Module	Assessment Type
Distributed and Cloud Systems Programming	Individual Report

Workshop 2

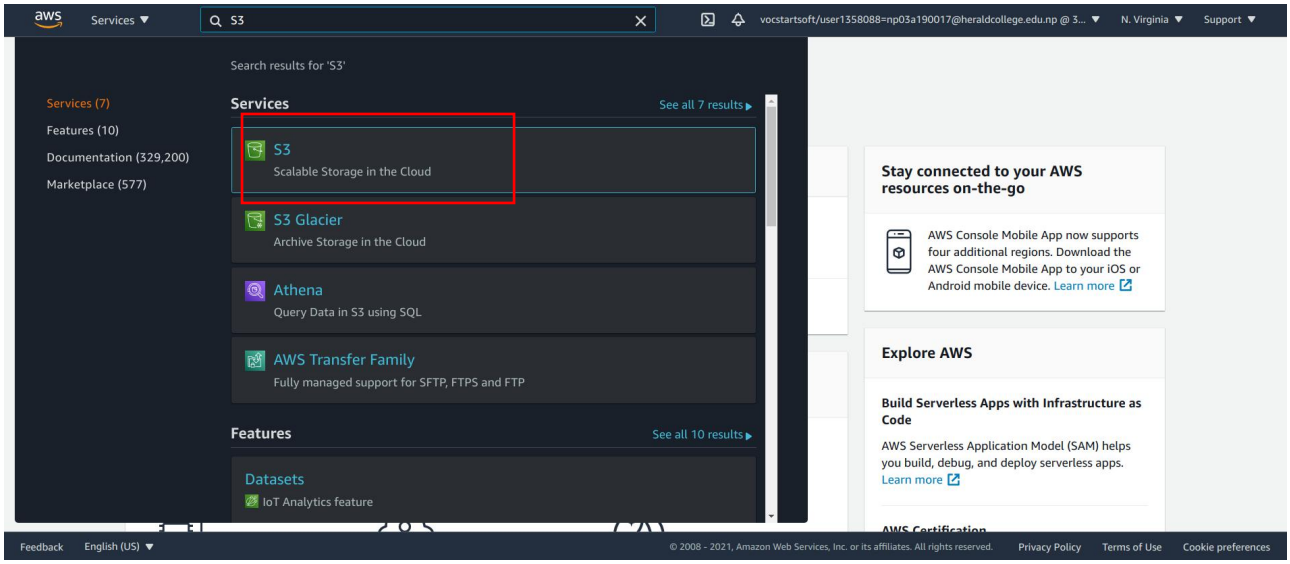
Student Id : 2049867 (NP03A190017)
Student Name : Roshan Parajuli
Section : L5CG3
Module Leader : Rupak Koirala
Lecturer /Tutor : Saroj Sharma
Submitted on : 2020-03-30

