



# CA400 Final Year Project

comet | User Manual

<b>Students</b>	<b>Kevin McGonigle</b> 16318486 <a href="mailto:kevin.mcgonigle2@mail.dcu.ie">kevin.mcgonigle2@mail.dcu.ie</a>  <b>James Miles</b> 16349533 <a href="mailto:james.miles2@mail.dcu.ie">james.miles2@mail.dcu.ie</a>
<b>Supervisor</b>	<b>Dr. Geoff Hamilton</b> <a href="mailto:geoffrey.hamilton@dcu.ie">geoffrey.hamilton@dcu.ie</a>
<b>Date</b>	16/05/2020
<b>Git Repo</b>	<a href="https://gitlab.computing.dcu.ie/mcgonik2/2020-ca400-mcgonik2-milesj2">https://gitlab.computing.dcu.ie/mcgonik2/2020-ca400-mcgonik2-milesj2</a>

# 0. Introduction

This document provides the steps and information required to install, set-up and use the comet web-based API and front-end web application. It is imperative that installation instructions are adhered to strictly and in order, as any divergence to the instructions may result in unexpected and undesired behaviour. This installation guide assumes the user is deploying both the server and web application to their own localhost. Deployment to services such as AWS may vary slightly. It also assumes basic command line knowledge.

## 1. Installation

### 1.1 Prerequisites

- Python 3.7+
  - Pip package-management system
- Node.js
  - npm package manager
- ANTLR v4 parser generator tool<sup>1</sup>
  - Requires java version 1.6+

### 1.2 Obtaining the Source Code

The source code for comet is hosted on GitLab at <https://gitlab.com/computing.dcu.ie/mcgonik2/2020-ca400-mcgonik2-milesj2>. Cloning this repository will download all of the necessary materials for installing and running both the comet API and web application.

### 1.3 Installing Dependencies

The next step in installation is to install the dependencies for both the Python-based API and Node.js web application.

#### 1.3.1 Python Dependencies

All external package dependencies necessary for running the Django web server are stored in the requirements.txt file at src/server. To install, open the command line, navigate to this directory and simply run the following command:

```
pip install -r requirements.txt
```

---

<sup>1</sup> ANTLR Getting Started Guide, <https://github.com/antlr/antlr4/blob/master/doc/getting-started.md>

### 1.3.2 Node.js Dependencies

Similarly, the package dependencies for the Node.js web application are contained in a file called package.json located at src/app. Change directory to this location and execute the following command:

```
npm install
```

It is possible that the installed packages will need auditing. Npm will inform you of this, usually suggesting that the following command be executed to resolve issues:

```
npm audit fix
```

## 1.4 ANTLR Parser Generation

The next step is to use ANTLR v4 to generate the base parser code for each of the supported grammars. To do this, ANTLR v4 must be set up in accordance with the Getting Started guide provided by ANTLR.

### 1.4.1 Python Parser

In the command-line, change directory to src/server/metrics/grammars. In this directory is a file call Python3.g4. This is the Python 3 ANTLR grammar and is what will be used to create the base classes for lexing and parsing Python code. To do this, run the following command, ensuring that the output directory, denoted by the -o flag, is set correctly such that files will be created in the src/server/metrics/parsers/python3/base directory.

```
antlr4 -o ../parsers/python3/base -DLanguage=Python3  
Python3.g4
```

### 1.4.2 C# Parser

Similarly, change directory to src/server/metrics/grammars/CSharp. This directory contains three separate grammar files, each of which will be used at once to generate the necessary resources for generating C# parse trees. Again, we wish the output directory to be set accordingly to src/server/metrics/parsers/csharp as follows:

```
antlr4 -o ../parsers/csharp/base -DLanguage=Python3 *.g4
```

## 1.5 Django Database Setup

Django is a data-driven web application framework, storing its many models and views seamlessly in a database provider or the user's choice. The data storage requirements for comet are relatively small and, as such, it is suitable for comet to rely purely on the default sqlite database natively provided by Django by default. To

set up the database, the database must be created and migrated to be up-to-date with the latest versions of comet's models. To do this, change directory to src/server and run the following command:

