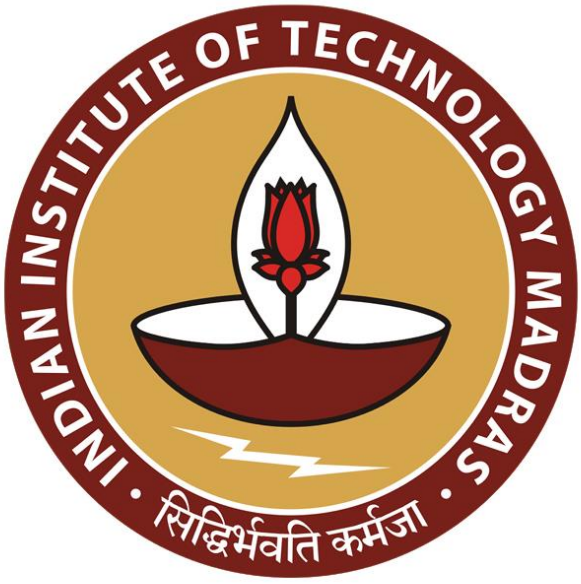
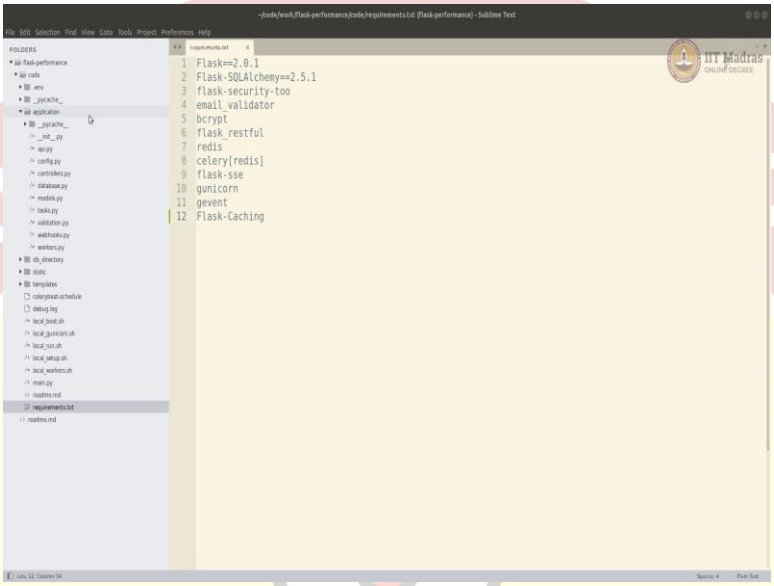


IIT Madras

ONLINE DEGREE

Modern Application Development-II
Software Consultant - Thejesh G N
Bachelor of Science in Programming and Data Science
Indian Institute of Technology, Madras
Performance Measurement and Caching

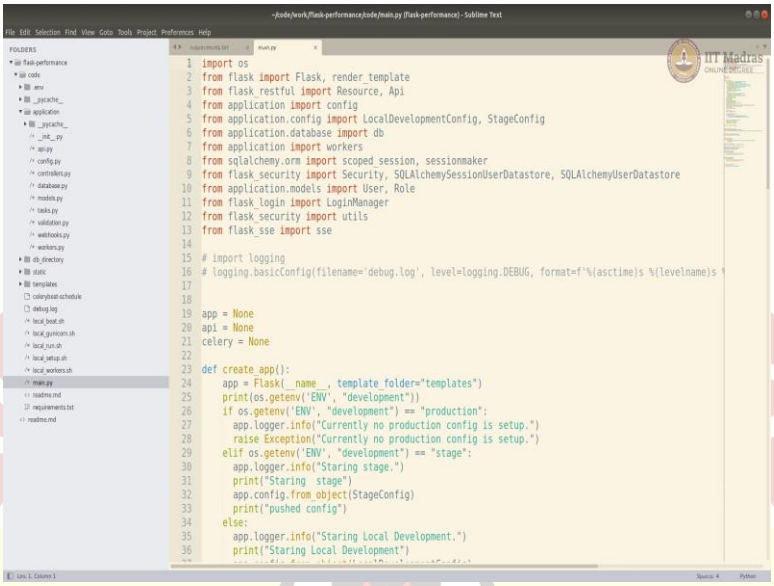
(Refer Slide Time: 00:16)



Welcome to the Modern Application Development Two Screencast. In this screencast, we will see how to measure and improve performance. For this, you will need a Linux environment, you will also need Redis. So, please install and run Redis. Now, performance measurement and improvement is a vast subject.

