Web Back-End Engineering Project 2

Course	CPSC-449	
Section	01	
Semester	Fall 2023	
Group Number	10	
Team Members	**Name**	**CWID**
1	David Harboyan	887059103
2	Micah Baumann	885449769
3	Ornella Irene Dsouza	885188318
4	Viditi Vartak	885146886

Problem statement:

Create a new user authentication service with read replication, and use it to implement authentication and load balancing through an API gateway.

Code repository:

https://github.com/micahbaumann/CPSC-449-Project-2

Tasks

Implement an authentication service

Created 4 new endpoints for this task

Method	Endpoint (fastapi)	Endpoint (krakend)	Description
POST	/register	/user/register	Registers a new user with appropriate roles
POST	/login	/user/login	Logs in registered user
POST	/checkpwd	/user/checkpwd	Checks if the password is correct
GET	/getuser/{uid}	/user/get/{uid}	Gets a user's information from the users database

Database requirements

3 New tables created for Users service

Table Name	**Column**	**Keys**
Registrations	UserId Username FullName Email UserPassword BearerToken	UserId - Primary Key
Roles	RoleId Rolename	RoleId - Primary Key

Userroles	Id RoleId UserId	Id - Primary Key RoleId - Foreign Key referencing RoleId in roles table UserId - Foreign Key referencing Userid in Registrations table

Configure authentication through the API gateway

krakend.json | primary.yml | secondary_1.yml | secondary_2.yml | mkjwk.py 1 | private.json | public.json

CPSC-449-Project-2 > etc > mkjwk.py > ...

```python
import os
import sys
import json

from jwcrypto import jwk


def usage():
    program = os.path.basename(sys.argv[0])
    print(f"Usage: {program} KEY_ID...", file=sys.stderr)


def generate_keys(key_ids):
    keys = [jwk.JWK.generate(kid=key_id, kty="RSA", alg="RS256") for key_id in key_ids]
    exported_keys = [
        key.export(private_key=private) for key in keys for private in [False, True]
    ]
    keys_as_json = [json.loads(exported_key) for exported_key in exported_keys]
    public_key = {"keys": [keys_as_json[0]]}
    private_key = {"keys": [keys_as_json[1]]}
    public_output = json.dumps(public_key, indent=4)
    private_output = json.dumps(private_key, indent=4)
    print("Public Key:")
    print(public_output)
    print("\nPrivate Key:")
    print(private_output)


if __name__ == "__main__":
    if len(sys.argv) == 1:
        usage()
        sys.exit(1)

    generate_keys(sys.argv[1:])
```

4 New Endpoints

Endpoint1: Register new user with appropriate role

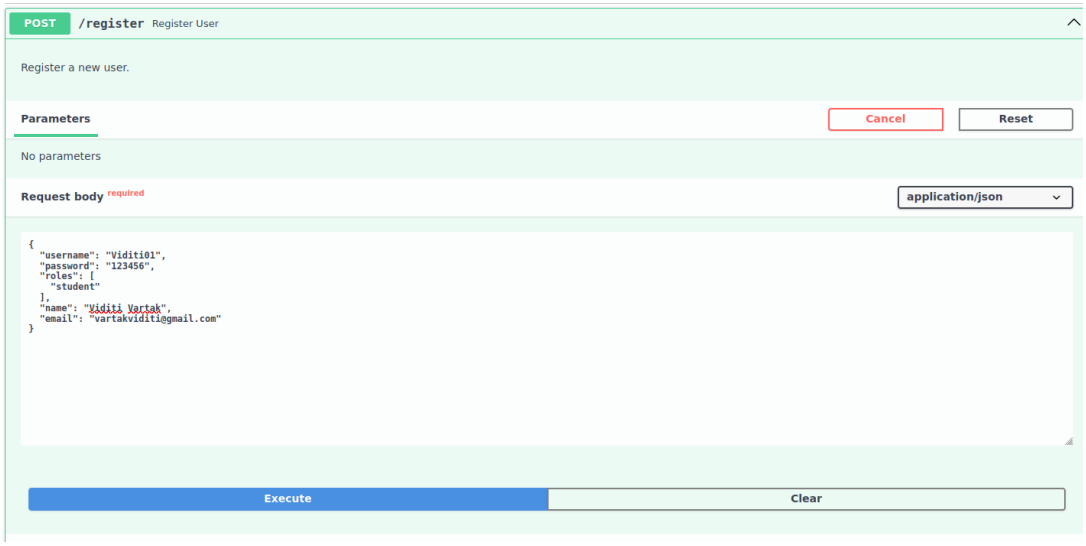
Task: To register a new user

Logic: 1. The user needs to fill in details by specifying the “username”, “password”, “role”, “name” and “email”.

