Deliverable 5 - Testing Document

Ian Catania, Anthony Clavette, Youmeng Hin

A testing model for the Tennis Club Reservation System software, composed of unit testing and integration testing, is outlined in this document. The system's backend is written in Python programming language, so the Python standard library module "unittest," a class-based test case framework, is used for these test cases.

Unit testing (Figure 1 & Figure 2)

Each unit of the software is isolated from the system and tested independently. The purpose is to find faults in objects or components that participate in the system's operation. The test cases pictured below are designed to test the presence of the application's pages and to make sure they are configured with the correct links.

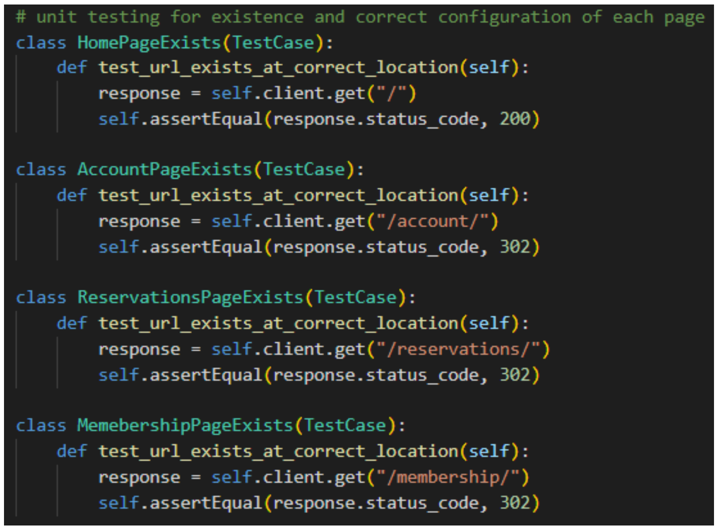
Integration testing (Figure 3 & Figure 4)

Rather than testing isolating components, multiple components of the system are grouped together and tested to ensure they can interact without problems to properly carry out system requirements. These test cases are designed to fully populate objects with example data and ensure the data remains accurate and accessible for each functionality of the application.

Results (Figure 5)

All test cases ran successfully and ensured no errors exist among the components within the system. Therefore, we did not observe any differences between the system's required behavior and its existing behavior.

Figure 1



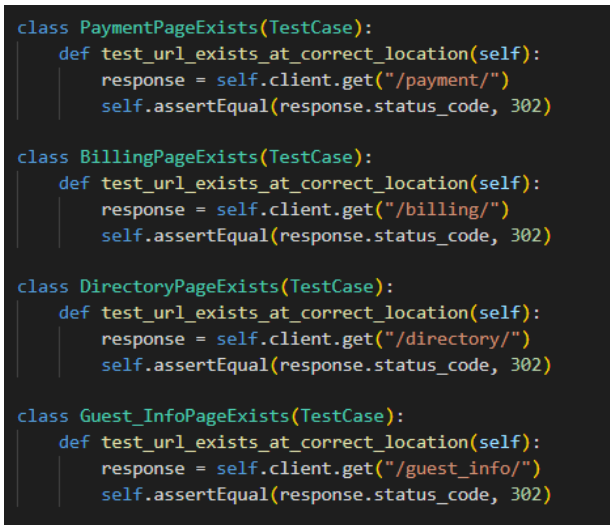
```python
# unit testing for existence and correct configuration of each page
class HomePageExists(TestCase):
    def test_url_exists_at_correct_location(self):
        response = self.client.get("/")
        self.assertEqual(response.status_code, 200)

class AccountPageExists(TestCase):
    def test_url_exists_at_correct_location(self):
        response = self.client.get("/account/")
        self.assertEqual(response.status_code, 302)

class ReservationsPageExists(TestCase):
    def test_url_exists_at_correct_location(self):
        response = self.client.get("/reservations/")
        self.assertEqual(response.status_code, 302)

class MemebershipPageExists(TestCase):
    def test_url_exists_at_correct_location(self):
        response = self.client.get("/membership/")
        self.assertEqual(response.status_code, 302)
```

