INTERNSHIP PROJECT REPORT

Azure Blob Storage – Static Website Hosting & Lifecycle

Submitted by: Oleti Raghu Sai Varun

Internship: Smarted Innovations (Training by Embrizon Team)

Date: June 2025

DECLARATION

I, **Oleti Raghu Sai Varun**, hereby declare that the project report titled **"Azure Blob Storage – Static Website Hosting & Lifecycle Management"** is the outcome of my own work carried out during my internship at **Smarted Innovations** (Training by **Embrizon Team**).

This project has been completed under the guidance provided by the training team and is submitted as part of the internship deliverables. The work presented is original and has not been submitted elsewhere for any academic or professional purpose.

Signature: Oleti Raghu Sai Varun
Date: June 2025

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **Smarted Innovations** and the **Embrizon Training Team** for giving me the opportunity to work on this project as part of my internship.

I extend my heartfelt thanks to the trainers and mentors who provided constant support, valuable insights, and hands-on training throughout the course of this project. Their guidance was instrumental in helping me understand the practical implementation of cloud technologies using Microsoft Azure.

I am also thankful to my college faculty, friends, and family for their encouragement and moral support during this internship. Their contributions helped me stay focused and motivated throughout the learning journey.

This internship project has been an enriching experience and has significantly enhanced my technical and professional skills in the field of cloud computing.

Oleti Raghu Sai Varun
Date: June 2025

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

This project focuses on hosting a static website using **Microsoft Azure Blob Storage**, a scalable and cost-effective cloud storage solution. Azure Blob Storage enables users to store unstructured data such as HTML files, images, and documents efficiently.

In this project, I hosted a basic static website using the $web container in Azure Blob Storage and configured public access for live preview. Additionally, I implemented **lifecycle management policies** to automatically transition blob storage tiers, optimizing long-term storage costs.

The hands-on experience helped me understand the process of cloud-based static hosting, storage access configuration, and data lifecycle optimization using Azure services.

CHAPTER 2: OBJECTIVE

The primary objective of this project is to gain practical experience in hosting a static website using **Microsoft Azure Blob Storage** and implementing **lifecycle management policies** to optimize storage usage and cost.

The project focuses on:

- Creating and configuring an Azure Storage Account
- Uploading and managing static web content (HTML files) in the $web container
- Enabling public access to host the site via a browser-accessible endpoint
- Defining a lifecycle rule to transition unused blobs to the **Cool** tier after a set period

This task helped enhance my understanding of how cloud storage works, how Azure services can be used for web deployment, and how automated policies help manage cloud resources efficiently

CHAPTER 3: TOOLS AND TECHNOLOGIES USED

This project was executed using various cloud services and supporting tools that enabled the development and deployment of a static website on Microsoft Azure. The key tools and technologies used are listed below:

- **Microsoft Azure Portal**

A cloud-based interface provided by Microsoft to manage Azure services. It was used to create the storage account, configure static website hosting, upload HTML files, and apply lifecycle rules.

- **Azure Blob Storage**

A scalable object storage service used to host static web content and store unstructured data such as HTML, images, and scripts.

- **HTML (index.html, error.html)**

Basic web files used to create and test the static website functionality. These files were uploaded to the $web container.

- **Azure Lifecycle Management**

A feature that automates blob tier transitions to help optimize storage cost. A rule was configured to move blobs to the **Cool** tier after 100 days.

- **Web Browser (Microsoft Edge/Chrome)**

Used to access the Azure Portal and verify the live static website through the public endpoint.

- **Operating System: Windows 10/11**

The environment from which all Azure configurations and uploads were performed.

CHAPTER 4: AZURE BLOB STORAGE OVERVIEW

Azure Blob Storage is Microsoft's cloud-based object storage solution designed for storing large amounts of unstructured data such as text, images, videos, and web files. It provides scalable, secure, and cost-effective storage that can be accessed from anywhere over HTTP or HTTPS.

Blob storage is organized into **containers**, and in the case of static website hosting, a special container named **$web** is used to serve HTML, CSS, JavaScript, and other static files directly to users via a public URL. This allows developers to host lightweight websites without the need for servers or additional backend services.

Azure Blob Storage also supports **different storage tiers** — Hot, Cool, and Archive — allowing users to manage data based on frequency of access and cost efficiency. Using **lifecycle management policies**, data can automatically transition between tiers, optimizing storage costs over time.