**python manage.py migrate**

You will then see all the various migrations beginning to be applied.

```
C:\Users\Mames\Desktop\2020-ca400-mcgonik2-milesj2\src\server>py manage.py migrate
Operations to perform:
  Apply all migrations: admin, api, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying api.0001_initial... OK
  Applying api.0002_auto_20200211_1037... OK
  Applying api.0003_auto_20200225_1227... OK
  Applying api.0004_class_method... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying sessions.0001_initial... OK
```

## 1.6 Django Settings

Django provides support for a multitude of different settings to support all manner of different aspects relating to maintaining and serving web applications. This is all handled in the Django settings module. Comet has two versions of these settings, depending on the user's intentions, and therefore the correct settings file to use must be specified using the DJANGO\_SETTINGS\_MODULE environment variable. On Windows, this can be set using the command:

```
set DJANGO_SETTINGS_MODULE=server.settings.production
or
set DJANGO_SETTINGS_MODULE=server.settings.development
```

With the UNIX equivalent being:

```
export DJANGO_SETTINGS_MODULE=server.settings.production
or
export DJANGO_SETTINGS_MODULE=server.settings.development
```

Additionally, in order for Django to run, comet requires a secret key to be established by setting the DJANGO\_SECRET\_KEY environment variable in a similar manner, ensuring that the chosen key is secure and not shared.

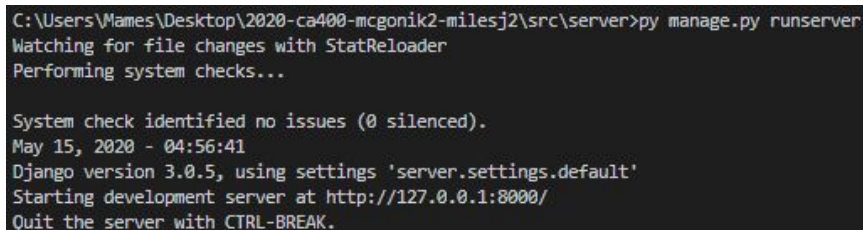
## 2. Running the Application

It is imperative to Comet's functionality that the server is running prior to the front-end. This is due to the front-end being dependent on the data served from the back-end.

### 2.1 Running the Server

To run the server, ensure you are pointing towards the src/server directory, and run the command:

```
python manage.py runserver
```



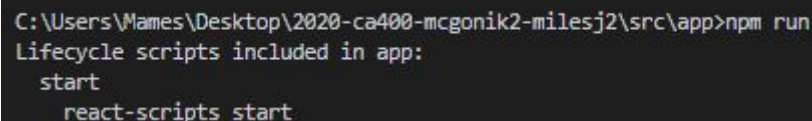
```
C:\Users\Mames\Desktop\2020-ca400-mcgonik2-milesj2\src\server>py manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 15, 2020 - 04:56:41
Django version 3.0.5, using settings 'server.settings.default'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

### 2.1 Running the Web Application

To run the web application, ensure you are located in the src/app directory, and run the command:

```
npm start
```



```
C:\Users\Mames\Desktop\2020-ca400-mcgonik2-milesj2\src\app>npm run
Lifecycle scripts included in app:
  start
    react-scripts start
```

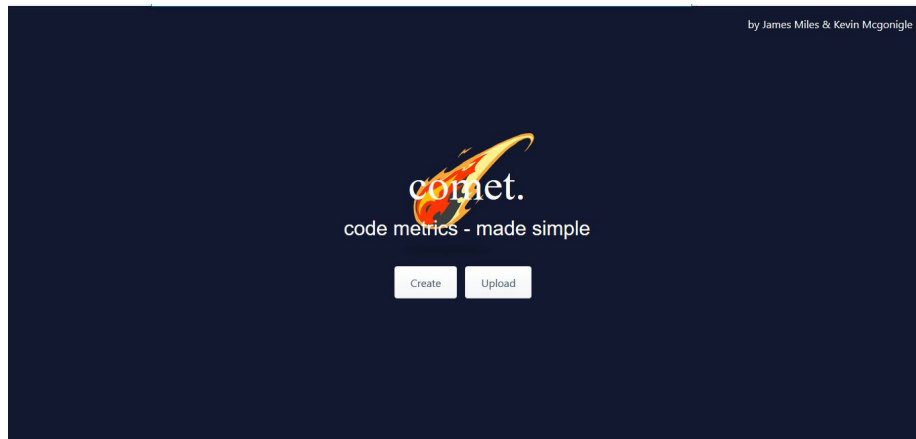
## 3. Navigating the Web Application

### 3.1 Prelude

This section is a demonstration and tutorial on using the web application. The below instructions are performed using the following python code snippet:

## 3.2 Navigation to Homepage

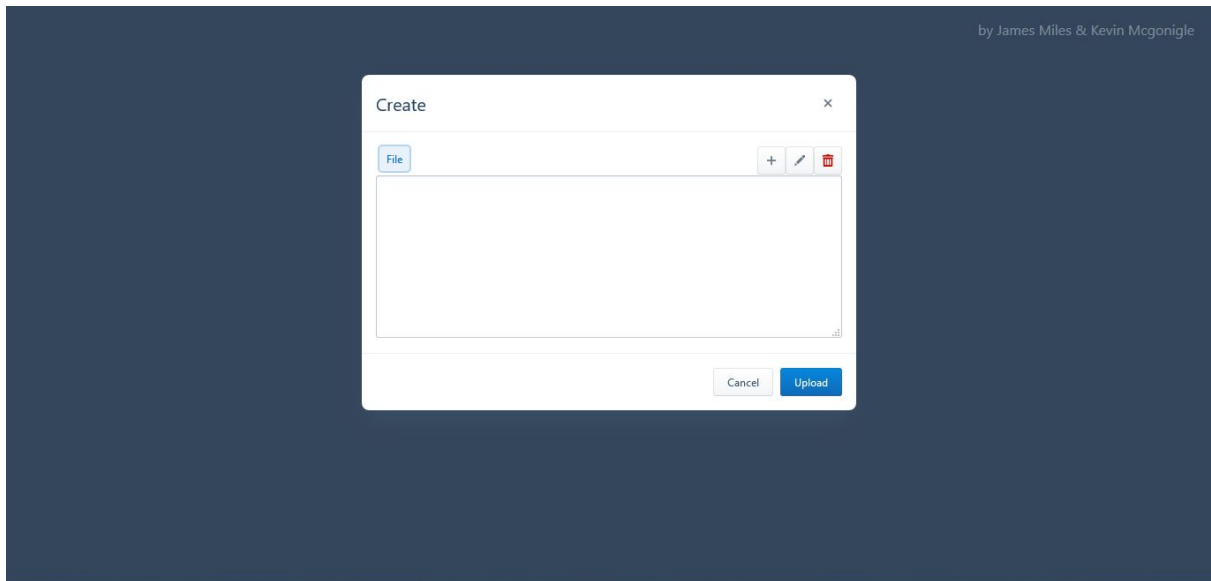
Once both the server and web application's dependencies have been installed and both are running, navigate to <http://localhost:3000/>.



## 3.3 Uploading a file

It is advised to ensure the successful running of the file on your own system to ensure that the following instructions are executed optimally.