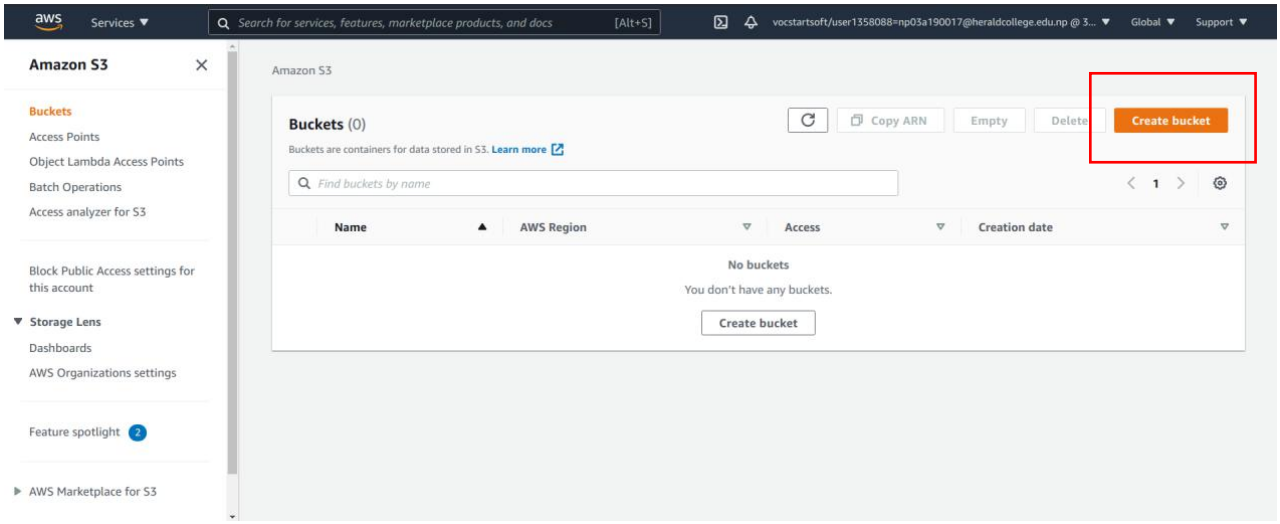
Table of Contents

Task 1: Static Web Hosting

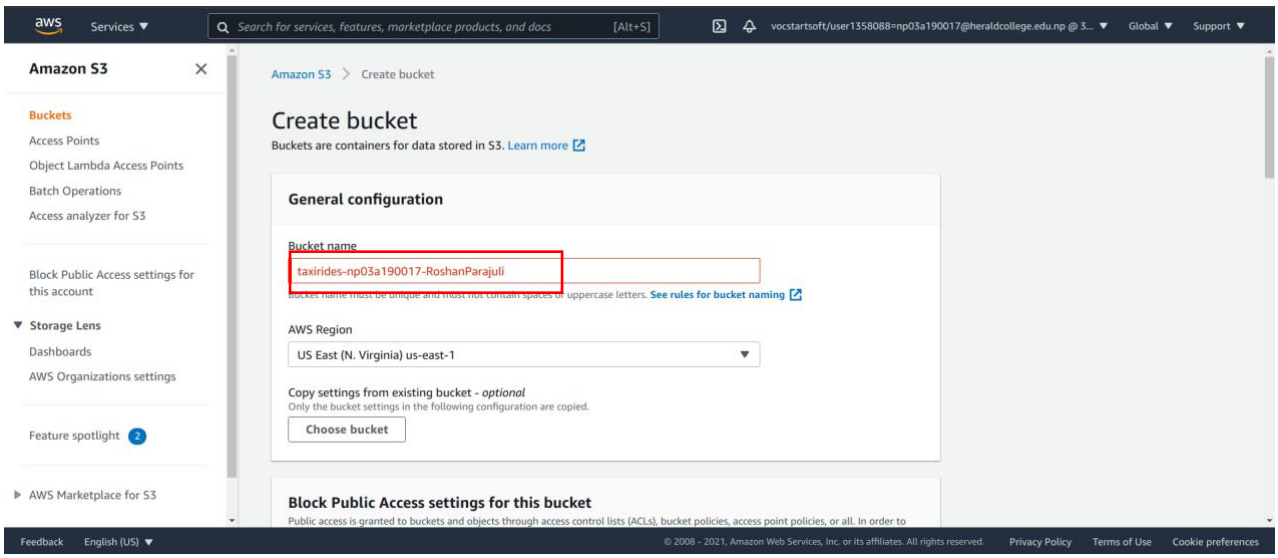
Creating an **S3 bucket** for hosting the static resources and pages for the web application. The dynamic functionalities are to be added later using Javascript to call remote RESTful APIs built with **AWS Lambda** and **Amazon API Gateway**.

First of all, S3 service is selected from the services section in AWS console.



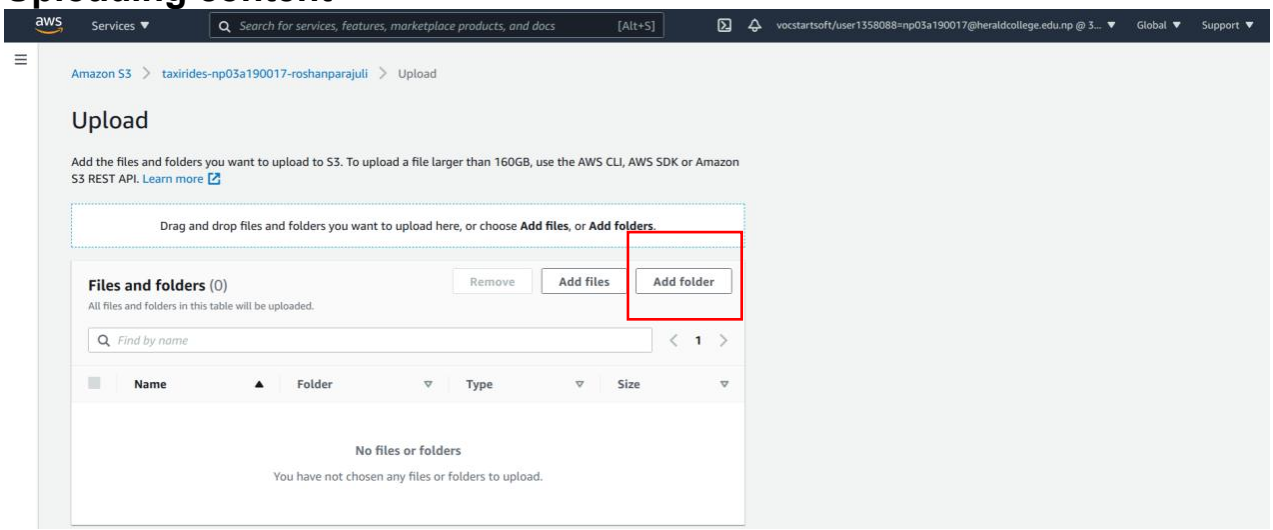


This is the page for the creation of an S3 bucket. A bucket with universally unique name is to be created for uploading the contents of a taxirides app and hosting the site.

