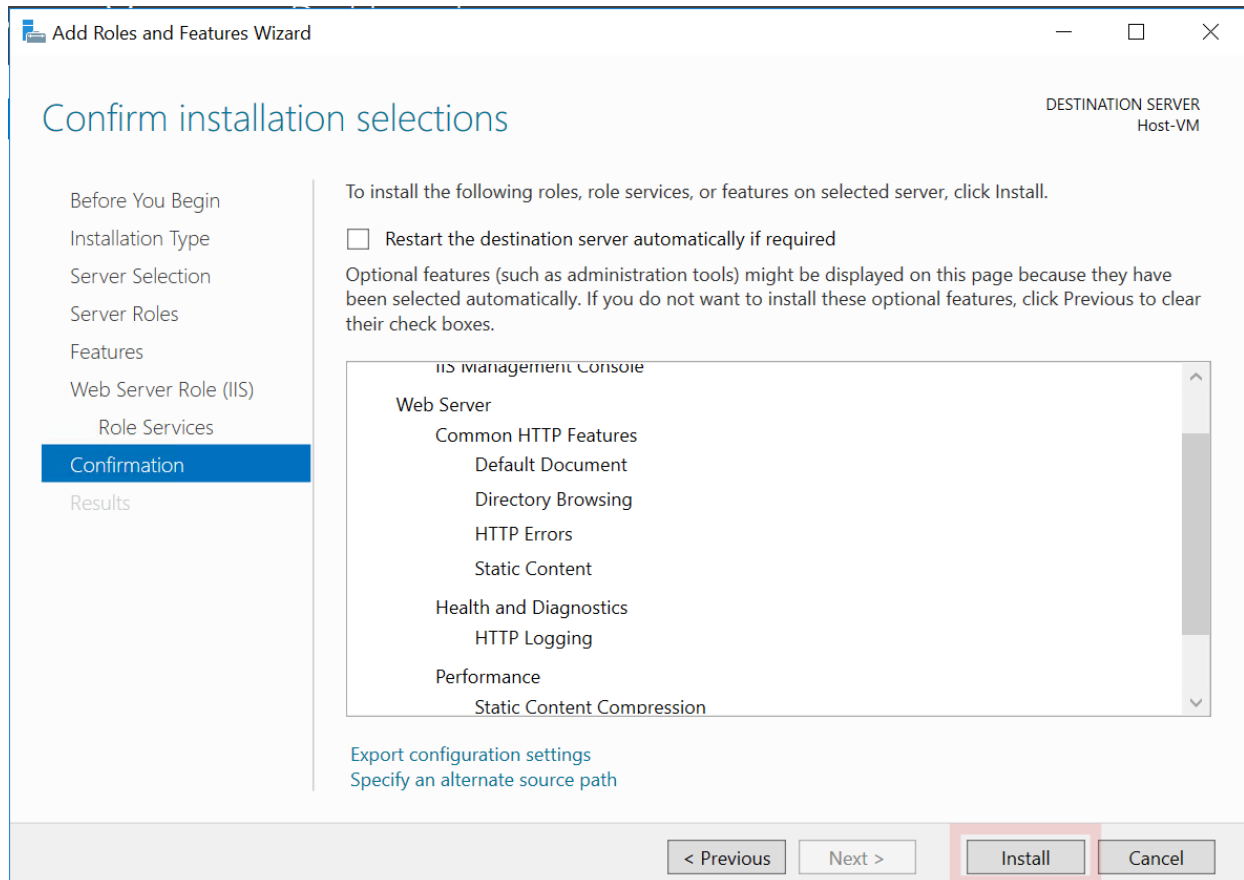
To install **IIS (Internet Information Services)** on a **Windows Virtual Machine (VM)** using server manager.



- Select Server Manager – Select Add roles and Features



- In Server Roles – Select and Enable Web Server (IIS) features



- In Confirmation – Select Install to finalize the Web IIS installation

3.4 CONNECTING VIRTUAL MACHINE TO APPLICATION GATEWAY

Step 1: Identify/Configure the Virtual Network

- Ensure the **VM** and **Application Gateway** are in the **same VNet** or connected through **VNet peering**.
- VM should have a **private IP** address from a subnet that the Application Gateway can reach.

Step 2: Configure the Backend Pool

- **Go to the Azure Portal.**
- Navigate to **Application Gateway** → your App Gateway.
- Under **Settings**, click on **Backend pools**.
- Click + **Add** to create a new backend pool.
 - **Name:** Give it a name (e.g., Final-backendPool).
 - **Target type:** Select **IP address or FQDN**.

- Under **Backend targets**, add the **private IP address** of your VM.
- Click **Add** to save the backend pool.

Step 3: Create or Configure a HTTP/HTTPS Listener

- Under **Settings**, go to **Listeners**.
- Click + **Add** to create a new listener or use an existing one.
 - **Name:** Give it a name (e.g., app-listener).
 - Choose **HTTP** or **HTTPS** depending on your configuration.
 - Assign a frontend IP (public or private).
 - For HTTPS, upload your **SSL certificate (PFX)** and provide the password.

Step 4: Configure Rules to Route Traffic

- Go to **Rules** under Application Gateway.
- Click + **Add** or edit an existing rule.
- Set the following:
 - **Name:** Give it a name (e.g., app-routingrule).
 - **Listener:** Select the one you just created.
 - **Backend target:** Choose the **backend pool** you just set up.
 - **HTTP settings:** Create or select HTTP settings (used to forward traffic to the VM).

Step 5: Configure HTTP Settings

- Go to **HTTP settings** under Application Gateway.
- Click + **Add** to create a new setting.
 - **Name:** Give it a name (e.g., app-backend-setting).
 - **Port:** Set to 80 (or 443 for HTTPS to backend).
 - **Backend protocol:** HTTP or HTTPS.
 - Enable **Use custom probe** if using a health probe (recommended).
 - Optionally enable **Cookie-based affinity**.
- Save the settings.

Edit backend pool ...

A backend pool is a collection of resources to which your application gateway can send traffic. A backend pool can contain virtual machines, virtual machines scale sets, IP addresses, domain names, or an App Service.

Name

final-backendpool

Add backend pool without targets

Yes

No

Backend targets

1 item

Target type	Target	
Virtual machine	web-vm-1381	...
IP address or FQDN		

Associated rule

app-routingrule

Fig 3.4.1 Adding VM in the application gateway backendpool

finalappGW | Backend settings

Search

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Resource visualizer

Settings

Configuration

Web application firewall

Backend pools

Backend settings

Frontend IP configurations

Private link

SSL settings

Listeners

Rules

Rewrites

Health probes

Properties

Locks

Monitoring

Automation

Help

Backend settings

Search Backend settings

Name	Port
app-backend-setting	80

Backend settings name

app-backend-setting

Backend protocol

☒ HTTP ☐ HTTPS

Backend port *

80

Additional settings

Cookie-based affinity ☐ Enable ☒ Disable

Connection draining ☐ Enable ☒ Disable

Request time-out (seconds) *

20

Override backend path

Host name

By default, the Application Gateway sends the same HTTP host header to the backend as it receives from the client. If your backend application/service requires a specific host value, you can override it using this setting.

Override with new host name

☐ Yes ☒ No

Use custom probe ☐ Yes ☒ No

Save **Cancel**

Configuring route via HTTP-80 port

Save to create the 80 port backend setting

Fig 3.4.2 Adding Backend Setting in Application Gateway

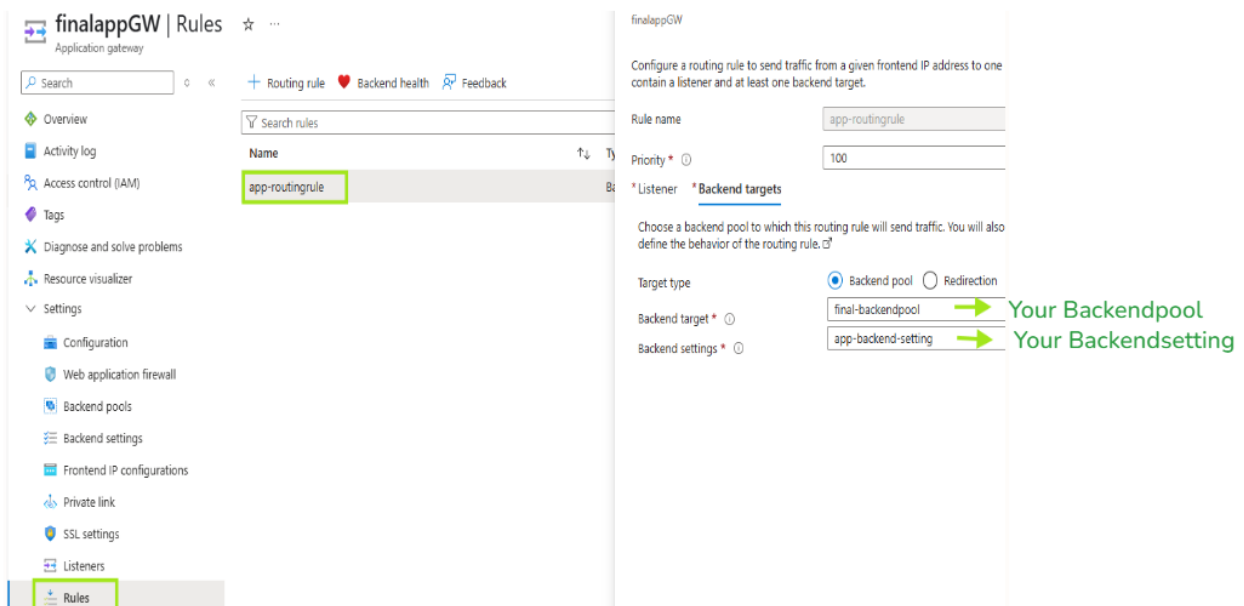


Fig 3.4.3 Adding Rule in Application Gateway



Fig 3.4.4 Application Gateway with VM Output


3.5 WEB APPLICATION PROVISION

[Home](#) > [App Services](#) >


Create Web App

Basics Database Deployment Networking Monitor + secure Tags Review + create

Summary

 **Web App**
by Microsoft

Basic (B1) sku
Estimated price - 1032.45 INR/Month

 Basic authentication for this app is currently disabled and may impact deployments. [Click to learn more.](#)

Details

Subscription	bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed
Resource Group	Final-Project
Name	projectapp
Secure unique default hostname	Enabled
Publish	Code
Runtime stack	Java 17
Java web server stack	Java SE (Embedded Web Server)

App Service Plan (New)

Name	ASP-FinalProject
Operating System	Linux
Region	East US
SKU	Basic
Size	Small
ACU	100 total ACU
Memory	1.75 GB memory

Monitor + secure

Application Insights	Not enabled
----------------------	-------------

Deployment

Basic authentication	Disabled
----------------------	----------

[Create](#) [< Previous](#) [Next >](#) [Download a template for automation](#)

Fig 3.5 Creation of Web application

Azure Application Gateway is a web traffic load balancer that enables you to manage traffic to your web applications. You can use it with an Azure Web App (App Service) as a backend, allowing for advanced routing, SSL termination, and security features.

