Logic Disparity

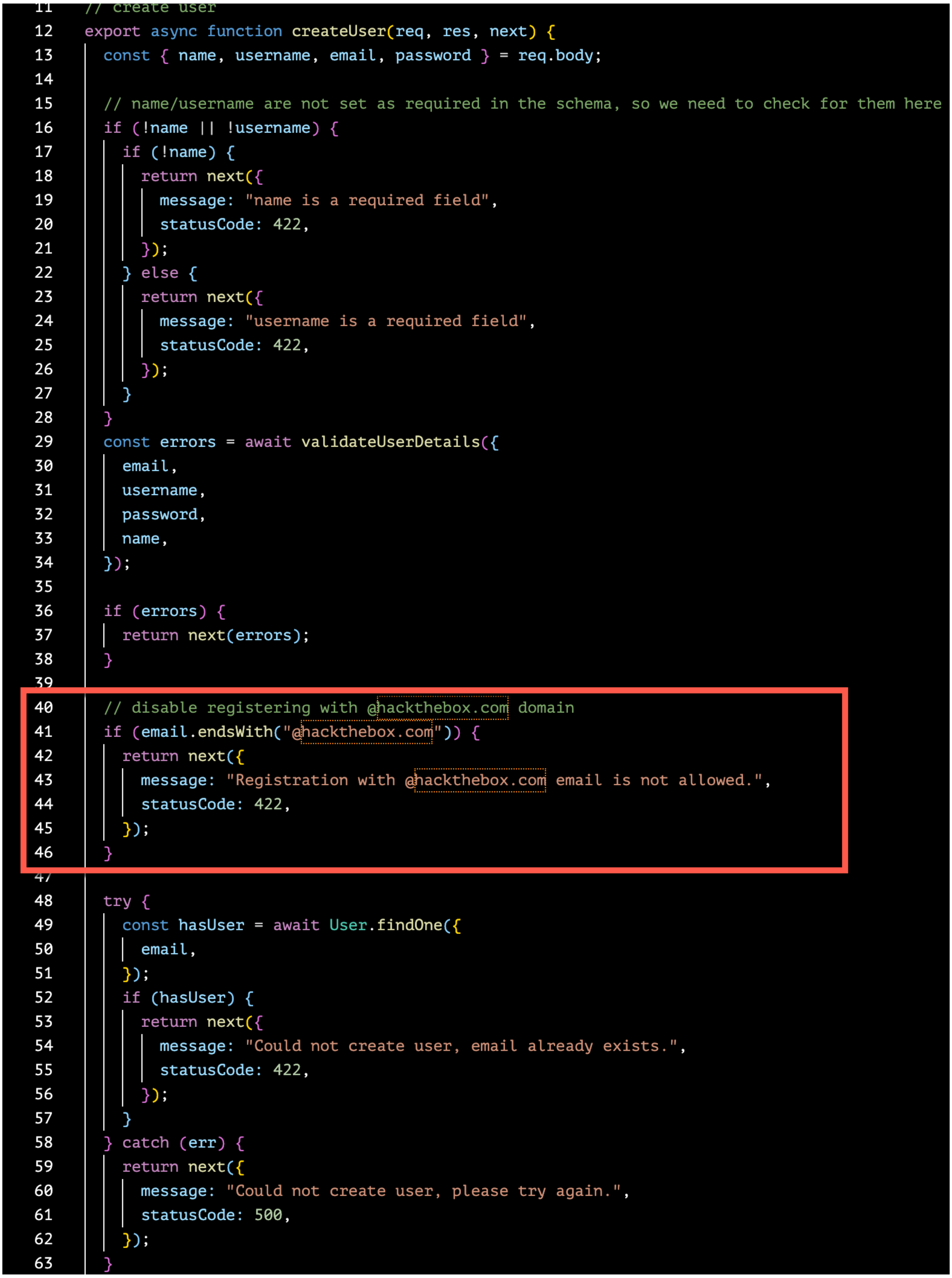
Summary

In this case, user registration prevents the user from registering where the emails ends with hackthebox.com

However, the update profile functionality does not have such restrictions.

