A Web-Based Meditation Timer

Enhancing Mindfulness Through Technology

Date :- 13th May 2025

Team Name :- Code Warriors

Team Lead

Aathira V Sajive

Vrushali Mujumdar

Developers

Prasad Gujarathi

Vaishnavi Dhobale

Pooja Kothawade

Cloud Engineers

Abhishek Choudhary

Gaurav Chaudhari

Aadheesh Phade

Documentation

Pramod Patil (Developer)

Prajakta Wanjari (Developer)

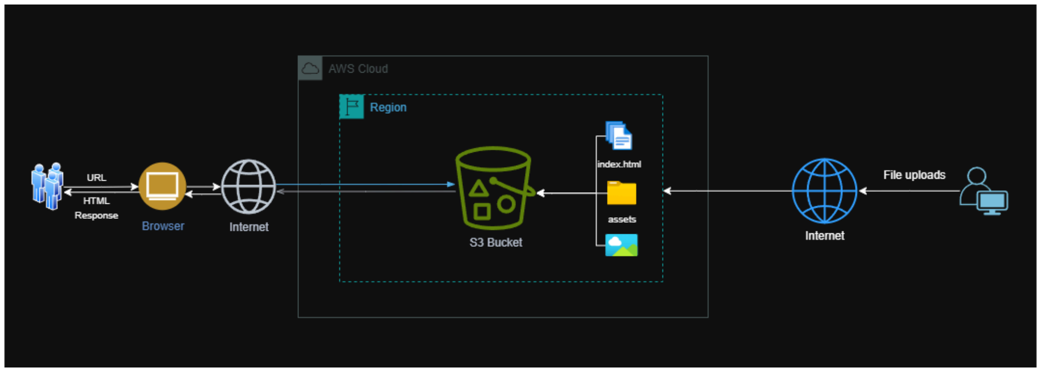
Parth Satish (Cloud Engineer)

Application Link: http://codewarriorshackathon.s3-website.ap-south-1.amazonaws.com/

GitHub Link: https://github.com/VaishnaviDhobale/Code-Warriors

Project Overview

The Mindfulness Timer is a web application designed to help users practice mindfulness and meditation through timed sessions. Users can choose predefined time durations and receive a gentle bell sound when the timer ends. This app aims to promote mental well-being and encourage brief, meaningful breaks.



Tech Stack

Frontend: HTML, CSS, JavaScript

Cloud Hosting: AWS S3 (Static Website Hosting)

Tools: Visual Studio Code, GitHub, AWS Management Console

Hosting & Deployment

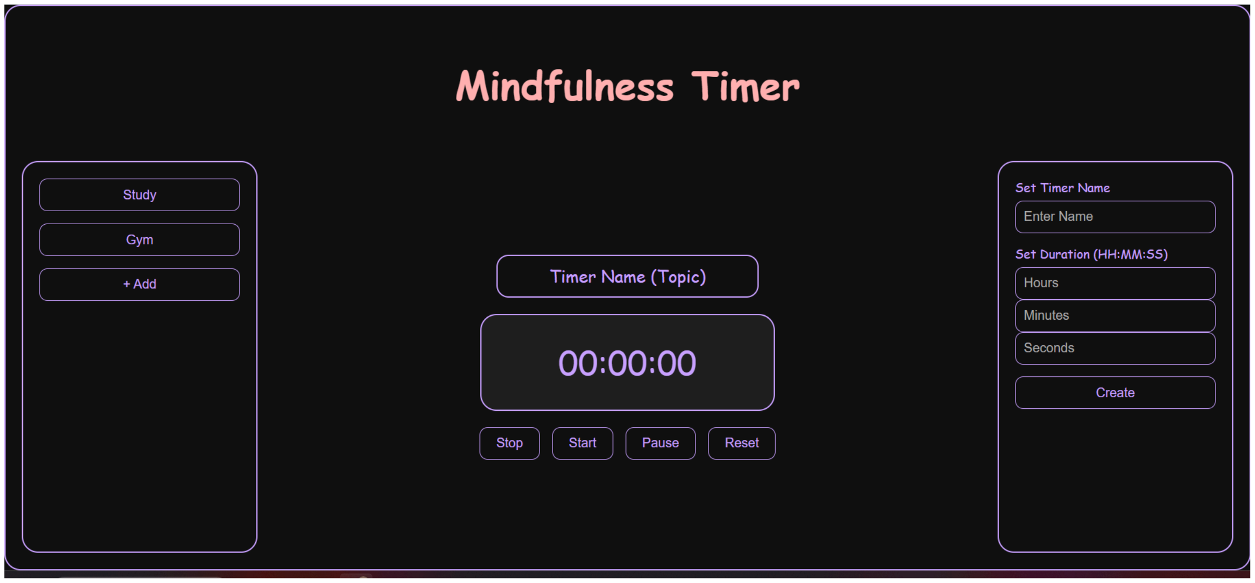
The project is deployed using Amazon S3 static website hosting.