When the create button is clicked, the create modal will appear. A sample file will be shown in the textarea, this will be replaced by the above code snippet. The file will also be renamed from the base name given by double clicking the name within the tab and renaming it. Upon the clicking of the create button shown within the modal, there are two possible events that can occur; the file will be successfully parsed and return metrics or the parsing of the file will generate an error and thus the information required cannot be generated. The latter indicates there are likely syntax errors within your given code.

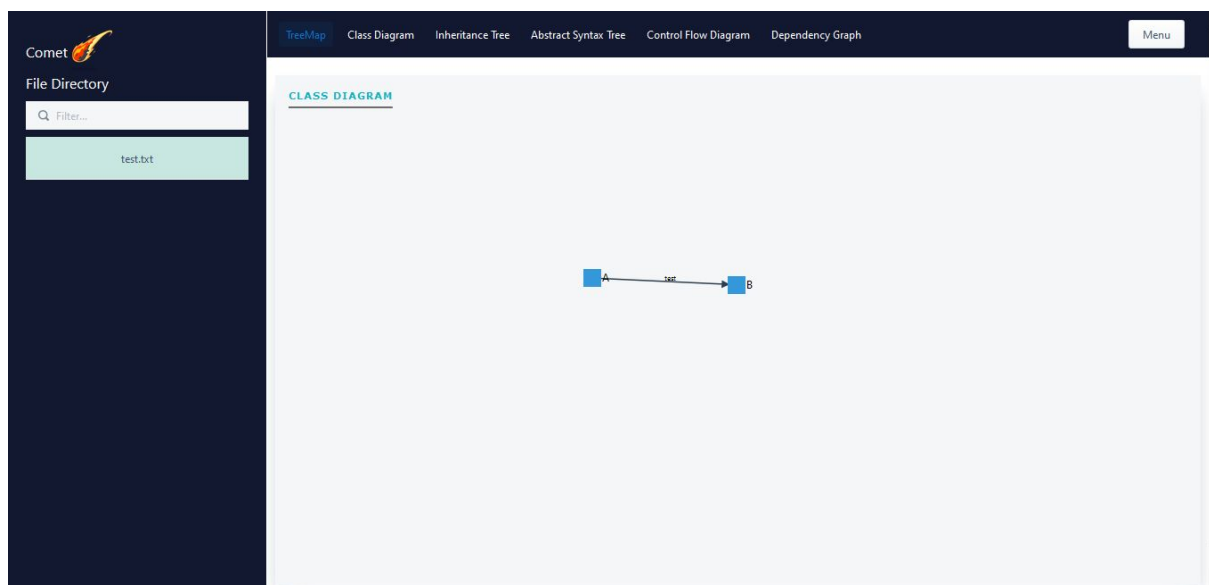