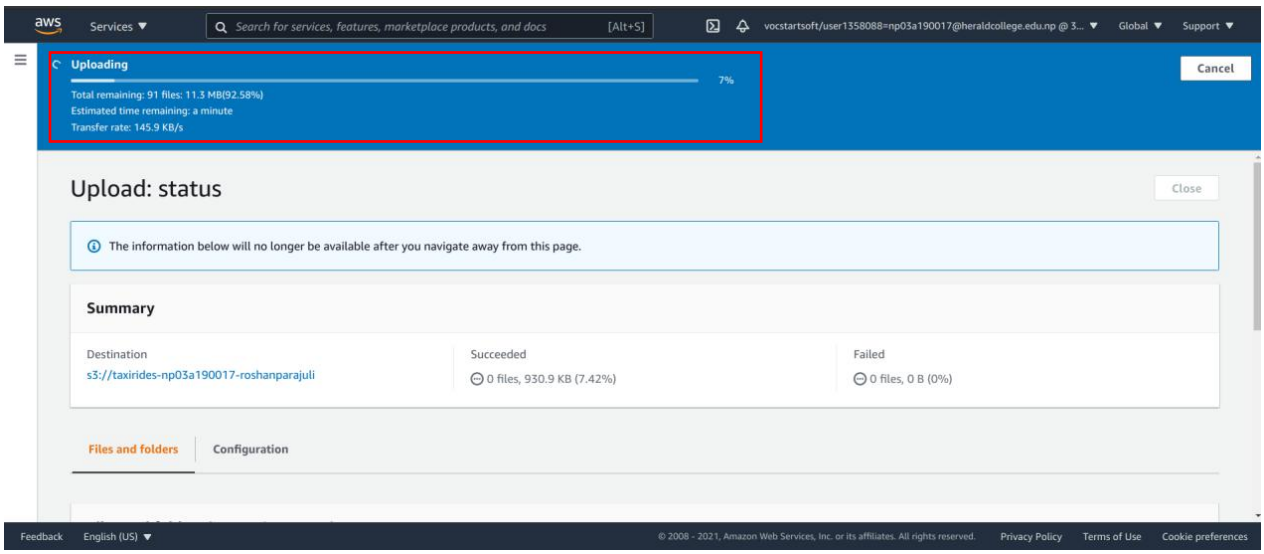


The region is selected to be US East. **AWS Regions** are large and widely dispersed into separate geographic locations with AWS Servers. After creating the bucket in a selected reason, the files are to be uploaded.

Uploading content

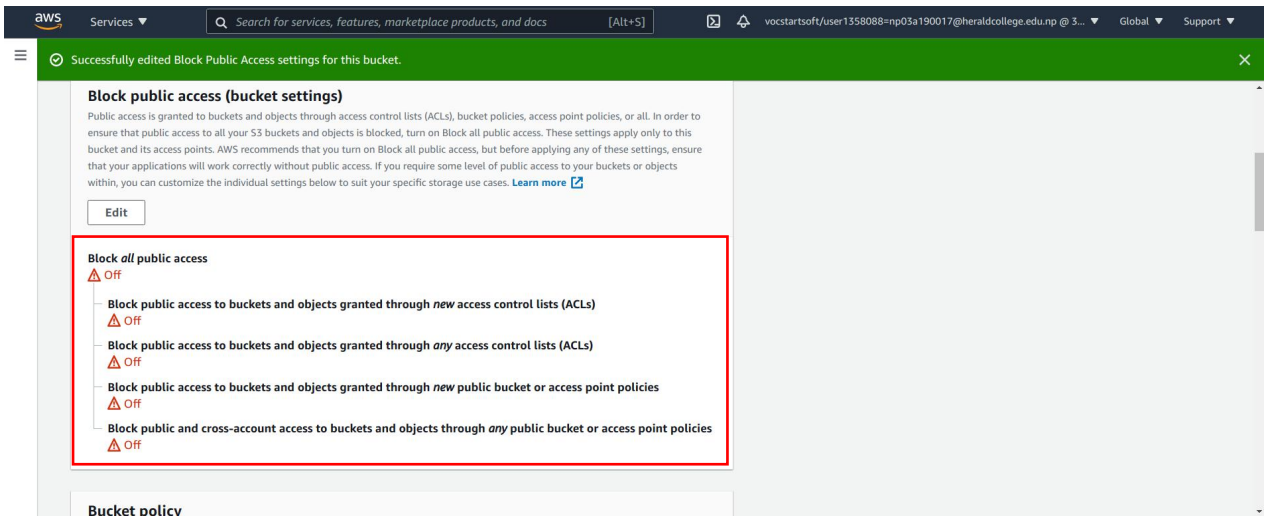


After clicking on the bucket's name, We can choose the "Upload" option and choose the required files for hosting the site.

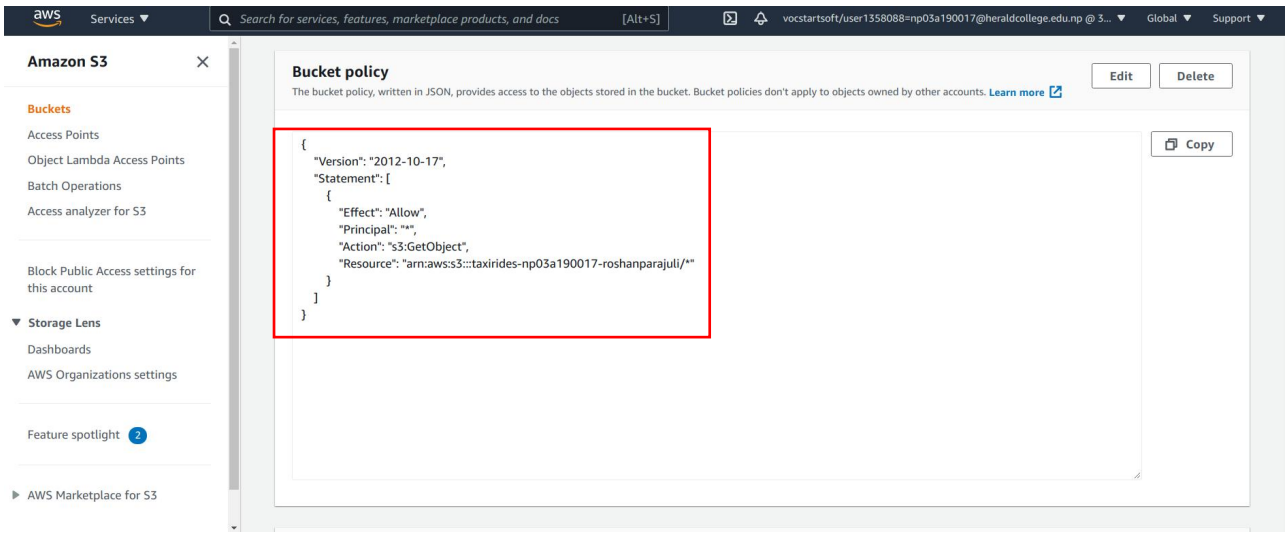


The files are being uploaded into the bucket and the progress is shown in the blue ribbon. After the upload is complete, the contents of the website directory listed in the S3 console.