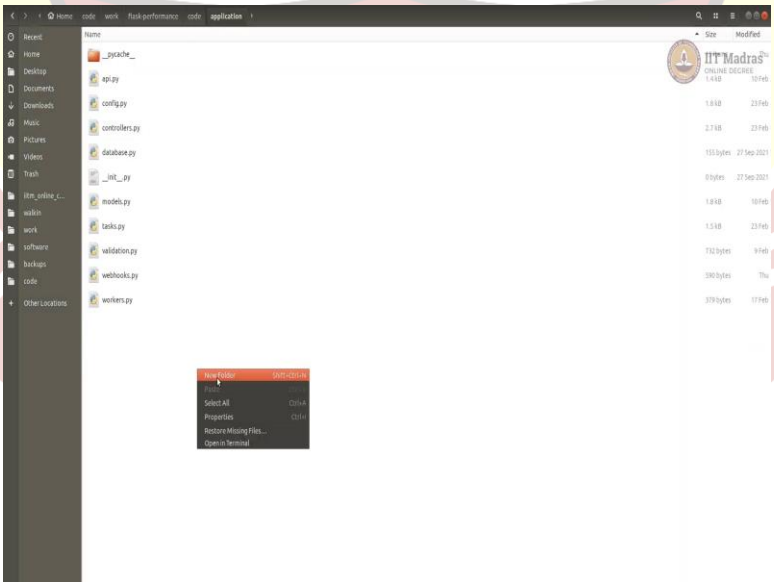
There are many areas of application in which a bottleneck might appear. But we will see some of the common scenarios. And we will also see how to mitigate them or improve the performance in such scenarios. Now, before we start, let us look at the application that we have. We have an application with various files inside the application folder.

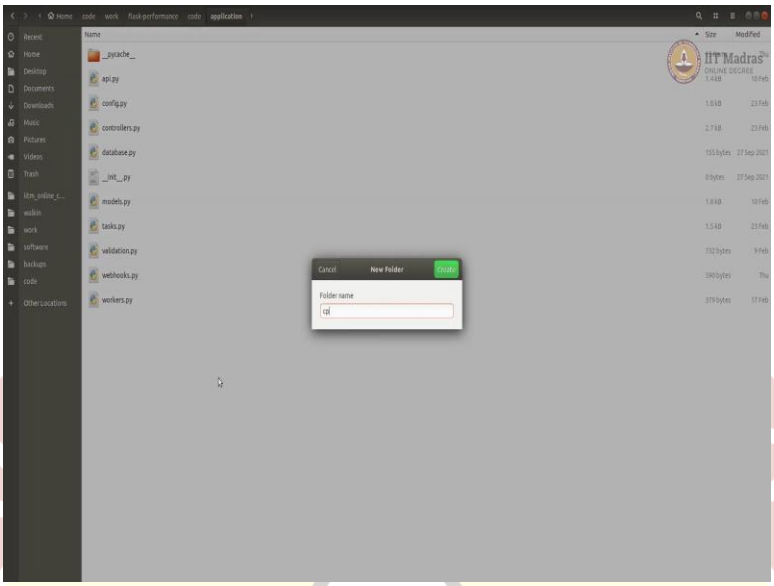
(Refer Slide Time: 01:04)