All static assets are served directly from S3.

Permissions and policies were configured to allow secure public access.

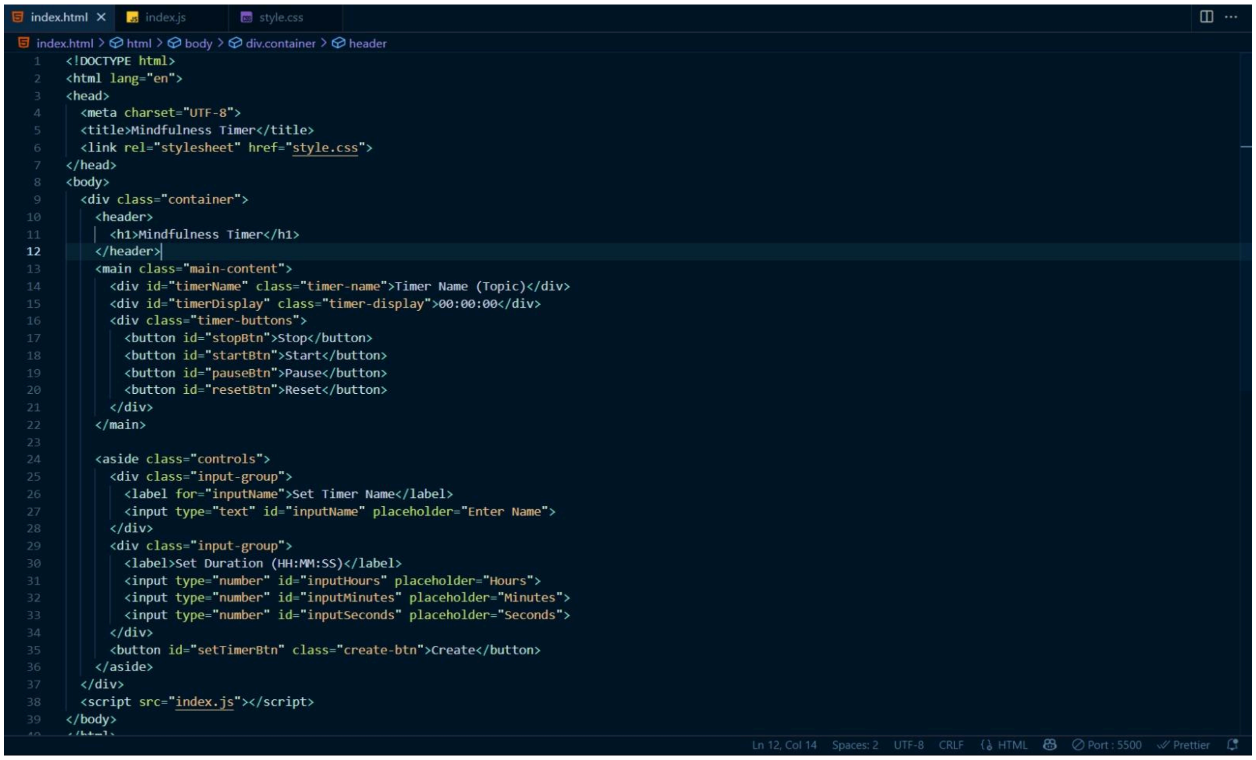
Implementation Details

- JavaScript's `setTimeout()` and `setInterval()` for timing logic
- LocalStorage for saving preference