In this project, Azure Blob Storage was used to:

- Host a static website using the $web container
- Enable public access to view the site through a browser
- Apply a lifecycle rule to move unused blobs to the Cool tier after 100 days

This service was essential in learning how modern cloud platforms can simplify web hosting and automate data lifecycle operations.

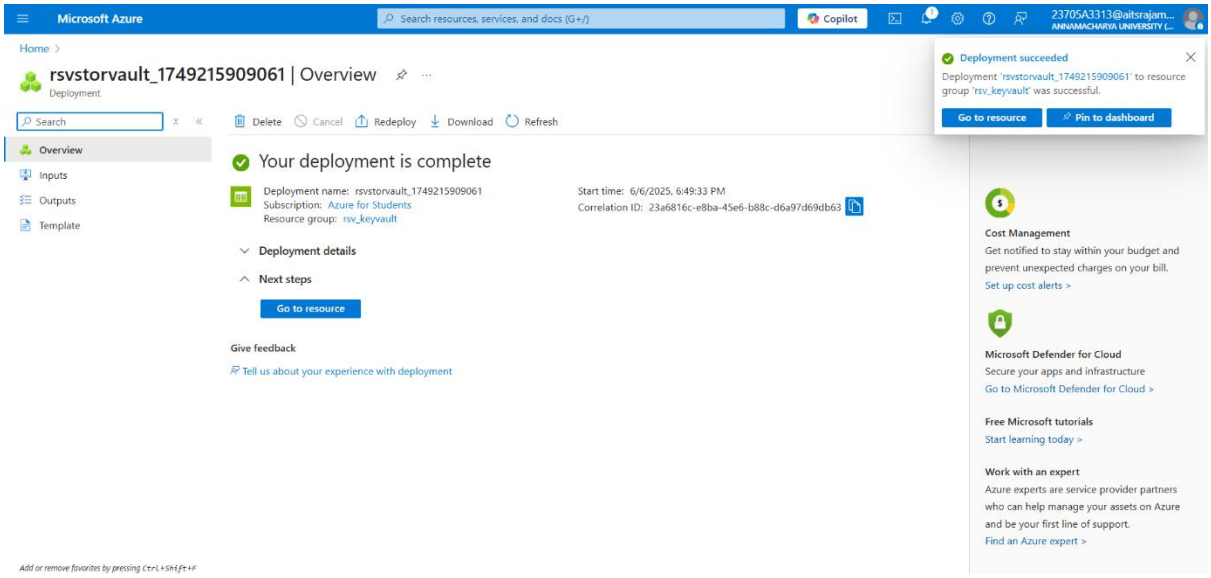


Figure 4.1: Azure Storage Account Dashboard Overview

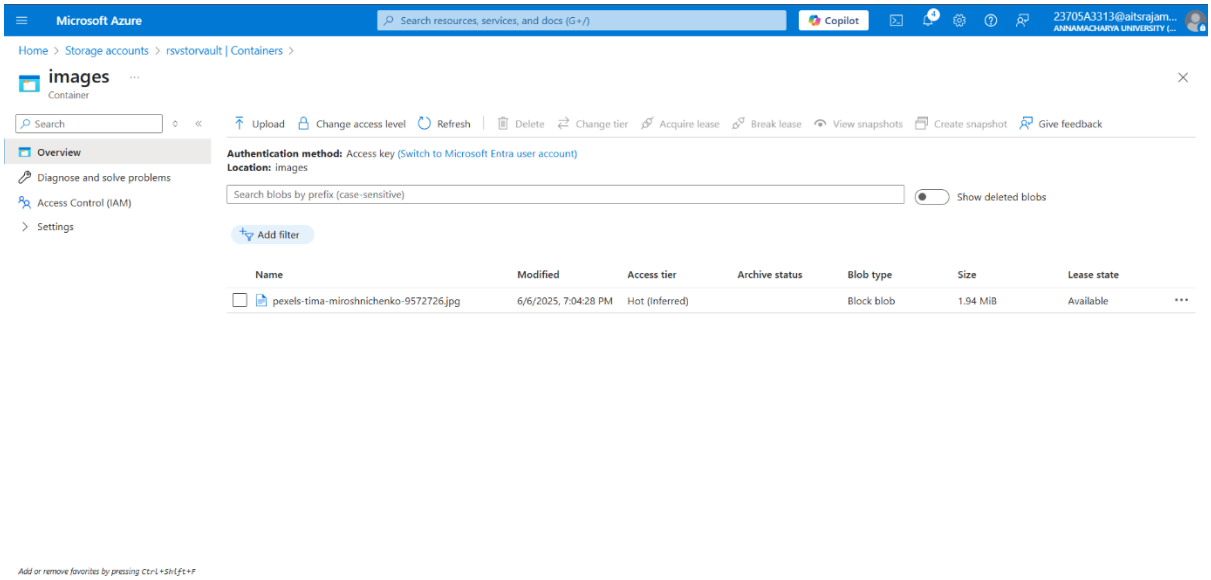


Figure 4.2: Image File Uploaded to Blob Storage Container

CHAPTER 5: IMPLEMENTATION STEPS

This chapter outlines the step-by-step implementation of hosting a static website using Azure Blob Storage and configuring a lifecycle management policy. The project was completed using the Azure Portal.