2. The username must be unique and if the username is already used before, raise HTTPException error - return status code 400 and also display the error as "Username already used, try a new username".

3. If all the details are correct, then the user will be successfully registered.

Output:



Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5000/register' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "username": "Viditi01",
  "password": "123456",
  "roles": [
    "student"
  ],
  "name": "Viditi Vartak",
  "email": "vartakviditi@gmail.com"
}'
```

Request URL

```
http://localhost:5000/register
```

Server response

Code	Details
200	**Response body**

```
{
  "status": "200 OK",
  "message": "User Viditi01 successfully registered with role ['student']."
}
```

Download

Response headers

```
content-length: 92
content-type: application/json
date: Fri,27 Oct 2023 18:01:39 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Endpoint2: Logs in registered user

Task: To login a user

Logic: 1. The user needs to enter his/her username and password to login.

2. If the username or password is incorrect, raise HTTPException error - return status Code 400 and also display the error message as "Incorrect username or password."
3. If both username and password are correct, then the response body would return an access token, name, and email of the user.

Output:

POST `/login` Login

Login an existing user and generate JWT token for future requests.

Parameters Cancel Reset

No parameters

Request body required application/json

```
{
  "username": "Viditi01",
  "password": "123456"
}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5000/login' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "username": "Viditi01",
  "password": "123456"
}'
```

Request URL

```
http://localhost:5000/login
```

Server response

Code	Details
200	**Response body**

```
{
  "access_token": {
    "aud": "localhost:5200",
    "iss": "localhost:5200",
    "sub": "Viditi01",
    "jti": "14",
    "roles": [
      "student"
    ],
    "exp": 1698432514,
    "name": "Viditi Vartak",
    "email": "vartakviditi@gmail.com"
  }
}
```

Download

Response headers

```
content-length: 185
content-type: application/json
date: Fri,27 Oct 2023 18:18:33 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Endpoint3: Checks if the password is correct

Task: To verify if the entered password is correct

Logic: 1. The user needs to specify the “username” and “password”.

2. If the entered username and password is correct, it will display the Message as “ password correct”.
3. If the username and password is incorrect it will raise a HTTPException error - return status Code 400 and also display the error message as “ Incorrect username or password”.

Output:

Correct Password:

POST /checkpwd Checkpwd

Check if the password is correct or not.

Parameters Cancel Reset

No parameters

Request body required application/json

```
{
  "username": "Viditi01",
  "password": "123456"
}
```

Execute Clear

Responses

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5000/checkpwd' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "username": "Viditi01",
  "password": "123456"
}'
```

Request URL

```
http://localhost:5000/checkpwd
```

Server response

Code	Details
200	**Response body**

```
{
  "detail": "Password Correct"
}
```

Download

Response headers

```
content-length: 29
content-type: application/json
date: Fri,27 Oct 2023 18:26:59 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Wrong Password:

POST /checkpwd Checkpwd

Check if the password is correct or not.

Parameters Cancel Reset

No parameters

Request body required application/json

```
{
  "username": "Viditi01",
  "password": "12346"
}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5000/checkpwd' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "username": "Viditi01",
  "password": "12346"
}'
```

Request URL

```
http://localhost:5000/checkpwd
```

Server response

Code	Details
400 *Undocumented*	Error: Bad Request

Response body

```
{
  "detail": "Incorrect username or password"
}
```

Download

Response headers

```
content-length: 43
content-type: application/json
date: Fri,27 Oct 2023 18:30:47 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Endpoint4: Gets a user's information from the users database

Task: To get user's information by uid.

Logic: 1. To get the user by entering appropriate user Id.

2. If the entered uid is incorrect then it will raise a HTTPException error - return status Code 400 and also display the error message as " No user found".
3. If the specified uid is correct it will display the details of that user.

