

```

class CourseTestCase(TestCase):
    def setUp(self):
        # create Course
        User.objects.create(username="It's me", first_name="best", last_name="jun")
        course = Course.objects.create(subject="Thai", subject_id="TH101", credit=3)
        Date.objects.create(subject_id=course, section="100001", start_time="9:30", end_time="12:30",
                             day="Monday", room="1024", year="2", semester="1", seat=1, status=True)

    def test_seat_available(self):
        course = Date.objects.first()

        self.assertTrue(course.is_seat_available())

    def test_seat_not_available(self):
        user1 = User.objects.first()

        student1 = Student.objects.create(name=user1)
        course = Date.objects.first()

        student1.course_enroll.add(course)
        course.seat -= 1

        self.assertFalse(course.is_seat_available())

class CourseUrlTestCase(TestCase):
    def setUp(self):
        User.objects.create(username="It's me", first_name="best", last_name="jun")
        course = Course.objects.create(subject="Thai", subject_id="TH101", credit=3)
        Date.objects.create(subject_id=course, section="100001", start_time="9:30", end_time="12:30",
                             day="Monday", room="1024", year="2", semester="1", seat=1, status=True)

        Student.objects.create(name=User.objects.first())

    def test_index_view_status_code(self):
        c = Client()
        response = c.get(reverse('Index'))
        self.assertEqual(response.status_code, 200)

    def test_index_view_context(self):
        c = Client()
        response = c.get(reverse('Index'))
        self.assertEqual(response.context['Courses'].count(), 1)

    def test_valid_course_page(self):
        c = Client()
        course = Course.objects.first()
        response = c.get(reverse('Course', args=(course.id,)))
        self.assertEqual(response.status_code, 200)

    def test_invalid_course_page(self):
        max_id = Course.objects.all().aggregate(Max("id"))['id_max']

        c = Client()
        response = c.get(f'Course/{max_id+1}')
        self.assertEqual(response.status_code, 404)

```

```

class CourseViewTest(TestCase):
    def setUp(self):
        User.objects.create(username="It's me", first_name="best", last_name="jun", password="Me123")
        course = Course.objects.create(subject="Thai", subject_id="TH101", credit=3)
        Date.objects.create(subject_id=course, section="100001", start_time="9:30", end_time="12:30",
                             day="Monday", room="1024", year="2", semester="1", seat=1, status=True)

    # test enroll method
    def test_enroll(self):
        student = Student.objects.first()

        course = Date.objects.first()
        course.seat = 1

        c = Client()
        c.post(reverse('enroll'), {'student': student.name, 'section': course.section, 'subject_id': course.subject_id.subject_id})

        self.assertEqual(Date.objects.get(subject_id=course.subject_id, section=course.section).seat, 0)

    # test del enroll method
    def test_del_enroll(self):
        student = Student.objects.first()

        course = Date.objects.first()
        course.seat = 0

        c = Client()
        c.post(reverse('del_enroll'), {'student': student.name, 'section': course.section, 'subject_id': course.subject_id.subject_id})

        self.assertEqual(Date.objects.get(subject_id=course.subject_id, section=course.section).seat, 1)

    # test enrolled view context
    def test_enrolled_view_context(self):
        student = Student.objects.first()
        student.course_enroll.add(Date.objects.first())
        self.client.force_login(student.name)
        response = self.client.post(reverse('enrolled'), {'student': student.name})
        self.assertEqual(response.context['Enrolled'].count(), 1)

    def test_search(self):
        student = Student.objects.first()
        self.client.force_login(student.name)
        searched = self.client.post(reverse('register'), {'student': student.name, 'searched': 'TH'}).context['searched']
        self.assertEqual(searched, 'TH')
        searched = self.client.post(reverse('register'), {'student': student.name, 'searched': 'SM'}).context['searched']
        self.assertEqual(searched, 'SM')

