Planter

Link: http://a3-env.eba-i6b8dj8x.us-east-1.elasticbeanstalk.com/

Summary:

Planter is a web app developed to allow people to share, display, and discuss plants with other users of the app, given they have signed in after registering.

Planter was developed and deployed using numerous services provided by Amazon Web Services, including Elastic Beanstalk, Lambda, AWS API, S3, DynamoDB, and AWS Backup. Using these services made deployment extremely simple as opposed to if they weren't used. Boto3 was used in order to access some of these services, namely DynamoDB and S3, using python.

With Planter users can:

- view posts made by registered users
- view a users profile
- view their own profiles
- search for posts with specific keywords
- register a new account
- sign in with an existing account
- logout if they are logged in
- make new posts, optionally with descriptions and an image
- and delete posts they want to remove

Introduction:

I was motivated to make a plant related management and social app because I was gardening recently, and had very few people to discuss this interest with, given the current circumstances.

Planter acts as a forum for people to discuss plants, where users are able to make posts with images. Users can also search for posts which contain specific words, such as “flower”. Additionally, users can view the profiles of users which make posts

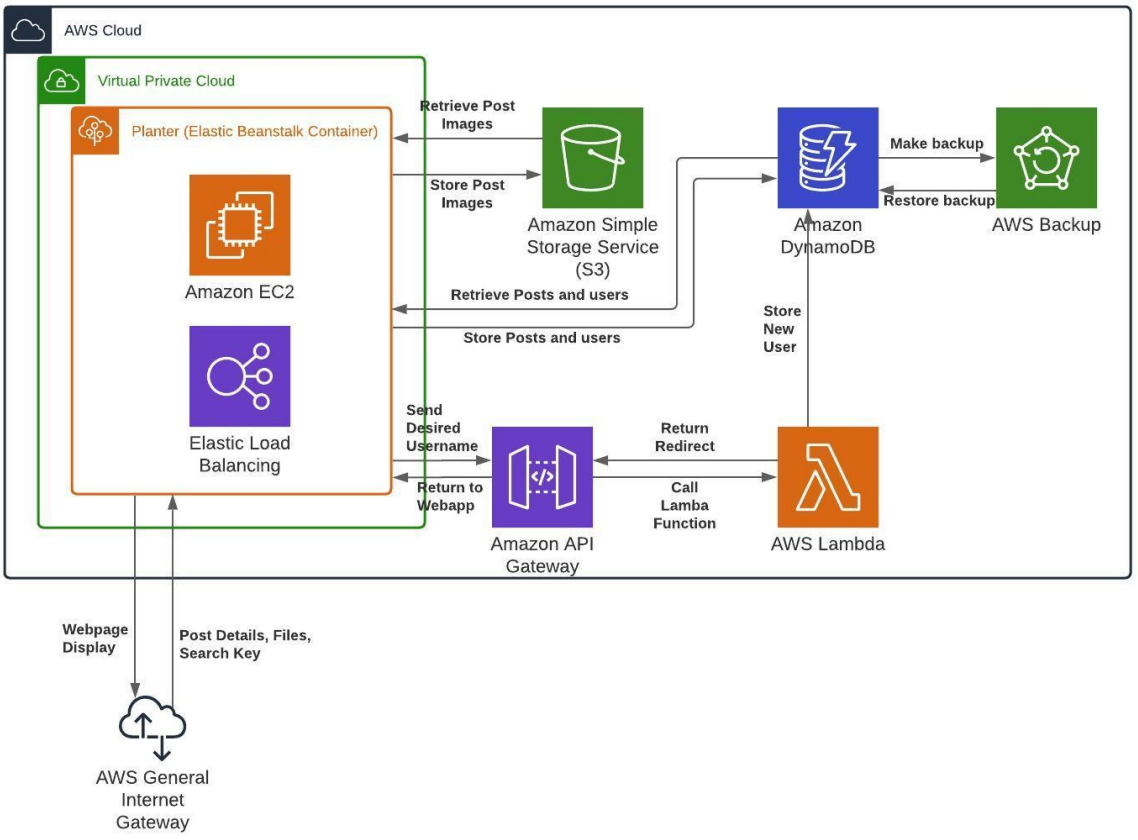
they like to potentially view posts similar to what made them interested in the users posts.

These functions make Planter useful to others who don't know where to go if they have an interest or hobby in looking at or discussing plants. Additionally, the app could be used as a time killer for people who travel to work or university using public transport.

Related Work:

A web app and mobile app that could be considered similar to Planter in terms of social interaction is Reddit[1], which hosts numerous user created and managed forums called subreddits where users can post content,rate other users content, and comment on other users' content. Much like Reddit, Planter users can view other users' posts, as well as post their own posts, with images.

System Architecture:



System Descriptions:

Elastic Beanstalk

Elastic beanstalk is a scalable PaaS where users can deploy applications without having to manually create a server and install each requirement needed to run the app. Because of this Elastic Beanstalk automatically contains an EC2 instance. This made it so that first uploading and updating the application in development was fast and simple, and easily configurable. In addition, Elastic Beanstalk has automatic load balancing, meaning more instances are created or deleted depending on the amount of people accessing the webapp at the same time.

S3

S3 is a storage service for AWS where files can be stored. For Planter, I have used S3 to store the images given by users when they make posts.

DynamoDB

I have used DynamoDB for Planter to store the details for users and posts. DynamoDB is a scalable noSQL database. I have also used it to find and retrieve posts for all users, specific users, and all posts which contain a string in their plant name or plant description.

AWS Backup

I chose to use AWS Backup, a storage service, to make daily backups of the dynamodb databases for users and plants. This allows table restoration in the event of an undesired deletion, or accidental table deletion. This would only be available for tables or posts which are saved in the daily backup

Lambda

Lambda is an AWS compute service used for serverless computing, which I have utilised to generate a random string of numbers which users who register use as passwords, in response to being triggered by AWS API gateway.

AWS API Gateway

I have used AWS API, a networking and content delivery service, to trigger AWS Lambda to generate a password and create new users in the DynamoDB database.

