Unit Test Report

team-project-25spring-34

May 2025

Contents

1 Introduction

This report summarizes the results of unit testing for several key modules of the project **team-project-25spring-34**. The testing process was conducted using Django's built-in test framework via `python manage.py test <app>`. Each section below corresponds to an individual Django app and details its testing results.

2 Test Summary

The following Django apps were tested:

- **ai_assistant**
- **IDE**
- **login**
- **self_learn**
- **register**

All tests were executed on a Windows system (Powershell terminal), and all tests passed successfully (**OK**). No issues were identified during the system checks.

3 Test Design and Implementation

3.1 Overview

To ensure the robustness and reliability of our Django backend modules, we implemented a comprehensive suite of unit and integration tests leveraging Django's `TestCase` framework. The primary objectives were to verify core business logic, file upload and deletion, user authentication, registration, PDF processing, and bookmark management features.

3.2 Tested Features

- **DeepSeek API Integration:**
  - Tests covered both simple text prompts and PDF file uploads, including cases for mind map and test question generation.
  - Error-handling and invalid method responses were checked to ensure graceful failure on unsupported HTTP methods or improper input.
  - Third-party dependent logic (`html_to_png`) was mocked where necessary to isolate the backend logic.
- **User Authentication:**
  - Login functionality was validated for both correct and incorrect credentials, checking for proper redirection or user feedback.
  - The login form rendering through GET requests was also tested to ensure frontend compatibility.

- **User Registration:**
  - Registration logic included cases for successful registration, duplicate usernames, password mismatch, and password policy enforcement.
  - The tests confirmed that only valid users could be created and appropriate errors were returned for invalid submissions.
- **PDF Upload and Deletion:**
  - Tests ensured that PDF files could be successfully uploaded and deleted, and validated server-side responses for both actions.
- **Bookmark Management:**
  - Bookmarks could be added, queried, and deleted per user and per course, with the tests verifying data isolation and correctness.
  - Edge cases, such as invalid or empty bookmark data, were also tested to guarantee the API's input validation mechanisms.

3.3 Test Methods

- All tests utilized Django's `Client` for simulating HTTP requests, enabling black-box testing of the API endpoints as an external user would interact with them.
- Temporary files and database records were created during each test run and isolated via the testing framework, ensuring repeatable and non-interfering test execution.
- Unique usernames were generated dynamically using UUIDs to avoid database conflicts and to ensure test independence.
- The output of API endpoints was checked not only for correct HTTP status codes, but also for proper response content, such as JSON keys and error messages.
- For file operations (PDF upload/delete), `SimpleUploadedFile` was used to mock user uploads in memory.
- Third-party integrations (such as image conversion or OCR) were mocked using Python's `unittest.mock.patch` to ensure consistent and isolated backend testing.

3.4 Code Example

Below is a sample test case illustrating the approach for DeepSeek API PDF uploads and prompt handling:

```
@patch("ai_assistant.views.html_to_png")
def test_post_with_pdf_and_prompt_for_map(self, mock_html_to_png):
    mock_html_to_png.return_value = None
    pdf = _create_fake_pdf("content with mind map keyword".encode("utf-8"))
    res = self.client.post(self.url, data={
        'message': 'Please generate a mind map',
        'pdf': pdf
    })
    self.assertIn(res.status_code, [200, 500])
    self.assertTrue('html_url' in res.json() or 'error' in res.json())
```

3.5 Conclusion

By thoroughly covering both standard and edge-case scenarios, these tests provide strong confidence in the correctness, stability, and security of our backend modules. All tests are automated and can be executed as part of the CI/CD workflow.

4 ai_assistant App

Command:

```
python manage.py test ai_assistant
```

Results:

- Number of tests run: **4**
- Total time: **103.868s**
- Result: **OK**

4.1 Sample Output

```
Found 4 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
....
----------------------------------------------------------------------
Ran 4 tests in 103.868s

OK
```

4.2 Mind Map Generation Example

A simple mind map HTML code was generated and exported. The PNG file could not be rendered (file existence check: **False**).

5 IDE App

Command:

```
python manage.py test IDE
```

Results:

- Number of tests run: **7**
- Total time: **5.888s**
- Result: **OK**

5.1 Sample Output

```
Found 7 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
test_add_course_duplicate Response JSON: {'error': 'Course already exists'}
test_add_course_duplicate Response content: b'{"error": "Course already exists"}'
.test_add_course_invalid_json Response JSON: {'error': 'Invalid JSON data'}
test_add_course_invalid_json Response content: b'{"error": "Invalid JSON data"}'
......
----------------------------------------------------------------------
Ran 7 tests in 5.888s

OK
```

6 login App

Command:

```
python manage.py test login
```

Results:

- Number of tests run: **3**
- Result: **OK**

6.1 Sample Output

```
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...
----------------------------------------------------------------------
Ran 3 tests in 1.868s

OK
```

7 self_learn App

Command:

```
python manage.py test self_learn
```

Results:

- Number of tests run: **3**
- Result: **OK**

7.1 Sample Output

```
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...
----------------------------------------------------------------------
Ran 3 tests in 1.789s

OK
```

8 register App

Command:

```
python manage.py test register
```

Results:

- Number of tests run: **4**
- Total time: **0.617s**
- Result: **OK**

8.1 Sample Output

```
Found 4 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
....
----------------------------------------------------------------------
Ran 4 tests in 0.617s

OK
```

9 Conclusion

All tests for the above modules have passed successfully. The system is stable and ready for further development or deployment. Any minor issues encountered (such as failed mind map PNG rendering) do not affect the core functionality of the tested apps.