Event and User Interactions
tester: Lorenzo Selvatici

Note: These tests cannot be strictly classified as functional tests since there are no really equivalence classes of input, but they do represent a form of Black Box Testing since we only care about the high level behaviour without thinking about how the models were implemented. However, I could not find a module in the codebase that would have been more suitable to the requirements of this document.

The following description refers to the tests present in the file *core/tests/event_user_interaction_tests.py*

These tests aim to ensure that the database model Event and User are consistent with the expected behaviour established during the Sprint Planning.
Since the correctness of the instances created are enforced at the database level (i.e. you cannot create two users with the same username, a user cannot join an event twice), only the interactions between the two tables are analyzed.

The expected behaviours and the steps followed are described below:

1. The event owner can be deleted from the participants without affecting the event
    a. Create a user U
    b. The user U creates an event E with legal data
    c. The user U joins the event E
    d. The user U leaves the event E
    e. Assert that the event E still exists and user U is no longer among the participants
2. If the event owner is deleted from the user database, then all the events belonging to the event owner should be deleted as well
    a. Create a user U
    b. The user U creates an event E with legal data
    c. The user U is deleted from the database
    d. Assert that an exception is raised when querying for the event E (it was deleted)
3. If an event is deleted, (obviously) the event owner should not be deleted.
    a. Create a user U
    b. The user U creates an event E with legal data
    c. The event E is deleted from the database
    d. Assert that the user U has not been deleted

EventViewSet
Tester: Andrea Silvestroni

Equivalence classes:

- Scopes:
    - Valid = { “all”, “interests”, “tag”, “name”, “date” } (Note: “all” and “interests” are mutually exclusive when using the UI to send requests)
    - Invalid = every other string
- Tag:
    - Valid = every existing tag name currently saved in the database (e.g. “existing tag”)
    - Invalid = every other string (e.g. “non existing tag”)
- Text:
    - Existing = every substring of all existing event names (e.g. “pong”)
    - Non-existing = every other string (e.g. “qwodiqnwocbqwfoiiqwne”)
- Date:
    - Valid = dates in the string form “mm/dd/yy” (e.g. “03/06/17”)
    - Invalid = every other string (e.g. “blabla”)

- Tests (Note: each test includes the value for each one of the variables):
    - No filters (scopes=[”all”], other vars ignored):
        - Return all events with end date greater than current time
    - Filtering by user interests (scopes=[“interests”], other vars ignored):
        - Return all events with end date greater than current time whose tag is in user’s interests
    - Filtering by tag (scopes=[“tag”], other vars ignored apart from **tag** variable):
        - Valid tag: return events with end date greater than current time whose tag corresponds to **tag**
        - Invalid tag: server error (exception not caught, described in working prototype document)
    - Filtering by name (scopes=[“name”], other vars ignored apart from **text** variable):
        - Existing substring: return events with end date greater than current time whose name include the substring
        - Non existing substring: return no events
    - Filtering by date (scopes=[“date”], other vars ignored apart from **date** variable):
        - Valid date: return events with end date greater than current time whose start date is greater than **date**
        - Invalid date: server error (exception not caught, described in working prototype document)

        - Combined scopes:
            - Scopes can be combined freely in order to use multiple filters, but:
                - “Interests” and “tag” filters are mutually exclusive, if both of them are present only “interests” is considered
                - “All” has the lowest priority, which means that if other scopes are used it gets ignored (the unit returns filtered results)
                - “Name” and “date” scopes require the corresponding request variables (**text** and **date**) to hold a valid value, otherwise an error is thrown

Users as members of events
Tester: Alessia Masola

After creating an event and setting the number of participants to be 2, the following inputs and actions were used to test that the behaviour of EventMemberView operates in conformance with the requirement specification:

- The user successfully joins the event
    - New user attempts to join the event
    - “Join” button changes into “Leave”, and indicates the event has been joined
    - The number of the participants increases by one instantaneously (from 0 it becomes 1)
    - The list of members updates with the new member name

- The user successfully leaves the event
    - User attempts to leave the event
    - “Leave” button changes into “Join” and indicates the event has been left
    - The number of participants decreases by 1
    - The list of members updates removing the name of the user leaving the event

- If the event is full (number of participants = 2/2) and a user tries to join the event, no error is raised
    - New user attempts to join event
    - The number of participants does not increase
    - The list of the members does not update
    - No error or warning raised
    - The action has no effect

