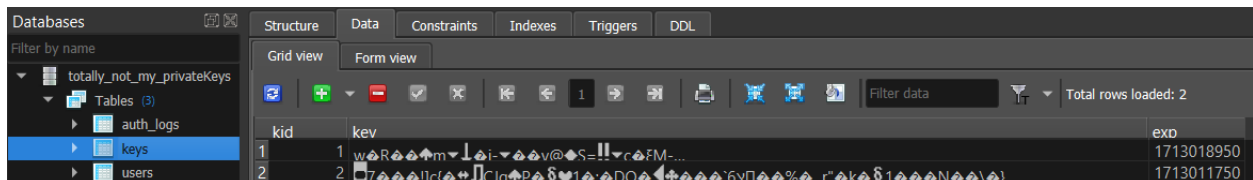


Requirements and screenshots of results

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));
```

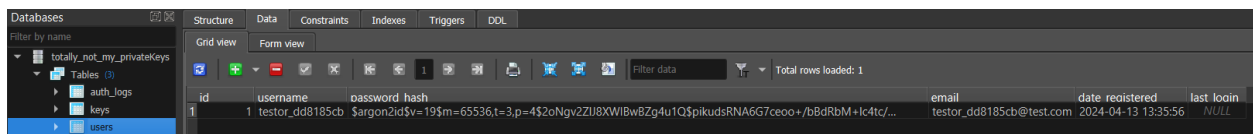


	kid	key	exp
1	1	wRm...i-v@S=!!cFM-...	1713018950
2	2	7...ClnP...1...DO...6vP...r"61...N...}	1713011750

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 a tree view of databases and tables. The main area displays the 'users' table in 'Grid view' mode. The table has five columns: 'id', 'username', 'password hash', 'email', and 'date registered'. A single row of data is visible, representing a user with ID 1, username 'testor_dd8185cb', a long Argon2 hash, email 'testor_dd8185cb@test.com', and a registration date of '2024-04-13 13:35:56'. The 'last login' column is empty.

id	username	password hash	email	date registered	last login
1	testor_dd8185cb	\$argon2id\$v=19\$m=65536,t=3,p=4\$2oNgvZUI8XWlBwBzG4u1Q\$pkudsRNA6G7ceoo+/bBdRbM+lc4tc/...	testor_dd8185cb@test.com	2024-04-13 13:35:56	NULL

Log Authentication Requests

- Create auth_logs table
- Store details in auth_logs table

The screenshot shows a database management interface. On the left, a tree view shows a database named 'totally_not_my_privateKeys' containing three tables: 'auth_logs', 'keys', and 'users'. The 'auth_logs' table is selected. On the right, the 'Structure' tab is active, showing the table's schema. The table has four columns: 'id', 'request id', 'request timestamp', and 'user id'. Below the schema, a 'Grid view' of the table data is displayed, showing 12 rows of data. Each row has an 'id' from 1 to 12, a 'request id' of '1', a 'request timestamp' of '2024-04-13 13:35:56' to '2024-04-13 13:35:58', and a 'user id' of '1'.

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

GradeBot Results

```
PS C:\Users\tyler\Documents\UNT\CyberSecurity\project3\CSCE3550-main> go run "c:\Users\tyler\Documents\UNT\CyberSecurity\project3\CSCE3550-main\main.go"
time=2024-04-13T08:23:04.977-05:00 level=ERROR msg="/register endpoint" err="sql: no rows in result set"
time=2024-04-13T08:23:07.041-05:00 level=ERROR msg="/auth is rate-limited (optional)" err="expected 200 status code, got 400"
```

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