

LAPORAN UAS PEMROGRAMAN SISI SERVER

NAMA : UUT PRASETIA

NIM : A11.2019.11867

1. Tampilan Swagger API

The screenshot shows the Swagger UI interface for the Simple LMS API. At the top, it displays the title "Simple LMS API" with version 1.0.0 and OAS 3.1. Below the title, there's a search bar and an "Authorize" button. The main area is titled "default" and lists several API endpoints:

- POST /api/lms/register** Register
- POST /api/lms/login** Login
- GET /api/lms/users** List Users
- GET /api/lms/courses** List Courses
- POST /api/lms/courses** Create Course
- DELETE /api/lms/courses/{course_id}** Delete Course
- GET /api/lms/test-session** Test Session
- GET /api/lms/lessons** List Lessons

Penjelasan:

Swagger UI digunakan sebagai dokumentasi dan media pengujian API. Seluruh endpoint backend dapat diuji langsung melalui browser tanpa frontend tambahan.

2. Register User

The screenshot shows the Swagger UI for the "/api/lms/register" endpoint. It includes sections for "Parameters" (No parameters), "Request body" (Required, application/json), and "Responses".

Request body (application/json):

```
{
  "username": "string",
  "password": "string",
  "email": "string",
  "role": "mahasiswa"
}
```

Responses:

Code	Description	Links
200	OK	No links

Example Value | Schema:

```
{
  "id": 0,
  "username": "string",
  "email": "string",
  "role": "string"
}
```

Penjelasan:

Pada tahap ini dilakukan registrasi user dengan role tertentu (admin/mahasiswa). Jika berhasil, sistem akan mengembalikan data user yang telah dibuat. Hal ini menunjukkan fitur registrasi berjalan dengan baik.

3. Login dan JWT Token

localhost:8000/api/docs#/default/lms_api_login

username * required
string
(query)
admin1

password * required
string
(query)
admin123

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8000/api/lms/login?username=admin1&password=admin123' \
  -H 'accept: */*' \
  -d ''
```

Request URL

http://localhost:8000/api/lms/login?username=admin1&password=admin123

Server response

Code	Details
200	<p>Response body</p> <pre>{ "access_token": "eyJhbGciOiJIUzI1NiIsInR5cGkiOiJpZC9jey3iYy21kTjox1C1c2Vylemt2STGImfkbluMS1sIn3vbg9jD1hZGlph1ts1mhdC19Ht2Nzg3OTE5Niw1ZX0wTjoxKzY300gyNzk2FQ.G-1TGPPVn5x-uuzEhPMIOheT1", "role": "admin" }</pre> <p>Download</p> <p>Response headers</p> <pre>access-control-allow-origin: * content-length: 252 content-type: application/json; charset=utf-8 cross-origin-opener-policy: same-origin date: Thu, 08 Jan 2026 13:53:16 GMT</pre>

Penjelasan:

User melakukan login menggunakan username dan password. Sistem mengembalikan JWT access token yang digunakan untuk mengakses endpoint yang dilindungi.

4. Authorize Menggunakan JWT

localhost:8000/api/docs#/default/lms_api_login

Simple LMS API (API DOK)

POST /api/lms/register Register

POST /api/lms/login Login

Parameters

Name	Description
username * required	string (query) admin1
password * required	string (query) admin123

Available authorizations

JWTAuth (http, Bearer)

Authorized

Value: *****

Logout Close

Penjelasan:

JWT token dimasukkan ke Swagger melalui fitur Authorize dengan format `Bearer <token>`. Setelah berhasil, endpoint yang membutuhkan autentikasi dapat diakses.

5. Pengujian RBAC (Admin)

```

curl -X 'GET' \
  'http://localhost:8000/api/lms/users' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cIjoiK2lyZmFtZTsiLCJyZWYiOiJkZW1lM5tsInJvbGUiIjoxLC1c2VybmtZG1pbW1sInlhdiC16MTC2Nzg3OTE5Niw2Xnw1oxNzY300gyNzk2fQ.G-1TGPPWn5x-UmzEHFRIOkNeT111gyc'

```

Request URL:
<http://localhost:8000/api/lms/users>

Server response

Code	Details
200	<p>Response body</p> <pre>[{ "id": 1, "username": "admin", "email": "admin@lms.com", "role": "admin" }, { "id": 2, "username": "dosen1", "email": "dosen1@lms.com", "role": "dosen" }, { "id": 3, "username": "admin", "email": "admin@lms.com", "role": "admin" }, { "id": 4, "username": "string", "email": "string", "role": "mahasiswa" }]</pre>

[Copy](#) [Download](#)

Penjelasan:

Endpoint ini hanya dapat diakses oleh user dengan role admin. Keberhasilan akses menunjukkan bahwa Role Based Access Control (RBAC) telah diterapkan dengan benar.

6. Pengujian RBAC (Mahasiswa – Akses Ditolak)

No parameters

Execute Clear

Responses

Curl

```

curl -X 'GET' \
  'http://localhost:8000/api/lms/users' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cIjoiK2lyZmFtZTsiLCJyZWYiOiJkZW1lM5tsInJvbGUiIjoxLC1c2VybmtZG1pbW1sInlhdiC16MTC2Nzg3OTE5Niw2Xnw1oxNzY300gyNzk2fQ.G-1TGPPWn5x-UmzEHFRIOkNeT111gyc'

```

Request URL:
<http://localhost:8000/api/lms/users>

Server response

Code	Details
403	<p>Undocumented Error: Forbidden</p> <p>Response body</p> <pre>{ "detail": "Forbidden: insufficient role" }</pre>

Response headers