ProfileView
Tester: Henry Ball

The following criteria are set in place to ensure that all functionality depending upon the profile view works to specification. The expected behavior of the profile view is as follows:

1. When a user visits his/her profile page the following must be true:
    a. The selected user model must be equivalent to the user model with the primary key matching the one contained in the url
    b. The username displayed must be equivalent to the username field contained by the selected user model
    c. The interests displayed must be equivalent to the interest_tags field contained by the selected user model
    d. The events displayed must match all events of which the selected user is either an owner or participant
    e. Owning and participating in an event are not mutually exclusive, however if an event is both owned and joined by a user it will be displayed as an owned event

2. When a user selects a new interest the following must be be true:
    a. The current primary key contained by the url must match that of the currently logged in user (if not the select box must be inaccessible)
    b. The selected tag must not be currently contained by the selected user models interest_tags field
    c. Once a tag is added, the list of all tags must be sent to the view which in turn updates the interest_tags field of the selected user model
    d. The interest_tags field of the selected user model must now be equivalent to the displayed tags

3. When a user deletes an interest the following must be true:
    a. The current primary key contained by the url must match that of the currently logged in user (if not the select box must be inaccessible)
    b. The deleted tag must be currently contained by the selected user models interest_tags field
    c. Once a tag is deleted, the list of all tags must be sent to the view which in turn updates the interest_tags field of the selected user model
    d. The interest_tags field of the selected user model must now be equivalent to the displayed tags

Map View
Tester: Mattheo Ioannou

The following tests cannot be strictly classified as functional tests since there is no input data that tests could take in. However, these tests are testing functionality in terms of user interface (as a View is being tested), ensuring a good user experience with the Map view. Therefore, the following is a list of tests that were manually performed multiple times with different data:

1. The user should be able to view the map page without logging in, but with limited functionality, by visiting https://pinned-app-deploy.herokuapp.com/map/

   a. Location of user is displayed
   b. Events are displayed on map as pins
   c. Upcoming events are visible
   d. Map filters, event search on map and date/time filters are not visible
   e. Dropping a pin on map doesn't allow for event creation
2. Having logged in, the user can see the map page with full functionality
   a. Location of user is displayed
   b. Events are displayed on map as pins
   c. Upcoming events are visible
   d. Map filters, event search on map and date/time filters are visible
   e. Dropping a pin on map allows for event creation
3. The user can filter the map using the dropdown list
   a. If the user selects 'All Events', all events are shown on the map.
   b. If the user selects 'My Interests', only the events whose tag is part of the user's interests are shown on the map.
   c. If the user selects a specific tag (part of their interests), only the events whose tag is the specified tag are shown on the map.
   d. The user doesn't see specific tags as options for filters that they aren't interested in.
4. The user can filter the map using the search bar
   a. If the user types something in the search bar, the events which contain the text the user typed are shown.
5. The user can filter the map using the datepicker
   a. If the user selects a date from the datepicker, only the events which start after the selected date are shown on the map.

The following was tested using two accounts, in which one account created events (user-A), and the other account was used to test the map view (user-B), with both users having a list of interests in their profiles:

1. When a user (e.g. user-A) drops a pin and creates an event:
   b. The event is saved to the database
   c. The event can be viewed by the user (e.g. user-A)
   d. The event can be viewed by all other users (e.g. user-B)
2. When a user drops a pin and creates an event that is less than 15 miles away from them (e.g. user-A), and whose tag is part of their interests:
   a. They do not get a notification telling them a new event is near them, because they are the owner of the event
3. When a user drops a pin and creates an event (e.g. user-A) that is less than or equal to 15 miles away from another user (e.g. user-B), and whose tag is part of the other user's interests:
   a. The other user (user-B) gets a notification telling them a new event is near them (with the event's name), without needing to refresh the page.

4. When a user drops a pin and creates an event (e.g. user-A) that is more than 15 miles away from another user (e.g. user-B), and whose tag is part of the other user's interests:
    a. The other user (user-B) doesn't get a notification telling them a new event is near them (with the event's name), because it is more than 15 miles away.
5. When a user drops a pin and creates an event (e.g. user-A) that is less than or equal to 15 miles away from another user (e.g. user-B), and whose tag is not part of the other user's interests:
    a. The other user (user-B) doesn't get a notification telling them a new event is near them (with the event's name), because they are not interested in this event's tag.