Opperator after seed:

**Login**

POST: api/login

{

Email: string

Password: string

}

* Get user by email
* Check if user terminated
* Check password hash in db if match password
* Add new refreshToken to DB with new expiration date

// if true: returns userDto + new JWT + new RefreshToken

// If false: “Incorrect email or password”

**Refresh**

POST: api/refresh

{

Id: number

refreshToken

}

* Check Auth middleware
* Check If user Terminated
* Get user by Id
* Check if refreshToken matches refreshToken in DB
* Add new RefreshToken + new refreshToken Expiration

// if true: returns userDto + new JWT + new RefreshToken

// If false: “unable to authenticate user”

**Add Operator**

POST: api/add-operator

{

firstName: string

lastName: string

email: string

Role: int

SendRegistration: bool

}

* Run user auth middleware
* Run checkUserRole (sudo, admin)
* Generate unique key
* Add user to database { IsRegistered = false } + unique key + expiration date to unique key
* If SendRegistration = true, send registration

// if true: return newly added userDto

// if false:

* Email already exists: “user already registered”
* Else: “An error occurred adding the user”

**Register**

1 GET: /registration/userId?uniqueKey

* Check if user registered before

// return register page with create password

// client side verify that password and confirm password match

2 POST: api/Register

{

UserId: int

Password: string

uniqueKey: string

}

* Get user by Id
* Check if user registered before
* Check if uniqueKey matches the one stored in db + unique key not expired
* Remove uniqueKey + set isRegistered to true + add refresh token

// if true: return userDto + refresh token + JWT

// if false:

If user registered OR uniqueKeyExpired: “Session Expired.”

If password too short: “Please make sure the password is 10 characters long, including at least 1

capital, lowercase, number and special character”

else: “Unable to register user.”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

GET: api/Toggle-termination/:userId

* Run user auth middleware
* Run is checkUserRole (sudo, admin)
* Get user by Id and Toggle termination

// if success: return true

// if fail: return “unable to terminate user”

**Edit user details**

PATCH: api/edit-user/:userId

{

opId: Number

firstName: string | null

lastName: string | null

email: string | null

}

- Run user auth middleware

- Run Check user role (Sudom admin)

- get user by Id and Edit User Details

// If Success: Return Updated UserDto

// If fail: Return “Unable to Update User Details”

**Edit Role**

PATCH: api/edit-Role/:userId

{

Role: string

}

- Run user auth middleware

- Run Check user role (Sudo, admin)

- get user by Id and Edit User Role

// If Success: Return Updated UserDto

// If fail: Return “Unable to Update User Role”

**Forgot password**

POST: api/forgot-password/:userId

{

Email: string

}

- get user By email

- Check If registered && is not terminated

- Update user with unique key

- send unique key in a url to user also containing userId

// If Success: Return “An Email Sent has been sent to reset the password”

// If fail: Return “An Email Sent has been sent to reset the password”

**logout**

GET: api/logout/:userId

- Run user auth middleware

- get user By Id

- Delete the Refresh token for this users session

// If Success: Return “You have logged out successfully”

// If fail: Return “An Error occurred while logging you out”

**Logout all devices**

GET: api/logout-all/:userId

- Run user auth middleware

- get user By Id

- Delete the Refresh token for this users session

// If Success: Return “You have logged out all devices successfully”

// If fail: Return “An Error occurred while logging you out”

**View user**

GET: api/operator/:userId

- Run user auth middleware

- Run Check user role (Sudo, admin)

- get user By Id

// If Success: Return UserDto

// If fail: Return “An error occurred while loading user data”

**View all users**

GET: api/operators

- Run user auth middleware

- Run Check user role (Sudo, admin)

- get users

// If Success: Return UserDto

// If fail: Return “An error occurred while loading the users”