API Endpoints

URL: (hidden)

BUSINESSACCOUNTS → primary key is email

Each account will have a primary key that is an email, username, password, a list of created resources, and a secret key that is automatically generated on account creation.

post a new business account

POST http://(hidden)/businessaccount/

→ can post with postman or your languages HTTP handling. For example in python you can make a post request like:

ACCOUNTS → primary key is email

Each account will have a primary key that is an email, username, password, a balance, and a list of resources the user owns.

post a new account

POST http://(hidden)/useraccount/

→ Can post with postman, or your languages HTTP handling. For example, in python you can make a post request like:

```
import requests

url = 'http://localhost:8080/useraccount'


request_data={
    "email":"Testemail@gmail.com",
    "username": "testuser",
    "password": "passwordtest"
}

x = requests.post(url, json = request_data)
```

→ will check if email is already in database

→ if not taken, create an email with balance and content owned to 0, and

resources to an empty list

→ This will return a status code of 200 if success, and an error code/message if it fails.

Edit the balance of an account

PATCH http:///(hidden)/useraccount/<email>/balance

→ Can patch with postman, or your language's HTTP handling. For example, in python you can make a patch request like:

```
import requests

url = 'http://localhost:8080/useraccount/test_email@gmail.com/balance'
body = {
    "balance": 500
    }

x = requests.patch(url, json=body)
```

→ update the user's monetary balance to the balance variable given

Add a resource to an account's list of owned content

PATCH http:///(hidden)/useraccount/<email>/resources → Can post with postman, or your language's HTTP handling. For example, in python you can make a patch request like:

```
import requests

url = 'http://localhost:8080/useraccount/test@gmail.com/resources'

body = {
    "resource_id": "sadfcpIOUSFB",
}
x = requests.patch(url, json = body)
```

→ update the user's content owned by appending the resource_id given in the URL

Get information of all user accounts in database

GET http:///(hidden)/useraccount

→ return a list of all of the users in the database, along with their balance and resources list

→ example in python:

```python
import requests

url = 'http://localhost:8080/useraccount'

x = requests.get(url)


print(x.text)
```

→ example output:


```
{
  "balance": 0.0,
  "password": "$2b$12$kTlLl6nxxN7hC55DmQyxjOMti.ubHpN5lYOLXC0gnREBOWid/cl2y",
  "resources": [
    "sadfcpIOUSFB"
  ],
  "username": "test",
  "email": "test@gmail.com"
},
{
  "balance": 2500.0,
  "password": "$2b$12$rdGEXP0A/dw0ZfbRzDT7POB3jHFcSv/ZbP0ycdIIOHN3ho3vf.4M6",
  "resources": [
    15.0,
    "test"
  ],
  "username": "Testusername",
  "email": "test@oregonstate.edu"
},
```


Get a specific accounts information by specifying an email address

GET http://(hidden)/useraccount/<email>

→ return the user's email,username, password balance, and content owned in json format

→ example input:

```python
import requests

url = 'http://localhost:8080/useraccount/test@oregonstate.edu'


x = requests.get(url)


print(x.text)
```

→ example output:

```
shoemakd@Olympus:~$ python3 curl_test.py
{
  "balance": 25.0,
  "password": "Testpassword",
  "resources": [
    15.0
  ],
  "username": "Testusername",
  "email": "test@oregonstate.edu"
}
```

Get balance associated with an account

GET http://(hidden)/useraccount/<email>/balance → return the user's account balance integer

→ example input:

```
import requests

url = 'http://localhost:8080/useraccount/test@oregonstate.edu/balance'


x = requests.get(url)


print(x.text)
```

→ example output:

```
shoemakd@Olympus:~$ python3 curl_test.py
750
```

Get the list of resources an account owns

GET http://(hidden)/useraccount/<email>/resources → return a list of resource_id's the given user owns.

→ example input:

```
import requests

url = 'http://localhost:8080/useraccount/test@oregonstate.edu/resources'


x = requests.get(url)


print(x.text)
```

→ Example output:

```
shoemakd@Olympus:~$ python3 curl_test.py
[
  "sadfcpIOUSFB"
]
```

Attempt to login with a given username and password

GET http://(hidden)/useraccount/login/username/<username>/password/<password> → If the accounts username and password match, return the information of the account.

→ If the username or password do not match, give back an error message.