Step 1: Create a Storage Account

1. Log in to https://portal.azure.com
2. Search for “Storage accounts” and click **+ Create**
3. Fill in required details:
    - Subscription, Resource Group, Storage Account Name, Region
4. Leave default settings for performance and redundancy
5. Click **Review + Create → Create**

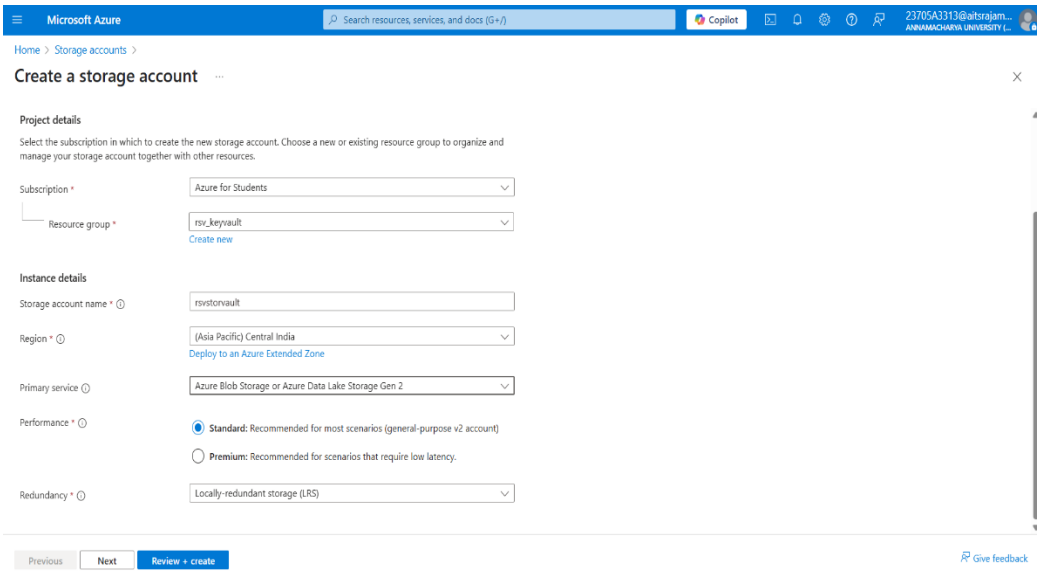


Figure 5.1: Created a new Azure Storage Account

Step 2: Enable Static Website Hosting

1. In the newly created storage account, go to **Static website** under **Data management**
2. Click **Enable**
3. Upload the file names:
    - **Index document name:** index.html
    - **Error document path:** error.html
4. Save the settings

