DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

IT314 - SOFTWARE ENGINEERING

STATIC TESTING

CRIME AND HAZARD MANAGEMENT SYSTEM

GROUP NO: 30

GROUP MEMBERS

202001410	PATEL AYUSH SANJAYBHAI
202001447	VAKANI HETAV ABHAYBHAI
202001467	JAY GROVER
202001421	PATEL KUNJ RAKESH
202001446	GONDALIYA VENIL CHANDUBHAI
202001275	AYUSH JAIN
202001458	KRIS PATEL
202001264	HARSH SANJAY MAKWANA
202001466	KALP KINJALBHAI PANDYA
202001438	NARODIA JEET NILESHKUMAR
202001457	HITARTH VYAS

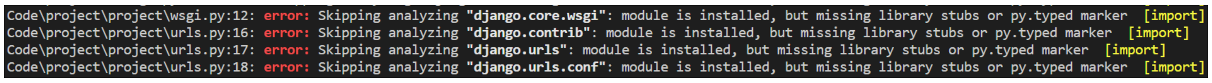
Static Testing

Static testing is a testing approach that helps identify issues with code before it is run. It involves analyzing the code statically, i.e., without executing it, to find errors and problems that may impact the functionality of the code. The main objective of static testing is to ensure the quality of the code and prevent potential issues from occurring during runtime.

In our case, we utilized two powerful Python development tools, mypy and pylint, for static testing. Mypy is a static type checker that helps in detecting errors before code execution. It examines the code for type discrepancies and possible runtime issues, ensuring that the code is well-typed and less prone to errors. Mypy employs type annotations to determine the type of variables, function arguments, and return values, making it easier to identify possible errors before runtime. By detecting errors early, mypy enhances the quality of the code and aids developers in avoiding runtime mistakes.

Testing using mypy

During the process of testing with mypy, we encountered an error related to the imported modules in a particular file that were not being utilized in the code. This type of error can often go unnoticed and can lead to unnecessary bloat in the code. By catching this error early in the testing phase, we were able to address it before it became a bigger issue.



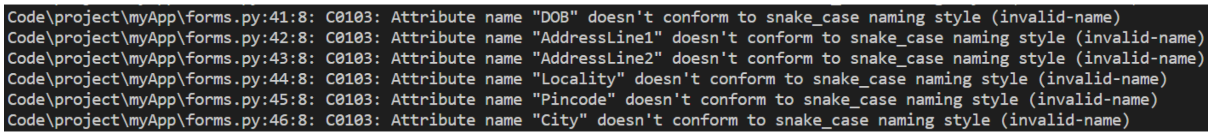
```
Code\project\project\wsgi.py:12: error: Skipping analyzing "django.core.wsgi": module is installed, but missing library stubs or py.typed marker  [import]
Code\project\project\urls.py:16: error: Skipping analyzing "django.contrib": module is installed, but missing library stubs or py.typed marker  [import]
Code\project\project\urls.py:17: error: Skipping analyzing "django.urls": module is installed, but missing library stubs or py.typed marker  [import]
Code\project\project\urls.py:18: error: Skipping analyzing "django.urls.conf": module is installed, but missing library stubs or py.typed marker  [import]
```

Detailed test log for mypy testing is available in static_test_mypy.txt.

Testing using Pylint

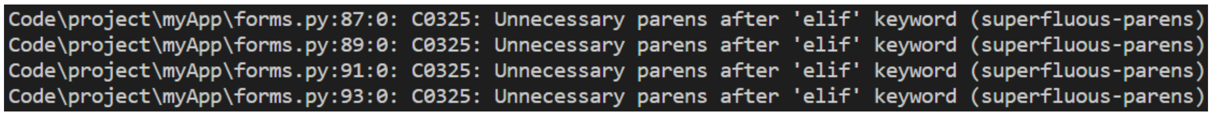
Pylint, on the other hand, is a powerful Python linter that scans code for conventions, mistakes, and probable flaws. It examines the code for consistency in naming patterns, style, and syntax issues, and evaluates the code in accordance with a set of preset guidelines. Pylint offers suggestions for code improvements, making it easier for developers to improve the quality of their code. By using Pylint, developers can ensure that their code complies with coding standards and is consistent with other code in the project.

During our testing with Pylint, we identified that our code was not following the naming convention of using snake case, which Pylint is designed to enforce. Instead, we had implemented a camel case in our code. However, it is important to note that using different case conventions is not technically incorrect and will not impact the functionality of our code.



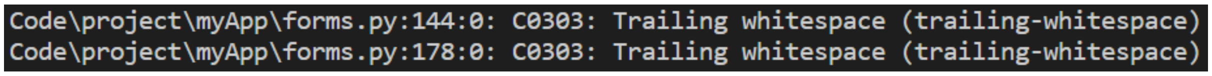
```
Code\project\myApp\forms.py:41:8: C0103: Attribute name "DOB" doesn't conform to snake_case naming style (invalid-name)
Code\project\myApp\forms.py:42:8: C0103: Attribute name "AddressLine1" doesn't conform to snake_case naming style (invalid-name)
Code\project\myApp\forms.py:43:8: C0103: Attribute name "AddressLine2" doesn't conform to snake_case naming style (invalid-name)
Code\project\myApp\forms.py:44:8: C0103: Attribute name "Locality" doesn't conform to snake_case naming style (invalid-name)
Code\project\myApp\forms.py:45:8: C0103: Attribute name "Pincode" doesn't conform to snake_case naming style (invalid-name)
Code\project\myApp\forms.py:46:8: C0103: Attribute name "City" doesn't conform to snake_case naming style (invalid-name)
```

Pylint identified that parentheses were used after an if else statement in the Python code. Although this is not required in Python, Pylint has flagged it as an error. It's important to note that even though this is an error according to Pylint, the code will still run without any issues.



```
Code\project\myApp\forms.py:87:0: C0325: Unnecessary parens after 'elif' keyword (superfluous-parens)
Code\project\myApp\forms.py:89:0: C0325: Unnecessary parens after 'elif' keyword (superfluous-parens)
Code\project\myApp\forms.py:91:0: C0325: Unnecessary parens after 'elif' keyword (superfluous-parens)
Code\project\myApp\forms.py:93:0: C0325: Unnecessary parens after 'elif' keyword (superfluous-parens)
```

Additionally, Pylint also identified errors related to extra spacing and modules that were imported but not used in our code. These types of errors can have a negative impact on the quality of the code and should be addressed. While these errors were flagged by Pylint, they did not directly affect the final execution of our code.



```
Code\project\myApp\forms.py:144:0: C0303: Trailing whitespace (trailing-whitespace)
Code\project\myApp\forms.py:178:0: C0303: Trailing whitespace (trailing-whitespace)
```

From our code, pylint rated our code with a rating 5.04 out of 10.

Detailed test log for pylint testing is available in static_test_pylint.txt.

By addressing these issues during the testing phase, we were able to improve the overall quality of our code and reduce the risk of errors occurring during runtime.