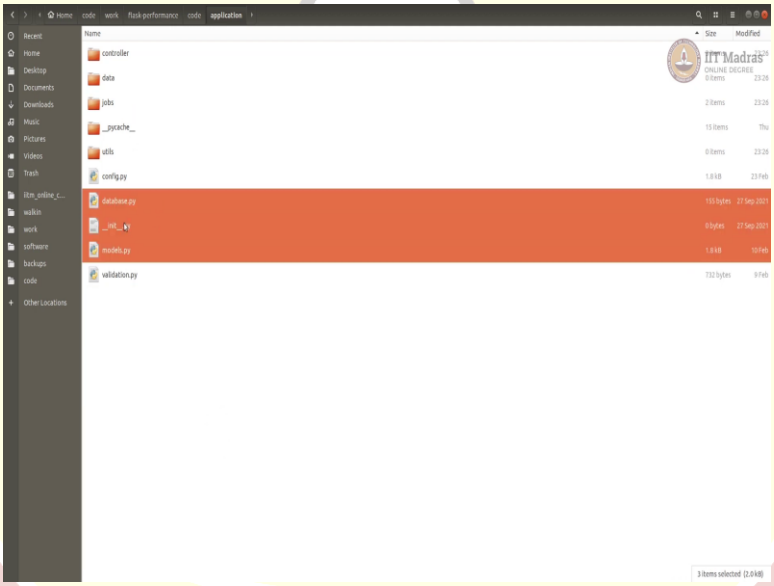


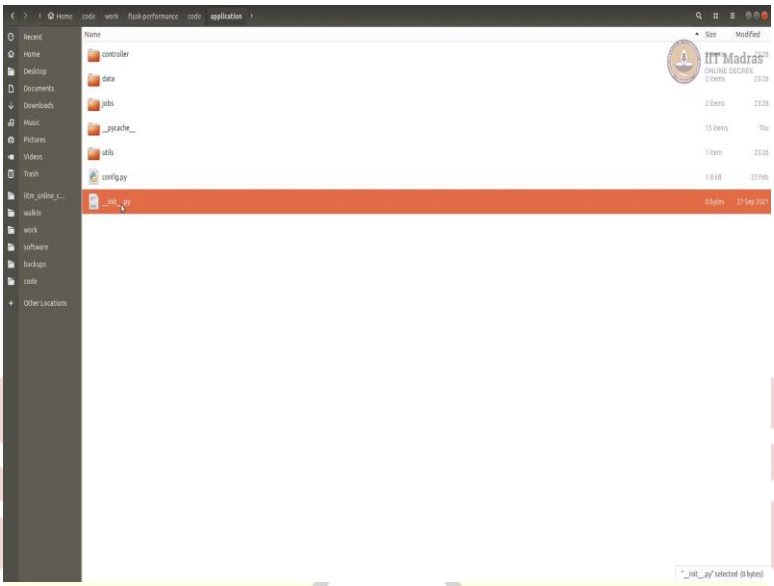
And our main app is in the main. Now before we continue, I want to organize it a little bit better into different modules. So that, going through it becomes easier and managing it becomes easier. Now you can skip it forward if you do not watch it. But it is quicker, just go ahead and do it in front of you.

(Refer Slide Time: 01:27)



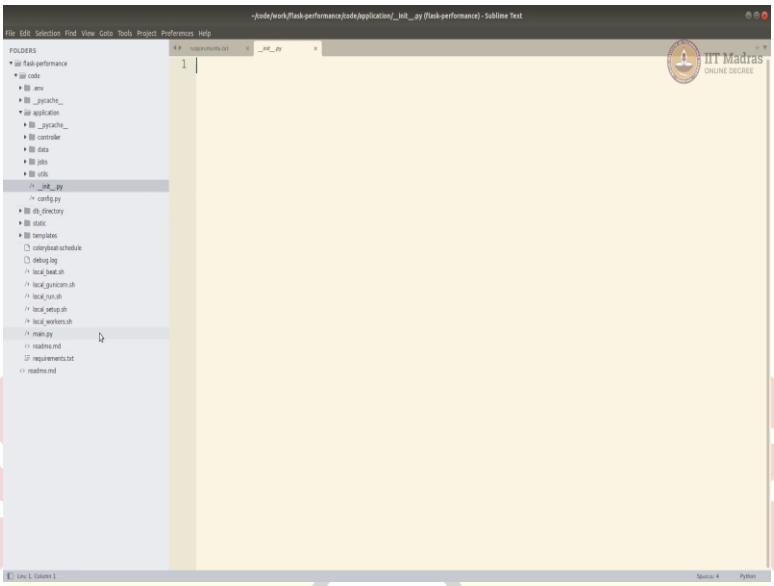

Home
code
work
flask-performance
code
application
Name
Size
Modified
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
__pycache__
api.py
config.py
controllers.py
database.py
__init__.py
models.py
tasks.py
validation.py
webhooks.py
workers.py
Cancel
New Folder
Create
Folder name
cpi


Home
code
work
flask-performance
code
application
Name
Size
Modified
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
controller
data
jobs
__pycache__
utils
config.py
database.py
__init__.py
models.py
validation.py
3 items selected (2.6 kB)