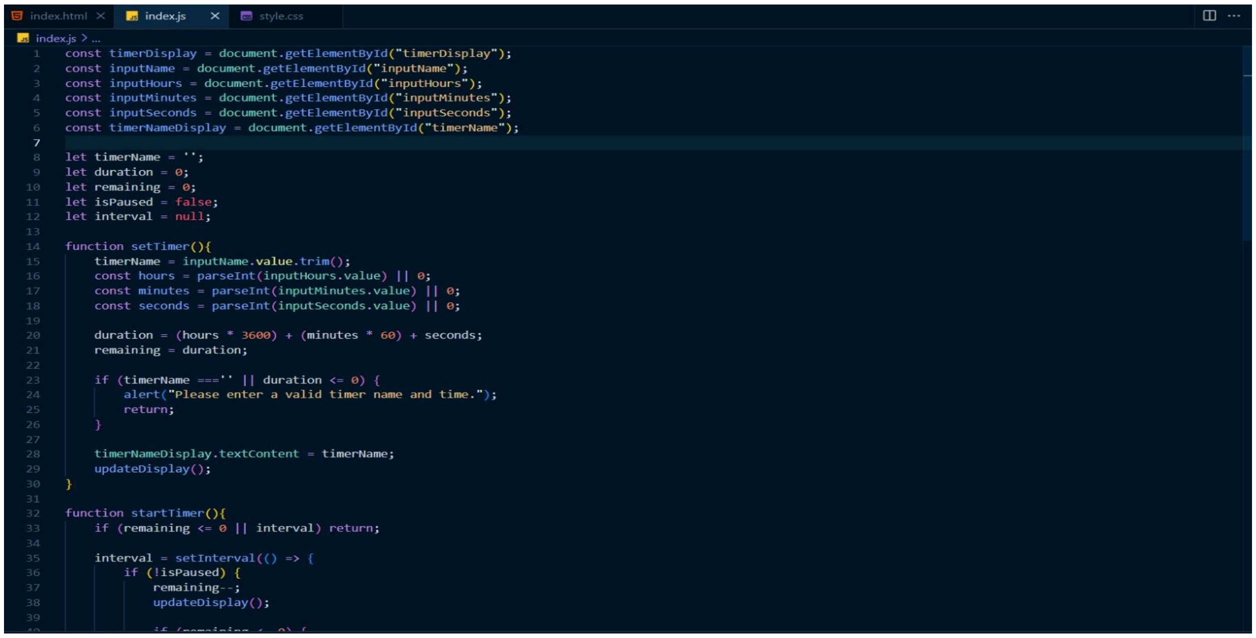
Coding documentation

Index.html file



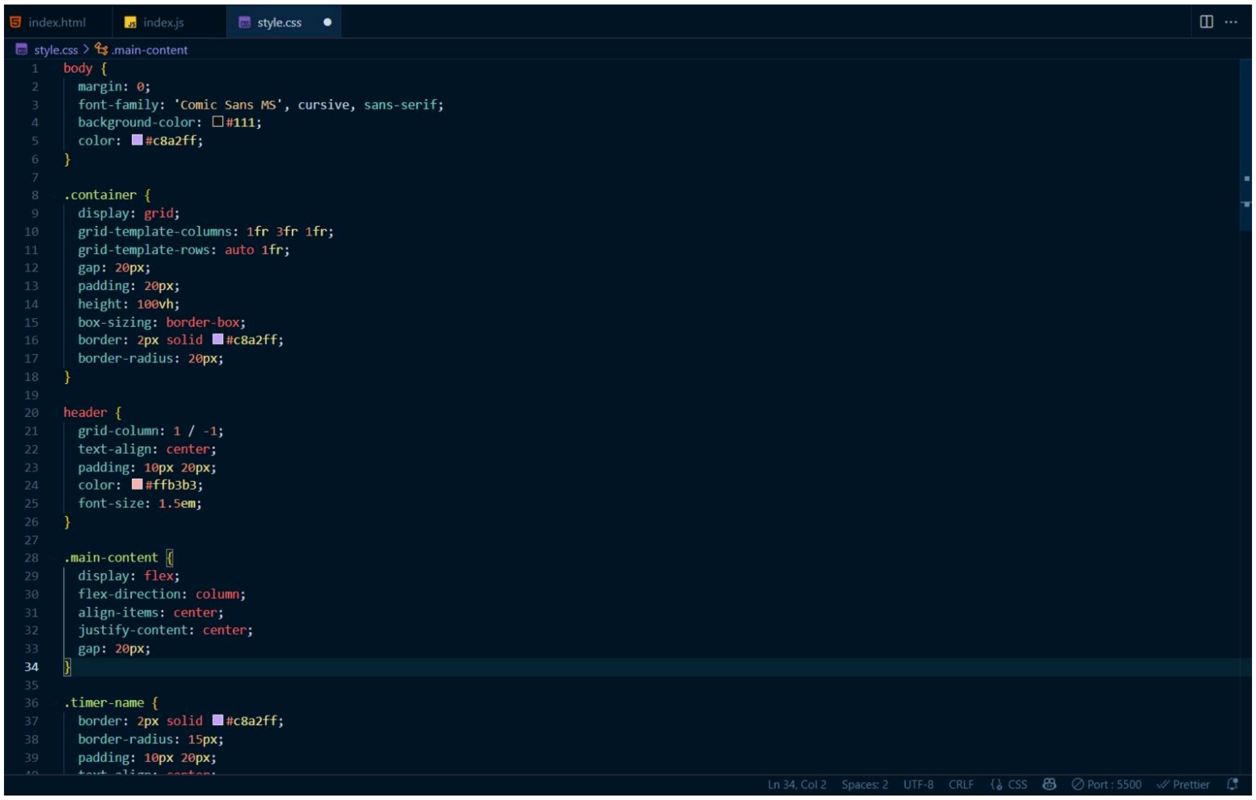
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Mindfulness Timer</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <header>
      <h1>Mindfulness Timer</h1>
    </header>
    <main class="main-content">
      <div id="timerName" class="timer-name">Timer Name (Topic)</div>
      <div id="timerDisplay" class="timer-display">00:00:00</div>
      <div class="timer-buttons">
        <button id="stopBtn">Stop</button>
        <button id="startBtn">Start</button>
        <button id="pauseBtn">Pause</button>
        <button id="resetBtn">Reset</button>
      </div>
    </main>

    <aside class="controls">
      <div class="input-group">
        <label for="inputName">Set Timer Name</label>
        <input type="text" id="inputName" placeholder="Enter Name">
      </div>
      <div class="input-group">
        <label>Set Duration (HH:MM:SS)</label>
        <input type="number" id="inputHours" placeholder="Hours">
        <input type="number" id="inputMinutes" placeholder="Minutes">
        <input type="number" id="inputSeconds" placeholder="Seconds">
      </div>
      <button id="setTimerBtn" class="create-btn">Create</button>
    </aside>
  </div>
  <script src="index.js"></script>
</body>
```

Javascript files



```javascript
const timerDisplay = document.getElementById("timerDisplay");
const inputName = document.getElementById("inputName");
const inputHours = document.getElementById("inputHours");
const inputMinutes = document.getElementById("inputMinutes");
const inputSeconds = document.getElementById("inputSeconds");
const timerNameDisplay = document.getElementById("timerName");

let timerName = '';
let duration = 0;
let remaining = 0;
let isPaused = false;
let interval = null;

function setTimer(){
    timerName = inputName.value.trim();
    const hours = parseInt(inputHours.value) || 0;
    const minutes = parseInt(inputMinutes.value) || 0;
    const seconds = parseInt(inputSeconds.value) || 0;

    duration = (hours * 3600) + (minutes * 60) + seconds;
    remaining = duration;

    if (timerName ===''  || duration <= 0) {
        alert("Please enter a valid timer name and time.");
        return;
    }

    timerNameDisplay.textContent = timerName;
    updateDisplay();
}

function startTimer(){
    if (remaining <= 0 || interval) return;