Adding a Bucket Policy to Allow Public Reads

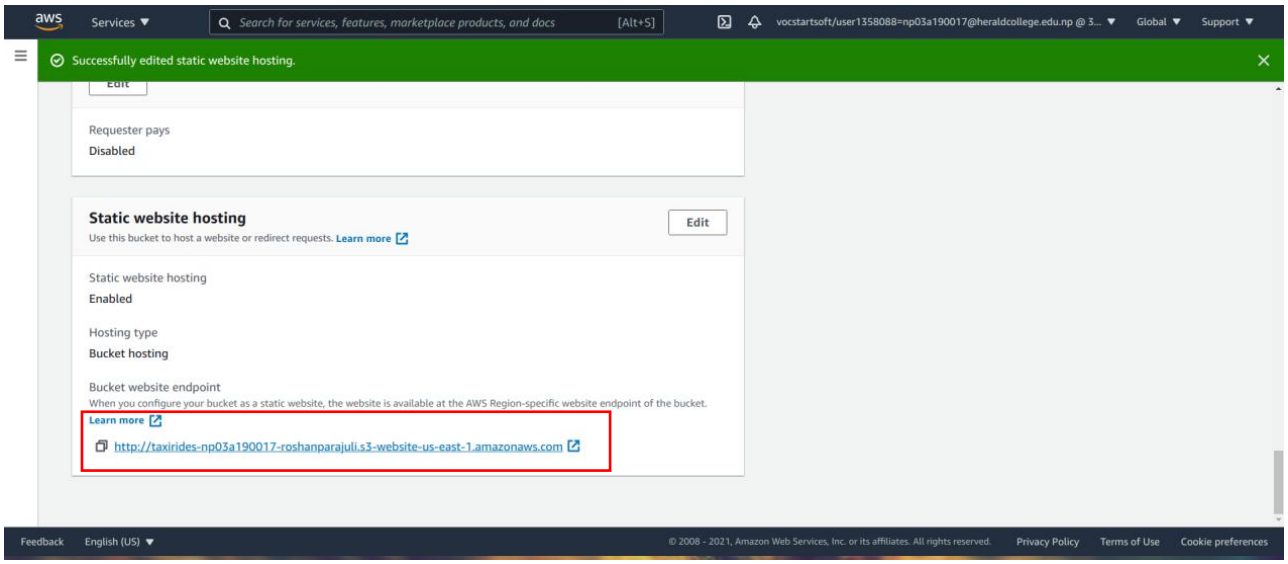


Since the website needs to be public, the public access is turned on the S3 bucket by adding a bucket policy and turning on all the related options. This enables the anonymous users to view the site. If the public access was not turned on, the bucket would only be accessible by authenticated users with access to the AWS account of the person who created the bucket.



Bucket policy is a JSON document that specifies what principles are allowed to execute various actions against the objects in the bucket.

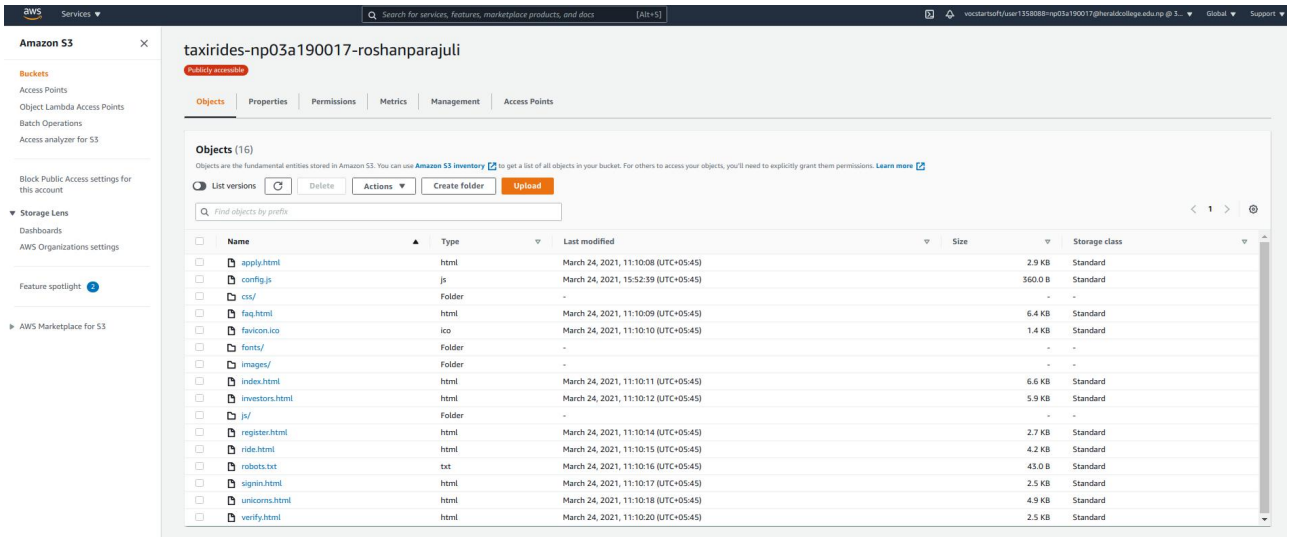
Enabling website hosting and testing the implementation



On the properties tab, at the end of the page, the domain was noted for the site deployed.

The site is now successfully deployed.