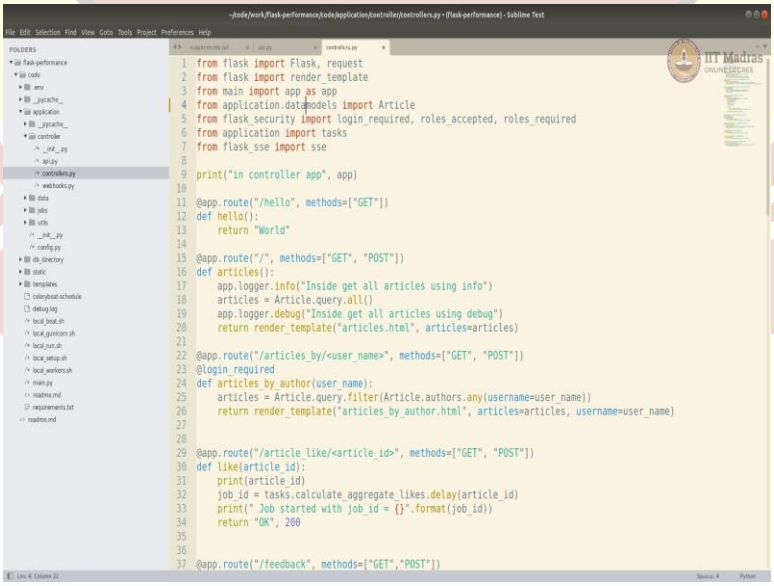
When in the application folder, I am going to create 4 new folders is called controller, and then a folder called data. A folder called a jobs and the folder called utils. Now I am going to move API controller and web books into a controller and workers tasks into Jobs folder, and database and models into data folder and validation into utils folder. Now since they are all modules or packages, I need to add a underscore init file.

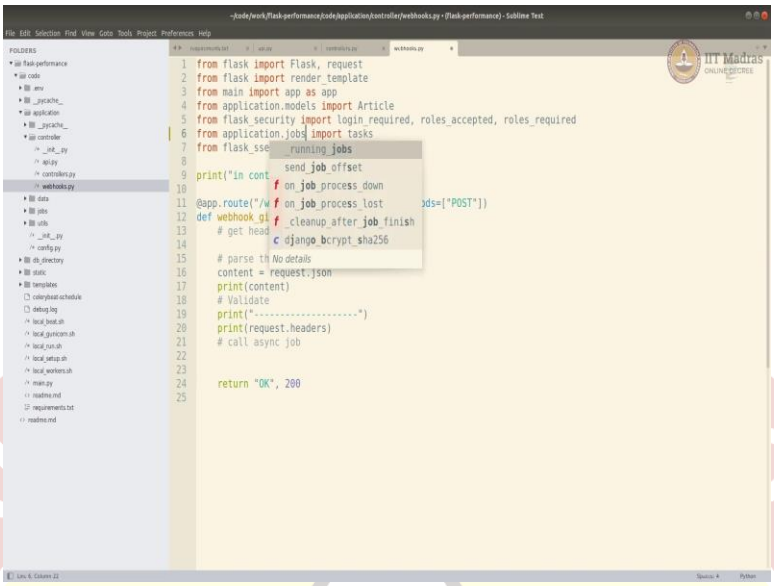
(Refer Slide Time: 02:28)