3.6 MODULE DESIGN

3.6.1 Sequence Diagram

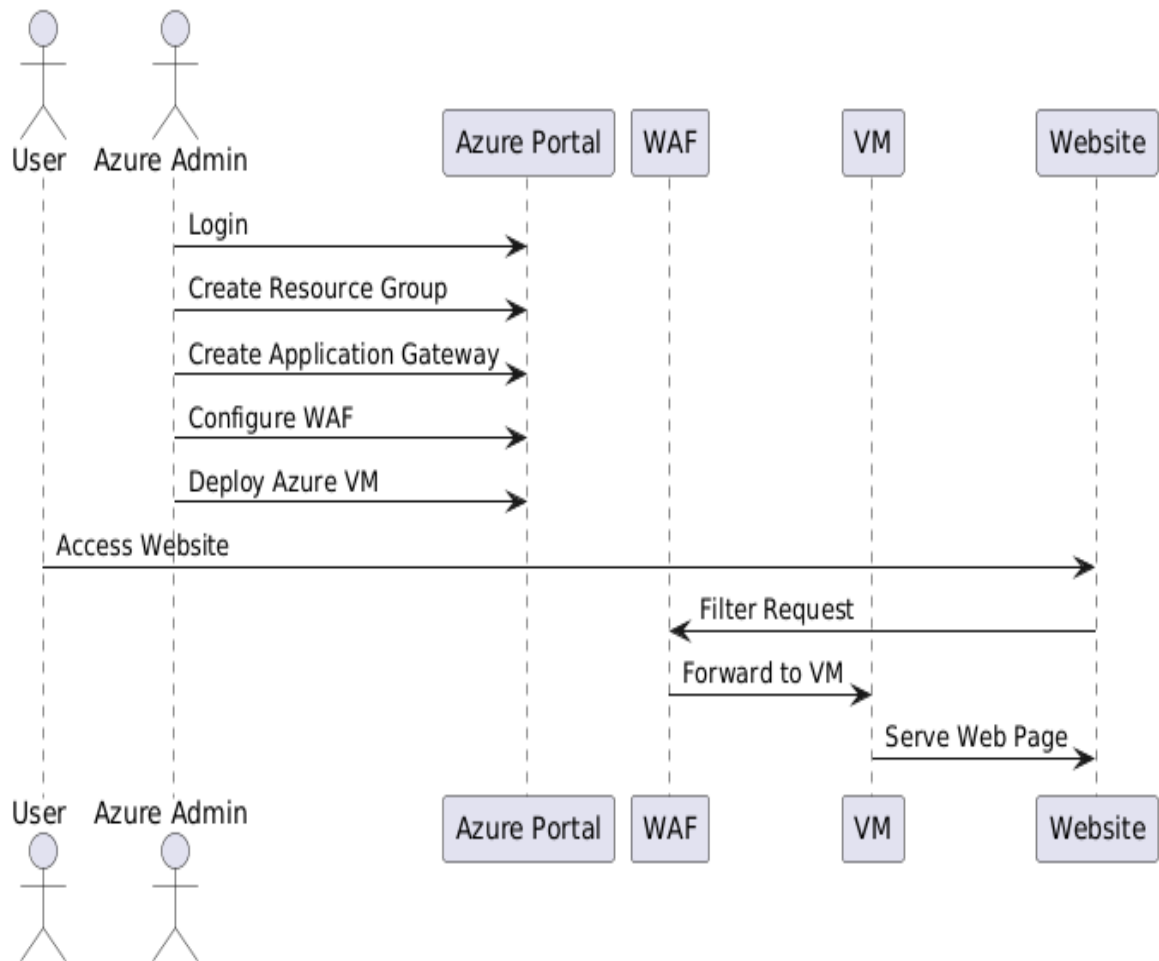


Fig 3.6.1 Sequence Diagram

The Sequence diagram outlines the chronological order of events in the system. It begins with the Admin logging into the Azure Portal, followed by the creation of a Resource Group, setup of the Application Gateway, configuration of the WAF, and deployment of the VM. After deployment, the User sends a request to the Application Gateway, which passes through the WAF for inspection before reaching the VM. The VM then serves the response back to the user. This linear and time-based sequence ensures traceable and secure communication flow across the architecture.

3.6.2 Class Diagram

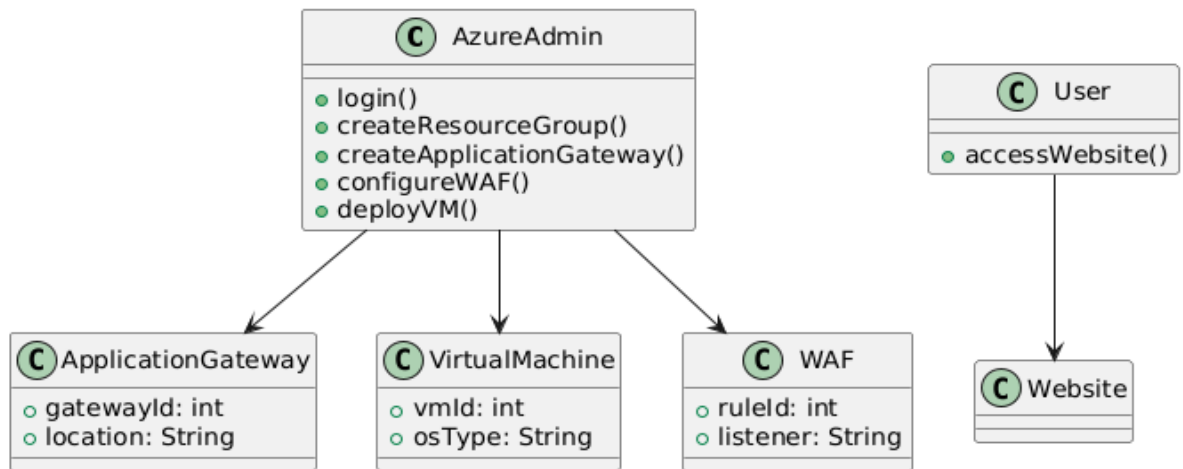


Fig 3.6.2 Class Diagram

The Class diagram structurally represents the components involved in the system. Key classes include Admin, AzurePortal, ResourceGroup, ApplicationGateway, WAF, VM, and User. Each class has specific responsibilities — for example, AzurePortal contains methods for login and resource creation; ApplicationGateway and WAF focus on traffic management and security. This diagram highlights the relationships and dependencies between core entities in a cloud infrastructure setup.

3.6.3 Use Case Diagram

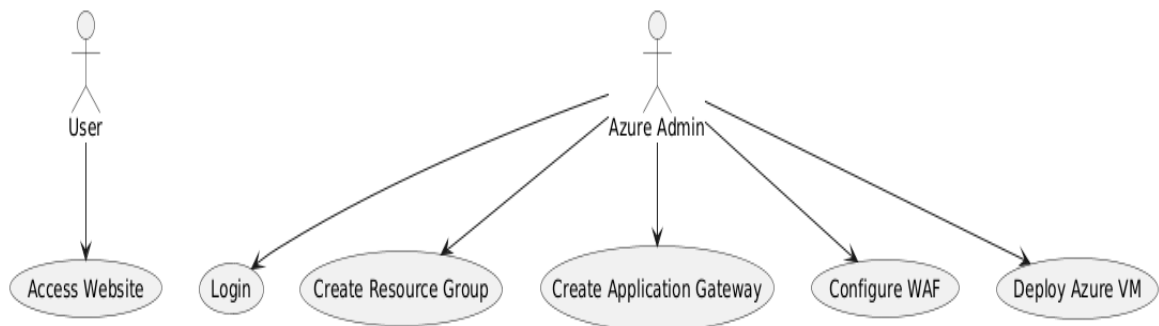


Fig 3.6.3 Use Case Diagram

The Use Case diagram captures the high-level interactions between the system and its actors. The Admin actor interacts with the Azure Portal to perform tasks like logging in, creating a Resource Group, deploying an Application Gateway, attaching a WAF, and launching a VM. The User actor interacts with the deployed infrastructure by accessing the web application through the Application Gateway. This shows clear separation of administrative and end-user roles, reinforcing best practices in cloud operations and security.

3.6.4 Activity Diagram

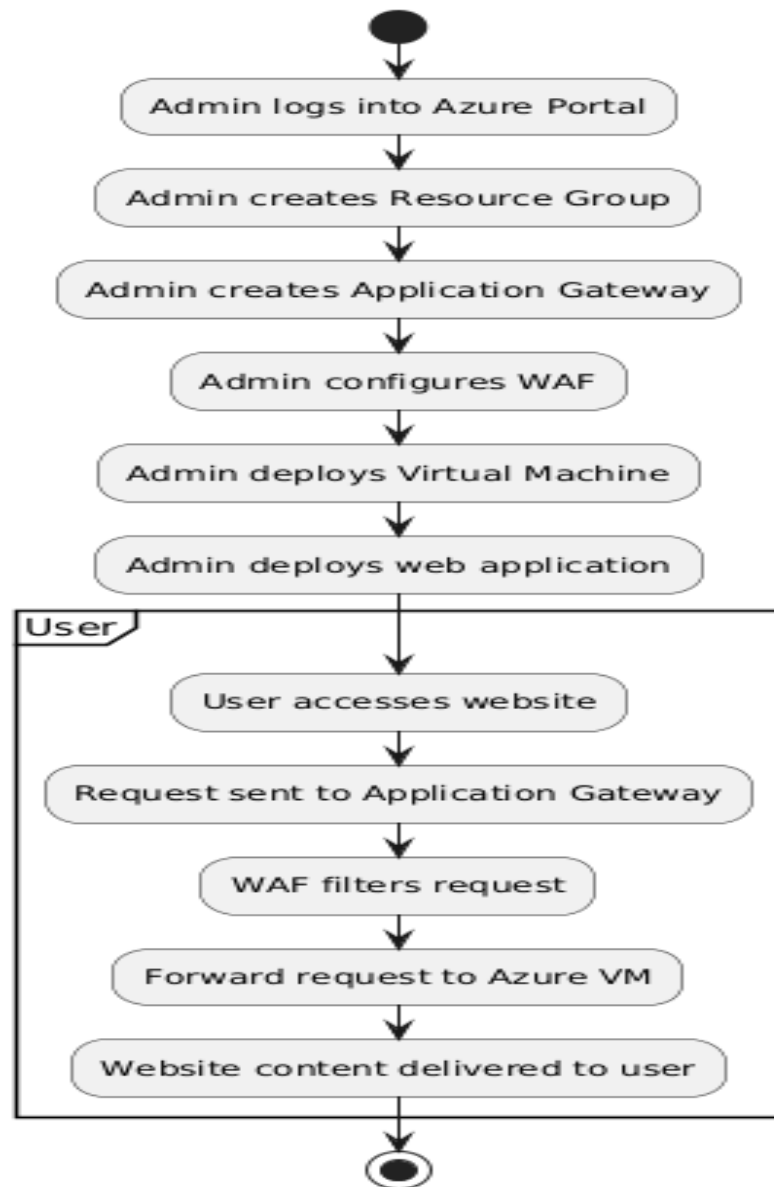


Fig 3.6.4 Activity Diagram

The Activity diagram represents the dynamic workflow of deploying and accessing the Azure-based infrastructure. It starts with the Admin logging in to the Azure Portal, followed by sequential actions such as creating a Resource Group, deploying the Application Gateway, configuring the WAF, and launching the VM. Once setup is complete, the User accesses the application through the secure gateway. This diagram clearly maps the step-by-step flow of tasks, ensuring visibility into the end-to-end deployment and access process.

3.6.5 Collaboration Diagram

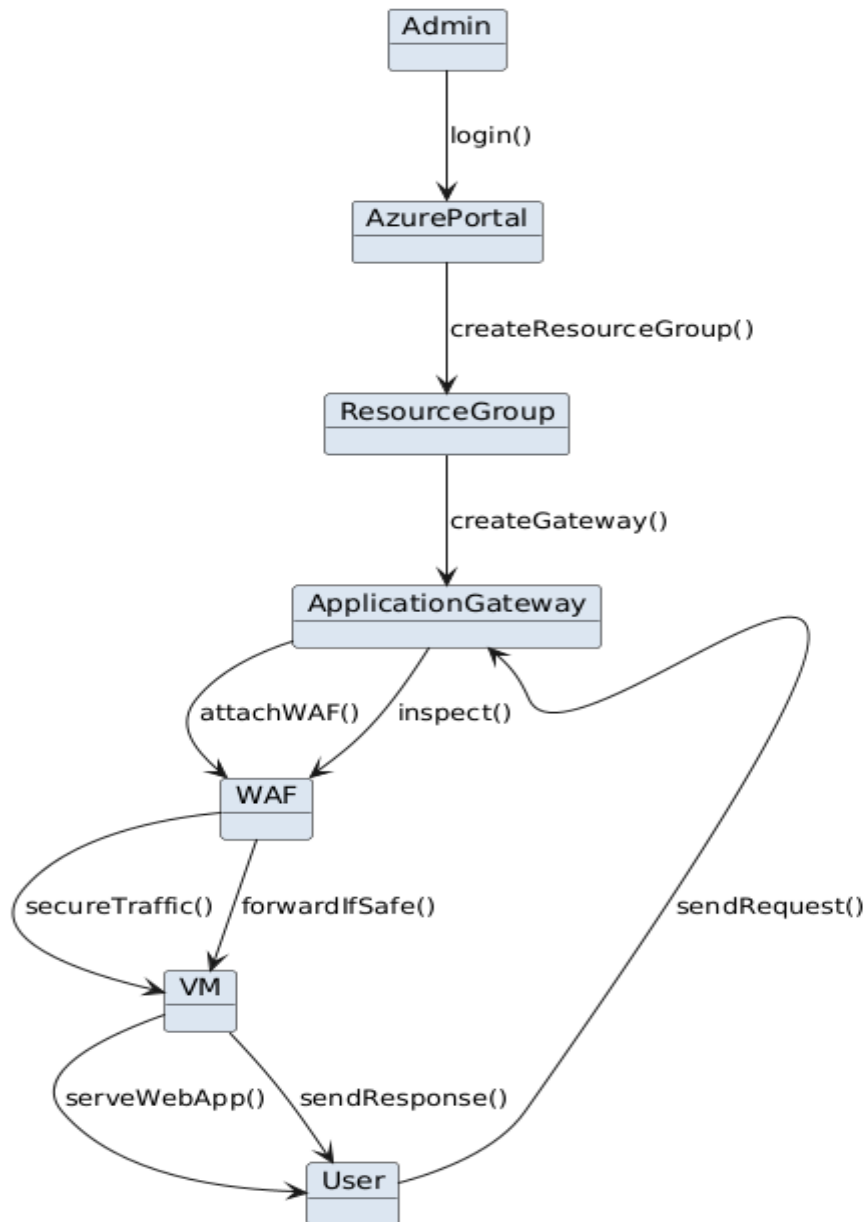


Fig 3.6.5 Collaboration Diagram

The Collaboration diagram focuses on the structural interaction between system components. It emphasizes how objects like Admin, Azure Portal, Application Gateway, WAF, and VM communicate with one another to complete the deployment and access lifecycle. Each interaction is numbered to reflect the execution order, showcasing message passing and object roles in a real-time environment. This diagram validates the logical coordination and reliability of the overall system architecture.