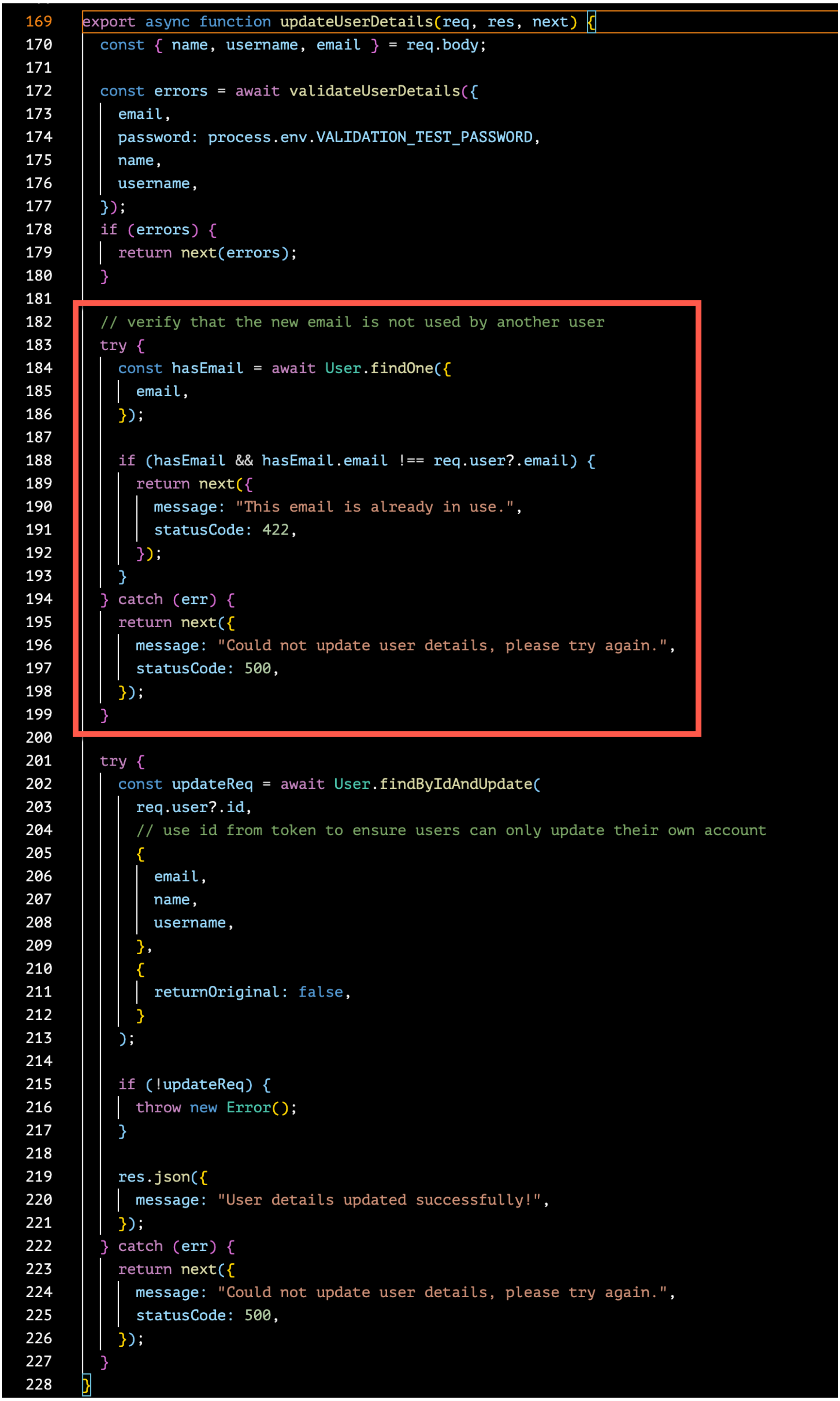
As a result, it was possible to abuse the update user profile functionality to obtain unlimited cubes.

User registration



```
// create user
export async function createUser(req, res, next) {
  const { name, username, email, password } = req.body;

  // name/username are not set as required in the schema, so we need to check for them here
  if (!name || !username) {
    if (!name) {
      return next({
        message: "name is a required field",
        statusCode: 422,
      });
    } else {
      return next({
        message: "username is a required field",
        statusCode: 422,
      });
    }
  }
  const errors = await validateUserDetails({
    email,
    username,
    password,
    name,
  });

  if (errors) {
    return next(errors);
  }

  // disable registering with @hackthebox.com domain
  if (email.endsWith("@hackthebox.com")) {
    return next({
      message: "Registration with @hackthebox.com email is not allowed.",
      statusCode: 422,
    });
  }

  try {
    const hasUser = await User.findOne({
      email,
    });
    if (hasUser) {
      return next({
        message: "Could not create user, email already exists.",
        statusCode: 422,
      });
    }
  } catch (err) {
    return next({
      message: "Could not create user, please try again.",
      statusCode: 500,
    });
  }
```

