# **Overview of the Testing Strategy**

This project’s testing approach focuses on unit testing with the **Google Test** framework (**gtest**). The primary goal of these tests is to ensure that the various components (classes) of the project function correctly by testing them in isolation.

Each test case is written using a **test fixture**, which ensures tests do not interfere with each other. The test fixture handles setup and teardown procedures for each test. The individual tests focus on verifying method outputs, setting and receiving properties, and ensuring proper object handling.

# **Breakdown of Test Cases**

### 1. ****Workshop Class (WorkshopTest)****

* **Test Instructor**: Verifies that the correct instructor name ("John Doe") is returned by the getInstructor() method.
* **Test Online Status**: Confirms that the workshop is online using the isOnline() method.
* **Test Display Information**: Tests the displayInformation() method without programmatically checking its output.

### 2. ****Individualized Instruction (oneOnOneTest)****

* **Test Participants**: Confirms the participants of the one-on-one meeting using getParticipant1() and getParticipant2().
* **Display Information Test**: Verifies that the displayInformation() method works as intended, though the output is not validated.

### 3. ****Class Meeting (MeetingTest)****

* **Test Agenda**: Confirms that the getAgenda() method correctly retrieves the agenda.
* **Test Display Information**: Verifies that the meeting details are displayed properly by the displayInformation() method.

### 4. ****General Events (GeneralTest)****

* **Test Is Public**: Confirms whether the event is marked as public using the getIsPublic() method.
* **Test Display Information**: Verifies that the event details are displayed by the displayInformation() method.

### 5. ****Club Events (ClubEventTest)****

* **Constructor Test**: Verifies that a club event is created with the appropriate attributes, including event type, title, and number of attendees.
* **Set Event Type Test**: Ensures that the setType() method can be used to modify the event type.
* **Add/Remove Attendee Test**: Confirms the correct number of attendees and tests the ability to add or remove guests.
* **Set Attendance Limit Test**: Verifies that the setAttendeeLimit() method can set and retrieve the event’s attendance limit.

# **Findings and Remarks**

* **Passing Tests**: All tests confirm the correctness of getter and setter methods, along with core features like adding/removing attendees and verifying methods like displayInformation().
* **Test Output Validation**: While output verification is not performed, tests such as displayInformation() and sendEmail() aim to confirm that the methods operate as expected. Output could be captured for validation or redirected to a file in more complex scenarios.
* **Functionality**: The tests validate the core functionality of the classes, including state management, initialization, and inter-class method interactions.
* **Tools Used**: The Google Test framework (**gtest**) was used for creating and running unit tests. The framework includes assertions like EXPECT\_EQ, EXPECT\_TRUE, etc., to verify expected values.

# **Summary of Testing Results**

* **Test Coverage**: The tests cover essential areas, including:
  + Initialization and object creation.
  + Setter and getter methods.
  + Core functionality related to email handling, workshops, one-on-one meetings, events, and meetings.
* **No Issues Found**: The tests confirm that all features, including methods and properties, work as intended.

# **Conclusion**

Based on the testing methodology and results, the core functionality of the classes has been thoroughly validated. Each test was used to isolate and confirm the behavior of individual classes.

Please let me know if you need further clarification or wish to add more examples to the testing methodology!