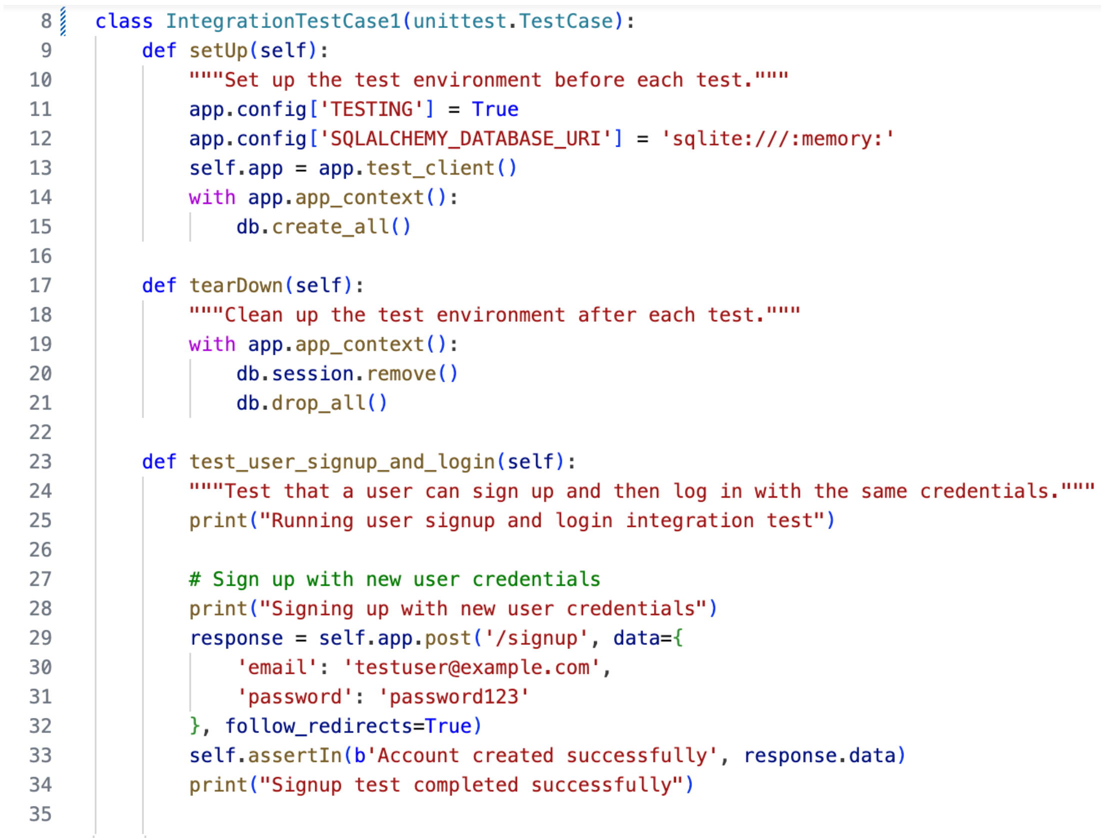
Integration Test Scripts 1:



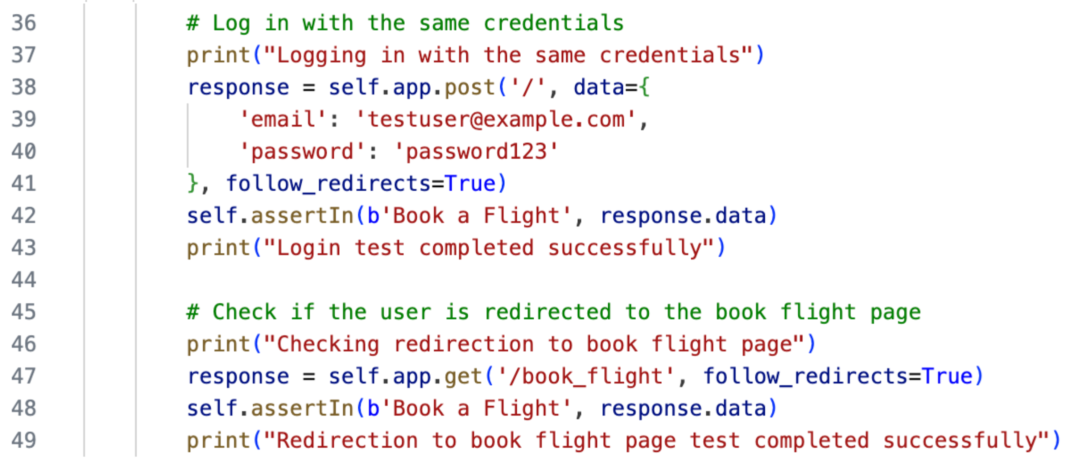
```python
class IntegrationTestCase1(unittest.TestCase):
    def setUp(self):
        """Set up the test environment before each test."""
        app.config['TESTING'] = True
        app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///:memory:'
        self.app = app.test_client()
        with app.app_context():
            db.create_all()

    def tearDown(self):
        """Clean up the test environment after each test."""
        with app.app_context():
            db.session.remove()
            db.drop_all()

    def test_user_signup_and_login(self):
        """Test that a user can sign up and then log in with the same credentials."""
        print("Running user signup and login integration test")

        # Sign up with new user credentials
        print("Signing up with new user credentials")
        response = self.app.post('/signup', data={
            'email': 'testuser@example.com',
            'password': 'password123'
        }, follow_redirects=True)
        self.assertIn(b'Account created successfully', response.data)
        print("Signup test completed successfully")

```



```python
        # Log in with the same credentials
        print("Logging in with the same credentials")
        response = self.app.post('/', data={
            'email': 'testuser@example.com',
            'password': 'password123'
        }, follow_redirects=True)
        self.assertIn(b'Book a Flight', response.data)
        print("Login test completed successfully")

        # Check if the user is redirected to the book flight page
        print("Checking redirection to book flight page")
        response = self.app.get('/book_flight', follow_redirects=True)
        self.assertIn(b'Book a Flight', response.data)
        print("Redirection to book flight page test completed successfully")
```

This test verifies the basic user authentication flow. It checks that a new user can successfully sign up and log in using their own credentials. After logging in, it confirms that the user is redirected to the “Book a Flight” page, ensuring that the signup and login functionalities work as intended and that the user experience proceeds smoothly to the flight booking interface.

The output for test scripts 1:

```
Running user signup and login integration test
Signing up with new user credentials
Signup test completed successfully
Logging in with the same credentials
Login test completed successfully
Checking redirection to book flight page
Redirection to book flight page test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.251s

OK
```

Integration Test Scripts 2:

```python
class IntegrationTestCase2(unittest.TestCase):
    def setUp(self):
        """Set up the test environment before each test."""
        app.config['TESTING'] = True
        app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///:memory:'
        self.app = app.test_client()
        with app.app_context():
            db.create_all()
            test_flight = Flight(
                flight_number="AB123",
                departure_airport="JFK",
                arrival_location="LAX",
                departure_time=datetime(2024, 11, 5, 14, 0),
                arrival_time=datetime(2024, 11, 5, 17, 30),
                cost=299.99,
                seats=generate_seat_map()
            )
            db.session.add(test_flight)
            db.session.commit()

            self.test_flight = Flight.query.filter_by(flight_number="AB123").first()

    def tearDown(self):
        """Clean up the test environment after each test."""
        with app.app_context():
            db.session.remove()
            db.drop_all()

    def test_login_book_flight(self):
        """Test user login, booking a flight, and viewing booking history."""
        print("Running login, book flight, and check booking history integration test")

        # Sign up with new user credentials
        print("Signing up with new user credentials")
        response = self.app.post('/signup', data={
            'email': 'testuser@example.com',
            'password': 'password123'
        }, follow_redirects=True)
        self.assertIn(b'Account created successfully', response.data)
        print("Signup test completed successfully")
        # Log in with the same credentials
        print("Logging in with the same credentials")
        response = self.app.post('/', data={
            'email': 'testuser@example.com',
            'password': 'password123'
        }, follow_redirects=True)
        self.assertIn(b'Book a Flight', response.data)
        print("Login test completed successfully")

        # Search for flights and select a flight
        print("Searching for flights and selecting a flight")
        response = self.app.post('/search_flights', data={
            'departure_airport': 'JFK',
            'arrival_location': 'LAX',
            'departure_date': '2024-11-05'
        }, follow_redirects=True)
        self.assertIn(b'AB123', response.data)
        print("Flight search test completed successfully")

        # Select a seat on the flight
        flight_id = 1  # Assuming flight ID 1 exists
        print("Selecting a seat for the flight")
        response = self.app.post(f'/select_seat/{flight_id}', data={
            'seat': '0,0'  # First row, first seat
        }, follow_redirects=True)
        self.assertIn(b'Payment Method', response.data)
        print("Seat selection test completed successfully")

        # Process payment and confirm booking
        print("Processing payment and confirming booking")
        response = self.app.post(f'/payment_method/{flight_id}', follow_redirects=True)
        self.assertIn(b'Payment successful!', response.data)
        print("Booking confirmation test completed successfully")
```