3.7 CSS - Index.html file

By using the advance editor tool in web app, the existing app page has been modified with Java script.

```
searchForm = document.querySelector('.search-form');

document.querySelector('#search-btn').onclick = () =>{
    searchForm.classList.toggle('active');
}

let loginForm = document.querySelector('.login-form-container');

document.querySelector('#login-btn').onclick = () =>{
    loginForm.classList.toggle('active');
}

document.querySelector('#close-login-btn').onclick = () =>{
    loginForm.classList.remove('active');
}

window.onscroll = () =>{

    searchForm.classList.remove('active');

    if(window.scrollY > 80){
        document.querySelector('.header .header-2').classList.add('active');
    }else{
        document.querySelector('.header .header-2').classList.remove('active');
    }
}

window.onload = () =>{

    if(window.scrollY > 80){
        document.querySelector('.header .header-2').classList.add('active');
    }else{
        document.querySelector('.header .header-2').classList.remove('active');
    }

    fadeOut();
}

function loader(){
    document.querySelector('.loader-container').classList.add('active');
}
```

```

function fadeOut(){
    setTimeout(loader, 4000);
}

var swiper = new Swiper(".books-slider", {
    loop:true,
    centeredSlides: true,
    autoplay: {
        delay: 9500,
        disableOnInteraction: false,
    },
    breakpoints: {
        0: {
            slidesPerView: 1,
        },
        768: {
            slidesPerView: 2,
        },
        1024: {
            slidesPerView: 3,
        },
    },
});

var swiper = new Swiper(".featured-slider", {
    spaceBetween: 10,
    loop:true,
    centeredSlides: true,
    autoplay: {
        delay: 9500,
        disableOnInteraction: false,
    },
    navigation: {
        nextEl: ".swiper-button-next",
        prevEl: ".swiper-button-prev",
    },
    breakpoints: {
        0: {
            slidesPerView: 1,
        },
        450: {
            slidesPerView: 2,
        },
        768: {
            slidesPerView: 3,
        },
        1024: {
            slidesPerView: 4,
        },
    },
});

```

```

    },
  });

var swiper = new Swiper(".arrivals-slider", {
  spaceBetween: 10,
  loop:true,
  centeredSlides: true,
  autoplay: {
    delay: 9500,
    disableOnInteraction: false,
  },
  breakpoints: {
    0: {
      slidesPerView: 1,
    },
    768: {
      slidesPerView: 2,
    },
    1024: {
      slidesPerView: 3,
    },
  },
});

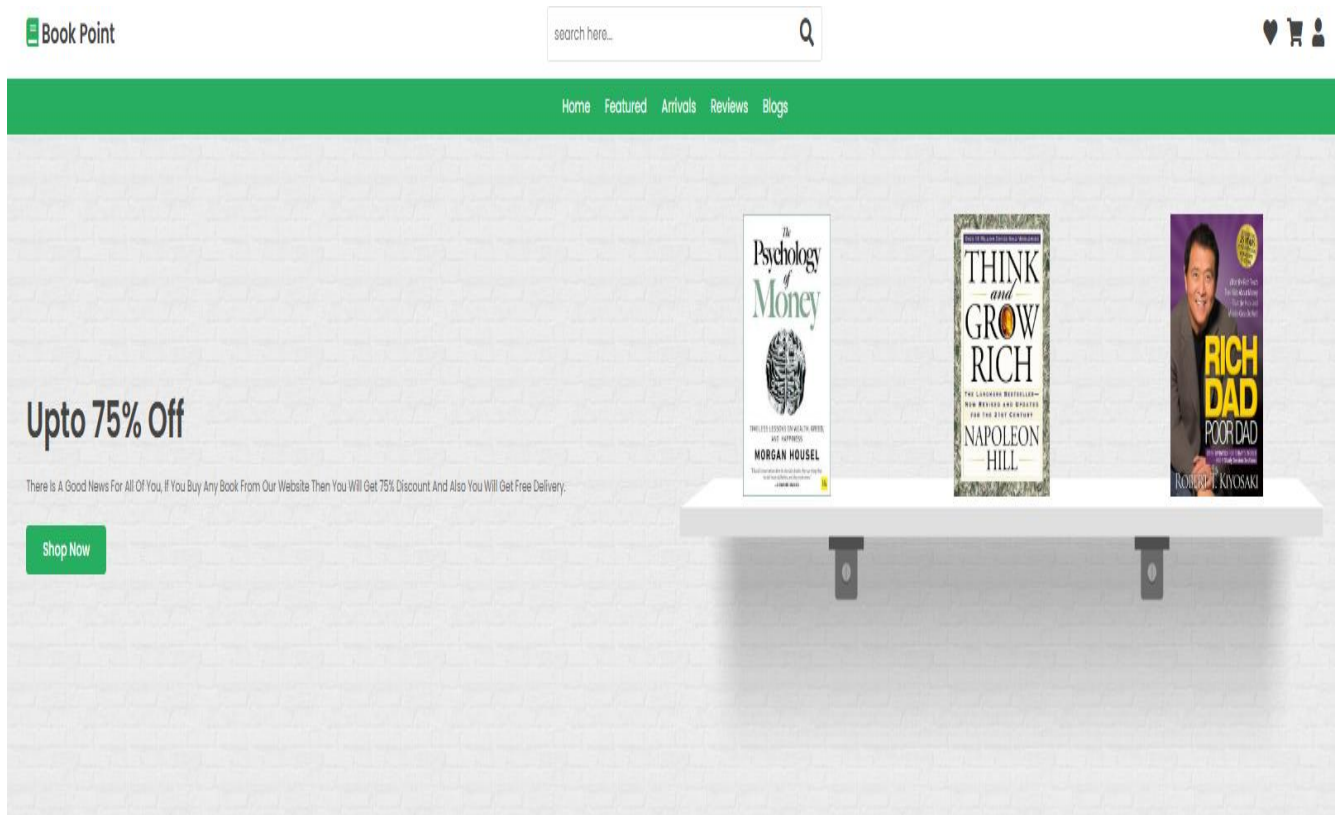
var swiper = new Swiper(".reviews-slider", {
  spaceBetween: 10,
  grabCursor:true,
  loop:true,
  centeredSlides: true,
  autoplay: {
    delay: 9500,
    disableOnInteraction: false,
  },
  breakpoints: {
    0: {
      slidesPerView: 1,
    },
    768: {
      slidesPerView: 2,
    },
    1024: {
      slidesPerView: 3,
    },
  },
});

var swiper = new Swiper(".blogs-slider", {
  spaceBetween: 10,
  grabCursor:true,
  loop:true,

```

```
centeredSlides: true,  
autoplay: {  
  delay: 9500,  
  disableOnInteraction: false,  
},  
breakpoints: {  
  0: {  
    slidesPerView: 1,  
  },  
  768: {  
    slidesPerView: 2,  
  },  
  1024: {  
    slidesPerView: 3,  
  },  
},  
});
```


3.7.1 WEB APPLICATION OUTPUT



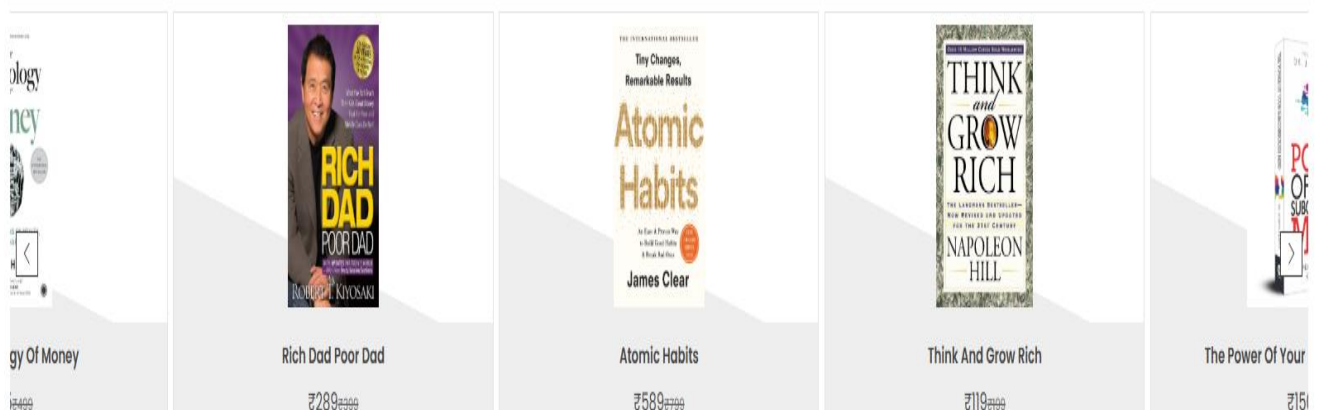
 **Free Shipping**
Order Over ₹200

 **Secure Payment**
100 Secure Payment

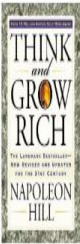
 **Easy Returns**
10 Days Returns

 **24/7 Support**
Call Us Anytime

Featured Books




New Arrivals




Think And Grow Rich

₹149 ₹199



Ikigai

₹345 ₹550



The Power Of Your Subconscious Mind

₹110 ₹175

Deal Of The Day

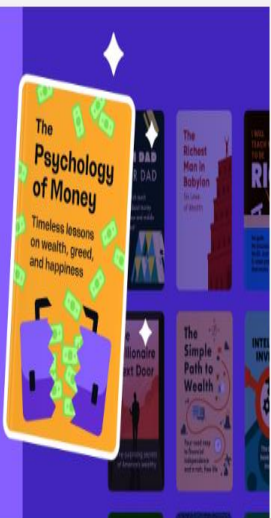
Upto 50% Off

Good News For You, If You Buy Any Book On Saturday And Sunday You Will Get 50% Off And Also You Will Get Free Delivery.

Shop Now

Books like

The Psychology of Money



Client's Reviews

Fig 3.7.1 Web application output

3.8 APPLICATION GATEWAY WITH WEB APP BACKENDPOOL

The Connecting a **web app (such as an Azure App Service)** with an **Azure Application Gateway (AGW)** as a backend is a common architecture for enabling advanced Layer 7 load balancing, SSL termination, Web Application Firewall (WAF), and other security features.

[Home](#) > [Load balancing | Application Gateway](#) > [finalappGW | Backend pools](#) >

Edit backend pool ...

A backend pool is a collection of resources to which your application gateway can send traffic. A backend pool can contain virtual machines, virtual machines scale sets, IP addresses, domain names, or an App Service.

Name

WEBAPPBACKEND

Add backend pool without targets

Yes ☒ No

Backend targets

1 item

Target type	Target
App Services	project-webapp
IP address or FQDN	

i While using a multitenant Azure service (such as App Service, Azure Spring Apps, Power Apps portal) as a backend target for application gateway, it is recommended to

- Register every hostname of the incoming client request as a custom domain on that backend service
- Not override the hostname in the Backend Settings of your application gateway
- Configure the DNS of the custom domain to point to your application gateway

Visit the [documentation](#) to know more about the configuration guidance.

Associated rule

[Rule-100](#)

Fig 3.8 AppGateway with Web app backendpool

3.8.1 DDOS ATTACK

A **DDoS (Distributed Denial of Service) attack** is a type of cyberattack where multiple systems, often compromised and controlled by a single attacker, are used to flood a target—such as a website, server, or network—with an overwhelming amount of traffic. The goal is to exhaust the target's resources, such as bandwidth, memory, or processing power, making it slow, unresponsive, or completely unavailable to legitimate users. These attacks are often carried out using a network of infected computers and devices known as a **botnet**, which the attacker controls remotely.