→ example of a get request:

```
import requests

url = 'http://localhost:8080/useraccount/login/username/logintest/password/Testpassword'


x = requests.get(url)


print(x.text)
```

→ example of a successful login:

```
shoemakd@Olympus:~$ python3 curl_test.py
{
  "balance": 0,
  "email": "logintest@oregonstate.edu",
  "password": "$2b$12$hfvLFwCnuKL83bNDh8kAY.nLZoogzo9tTBpyM8nKOrCPagxvICLmC",
  "resources": [],
  "username": "logintest"
}
```

→ example of failed login:

```
shoemakd@Olympus:~$ python3 curl_test.py
{
  "Fail": "Passwords do not match"
}
```

or

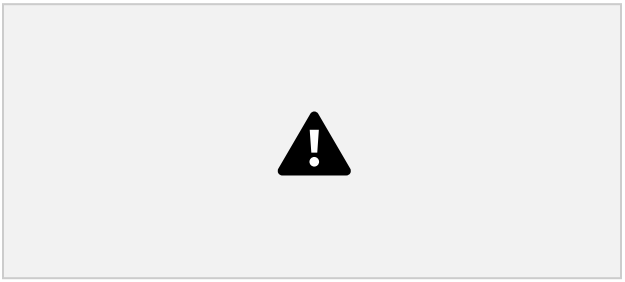
RESOURCES → primary key is resource_id (can create random ID when posting)

Each resource will have a resource_id, a cost, and a decryption key.

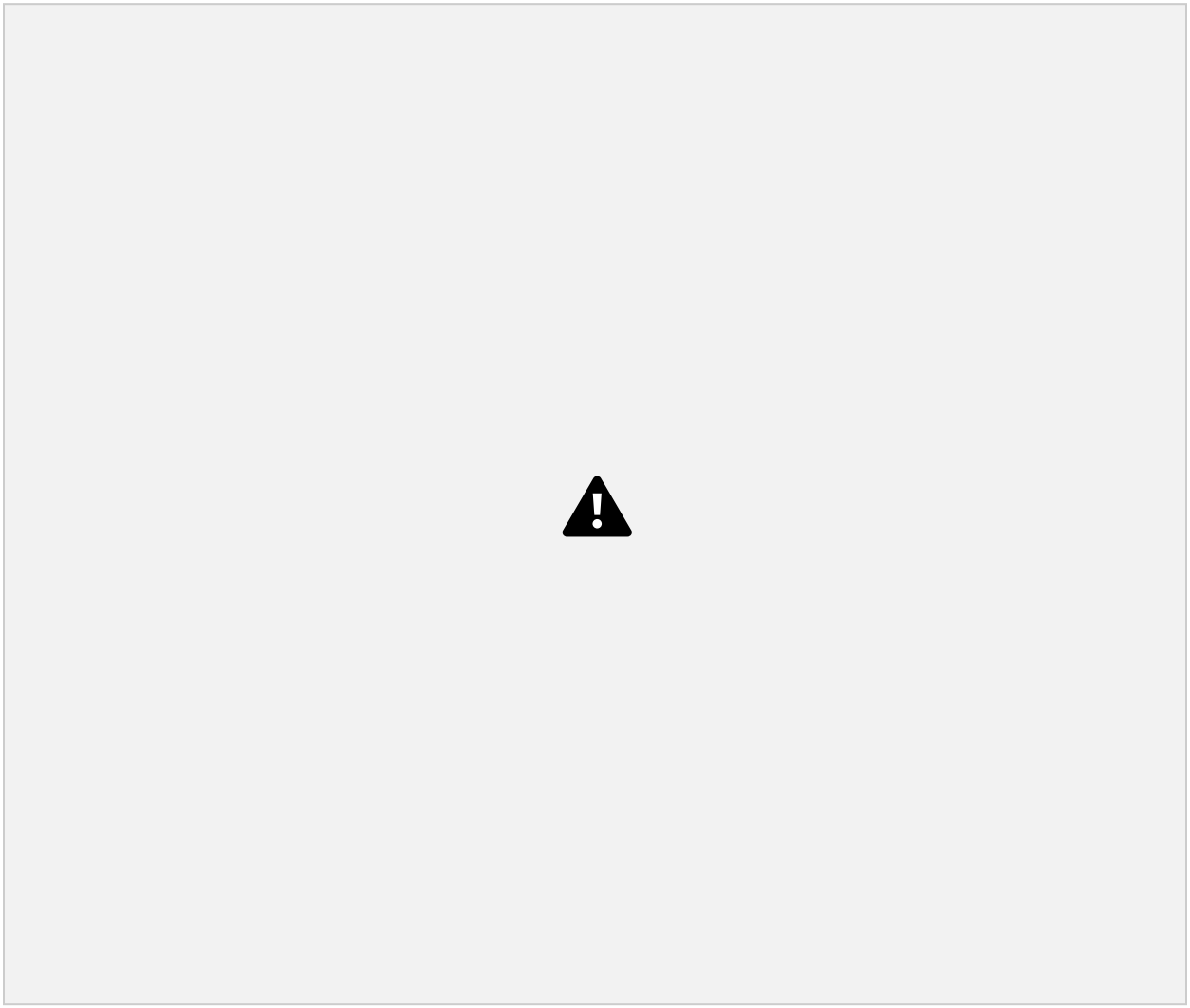
Create a new resource

POST http://(hidden)/resource

→ To create a resource, you must enter in header values into the request. The header values:



APP_ID is the email address of the business account.
Nonce is a unique identifier decided at the time of sending the request(it is used in the hmac verification process)
Timestamp is to verify that the request is not too old (older than 2 seconds). It is in epoch time.
Finally, Signature is the hmac hashing signature that is sent over that is a hash of the nonce and the secret key that each business account has.
An example of a request in python:



Also, here is the SHA512 code using the CryptoJS library in postman. May help with coding the requests using javascript:

→ check if resource already exists
→ if doesn't exist, create it with a cost that is given in body of HTTP request, and with a decryption key also provided.

