Matlab Usage Statistics

Akshay Khadse Raghav Gupta

Indian Institute of Technology, Bombay

November 20, 2016

Overview

Plotly

Features

- Interactive Plotting Library
- Supports bar, line, scatter, maps and many more
- Supports online, offline plotting
- Allows styling in Python

Why Plotly?

- Offline plotting is easy
- Allows export to HTML div tag
- No other javascript dependencies

LDAP3

- LDAP operations are clumsy
- `python-ldap` module works with Python2
- `ldap3` developed in Python3 native, works for both
- Simplified Query Construction
- Allows secured access
- Conforms to current standards

Django

Features

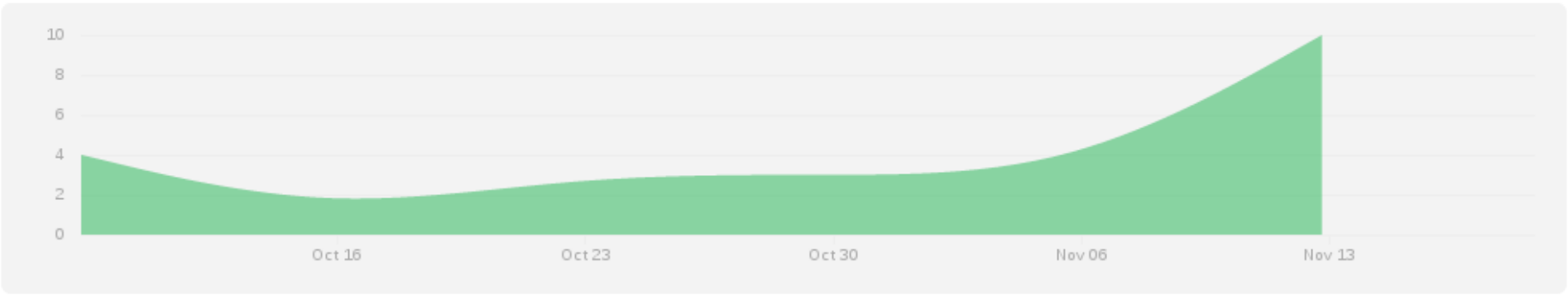
- Django is a high-level Python Web framework
- Takes care of much of the hassle
  eg. authentication, site maps, RSS feeds etc
- Fast, Secure and Scalable
- Shortucts for common tasks

Why Django?

- Project has institute level scope
- Reusable apps can be used by other projects easily
- Good learning opportunity

Git

https://www.github.com/akshaykhadse/matlab-usage-stats/



Continous Integration

Travis CI

- Installs Dependencies
- Runs tests
- Checks PEP8

Coveralls

- 99% coverage
- Tried to test every bit of code authored by us
- Django setup code ignored
- Non Runnable code ignored

Documentation

- Docs hosted at `https://matlab-usage-stats.readthedocs.io/`
- Based on Sphinx, ReST
- Provided Docstrings for models, views and forms
- Non app processing functions also included

Tests

- Used unittests and mock
- **Models**
  - Tested Object Creation
  - Object saving and querying
- **Views**
  - Tested Context
  - Tested response content
- **django.test.Client** allows makeing requests like browser
- **Forms**
  - Not Tested
  - Testing views covers forms

How to use?

- Download zip from repo
- `$ python3 setup.py install`
- Move log files into `data` folder in project root
- `$ python3 manage.py runserver`