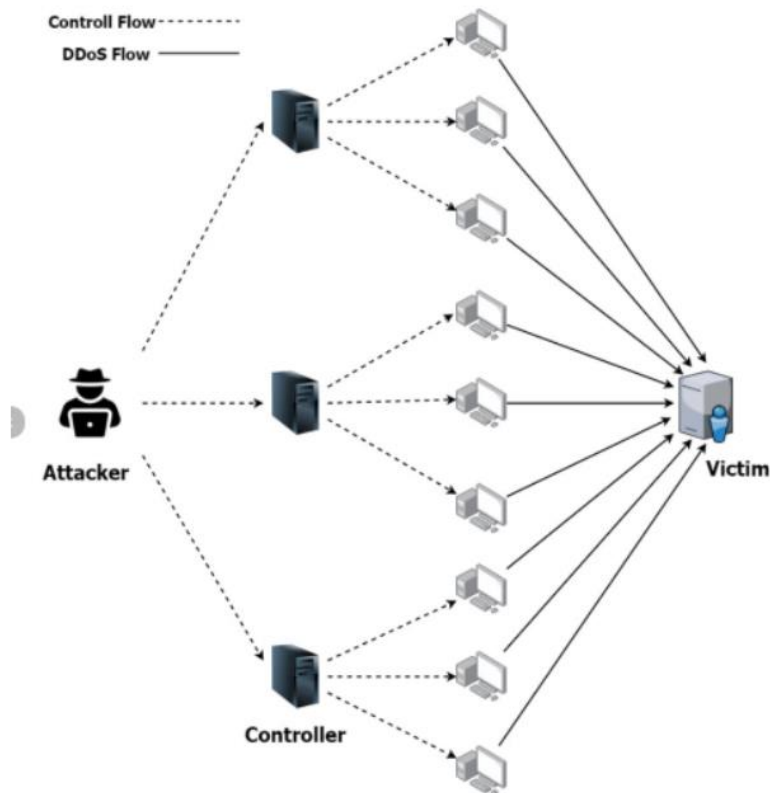


Fig 3.8.1 DDOS Attack

3.8.2 THREE-WAY HANDSHAKE

The three-way handshake is a TCP-level process used to establish a reliable connection between two systems (e.g., a client and a web server).

It involves:

SYN – Client sends a request to connect.

SYN-ACK – Server acknowledges and responds.

ACK – Client acknowledges and connection is established.

3.8.3 THREE-WAY HANDSHAKE IN AZURE APPLICATION GATEWAY

- A Layer 7 (Application Layer) load balancer.
- Used to manage HTTP/HTTPS traffic.
- Supports SSL termination, cookie-based session affinity, and Web Application Firewall (WAF).

3.8.4 STEP BY STEP FLOW

1. Client → Application Gateway (Frontend)

- The client initiates a TCP connection with the Application Gateway's public IP.
- The standard TCP three-way handshake (SYN → SYN-ACK → ACK) takes place between the client and the Application Gateway.

2. Application Gateway (Frontend) → Backend Pool

- The Application Gateway terminates the client connection (especially if SSL termination is enabled).
- Then, it initiates a new TCP connection to the selected backend server (from the backend pool).
- Another TCP three-way handshake happens between Application Gateway and the backend server.

- **Client → Gateway:**

TCP SYN → SYN-ACK → ACK → [Connection established]

- **Gateway → Backend:**

TCP SYN → SYN-ACK → ACK → [Connection established]

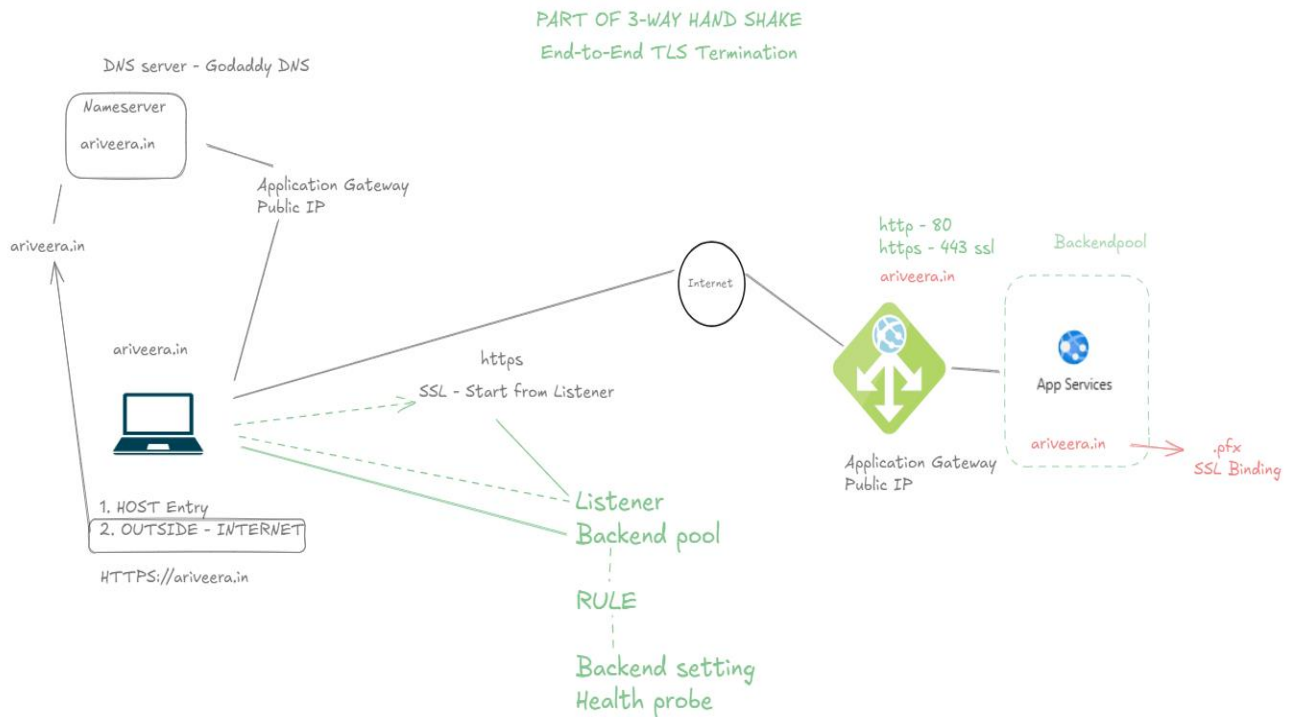


Fig 3.8.4 Three-way Handshake

3.8.5 MAPPING AZURE APPLICATION GATEWAY PUBLIC IP INTO GODADDY DOMAIN

- Application Gateway > Frontend IP configuration.
- Copy the public IP address
- Sign in to your **GoDaddy account**.
- Project.life click DNS or Manage DNS.
- We have configure **A record**
- Application Gateway has a **static public IP**.
- Record Type: A
- Name: @ (for root domain) or www (for subdomain)
- Value: Paste your **Application Gateway's public IP address**
- We have to Ensure our **HTTP/HTTPS listeners** and **rules** are properly configured in Application Gateway.

3.9 JSON SCRIPT OF APPLICATION GATEWAY

```
{
  "name": "Final-appgateway",
  "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway",
  "etag": "W/\"e2ef021f-4436-406f-9041-0cc113f71a61\"",
  "type": "Microsoft.Network/applicationGateways",
  "location": "eastus",
  "tags": {},
  "zones": [
    "1",
    "2",
    "3"
  ],
  "properties": {
    "provisioningState": "Succeeded",
    "resourceGuid": "a6b139eb-3c3d-43a0-b3e8-833402cfbd83",
    "sku": {
      "name": "WAF_v2",
      "tier": "WAF_v2",
      "family": "Generation_1",
      "capacity": 2
    },
    "operationalState": "Running",
    "gatewayIPConfigurations": [
      {
        "name": "appGatewayIpConfig",
        "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/gatewayIPConfigurations/appGatewayIpConfig",
        "etag": "W/\"e2ef021f-4436-406f-9041-0cc113f71a61\"",
        "properties": {
          "provisioningState": "Succeeded",
          "subnet": {
            "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/virtualNetworks/HUB-VNET/subnets/Appgatewaysubnet"
          }
        }
      }
    ]
  }
}
```

```

    "type": "Microsoft.Network/applicationGateways/gatewayIPConfigurations"
  }
],
"sslCertificates": [
  {
    "name": "project25",
    "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/sslCertificates/project25",
    "etag": "W/\"e2ef021f-4436-406f-9041-0cc113f71a61\"'",
    "properties": {
      "provisioningState": "Succeeded",
      "publicCertData":
"MIHKuQYJKoZIhvcNAQcCoIIKqjCCCqYCAQExADALBgkqhkiG9w0BBwGgggqOMIIFgTCCBGmgAwIBAgISBoISfSokVtMedyBsmHeT3a5/MA0GCSqGSIb3DQEBCwUAMDmxZAJBgNVBAYTAIVTMRYwFAyDVQQKEw1MZXXQncyBFbmNyeXB0MQwwCgYDVQQDEwNSMTAwHhcN/l7A9sLPB8S4FzkOSwiSHKfNjPLuBqHLA9acpvWsrkCAwEAaOCAiMwggIfMA4GA1UdDwEB/ +
+YhD+jozLW2p9W4959Bz2EiRmqDtmiXLnzqTpXbI+suyCsohKRg6Un0RC47+cpiVwHiXZAW+cn8eiNIjqbVgXLxKPpdzvvTnOPIc7SQZSYmdunr3Bf9b77AiC/ZidstK36dRILKz7OA54xAA=="
    },
    "type": "Microsoft.Network/applicationGateways/sslCertificates"
  },
  {
    "name": "project25.life",
    "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/sslCertificates/project25.life",
    "etag": "W/\"e2ef021f-4436-406f-9041-0cc113f71a61\"'",
    "properties": {
      "provisioningState": "Succeeded",
      "publicCertData":
"MIHKsQYJKoZIhvcNAQcCoIIKojCCCp4CAQExADALBgkqhkiG9w0BBwGgggqGMIIFeDCCBGCGAwIBAgISBg8IISwrgrFLasw4+W0rmoswMWgTILn1Wu0mrks7/qpdWfS6PJ1jty80r2VKsM/Dj3YIDfbjXKdaFU5C+8bhfJGqU3taKauuz0wHVGt3eo6FIWkWYtbt4pgdamlwVeZEW+LM7qZEJEsMNPrfC03APKmZsJgpWCDWO/W0rmoswMWgTILn1Wu0mrks7/qpdWfS6PJ1jty80r2VKsM/Dj3YIDfbjXKdaFU5C+8bhfJGqU3taKauuz0wHVGt3eo6FIWkWYtbt4pgdamlwVeZEW+LM7qZEJEsMNPrfC03APKmZsJgpWCDWOKZvkZcvjVuYkQ4omYCTX5ohy+knMjdOmdH9c7SpqEWBDC86fiNex+O0XOMEZSa8DAMQA=",
      "httpListeners": [
        {
          "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-

```


38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/
Final-appgateway/httpListeners/HTTPS-443"

```
    }
  ]
},
"type": "Microsoft.Network/applicationGateways/sslCertificates"
}
],
"frontendIPConfigurations": [
{
  "name": "appGwPublicFrontendIpIPv4",
  "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/  
Final-appgateway/frontendIPConfigurations/appGwPublicFrontendIpIPv4",
  "etag": "W/\"e2ef021f-4436-406f-9041-0cc113f71a61\"\"",
  "type": "Microsoft.Network/applicationGateways/frontendIPConfigurations",
  "properties": {
    "provisioningState": "Succeeded",
    "privateIPAllocationMethod": "Dynamic",
    "publicIPAddress": {
      "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/publicIPAddresses/A  
PPGWIP"
    },
    "httpListeners": [
      {
        "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/  
Final-appgateway/httpListeners/Listener-80"
      },
      {
        "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/  
Final-appgateway/httpListeners/HTTPS-443"
      }
    ]
  }
}
],
"frontendPorts": [
{
  "name": "port_80",
  "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-
```

```

38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/
Final-appgateway/frontendPorts/port_80",
    "etag": "W/\"e2ef021f-4436-406f-9041-0cc113f71a61\"",
    "properties": {
        "provisioningState": "Succeeded",
        "port": 80,
        "httpListeners": [
            {
                "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-
38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/
Final-appgateway/httpListeners/Listener-80"
            }
        ]
    },
    "type": "Microsoft.Network/applicationGateways/frontendPorts"
},
{
    "name": "port_443",
    "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-
38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/
Final-appgateway/frontendPorts/port_443",
    "etag": "W/\"e2ef021f-4436-406f-9041-0cc113f71a61\"",
    "properties": {
        "provisioningState": "Succeeded",
        "port": 443,
        "httpListeners": [
            {
                "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-
38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/
Final-appgateway/httpListeners/HTTPS-443"
            }
        ]
    },
    "type": "Microsoft.Network/applicationGateways/frontendPorts"
}
],
"backendAddressPools": [
    {
        "name": "webbackend",
        "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-
38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/
Final-appgateway/backendAddressPools/webbackend",
        "etag": "W/\"e2ef021f-4436-406f-9041-0cc113f71a61\"",