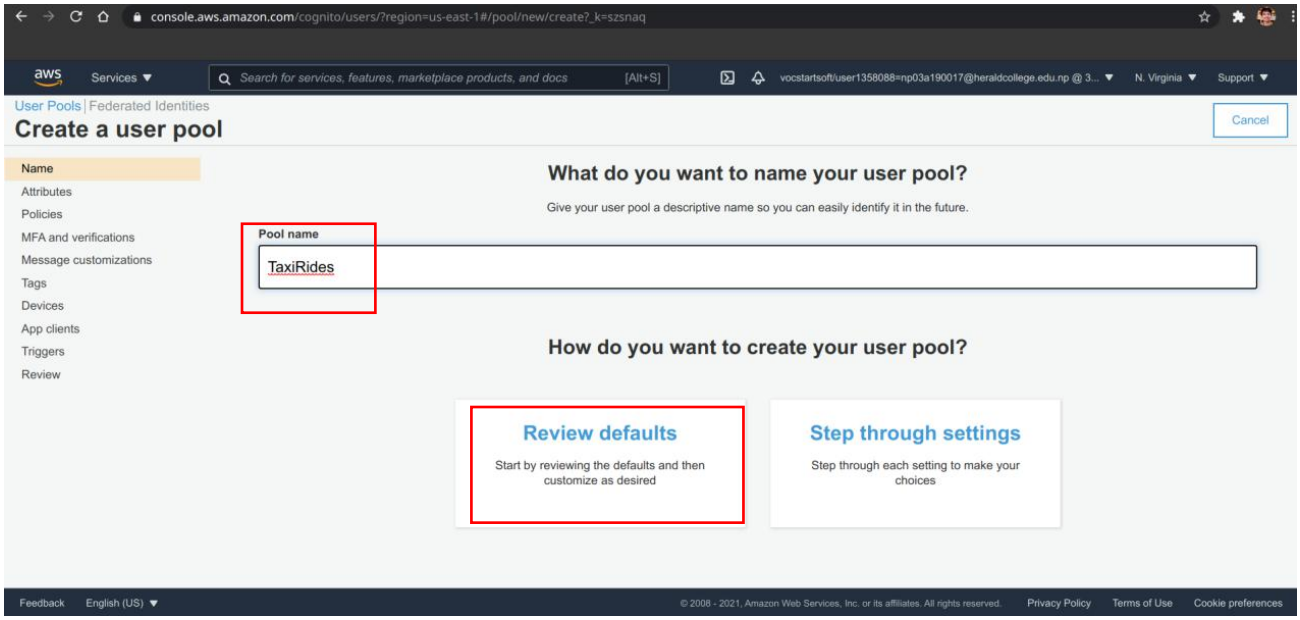
Screenshot of the S3 bucket



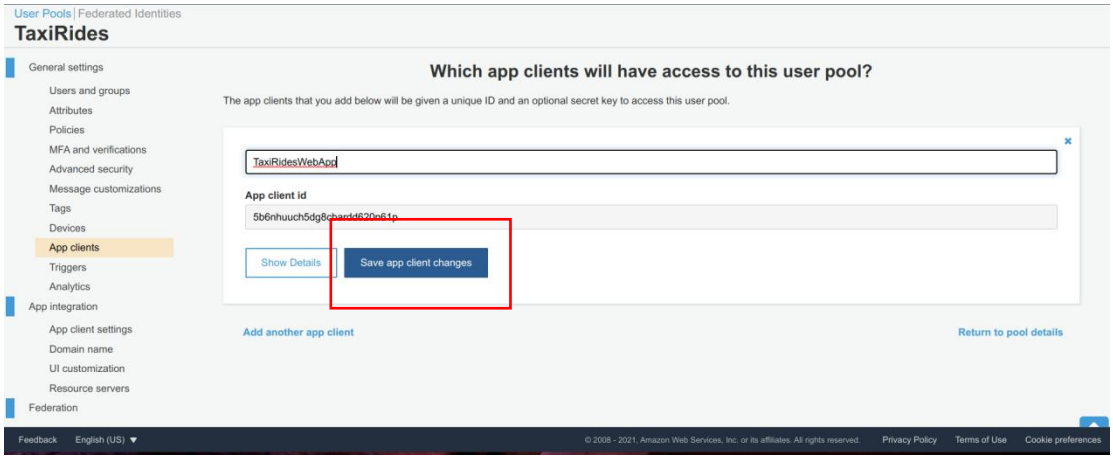
Task 2: User Management

Creating an Amazon Cognito User Pool

Cognito user pool service is used to add the authentication (sign-up and sign-in functionality) to the web app. For that, a new user pool needs to be created at first.