List View

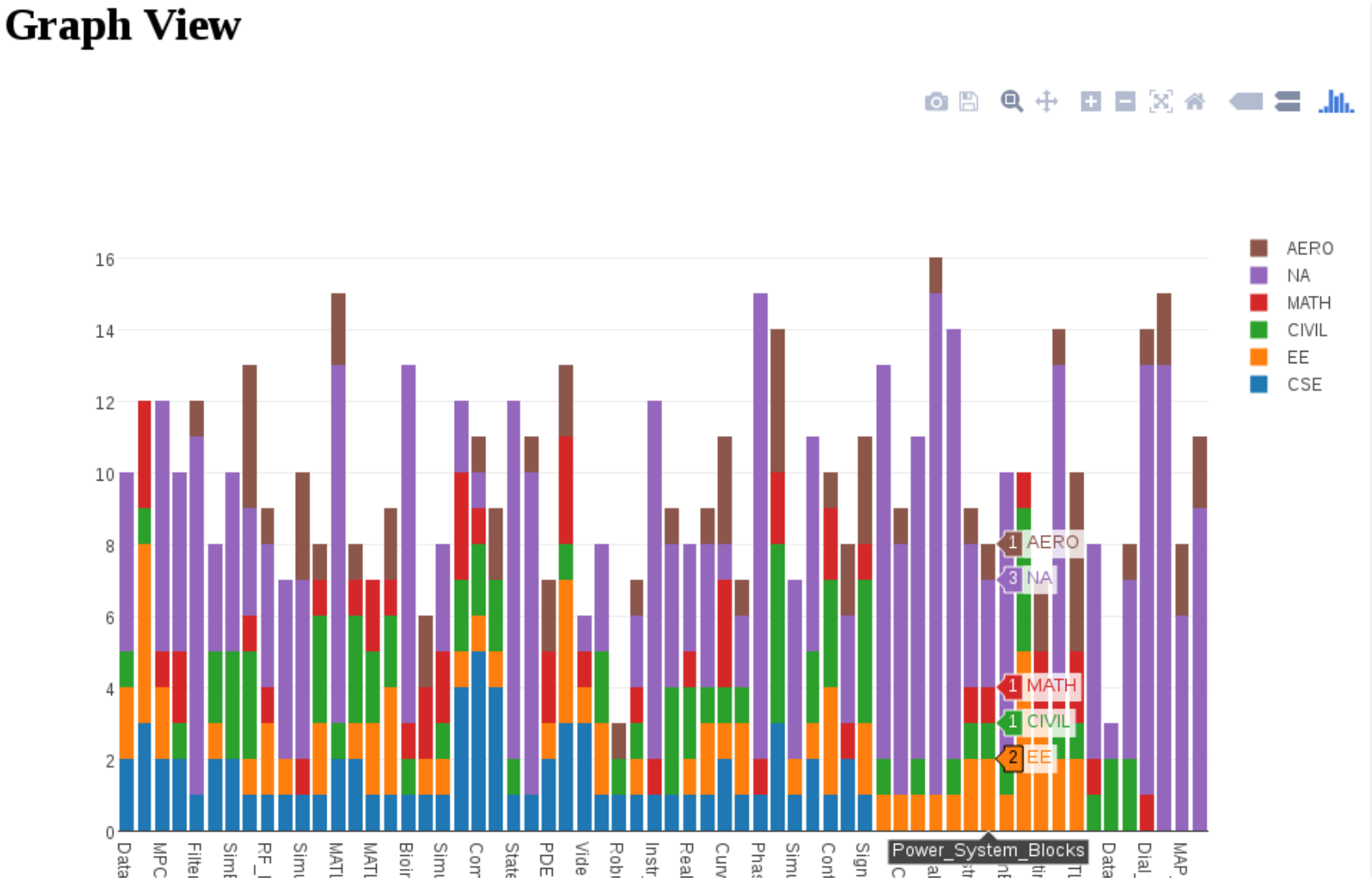
On request, all entries are fetched from DB rendered on template for table and sent back as response.

Matlab Usage Report

Out Time	Package	User	Roll No	Department	Type	In Time
14:40:31	Distrib_Computing_Toolbox	xxxxx	xxxxx	MATH	pg	15:38:56
14:40:41	SimMechanics	xxxxx	xxxxx	CIVIL	pg	14:53:10
14:40:49	Real-Time_Workshop	xxxxx	xxxxx	MATH	pg	15:25:30
14:40:55	MBC_Toolbox	xxxxx	xxxxx	EE	dd	15:18:00
14:41:01	Statistics_Toolbox	xxxxx	xxxxx	MATH	pg	15:28:43
14:41:10	MATLAB_Coder	xxxxx	xxxxx	CSE	dd	15:19:29
14:41:16	Statistics_Toolbox	xxxxx	xxxxx	MATH	pg	15:34:33
14:41:21	Identification_Toolbox	xxxxx	xxxxx	CIVIL	pg	14:53:39
14:41:31	Fixed_Point_Toolbox	xxxxx	xxxxx	AERO	pg	14:57:13
14:41:36	OPC_Toolbox	xxxxx	xxxxx	AERO	pg	15:17:04
14:41:45	Stateflow	xxxxx	xxxxx	CIVIL	pg	15:25:50
14:41:54	Fixed_Point_Toolbox	xxxxx	xxxxx	CSE	pg	15:18:30
14:42:02	Instr_Control_Toolbox	xxxxx	xxxxx	MATH	pg	15:28:44
14:42:07	MATLAB_Distrib_Comp_Engine	xxxxx	xxxxx	MATH	pg	15:22:14
14:42:15	Fuzzy_Toolbox	xxxxx	xxxxx	AERO	pg	15:25:47

