

## Security features

### Role-Based Authorization:

The service uses role-based authorization to separate access control among different user roles (admin and user).

### Secure Communication:

Use SSL/TLS encryption, all communications between clients and the server are encrypted.

```
app.run(ssl_context='adhoc')
```

### Audit and Logging:

Used logging library to log all significant events like login attempts, registration, password changes, and unauthorized access attempts. The log is saved in the app.log file.

### Secure Session Management:

By using configurations such as SESSION\_COOKIE\_SECURE, SESSION\_COOKIE\_HTTPONLY, and SESSION\_COOKIE\_SAMESITE, ensuring data is secured. Also sessions will expire after 10 minutes.

### Password Management:

Using hashing algorithms provided by the Werkzeug library. This ensures passwords are stored securely in the database.

### Input Validation and Sanitization:

Input validation is enforced through WTForms. Usernames can only contain letters, numbers, and underscores. This helps to prevent XSS attacks. Passwords are at least 8 characters long, contain at least one letter, one number, and may contain special characters. This password

policy helps to ensure that users choose strong passwords.

```
3 class RegistrationForm(Form):
4     username = StringField('Username', [validators.Length(min=4, max=25), validators.DataRequired()])
5     password = PasswordField('Password', [
6         validators.DataRequired(),
7         validators.EqualTo('confirm', message='Passwords must match'),
8         validators.Length(min=8)
9     ])
10    confirm = PasswordField('Repeat Password')
11
12 class LoginForm(Form):
13     username = StringField('Username', [validators.Length(min=4, max=25), validators.DataRequired()])
14     password = PasswordField('Password', [validators.DataRequired()])
15
16 class ChangePasswordForm(Form):
17     old_password = PasswordField('Old Password', [validators.DataRequired()])
18     new_password = PasswordField('New Password', [
19         validators.DataRequired(),
20         validators.EqualTo('confirm', message='Passwords must match'),
21         validators.Length(min=8)
22     ])
23    confirm = PasswordField('Repeat Password')
```

## Security Headers:

Various HTTP security headers such as X-Content-Type-Options, X-Frame-Options, X-XSS-Protection, and Content-Security-Policy are set to mitigate risks.

```
50 # Security headers
51 @app.after_request
52 def apply_security_headers(response):
53     response.headers["X-Content-Type-Options"] = "nosniff" # prevents MIME type sniffing
54     response.headers["X-Frame-Options"] = "SAMEORIGIN" # prevents clickjacking
55     response.headers["X-XSS-Protection"] = "1; mode=block" # protect against XSS attacks
56     response.headers["Content-Security-Policy"] = "default-src 'self'" # restricts the sources from which content can be loaded
57     return response
```

## Database Security:

Using SQLAlchemy for database interactions, the service benefits from its defense against SQL injection

**We use postman to test our APIs, here are the tests we did to check functionality:**

**POST /register:**

POST

https://127.0.0.1:5000/register

Send

Params

Authorization

Headers (12)

Body

Pre-request Script

Tests

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

1

2

3

4

5

6

7

8

```
1  {
2    "username": "user3",
3    "password": "strongpassword123",
4    "confirm": "strongpassword123"
5  }
6
7
8
```

Cookies (1)

Headers (9)

Test Results

Pretty

Raw

Preview

Visualize

HTML

785 / register

POST

https://127.0.0.1:5000/register

Send

Params

Authorization

Headers (12)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```

1  {}
2  {
3    "username": "user4",
4    "password": "weakpassword",
5    "confirm": "weakpassword"
6  }
7  // {
8  //   "username": "user4",
9  //   "password": "strongpassword123",
10 //   "confirm": "strongpassword123"
11 // }
12
13
14

```

Body

Cookies (1)

Headers (9)

Test Results

Pretty

Raw

Preview

Visualize

JSON

400 BAD REQUEST

Time: 12 ms

Size: 459 B

Save as example

```

1  {
2    "error": {
3      "password": [
4        "Password must be at least 8 characters, contain at least one letter, one number, and may contain special characters."
5      ]
6    }
7  }

```

## POST /login

785 / login

Save

Send

POST

https://127.0.0.1:5000/login

Params

Authorization

Headers (11)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

6

7

8

9

```

1  {
2    "username": "user3",
3    "password": "strongpassword123"
4  }
5
6  // {
7  //   "username": "admin",
8  //   "password": "admin123"
9  // }

```

Body

Cookies (1)

Headers (11)

Test Results

Status: 200 OK

Time: 160 ms

Size: 626 B

Save as example

| Name    | Value           | Value     | Path | Expires | HttpOnly | Secure |
|---------|-----------------|-----------|------|---------|----------|--------|
| session | .eJwIzkGuAJE... | 127.0.0.1 | /    | Session | true     | true   |

785 / login

POST https://127.0.0.1:5000/login

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "username": "user3",
3   "password": "strongpassword12"
4 }
5
6 // {
7 //   "username": "admin",
8 //   "password": "admin123"
9 // }
```

Status: 401 UNAUTHORIZED Time: 160 ms Size: 459 B Save as example

No cookies received from the server

All your cookies and their associated domains will appear here.

