

CS350 Milestone 3 - Testing

Evan Duffield

CRUD Testing (tests.py)

- Created methods for Creating, Getting, Updating, and Deleting users for our current user data model.
- Each test is independent, a new individual is generated for the database every time.

```
✓ class UserTests(TestCase):  
>     def test_CreateUser(self): ...  
  
✓     def test_ReadUser(self):  
✓         user = AppUser.objects.create(  
            userName='getuser',  
            firstName='Alice',  
            lastName='Smith',  
            email='getuser@example.com',  
            birthDate='1995-10-20',  
            location='Townsville',  
            profilePicture='profile2.jpg',  
            isLoggedIn=True,  
        )  
  
        retrieved_user = AppUser.objects.get(userName='getuser')  
  
        self.assertEqual(retrieved_user.userName, 'getuser')  
        self.assertEqual(retrieved_user.email, 'getuser@example.com')  
  
>     def test_UpdateUser(self): ...  
  
>     def test_DeleteUser(self): ...
```

Unit Testing (tests.py)

- Created way of adding users based off of BACS350 console commands.
- Had chatgpt write out 20 different test cases based off my code, each method goes through different potential extremes of input for creating a User.

```
class DatabaseUnitTests(TestCase):
> def test_CreateUser1(self): ...
> def test_CreateUser2(self): ...
> def test_CreateUser3(self): ...
> def test_CreateUser4(self): ...
> def test_CreateUser5(self): ...
> def test_CreateUser2(self): ...
> def test_CreateUser3(self):
    user_data = {
        'userName': 'user3',
        'firstName': 'Eva',
        'lastName': 'Brown',
        'email': 'eva@example.com',
        'birthDate': '2001-03-30',
        'location': 'Villageville',
        'profilePicture': 'profile3.jpg',
    }
    user = AppUser.objects.create(**user_data)
    self.assertIsNotNone(user)
    created_user = AppUser.objects.get(userName='user3')
    self.assertIsNotNone(created_user)
    self.assertEqual(created_user.userName, 'user3')
    self.assertEqual(created_user.email, 'user3@example.com')
> def test_CreateUser4(self): ...
> def test_CreateUser5(self): ...
```

View Testing (tests.py)

- Simple client testing to see if standard views are properly served to user.
- First page is always launched, all other methods test views that are only returned when a database is populated.

```
Report.md tests.py X
Application > helloapp > tests.py > UserViewTests
553
554 class UserViewTests(TestCase):
555     def test_HelloPage(self):
556         response = self.client.get("helloapp/")
557         self.assertEqual(response.status_code, 200)
558
559     def test_UserListView(self):
560         response = self.client.get("user/list.html")
561         self.assertEqual(response.status_code, 200)
562
563     def test_UserDetailView(self):
564         response = self.client.get("user/detail.html")
565         self.assertEqual(response.status_code, 200)
566
567     def test_UserCreateView(self):
568         response = self.client.get("user/add.html")
569         self.assertEqual(response.status_code, 200)
570
571     def test_UserUpdateView(self):
572         response = self.client.get("user/edit.html")
573         self.assertEqual(response.status_code, 200)
574
575     def test_UserDeleteView(self):
576         response = self.client.get("user/delete.html")
577         self.assertEqual(response.status_code, 200)
```

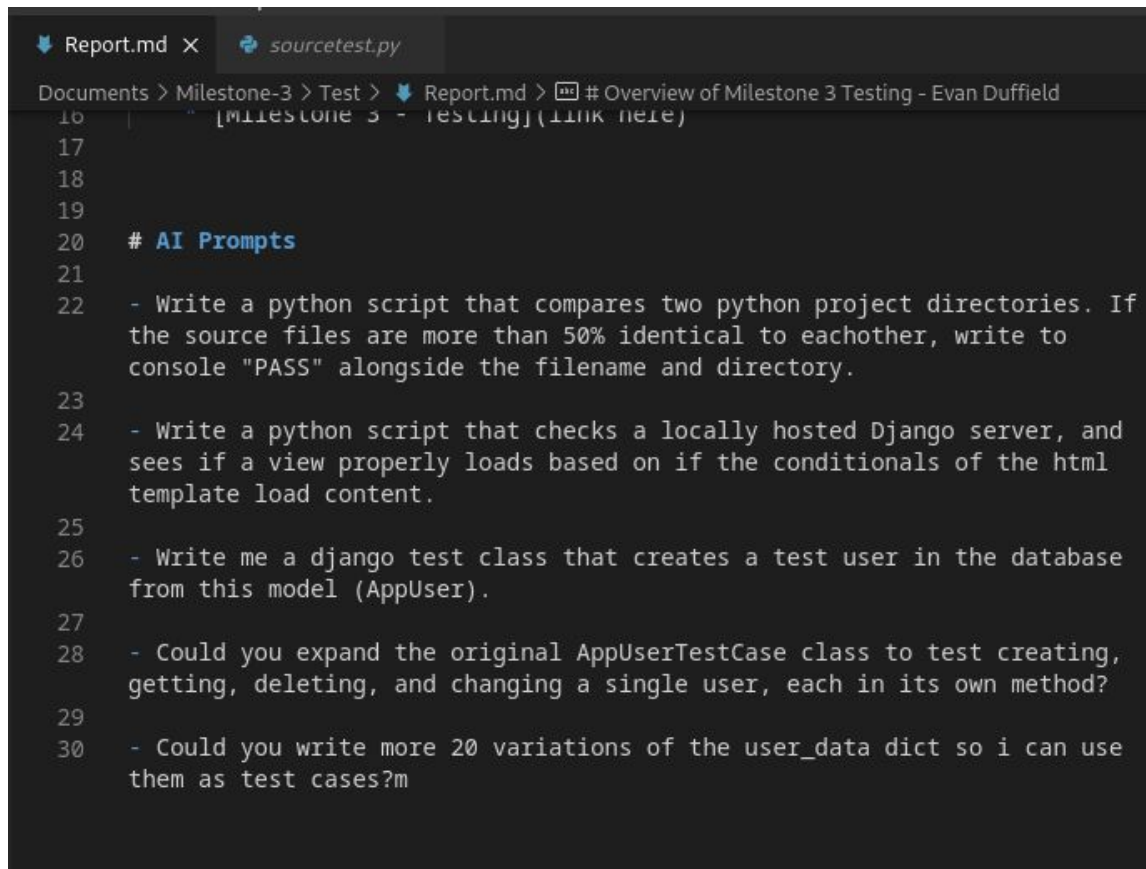
Test-driven development (Milestone-3/Test)

- Simple script that tests how many lines are from tests.py out of the rest of the project.

```
def count_lines_in_file(file_path):  
    with open(file_path, 'r') as file:  
        return len(file.readlines())  
  
def main(project_directory):  
    source_file_count = 0  
    source_file_lines = 0  
  
    for root, dirs, files in os.walk(project_directory):  
        for file in files:  
            if file.endswith(".py"):  
                source_file_count += 1  
                source_file_lines += count_lines_in_file(os.path.join(root, file))  
  
    tests_file = os.path.join(project_directory, 'tests.py')  
  
    if source_file_count == 0:  
        return False # No source files in the project  
  
    if not os.path.exists(tests_file):  
        return False # No tests.py file in the project  
  
    tests_lines = count_lines_in_file(tests_file)  
  
    return tests_lines >= 0.5 * (source_file_lines / source_file_count)
```

Markdown Updates (Milestone-3/Test)

- Updated Test directory in Milestone-3.
- Filled out a markdown update file Report.md alongside adding sourcetest.py.
- AI Prompts are included in the markdown file.



The screenshot shows a code editor with two tabs: 'Report.md' and 'sourcetest.py'. The 'Report.md' tab is active, displaying a markdown document. The document's path is 'Documents > Milestone-3 > Test > Report.md', and its title is '# Overview of Milestone 3 Testing - Evan Duffield'. The content includes a link to 'Milestone 3 - Testing (link here)' and a section titled '# AI Prompts'. This section contains five bullet points, each preceded by a hyphen and a line number (22-30). The prompts are: 1. Write a python script that compares two python project directories. If the source files are more than 50% identical to each other, write to console "PASS" alongside the filename and directory. 2. Write a python script that checks a locally hosted Django server, and sees if a view properly loads based on if the conditionals of the html template load content. 3. Write me a django test class that creates a test user in the database from this model (AppUser). 4. Could you expand the original AppUserTestCase class to test creating, getting, deleting, and changing a single user, each in its own method? 5. Could you write more 20 variations of the user_data dict so i can use them as test cases?m

```
16 | "Milestone 3 - Testing"(link here)
17
18
19
20 # AI Prompts
21
22 - Write a python script that compares two python project directories. If
the source files are more than 50% identical to each other, write to
console "PASS" alongside the filename and directory.
23
24 - Write a python script that checks a locally hosted Django server, and
sees if a view properly loads based on if the conditionals of the html
template load content.
25
26 - Write me a django test class that creates a test user in the database
from this model (AppUser).
27
28 - Could you expand the original AppUserTestCase class to test creating,
getting, deleting, and changing a single user, each in its own method?
29
30 - Could you write more 20 variations of the user_data dict so i can use
them as test cases?m
```

Thanks for watching!