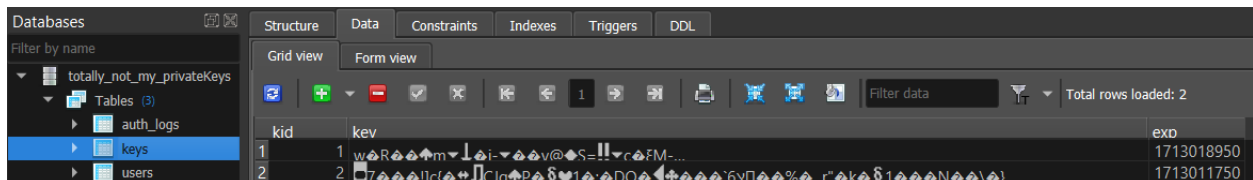


## Requirements and screenshots of results

### AES encryption

- Encrypt private keys using AES encryption.
- Use environment variable NOT\_MY\_KEY

```
2. const AESkey = process.env.NOT_MY_KEY;
3. const algorithm = 'aes-256-cbc';
4. const cipher = crypto.createCipheriv(algorithm, AESkey, iv);
5. const encryptedKey = cipher.update(keyPair.toPEM(true));
6. const expiredEncryptedKey = cipher.update(expiredKeyPair.toPEM(true));
```



	kid	key	exp
1	1	wRm...i-v@S=!!cFM-...	1713018950
2	2	7...ClnP...1...DO...6vP...r"61...N...}	1713011750

## User Registration

- Create users table
- Create POST:/register endpoint
- Create password using UUIDv4
- Hash password using Argon2
- Store details and hashed password in users table

```
7. app.post('/register', (req, res) => {
8.
9.   username = req.body.username;
10.  email = req.body.email;
11.  let uuidv4Password = uuid();
12.  argon2.hash(uuidv4Password).then(hash => {
13.    let sqlQuery = 'INSERT INTO users (username, password_hash, email)
    VALUES(:username, ";
14.    sqlQuery += hash;
15.    sqlQuery += ', :email)';
16.    db.run(sqlQuery, [username, email]);
17.
18.    res.setHeader('Content-Type', 'application/json');
19.    res.status(200).json({ password: uuidv4Password });
20.  })
21.})
```

The screenshot shows a database management interface with a sidebar on the left containing 'Databases', 'Tables (9)', 'auth\_logs', 'keys', and 'users'. The 'users' table is selected, and its structure is displayed in the main panel. The table has columns: id, username, password hash, email, date registered, and last login. A single row of data is shown.

id	username	password hash	email	date registered	last login
1	testor_dd8185cb	\$argon2id\$v=19\$m=65536,t=3,p=4\$2oNgv2ZU8XWlBw8Zg4u1Q\$pkudsRNA6G7ceoo+/bBdRbM+lc4tc/...	testor_dd8185cb@test.com	2024-04-13 13:35:56	NULL

The screenshot shows a REST client interface with a URL bar at the top set to 'http://localhost:8080/register'. The request method is 'POST' and the body is a JSON object: `{ "username": "testing", "email": "testing@email.com" }`. The response is shown in the bottom panel, indicating a '200 OK' status with a response time of 289 ms and a body size of 286 B. The response body is a JSON object: `{ "password": "8a9a29e3-fb33-47bd-9424-c2b8d581e4d3" }`.

Log Authentication Requests

- a. Create auth\_logs table
- b. Store details in auth\_logs table

Databases

Filter by name

totally\_not\_my\_privateKeys

Tables (3)

auth\_logs

keys

users

Views

Structure

Data

Constraints

Indexes

Triggers

DDL

Grid view

Form view

+

-

1

	id	request id	request timestamp	user id
1	1	::ffff:127.0.0.1	2024-04-13 13:35:56	1
2	2	::ffff:127.0.0.1	2024-04-13 13:35:57	1
3	3	::ffff:127.0.0.1	2024-04-13 13:35:58	1
4	4	::ffff:127.0.0.1	2024-04-13 13:35:58	1
5	5	::ffff:127.0.0.1	2024-04-13 13:35:58	1
6	6	::ffff:127.0.0.1	2024-04-13 13:35:58	1
7	7	::ffff:127.0.0.1	2024-04-13 13:35:58	1
8	8	::ffff:127.0.0.1	2024-04-13 13:35:58	1
9	9	::ffff:127.0.0.1	2024-04-13 13:35:58	1
10	10	::ffff:127.0.0.1	2024-04-13 13:35:58	1
11	11	::ffff:127.0.0.1	2024-04-13 13:35:58	1
12	12	::ffff:127.0.0.1	2024-04-13 13:35:58	1

GradeBot Results

```
PS C:\Users\tyler\Documents\UNT\CyberSecurity\project3\CSCE3550-main> go run "c:\Users\tyler\Docu
time=2024-04-13T08:23:04.977-05:00 level=ERROR msg="/register endpoint" err="sql: no rows in resu
time=2024-04-13T08:23:07.041-05:00 level=ERROR msg="/auth is rate-limited (optional)" err="expect
```

RUBRIC ITEM	ERROR?	POSSIBLE	AWARDED
Create users table		5	5
/register endpoint	sql: no rows in result set	20	5
Private Keys are encrypted in the database		25	25
Create auth_logs table		5	5
/auth requests are logged		10	10
/auth is rate-limited (optional)		25	0
	TOTAL	90	50