Figure 2



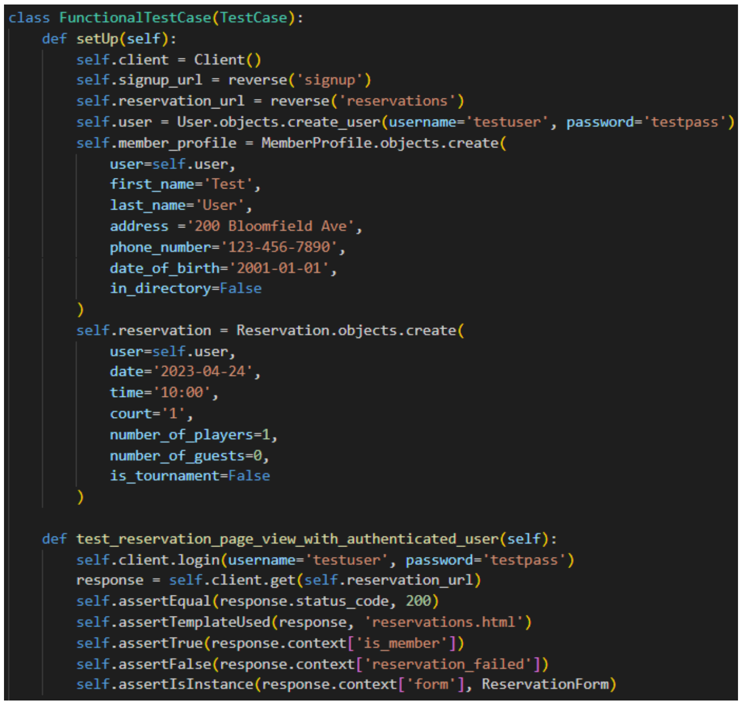
```python
class PaymentPageExists(TestCase):
    def test_url_exists_at_correct_location(self):
        response = self.client.get("/payment/")
        self.assertEqual(response.status_code, 302)

class BillingPageExists(TestCase):
    def test_url_exists_at_correct_location(self):
        response = self.client.get("/billing/")
        self.assertEqual(response.status_code, 302)

class DirectoryPageExists(TestCase):
    def test_url_exists_at_correct_location(self):
        response = self.client.get("/directory/")
        self.assertEqual(response.status_code, 302)

class Guest_InfoPageExists(TestCase):
    def test_url_exists_at_correct_location(self):
        response = self.client.get("/guest_info/")
        self.assertEqual(response.status_code, 302)
```

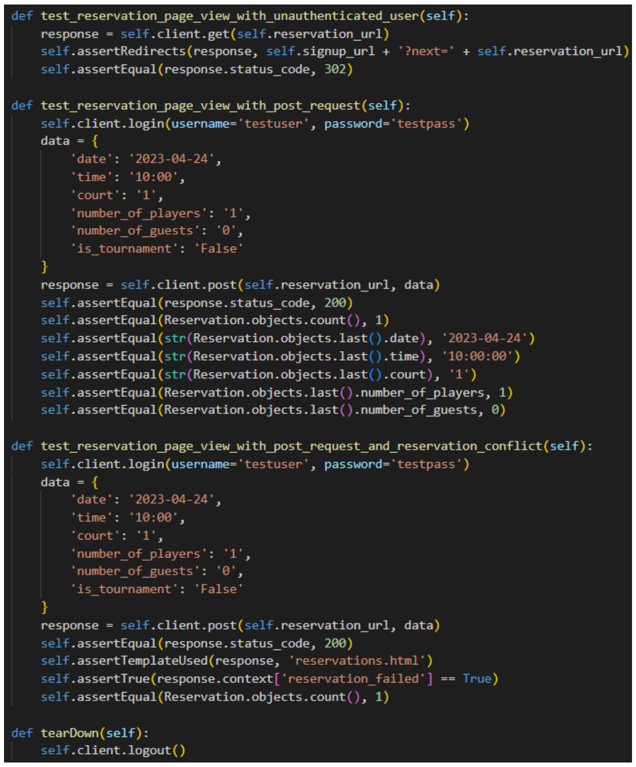
Figure 3



```python
class FunctionalTestCase(TestCase):
    def setUp(self):
        self.client = Client()
        self.signup_url = reverse('signup')
        self.reservation_url = reverse('reservations')
        self.user = User.objects.create_user(username='testuser', password='testpass')
        self.member_profile = MemberProfile.objects.create(
            user=self.user,
            first_name='Test',
            last_name='User',
            address ='200 Bloomfield Ave',
            phone_number='123-456-7890',
            date_of_birth='2001-01-01',
            in_directory=False
        )
        self.reservation = Reservation.objects.create(
            user=self.user,
            date='2023-04-24',
            time='10:00',
            court='1',
            number_of_players=1,
            number_of_guests=0,
            is_tournament=False
        )

    def test_reservation_page_view_with_authenticated_user(self):
        self.client.login(username='testuser', password='testpass')
        response = self.client.get(self.reservation_url)
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'reservations.html')
        self.assertTrue(response.context['is_member'])
        self.assertFalse(response.context['reservation_failed'])
        self.assertIsInstance(response.context['form'], ReservationForm)
```