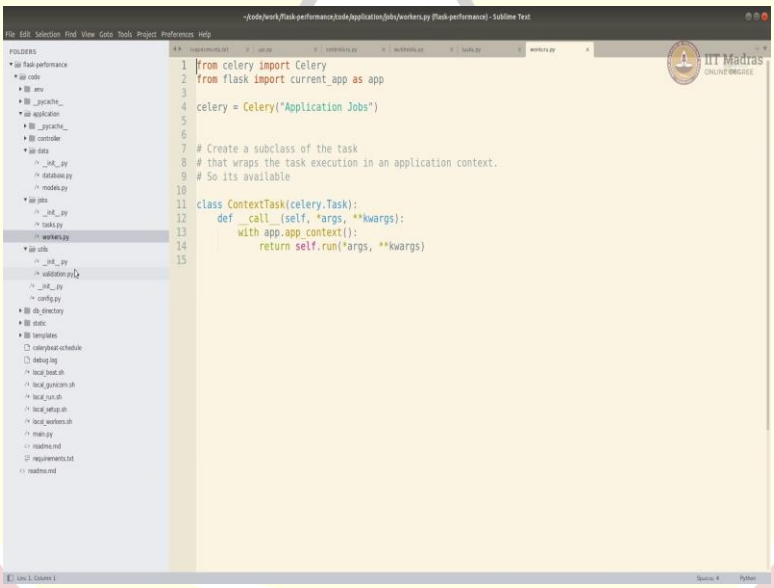


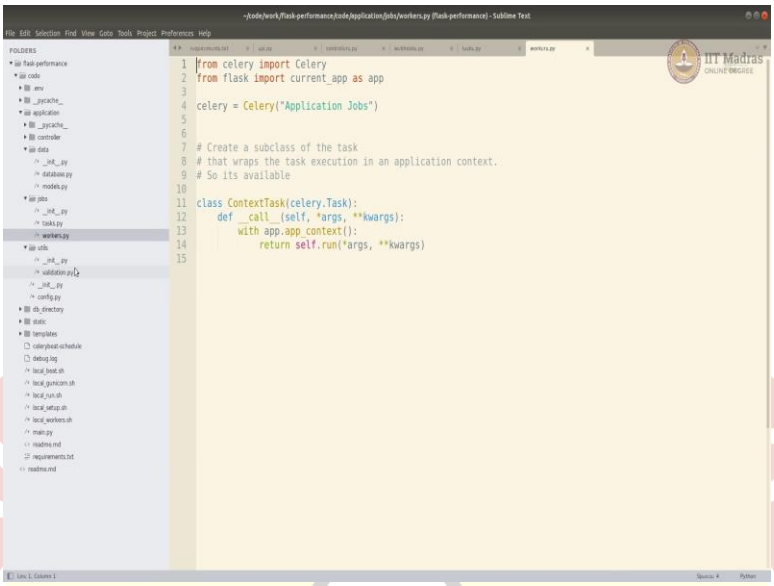
So, I am going to just do that, copy this, just paste here, paste here, paste here, then paste here, kind of almost done. So, we just need to change the package path, this is just to show that all the controllers without its API, or generic web controllers, or web controllers are all in the control of folder, or module. And all the database related things are in data, all the jobs related things are in the jobs, and all other utils if you have any more, you can put in a utils, makes it a little more cleaner.

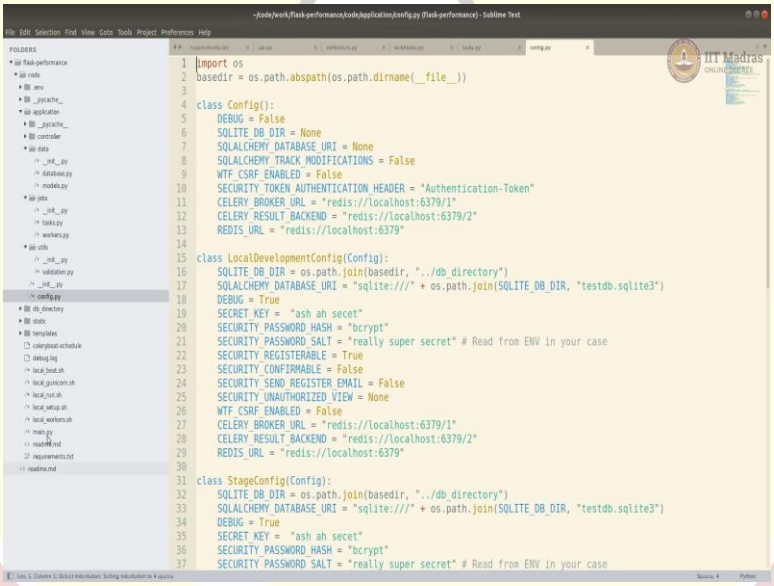
(Refer Slide Time: 03:19)

