Unit Test Plan

Module #:	Application #:
Tester: HAPPI NOUWE MOREL ALBAN	Test Manager: TIENTCHEU TAKOU DONALD
Module Overview : event.js, search.js, filters.js	
The module tests the functionality of different parts of an event management system, including the creation and management of events, filtering events based on their dates, and searching for events based on specific criteria.	
Module Inputs	
Event **data**: Objects containing details of events such as name, date, ticket price, total tickets, tickets remaining. Search **predicates**: Functions used to filter or search events based on specific conditions. **Date values**: Used to filter events based on their occurrence within certain date ranges (e.g., today, next 7 days, next 30 days).	
Module Outputs	
Boolean values: Indicating if an event is sold out, if it occurs today, or within specified date ranges. **Tag lines**: Strings providing information about event status (e.g., "Event Sold Out!", "Hurry only X tickets left!"). **Event objects**: Created events or filtered lists of events based on search criteria.	
Logic Flow	

event.js

1. **Création d'un événement**:
   - **Entrée**: id, name, ticketPrice, totalTickets, ticketsRemaining, date
   - **Processus**:
     - Validation des paramètres.
     - Si une validation échoue, lever une exception correspondante (InvalidEventNameError, InvalidEventPriceError).
     - Sinon, créer un nouvel objet Event avec les propriétés fournies.
   - **Sortie**: Objet Event.

2. **Vérification si un événement est sold out**:
   - **Entrée**: Objet Event.
   - **Processus**:
     - Vérifier si ticketsRemaining est 0.
   - **Sortie**: Boolean (true si sold out, false sinon).

3. **Génération de la tagline d'un événement**:
   - **Entrée**: Objet Event, minimumTicketCount, isPopular.
   - **Processus**:
     - Si ticketsRemaining est 0, retourner "Event Sold Out!".
     - Si ticketsRemaining < minimumTicketCount, retourner "Hurry only X tickets left!".
     - Si l'événement est populaire, retourner un message promotionnel spécifique.
     - Sinon, retourner "Don't miss out, purchase your ticket now!".
   - **Sortie**: String.

4. **Création d'un événement via `createEvent`**:
   - **Entrée**: name, price, availableTickets.
   - **Processus**:
     - Valider les paramètres.
     - Si une validation échoue, lever une exception correspondante (InvalidEventNameError, InvalidEventPriceError).
     - Sinon, créer un nouvel objet Event.
   - **Sortie**: Objet Event.

filters.js

1. **Filtrer les événements pour aujourd'hui**:
   - **Entrée**: Objet Event.
   - **Processus**:
     - Comparer la date de l'événement à la date actuelle.
   - **Sortie**: Boolean (true si l'événement a lieu aujourd'hui, false sinon).

search.js

1. **Recherche d'événements via un prédicat**:
   - **Entrée**: Liste d'objets Event, fonction prédicat.
   - **Processus**:
     - Appliquer la fonction prédicat à chaque événement de la liste.
     - Filtrer les événements qui satisfont le prédicat.
   - **Sortie**: Liste d'objets Event qui satisfont le prédicat.

Test Data

List all test cases to be executed.