## POST /changepw

785 / changepw

POST https://127.0.0.1:5000/changepw

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "old_password": "strongpassword123",
3   "new_password": "strongpassword1234",
4   "confirm": "strongpassword1234"
5 }
6
7 // {
8 //   "old_password": "wrongpassword",
9 //   "new_password": "strongpassword123",
10 //   "confirm": "strongpassword123"
11 // }
12
13
14
```

Status: 201 CREATED Time: 245 ms Size: 331 B Save as example

Pretty Raw Preview Visualize HTML

1

785 / changepw

POST https://127.0.0.1:5000/changepw

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 // {
2 //   "old_password": "strongpassword123",
3 //   "new_password": "strongpassword1234",
4 //   "confirm": "strongpassword1234"
5 // }
6
7
8 // {
9 //   "old_password": "wrongpassword",
10 //   "new_password": "strongpassword123",
11 //   "confirm": "strongpassword123"
12 // }
13
14
```

Body Cookies (1) Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "error": "Incorrect old password"
3 }
```

Status: 400 BAD REQUEST Time: 247 ms Size: 363 B Save as example

## GET /admin

### 1. Success login

785 / login

POST https://127.0.0.1:5000/login

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 // {
2 //   "username": "user3",
3 //   "password": "strongpassword123"
4 // }
5
6
7 // {
8 //   "username": "admin",
9 //   "password": "admin123"
10 // }
11
```

Body Cookies (1) Headers (11) Test Results

Status: 200 OK Time: 132 ms Size: 627 B Save as example

| Name    | Value          | Value     | Path | Expires | HttpOnly | Secure |
|---------|----------------|-----------|------|---------|----------|--------|
| session | aJwIzkGuAJE... | 127.0.0.1 | /    | Session | true     | true   |

### 2. Logged in as user

785 / admin

Save

GEThttps://127.0.0.1:5000/adminSend

ParamsAuthorizationHeaders (11)Body●Pre-request ScriptTestsSettingsCookies

Query Params

| Key | Value | Description | Bulk Edit |
|-----|-------|-------------|-----------|
| Key | Value | Description |           |

BodyCookies (1)Headers (10)Test Results

Status: 403 FORBIDDENTime: 9 msSize: 351 BSave as example

PrettyRawPreviewVisualizeJSON

```
1  {
2    "error": "Unauthorized"
3  }
```

### 3. Failed login

785 / admin

Save

GEThttps://127.0.0.1:5000/adminSend

ParamsAuthorizationHeaders (10)Body●Pre-request ScriptTestsSettingsCookies

Query Params

| Key | Value | Description | Bulk Edit |
|-----|-------|-------------|-----------|
| Key | Value | Description |           |

BodyCookiesHeaders (9)Test Results

Status: 401 UNAUTHORIZEDTime: 4 msSize: 641 BSave as example

PrettyRawPreviewVisualizeHTML

```
1  <!doctype html>
2  <html lang=en>
3  <title>401 Unauthorized</title>
4  <h1>Unauthorized</h1>
5  <p>The server could not verify that you are authorized to access the URL requested. You either supplied the wrong
6  | credentials (e.g. a bad password), or your browser doesn't understand how to supply the credentials required.
7  </p>
```

### GET /user

## 1. Success login

785 / user

GET https://127.0.0.1:5000/user Send

Params Authorization Headers (11) Body • Pre-request Script Tests Settings Cookies

Query Params

| Key | Value | Description | Bulk Edit |
|-----|-------|-------------|-----------|
| Key | Value | Description |           |

Body Cookies (1) Headers (10) Test Results

Status: 200 OK Time: 7 ms Size: 350 B Save as example

Pretty Raw Preview Visualize HTML

1 Logged in as user user3

## 2. Logged in as admin

785 / user

GET https://127.0.0.1:5000/user Send

Params Authorization Headers (11) Body • Pre-request Script Tests Settings Cookies

Query Params

| Key | Value | Description | Bulk Edit |
|-----|-------|-------------|-----------|
| Key | Value | Description |           |

Body Cookies (1) Headers (10) Test Results

Status: 403 FORBIDDEN Time: 5 ms Size: 351 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "error": "Unauthorized"
3 }
```

## 3. Login failure

GEThttps://127.0.0.1:5000/user

Send

ParamsAuthorizationHeaders (10)Body●Pre-request ScriptTestsSettings

Cookies

| Query Params |       |             |             |
|--------------|-------|-------------|-------------|
| Key          | Value | Description | ⋮ Bulk Edit |
| Key          | Value | Description |             |

BodyCookiesHeaders (9)Test Results

Status: 401 UNAUTHORIZEDTime: 4 msSize: 641 BSave as example

PrettyRawPreviewVisualizeHTML

```
1 <!doctype html>
2 <html lang=en>
3 <title>401 Unauthorized</title>
4 <h1>Unauthorized</h1>
5 <p>The server could not verify that you are authorized to access the URL requested. You either supplied the wrong
6 | credentials (e.g. a bad password), or your browser doesn't understand how to supply the credentials required.
7 </p>
```