This creates a special container called **$web**

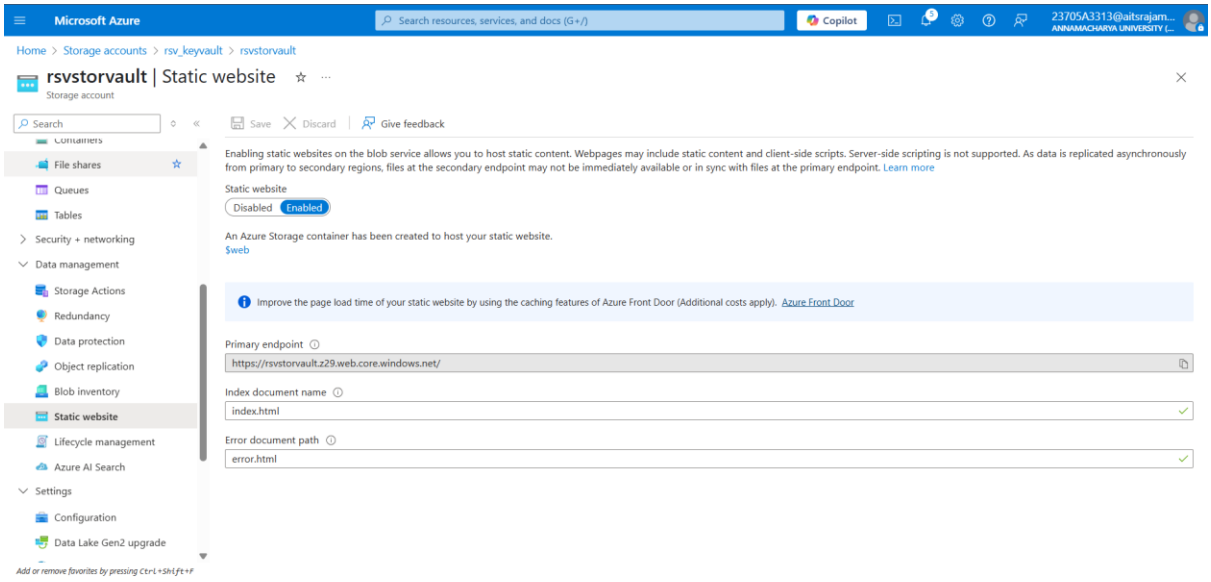


Figure 5.2: Enabling Static Website Feature in Azure Portal

- **Step 3: Upload HTML Files to $web**

1. Go to **Containers** → Select **$web**
2. Click **Upload**
3. Select index.html and error.html files from your system
4. Click **Upload** to publish them to the container

File Edit View

```
<!DOCTYPE html>
<html>
<head>
    <title>Page Not Found</title>
</head>
<body style="text-align:center; margin-top:50px; color:red;">
    <h1>404 - Page Not Found</h1>
    <p>The page you're looking for doesn't exist.</p>
</body>
</html>
```

File Edit View

```
<!DOCTYPE html>
<html>
<head>
    <title>Welcome to My Static Website</title>
</head>
<body style="text-align:center; margin-top:50px;">
    <h1>Welcome to My Azure Static Website!</h1>
    <p>This page is served from an Azure Blob Storage container.</p>
</body>
</html>
```

Figure 5.3: 'index.html' and 'error.html'

- **Step 4: Test Public Endpoint**

1. After uploading, go back to **Static Website** section
2. Copy the **Primary endpoint URL**
3. Paste the URL into a browser to test:
    - Your index.html page should load
    - Visit a wrong path to check error.html