Positive Test cases

1. **Test Case 1**: `isSoldOut` should return true when tickets remaining is 0.
   - **Expected Outcome**: `true`
2. **Test Case 2**: `isSoldOut` should return false when tickets remaining is greater than 0.
   - **Expected Outcome**: `false`
3. **Test Case 3**: `getTagLine` should return "Event Sold Out!" when event is sold out.
   - **Expected Outcome**: `"Event Sold Out!"`
4. **Test Case 4**: `getTagLine` should return "Hurry only X tickets left!" when tickets remaining is less than minimum.
   - **Expected Outcome**: `"Hurry only 5 tickets left!"`
5. **Test Case 5**: `getTagLine` should return "Hurry only 1 ticket left!" when only one ticket is remaining.
   - **Expected Outcome**: `"Hurry only 1 ticket left!"`
6. **Test Case 6**: `getTagLine` should return a message for popular events.
   - **Expected Outcome**: `"This Event is getting a lot of interest. Don't miss out, purchase your ticket now!"`
7. **Test Case 7**: `getTagLine` should return a standard message if event is not sold out or popular.
   - **Expected Outcome**: `"Don't miss out, purchase your ticket now!"`
8. **Test Case 8**: `createEvent` should create a valid event with correct parameters.
   - **Expected Outcome**: Event object with specified parameters
9. **Test Case 9**: `today` should return true if the event is today.
   - **Expected Outcome**: `true`
10. **Test Case 10**: `next7Days` should return true if the event is within the next 7 days.
    - **Expected Outcome**: `true`
11. **Test Case 11**: `next30Days` should return true if the event is within the next 30 days.
    - **Expected Outcome**: `true`
12. **Test Case 12**: `getEvents` should return events that satisfy the predicate.
    - **Expected Outcome**: Filtered events based on the predicate

Number each test case. Indicate the test to be performed and expected outcome

Negative Test Cases

Test Case 13: `createEvent` should throw an error if event name exceeds 200 characters.

- **Expected Outcome**: Throw `InvalidEventNameError`

Test Case 14: `createEvent` should throw an error if event price is negative.

- **Expected Outcome**: Throw `InvalidEventPriceError`

Test Case 15: `createEvent` should throw an error if total tickets are less than 1.

- **Expected Outcome**: Throw `InvalidEventPriceError`

Test Case 16: `today` should return false if the event is not today.

- **Expected Outcome**: `false`

Test Case 17: `next7Days` should return false if the event is today.

- **Expected Outcome**: `false`

Test Case 18: `next7Days` should return false if the event is after the next 7 days.

- **Expected Outcome**: `false`

Test Case 19: `next7Days` should return false if the event is in the past.

- **Expected Outcome**: `false`

Test Case 20: `next30Days` should return false if the event is today.

- **Expected Outcome**: `false`

Test Case 21: `next30Days` should return false if the event is after the next 30 days.

- **Expected Outcome**: `false`

Test Case 22: `next30Days` should return false if the event is in the past.

- **Expected Outcome**: `false`

Test Case 23: `getEvents` should return an empty array if no events satisfy the predicate.

- **Expected Outcome**: `[]`

Test Case 24: `getEvents` should return an empty array if the events array is empty.

- **Expected Outcome**: `[]`

Listin valid data selections

Interface Modules

Output Data:

Event status: Results from `isSoldOut`, `getTagLine`, `today`, `next7Days`, `next30Days`

Event lists: Results from `getEvents`

Data Input:

Event objects: Inputs for creating and testing events

Predicates: Functions for filtering events

Internal Program Interface:

- **Module-to-Module**: Function calls within the event management system

External Program Interface:

- **Test Suite Interface**: Interaction with the Vitest testing framework

Test Tools

Unit Testing Software

- **Vitest**: JavaScript testing framework used for running the tests.

Regression Testing

- **Vitest**: Future regression tests can be conducted using the same Vitest framework.
- **Location**: Tests are located in the tests directory alongside the respective modules.

Identify software used for unit testing.
Identify names and locations of software for future regression testing.

Archive Plan

Location: Archived data and test results will be stored in the project's `archive` directory.

Access Procedures: Access to the archived data will require appropriate project permissions and can be obtained via project maintainers or through a request process defined in the project documentation.

Specify location of archived data.

Define procedures required to obtain access to this data.

Updates

Procedure: Any updates to the unit test plan will be documented in the project's change log and will follow the standard project update procedure.

Responsibility: Updates will be managed by the QA team or designated project maintainers, ensuring consistency and accuracy across all test cases.

Identify how unit test plan will be updated.

The unit test plan will be updated