Developer Manual:

DynamoDB
To create your dynamo db tables, use the following code
USERS:

```
{
   "TableName": "Users",
   "KeySchema": [
     { "AttributeName": "Username", "KeyType": "HASH" },
     { "AttributeName": "Password", "KeyType": "RANGE" }
   ],
   "AttributeDefinitions": [
     { "AttributeName": "Username", "AttributeType": "S" },
     { "AttributeName": "Password", "AttributeType": "S" }
   ],
   "ProvisionedThroughput": {
     "ReadCapacityUnits": 5,
     "WriteCapacityUnits": 5
   }
}
```

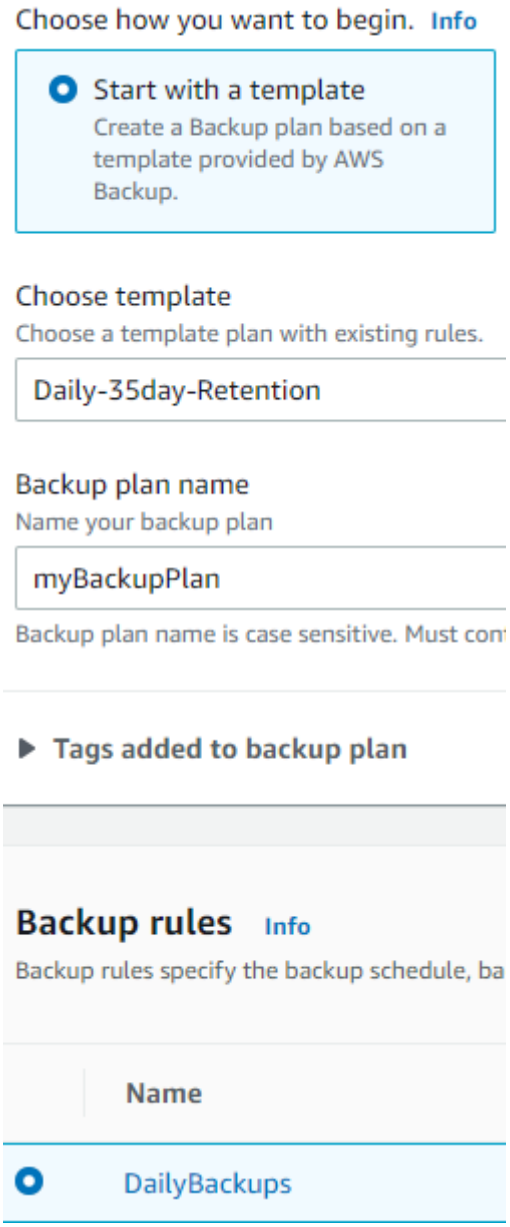
PLANTS

```
{
   "TableName": "Plant",
   "KeySchema": [
     { "AttributeName": "Username", "KeyType": "HASH" },
     { "AttributeName": "Plantname", "KeyType": "RANGE" }
   ],
   "AttributeDefinitions": [
     { "AttributeName": "Username", "AttributeType": "S" },
     { "AttributeName": "Plantname", "AttributeType": "S" }
   ],
   "ProvisionedThroughput": {
     "ReadCapacityUnits": 5,
     "WriteCapacityUnits": 5
   }
}
```

Then follow this tutorial from “Creating a test table” onwards
https://hackernoon.com/using-aws-dynamodb-with-flask-9086c541e001

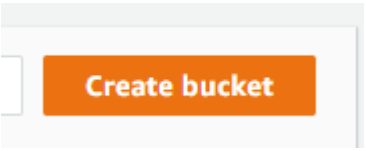
AWS Backup
Using AWS backup, accessed at
https://console.aws.amazon.com/backup/home?region=us-east-1#/backupplan
Select

