The Volunteer-CRM

https://www.platinumvolunteers.com/

Produced by Azriel Bachrach, Arieh Chaikin, Isaac Gutt, Eli Levy, and Lawrence Snow

Mentor: Alex Porcelain

Github:
https://github.com/azrielb1/Platinum-Volunteers

Table of Contents

About Us

We are a team of undergraduate students at Yeshiva University studying Computer Science. In the summer of 2021, we signed up to spend our summer working on a Customer Relationship Management Website. After a lot of research, we decided that a CRM to help volunteers find and sign up for events was the most useful app we could build. Our app includes advanced features in addition to basic functionality. We came into the project with no experience with app development or Amazon Web Services.

Our mentor, Alex Porcelain, is a lead security architect for Goldman Sachs. He has helped us face new challenges that we have never faced before and think in new ways. Because of the skills we have gained under Alex's guidance, we now feel confident that we are capable of building an app and using AWS on our own, even with technologies Alex has not specifically taught us.

What is a CRM

A customer relationship management, or CRM, is a system which connects an organization or business with their customers. Data storage is central to a CRM software, enabling both the customers and the organization to understand each other and connect more effectively.

Target audience

We have created a web app for those looking for volunteers for events and also those looking for volunteer opportunities.

Our app allows event **organizers** to connect with potential **volunteers** and manage their events more effectively.

Features of our Application

1. Allows users to organize events by posting their upcoming events and volunteer opportunities. Enables organizers to view who signs up and allows them to check in those volunteers on the day of the event.

2. Users can browse upcoming events and sign up for future positions. Users can filter and sort upcoming events. Users can view their upcoming events and receive email reminders before an event.

3. Collects data about the user's past events in order to suggest upcoming events that they might be interested in.

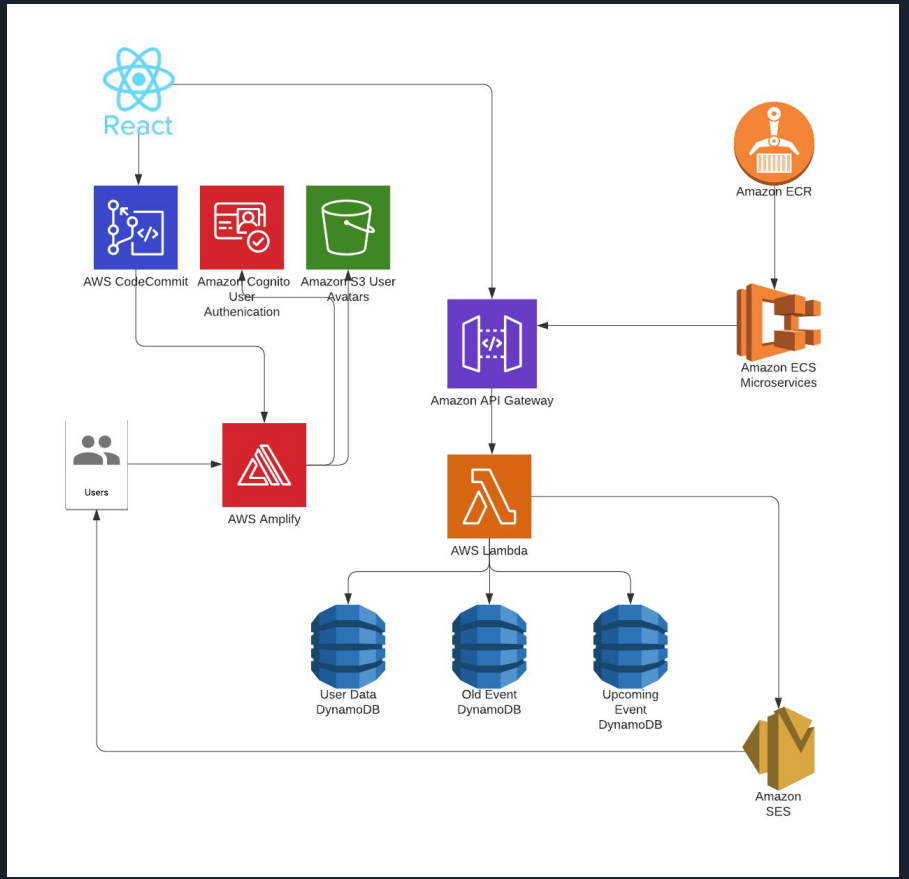
Key Challenges

- Providing an intuitive user interface
- Secure signup and login
- Storing upcoming and past events
- Collecting and storing user data
- Sending notifications to users
- 3rd party integration (Weather, Google Maps, PayPal)

How we solved those challenges

- We used Javascript and React to develop a modern and intuitive user interface
- We used many AWS services to solve key challenges
  - We used Amazon Cognito to allow secure user sign up and login
  - We used DynamoDB to store upcoming and past events, who is signed up for which event, as well as additional user data
  - We used Amazon SES to send users notifications.
  - We also made use of multiple other AWS services that we will discuss later