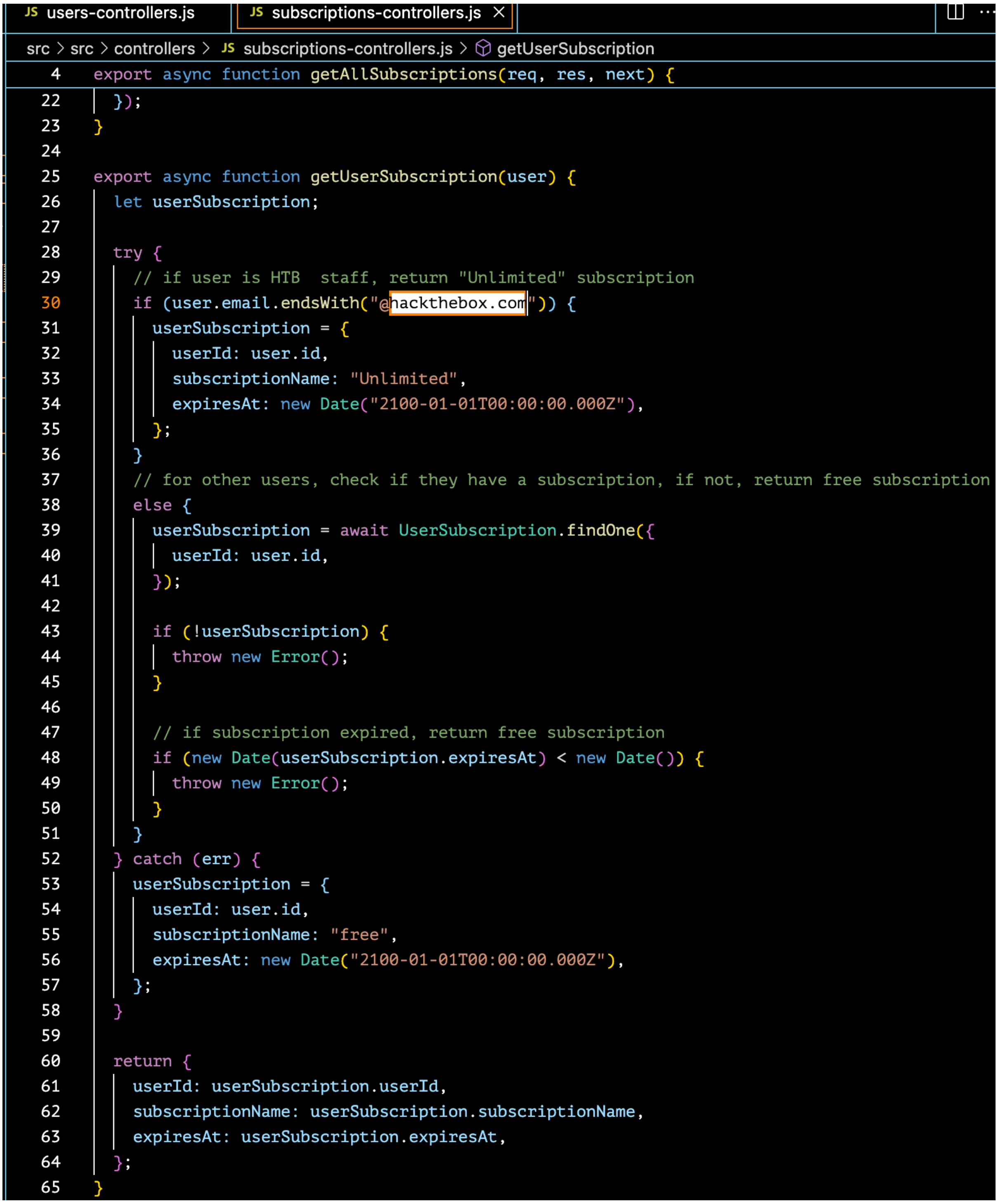
Update profile



```javascript
export async function updateUserDetails(req, res, next) {
  const { name, username, email } = req.body;

  const errors = await validateUserDetails({
    email,
    password: process.env.VALIDATION_TEST_PASSWORD,
    name,
    username,
  });
  if (errors) {
    return next(errors);
  }

  // verify that the new email is not used by another user
  try {
    const hasEmail = await User.findOne({
      email,
    });

    if (hasEmail && hasEmail.email !== req.user?.email) {
      return next({
        message: "This email is already in use.",
        statusCode: 422,
      });
    }
  } catch (err) {
    return next({
      message: "Could not update user details, please try again.",
      statusCode: 500,
    });
  }

  try {
    const updateReq = await User.findByIdAndUpdate(
      req.user?.id,
      // use id from token to ensure users can only update their own account
      {
        email,
        name,
        username,
      },
      {
        returnOriginal: false,
      }
    );

    if (!updateReq) {
      throw new Error();
    }

    res.json({
      message: "User details updated successfully!",
    });
  } catch (err) {
    return next({
      message: "Could not update user details, please try again.",
      statusCode: 500,
    });
  }
}
```