```

```

    "properties": {
      "provisioningState": "Succeeded",
      "backendAddresses": [
        {
          "fqdn": "change1121-b6fsg5hhhmdkgxaq.eastus-01.azurewebsites.net"
        }
      ],
      "requestRoutingRules": [
        {
          "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/requestRoutingRules/Rule-1"
        },
        {
          "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/requestRoutingRules/HTTPS-Rule"
        }
      ]
    },
    "type": "Microsoft.Network/applicationGateways/backendAddressPools"
  },
  "backendHttpSettingsCollection": [
    {
      "name": "bcksetting",
      "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/backendHttpSettingsCollection/bcksetting",
      "etag": "W/\"e2ef021f-4436-406f-9041-0cc113f71a61\"",
      "properties": {
        "provisioningState": "Succeeded",
        "port": 80,
        "protocol": "Http",
        "cookieBasedAffinity": "Disabled",
        "pickHostNameFromBackendAddress": false,
        "requestTimeout": 20,
        "requestRoutingRules": [
          {
            "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/requestRoutingRules/Rule-1"
          }
        ]
      }
    }
  ]
}

```

```

        }
    ]
},
"type": "Microsoft.Network/applicationGateways/backendHttpSettingsCollection"
},
{
    "name": "HTTPS-443bckend",
    "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/backendHttpSettingsCollection/HTTPS-443bckend",
    "etag": "W/\"e2ef021f-4436-406f-9041-0cc113f71a61\"\"",
    "properties": {
        "provisioningState": "Succeeded",
        "port": 443,
        "protocol": "Https",
        "cookieBasedAffinity": "Disabled",
        "hostName": "project25.life",
        "pickHostNameFromBackendAddress": false,
        "path": "",
        "requestTimeout": 20,
        "requestRoutingRules": [
            {
                "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/requestRoutingRules/HTTPS-Rule"
            }
        ]
    },
    "type": "Microsoft.Network/applicationGateways/backendHttpSettingsCollection"
}
],
"httpListeners": [
    {
        "name": "Listener-80",
        "etag": "W/\"e2ef021f-4436-406f-9041-0cc113f71a61\"\"",
        "properties": {
            "provisioningState": "Succeeded",
            "frontendIPConfiguration": {

        },
        "frontendPort": {

```

```

        "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/frontendPorts/port_80"
      },
      "protocol": "Http",
      "requireServerNameIndication": false,
      "requestRoutingRules": [
        {
          "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/requestRoutingRules/Rule-1"
        }
      ]
    },
    "type": "Microsoft.Network/applicationGateways/httpListeners"
  },
  {
    "name": "HTTPS-443",
    "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/httpListeners/HTTPS-443",
    "etag": "W/\"e2ef021f-4436-406f-9041-0cc113f71a61\"",
    "properties": {
      "provisioningState": "Succeeded",
      "frontendIPConfiguration": {
        "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/frontendIPConfigurations/appGwPublicFrontendIpIPv4"
      },
      "frontendPort": {
        "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/frontendPorts/port_443"
      },
      "protocol": "Https",
      "sslCertificate": {
        "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/sslCertificates/project25.life"
      },
      "hostName": "project25.life",
      "requireServerNameIndication": true,
      "requestRoutingRules": [

```

```

        {
            "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-
38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/
Final-appgateway/requestRoutingRules/HTTPS-Rule"
        }
    ],
    },
    "type": "Microsoft.Network/applicationGateways/httpListeners"
}
],
"urlPathMaps": [],
"requestRoutingRules": [
    {
        "name": "Rule-1",
        "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-
38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/
Final-appgateway/requestRoutingRules/Rule-1",
        "etag": "W/\"e2ef021f-4436-406f-9041-0cc113f71a61\"",
        "properties": {
            "provisioningState": "Succeeded",
            "ruleType": "Basic",
            "httpListener": {
                "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-
38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/
Final-appgateway/httpListeners/Listener-80"
            },
            "backendAddressPool": {
                "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-
38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/
Final-appgateway/backendAddressPools/webbackend"
            },
            "httpListener": {
                "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-
38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/
Final-appgateway/httpListeners/HTTPS-443"
            },
            "backendAddressPool": {
                "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-
38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/
Final-appgateway/backendAddressPools/webbackend"
            },
            "backendHttpSettings": {

```

```

        "id": "/subscriptions/bf6e09fe-e8d6-4494-b2bf-38eab2bc8fed/resourceGroups/project2025/providers/Microsoft.Network/applicationGateways/Final-appgateway/backendHttpSettingsCollection/HTTPS-443bckend"
    }
},
    "type": "Microsoft.Network/applicationGateways/requestRoutingRules"
}
],
    "probes": [],
    "redirectConfigurations": [],
    "enableHttp2": true
},
    "apiVersion": "2017-10-01"
}

```

3.10 APPLICATION GATEWAY SECURE WEB HOSTING ENABLE WITH WAF

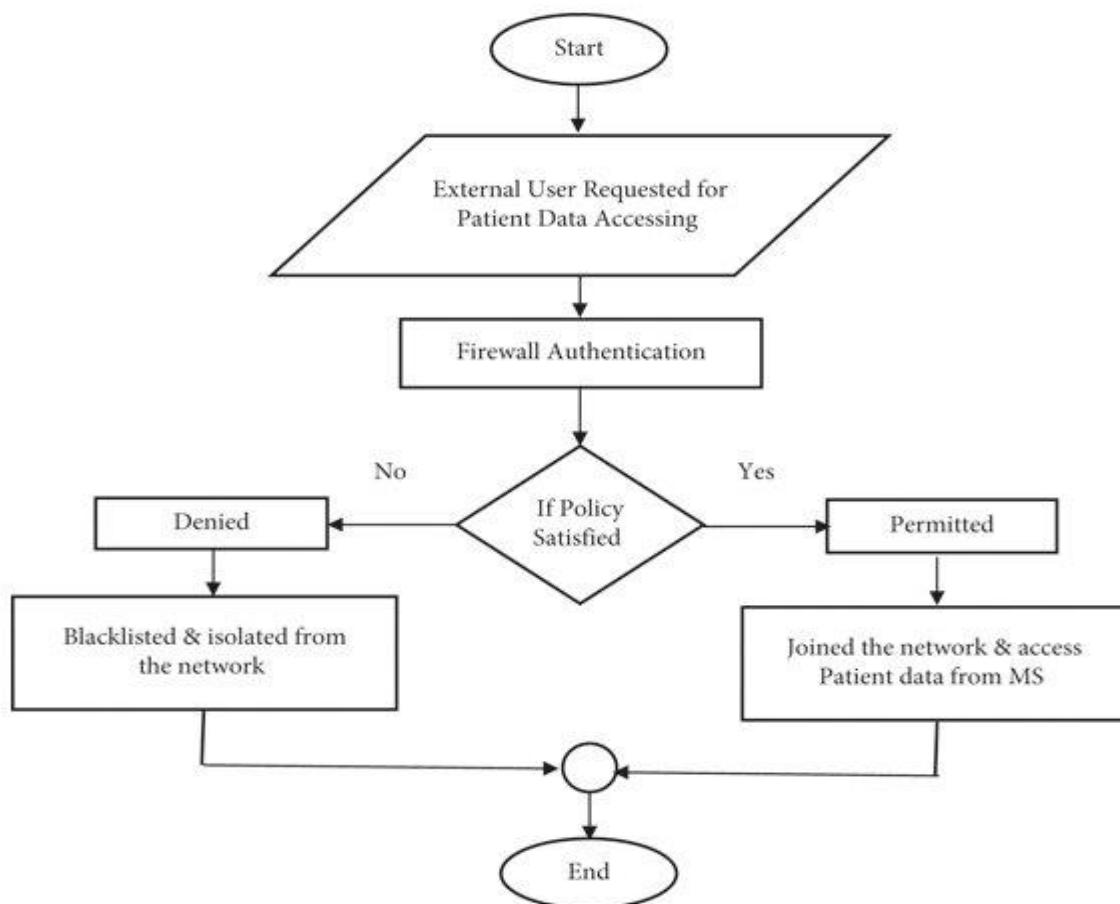


Fig 3.10 Activity diagram for WAF Authentication

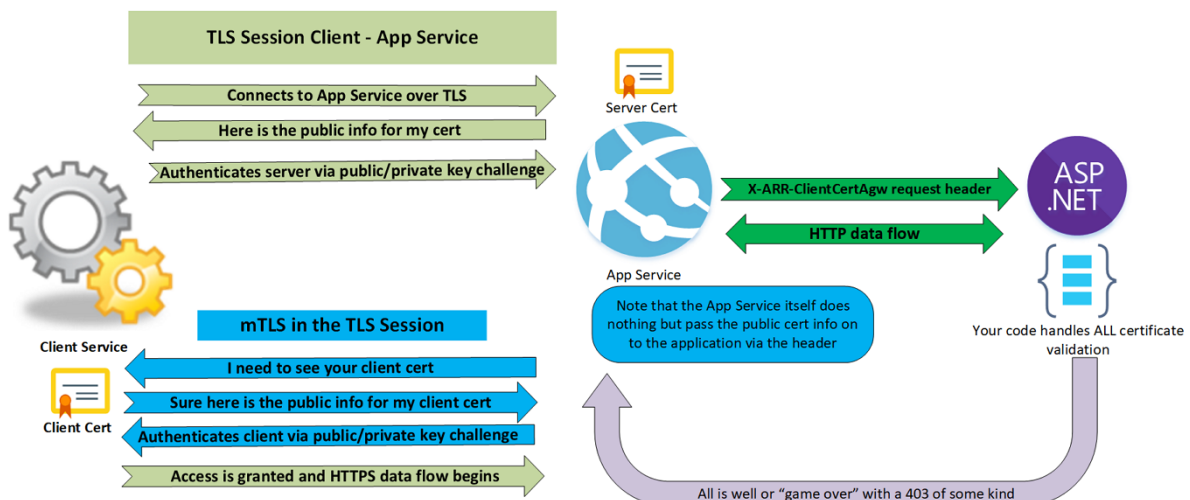
3.11 WEB HOSTING WITH SSL

3.11.1 WEB APPLICATION CUSTOM DOMAIN BINDINGS

In **Azure App Service**, **custom domain bindings** allow you to access your web app using your own domain (e.g., project25.life) instead of the default Azure domain (yourapp.azurewebsites.net). You can even bind **multiple custom domains** to a single Azure Web App.

In Azure App Service, you can bind multiple custom domains to a single web app, allowing it to be accessible through different domain names such as www.example.com, example.com, or even subdomains like api.example.com. To do this, first deploy your web app and then go to the Azure portal. In your web app's settings, navigate to the “Custom domains” section and click on “Add custom domain.” You'll be prompted to enter the domain you want to bind. Before Azure allows you to bind the domain, it must verify that you own it. This is typically done by updating your domain registrar's DNS settings—either by adding a CNAME record (ideal for subdomains) pointing to your Azure app's default domain (yourapp.azurewebsites.net), or an A record (used for root domains) pointing to the Azure App Service's IP address, along with a required TXT record for validation.

Client connects directly to the Azure App Service which requires mTLS



Once the DNS changes propagate and Azure successfully verifies the domain, you can bind it to the web app. You can repeat this process to add additional domains. After binding, it's highly recommended to secure each domain using HTTPS. Azure offers free App Service-managed SSL certificates, which can be added under the "TLS/SSL Settings" > "Bindings" section. For each domain, select the free certificate and enable the binding, which will automatically handle renewals. This process ensures that users accessing your app through any of the custom domains do so securely and with consistent branding.

3.11.2 SSL BINDING WITH CUSTOM DOMAIN

SNI SSL (Server Name Indication SSL) binding in Azure App Service allows you to securely serve your web app over HTTPS using a custom domain, even when multiple domains share the same IP address. Unlike IP-based SSL, which requires a dedicated IP for each SSL certificate, SNI SSL leverages the SNI extension of the TLS protocol to allow multiple SSL certificates to coexist on a single IP. To configure SNI SSL binding in Azure, you first need to add your custom domain to the web app through the "Custom domains" section.

