# Test Suite Documentation

**Overview**

This test suite covers multiple aspects of the Volunteer Management System, including user registration and login, CRUD operations for models like TeamMember, Team, VolunteerPoints, Event, and API endpoint tests. The tests are designed to ensure correct functionality of the system by simulating real-world scenarios and verifying API responses and model behaviors.

In addition to functionality tests, the suite also includes a **Nikto Web Vulnerability Report**, which scans the system for security vulnerabilities and misconfigurations. The Nikto scan identifies potential security issues, such as missing anti-clickjacking headers, improper CORS configurations, and other uncommon HTTP headers that may expose the system to risks. Recommendations for mitigating these vulnerabilities are also provided to enhance the security posture of the application.

## Django test

**Technologies and Frameworks**

* **Django**: Core web framework used in the project.
* **Django REST Framework (DRF)**: Used for building APIs.
* **REST API Testing**: Using APIClient and APITestCase to simulate HTTP requests.
* **Unit Tests**: Standard Django TestCase for model testing.
* **JWT Authentication**: Tests include login and token-based authentication.

**Test Classes Overview**

**1. UserRegistrationTestCase**

Tests the user registration process, including the ability to register a new user and create an admin user.

* **Methods**:
  + test\_user\_registration: Tests successful user registration and checks for JWT tokens.
  + test\_create\_admin\_user: Verifies that an admin user can be created, and its role is correctly assigned.

**2. UserLoginTestCase**

Tests the user login process, ensuring that only valid credentials are accepted.

* **Methods**:
  + test\_invalid\_user\_login: Ensures incorrect credentials result in a 401 response.
  + test\_valid\_user\_login: Verifies successful login and checks for returned JWT tokens.

**3. UserModelTest**

Tests the User model, focusing on user creation, admin rights, and string representation.

* **Methods**:
  + test\_user\_creation: Verifies user creation and checks default admin status.
  + test\_create\_superuser: Tests superuser creation and verifies admin rights.
  + test\_string\_representation: Tests the string representation of a user.

**4. TeamMemberModelTest**

Tests the TeamMember model's creation and string representation.

* **Methods**:
  + test\_create\_member: Verifies member creation with correct data.
  + test\_str: Ensures correct string representation.

**5. TeamModelTest**

Tests the Team model's creation, including team leader assignments and edge cases.

* **Methods**:
  + test\_team\_creation: Verifies successful team creation with a leader.
  + test\_team\_creation\_without\_leader: Ensures teams can be created without a leader.
  + test\_team\_creation\_with\_long\_name: Tests edge case of a long team name.

**6. VolunteerPointsModelTest**

Tests the VolunteerPoints model, including point calculations for both on-water and off-water events.

* **Methods**:
  + test\_volunteer\_points\_on\_water\_event: Verifies point calculation for on-water events.
  + test\_volunteer\_points\_off\_water\_event: Verifies point calculation for off-water events.

**7. ActivityModelTest**

Tests the Activity model's creation and string representation.

* **Methods**:
  + test\_create\_activity: Verifies successful creation of an activity.
  + test\_str: Ensures correct string representation of an activity.

**8. EventModelTest**

Tests the Event model's creation, ensuring correct team and event type assignments.

* **Methods**:
  + test\_create\_event: Verifies event creation with correct attributes.
  + test\_str: Ensures correct string representation of an event.

**9. EventViewSetTests**

Tests CRUD operations for Event API endpoints, focusing on authentication and validation.

* **Methods**:
  + test\_create\_event: Tests successful creation of an event via API.
  + test\_get\_event: Verifies retrieval of event details via API.
  + test\_delete\_event: Tests successful event deletion.
  + test\_delete\_event\_with\_linked\_volunteer\_points: Ensures an event with linked volunteer points cannot be deleted.

**10. VolunteerPointsViewSetTests**

Tests CRUD operations for the VolunteerPoints API, ensuring correct creation, updating, and deletion of volunteer points.

* **Methods**:
  + test\_create\_volunteer\_point: Verifies successful creation of volunteer points.
  + test\_get\_volunteer\_points: Ensures correct retrieval of volunteer points.
  + test\_update\_volunteer\_point: Tests updating volunteer point details.
  + test\_delete\_volunteer\_point: Verifies successful deletion of volunteer points.

**11. AllMembersPointsAPIViewTest**

Tests the API endpoint that retrieves all members' volunteer points.

* **Methods**:
  + test\_get\_all\_members\_points: Verifies successful retrieval of volunteer points for all members.
  + test\_get\_all\_members\_points\_with\_filter: Tests retrieval of volunteer points based on a filter, such as date range or event type.

**Running the Tests**

To run the tests, use Django's built-in testing command:

python manage.py test events

## Security test

**Nikto Web Vulnerability Report**

A computer screen shot of a computer

Description automatically generated

**Target Details:**

* **Target IP:** 127.0.0.1
* **Target Hostname:** localhost
* **Target Port:** 3000
* **Start Time:** 2024-10-08 21:16:11 (GMT+8)
* **End Time:** 2024-10-08 21:16:44 (GMT+8)
* **Total Items Scanned:** 6544
* **Items Reported:** 7

**Findings:**

1. **Server Banner:** No banner retrieved.
2. **Technologies Identified:** Express.js identified via the x-powered-by header.
3. **Vulnerabilities and Misconfigurations:**
   * **Anti-Clickjacking Header Missing**: No X-Frame-Options header, making the app vulnerable to clickjacking.
4. **Uncommon HTTP Headers Found:**
   * **access-control-allow-origin**: Allows all origins, exposing sensitive data.
   * **access-control-allow-methods**: Permits all HTTP methods, including risky ones.
   * **access-control-allow-headers**: Permits all headers.
   * **content-security-policy**: Set to block all external sources, might need adjustments.
   * **x-content-type-options**: Prevents MIME type guessing, a good security measure.
5. **No CGI Directories Found**.
6. **Robots.txt File**: No **disallow** entries.

**Recommendations:**

1. Add the X-Frame-Options header to mitigate clickjacking.
2. Restrict access-control-allow-origin, access-control-allow-methods, and access-control-allow-headers to trusted values.
3. Update the Content Security Policy (CSP) to allow specific trusted sources if needed.