## 3.4 Metrics Page

Upon a successful upload and file parse, there is a redirection to the metrics page. The page now displays a file directory of the uploaded file/s, a toolbar with a tab menu and a pane displaying the corresponding metric representation of the uploaded file.

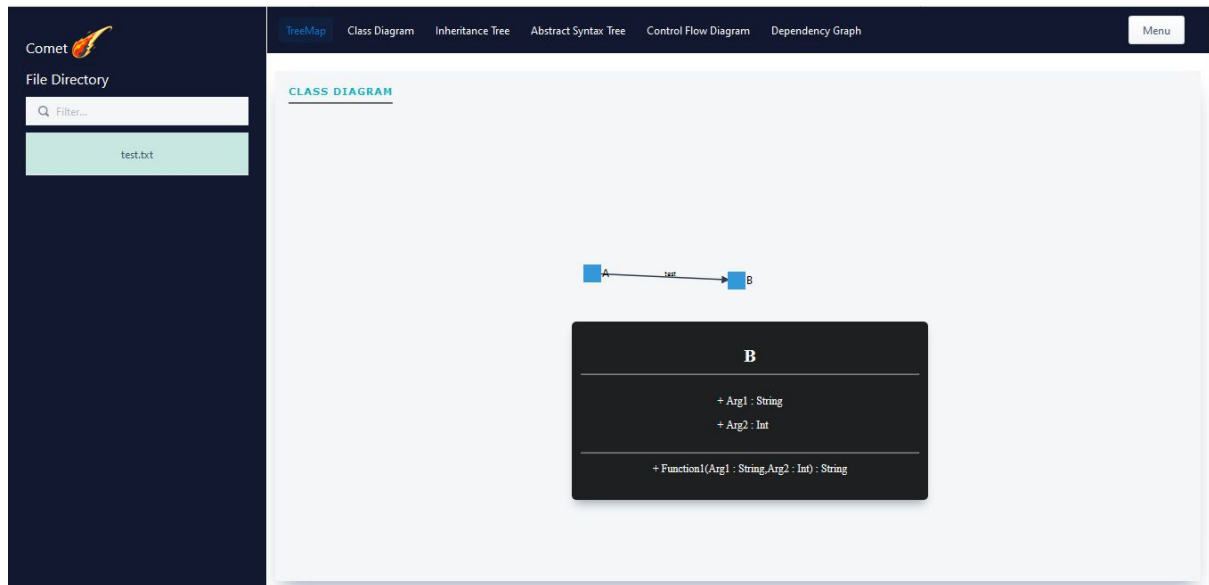
### 3.4.1 Class Diagram

The tab initially begins at the Class Diagram tab and is seen to be highlighted. A pane displaying the class diagram of the selected file.

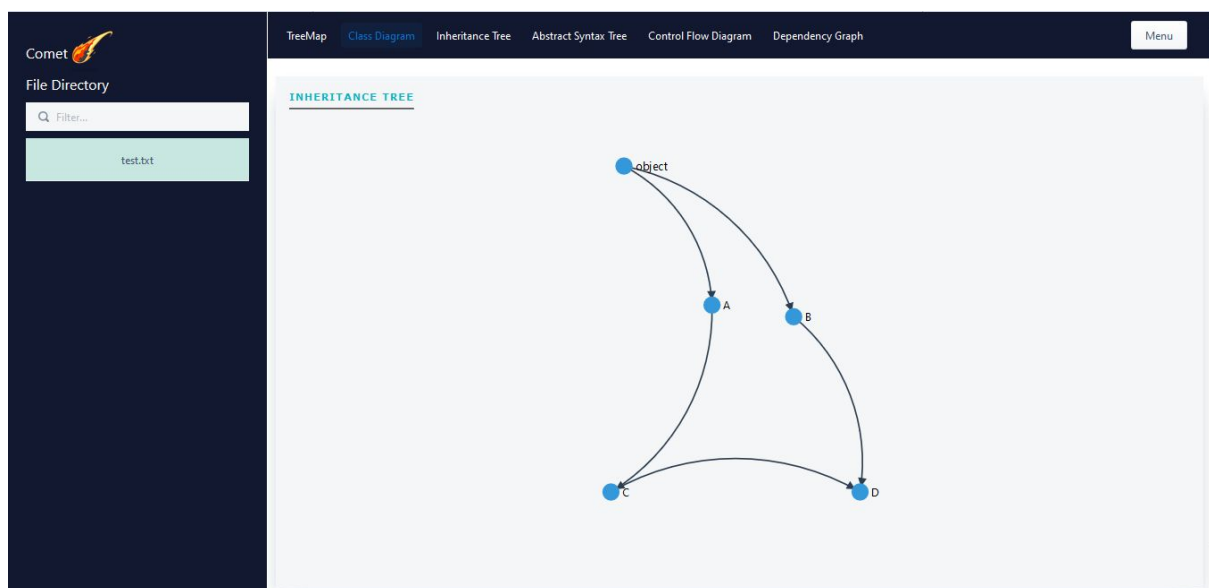


### 3.4.2 Class Diagram Information

Clicking a graph node will reveal a modal displaying a class name, the arguments and functions within the class and their corresponding types.