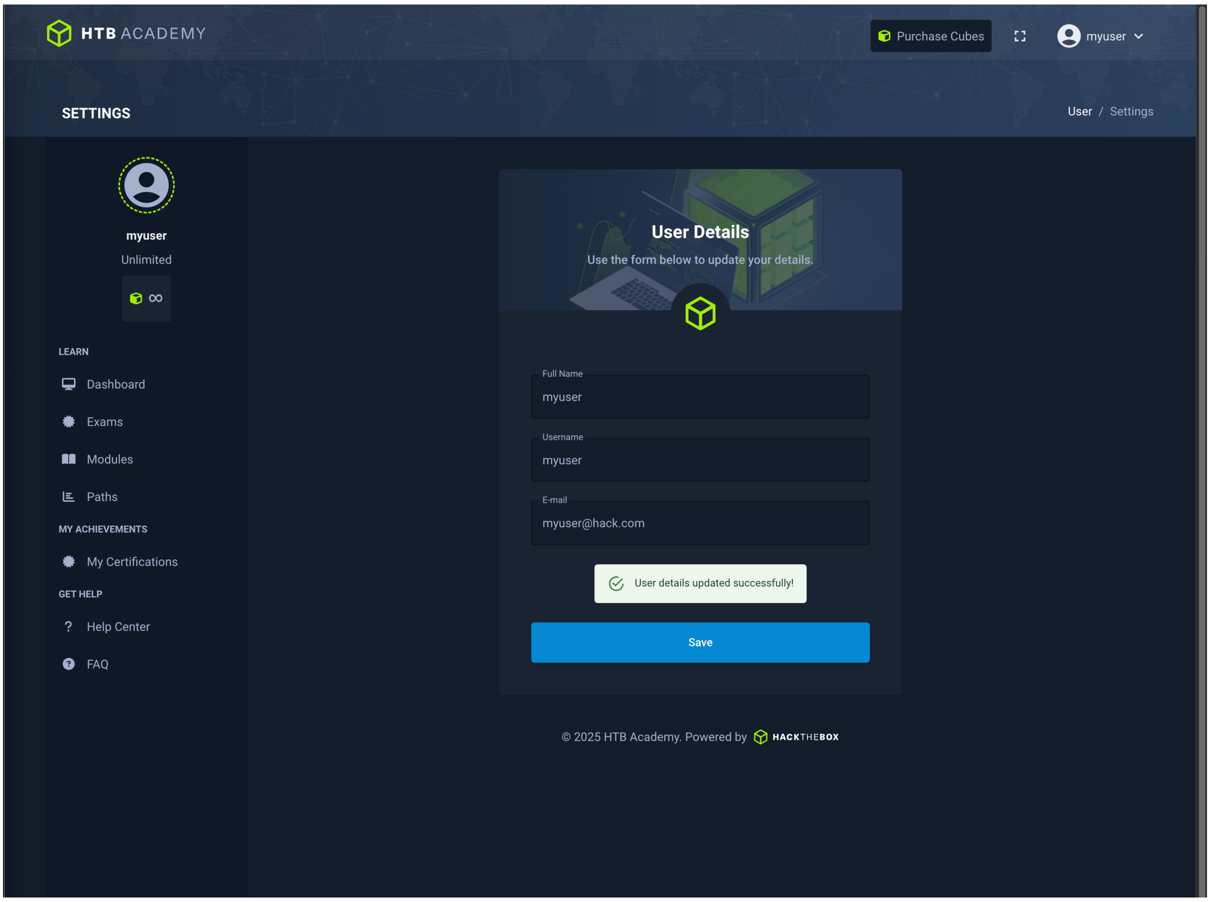
Subscription controller

As you can see, emails that ends with hackthebox.com has unlimited number of cubes.