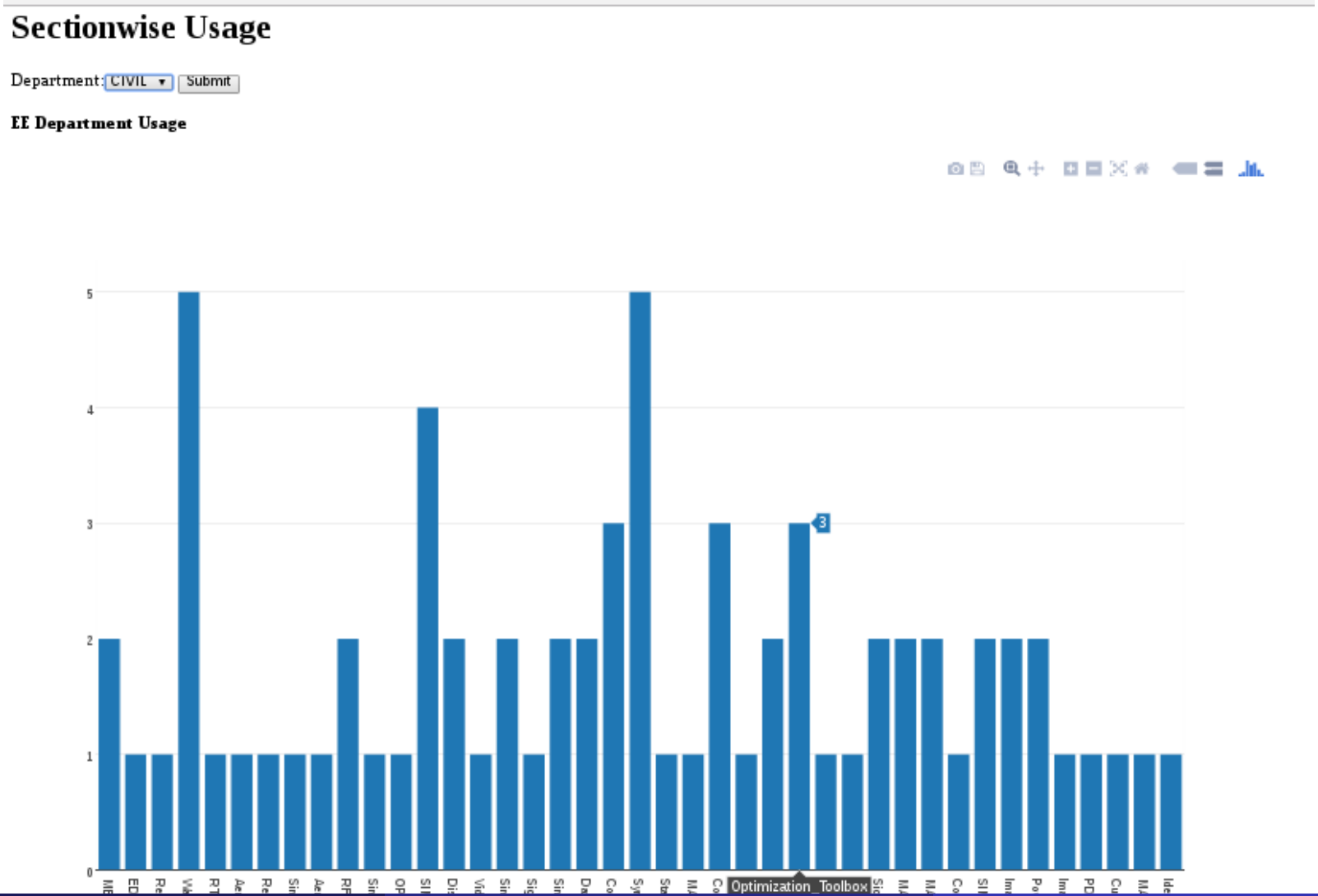
Graph View

On request, all entries are fetched from DB, plotly graph is generated and sent back as response.