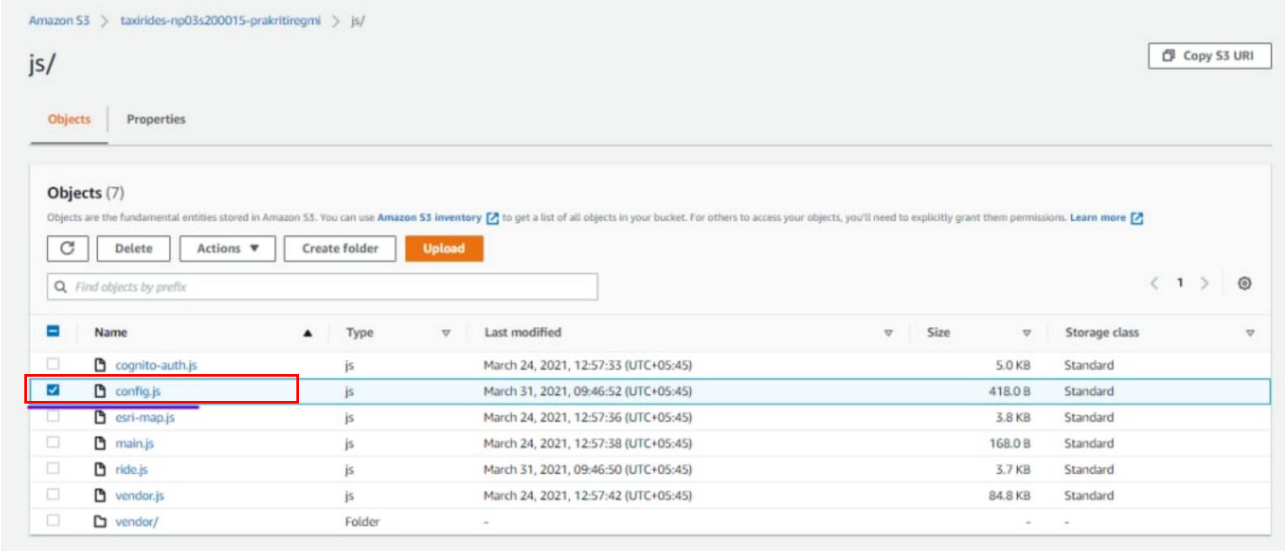
Creating a user pool named TaxiRides and adding a client for the pool with the name TaxiRidesWebApp and the app client id was noted.

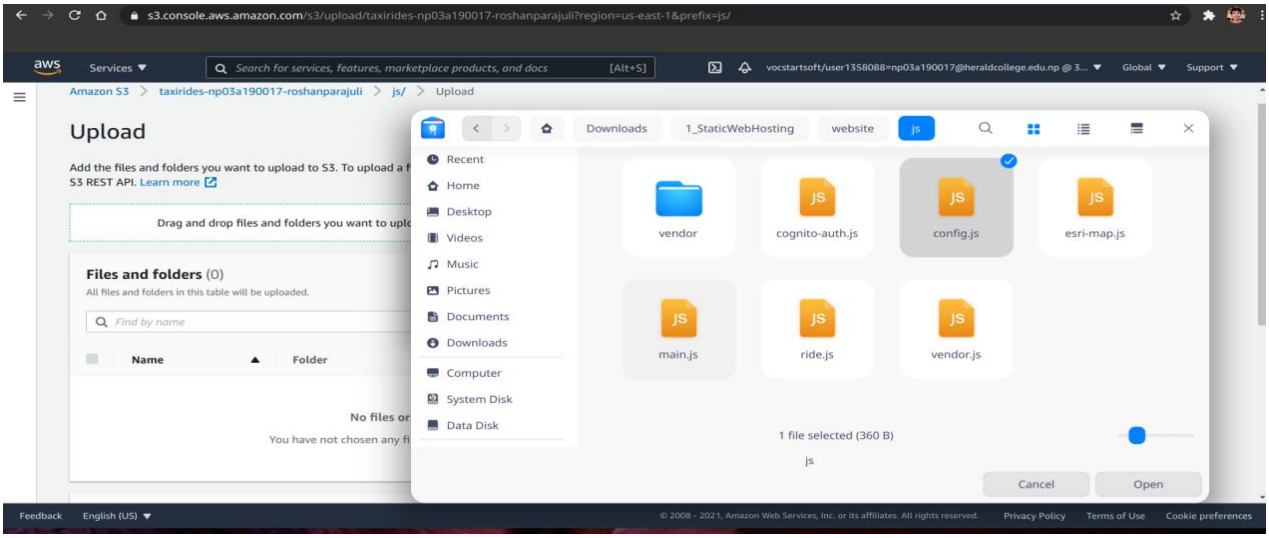


The config.js file on the js folder was downloaded and the following configuration was changed.

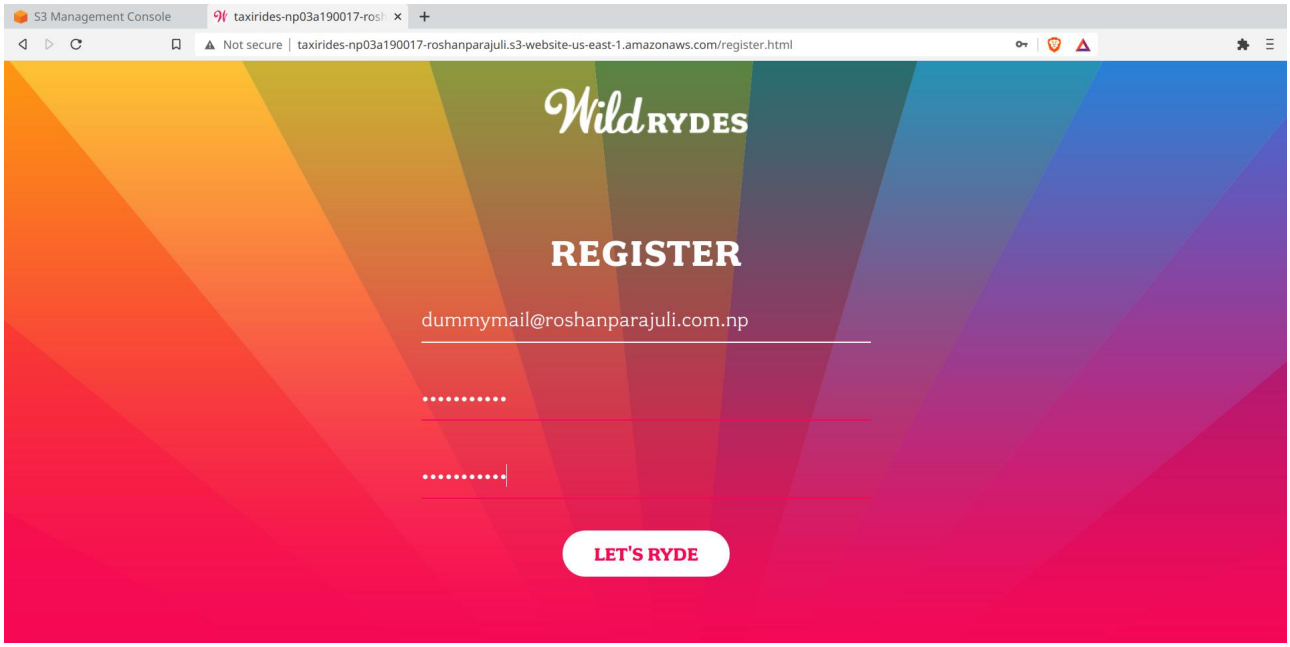
```
window._config = {
    cognito: {
        userPoolId: 'us-east-1_lSvDn1TXb', // e.g. us-east-2_uXboG5pAb
        userPoolClientId: '22r5r4bf4fvo8s0pr9frv571c5', // e.g. 25ddkmj4v6hfsfvruhpfi7n4hv
        region: 'us-east-1' // e.g. us-east-2
    },
    api: {
        invokeUrl: '' // e.g. https://rc7nyt4tql.execute-api.us-west-2.amazonaws.com/prod,
    }
};
```