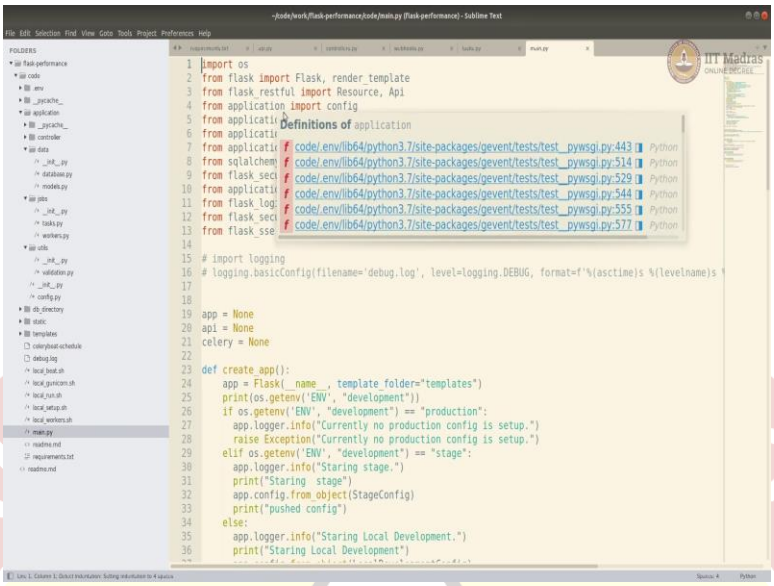


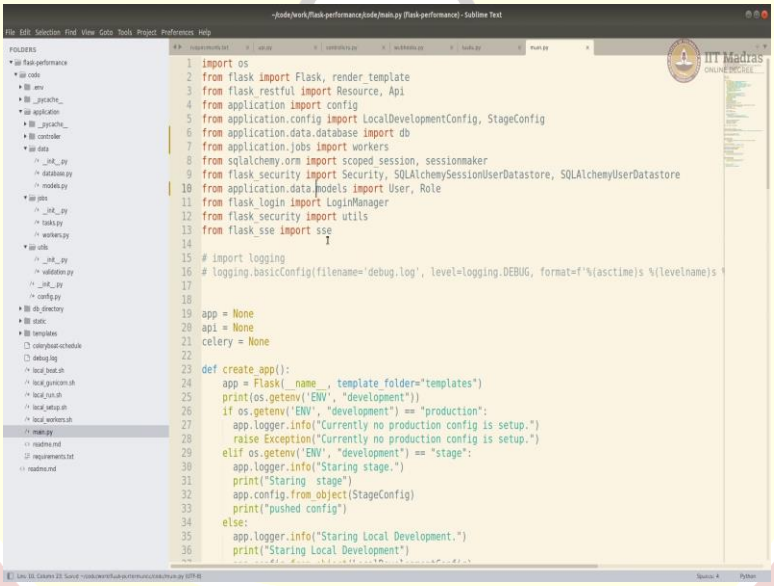

~/code/work/flask-performance/code/application/controller/webhooks.py • (flask-performance) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
flask-performance
code
controller
webhooks.py
from flask import Flask, request
from flask import render_template
from main import app as app
from application.models import Article
from flask_security import login_required, roles_accepted, roles_required
from application.jobs import tasks
from flask_sse
print("in cont
@app.route("/w
def webhook_gi
# get head
# parse th
content = request.json
print(content)
# Validate
print("--------------------")
print(request.headers)
# call async job
return "OK", 200
_running_jobs
send_job_offset
f on_job_process_down
f on_job_process_lost
f _cleanup_after_job_finish
c django_bcrypt_sha256
No details
IIT Madras
ONLINE DEGREE