    interval = setInterval(() => {
        if (!isPaused) {
            remaining--;
            updateDisplay();

```

Stylesheet(CSS) Files

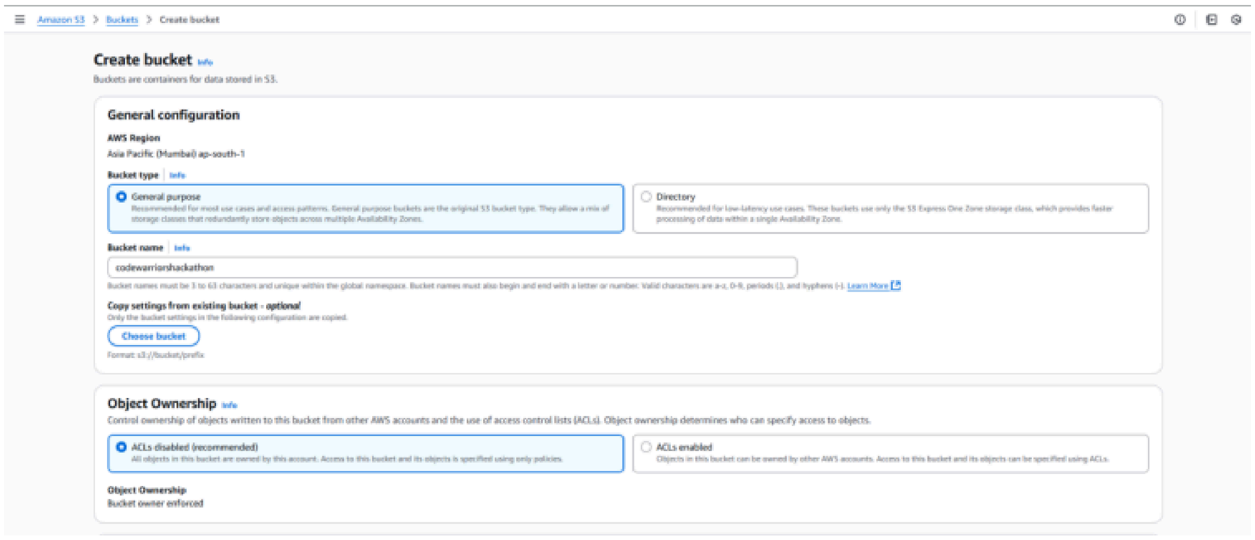

index.html
index.js
style.css
style.css > .main-content
Ln 34, Col 2   Spaces: 2   UTF-8   CRLF   CSS   Port : 5500   Prettier

```css
body {
  margin: 0;
  font-family: 'Comic Sans MS', cursive, sans-serif;
  background-color: #111;
  color: #c8a2ff;
}

.container {
  display: grid;
  grid-template-columns: 1fr 3fr 1fr;
  grid-template-rows: auto 1fr;
  gap: 20px;
  padding: 20px;
  height: 100vh;
  box-sizing: border-box;
  border: 2px solid #c8a2ff;
  border-radius: 20px;
}

header {
  grid-column: 1 / -1;
  text-align: center;
  padding: 10px 20px;
  color: #ffb3b3;
  font-size: 1.5em;
}

.main-content {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  gap: 20px;
}