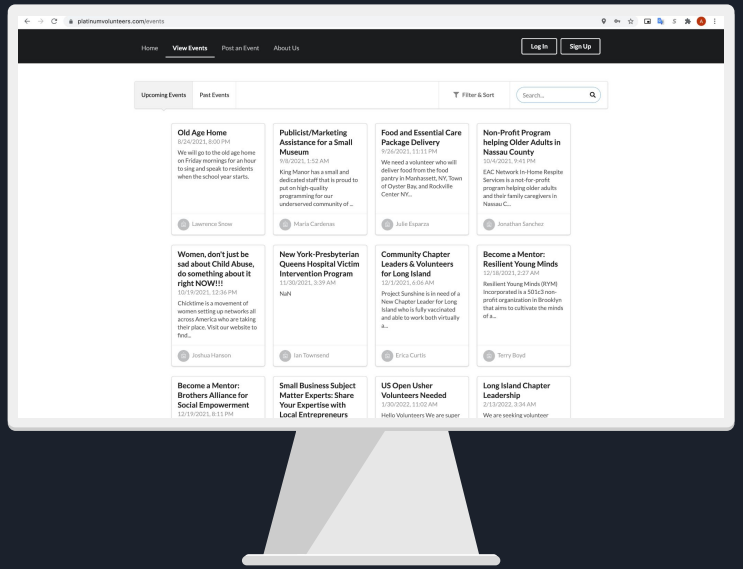
- Front end written in React
- Hosted by AWS Amplify
- Cognito and S3 integration
- API Gateway triggers a Lambda function which completes operations on the database
- ECS cluster running microservices



Overview of Application Architecture

Application front end

- Our web app is written in Javascript using the React library (https://reactjs.org)
- We used Semantic UI (https://semantic-ui.com) components to build the user interface
- A React router (https://reactrouter.com) was used to display different pages
- React states were used to display dynamic content



Application Flow – User signup & log in

- New users are asked to create an account with their email address and a password.
- They must also provide some additional information such as their name and date of birth.
- We collect the information a user enters in a react state
- We call Amplify authentication's Auth.signUp() function and pass in the user data
- Amplify will email the user a confirmation code that they must enter to ensure user authentication.
- Amplify then adds the new user into the Amazon Cognito user pool
- We also create a new entry in a DynamoDB table of users in order to store additional user data
- Now that users have an account, they can log in with their email address and password.
- We use Amplify authentication to handle login

Application Flow – Create event

- Users have the ability to give their event a title, description, additional information, location, date, and tags
- Tags are used to categorize events for more efficient searching and suggesting of events
- When a user goes to post an event, they must enter the event information which we temporarily store in a react state
- Once the user submits the event, we create an entry in the database for the event by calling the API gateway which passes the information to the Lambda function, which stores it in DynamoDB
- When the event is first added to the database, it is flagged as “not approved.” From the admin panel, site administrators can approve or reject events.

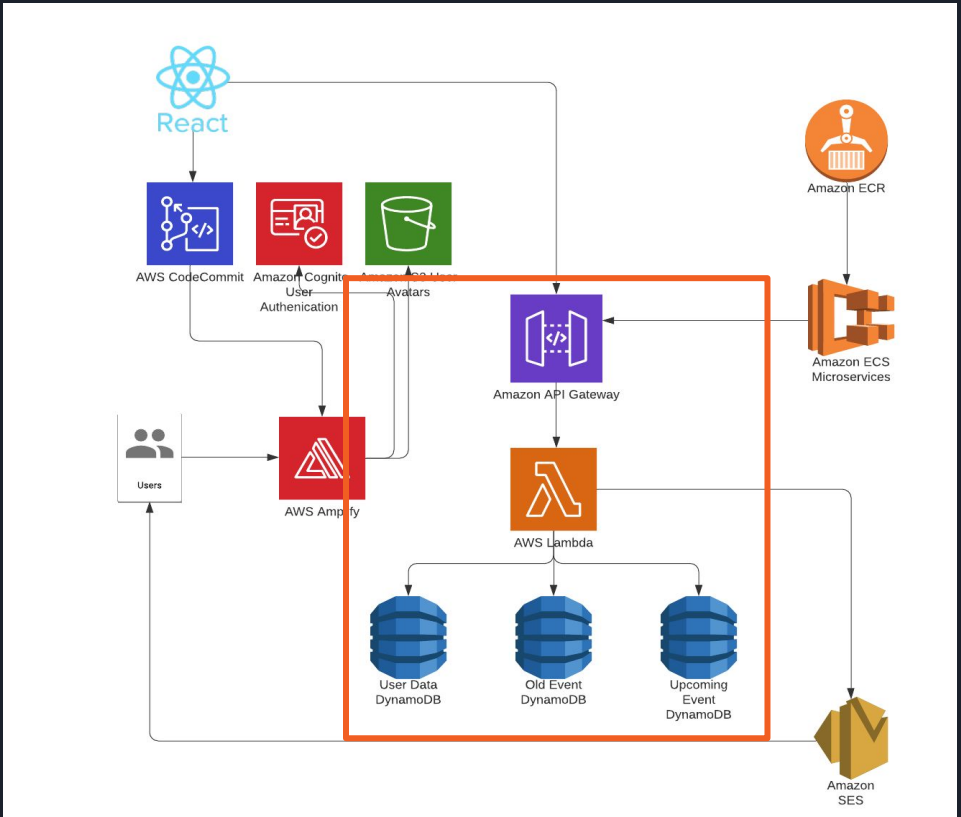
Application Flow – Searching events

- When a user opens the events page, a call is made to the API gateway which calls the Lambda function that retrieves and returns the approved events from the DynamoDB table
- The events are saved in a react state and displayed to the user
- The user has the option to sort the events by date, distance, or alphabetically
- Users can search events by keyword in addition to being able to filter events by their tags or distance
- Users also have the ability to view past events
- When a user clicks on an event, they can view information about the event including the description, address (with a google map), weather, event organizer, as well as the names of other volunteers.