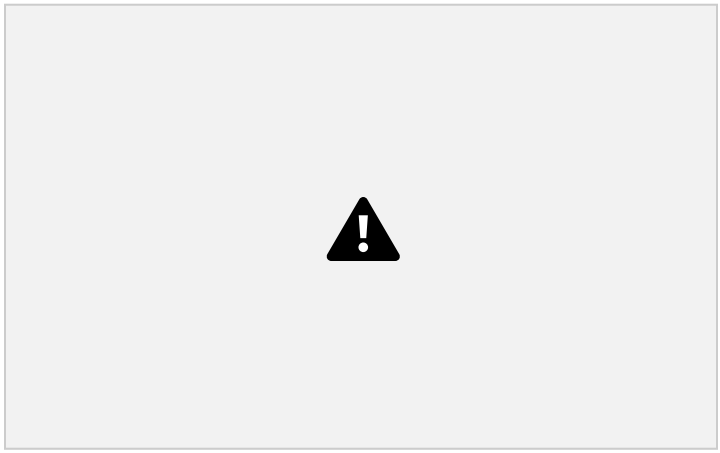
Get a list of all the resources in the database

GET http://(hidden)/resource

→ return a list of all resources in database with their costs and decryption keys

→ example request in python:

→ example output:



GET http://(hidden)/resource/<resource_id>

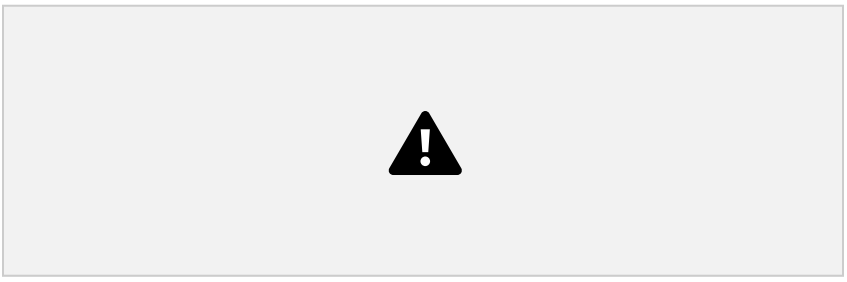
→ Return the all information about a specified resource

→ example output:

Get the cost that is associated with specified resource

GET http://5(hidden)/resource/<resource_id>/cost → return the cost integer of a given resource

→ example request in python:



→ example output:

Get the decryption key associated with a resource

GET http://(hidden)/resource/<resource_id>/dkey → return the decryption key string associated with a resource specified → example request in python:

→ example output:

Edit the cost associated with a resource

PATCH http://(hidden)/resource/<resource_id>/cost

→ update the cost associated with a resource.

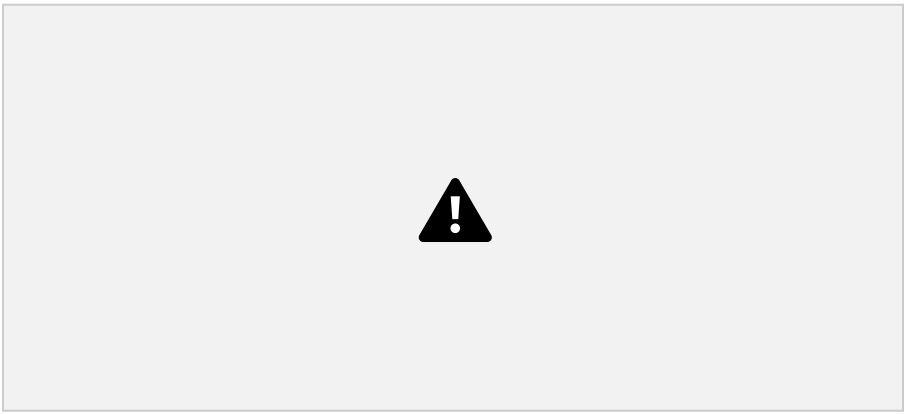
→ grab the cost from the body of the HTTP request.

→ example of json body:

Edit the decryption key associated with a resource

PATCH http://(hidden)/resource/<resource_id>/dkey → update the decryption key associated with a resource.

→ grab the dkey from the body of the HTTP request.
→ example patch request in python:



confirm a transaction
POST http://(hidden)/confirmtransaction/email/<email>/resource/<resource_id>

→ Will check if the user has enough funds (allows a -$25 balance).
→ Checks if user already owns resource
→ If conditions met, will place the resource ID into the user's list of owned resources.

Example of a confirm transaction request: