SOFTWARE ENGINEERING

MILESTONE 5

Introduction

A test-driven development approach was used to transform the components and APIs identified for the ticket management application into actual code. The following section describes subset of the test cases formulated for implementing and testing functionality of the application. The code for the application and the unit tests are available in this like.

Endpoints setup

The application currently includes the following endpoints and functions:

Home Endpoint

Route: '/' with HTTP method GET

Function name: home()

Description: This endpoint returns a string "You have found this API. Badhai ho" as a response. It is accessed with a GET request to the root URL of the application.

User register Endpoint

Route: '/register' with HTTP method POST

Function name: register()

Description: This endpoint allows users to register by sending a JSON payload in the request body. The payload should contain name, username, email, password, and admin fields. The function hashes the password and creates a new User object in the database with the provided data. It returns a JSON response with a success key set to True upon successful registration.

User login Endpoint

Route: '/login' with HTTP method POST

Function name: login()

Description: This endpoint allows users to log in by sending HTTP basic authentication credentials in the request header. The function calls an authenticate_user() function to authenticate the user and returns the authentication data as a JSON response.

Ticket creation Endpoint

Route: '/ticket' with HTTP method POST

Function name: create_ticket(current_user)

Description: This endpoint allows authenticated users with a token to create a new ticket by sending a JSON payload in the request body. The payload should contain title, content, date, and user_id fields. The function creates a new Ticket object in the database with the provided data and returns a JSON response with a success key set to True upon successful ticket creation. This endpoint also has a token_required decorator which requires the user to be authenticated with a valid token.

FAQ Endpoint

Route: '/faq' with HTTP method GET

Function name: all_faqs()

Description: This endpoint returns all FAQs from the database as a JSON response. It retrieves all Faq objects from the database, serializes them, and returns them in a JSON response with a faqs key containing the serialized FAQs. It is accessed with a GET request to the /faq URL of the application.

Fixture Setup

The following fixtures were used to create a test environment and initialize a fixture indicative of a temporary database to perform, among other things, perform CRUD operations on tickets, ensuring the authorized user access the required endpoints, etc.

```
@pytest.fixture()
def app():
    app = Flask(__name__)
    app.config.update(
        SECRET_KEY=SECRET_KEY,
        SQLALCHEMY_DATABASE_URI=TEST_SQLALCHEMY_DATABASE_URI
    )
    app.register_blueprint(appc)
    app.app_context().push()
    db.init_app(app)
    Ticket.query.delete()
    db.session.commit()
    return app


@pytest.fixture()
def client(app):
    return app.test_client()
```

Test Cases

Test Case 1:

Page being tested: '/ticket'

Inputs: POST request without authentication headers, with JSON data including "date", "title", and "content"

Expected Output: The response should typically return a HTTP Status Code: 401 (Unauthorized) with a message stating "Unauthorized request". Further, since a request to create ticket is made by user without the authentication header/token, the database will not be updated, thereby returning 0 as Number of tickets in the database.

Actual Output:

HTTP Status Code: 401 (Unauthorized)

Response Data: "Unauthorized request" in the response data

Number of tickets in the database: 0

Result: Success

```
(se) D:\Projects\IITM\SE\proj\source>pytest --verbose -k test_create_ticket_admin tests.py
========================================= test session starts =========================================
platform win32 -- Python 3.10.5, pytest-7.2.2, pluggy-0.13.1 -- D:\Projects\IITM\SE\proj\source\
se\Scripts\python.exe
cachedir: .pytest_cache
rootdir: D:\Projects\IITM\SE\proj\source
collected 5 items / 4 deselected / 1 selected

tests.py::test_create_ticket_admin PASSED                                                [100%]

========================================== warnings summary ===========================================
database.py:5
  D:\Projects\IITM\SE\proj\source\database.py:5: MovedIn20Warning: The ``declarative_base()`` fu
nction is now available as sqlalchemy.orm.declarative_base(). (deprecated since: 2.0) (Backgroun
d on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
    Base = declarative_base()

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
======================== 1 passed, 4 deselected, 1 warning in 0.62s ========================
```

Test Case 2:

Page being tested: '/ticket'

Inputs: POST request with authentication headers for an admin user, with JSON data including "date", "title", and "content"

Expected Output: The response should typically return a HTTP Status Code: 403 (Forbidden) with a message stating "Forbidden". Further, since a request to create ticket is made by an unauthorized user, the database will not be updated, thereby returning 0 as Number of tickets in the database.

Actual Output:

HTTP Status Code: 403 (Forbidden)

Response Data: "Forbidden" in the response data

Number of tickets in the database: 0

Result: Success

```
(se) D:\Projects\IITM\SE\proj\source>pytest --verbose -k test_create_ticket_admin tests.py
==================================== test session starts ====================================
platform win32 -- Python 3.10.5, pytest-7.2.2, pluggy-0.13.1 -- D:\Projects\IITM\SE\proj\source\
se\Scripts\python.exe
cachedir: .pytest_cache
rootdir: D:\Projects\IITM\SE\proj\source
collected 5 items / 4 deselected / 1 selected

tests.py::test_create_ticket_admin PASSED                                            [100%]

==================================== warnings summary ====================================
database.py:5
  D:\Projects\IITM\SE\proj\source\database.py:5: MovedIn20Warning: The ``declarative_base()`` fu
nction is now available as sqlalchemy.orm.declarative_base(). (deprecated since: 2.0) (Backgroun
d on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
    Base = declarative_base()

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
============================ 1 passed, 4 deselected, 1 warning in 0.69s =============================
```

Test Case 3:

Page being tested: '/ticket'

Inputs: POST request with authentication headers for a student user, with JSON data including "date", "title", and "content"

Expected Output: The response returns an HTTP Status Code: 200 (OK). Further, since a request to create ticket is made by an authorized user, the database will be updated, thereby returning 1 as Number of tickets in the database. Further, details of the ticket would be returned as follows:

Ticket details in the database:

date: "123123123"

title: "test test1"

content: "test_create_ticket_without_auth"

status: "Open"

likes: 0

Actual Output:

HTTP Status Code: 200 (OK)

Number of tickets in the database: 1

Ticket details in the database:

date: "123123123"

title: "test test1"

content: "test_create_ticket_without_auth"

status: "Open"

likes: 0

Result: Success

```
(se) D:\Projects\IITM\SE\proj\source>pytest --verbose -k test_create_ticket_student_with_complet
ejson tests.py
================================== test session starts ==================================
platform win32 -- Python 3.10.5, pytest-7.2.2, pluggy-0.13.1 -- D:\Projects\IITM\SE\proj\source\
se\Scripts\python.exe
cachedir: .pytest_cache
rootdir: D:\Projects\IITM\SE\proj\source
collected 5 items / 4 deselected / 1 selected

tests.py::test_create_ticket_student_with_completejson PASSED                          [100%]

==================================== warnings summary ====================================
database.py:5
  D:\Projects\IITM\SE\proj\source\database.py:5: MovedIn20Warning: The ``declarative_base()`` fu
nction is now available as sqlalchemy.orm.declarative_base(). (deprecated since: 2.0) (Backgroun
d on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
    Base = declarative_base()

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
======================== 1 passed, 4 deselected, 1 warning in 0.61s ========================
```

Test Case 4:

Page being tested: '/ticket'

Inputs: POST request with authentication headers for a student user, with JSON data including "date", "title", and "content", and extra data including "likes" and "status"

Expected Output: The response returns an HTTP Status Code: 200 (OK). Further, since a request to create ticket is made by an authorized user, the database will be updated, thereby returning 1 as Number of tickets in the database. Further, details of the ticket would be returned as follows:

Ticket details in the database:

date: "123123123"

title: "test test1"

content: "test_create_ticket_without_auth"

status: "Open"

likes: 0 (likes field should not be updated as per the input)

Actual Output:

HTTP Status Code: 200 (OK)

Number of tickets in the database: 1

Ticket details in the database:

date: "123123123"

title: "test test1"

content: "test_create_ticket_without_auth"

status: "Open"

likes: 0 (likes field should not be updated as per the input)

Result: Success

```
(se) D:\Projects\IITM\SE\proj\source>pytest --verbose -k test_create_ticket_student_extrajson te
sts.py
==================================== test session starts =====================================
platform win32 -- Python 3.10.5, pytest-7.2.2, pluggy-0.13.1 -- D:\Projects\IITM\SE\proj\source\
se\Scripts\python.exe
cachedir: .pytest_cache
rootdir: D:\Projects\IITM\SE\proj\source
collected 5 items / 4 deselected / 1 selected

tests.py::test_create_ticket_student_extrajson PASSED                                [100%]

====================================== warnings summary ======================================
database.py:5
  D:\Projects\IITM\SE\proj\source\database.py:5: MovedIn20Warning: The ``declarative_base()`` fu
nction is now available as sqlalchemy.orm.declarative_base(). (deprecated since: 2.0) (Backgroun
d on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
    Base = declarative_base()

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
=========================== 1 passed, 4 deselected, 1 warning in 0.68s ========================
```

Test Case 5:

Page being tested: '/ticket'

Inputs: POST request with authentication headers for a student user, with incomplete JSON data missing required fields

Expected Output: The response should typically return a HTTP Status Code: 400 with a message stating "Bad Request". Further, since a request to create ticket is without the required information, the database will not be updated, thereby returning 0 as Number of tickets in the database.

Actual Output:

Response status code should be 400 (Bad Request)

Response data should contain the message "Bad Request"

Number of tickets in the database should be 0

Result: Success