Figure 4



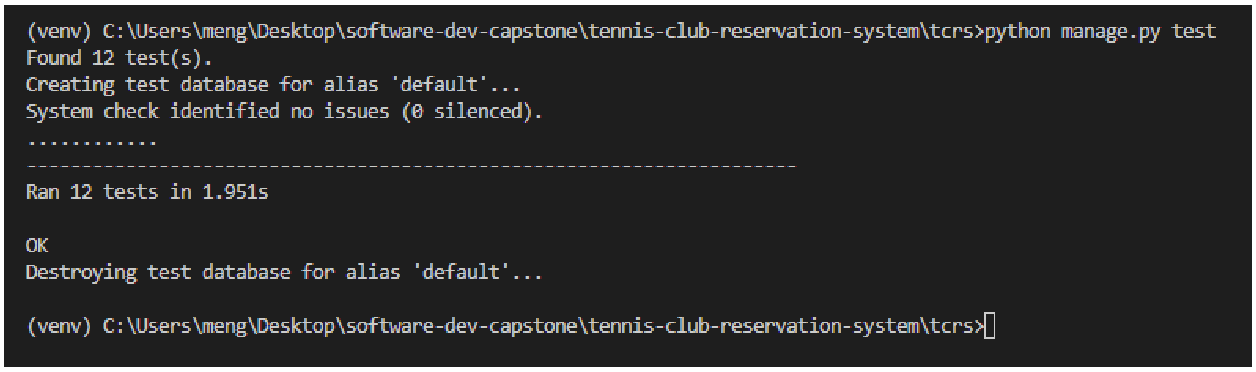
```python
def test_reservation_page_view_with_unauthenticated_user(self):
    response = self.client.get(self.reservation_url)
    self.assertRedirects(response, self.signup_url + '?next=' + self.reservation_url)
    self.assertEqual(response.status_code, 302)

def test_reservation_page_view_with_post_request(self):
    self.client.login(username='testuser', password='testpass')
    data = {
        'date': '2023-04-24',
        'time': '10:00',
        'court': '1',
        'number_of_players': '1',
        'number_of_guests': '0',
        'is_tournament': 'False'
    }
    response = self.client.post(self.reservation_url, data)
    self.assertEqual(response.status_code, 200)
    self.assertEqual(Reservation.objects.count(), 1)
    self.assertEqual(str(Reservation.objects.last().date), '2023-04-24')
    self.assertEqual(str(Reservation.objects.last().time), '10:00:00')
    self.assertEqual(str(Reservation.objects.last().court), '1')
    self.assertEqual(Reservation.objects.last().number_of_players, 1)
    self.assertEqual(Reservation.objects.last().number_of_guests, 0)

def test_reservation_page_view_with_post_request_and_reservation_conflict(self):
    self.client.login(username='testuser', password='testpass')
    data = {
        'date': '2023-04-24',
        'time': '10:00',
        'court': '1',
        'number_of_players': '1',
        'number_of_guests': '0',
        'is_tournament': 'False'
    }
    response = self.client.post(self.reservation_url, data)
    self.assertEqual(response.status_code, 200)
    self.assertTemplateUsed(response, 'reservations.html')
    self.assertTrue(response.context['reservation_failed'] == True)
    self.assertEqual(Reservation.objects.count(), 1)

def tearDown(self):
    self.client.logout()
```

Figure 5



```
(venv) C:\Users\meng\Desktop\software-dev-capstone\tennis-club-reservation-system\tcrs>python manage.py test
Found 12 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
............
----------------------------------------------------------------------
Ran 12 tests in 1.951s

OK
Destroying test database for alias 'default'...

(venv) C:\Users\meng\Desktop\software-dev-capstone\tennis-club-reservation-system\tcrs>
```