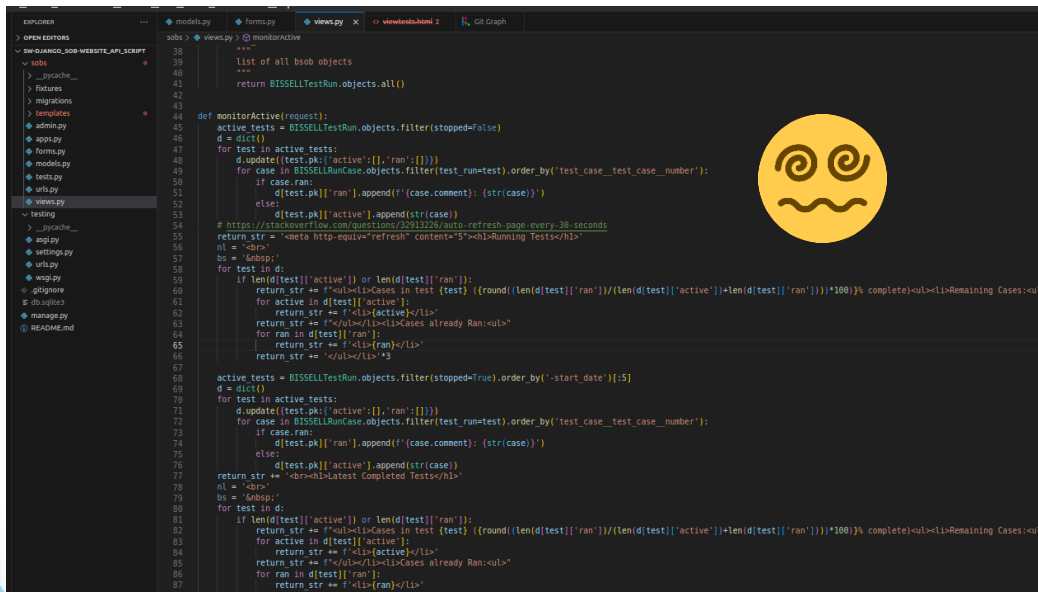


# IoT Team's Sprint Testing Problems

- Running tests by hand requires working on one phone and one durable at a time
- Resetting the app by hand every time following a test case gets extremely tedious
- Team is working on automated test cases, but there is no good way to store and view data collected from the tests
- Current Demo for GUI makes you input each test case by hand for each test run and is hard to look through
- Team is new to Django and code for the current GUI is difficult to understand and hard to scale up



The image shows a screenshot of a code editor displaying a Django project's views.py file. The code is written in Python and includes a function named monitorActive. The function is responsible for monitoring active tests and updating the test status. It uses Django's ORM to interact with the database. The code is somewhat complex and difficult to read, which is highlighted by a yellow emoji with a confused face (🤔) overlaid on the right side of the code block. The emoji has a yellow face with two large, swirling eyes and a wavy mouth, indicating confusion or difficulty understanding the code.

```
38 """
39 list of all bsob objects
40 """
41 return BISSELLTestRun.objects.all()
42
43
44 def monitorActive(request):
45     active_tests = BISSELLTestRun.objects.filter(stopped=False)
46     d = dict()
47     for test in active_tests:
48         d.update(test.pk: {'active': [], 'ran': []})
49         for case in BISSELLRunCase.objects.filter(test_run=test).order_by('test_case__test_case_number'):
50             if case.ran:
51                 d[test.pk]['ran'].append('({case.comment}): {str(case)}')
52             else:
53                 d[test.pk]['active'].append(str(case))
54         # https://stackoverflow.com/questions/3293326/auto-refresh-page-every-30-seconds
55     return_str = '<meta http-equiv="refresh" content="5"><h1>Running Tests</h1>'
56     nl = '<br>'
57     bs = '<br>'
58     for test in d:
59         if len(d[test]['active']) or len(d[test]['ran']):
60             return_str += f"<ul><li>Cases in test {test} ((round((len(d[test]['ran'])/(len(d[test]['active'])+len(d[test]['ran']))) * 100) % complete)<ul><li>Remaining Cases</li>"
61             for active in d[test]['active']:
62                 return_str += f"<li>{active}</li>"
63             return_str += f"</ul></li><li>Cases already Ran</li>"
64             for ran in d[test]['ran']:
65                 return_str += f"<li>{ran}</li>"
66             return_str += "</ul></li>" * 3
67     active_tests = BISSELLTestRun.objects.filter(stopped=True).order_by('-start_date')[:5]
68     d = dict()
69     for test in active_tests:
70         d.update(test.pk: {'active': [], 'ran': []})
71         for case in BISSELLRunCase.objects.filter(test_run=test).order_by('test_case__test_case_number'):
72             if case.ran:
73                 d[test.pk]['ran'].append('({case.comment}): {str(case)}')
74             else:
75                 d[test.pk]['active'].append(str(case))
76     return_str += "<br><h1>Latest Completed Tests</h1>"
77     nl = '<br>'
78     bs = '<br>'
79     for test in d:
80         if len(d[test]['active']) or len(d[test]['ran']):
81             return_str += f"<ul><li>Cases in test {test} ((round((len(d[test]['ran'])/(len(d[test]['active'])+len(d[test]['ran']))) * 100) % complete)<ul><li>Remaining Cases</li>"
82             for active in d[test]['active']:
83                 return_str += f"<li>{active}</li>"
84             return_str += f"</ul></li><li>Cases already Ran</li>"
85             for ran in d[test]['ran']:
86                 return_str += f"<li>{ran}</li>"
87             return_str += "</ul></li>" * 3
```

# Needs and Solutions

---

- Storing past test runs to allow for comparing to newer ones
  - Store past test runs in an SQLite database and display them in an organized manner through use of html and JavaScript formatting
- A way to make test runs easily replicable
  - Create “Profiles” for test runs to load up and run with buttons to select more than one test case at a time
- Code that allows for growth
  - Rewrite current GUI to allow for more scalable additions in the future by simplifying the code and making it less specific
- A way to access and change specific parts of test cases
  - An intuitive GUI that allows you to reach any kind of data you need to look at with Code that won't break when you try and change things