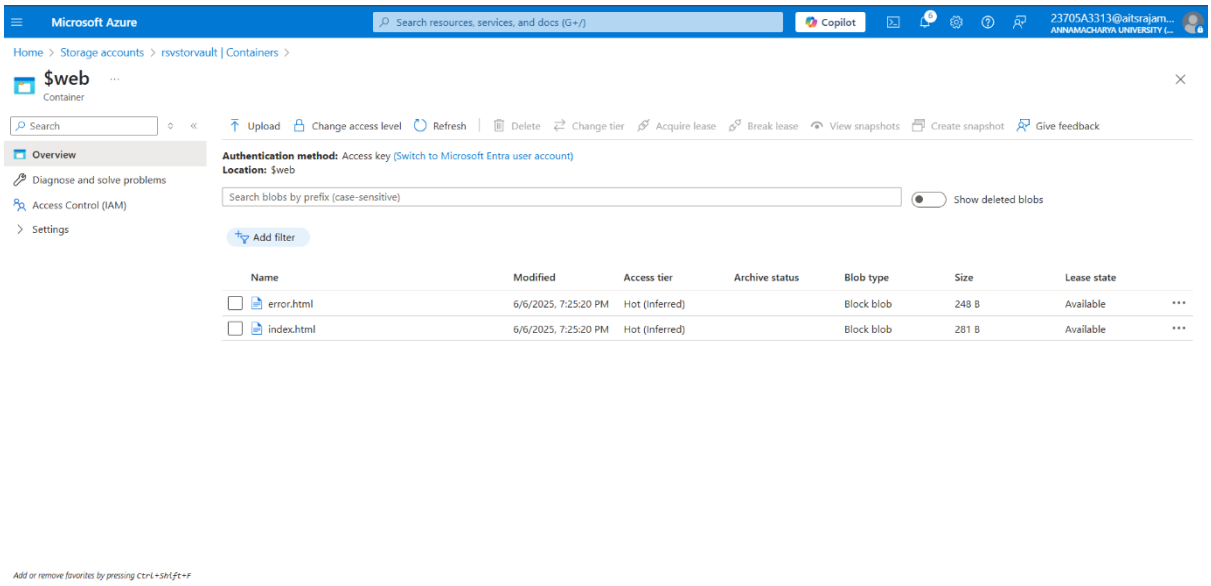


Figure 5.4: Test Public Endpoint

- **Step 5: Configure Lifecycle Management**

1. In the storage account menu, click **Lifecycle management**
2. Click **+ Add a rule**
3. Set the rule name (e.g., cool-tier-rule)
4. Under **Blob type**: choose **Block blob**
5. Add a condition:
    - If blob is **more than 100 days old**, move to **Cool tier**
6. Click **Review + Add**

This rule ensures older blobs are automatically moved to a lower-cost storage tier.

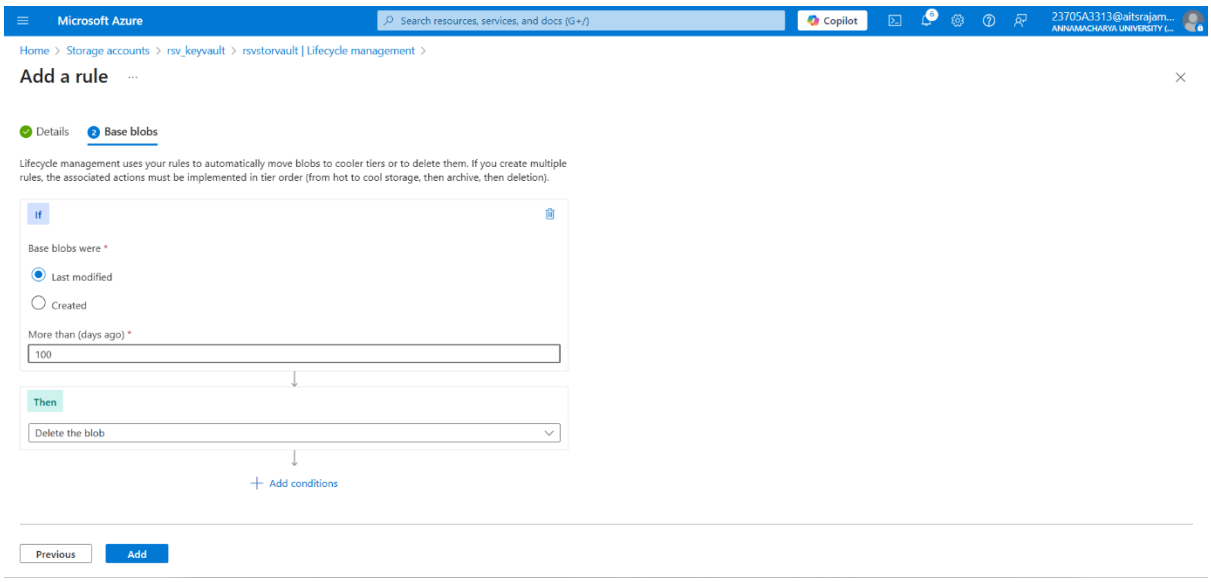


Figure 5.5: Lifecycle Rule Configuration – Final Rule Summary

CHAPTER 6: SCREENSHOTS

This chapter contains visual documentation of each key implementation step. The following screenshots were captured from the Azure Portal during the execution of the project.