Use the following settings, change the plan name if you wish

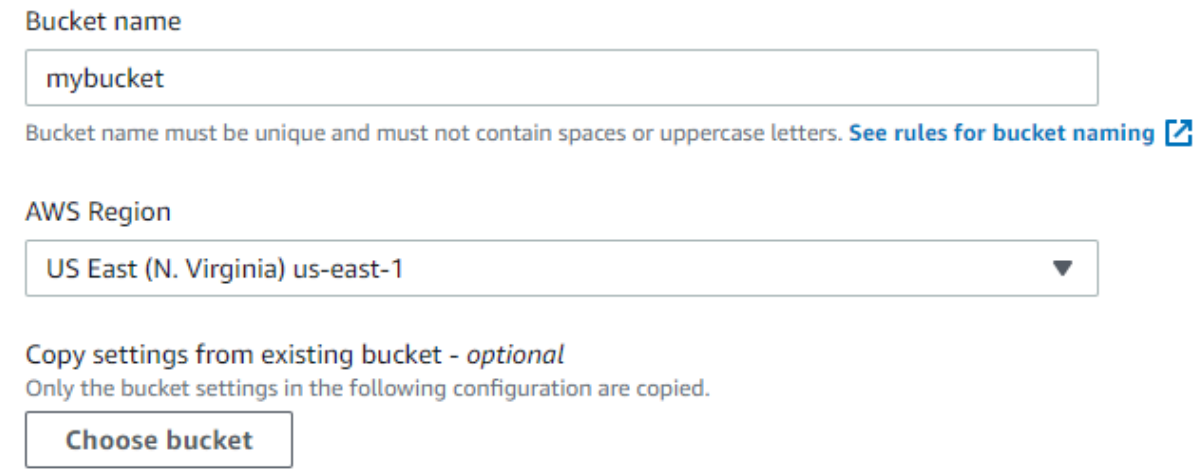


S3

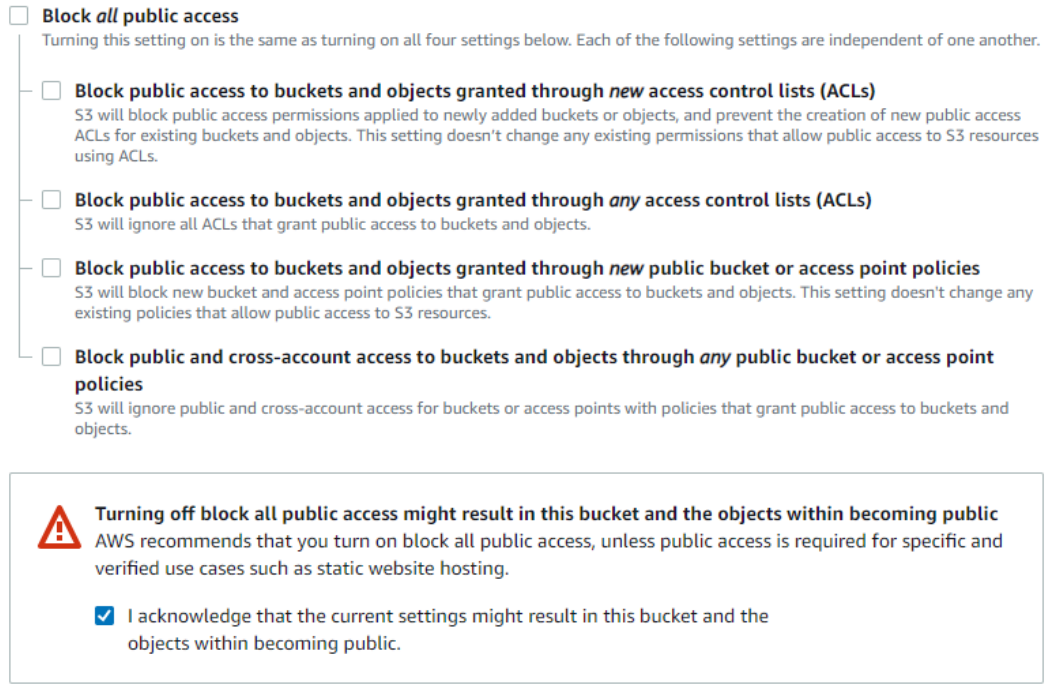
From https://s3.console.aws.amazon.com/s3/home?region=us-east-1 we can create a bucket to store all the images for out plants



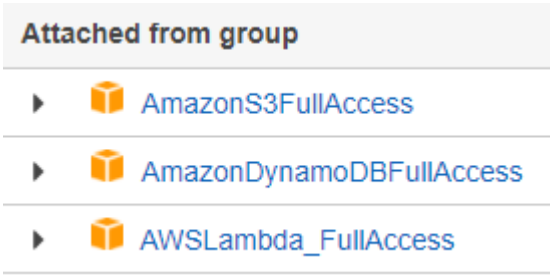
Name the new bucket and make sure it's in the same region as your application.



Set access to public so images can load properly



Add the following roles to the new IAM user
https://console.aws.amazon.com/iamv2/home?#/groups/details/DB?section=permissions



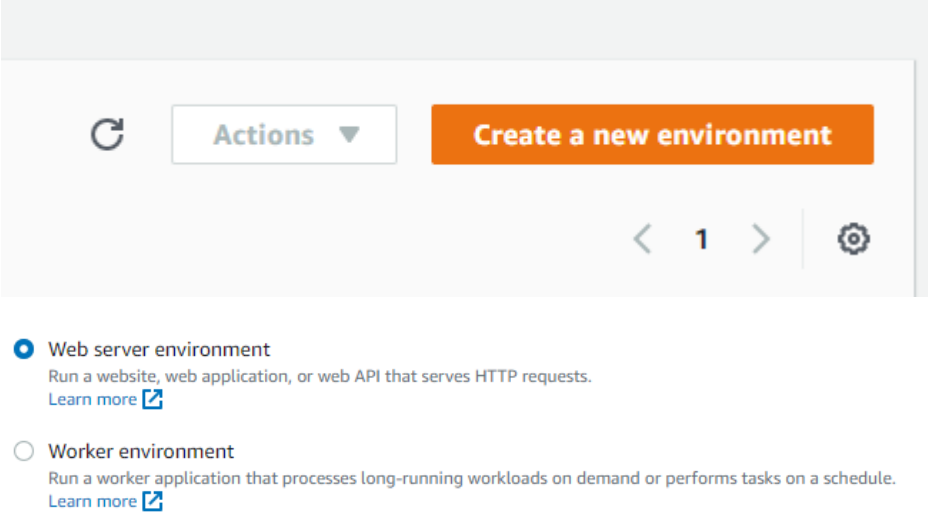
AWS API Gateway and Lambda

Follow the tutorial here https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-dynamo-db.html, except skip step 1, make a python lambda and use the following code for step 2, and for step 3 change the route to be “/{username}”:

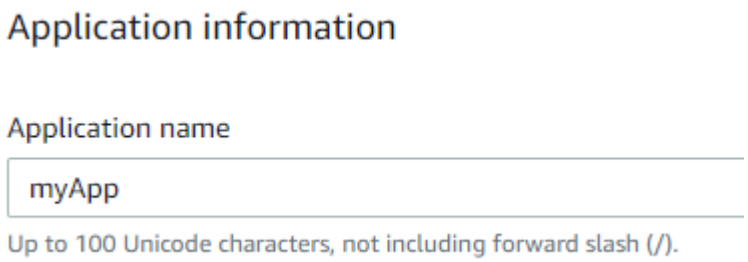
```
import json
import boto3
import datetime
import random
from boto3.dynamodb.conditions import Key

dbc = boto3.client("dynamodb")
dbr = boto3.resource("dynamodb")

def lambda_handler(event, context):
    username = event['pathParameters']['Username']

    passw = ""

    for i in range(0,6):
        passw += str(random.randint(0, 9))

    users = dbr.Table('Users')
    users.put_item(
        Item={
            "Username" : username,
            "Password" : passw,
            "Date" : datetime.datetime.today().strftime("%d/%m/%Y")
        }
    )

    response = {}
    response["statusCode"]=302
    response["headers"]={'Location':
"http://a3-env.eba-i6b8dj8x.us-east-1.elasticbeanstalk.com/registered/"+passw+"/"+u
sername}
    data = {}
    response["body"]=json.dumps(data)
    return response
```

Make sure to replace “http://a3-env.eba-i6b8dj8x.us-east-1.elasticbeanstalk.com” with the url of your app once you have deployed it using elastic beanstalk.
The response configuration was adapted from [6]

