

User Manual

Web Application:

Run the command

python3 hello.py

```
(venv) kathu@kathu-Aspire-E5-575G: ~/Desktop/project209$  
(venv) kathu@kathu-Aspire-E5-575G:~/Desktop/project209$ python3 hello.py  
* Serving Flask app "hello" (lazy loading)  
* Environment: production  
  WARNING: Do not use the development server in a production environment.  
  Use a production WSGI server instead.  
* Debug mode: on  
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 303-945-086
```

Centralized Authority

Run the command

python3 KDC.py

```
kathu@kathu-Aspire-E5-575G: ~/Desktop/project209  
kathu@kathu-Aspire-E5-575G:~/Desktop/project209$ python3 KDC.py  
KDC listening....
```

Database

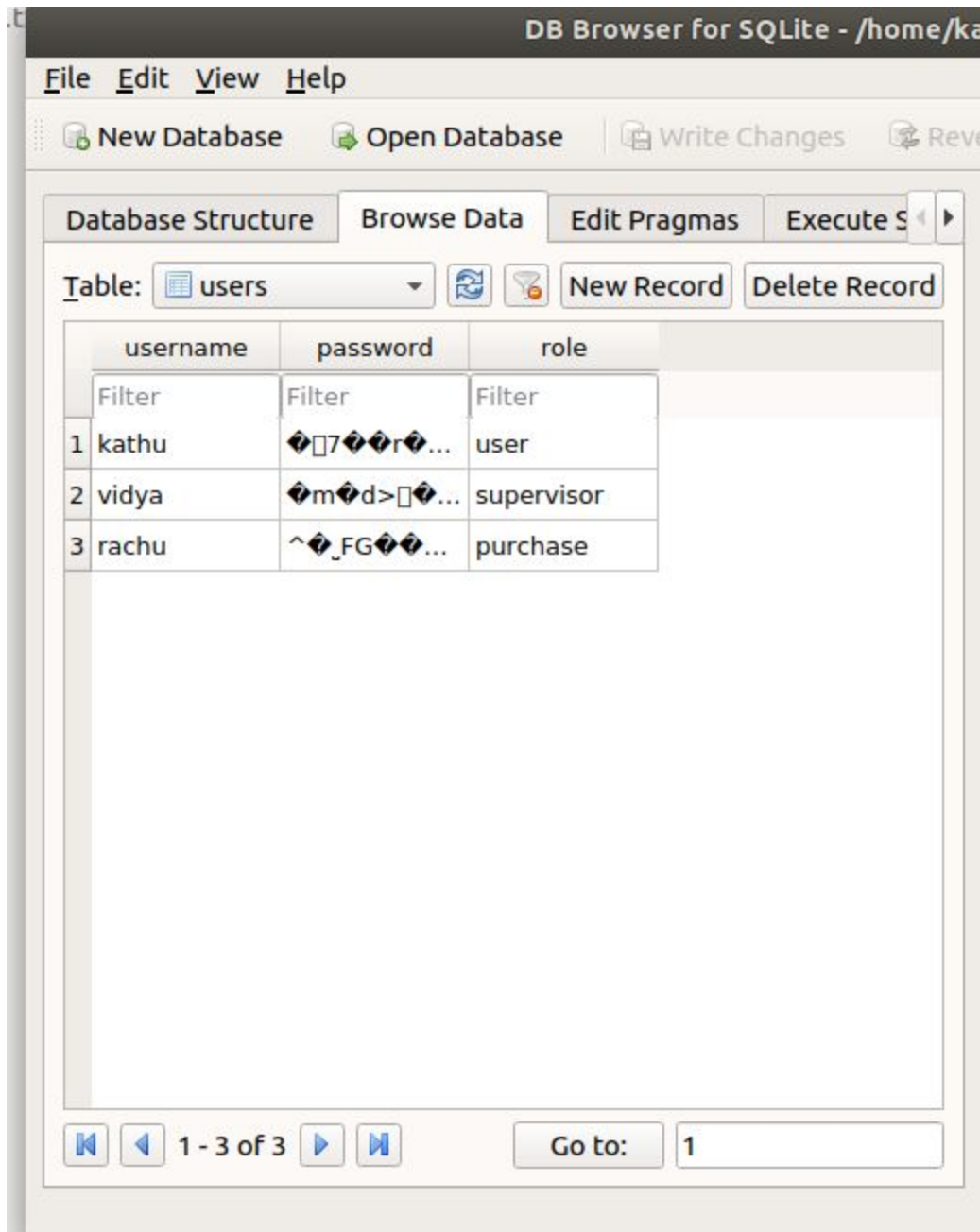
The sqlite3 browsers database.db and db1.db should be in same folder as hello.py and KDC.py

Role based login

The system has a role based log in. So the web pages appearing for each role will be different. The roles of each user can be verified using “users” table in database.db

