USERS MICROSERVICE

To be completed: Queries, Routers, Security (Authentication) + Roles (Authorization) , S3 / Local Stack (profile photo storage), Communication

Registration
POST /Users
GET /Token (i.e login, logout endpoints)

Roles
Admin, Volunteer, Organizer
Ads, Open ID Connect, OAuth, AuthV2

Upload Profile Photos
Local Stack
https://github.com/localstack/localstack#example

Svelte Kit S3
https://rodneylab.com/sveltekit-s3-compatible-storage/

USERS

Queries

User.py
Post method that utilizes hash password function, security regarding users
def db_create_user
Put method
def update_user
Delete method (protected endpoint)
def delete_user
Authenticator.py [or something with login/logout]
def login_user
def logout_user

Routers
User.py
@router.post
@router.put
@router.delete

Authenticator.py [login/logout]
@router.get
@router.delete

Admin: The admin role has full access and control over the application. They can manage users, create, update, and delete events, and have access to all features and functionalities.

- Manage Users
DELETE /users/{username}, {email}
PUT /users/{username}, {email}

- User Analytics
- Analytics Endpoint
- Matplotlib or some sort of Analysis
middleware/framework

- Admin Tasks [Protected Endpoint]
- [(MS) Users -> Tasks]

- CRUD Events (Full Suite)

- Communicate with admins, organizers, volunteers

Access to other features (not complete)
- Tasks [(MS) Users -> Tasks]
- Access Task Lists [GET]
- Mark as Complete [PUT /tasks/{task_id}, /tasklists/{tasklist_id}
- RSVP/Favorite [(MS) Users -> Events]

Communication between Users

Websockets (?)
or just Pub/Sub to a Communication Channel

Organizer: Organizers have the ability to create, update, and delete events. They can manage tasks, view RSVPs, and communicate with volunteers. However, they do not have administrative privileges such as user management.

- CRUD Events (Full Suite)

- Management of Tasks, View Rsvps, Communication

- Communicate with organizers, volunteers

Access to other features
- Tasks [(MS) Users -> Tasks]
- Access Task Lists [GET]
- Mark as Complete [PUT /tasks/{task_id}, /tasklists/{tasklist_id}
- RSVP/Favorite [(MS) Users -> Events]
- View Event Details [(MS) Users -> Events]

Volunteer: Volunteers can RSVP for events, view event details, and access task lists. They can mark tasks as complete and communicate with organizers or other volunteers. Volunteers have limited access compared to admins and organizers.

Access to other features
- Tasks [(MS) Users -> Tasks]
- Access Task Lists [GET]
- Mark as Complete [PUT /tasks/{task_id}, /tasklists/{tasklist_id}
- RSVP/Favorite [(MS) Users -> Events]
- View Event Details [(MS) Users -> Events]

Operation

3 paths being: (a) Create Account -> Logged In, (b) Log In, (c) Logout

(a)
User is creating an account (with unique email and username combo) and will be logged in following account creation

Create account and user chooses role (POST /users) -> Login to user's created account (/LOGIN) -> (GET /Token with role credentials) -> User Session (user logged in and has access to protected endpoints allowed by role

- (User Data) Create account allows one to input username, email, first name, last name, password, profile photo, about me section (`user.description`), profile photo (upload profile photo)

(i) POST /users
Ensure that user info being entered fulfills: (1) no duplicate accounts (ensure no same combo of username and email), (2) email is valid
(ii) Password being posted by user is hashed (Suggested: JWTS, OAuth, AuthV2)

- Stretch Goal
Select interests (drawn from a db that has all potential interests to match to a recommended event/events)

Unsuccessful Attempts (status code 422?)
- bad create account attempts -> tells user that 'x' is wrong
(a) user inputs data that is present with another user
- data that would present errors: same username and same email
(b) invalid username, invalid first or last, invalid email

(b)
User has previously created account and want to log in to use app
GET /USERS
user.password and user.username is correct
GET /token
Create user token, session

Ensure protected endpoints given logged in user and correct role

Unsuccessful Attempts (status code 400?)
(b) invalid username and invalid password
- bad login attempt -> tell user there is no user with 'x' username, 'y' password
- 'user with (x) username not found'
- 'password and username combo does not match'

(c)
Sign out [DELETE TOKEN and END SESSION]
- No user logged in, no token

GROUPS

Queries

Group.py
Post method that utilizes hash password function, security regarding users
def db_create_group

Put method
def update_group
- assign users to groups

Delete method (protected endpoint)
def delete_group

Routers

Group.py
@router.post

@router.put
PUT /groups
- groups/{group_id}/users
group.users.append()
(user1, user2, ...)

@router.delete

TBD

Assign Tasklist to a group
groups/ {group_id} /tasklist
group.tasklist

Either assign group to an event or group.tasklist.event relationship