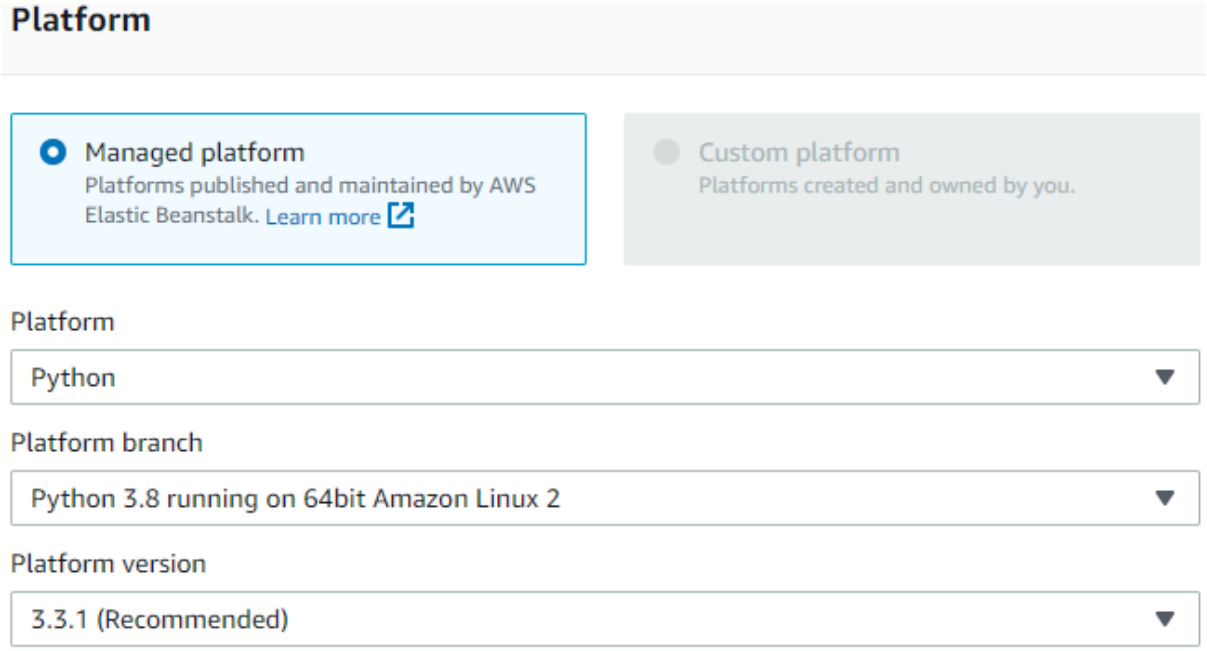
Elastic Beanstalk
Finally we can deploy the app using elastic beanstalk at the following link https://console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/environments



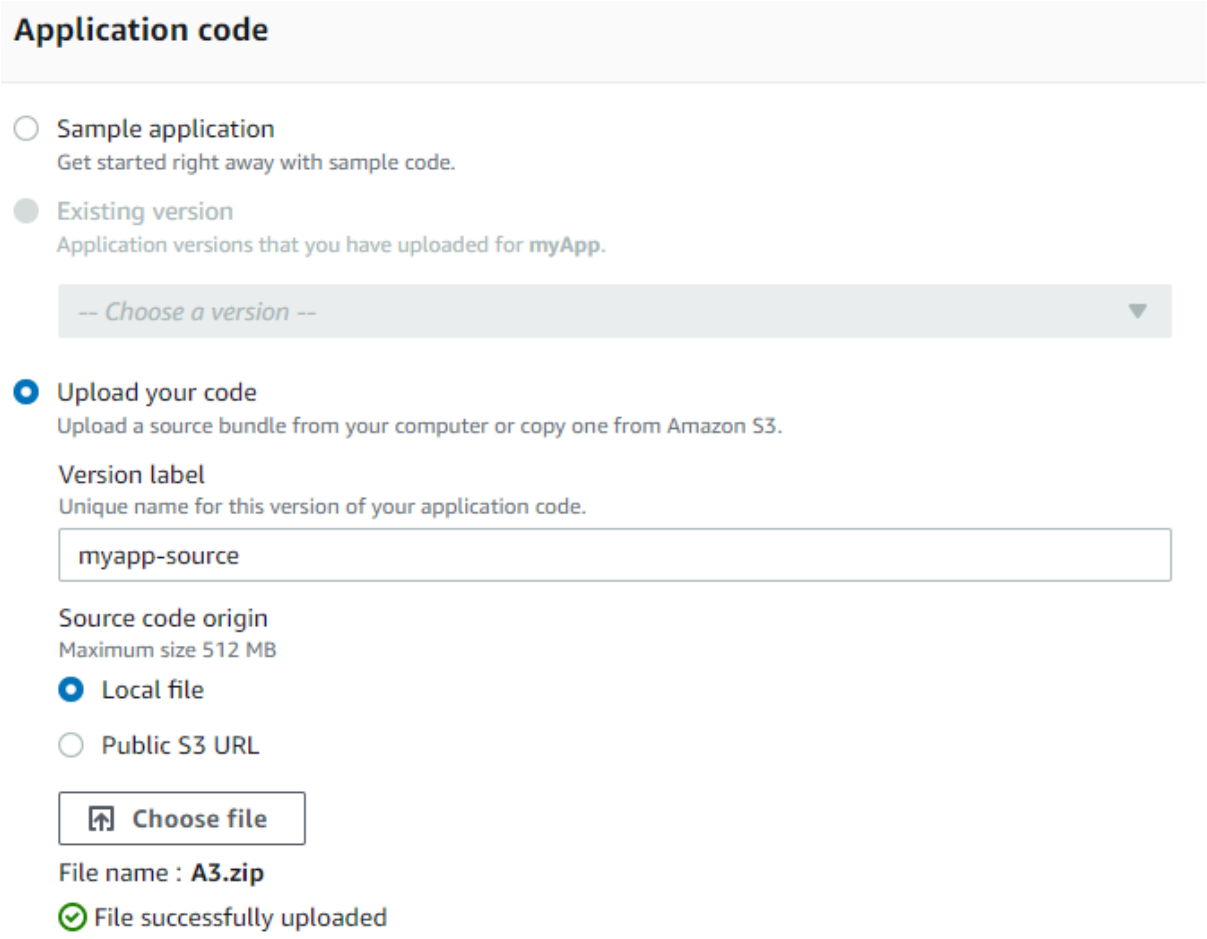
Name The app whatever you want, all the other settings can be left as default