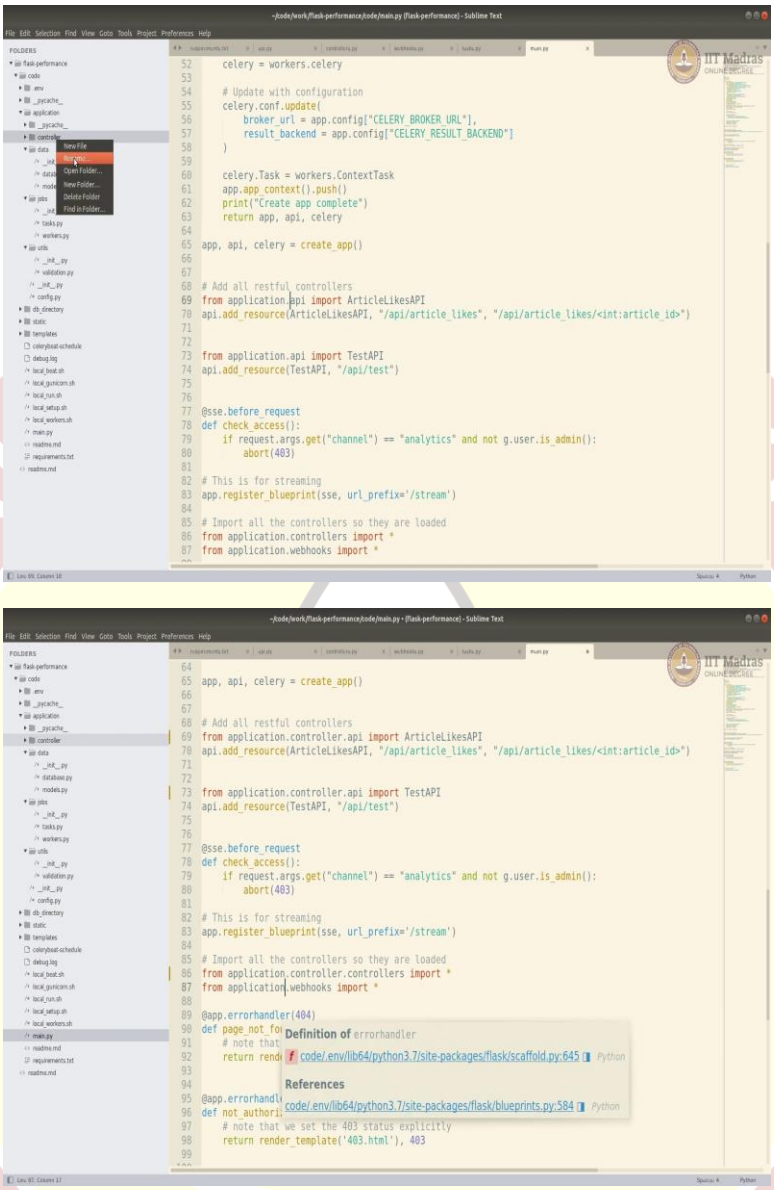

~/code/work/flask-performance/code/application/jobs/workers.py (flask-performance) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
flask-performance
code
data
jobs
workers.py
from celery import Celery
from flask import current_app as app
celery = Celery("Application Jobs")
# Create a subclass of the task
# that wraps the task execution in an application context.
# So its available
class ContextTask(celery.Task):
def __call__(self, *args, **kwargs):
with app.app_context():
return self.run(*args, **kwargs)
IIT Madras
ONLINE DEGREE






~/code/work/flask-performance/code/main.py (flask-performance) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
flask-performance
code
.env
__pycache__
application
__pycache__
controller
data
__init__.py
database.py
models.py
jobs
__init__.py
tasks.py
workers.py
utils
__init__.py
validation.py
__init__.py
config.py
db_directory
static
templates
celerybeat-schedule
debug.log
local_beat.sh
local_gunicorn.sh
local_run.sh
local_setup.sh
local_workers.sh
main.py
readme.md
requirements.txt
readme.md
Definitions of application
code/.env/lib64/python3.7/site-packages/gevent/tests/test__pywsgi.py:443
code/.env/lib64/python3.7/site-packages/gevent/tests/test__pywsgi.py:514
code/.env/lib64/python3.7/site-packages/gevent/tests/test__pywsgi.py:529
code/.env/lib64/python3.7/site-packages/gevent/tests/test__pywsgi.py:544
code/.env/lib64/python3.7/site-packages/gevent/tests/test__pywsgi.py:555
code/.env/lib64/python3.7/site-packages/gevent/tests/test__pywsgi.py:577
Python
IIT Madras
ONLINE DEGREE
import os
from flask import Flask, render_template
from flask_restful import Resource, Api
from application import config
from application.config import LocalDevelopmentConfig, StageConfig
from application.data.database import db
from application.jobs import workers
from sqlalchemy.orm import scoped_session, sessionmaker
from flask_security import Security, SQLAlchemySessionUserDatastore, SQLAlchemyUserDatastore
from application.data.models import User, Role
from flask_login import LoginManager
from flask_security import utils
from flask_sse import sse
# import logging
# logging.basicConfig(filename='debug.log', level=logging.DEBUG, format=f'%(asctime)s %(levelname)s %
app = None
api = None
celery = None
def create_app():
app = Flask(__name__, template_folder="templates")
print(os.getenv('ENV', "development"))
if os.getenv('ENV', "development") == "production":
app.logger.info("Currently no production config is setup.")
raise Exception("Currently no production config is setup.")
elif os.getenv('ENV', "development") == "stage":
app.logger.info("Staring stage.")
print("Staring stage")
app.config.from_object(StageConfig)
print("pushed config")
else:
app.logger.info("Staring Local Development.")
print("Staring Local Development")