.timer-name {
  border: 2px solid #c8a2ff;
  border-radius: 15px;
  padding: 10px 20px;
```

Set Up AWS Environment

Step 1: Creating an S3 bucket in AWS Console.

- Go to Aws Console
- Search for S3
- Create New S3 Bucket
- Select the Bucket type As General purpose.
- Unmark the Block Public Access Settings for this Bucket so that Everyone can access from the URL.
- Now Click on Create Bucket.
- The new Bucket is Now Available.



Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region
Asia Pacific (Mumbai) ap-south-1

Bucket type Info

[✓ General purpose]
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info
codewarriorshackathon
Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). Learn More

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.

Format: s3://bucket/prefix

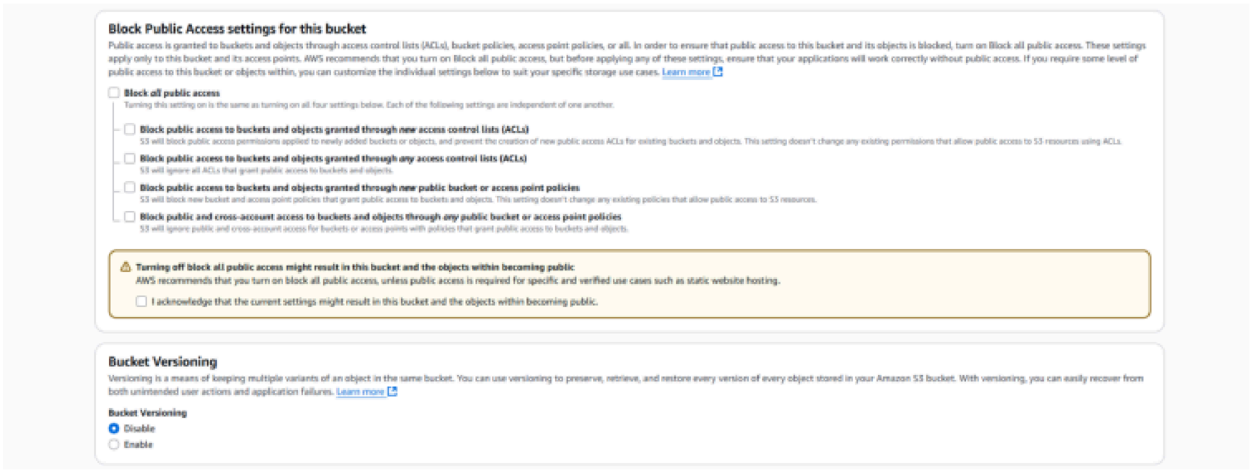
Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

[✓ ACLs disabled (recommended)]
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced



Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

- [ ] **Block *all* public access**
  Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
  - [ ] **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
    S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
  - [ ] **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
    S3 will ignore all ACLs that grant public access to buckets and objects.
  - [ ] **Block public access to buckets and objects granted through *new* public bucket or access point policies**
    S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
  - [ ] **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
    S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

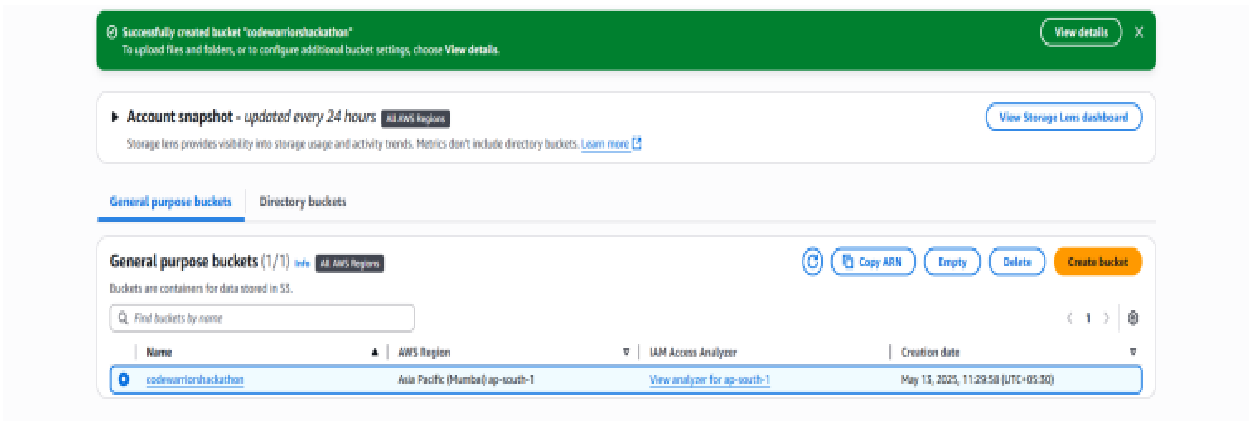
Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

- [ ] I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning