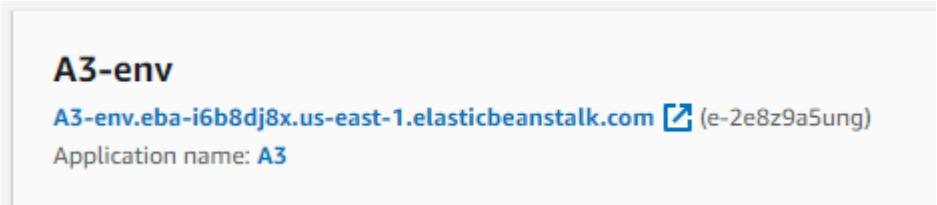
Select Python as your platform



Choose to upload the zipped application, and create the environment



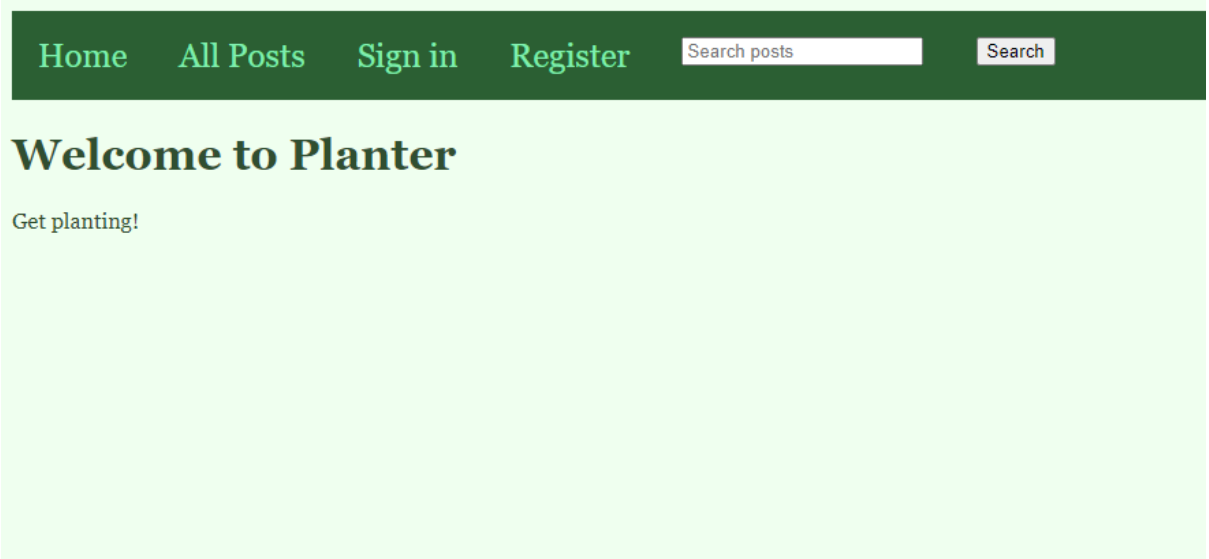
Finally visit the app at the link shown



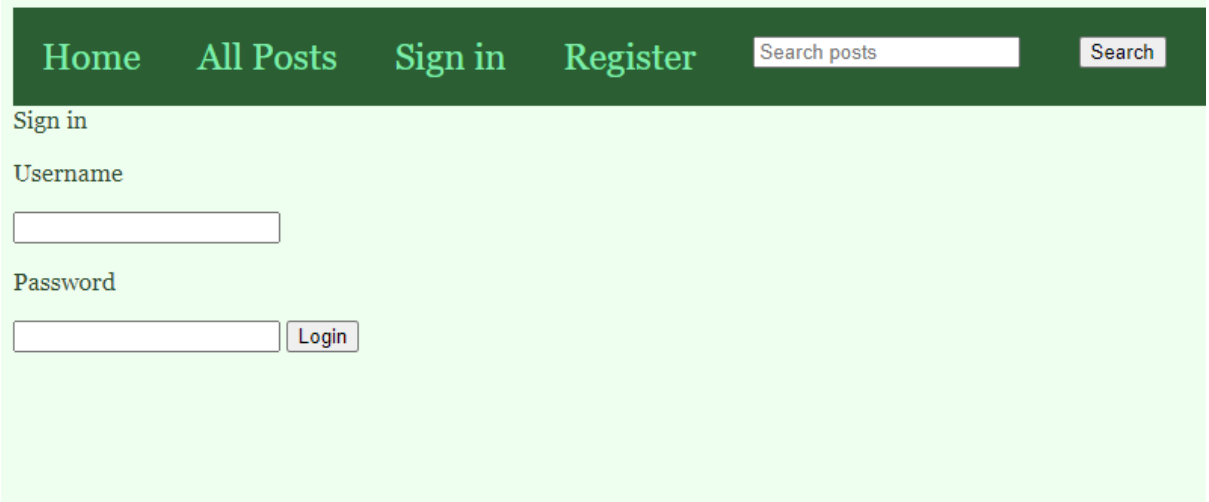
User Manual:

Users can open the provided link to be sent to the Planter homepage, where they will see a welcome message from the web app, from there they can login, register, look at users profiles, view all posts, and search for specific posts, or if they're logged in, they can look at their plants, post new plants, or logout.

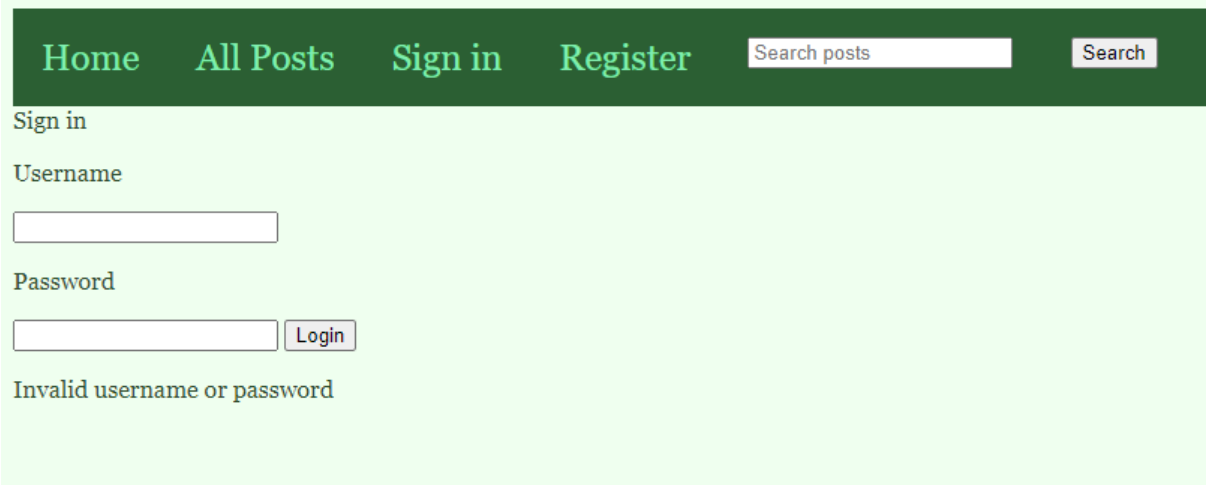
This is the homepage users will be greeted with if they aren't signed in. If the user has signed in previously without logging out, they will still be logged in upon opening the webpage.