The config file now has the configuration files for the cognito services. After editing the file, it was deleted in the server and uploaded again in the same directory.

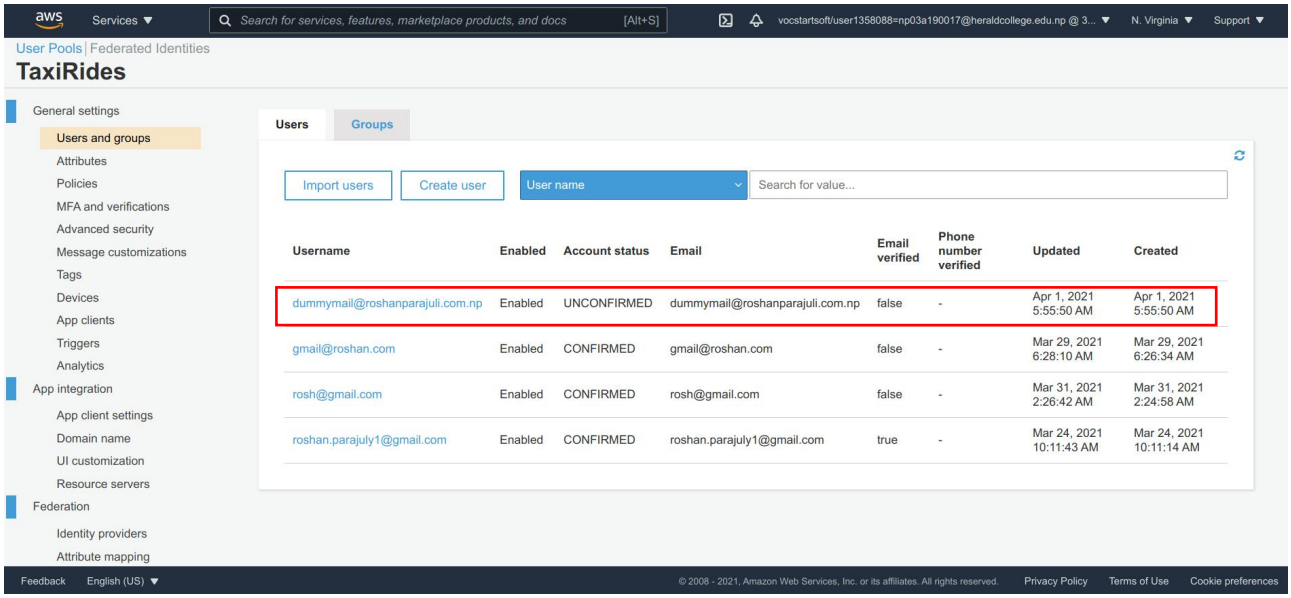




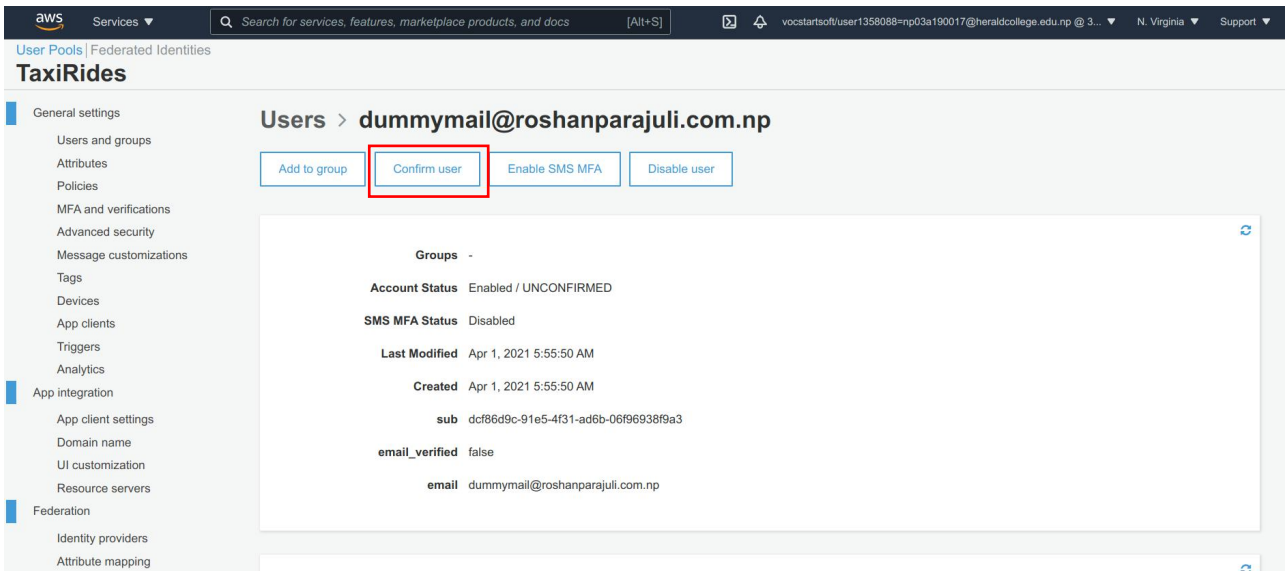
Testing the implementation



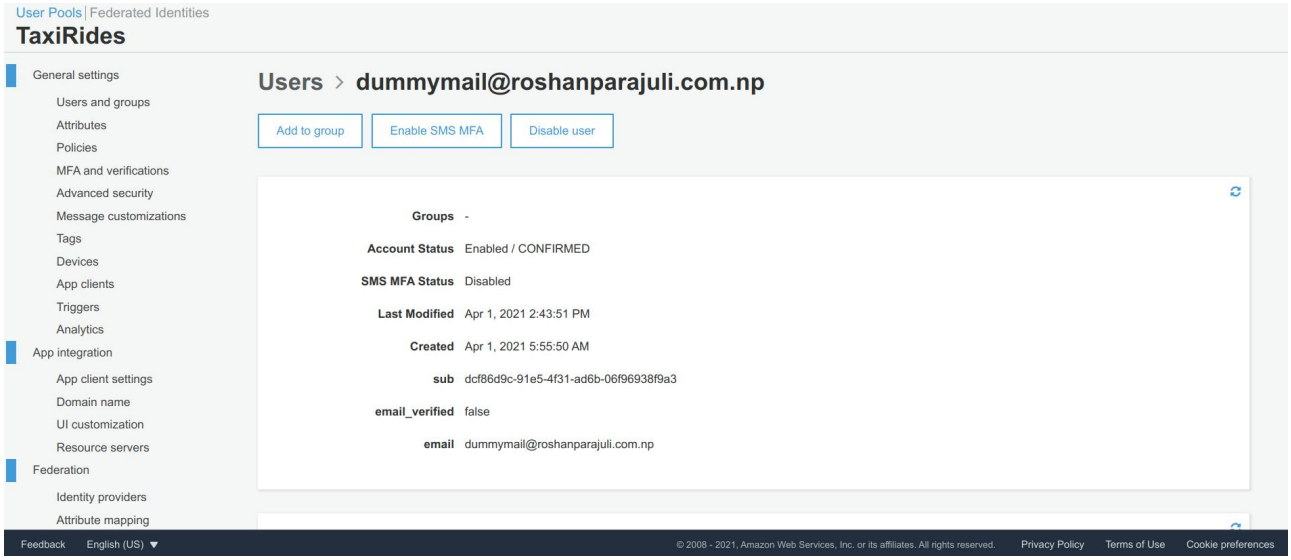
A dummy email is registered in the site for testing its implementation.