Application Flow – Registering for events

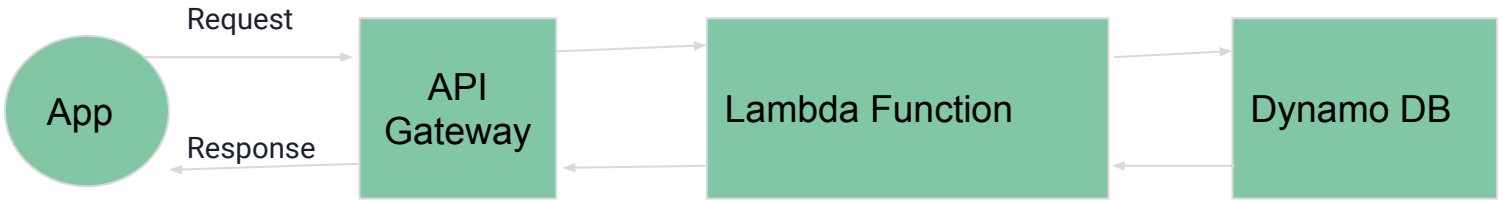
- When a user registers for an event, a call is made to the API Gateway which handles the joining of event
  - The user's ID is added to the event's list of participants in the event database
  - The event's ID is added to the user's list of upcoming events in the user database
- An additional API call is made to send an email to the user thanking them for joining the event and giving them the event information
- Users can also unregister which calls the API Gateway to remove the user from the event participants and remove the event from the users upcoming events
- Users are assigned a unique ID for the event to generate a QR code that the event organizer is able to scan on the day of the event to check in volunteers

Database Management

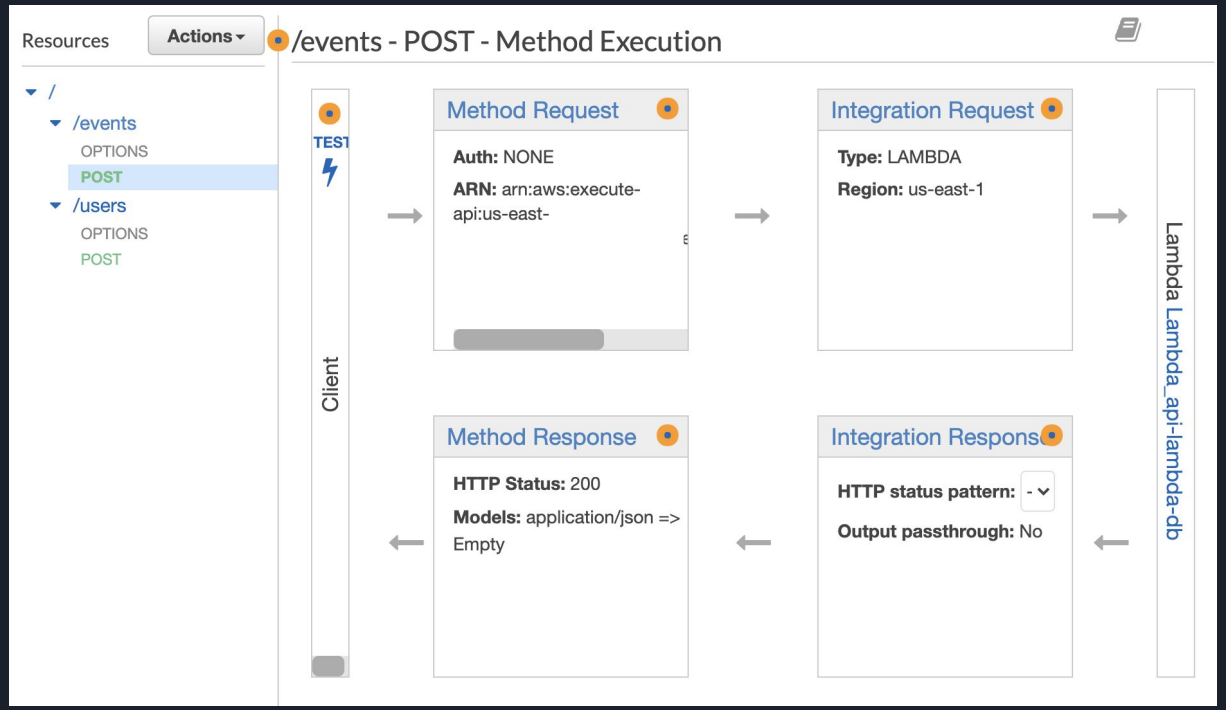


Data Flow

- Data is written, read, updated, and deleted through a Lambda function that manages the database.
- This Lambda function is accessed by the front and backend code through an API Gateway