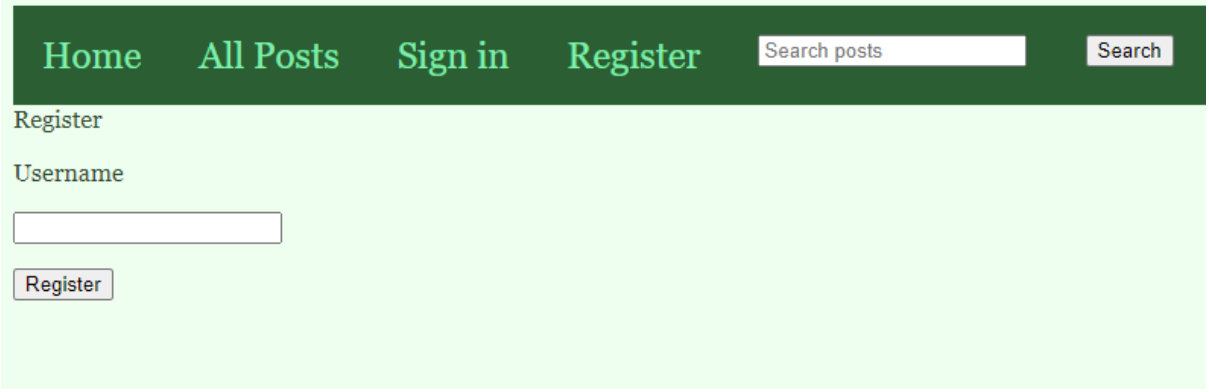
This is the page where users can sign in to the app. Both inputs are required, and will prevent the user from pressing the "Login" button if empty.



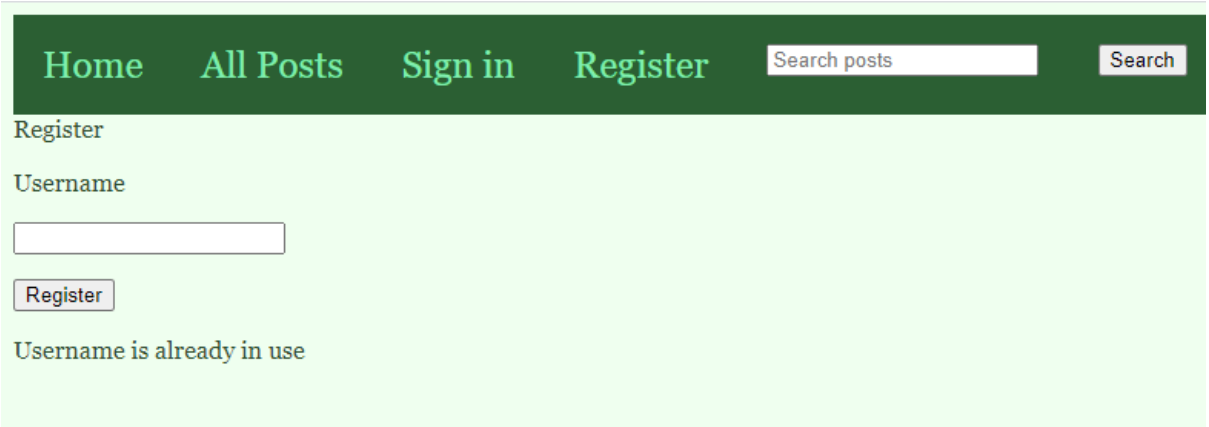
If the username is not found, or the password does not match with the username in the DynamoDB “Users” database Planter will indicate that the login attempt was invalid



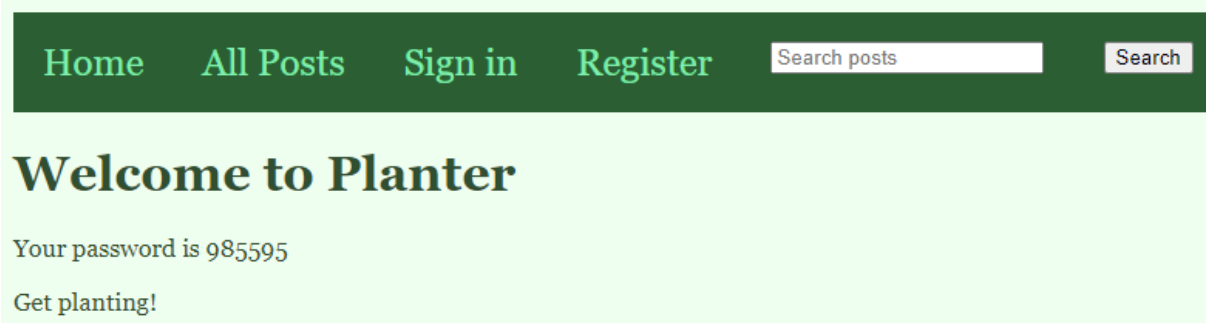
This is the page where users can sign in