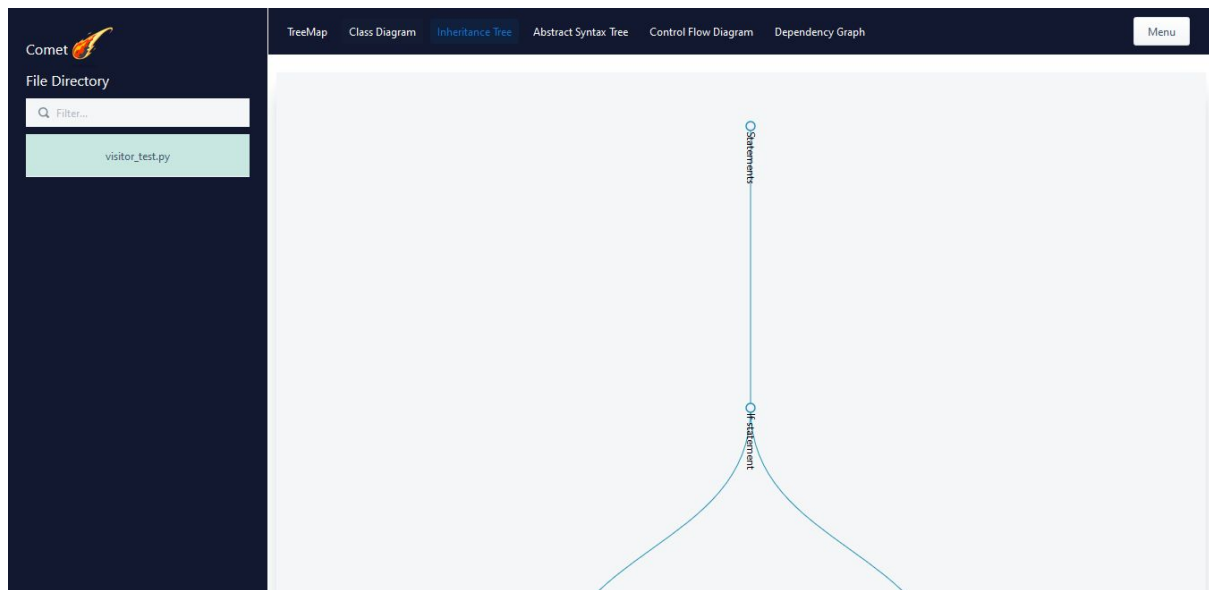


### 3.4.3 Inheritance Tree

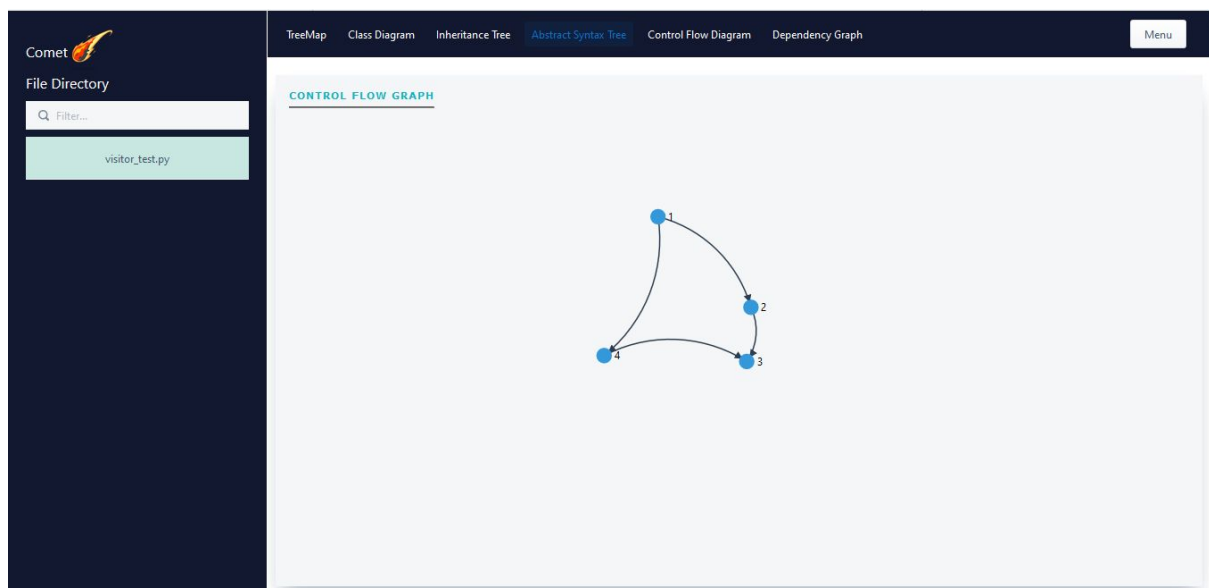




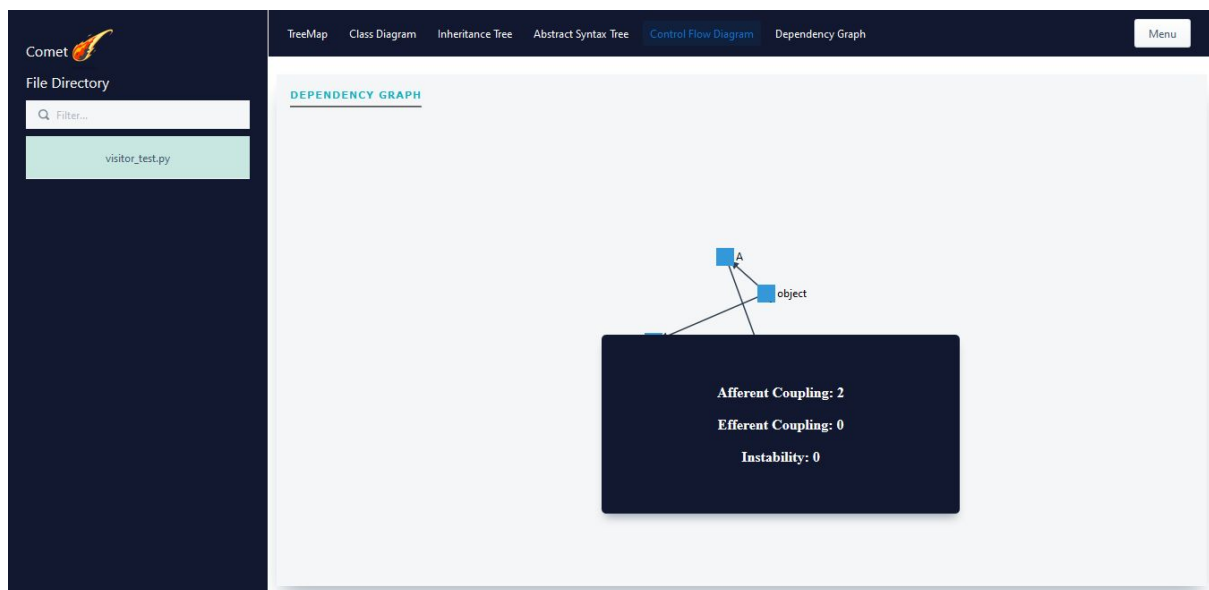
### 3.4.4 Abstract Syntax Tree



### 3.4.5 Dependency Graph



### 3.4.5.1 Afferent Coupling, Efferent Coupling & Instability



## 4. Tests

### 4.1 Back-End Tests

Backend testing is performed using Python's built-in unit testing framework, PyUnit, in conjunction with the dependency mocking library that comes with it. Many of the structures and visitors implemented by comet have accompanying test cases which can be run using the python console or via the command line. Testing suites may also be set up to execute groups of tests simultaneously.

### 4.2 Front-End Tests

#### 4.2.1 Unit Tests

To execute all tests together, enter the first command. The addition of a third argument to this command executes the tests contained within the specified test file.

```
npm test
```

#### 4.2.2 Coverage Report

To produce a coverage report, enter the below command.

```
npm test --coverage --watchAll=false
```

Upon the completion and generation of the coverage report, navigate to and open the file found at the path: `src/coverage/lloc-report/index.html`.

### 4.2.3 Integration Testing

An integral component to improve the health of the system. The following is the command that visibly simulates the actions a user would take to use the system's features, via Puppeteer.

```
npm test e2e
```