AWS API Gateway



- REST API with /events and /users
- Request body passed to AWS Lambda via POST
- CORS Enabled

AWS Lambda Function

- Written in Node v12
- Triggered by the API Gateway
- Parses the body of the POST request and completes the desired operations
- Originally it only supported CRUD operations. However, as the application evolved, more complex operations were added. For example: ➞

```
case 'joinEvent':
    dynamo.update(event.userpayload, (userErr, userData) => {
        if (userErr) {
            console.log({
                "userError": userErr
            });
        }
        else {
            dynamo.update(event.eventpayload, (eventErr, eventData) => {
                if (eventErr) {
                    console.log({
                        "eventError": eventErr
                    });
                }
                else {
                    console.log({
                        "success": 'event update succeeded!',
                        "data": eventData
                    })
                    callback(null, {
                        event: eventData,
                        user: userData
                    })
                }
            })
        }
    });
    break;
```

Dynamo DB

- Three tables were used to manage app data- Users, Events, and Old Events
- The access to these tables is restricted to the Lambda function which is responsible for the data management

Scan: [Table] Events_api-lambda-db: id — Viewing 1 to 33 items

creator	date	title
Ariana Ellison	2021-12-09 00:31:55	Volunteer to get America Vaccinated!
Terry Boyd	2021-12-18 02:27:32	Become a Mentor: Resilient Young Minds
Susan Morris	2022-07-07 03:45:51	Volunteers needed for the NY Mets Games
Teresa Jackson	2022-05-23 19:20:33	Nassau County Food Rescue Opportunity
Julie Esparza	2021-09-26 23:11:00	Food and Essential Care Package Delivery
Lindsey Serrano	2022-04-13 07:39:32	Interns Needed for Community Development
Mitchell Benitez	2022-05-06 15:32:07	Volunteers needed for the US OPEN Tennis in NY
Stacy Lewis	2021-10-17 06:27:16	Volunteers needed for the Governors Ball in NY
Joshua Hughes	2022-04-16 06:02:14	Help a Neighbor in Flushing with a Homemade Meal
Ashley Marshall	2021-12-23 22:49:14	Save a Life: Join the Gift of Live Marrow Registry
Maria Cardenas	2021-09-08 01:52:06	Publicist/Marketing Assistance for a Small Museum
Sheryl Rogers	2021-09-28 17:12:02	Great Oaks AmeriCorps Fellowship (August 2021 - June 2022)
Brittany Boyd	2022-01-01 19:59:49	Volunteers Needed for the 2021 New York Mets Home Games
Diane Johnson	2021-08-16 13:45:04	Fundraisers needed to help write grant applications
Erica Curtis	2021-12-01 06:06:11	Community Chapter Leaders & Volunteers for Long Island
Jonathan Sanchez	2021-10-04 21:41:55	Non-Profit Program helping Older Adults in Nassau County

```
"Item": {
        "id": {"S": "0"},
        "name": {"S": "Jimmy Smith"},
        "email": {"S": "downsjoshua@hotmail.com"},
        "phone": {"S": "7217881508"},
        "upcomingEvents": {
          "L": [{"S": "12400"},
                {"S": "3695"},
                {"S": "15500"}]},
        "pastEvents": {
          "L": [{"S": "5790"},
            {"S": "5885"},
            {"S": "6810"},
            {"S": "7565"},
            {"S": "9730"}]
        },
        "bio": {"S": "Put able I police long class. Southern reflect attack music
likely visit she time. Single per think unit military. But someone civil explain black
seat. Son sit down offer."}
      }
    }
```