Password of each entity

username: kathu	password: kathu123
username: vidya	password: vidya123
username: rachu	password: rachu123



The screenshot shows the 'DB Browser for SQLite' application window. The title bar indicates the file path is '/home/ka...'. The menu bar includes 'File', 'Edit', 'View', and 'Help'. The toolbar contains icons for 'New Database', 'Open Database', 'Write Changes', and 'Reveal'. Below the toolbar, there are tabs for 'Database Structure', 'Browse Data', 'Edit Pragmas', and 'Execute SQL'. The 'Browse Data' tab is active, showing a table named 'users'. Above the table, there are buttons for 'New Record' and 'Delete Record'. The table has three columns: 'username', 'password', and 'role'. The data is as follows:

	username	password	role
1	kathu	7r...	user
2	vidya	m d>...	supervisor
3	rachu	^_FG...	purchase

At the bottom of the window, there are navigation buttons (first, previous, next, last) and a status bar showing '1 - 3 of 3'. A 'Go to:' field with the value '1' is also present.

Tables in database.db

Table name: users

Provides username, hash of password and role

Table name: orders

Provides the order details and its status

Table name: supervisor

Stores the order number, encrypted order details send by user and signature of user

Table name: purchasedepartment

Stores the order numbers encrypted order details send by user and supervisor, and also their respective signatures

Tables in db1.db

Table name: purchasedepartment

Stores the role, user id , ip and public key path

Public Key and Private Key Storage

The public and private keys are stored as .pem files in the system. The Centralized Authority stores the file path of the keys in database db1.db. So while running the web application please change the public key path to the appropriate path inside the system in the table KCDDirectory

The screenshot shows the DB Browser for SQLite interface. The main window displays the 'KCDirectory' table with the following data:

ole	IP	ID	PubKeyFilePath
1	127.0.0.1:9...	vidya	/home/kathu/Desktop/proj...
2	127.0.0.1:8...	rachu	/home/kathu/Desktop/proj...
3	127.0.0.1:5...	kathu	/home/kathu/Desktop/proj...

The 'Edit Database Cell' panel on the right shows the 'Text' mode and a large empty text area. Below it, the 'SQL Log' panel displays the following SQL queries:

```
5 SELECT `_rowid_`,* FROM `KCDirectory` ORDER BY `_rowid_` ASC
6 SELECT COUNT(*) FROM (SELECT `_rowid_`,* FROM `KCDirectory`
7 SELECT `_rowid_`,* FROM `KCDirectory` ORDER BY `_rowid_` ASC
8 SELECT COUNT(*) FROM (SELECT `_rowid_`,* FROM `KCDirectory`
9 SELECT `_rowid_`,* FROM `KCDirectory` ORDER BY `_rowid_` ASC
10 SELECT COUNT(*) FROM (SELECT `_rowid_`,* FROM `KCDirectory`
11 SELECT `_rowid_`,* FROM `KCDirectory` ORDER BY `PubKeyFilePa
12
```

The bottom status bar indicates 'UTF-8' encoding.

HTML Page storage

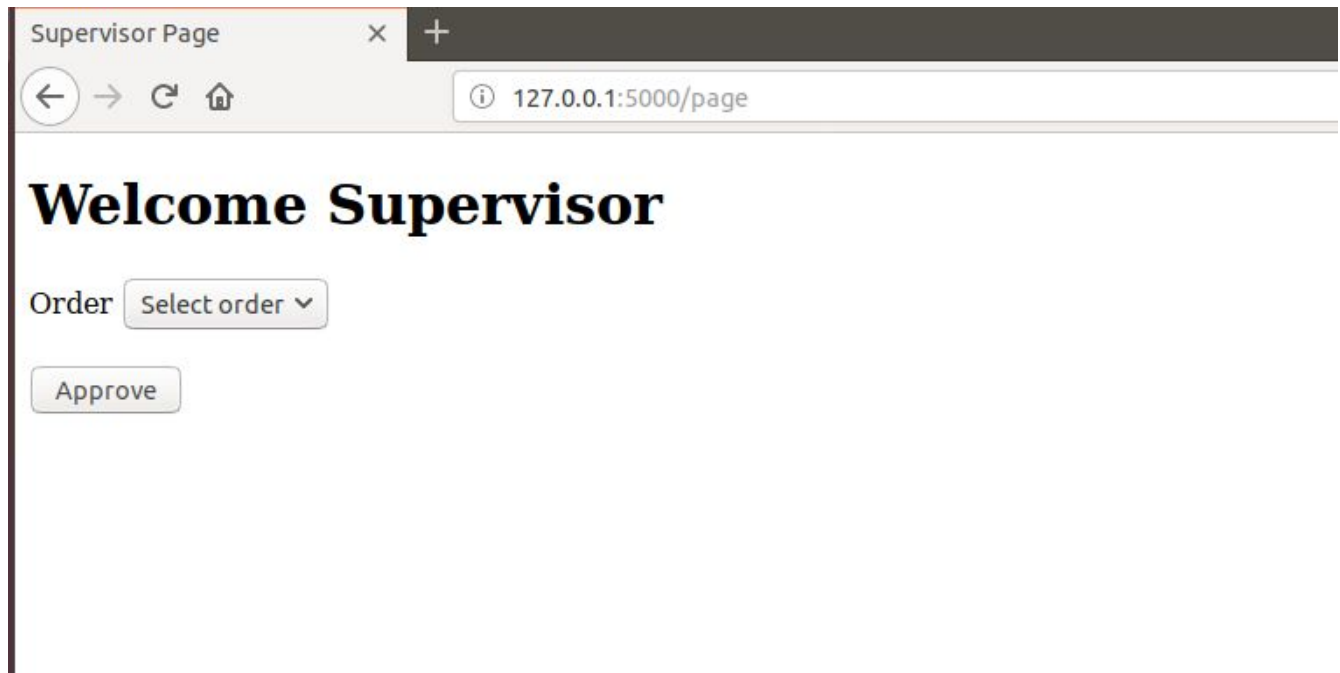
All html pages are stored in templates folder. So if you wish to make any edits or verify any page, that's the folder to look in to.