Figure 6.1 – Creating a New Azure Storage Account

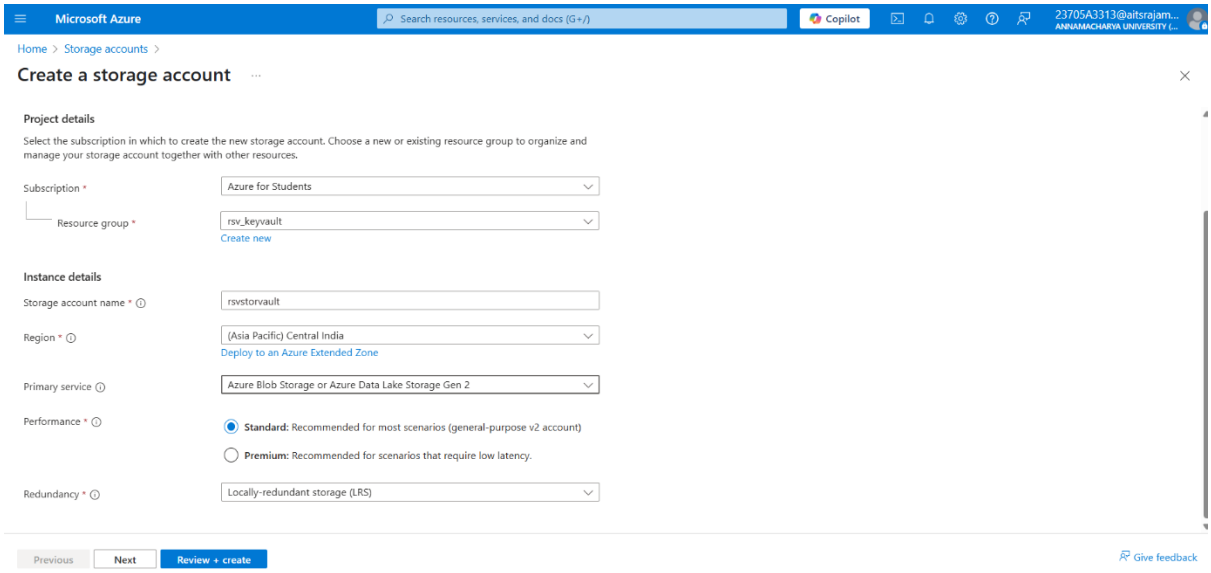


Figure 6.2 – Enabling Static Website Hosting

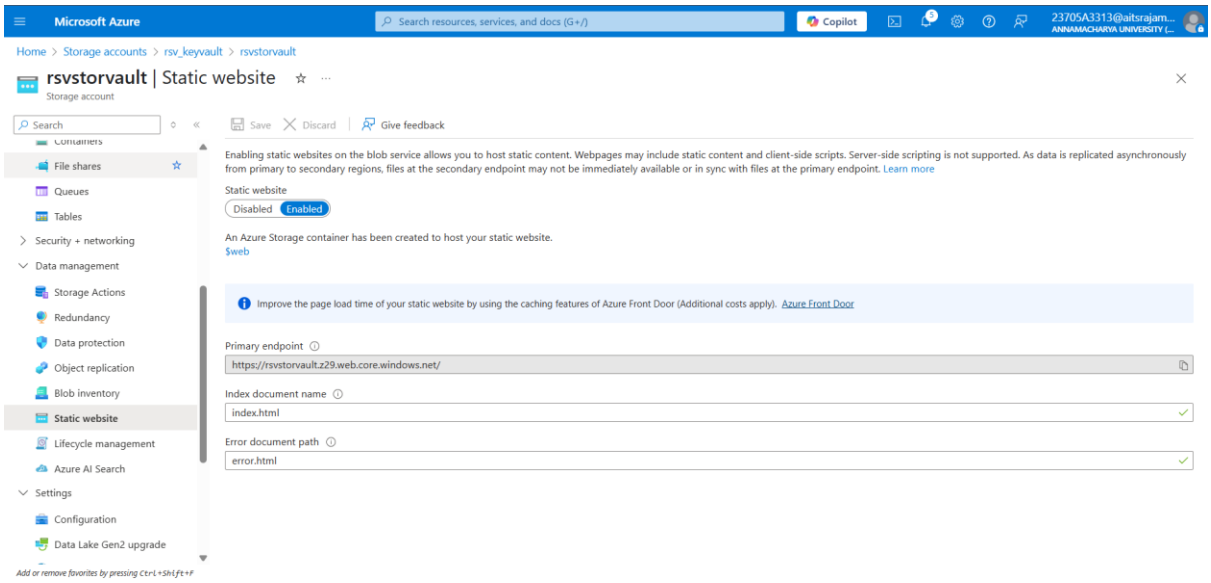