```

content-length: 42
content-type: application/json; charset=utf-8
cross-origin-opener-policy: same-origin
date: Thu, 01 Jun 2023 10:48 GMT
referrer-policy: origin-when-cross-origin
server: WSGIServer/0.2 Python/3.12.12
vary: origin
x-content-type-options: nosniff
x-frame-options: DENY

```

Penjelasan:

Mahasiswa tidak memiliki hak akses ke endpoint admin. Respon forbidden membuktikan pembatasan akses berdasarkan role berjalan sesuai kebutuhan.

7. Pengujian Redis Session

```

curl -X 'GET' \
  'http://localhost:8000/api/lms/test-session' \
  -H 'Accept: */*' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInB1c2giOiJzIy5cCf61kpXCI9.eyJ1c2VyX2lkIjoiLCJ1c2VydFt2S16InIoc191YXN0LCTyb2x1TjoibmF0YXNpZ3BhYiwiak#07jonNzY3O0g#FTsd.C11eIAiOJE3Nj-400K3MTB9-1F30C3A056G2iJN407xdet37KCn'

```

Request URL
<https://localhost:8000/api/lms/test-session>

Server response

Code	Details						
200	<p>Response body</p> <pre>{ "message": "session stored in Redis (via cache backend).", "key": "2nc98lsfuhjd2uj7yk1hx5z2y28u8ge1", "count": 4, "user": "ahs_uas", "role": "mahasiswa" }</pre> <p>Response headers</p> <pre> content-length: 162 content-type: application/json; charset=utf-8 cross-origin: true date: Thu, 08 Jan 2026 13:52:29 GMT referrer-policy: same-origin server: WSGI/1.0 Python/3.12.12 vary: cookie,origin x-content-type-options: nosniff x-frame-options: DENY </pre>						
Responses	<table border="1"> <thead> <tr> <th>Code</th> <th>Description</th> <th>Links</th> </tr> </thead> <tbody> <tr> <td>200</td> <td>OK</td> <td>No links</td> </tr> </tbody> </table> <p>Example Value "string"</p>	Code	Description	Links	200	OK	No links
Code	Description	Links					
200	OK	No links					

Penjelasan:

Endpoint ini digunakan untuk membuktikan bahwa session tersimpan di Redis. Nilai counter yang meningkat menandakan session bersifat persisten.

8. Verifikasi Session di Redis (Terminal)

```

1) "1:djangosession.cache2nc98lsfuhjd2uj7yk1hx5z2y28u8ge1"
27.0.0.1:6379[1]> KEYS "django"
1) "1:djangosession.cache2nc98lsfuhjd2uj7yk1hx5z2y28u8ge1"
27.0.0.1:6379[1]> INFO keyspace
# Keyspace
db1:keys=1,expires=1,avg_ttl=1209524764,subexpiry=0
27.0.0.1:6379[1]> SELECT 1
OK
27.0.0.1:6379[1]> KEYS "*"
1) "1:djangosession.cache2nc98lsfuhjd2uj7yk1hx5z2y28u8ge1"
27.0.0.1:6379[1]>

```

Ln 153, Col 1 Spaces: 2 UTF-8 CRLF

Penjelasan:

Pengujian session dilakukan dengan mengakses Redis melalui container menggunakan perintah docker compose exec redis redis-cli. Setelah menjalankan endpoint test-session, dilakukan pengecekan menggunakan perintah INFO keyspace yang menunjukkan adanya key pada database Redis. Selanjutnya dilakukan SELECT 1 karena konfigurasi Django menggunakan database Redis index 1. Hasil perintah KEYS "*" menampilkan key bertipe django.contrib.sessions, yang membuktikan bahwa session pengguna berhasil disimpan di Redis sesuai dengan konfigurasi session backend.

9. Docker Compose Berjalan : docker compose ps

```

PS D:\simple_lms_docker> docker compose ps
>>
      NAME            IMAGE           COMMAND          SERVICE    CREATED         STATUS        PORTS
simple_lms_redis  redis:7       "docker-entrypoint.s..."  redis      7 seconds ago  Up 6 seconds  0.0.0.0:6379->6379/tcp, [::]:6379
simple_lms_web    simple_lms_web  "/entrypoint.sh bash..."  web        6 seconds ago  Up 5 seconds  0.0.0.0:8000->8000/tcp, [::]:80

```

Penjelasan:

Container web dan redis dalam kondisi Up, yang menunjukkan aplikasi berjalan menggunakan Docker Compose sesuai ketentuan.

Kesimpulan

Berdasarkan hasil pengujian dan screenshot yang ditampilkan, dapat disimpulkan bahwa aplikasi Simple LMS API telah berjalan dengan baik dan memenuhi seluruh kriteria UAS, yaitu menggunakan JWT Authentication, Role Based Access Control, Redis sebagai session

backend, serta dijalankan menggunakan Docker Compose. Seluruh fitur telah diuji melalui Swagger UI dan Redis, sehingga project ini dinyatakan siap untuk dikumpulkan.