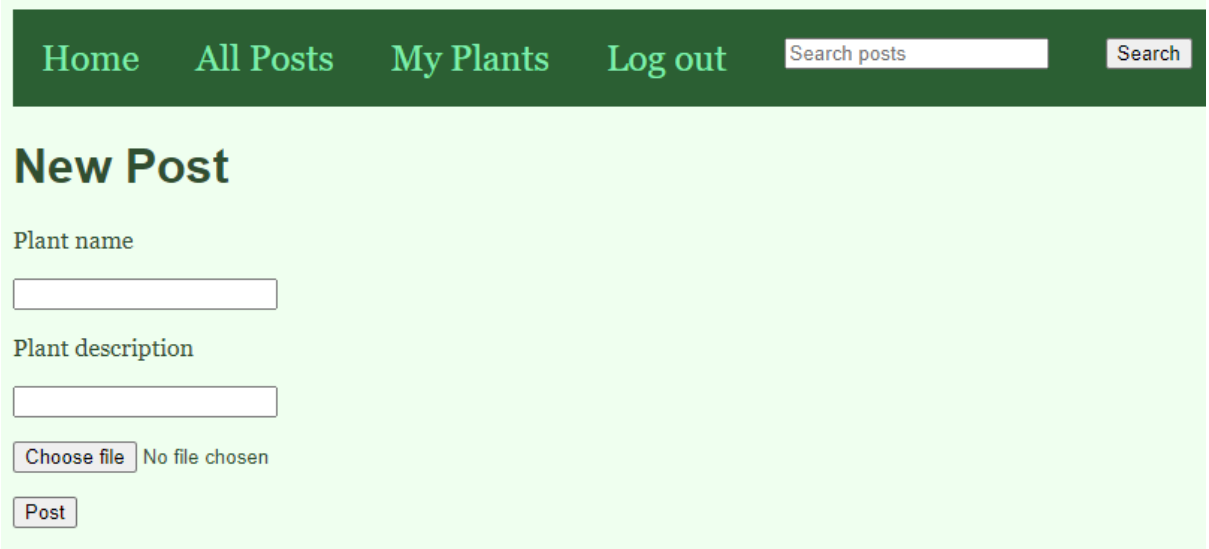
Planter will tell the user if the username is already in use



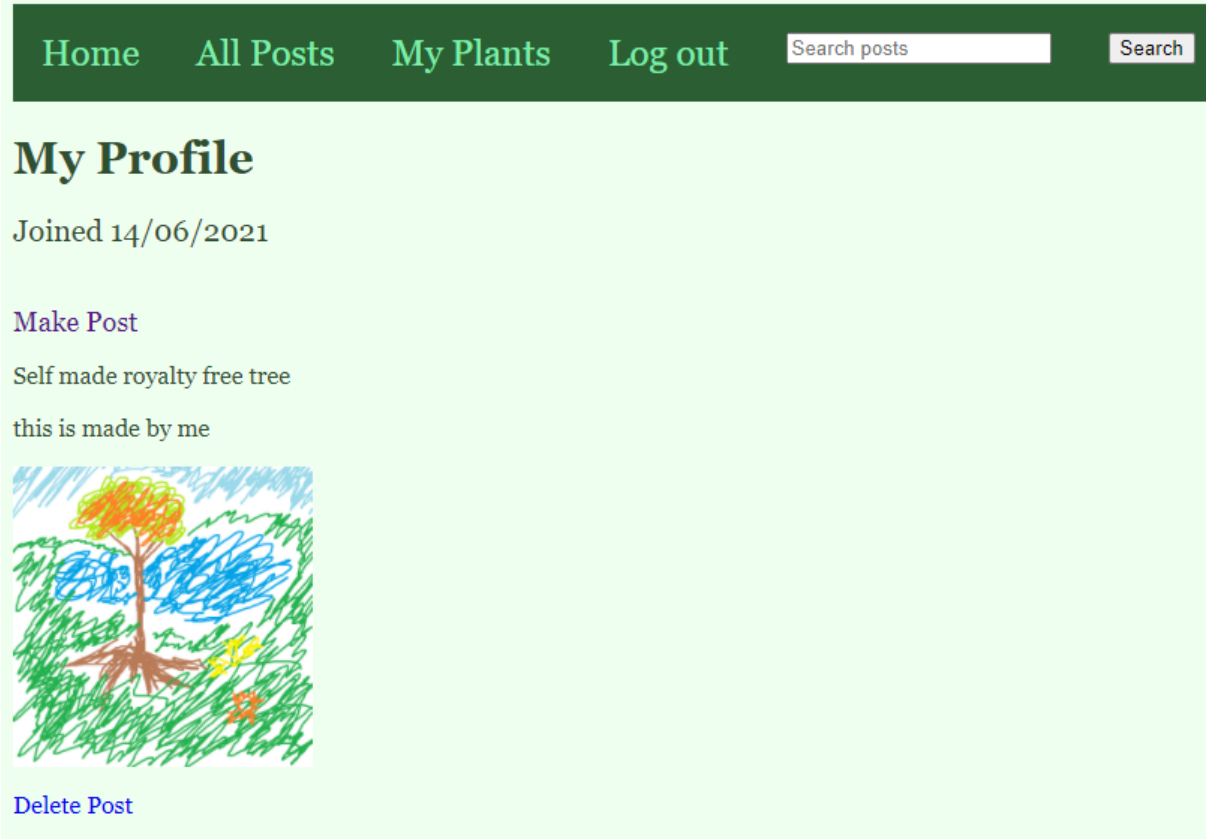
Once the new user is made the user will be redirected to the Planter homepage where they will be told their password



If the user selects “Make Post” they will be redirected to this page where they can submit a plant name, a plant description, and an image. The user is only required to input a plat name to make the post



Once the user confirms the post, the new post will appear in “All Posts” and “My profile”, the latter being where the user will be redirected to. Should the user wish to they can remove the post by selecting “Delete Post” from the DynamoDB database