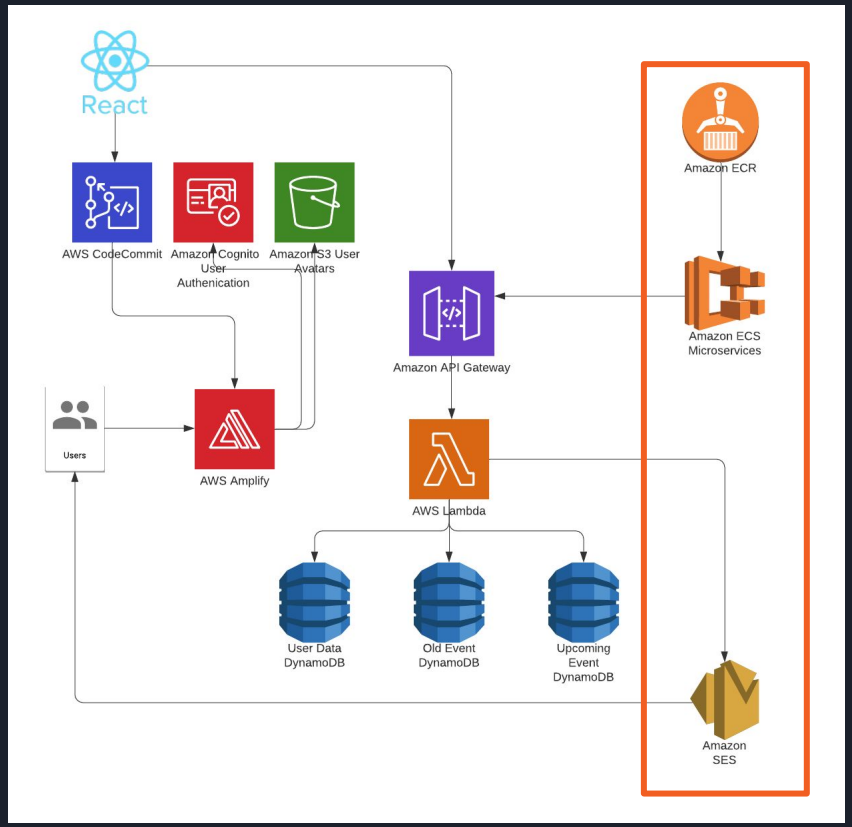

User Data

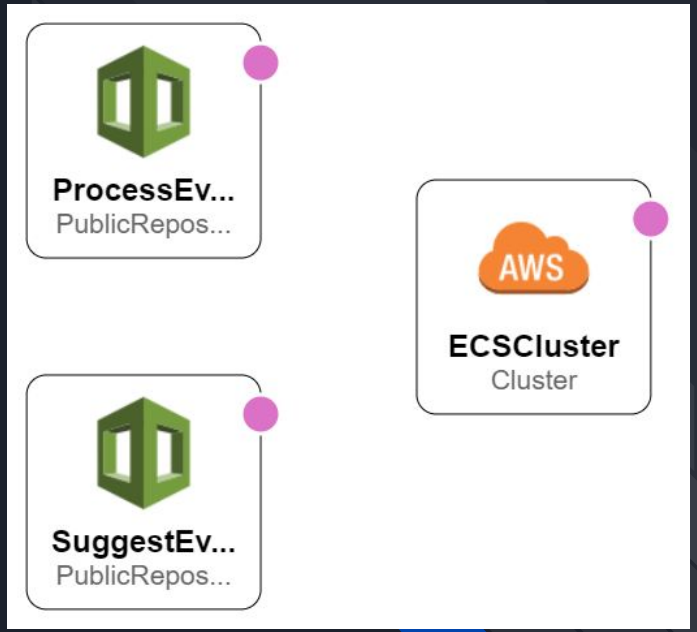
- ID- a unique string generated by AWS Cognito
- Name
- Email- To send reminders about their upcoming events
- Past Events- Past events that the user volunteered for
- Upcoming events- Future events that the user volunteered for
- Bio- The user can update the bio on their profile
- Phone- The user could set their phone number on their profile page

Event Data


```
"Item":{
      "title": {"S": "Become a Mentor: Resilient Young Minds"},
      "id": {"N": "17555"},
      "creator": {"S": "Terry Boyd"},
      "creatorEmail": {"S": "broberts@yahoo.com"},
      "location": {"S": "595 Brown Dr.\r\nSouth Ozone Park, NY 11420"},
      "date": {"S": "2021-12-18 02:27:32"},
      "description": {"S": "Resilient Young Minds (RYM) Incorporated is a 501c3
      non-profit organization in Brooklyn that aims to cultivate the minds of at risk
      youth into resilient ones. Throughout the mentoring year, which starts in
      September and ends in June, RYM works towards a goal of enhancing the ability
      of adolescents to achieve self-efficacy despite difficult or traumatic
      experiences. It is in RYM's mission to invest"},
      "approved": {"BOOL": true},
      "public": {"BOOL": true},
      "tags": {"L": []},
      "participants": {"L": [
                              {"S": "222fdf74-2767-4b88-8623-3ad8a1d4b921"},
                              {"S": "4"}
                              ]
                        }
        }
```

Microservices





Microservices

Code for microservices was containerized with Docker, uploaded to AWS Elastic Container Registry (ECR), and run on an AWS Elastic Container Service (ECS) cluster every 6 hours

Information:

- Code was built in Node js v12
- Containerized using Docker
- Pushed to AWS Elastic Container Registry