[✓ Disable]
Enable

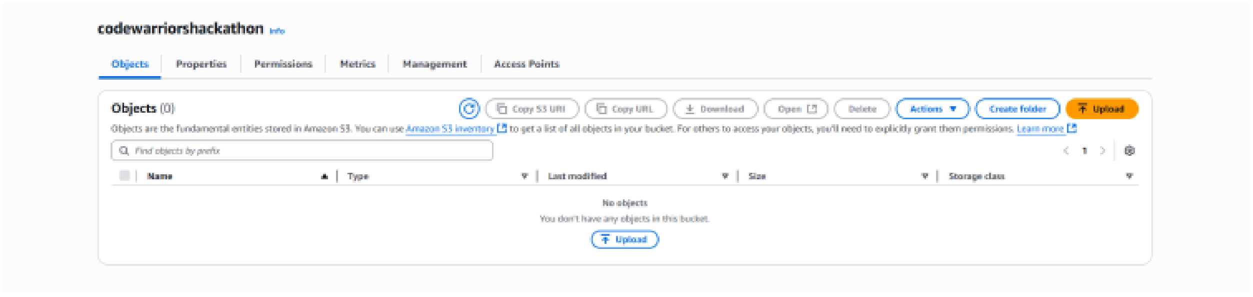


Successfully created bucket "codewarriorshackathon"
To upload files and folders, or to configure additional bucket settings, choose **View details**.

Account snapshot - *updated every 24 hours* All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Metrics don't include directory buckets. Learn more

General purpose buckets (1/1) Info All AWS Regions
Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
codewarriorshackathon	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	May 13, 2025, 11:29:58 (UTC+05:30)



codewarriorshackathon Info

Objects (0)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class

No objects
You don't have any objects in this bucket.

Step 2: Add Policy

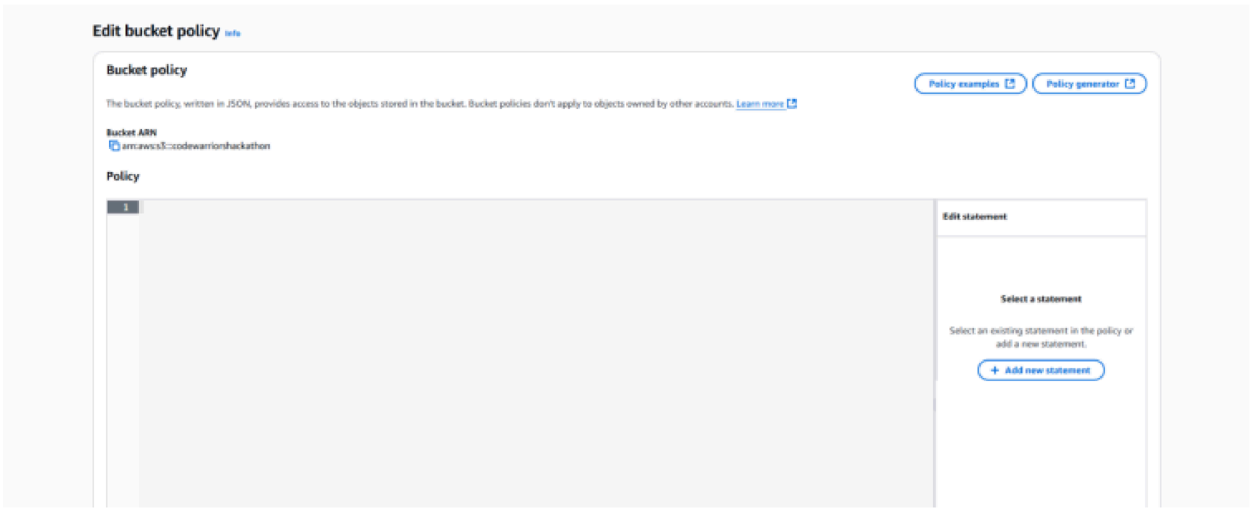
An S3 bucket policy in AWS is a resource-based policy that grants access permissions to your S3 bucket and the Object within it is the JSON document that defines permission for accessing a specific bucket.

- Click on Edit Bucket Policy
- Click on Policy Generator.
- Select the Type as S3 Bucket Policy.
- In Add Statement Allow the Effect and Enter * in principal
- Enter the ARN which is available in bucket details.

The policy will be generated. Copy the policy and paste it.

Step 3: Public Block Access disable

If the Block Public Access bucket setting is ON By default, then turn it to OFF , so that it will Not Block Any Access Request


Edit bucket policy Info

Bucket policy

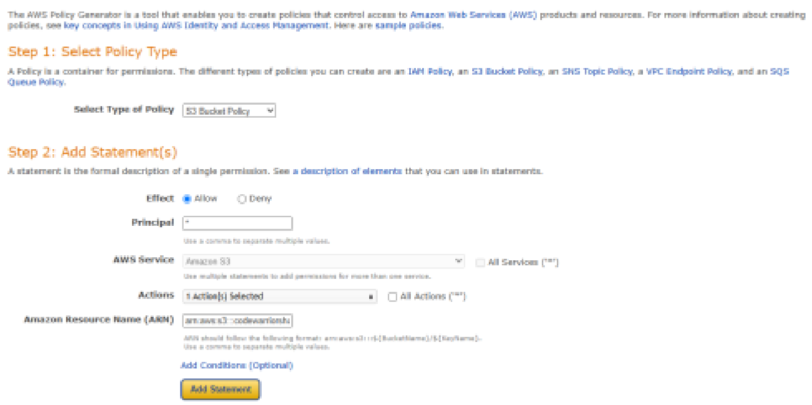
The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

Bucket ARN
arn:aws:s3:::codewarriorshackathon

Policy

Select a statement

Select an existing statement in the policy or add a new statement.


The AWS Policy Generator is a tool that enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see key concepts in Using AWS Identity and Access Management. Here are sample policies.

Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an IAM Policy, an S3 Bucket Policy, an SNS Topic Policy, a VPC Endpoint Policy, and an SQS Queue Policy.

Select Type of Policy: S3 Bucket Policy

Step 2: Add Statement(s)

A statement is the formal description of a single permission. See a description of elements that you can use in statements.

Effect: [✓ Allow] Deny

Principal: *

Use a comma to separate multiple values.

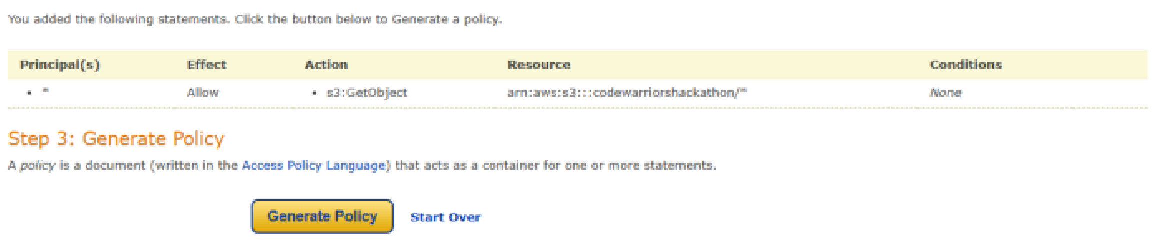
AWS Service: Amazon S3 ☐ All Services ('*')

Use multiple statements to add permissions for more than one service.

Actions: 1 Action(s) Selected ☐ All Actions ('*')

Amazon Resource Name (ARN): arn:aws:s3:::codewarriorshi

ARN should follow the following format: arn:aws:s3:::${BucketName}/${KeyName}.
Use a comma to separate multiple values.



You added the following statements. Click the button below to Generate a policy.

Principal(s)	Effect	Action	Resource	Conditions
• *	Allow	• s3:GetObject	arn:aws:s3:::codewarriorshackathon/*	*None*

Step 3: Generate Policy

A *policy* is a document (written in the Access Policy Language) that acts as a container for one or more statements.

For Improvement

While the some functionality was successfully implemented, the following areas are identified for future improvement:

1. Add Persistent Data Storage
Currently, no user data is saved (e.g., timer history or preferences).
Future versions can integrate a database (like AWS DynamoDB).

2. Save custom timers
Track daily meditation/study logs
Personalize user experience

3. Deployment Upgrades
Add CDN (via CloudFront) for faster global access
Enable HTTPS using AWS Certificate Manager and custom domain via Route 53

Note: Delete the resources once you are done with the project.

THANK YOU