### 1. \*\*Test Plan for Unit Tests\*\*

\*\*Test Plan Objective:\*\*

The objective of this test plan is to ensure that the appointment scheduling system correctly handles time conflicts and verifies that the database connection is properly established.

\*\*Test Cases:\*\*

1. \*\*TestDatabaseConnection\*\*:

- \*\*Objective\*\*: Verify that a connection to the database can be established without throwing exceptions.

- \*\*Steps\*\*:

1. Open a connection to the database using the provided connection string.

2. Ensure that no exceptions are thrown during the connection process.

- \*\*Expected Result\*\*: The connection should be established successfully without exceptions.

2. \*\*TestIsTimeConflicted\_NoConflict\*\*:

- \*\*Objective\*\*: Verify that no time conflict is reported when there are no overlapping appointments.

- \*\*Steps\*\*:

1. Provide a start and end time for an appointment that should not conflict with existing appointments.

2. Call the `CheckTimeConflict` method with these times.

3. Assert that the result is `False`.

- \*\*Expected Result\*\*: The method should return `False`, indicating no conflict.

3. \*\*TestIsTimeConflicted\_WithConflict\*\*:

- \*\*Objective\*\*: Verify that a time conflict is correctly reported when the provided times overlap with existing appointments.

- \*\*Steps\*\*:

1. Provide a start and end time for an appointment that should conflict with existing appointments.

2. Call the `CheckTimeConflict` method with these times.

3. Assert that the result is `True`.

- \*\*Expected Result\*\*: The method should return `True`, indicating a conflict.

\*\*Screenshots:\*\*

While screenshots are not provided here, you would typically include screenshots of:

- The test results in your test runner or IDE showing passed/failed status.

- Any relevant portions of your configuration or connection settings.

### 2. \*\*Unit Test Scripts\*\*

```csharp

[TestFixture]

public class AppointmentTests

{

private string connectionString;

[SetUp]

public void Setup()

{

connectionString = ConfigurationManager.ConnectionStrings["localdb"]?.ConnectionString;

if (string.IsNullOrEmpty(connectionString))

{

throw new InvalidOperationException("The connection string 'localdb' could not be found.");

}

}

[Test]

public void TestDatabaseConnection()

{

using (MySqlConnection con = new MySqlConnection(connectionString))

{

Assert.DoesNotThrow(() => con.Open(), "Database connection should open without throwing an exception.");

}

}

[Test]

public void TestIsTimeConflicted\_NoConflict()

{

// Arrange

var start = new DateTime(2024, 07, 25, 10, 0, 0);

var end = new DateTime(2024, 07, 25, 11, 0, 0);

// Act

var isConflicted = CheckTimeConflict(start, end);

// Assert

Assert.That(isConflicted, Is.False, "There should be no conflict with the existing appointments.");

}

[Test]

public void TestIsTimeConflicted\_WithConflict()

{

// Arrange

var start = new DateTime(2024, 07, 23, 09, 0, 0);

var end = new DateTime(2024, 07, 23, 10, 0, 0);

// Act

var isConflicted = CheckTimeConflict(start, end);

// Assert

Assert.That(isConflicted, Is.True, "There should be a conflict with an existing appointment.");

}

private bool CheckTimeConflict(DateTime start, DateTime end)

{

var conflictFound = false;

using (var connection = new MySqlConnection(connectionString))

{

connection.Open();

var query = "SELECT COUNT(\*) FROM appointment WHERE (start < @end AND end > @start)";

using (var command = new MySqlCommand(query, connection))

{

command.Parameters.AddWithValue("@start", start);

command.Parameters.AddWithValue("@end", end);

var count = Convert.ToInt32(command.ExecuteScalar());

if (count > 0)

{

conflictFound = true;

}

}

}

return conflictFound;

}

}

```

### 3. \*\*Results of the Unit Tests\*\*

- \*\*Database Connection\*\*: The test checks if the connection to the database can be successfully established without exceptions. A successful result confirms that the connection string is correct and the database is reachable.

- \*\*No Conflict\*\*: For the test case with no expected conflict, the method should return `False`. This means that the times tested do not overlap with existing appointments.

- \*\*With Conflict\*\*: For the test case with an expected conflict, the method should return `True`, indicating that the times do overlap with existing appointments.

\*\*All tests have passed as seen below.\*\*

A screenshot of a computer

Description automatically generated

### 4. \*\*Summaries of Changes Resulting from Completed Tests\*\*

Since all tests passed:

- \*\*Database Connection\*\*: The connection setup is correct and no changes are necessary.

- \*\*No Conflict\*\*: The implementation correctly identifies non-conflicting times, and no changes are required.

- \*\*With Conflict\*\*: The implementation correctly identifies conflicting times, and no changes are required.