Microservice #1- Process Upcoming Events

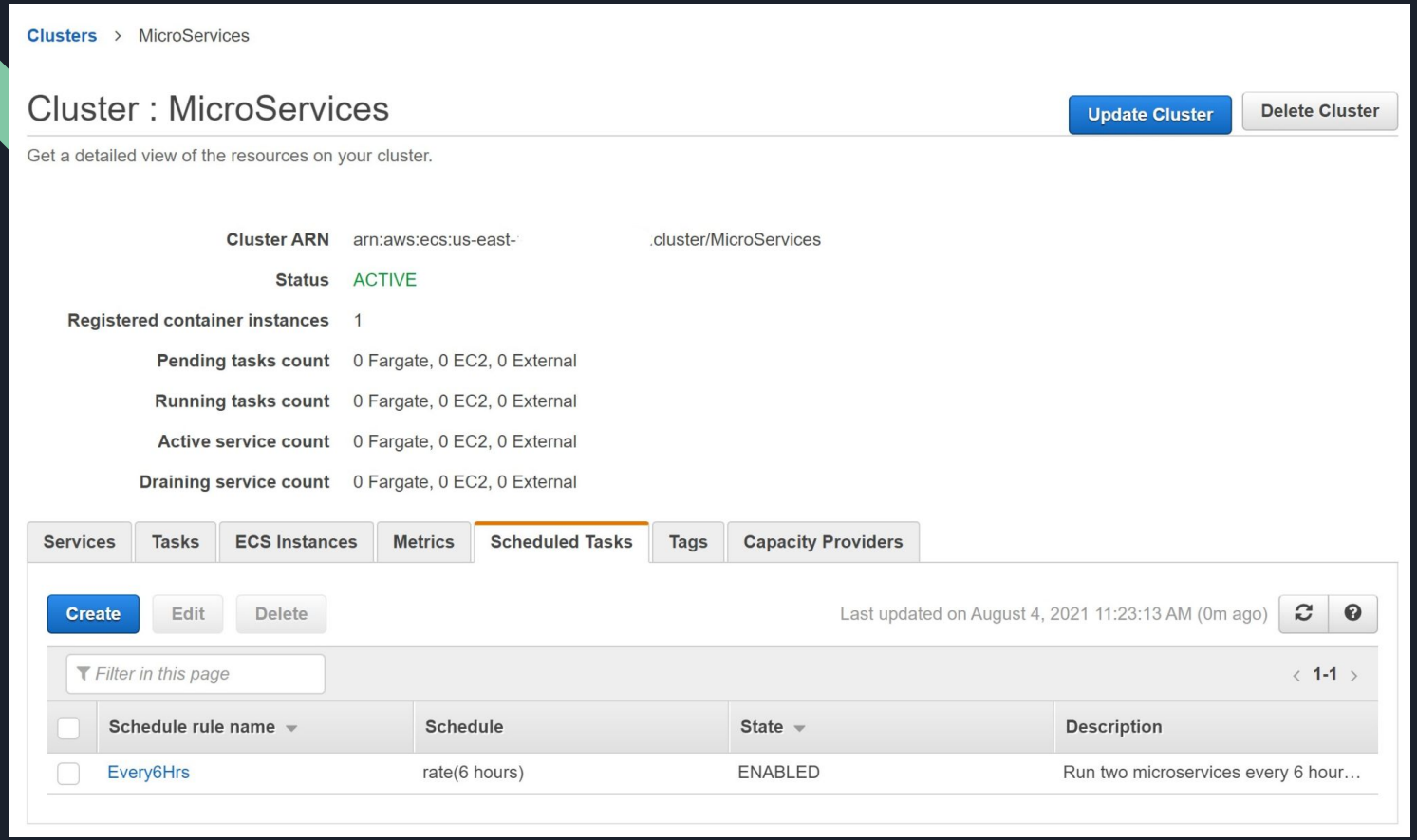
Function:

- Iterate through all events in the upcoming events table
- Remove events that have past and place them in the old events table
- Send a reminder email to the participants of immanent events

Microservice #2- Suggest Events

Function:

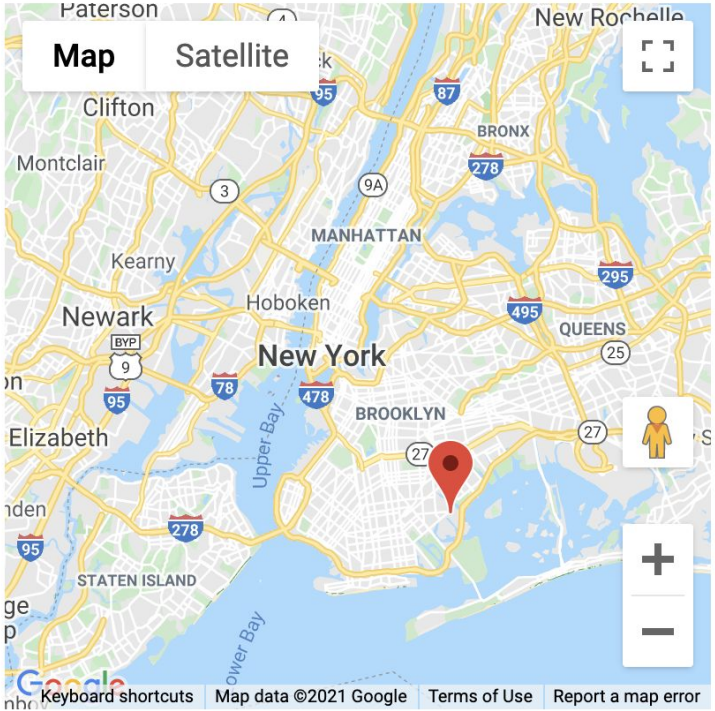
- For each user, count up the number of times each event tag is found in their past events
- Based on this data, rank upcoming events on how well they match this users preferences
- Suggest the top four matches to the user


Clusters > MicroServices
Cluster : MicroServices
Update Cluster
Delete Cluster
Get a detailed view of the resources on your cluster.
Cluster ARN arn:aws:ecs:us-east- .cluster/MicroServices
Status ACTIVE
Registered container instances 1
Pending tasks count 0 Fargate, 0 EC2, 0 External
Running tasks count 0 Fargate, 0 EC2, 0 External
Active service count 0 Fargate, 0 EC2, 0 External
Draining service count 0 Fargate, 0 EC2, 0 External
Services
Tasks
ECS Instances
Metrics
Scheduled Tasks
Tags
Capacity Providers
Create
Edit
Delete
Last updated on August 4, 2021 11:23:13 AM (0m ago)
Filter in this page
1-1
Schedule rule name
Schedule
State
Description
Every6Hrs
rate(6 hours)
ENABLED
Run two microservices every 6 hour…

Google maps Integration

- We retrieve the latitude and longitude from the place the user entered using an API call to Google Geocode.
- The latitude and longitude was then used to pull up a google map representing the location on the view events page using an api call to Google Maps.