Output:

GET /getuser/{uid} Getuser

Parameters Cancel

Name	Description
uid * required integer *(path)*	1

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5000/getuser/1' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/getuser/1
```

Server response

Code	Details

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5000/getuser/1' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/getuser/1
```

Server response

Code	Details
200	**Response body**

```
{
  "email": "fsmith@csu.fullerton.edu",
  "name": "Fara Smith",
  "userid": 1,
  "username": "fara",
  "roles": [
    "Student"
  ]
}
```

Download

Response headers

```
content-length: 105
content-type: application/json
date: Sat,28 Oct 2023 18:21:43 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

GET /getuser/{uid} Getuser

Parameters Cancel

Name	Description
uid * required integer *(path)*	22

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5000/getuser/22' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/getuser/22
```

Server response

Code	Details
400 *Undocumented*	Error: Bad Request **Response body**

Server response

Code	Details
400 *Undocumented*	Error: Bad Request **Response body**

```
{
  "detail": "No User Found"
}
```

Download

Response headers

```
content-length: 26
content-type: application/json
date: Sat,28 Oct 2023 18:25:15 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response Media type	*No links*

Configure load balancing for the enrollment service

For this step, we restarted Foreman with the command:

foreman start --formation krakend=1,users=1,enroll=3

This creates 1 instance of krakend, 1 instance of the users service, and 3 instances of the enroll service. We then confirmed that we could access each instance separately.

After that we configured load balancing to round robin between the 3 instances of the enroll service by going into our krakend.json and updating the host of each endpoint belonging to the enroll service. Here are a few examples of updated endpoint hosts in krakend.json (Note: These screenshots were taken after the adding LiteFS, which caused our port numbers to change for the 3 instances of the enroll service. The main thing that changed was the port numbers.):

```
{
    "endpoint": "/student/list",
    "method": "GET",
    "backend": [
    {
        "url_pattern": "/list",
        "method": "GET",
        "host": [
            "http://localhost:5300",
            "http://localhost:5400",
            "http://localhost:5500"
        ],
        "extra_config": {
            "backend/http": {
                "return_error_details": "backend_alias"
            }
        }
    }
    ],
    "extra_config": {
        "auth/validator": {
            "alg": "RS256",
            "roles": ["Student"],
            "jwk_local_path": "./etc/public.json",
            "disable_jwk_security": true,
            "operation_debug": true
        }
    }
},
```

```
{
    "endpoint": "/instructor/dropped/{classid}/{sectionid}",
    "method": "GET",
    "output_encoding": "no-op",
    "backend": [
    {
        "url_pattern": "/dropped/{JWT.jti}/{classid}/{sectionid}/{JWT.name}/{JWT.sub}/{JWT.email}/{JWT.roles}",
        "method": "GET",
        "host": [
            "http://localhost:5300",
            "http://localhost:5400",
            "http://localhost:5500"
        ],
        "encoding": "no-op",
        "extra_config": {
            "backend/http": {
                "return_error_details": "backend_alias"
            }
        }
    }
    ],
    "extra_config": {
        "auth/validator": {
            "alg": "RS256",
            "roles": ["Instructor"],
            "jwk_local_path": "./etc/public.json",
            "disable_jwk_security": true,
            "operation_debug": true
        }
    }
},
```

Once we updated krakend.json, we tested to see if the load balancing worked as expected and it did. When we made multiple requests, it would switch between the different instances of the enroll service. Here is a screenshot of load balancing taking place:

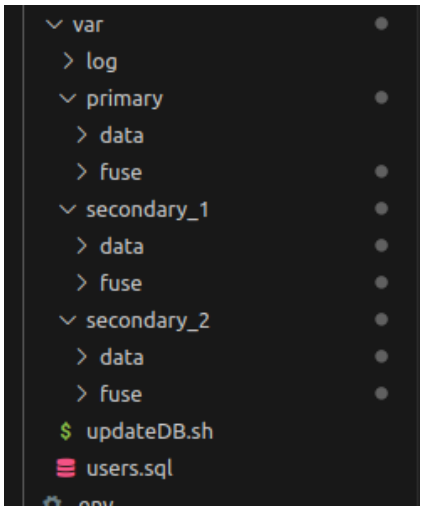
```
09:45:51 enroll_3.1    | INFO:     None:0 - "GET /list HTTP/1.1" 200 OK
09:45:51 krakend.1     | [GIN] 2023/10/28 - 09:45:51 | 200 |     2.33464ms |       127.0.0.1 | GET      "/student/list"