After verifying domain ownership and completing the domain binding, you navigate to the "TLS/SSL Settings" section and open "Bindings." There, you can add a new TLS/SSL binding by selecting your custom domain, choosing an SSL certificate (either an App Service Managed Certificate or one you've uploaded), and setting the binding type to SNI SSL. This enables encrypted HTTPS communication for that specific custom domain without needing a dedicated IP. SNI SSL is cost-effective and scalable, especially for apps with multiple custom domains, and it's supported by all modern browsers.

Custom domains ☆ ...

Refresh

Troubleshoot

Configure and manage custom domains assigned to your app. [Learn more](#)

IP address: 40.76.218.33,40.71.11.170

Custom Domain Verification ID: 1687DD76ACE661F3477F8629C3A20DB8D1446888F50248...

Filter by keywords

Add filter

2 items

+ Add custom domain

+ Buy App Service domain

Delete

Custom domains	Status	Solution	Binding type	Certificate used
<input checked="" type="checkbox"/> project25.life	Secured	-	SNI SSL	project25.life-change1121
change1121-b6fsg5hhmdkgxaq...	Secured	-	-	-

Update TLS/SSL binding

Domain

project25.life

Certificate *

project25.life-change1121

TLS/SSL type *

Current certificates used

project25.life-change1121

335956CAC18717F3D8D9A110F6DE1D03C68E68

25

(Expires on 8/18/2025)

3.11.3 APPLICATION GATEWAY BACKEND TARGET

Attaching an Azure Application Gateway backend target as a Web App (App Service) allows you to route and manage traffic to your web app using advanced features like URL-based routing, SSL termination, Web Application Firewall (WAF), and autoscaling. To configure this setup, you need to ensure that your Azure Web App is integrated with a Virtual Network (VNet), since Application Gateway requires backend targets to be accessible via private IP or internal DNS names. This is achieved by enabling VNet Integration on the Web App and using Private Endpoint or App Service Environment (ASE) for secure communication within the VNet.

Once your web app is integrated into the VNet, you can proceed to configure the Application Gateway. In the Application Gateway settings, under "Backend Pools", add a new backend pool and specify the private IP address or private FQDN of the Web App (accessible through the private endpoint). Then configure the HTTP settings to match your web app's requirements, such as enabling or disabling cookie-based affinity, setting the backend protocol (HTTP or HTTPS), and uploading the necessary SSL certificate if HTTPS is used. After this, you link the backend pool to routing rules, specifying how traffic should be forwarded to the web app.

52

Finally, update your DNS or custom domain configuration to point to the Application Gateway's frontend IP address. This setup provides a robust and secure architecture where Application Gateway handles incoming traffic and forwards it to your web app over the Azure backbone, leveraging security and performance features.

3.11.4 CUSTOM LISTENERS

To configure an **Azure Application Gateway listener** with a **custom domain over HTTPS**, you need to set up a listener that can handle secure traffic and route it to the appropriate backend—typically an Azure Web App or virtual machine—using a custom domain like `www.example.com`. This process involves several components, including SSL certificates, DNS configuration, and routing rules.

Steps: To use a custom domain with HTTPS on Azure Application Gateway, you begin by obtaining an **SSL certificate** for your domain—either from a trusted Certificate Authority (CA) or by using **Azure Key Vault** with a certificate stored there. Once your Azure Application Gateway is deployed, go to the **Listeners** section and create a **multi-site HTTPS listener**. This type of listener supports multiple hostnames (domains), making it ideal for routing traffic based on the Host header. In the listener configuration, specify the frontend port (usually 443 for HTTPS), set the protocol to HTTPS, and upload your PFX certificate or select it from Key Vault. Make sure the listener is set to respond only to requests for your custom domain (e.g., `www.example.com`).

Next, configure a **backend pool** that contains the destination (e.g., a Web App with VNet integration or a virtual machine), and create an **HTTP setting** that defines how the Application Gateway communicates with the backend (HTTP or HTTPS, custom probe settings, etc.). Then, set up a **routing rule** that ties the listener, backend pool, and HTTP settings together.

This rule tells the gateway how to handle requests that match your domain and port.

Finally, update your domain registrar's **DNS settings** to point your custom domain (e.g., `www.example.com`) to the **public IP address of the Application Gateway**. This ensures that incoming HTTPS traffic is received by the Application Gateway, decrypted using your SSL certificate, and forwarded to the correct backend securely. This configuration gives you full control over HTTPS traffic, including SSL offloading, Web Application Firewall protection, and URL-based routing—all while using your custom domain.

3.12 APPLICATION GATEWAY RULES

Azure Application Gateway rules define how incoming web traffic is routed to backend servers. These rules control the flow of traffic based on conditions like URL paths, hostnames, and HTTP settings. There are two main types of rules: basic and path-based. Basic rules route all incoming requests on a listener to a single backend pool, while path-based rules allow more granular routing based on specific URL paths.

For example, traffic to `/images` could be sent to one backend pool, while `/videos` goes to another. Each rule is associated with components like a listener (to capture traffic), a backend pool (the target servers), and HTTP settings (such as port, protocol, and cookie handling). Together, these rules enable precise control over how web requests are distributed, improving performance, security, and scalability of applications hosted in Azure.

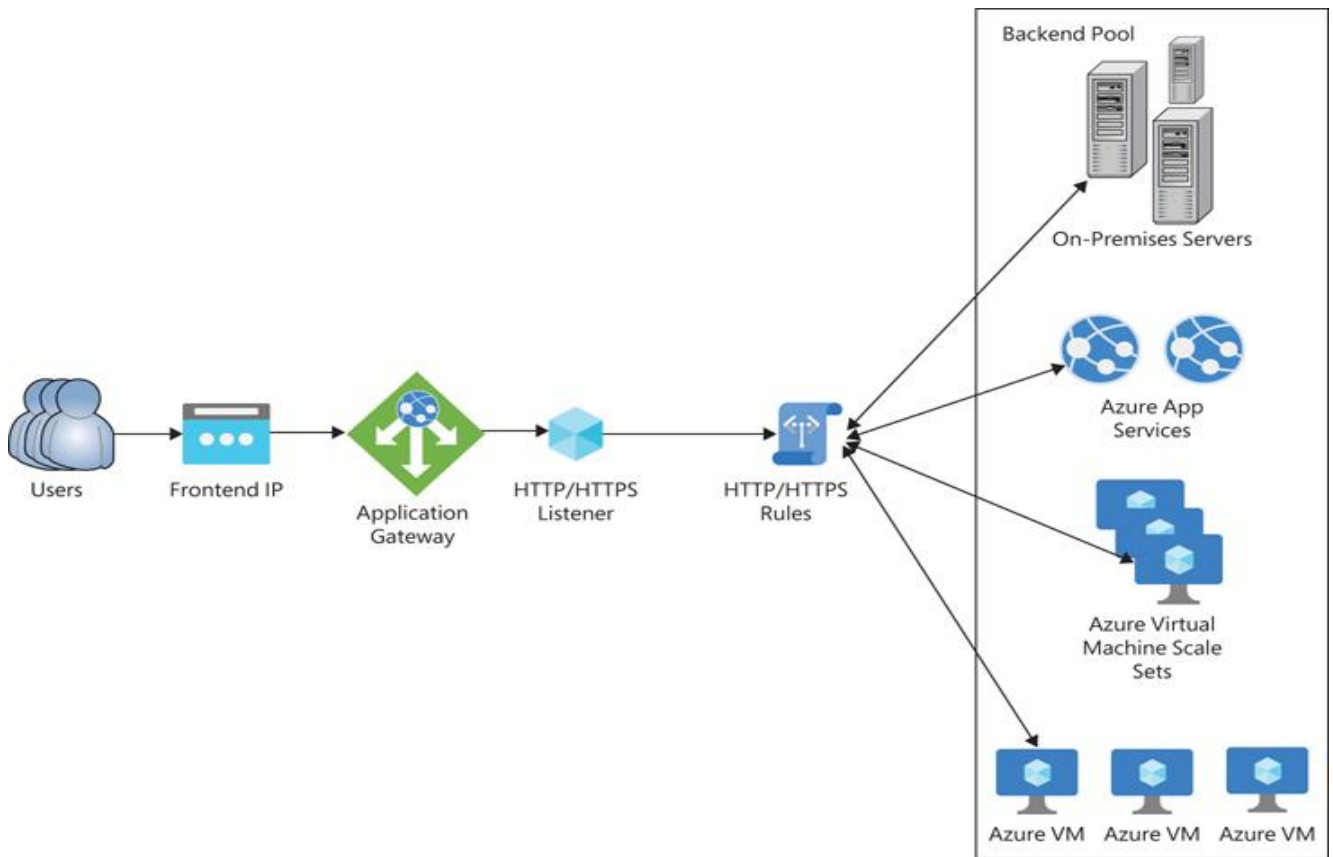


Fig 3.12 AppGateway Rules architect

3.13 HEALTH PROBE

Health probes in Azure Application Gateway are configured to test the availability of backend resources by sending requests at regular intervals to a specified path (e.g., /health or /status). You can customize the probe to use either HTTP or HTTPS, specify the hostname, path, port, and expected status codes (usually 200–399). You also set parameters like the interval between checks, the number of successful responses required to mark an endpoint as healthy, and the number of failures needed to mark it unhealthy.

For example, if a probe is configured to send an HTTPS request to /health every 30 seconds, and expects a 200 response, it will check that path on all backend instances. If one of the instances fails to return a 200 response three times in a row, the Application Gateway will remove that instance from the load balancing rotation. Once the instance starts responding successfully again for the configured number of times, it is marked as healthy and resumes receiving traffic.

