Code :

```
from datetime import datetime


class Event:
    def init(self, event_id, name, start_time, end_time):
        self.event_id = event_id
        self.name = name
        self.start_time = datetime.strptime(start_time, '%Y-%m-%d %H:%M')
        self.end_time = datetime.strptime(end_time, '%Y-%m-%d %H:%M')

    def repr(self):
        return f"Event({self.event_id}, '{self.name}', '{self.start_time}',
'{self.end_time}')"
##

### 2. The EventCalendar Class

#python
class EventCalendar:
    def init(self, calendar_id):
        self.calendar_id = calendar_id
        self.events = {}

    def add_event(self, event):
        if event.event_id in self.events:
            raise ValueError("Event ID already exists")
        self.events[event.event_id] = event

    def get_event(self, event_id):
        return self.events.get(event_id)
```

```
 def update_event(self, event_id, **kwargs):
        if event_id not in self.events:
            raise ValueError("Event not found")

        event = self.events[event_id]
        if 'name' in kwargs:
            event.name = kwargs['name']
        if 'start_time' in kwargs:
            event.start_time = datetime.strptime(kwargs['start_time'], '%Y-%m-
%d %H:%M')
        if 'end_time' in kwargs:
            event.end_time = datetime.strptime(kwargs['end_time'], '%Y-%m-
%d %H:%M')

    def delete_event(self, event_id):
        if event_id in self.events:
            del self.events[event_id]
        else:
            raise ValueError("Event not found")

    def list_events(self):
        return list(self.events.values())

    def repr(self):
        return f"EventCalendar('{self.calendar_id}', Events={len(self.events)})"
```

### 3. The OnlinePlatformSync Class

#python

```python
class OnlinePlatformSync:
    def sync_events_with_platforms(self, calendar, platform_data):
        # Fake synchronization logic
        print("Syncing events:")
        for event in calendar.list_events():
            print(f" - {event.name} synced to platform {platform_data['platform_name']}")
```

#

### Unit Tests

#We need to ensure our classes work correctly. We'll use unittest for writing the test cases.

#python

```python
import unittest


class TestEventManagement(unittest.TestCase):
    def setUp(self):
        self.calendar = EventCalendar('Theater123')
        self.event = Event('1', 'Hamlet', '2023-04-10 18:00', '2023-04-10 21:00')
        self.calendar.add_event(self.event)

    def test_event_creation(self):
        self.assertEqual(repr(self.event), "Event(1, 'Hamlet', '2023-04-10 18:00:00', '2023-04-10 21:00:00')")

    def test_add_event(self):
        self.assertEqual(len(self.calendar.events), 1)
    def test_update_event(self):
        self.calendar.update_event('1', name='Hamlet Revised')
        self.assertEqual(self.calendar.get_event('1').name, 'Hamlet Revised')

    def test_delete_event(self):
        self.calendar.delete_event('1')
        self.assertEqual(len(self.calendar.events), 0)

    def test_sync_events(self):
        sync = OnlinePlatformSync()
        with self.assertLogs() as cm:
            sync.sync_events_with_platforms(self.calendar, {'platform_name':
'Ticketmaster'})
            self.assertIn('Hamlet synced', cm.output[0])
if _name_ == 'main':
    unittest.main()
```