09:45:55 enroll_1.1    | INFO:     None:0 - "GET /list HTTP/1.1" 200 OK
09:45:55 krakend.1     | [GIN] 2023/10/28 - 09:45:55 | 200 |    2.792307ms |       127.0.0.1 | GET      "/student/list"

09:45:58 enroll_2.1    | INFO:     None:0 - "GET /list HTTP/1.1" 200 OK
09:45:58 krakend.1     | [GIN] 2023/10/28 - 09:45:58 | 200 |    2.408707ms |       127.0.0.1 | GET      "/student/list"
```

Add read replication to the users service

Created three LiteFS configuration files in `./etc` named `primary.yml`, `secondary_1.yml`, and secondary_2.yml.

Three directories in ./var

Primary, Secondary_1,Secondary_2.



Configuring Primary Replica:
Set the fuse.dir and data.dir parameters to define the directories for the primary replica

section for the primary replica's HTTP API Server configuration, execute the uvicorn command from the Procfile. Specify the port (20202) for the primary replica

```yaml
fuse:
  dir: users/var/primary/fuse
  allow-other: false

data:
  dir: users/var/primary/data
  compress: true

http:
  addr: ":20202"

lease:
  type: "static"
  candidate: true
  advertise-url: "http://localhost:20202"

exec: "uvicorn --port $PORT users.auth:app --reload"
```

Configuring Secondary_1 Replica:
Set the fuse.dir and data.dir parameters for the secondary replica to specify the directories:

In the secondary replica's HTTP API Server configuration, it listens on a different port (20203)

```yaml
fuse:
  dir: users/var/secondary_1/fuse
  allow-other: false

data:
  dir: users/var/secondary_1/data
  compress: true

http:
  addr: ":20203"

lease:
  type: "static"
  candidate: false
  advertise-url: "http://localhost:20202"
```

Configuring Secondary_2 Replica:
Set the fuse.dir and data.dir parameters for the secondary replica to specify the directories:

In the secondary replica's HTTP API Server configuration, it listens on a different port (20204)

```yaml
fuse:
  dir: users/var/secondary_2/fuse
  allow-other: false

data:
  dir: users/var/secondary_2/data
  compress: true

http:
  addr: ":20204"

lease:
  type: "static"
  candidate: false
  advertise-url: "http://localhost:20202"
```

Add a Process Type for Primary Replica:

Created a new process type for the primary replica.

Specified the command to run 'bin/litefs mount -config' with the configuration file for the primary replica.

Add a Process Type for Secondary_1 Replica:.

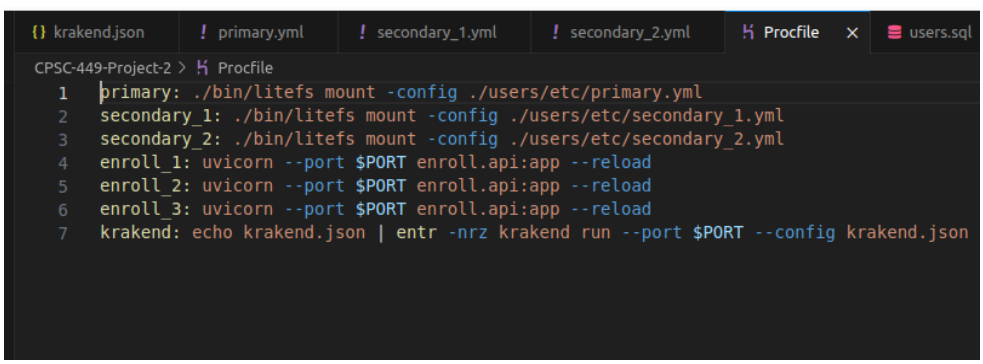
Create a new process type for the secondary_1 replica.

Specified the command to run 'bin/litefs mount -config' with the configuration file for the secondary replica.

Add a Process Type for Secondary_2 Replica:.

Create a new process type for the secondary_1 replica.

Specified the command to run 'bin/litefs mount -config' with the configuration file for the secondary replica.



After running, the foreman start. Database Replicas created in the Fuse directory of secondary_1 and Secondary_2.

Primary users.db

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

3|andy|Andy Jones|ajones@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
4|tim|Tim Raft|traft@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
5|elizabeth|Elizabeth Barnes|ebarnes@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
6|george|George Derns|gderns@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
7|pheobe|Pheobe Essek|pessek@fsmithcsu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
8|earl|Earl Poppins|epoppins@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
9|sarah|Sarah Colyt|fsmith@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
10|anna|Anna Kant|akant@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
11|micah|Micah Baumann|mbaumann@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
12|x|X|example@example.com|pbkdf2_sha256$600000$c724031167b08929914c21537ae14ab7$5b89KdqevI8OON+nvPuP/ZDm3wS9VmP1ZMuWxxpfdbQ=|
13|viditi21|viditi vartak|viditivartak08@gmail.com|pbkdf2_sha256$600000$44ee04de20c1a37c9e1c19f95ceebc3e$+/AIOyuWuUQ0+SBgBUgmQL+slykQskTcV/LI8ch2nfQ=|
14|Viditi01|Viditi Vartak|vartakviditi@gmail.com|pbkdf2_sha256$600000$9a229aee3594d85503c9eab8c55732af$0kDUlcUtOm6LlHmrMbc/uGFJGkL7Xulw8kYKqeKBdrI=|
sqite>
                                                                                                    Ln 56, C
```