3.13.1 BACKEND – HEALTH PROBE DIAGRAM

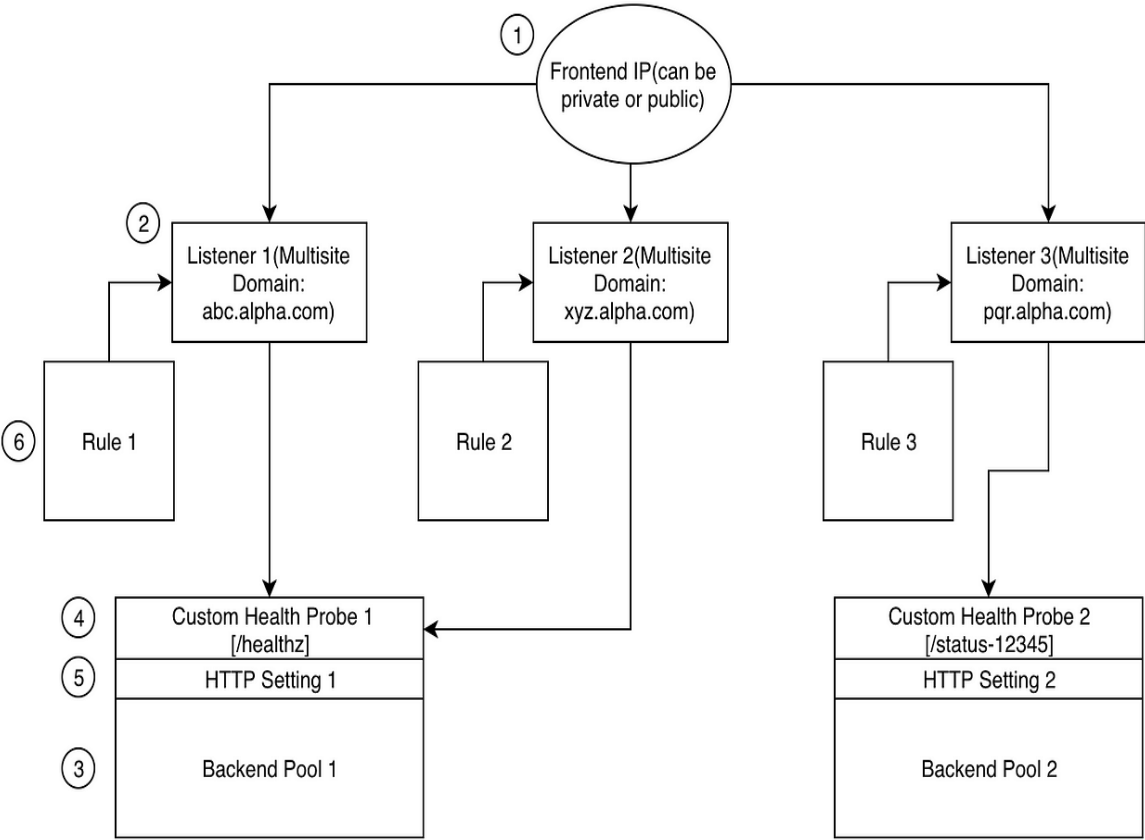


Fig. 3.13.1 Diagram of Backend Health Probe

3.13.2 APPLICATION GATEWAY WAF DATA FLOW DIAGRAM

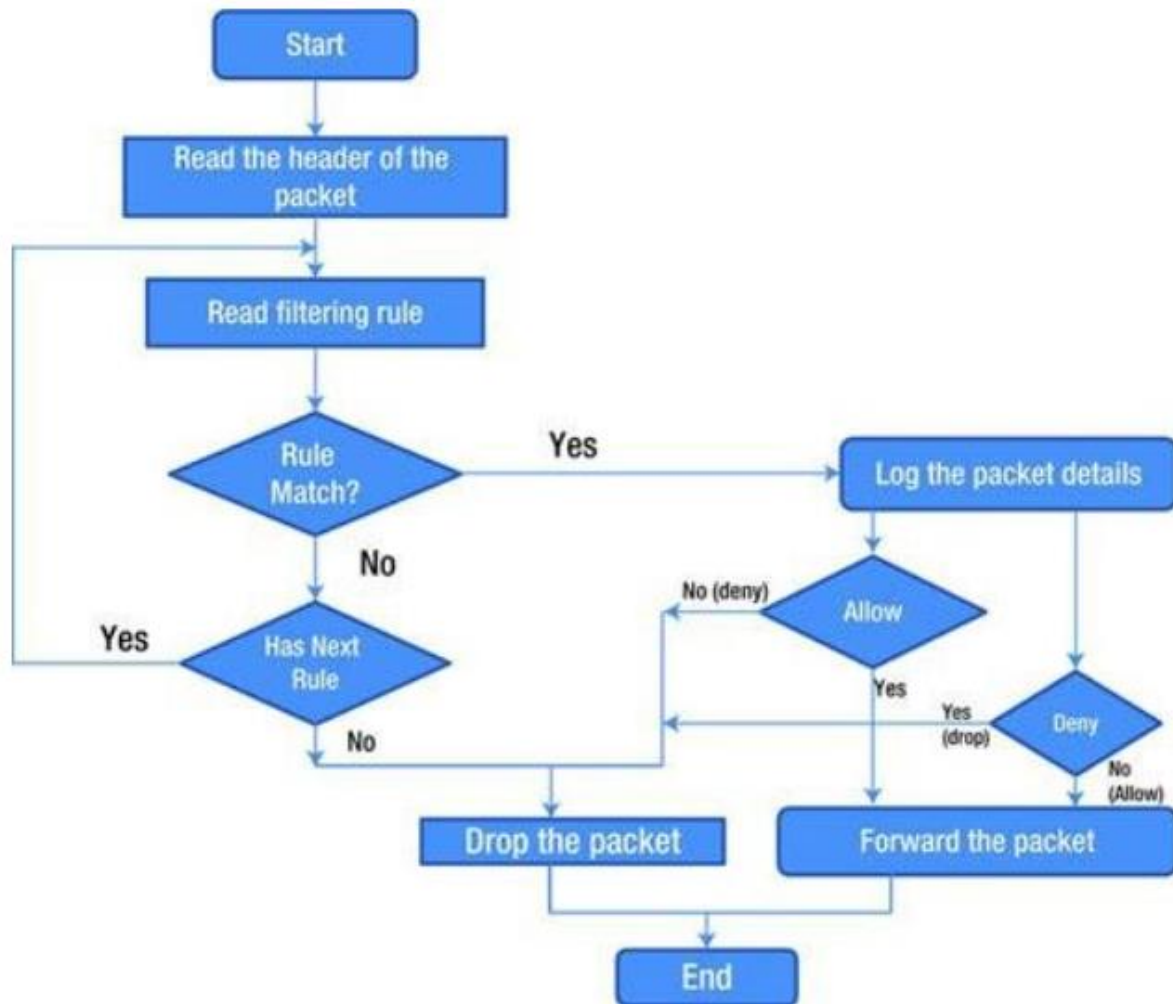


Fig 3.13.2 Application Gateway WAF Data Flow Diagram

CHAPTER – 4

WEB APPLICATION FIREWALL

4.1 WEB APPLICATION FIREWALL DESCRIPTION

A **Web Application Firewall (WAF)** is a specialized security system designed to **protect web applications** from common cyberattacks and malicious traffic. In the context of **Azure**, the **Azure Web Application Firewall** is a feature of **Azure Application Gateway** and **Azure Front Door** that provides centralized protection for your web apps against threats such as **SQL injection**, **cross-site scripting (XSS)**, **local file inclusion**, and other **OWASP vulnerabilities**.

A **Web Application Firewall** acts as a shield between your web application and incoming traffic. It inspects every HTTP and HTTPS request and applies a set of security rules to detect and block potentially harmful traffic before it reaches your application. In Azure, WAF policies can be customized or based on **predefined rule sets** maintained by Microsoft, such as the **OWASP Core Rule Set (CRS)**. These rules can detect patterns of known exploits or suspicious behaviors, such as malformed requests, code injection attempts, or abnormal usage patterns.

WAF in Azure also supports **bot protection**, **geo-filtering**, **rate limiting**, and **custom rules**, which allow developers and security teams to fine-tune security to their application's specific needs. Deployed with services like **Application Gateway**, WAF provides real-time threat prevention, centralized logging via **Azure Monitor**, and integration with **Security Center**—all without changing your app code. This makes WAF a powerful and scalable solution for securing public-facing applications against modern web threats.

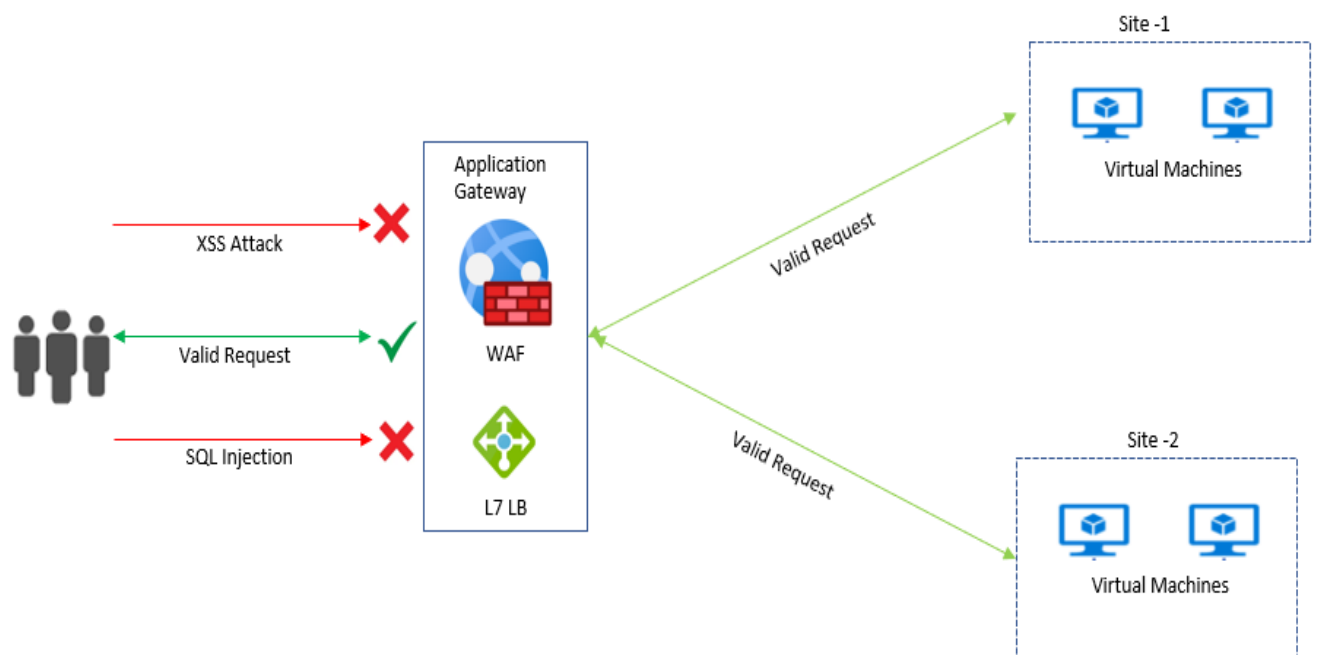


Fig 4.1 Web Application Firewall (WAF)

The **Web Application Firewall (WAF)** in platforms like **Azure**, **AWS**, or **Cloudflare** is built to defend web apps against threats categorized in the **OWASP Top 10**—a widely accepted list of the most critical web application security risks published by the **Open Web Application Security Project (OWASP)**.

In Azure, WAF uses the **OWASP Core Rule Set (CRS)**, currently based on **OWASP CRS 3.x**, which contains detailed rules and signatures designed to detect and block the most common and dangerous attack patterns.

4.2 OWASP TOP 10 CRS:

The OWASP Top 10 is a list of the most critical web application security risks, updated periodically by security experts. Azure WAF and other enterprise-grade firewalls align their protection mechanisms to detect and mitigate these 10 categories. Here's how Azure WAF handles each one:

1. Broken Access Control

WAF uses rule sets to detect unauthorized access attempts to admin pages or restricted resources (e.g., /admin, /config). It can block or log requests that violate access patterns, such as forced browsing or path manipulation.

2. Cryptographic Failures (formerly Sensitive Data Exposure)

Though WAF can't fix cryptographic issues (like misconfigured HTTPS or weak ciphers), it can **block traffic trying to exploit them**, such as downgrade attacks or the leakage of tokens via query strings.

3. Injection (SQL, NoSQL, Command)

One of the core features of WAF is to **detect injection patterns** like:

' OR '1'='1

-- (SQL comment)

Shell injection attempts (; rm -rf /)

These are caught using **regex-based matching rules** and **anomaly scoring** to determine intent.

4. Insecure Design

WAFs **can't directly fix poor architectural decisions**, but they can mitigate the **exploitation** of those flaws. For example, WAF can block unexpected HTTP methods (PUT, DELETE) or malformed inputs that may hint at insecure designs.

5. Security Misconfiguration

WAF can detect and prevent exploitation of common misconfigurations, such as:

- Exposed admin consoles (/phpmyadmin)
- Unused HTTP methods
- Cross-origin requests from unexpected origins

6. Vulnerable and Outdated Components

While WAF doesn't scan your code or dependencies, it can **block**

known exploit patterns targeting vulnerable versions of platforms (like WordPress, Drupal, Log4j) using signature rules.

7. Identification and Authentication Failures

WAFs can enforce security by blocking:

- **Brute force attacks** (repeated login attempts)
- **Credential stuffing**

Requests that bypass authentication headers or manipulate sessions

8. Software and Data Integrity Failures

WAF can block unauthorized modifications or exploit payloads targeting CI/CD backdoors or unsafe file uploads. For instance, it can inspect file types or block suspicious upload patterns.

9. Security Logging and Monitoring Failures

While WAF doesn't generate application logs, it **integrates with Azure Monitor, Log Analytics, and Sentinel** to provide centralized logging of all security events and blocked requests.

10. Server-Side Request Forgery (SSRF)

Advanced WAFs can detect SSRF attempts by analyzing URLs and payloads for internal IP address access (169.254.x.x, localhost, etc.) and **block outbound redirection or internal service calls**.

CHAPTER 5

RESULT & DISCUSSION

5.1 OUTPUT

We have configure a **custom domain** (project25.life) with **Azure Application Gateway**, enabling to access backend via a **friendly domain name** like project25.life instead of using an IP address or default Azure DNS name. This process involves integrating **DNS settings**, **Application Gateway listeners**, **SSL certificates (for HTTPS)**, and **backend pool routing**.