Web Page Logins

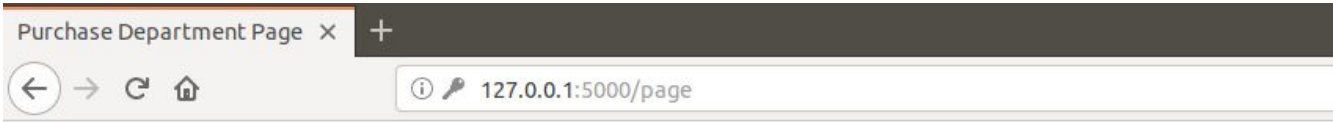
Role: user



Role: Supervisor



Role: Purchase Department



Welcome to Purchase Department

Order

Understanding the source code

hello.py

```
44 #Validation of the user
45 @app.route('/page', methods = ['POST', 'GET'])
46 def role():
47     if request.method == 'POST':
48         session['user'] = request.form['username']
49         pwd = request.form['password']
50         print(pwd)
51     #hashing the password and comparing it with the value in database
52     pwd = hashFile(pwd)
53     conn = connectionOpen()
54     cur = conn.cursor()
55     print(session['user'])
56     print(pwd)
57     #Rows will only be returned if the hash value of password matches the value in databases
58     cur.execute("SELECT * FROM users WHERE username=? AND password=?", (session['user'],pwd,))
59     rows = cur.fetchall()
60     if rows == []:
61         print("hash value of password not matching or username not valid")
62         abort(500, 'Invalid login credentials')
63     role = rows[0][2]
64     print(role)
65     #displaying web pages based on roles
66     if role == 'user':
67         return render_template('page.html',msg="Welcome "+ session['user'])
68     elif role == 'supervisor':
69         cur.execute("SELECT * FROM supervisor WHERE orderstatus=?", ('CREATED',))
70         rows = cur.fetchall()
71         return render_template('supervisor.html', rows = rows)
72     elif role == 'purchase':
73         cur.execute("SELECT * FROM orders WHERE status=?", ('PROCESSING',))
74         rows = cur.fetchall()
75         return render_template('purchase.html', rows=rows)
76     connectionClose(conn)
77
78 #Displaying Order Creation Page
```

Line 45: `@app.route` shows the html page extension and `def role()` is the function the application will execute when *page* is called.

generateRSAkeys.py

command: `python3 generateRSAkeys.py`

This file is generates 1024-bit RSA public and private keys and saves as .pem files

```
1 import rsa
2 (pubkey, pvtkey) = rsa.newkeys(1024)
3 #Saving public key and private key in PEM format
4 exppub =pubkey.save_pkcs1(format='PEM')
5 exppriv = pvtkey.save_pkcs1(format='PEM')
6 f1 = open("pubpurDept.txt", "wb")
7 f2 = open("prvpurDept.txt", "wb")
8 f1.write(exppub)
9 f2.write(exppriv)
10 f1.close()
11 f2.close()
```

hash.py

command `python3 hash.py`

On prompting please input username and also the new password

```
1 import hashlib
2 import sqlite3
3
4 hashObject = hashlib.sha3_224()
5 username = input("enter the username")
6 password = input("password")
7 conn = sqlite3.connect('database.db')
8 cur = conn.cursor()
9
10 for i in range (len(password)):
11     hashObject.update(password[i].encode("utf8"))
12     digest = hashObject.digest()
13     cur.execute("UPDATE users SET password=? WHERE username=?", (digest,username))
14     conn.commit()
15
16 conn.close()
17 |
```

Please view the demo video provided to know more about the operation.