The dummy email registered can now be seen in the “Users and Group” tab in the **cognito** service. It can be clearly seen that the account status is **unconfirmed**.

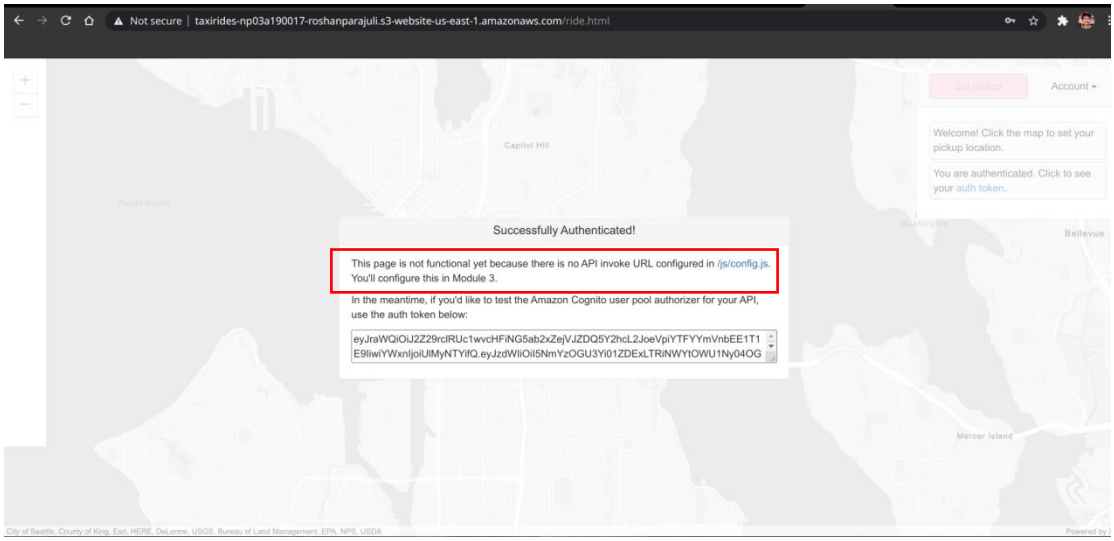


The user is clicked upon and the user is verified manually through the cognito portal.



The user is now verified.

Ride.html page screenshot:



ride.html page successfully authenticates the load but is not functional because of no API invocation yet.