Figure 6.3 – Uploading HTML Files to $web Container

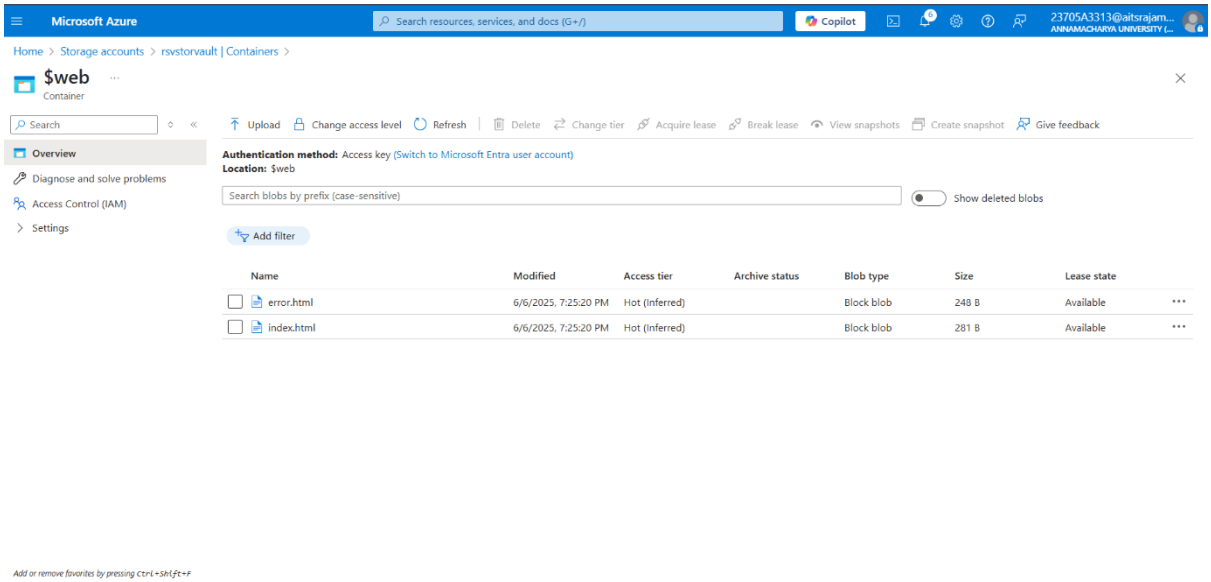


Figure 6.4 – Testing Public Endpoint in Browser

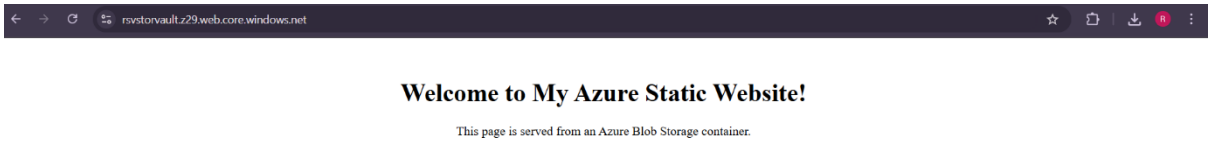
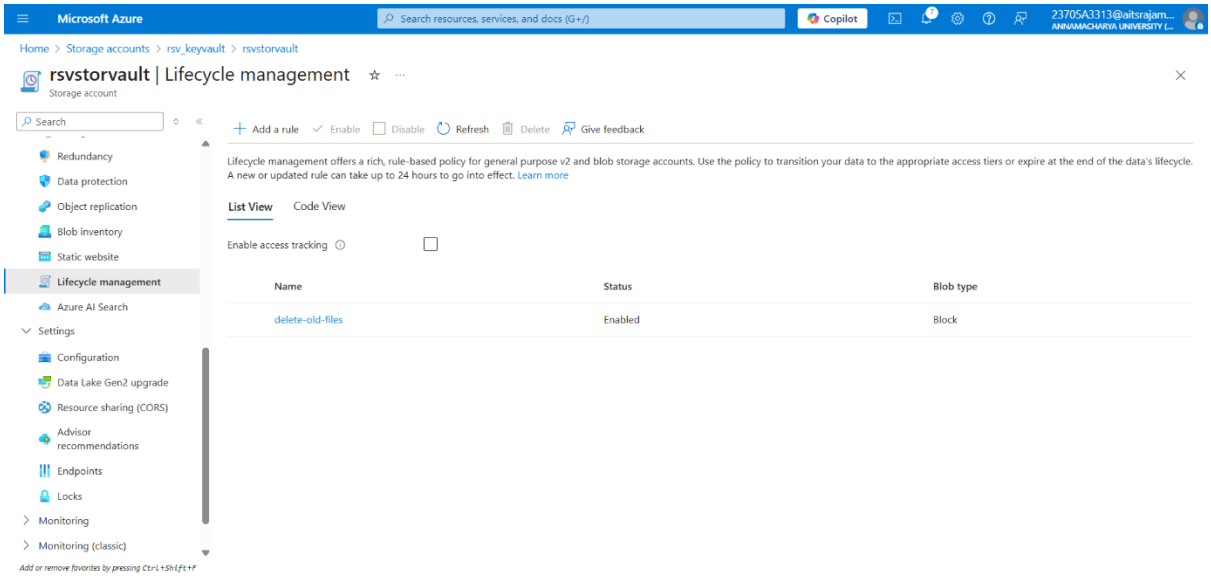