All posts shows the user all the posts made by themselves, and other users

Home All Posts My Plants Log out Search posts Search

All Posts

Posted by Ogga Booga

EERT

ITS TREESON THEN

Posted by plantLover

Self made royalty free tree

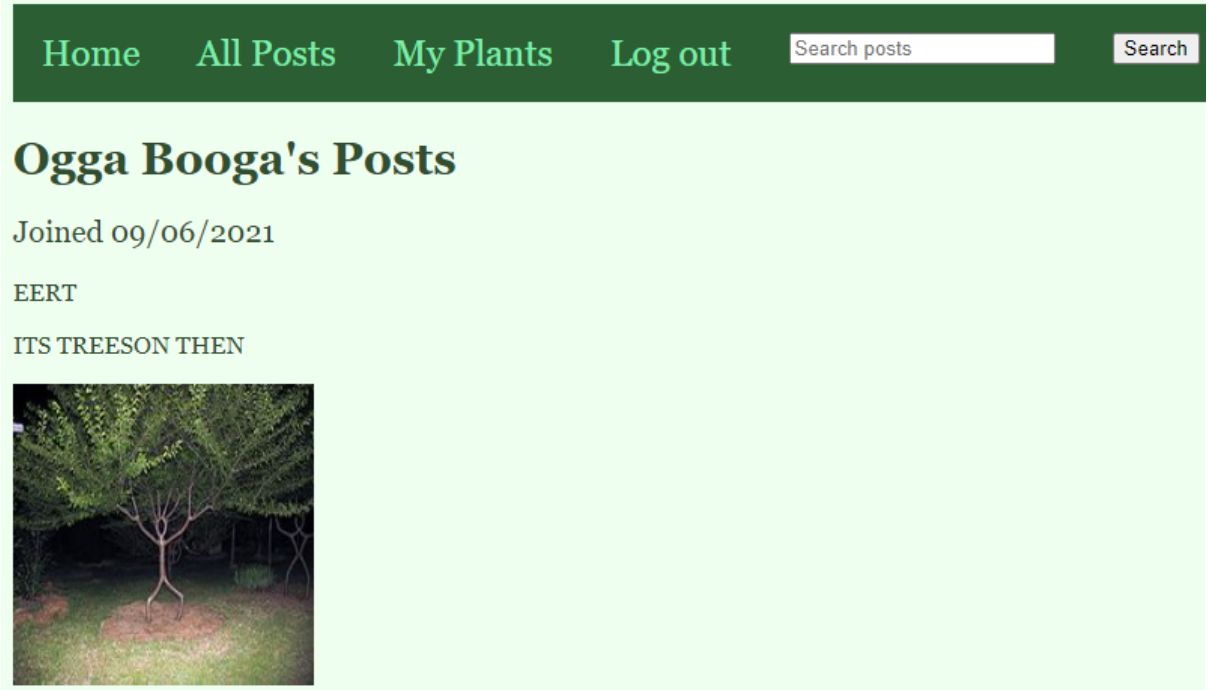
this is made by me

Posted by a

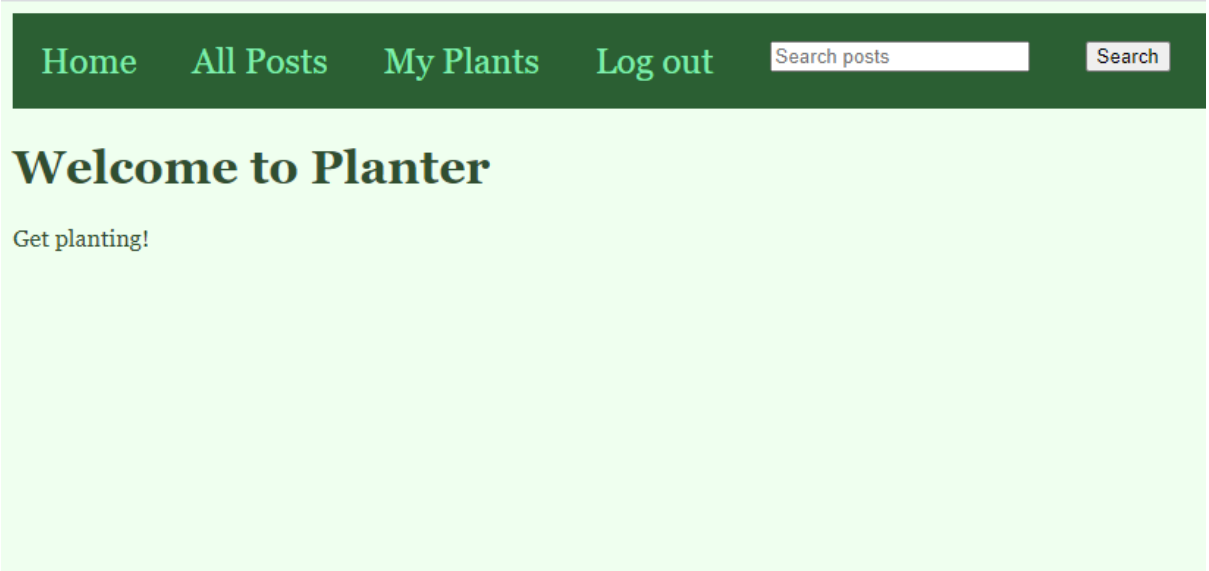
Wowzers

no image

Clicking on a posts poster will redirect the user to that poster’s profile, where all their posts are displayed, along with the users join date.



The user is redirected to the home page once they are logged in. Clicking “Log out” will redirect the user to the signed out homepage shown at the start of this manual



References:

[1]"reddit: the front page of the internet", Reddit.com, 2021. [Online]. Available: https://www.reddit.com/. [Accessed: 09- Jun- 2021].
[2]"Step 3: Create, Read, Update, and Delete an Item with Python - Amazon DynamoDB", Docs.aws.amazon.com, 2021. [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Python.03.html. [Accessed: 14- Jun- 2021].
[3]"Step 4: Query and Scan the Data - Amazon DynamoDB", Docs.aws.amazon.com, 2021. [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Python.04.html. [Accessed: 14- Jun- 2021].
[4]F. flask, "File upload error in flask", Stack Overflow, 2021. [Online]. Available: https://stackoverflow.com/questions/47679398/file-upload-error-in-flask. [Accessed: 14- Jun- 2021].
[5]"Uploading files — Boto3 Docs 1.14.31 documentation", Boto3.amazonaws.com, 2021. [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/1.14.31/guide/s3-uploading-files
[6]R. Proxy, "Redirect from a Python AWS Lambda with AWS Gateway API Proxy", Stack Overflow, 2021. [Online]. Available: https://stackoverflow.com/questions/55022035/redirect-from-a-python-aws-lambda-with-aws-gateway-api-proxy. [Accessed: 14- Jun- 2021].