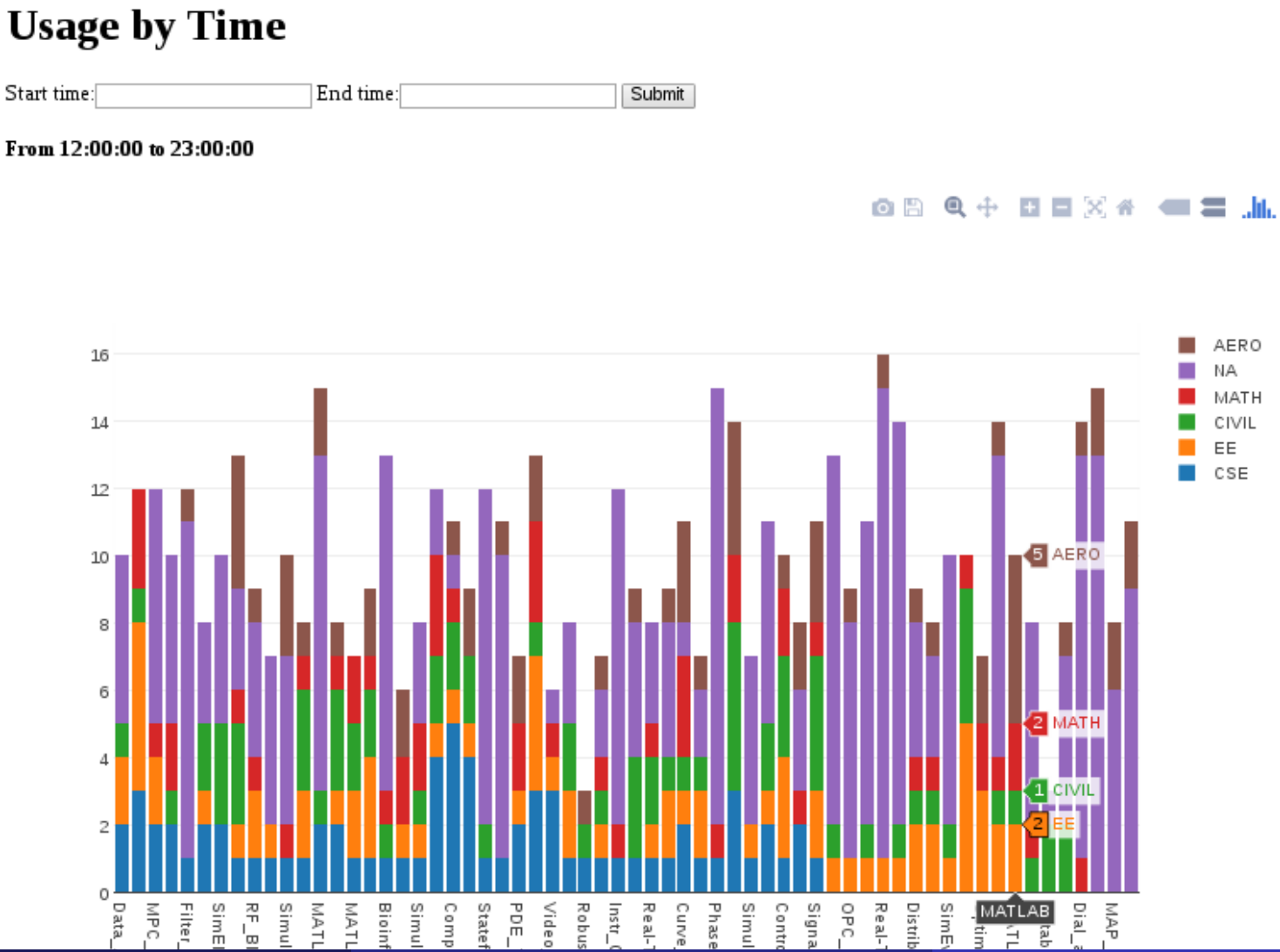
Departments View

If form is submitted, all entries from selected department are fetched from DB, plotly graph is formed and sent back as response.



Time View

If form is submitted, all entries from selected time frame are fetched from DB, plotly graph is formed and sent back as response.



Issues

Excluding from Coverage

- `coverage run --rcfile=.coveragerc manage.py test parser reports`
- `.coveragerc`

```
omit =
    */__init__.py
    reports/apps.py
exclude_lines =
    if __name__ == .__main__.:
```

Issues

Testing LDAP

- Testing `ldap_search` by mock

```
@mock.patch('parser.ldap_search.Connection')
class Test_ldap_search(TestCase):
    def test_ldap_search(self, mock_connection):
        ...
        expected = [mock.call('ldap.iitb.ac.in',
                              auto_bind=True),
                    mock.call().search(basedn, query,
                                       attributes=attrs)]
        mock_connection.assert_has_calls(expected)
```

Future Scope

- **Synchronous Queue Processing**
  Process of updating database from logs is manual
  This can be periodically automated with RabbitMQ and Celery
- **Revamping Interface**
  Present web interface is plain html.
  Ignored due to out of subject's scope.
  Templating using Bootstrap would take very little effort
- **Extension to other license servers**
  Other could be included based on similar strategy.