```

```

class TestUser(TestCase):
    def setUp(self):
        User.objects.create(username="It's me", first_name="best", last_name="jun", password="Me123")

    def test_index(self):
        self.client = Client()
        self.client.login(username='It's me', password='Me123')
        response = self.client.get(reverse('index'))
        self.assertEqual(response.status_code, 302)

    def test_login(self):
        self.client = Client()
        response = self.client.post(reverse('login'), {'username': 'It's me', 'password': 'Me123'})
        self.assertEqual(response.status_code, 200)

    def test_logout(self):
        self.client = Client()
        self.client.login(username='It's me', password='Me123')
        response = self.client.get(reverse('logout'))
        self.assertEqual(response.status_code, 200)

```

อธิบาย test

- setUp การสร้าง data ไว้สำหรับ test
- test seat available เช็คว่ามีที่เรียนว่างหรือไม่
- test seat unavailable เช็คว่าที่เรียนเต็มหรือไม่
- test index view status code เช็คว่าหน้า index ทำงานตามปกติไหมถูก redirect ถูกหรือไม่
- test index view context เช็คว่าจำนวน data course ใน query มีจำนวนถูกหรือไม่
- test valid course page เช็คว่ามีหน้า web page ของ course มีหรือไม่
- test invalid course page เช็คว่า course ถ้าไม่มี web page จะ response 404
- test enroll เช็คว่าสามารถลงเรียนในวิชานั้นได้หรือไม่
- test del enroll เช็คว่าสามารถถอนวิชาเรียนได้หรือไม่
- test enroll view context จะส่งจำนวนคอร์สเรียนที่ลงเรียนกลับมา
- test search เช็คว่าสามารถค้นหาคอร์สเรียนได้หรือไม่
- text index เช็คว่าหน้า index ของ user ถูกหรือไม่
- test login เช็คว่าสามารถ login ได้หรือไม่
- test logout เช็คว่าสามารถ logout ได้หรือไม่

test result

```

Found 13 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
-----
Ran 13 tests in 1.117s

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

```

Coverage report

Name	Stats	Miss	Cover
Course_reg__init__.py	0	0	100%
Course_reg\admin.py	5	0	100%
Course_reg\apps.py	4	0	100%
Course_reg\migrations\0001_initial.py	5	0	100%
Course_reg\migrations\0002_course_seat_date_section.py	5	0	100%
Course_reg\migrations\0003_date_subject_id_alter_date_section.py	5	0	100%
Course_reg\migrations\0004_course_users.py	5	0	100%
Course_reg\migrations\0005_date_room.py	4	0	100%
Course_reg\migrations\0006_rename_time_date_end_time_date_start_time.py	4	0	100%
Course_reg\migrations\0007_remove_course_users_student.py	6	0	100%
Course_reg\migrations\0008_alter_student_course_enroll_enrollment.py	5	0	100%
Course_reg\migrations\0009_alter_enrollment_course.py	5	0	100%
Course_reg\migrations\0010_delete_enrollment.py	4	0	100%
Course_reg\migrations\0011_enrollment.py	5	0	100%
Course_reg\migrations\0012_alter_student_course_enroll_delete_enrollment.py	4	0	100%
Course_reg\migrations\0013_remove_course_seat_date_seat.py	4	0	100%
Course_reg\migrations\0014_date_status.py	4	0	100%
Course_reg\migrations\0015_date_semester_date_year.py	4	0	100%
Course_reg\migrations__init__.py	0	0	100%
Course_reg\models.py	36	2	94%
Course_reg\test_models.py	18	0	100%
Course_reg\test_urls.py	29	0	100%
Course_reg\tests.py	1	0	100%
Course_reg\urls.py	4	0	100%
Course_reg\views.py	8	0	100%
Course_registrations__init__.py	0	0	100%
Course_registrations\asgi.py	4	4	0%
Course_registrations\settings.py	23	0	100%
Course_registrations\urls.py	4	0	100%
Course_registrations\wsgi.py	4	4	0%
manage.py	12	2	83%
users__init__.py	0	0	100%
users\admin.py	1	0	100%
users\apps.py	4	0	100%
users\migrations__init__.py	0	0	100%
users\models.py	1	0	100%
users\test_views.py	54	0	100%
users\tests.py	1	0	100%
users\urls.py	4	0	100%
users\views.py	63	9	86%
TOTAL	349	21	94%