Figure 6.5 – Configuring Lifecycle Rule



CHAPTER 7: RESULT

The project was successfully implemented using Microsoft Azure Blob Storage. A fully functional static website was hosted through Azure's $web container and made publicly accessible via a browser endpoint. The uploaded index.html file loaded as the homepage, and the error.html file was triggered appropriately when invalid paths were accessed.

In addition, a lifecycle management rule was configured to automatically transition blobs to the Cool storage tier after 100 days of inactivity. This automation supports long-term storage optimization and cost control in cloud environments.

All configuration steps were executed through the Azure Portal without any errors. The result demonstrates that Azure Blob Storage can efficiently support static site hosting as well as intelligent data management through lifecycle policies.

Screenshots were captured at each stage to validate the process and are included in the report for reference.

CHAPTER 8: CONCLUSION

This project provided valuable hands-on experience in leveraging **Microsoft Azure Blob Storage** to host static websites and manage unstructured data efficiently in the cloud. By enabling static website hosting, uploading HTML files, and configuring public access, I was able to understand the practical use of Azure Storage Accounts for real-world web deployment.

The implementation of a **lifecycle management policy** further demonstrated how cloud resources can be optimized automatically over time, contributing to cost-effectiveness and better resource planning. This rule-based automation is especially beneficial in production environments with large volumes of data.

Through this task, I gained practical skills in working with storage containers, blob types, public endpoint testing, and configuring data retention rules using Azure's built-in services. It also gave me insight into how developers and organizations can reduce operational overhead using platform-as-a-service (PaaS) offerings.

Overall, this project has improved my understanding of cloud storage architecture, introduced me to essential Azure features, and prepared me to handle similar cloud-hosting challenges in future academic or professional environments.

In addition, I gained exposure to the importance of cloud-native tools in simplifying deployments and managing large-scale content delivery without the need for traditional servers. The process of hosting a site with just a storage account demonstrated Azure’s capability for serverless web applications.

This experience also emphasized the importance of monitoring and automation in long-term cloud storage planning. Learning to transition blobs based on age or usage helped me understand how intelligent storage decisions reduce unnecessary costs.

The skills I gained from this project are not only technically valuable, but also align with modern industry practices in cloud-based web hosting and storage optimization. It has boosted my confidence to work on more advanced Azure services in future projects or roles.

CHAPTER 9: FUTURE ENHANCEMENTS

While the current implementation effectively demonstrates how to host a static website using Azure Blob Storage and configure lifecycle management, there are several potential improvements that could enhance the functionality and scalability of the project:

- **1. Add Custom Domain & SSL**

Integrate a custom domain with the Azure Blob-hosted site and configure HTTPS using Azure CDN or Front Door to enhance branding and security.

- **2. Automate Deployment Using Azure CLI**

Instead of manual uploads, future deployments can be automated using Azure CLI or PowerShell scripts, making it easier to manage and scale updates.

- **3. Versioning of Blobs**

Enable blob versioning to maintain historical copies of files, which is useful for recovering previous versions or managing frequent content changes.

- **4. Advanced Lifecycle Rules**

Define multiple lifecycle rules based on blob prefixes or tags to handle different types of content (e.g., move images to archive tier after 30 days, HTML after 90).

- **5. Monitor Blob Usage with Azure Monitor**

Integrate with Azure Monitor and set up alerts to track usage patterns, access logs, and storage costs in real-time.

CHAPTER 10: REFERENCES

The following official resources and documentation were used to guide the implementation of this project:

1. Microsoft Learn – *Host a static website using Azure Blob Storage* 🔗 https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blob-static-website
2. Microsoft Learn – *Lifecycle management for Azure Blob Storage* 🔗 https://learn.microsoft.com/en-us/azure/storage/blobs/lifecycle-management-policy-configure
3. Microsoft Azure Documentation – *Azure Storage Account Overview* 🔗 https://learn.microsoft.com/en-us/azure/storage/common/storage-account-overview
4. Microsoft Azure Portal – https://portal.azure.com