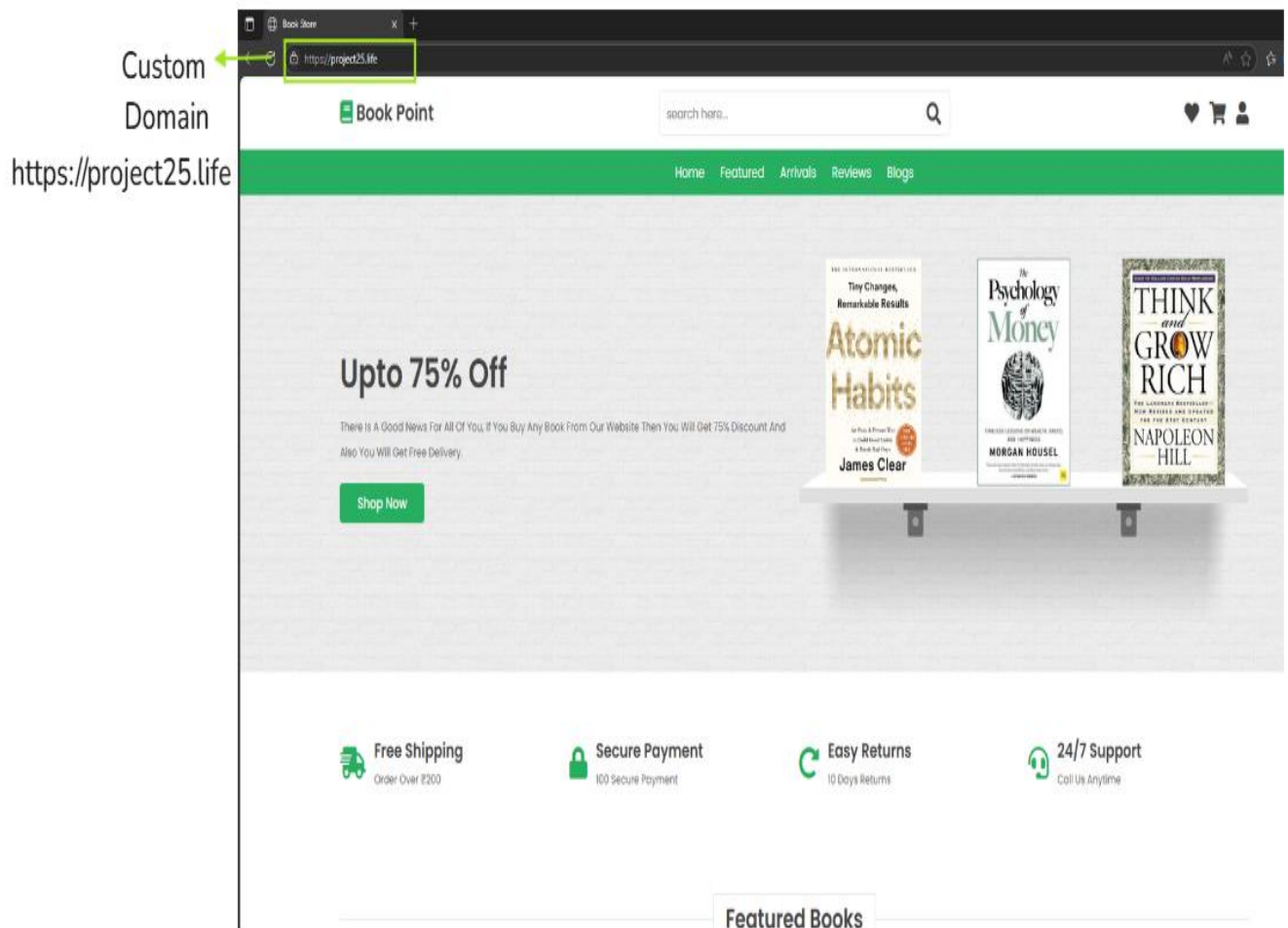


Fig 5.1 Custom Domain Output

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

Integrating **Azure Application Gateway** with a **Web App (App Service)** provides a highly secure, scalable, and manageable architecture for hosting modern web applications. The Application Gateway acts as a **reverse proxy and Layer 7 load balancer**, offering features like **SSL offloading**, **Web Application Firewall (WAF)** protection, **URL-based routing**, and **custom domain support**. When paired with Azure Web App, it enables better **traffic control**, **security**, and **performance optimization** without requiring changes to application code. This setup also allows seamless deployment and centralized monitoring through Azure Monitor and Log Analytics, making it ideal for enterprise-grade applications with high availability requirements.

6.2 FUTURE WORK

While the current setup provides robust security and routing capabilities, future enhancements could include:

1. Auto-Scaling with Backend Health Awareness:

Integrate Application Gateway with **Azure Autoscale** policies to dynamically adjust backend resources based on WAF logs or traffic patterns.

2. Advanced Routing and Global Redundancy:

Combine Application Gateway with **Azure Front Door** or **Traffic Manager** for **multi-region deployments**, global failover, and latency-based routing.

3. Enhanced Monitoring and AI-Driven Threat Detection:

Leverage **Microsoft Defender for Cloud** and **Sentinel** for deeper analytics, threat detection, and automated responses to WAF alerts.

4. Zero-Trust Networking with Private Link:

Secure backend Web Apps further using **Private Endpoints**, ensuring that traffic never traverses the public internet.

5. Infrastructure as Code (IaC):

Automate the deployment and configuration of Application Gateway and Web Apps using **Bicep**, **Terraform**, or **ARM templates** to improve consistency and DevOps integration.

6. Custom WAF Rule Optimization:

Fine-tune WAF rules to reduce false positives and incorporate **business-specific protection rules** using custom conditions and anomaly scoring.

REFERENCES

- [1] Microsoft Corporation, *Azure Application Gateway documentation*, Microsoft Learn, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/application-gateway/>
- [2] Microsoft Corporation, *Add a web app to Application Gateway*, Microsoft Learn, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/application-gateway/configure-web-app>
- [3] Microsoft Corporation, *Configure custom domain name in Application Gateway*, Microsoft Learn, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/application-gateway/custom-domain>
- [4] Microsoft Corporation, *Web Application Firewall documentation*, Microsoft Learn, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/web-application-firewall/>
- [5] OWASP Foundation, *OWASP Top 10: The Ten Most Critical Web Application Security Risks*, 2021. [Online]. Available: <https://owasp.org/www-project-top-ten/>
- [6] OWASP Core Rule Set Team, *OWASP ModSecurity Core Rule Set (CRS)*, OWASP Foundation, 2023. [Online]. Available: <https://coreruleset.org/>
- [7] Microsoft Corporation, *Application Gateway health probes overview*, Microsoft Learn, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/application-gateway/application-gateway-probe-overview>
- [8] [8] Microsoft Corporation, *Listeners in Application Gateway*, Microsoft Learn, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/application-gateway/application-gateway-listeners>

Conference Certificate and Github Link

**GOJAN SCHOOL OF BUSINESS AND TECHNOLOGY**
AN AUTONOMOUS INSTITUTION
APPROVED BY AICTE | AFFILIATED TO ANNA UNIVERSITY | ACCREDITED BY NAAC
80 FEET ROAD, EDAPALAYAM, ALAMATHI VIA, REDHILLS, CHENNAI - 600052

INSTITUTION'S INNOVATION COUNCIL
Member of AICTE (2019)

ASSOCIATE PARTNER
**VIPS.Tech**
Virtual Intelligence and Platform Services

NATIONAL CONFERENCE ON RECENT TRENDS IN ENGINEERING & TECHNOLOGY (NCRTE'25)

CERTIFICATE OF PARTICIPATION

25th & 26th April 2025

This is to certify that [He/She] SURESH KUMAR D / CSE has presented a paper entitled SECURE WEB HOSTING WITH HTTPS ON MICROSOFT AZURE - CLOUD COMPUTING from GOJAN SCHOOL OF BUSINESS AND TECHNOLOGY in the National Conference on Recent Trends in Engineering & Technology (NCRTE'25) organized by Gojan School of Business and Technology, Chennai, in association with VIPS.Tech, Chennai, held on 25th & 26th April 2025.


Dr. Nafeez Ahmed L
Director - R&D
Conference chair


Dr. Selvakumar C
Principal
Gojan School of Business and Technology



Mr. Dev Anand D
CEO & Founder
VIPS.Tech


 **AWS** SPONSORSHIP & STRATEGIC PARTNERSHIPS


 **Microsoft** Silver Partner





**GOJAN SCHOOL OF BUSINESS AND TECHNOLOGY**
AN AUTONOMOUS INSTITUTION
APPROVED BY AICTE | AFFILIATED TO ANNA UNIVERSITY | ACCREDITED BY NAAC
80 FEET ROAD, EDAPALAYAM, ALAMATHI VIA, REDHILLS, CHENNAI - 600052

INSTITUTION'S INNOVATION COUNCIL
Member of AICTE (2019)


ASSOCIATE PARTNER
**VIPS.Tech**
Virtual Intelligence and Platform Services


NATIONAL CONFERENCE ON RECENT TRENDS IN ENGINEERING & TECHNOLOGY (NCRTE'25)

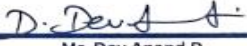
CERTIFICATE OF PARTICIPATION


25th & 26th April 2025


This is to certify that [He/She] VIMALESH C / CSE has presented a paper entitled SECURE WEB HOSTING WITH HTTPS ON MICROSOFT AZURE - CLOUD COMPUTING from GOJAN SCHOOL OF BUSINESS AND TECHNOLOGY in the National Conference on Recent Trends in Engineering & Technology (NCRTE'25) organized by Gojan School of Business and Technology, Chennai, in association with VIPS.Tech, Chennai, held on 25th & 26th April 2025.



Dr. Nafeez Ahmed L
Director - R&D
Conference chair



Dr. Selvakumar C
Principal
Gojan School of Business and Technology

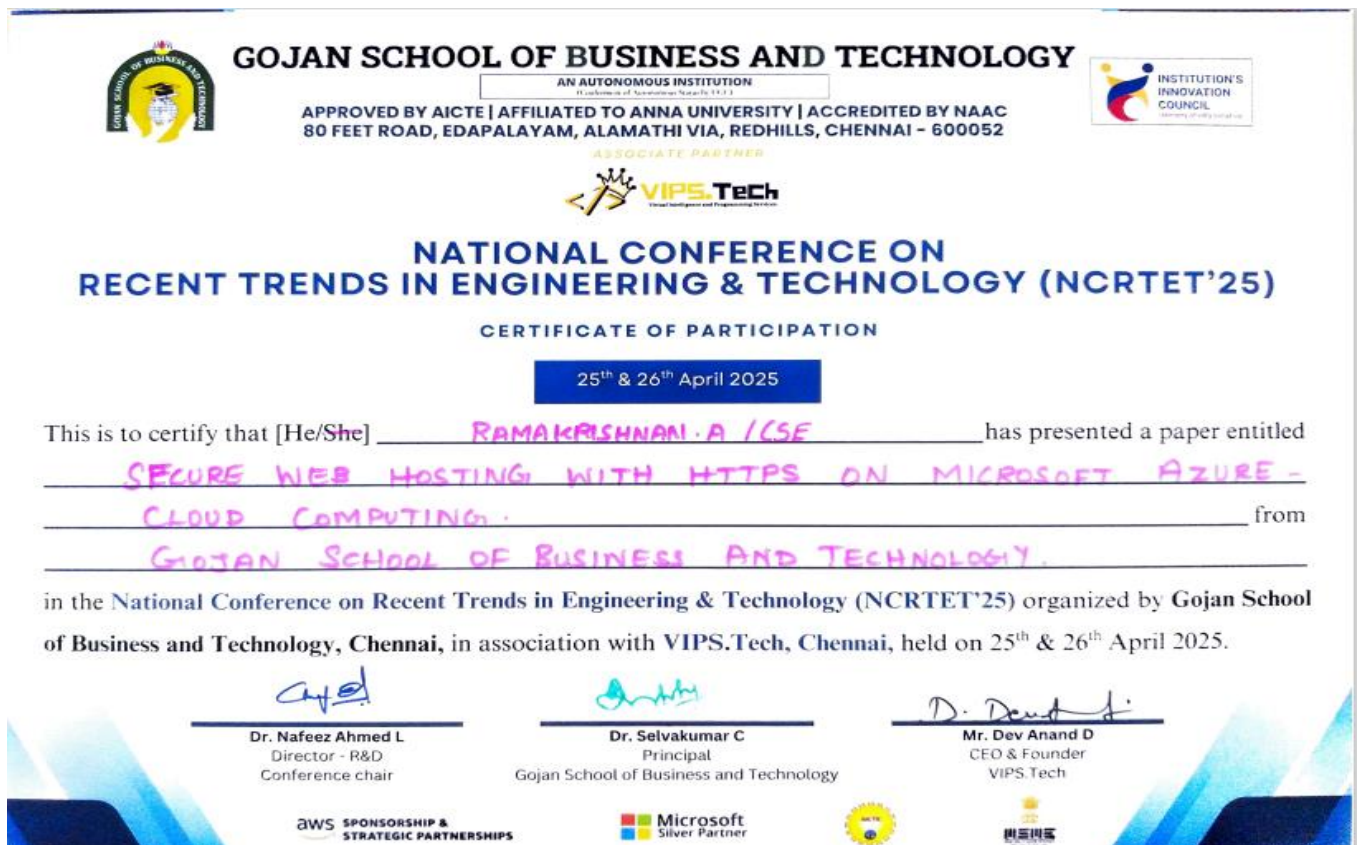

Mr. Dev Anand D
CEO & Founder
VIPS.Tech

 **AWS** SPONSORSHIP & STRATEGIC PARTNERSHIPS

 **Microsoft** Silver Partner







GITHUB Repository Link

<https://github.com/SURESHKUMAR-007/Implementation-of-Secure-web-hosting-using-https-on-Microsoft-Azure-.git>