Secondary_1 users.db

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

3|andy|Andy Jones|ajones@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
4|tim|Tim Raft|traft@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
5|elizabeth|Elizabeth Barnes|ebarnes@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
6|george|George Derns|gderns@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
7|pheobe|Pheobe Essek|pessek@fsmithcsu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
8|earl|Earl Poppins|epoppins@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
9|sarah|Sarah Colyt|fsmith@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
10|anna|Anna Kant|akant@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
11|micah|Micah Baumann|mbaumann@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
12|x|X|example@example.com|pbkdf2_sha256$600000$c724031167b08929914c21537ae14ab7$5b89KdqevI8OON+nvPuP/ZDm3wS9VmP1ZMuWxxpfdbQ=|
13|viditi21|viditi vartak|viditivartak08@gmail.com|pbkdf2_sha256$600000$44ee04de20c1a37c9e1c19f95ceebc3e$+/AIOyuWuUQ0+SBgBUgmQL+slykQskTcV/LI8ch2nfQ=|
14|Viditi01|Viditi Vartak|vartakviditi@gmail.com|pbkdf2_sha256$600000$9a229aee3594d85503c9eab8c55732af$0kDUlcUtOm6LlHmrMbc/uGFJGkL7Xulw8kYKqeKBdrI=|
sqite>
                                                                                                    Ln 56, C
```

Secondary_2 users.db

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

3|andy|Andy Jones|ajones@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
4|tim|Tim Raft|traft@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
5|elizabeth|Elizabeth Barnes|ebarnes@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
6|george|George Derns|gderns@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
7|pheobe|Pheobe Essek|pessek@fsmithcsu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
8|earl|Earl Poppins|epoppins@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
9|sarah|Sarah Colyt|fsmith@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
10|anna|Anna Kant|akant@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
11|micah|Micah Baumann|mbaumann@csu.fullerton.edu|pbkdf2_sha256$600000$c9fc625a0e406cec90594958016ac631$5iDVmwSTF6K9K110LWBxH/xi0ZvwpgKt3y8gAMz0GzQ=|
12|x|X|example@example.com|pbkdf2_sha256$600000$c724031167b08929914c21537ae14ab7$5b89KdqevI8OON+nvPuP/ZDm3wS9VmP1ZMuWxxpfdbQ=|
13|viditi21|viditi vartak|viditivartak08@gmail.com|pbkdf2_sha256$600000$44ee04de20c1a37c9e1c19f95ceebc3e$+/AIOyuWuUQ0+SBgBUgmQL+slykQskTcV/LI8ch2nfQ=|
14|Viditi01|Viditi Vartak|vartakviditi@gmail.com|pbkdf2_sha256$600000$9a229aee3594d85503c9eab8c55732af$0kDUlcUtOm6LlHmrMbc/uGFJGkL7Xulw8kYKqeKBdrI=|
sqite>
                                                                                                    Ln 56, C
```

Successfully reflected the changes in all database replicas.