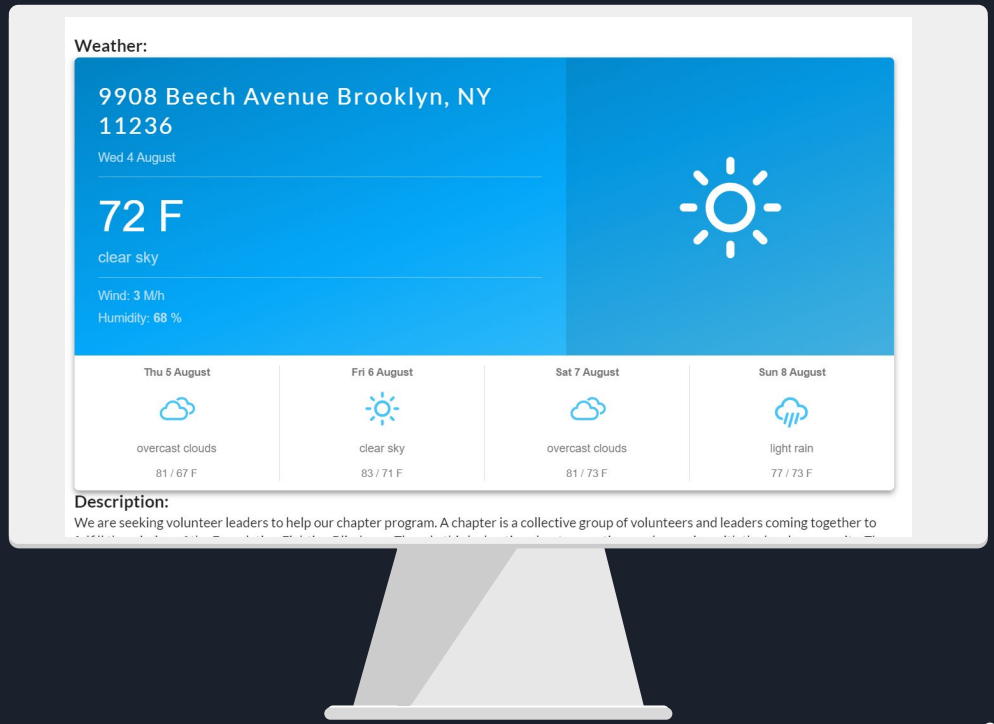
Location:
8253 W. Arnold St. Brooklyn, NY 11234



Date and time:
Friday, April 15, 2022, 3:32 AM

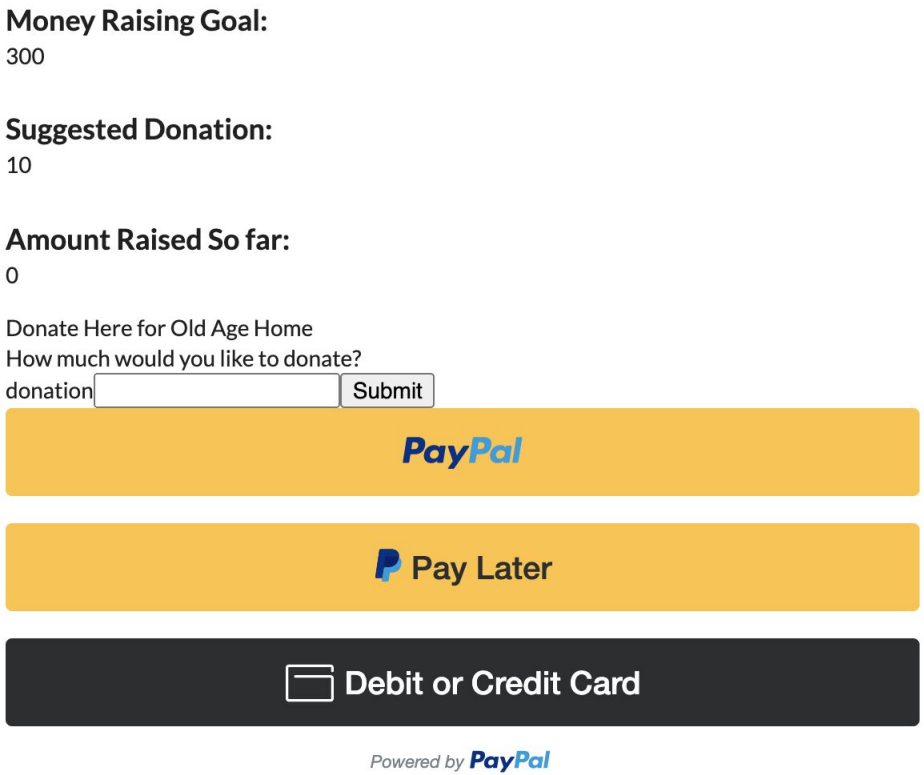
Weather Integration

- We called the react open weather api to show the weather so the user will know what the weather will be like at the time of the event.
- The weather forecast shows up on the view event page with all the other information about the event.