This test simulates the complete flight booking process for a user. It begins with user signup and login, followed by searching for flights based on specific departure and arrival locations and dates. The test then proceeds to select a flight from the search results, choose a seat, and process payment. Finally, it confirms that the booking is successful.

The output for test scripts 2:

```
Running login, book flight, and check booking history integration test
Signing up with new user credentials
Signup test completed successfully
Logging in with the same credentials
Login test completed successfully
Searching for flights and selecting a flight
Flight search test completed successfully
Selecting a seat for the flight
/Users/lucaschu/Documents/GitHub/cisc327-group11/.venv/lib/python3.11/site-packages/flask_sqlalchemy/query.py:
30: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becom
es a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background
 on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  rv = self.get(ident)
Seat selection test completed successfully
Processing payment and confirming booking
Booking confirmation test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.276s

OK
```

Integration Test Scripts 3:

```python
class IntegrationTestCase3(unittest.TestCase):
    def setUp(self):
        """Set up the test environment before each test."""
        app.config['TESTING'] = True
        app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///:memory:'
        self.app = app.test_client()
        with app.app_context():
            db.create_all()

    def tearDown(self):
        """Clean up the test environment after each test."""
        with app.app_context():
            db.session.remove()
            db.drop_all()

    def test_signup_login_logout(self):
        """Test user signup, login, booking a flight, and viewing booking history."""
        print("Running signup, login, book flight, and check booking history integration test")
        # Sign up and login tests
        response = self.app.post('/signup', data={
            'email': 'testuser@example.com',
            'password': 'password123'
        }, follow_redirects=True)
        self.assertIn(b'Account created successfully', response.data)

        response = self.app.post('/', data={
            'email': 'testuser@example.com',
            'password': 'password123'
        }, follow_redirects=True)
        self.assertIn(b'Book a Flight', response.data)
        #User attempt logout
        response = self.app.get('/logout', follow_redirects=True)
        self.assertEqual(response.status_code, 200)
        self.assertIn(b'You have been logged out', response.data)
        print("Signup, login, and logout test completed successfully")
```

This test focuses on the user session management, specifically the logout functionality. After a user sign up and log in, the test verifies that the user can successfully log out of the system. It checks for a confirmation message indicating that the user has been logged out and ensures that the session is properly terminated. This test is crucial for validating that users can securely end their sessions, which is important for account security and proper application behavior.

The output for test scripts 3:

```
Running signup, login, book flight, and check booking history integration test
Signup, login, and logout test completed successfully
.
----------------------------------------------------------------------
Ran 1 test in 0.255s

OK
```

Instructions:

1. To run the integration test scripts, it is important to delete the users.db database file to ensure a clean database state.

   You can run

   ```
   rm -f instance/users.db
   ```

   in terminal to delete the database.

2. Run the Integration Tests:
   Execute the test script using the Python interpreter:
   ```
   python -m unittest test_integration.py
   ```

   The script will automatically discover and run all test cases defined in the file.

   Or, if you want to run a specific test case, you can specify it in the command:
   ```
   python -m unittest test_integration.IntegrationTestCase1
   ```