JS users-controllers.js
JS subscriptions-controllers.js
src > src > controllers > JS subscriptions-controllers.js > getUserSubscription

```javascript
export async function getAllSubscriptions(req, res, next) {
  });
}

export async function getUserSubscription(user) {
  let userSubscription;

  try {
    // if user is HTB  staff, return "Unlimited" subscription
    if (user.email.endsWith("@hackthebox.com")) {
      userSubscription = {
        userId: user.id,
        subscriptionName: "Unlimited",
        expiresAt: new Date("2100-01-01T00:00:00.000Z"),
      };
    }
    // for other users, check if they have a subscription, if not, return free subscription
    else {
      userSubscription = await UserSubscription.findOne({
        userId: user.id,
      });

      if (!userSubscription) {
        throw new Error();
      }

      // if subscription expired, return free subscription
      if (new Date(userSubscription.expiresAt) < new Date()) {
        throw new Error();
      }
    }
  } catch (err) {
    userSubscription = {
      userId: user.id,
      subscriptionName: "free",
      expiresAt: new Date("2100-01-01T00:00:00.000Z"),
    };
  }

  return {
    userId: userSubscription.userId,
    subscriptionName: userSubscription.subscriptionName,
    expiresAt: userSubscription.expiresAt,
  };
}
```

Evidence that profile update was successful



User Details

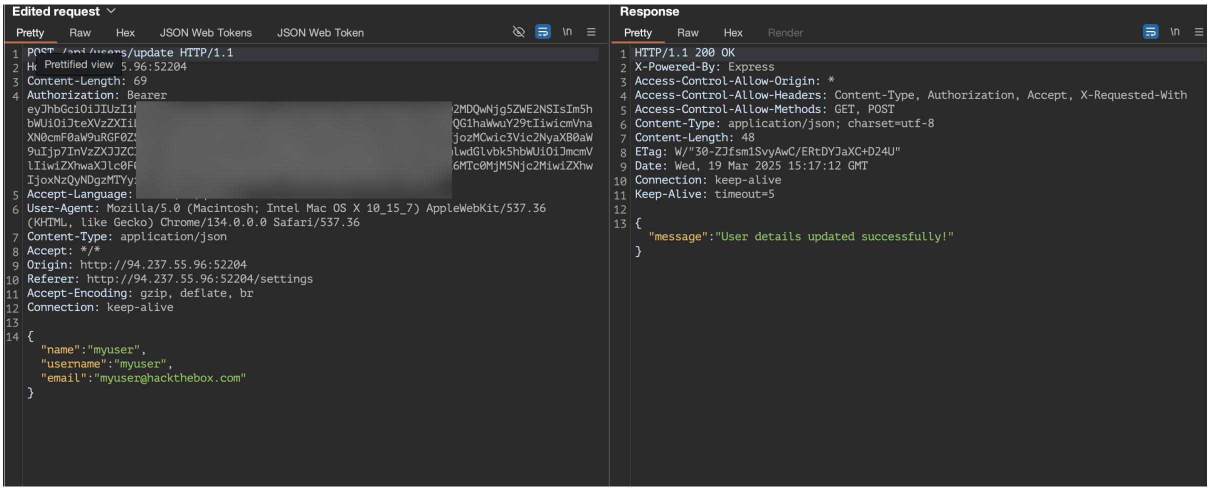
Use the form below to update your details.

Full Name: myuser

Username: myuser

E-mail: myuser@hack.com

✓ User details updated successfully!



```
POST /api/users/update HTTP/1.1
H                 5.96:52204
Content-Length: 69
Authorization: Bearer
eyJhbGciOiJIUzI1N [redacted] 2MDQwNjg5ZWE2NSIsIm5h
bWUiOiJteXVzZXIiL                    QG1haWwuY29tIiwicmVna
XN0cmF0aW9uRGF0ZS                    jozMCwic3Vic2NyaXB0aW
9uIjp7InVzZXJJZCI                    lwdGlvbk5hbWUiOiJmcmV
lIiwiZXhwaXJlc0F                     6MTc0MjM5Njc2MiwiZXhw
IjoxNzQyNDgzMTYy
Accept-Language:
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://94.237.55.96:52204
Referer: http://94.237.55.96:52204/settings
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

{
  "name":"myuser",
  "username":"myuser",
  "email":"myuser@hackthebox.com"
}
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Content-Type, Authorization, Accept, X-Requested-With
Access-Control-Allow-Methods: GET, POST
Content-Type: application/json; charset=utf-8
Content-Length: 48
ETag: W/"30-ZJfsm1SvyAwC/ERtDYJaXC+D24U"
Date: Wed, 19 Mar 2025 15:17:12 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
  "message":"User details updated successfully!"
}
```