PayPal Integration

- We called the paypal api to give the user an option to donate to help raise money for events.
- When the user donates the amount of money raised so far in the database gets updated



How to Set Up the App

You will need:

- An active AWS account
- The AWS command line interface
- NodeJS with NPM
- AWS Amplify CLI (You can install this using NPM)
- Docker

How to Set Up the App

- Start by cloning the repository
  - `git clone https://github.com/azrielb1/Platinum-Volunteers.git`
- Run cloudformation templates
  - `aws cloudformation create-stack --stack-name API-Lambda-DB --template-body file://./cloudformation/API-Lambda-DB/template.json --parameters ParameterKey=LambdaFuncName,ParameterValue=CRMLambda ParameterKey=UsersTableName,ParameterValue=CRMUsersTable ParameterKey=APIName,ParameterValue=CRMAPI ParameterKey=EnvironmentName,ParameterValue=Prod --capabilities CAPABILITY_IAM`
  - `aws cloudformation create-stack --stack-name MicroServices --template-body file://./cloudformation/MicroServices/template.json`
- Complete API setup
  - enable cors
  - Add APIGateway as a trigger for the Lambda
    - Open AWS Console and go to Lambda
    - Choose CRMLambda
    - Choose "Add Trigger"
    - Select API Gateway and choose the CRMAPI
  - Add your endpoint URL to Platinum-Volunteers/src/APIFunctions.js as well as in Platinum-Volunteers/EC2_scripts/EventSuggesterMicroServ/APIHelperFunctions.js and Platinum-Volunteers/EC2_scripts/ProcessEventsMicroServ/APIHelperFunctions.js

How to Set Up the App

- Complete MicroServices setup
  - Create Docker image for both sub-directories

```
cd ProcessEventsMicroServ
docker build -t process_old_events .
cd ../EventSuggesterMicroServ
docker build -t Event_Suggester .
cd ..
```

  - Push images to ECR
  - Schedule task with ECS
    a. Go to AWS ECS console
    b. Create new task definition
    c. Use the images you pushed to ECR
    d. Schedule task to run every 6 hours on the cluster created by cloud formation
-

How to Set Up the App

- Set up Amplify backend with AWS Cognito for user authentication, and S3 for avatar storage.
  - Initialize the amplify project.

    ```
    amplify init
    ```
- Configure an Amazon Cognito User Pool to manage user credentials.

  ```
  amplify add auth
  # When asked how you why do users to sign in choose "Email"
  ```
- Configure an Amazon S3 bucket to store avatars.

  ```
  amplify add storage
  ```
- Deploy your backend.

  ```
  amplify push
  # When asked if you would like to generate client code, you can
  # say no since we are using plain JavaScript.
  ```
- You may set up the front end from the amplify console

How to Set Up the App

- Set up the .ENV file with API keys to enable various 3rd party integrations.
  - obtain a open weather API key and Google Maps API key
  - enter them in the .env file
- Install dependencies

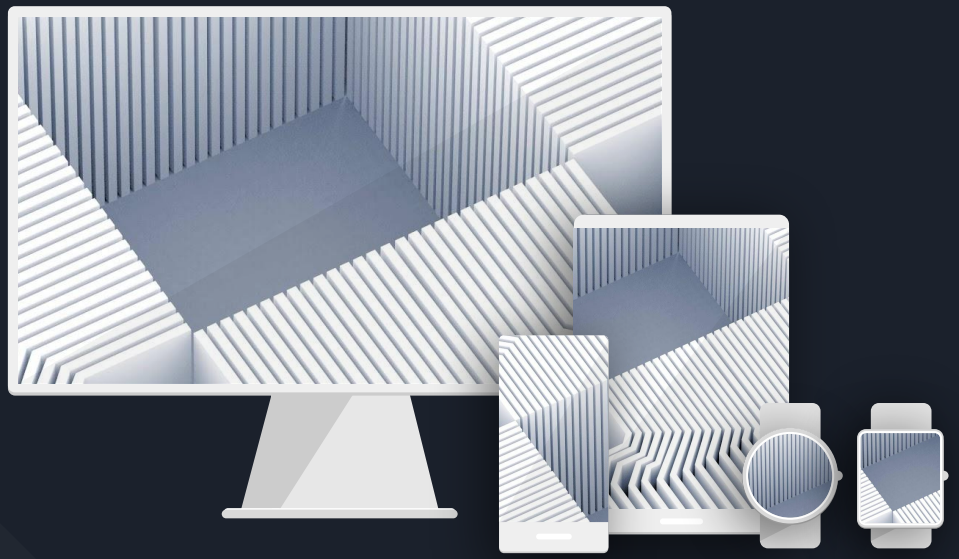
  ```
  npm i
  ```
- Run the app

  ```
  npm start
  ```

Main takeaways/lessons learned

- Agile development
- Using Git to collaborate effectively and efficiently with a team
- Familiarity with powerful AWS services such as Amplify, API Gateway, Lambda, DynamoDB, ECS, Cognito, SES, S3, and CodeCommit
- Experience with web development using React.js
- Experience designing and invoking REST APIs

Thank you!



Github:
https://github.com/azrielb1/Platinum-Volunteers