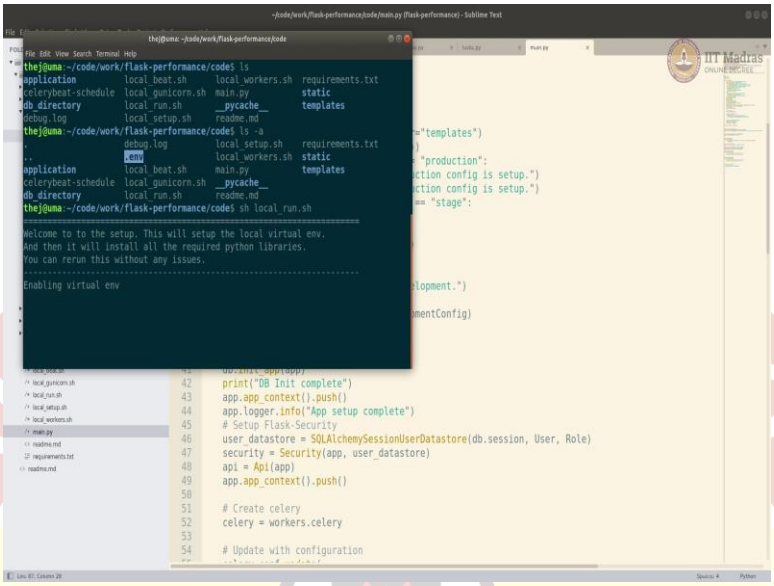

~/code/work/flask-performance/code/main.py (flask-performance) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
flask-performance
code
.env
__pycache__
application
__pycache__
controller
data
__init__.py
database.py
models.py
jobs
__init__.py
tasks.py
workers.py
utils
__init__.py
validation.py
__init__.py
config.py
db_directory
static
templates
celerybeat-schedule
debug.log
local_beat.sh
local_gunicorn.sh
local_run.sh
local_setup.sh
local_workers.sh
main.py
readme.md
requirements.txt
readme.md
IIT Madras
ONLINE DEGREE
import os
from flask import Flask, render_template
from flask_restful import Resource, Api
from application import config
from application.config import LocalDevelopmentConfig, StageConfig
from application.data.database import db
from application.jobs import workers
from sqlalchemy.orm import scoped_session, sessionmaker
from flask_security import Security, SQLAlchemySessionUserDatastore, SQLAlchemyUserDatastore
from application.data.models import User, Role
from flask_login import LoginManager
from flask_security import utils
from flask_sse import sse
# import logging
# logging.basicConfig(filename='debug.log', level=logging.DEBUG, format=f'%(asctime)s %(levelname)s %
app = None
api = None
celery = None
def create_app():
app = Flask(__name__, template_folder="templates")
print(os.getenv('ENV', "development"))
if os.getenv('ENV', "development") == "production":
app.logger.info("Currently no production config is setup.")
raise Exception("Currently no production config is setup.")
elif os.getenv('ENV', "development") == "stage":
app.logger.info("Staring stage.")
print("Staring stage")
app.config.from_object(StageConfig)
print("pushed config")
else:
app.logger.info("Staring Local Development.")
print("Staring Local Development")

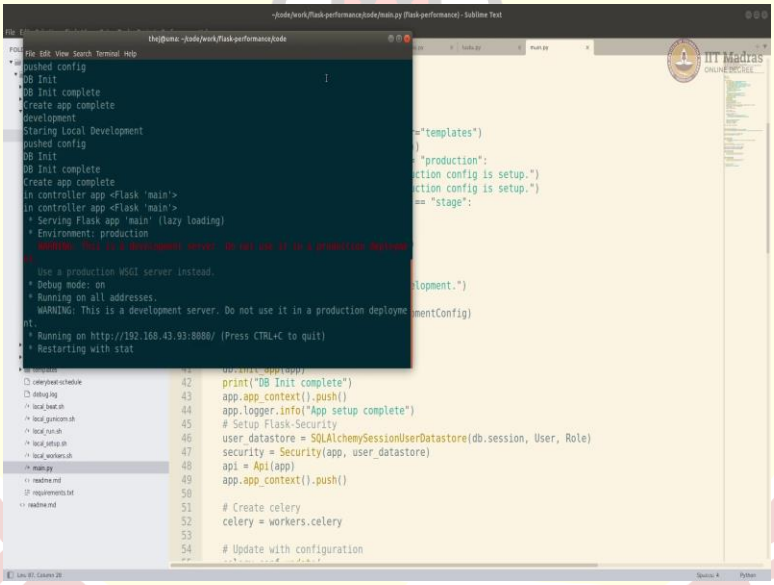


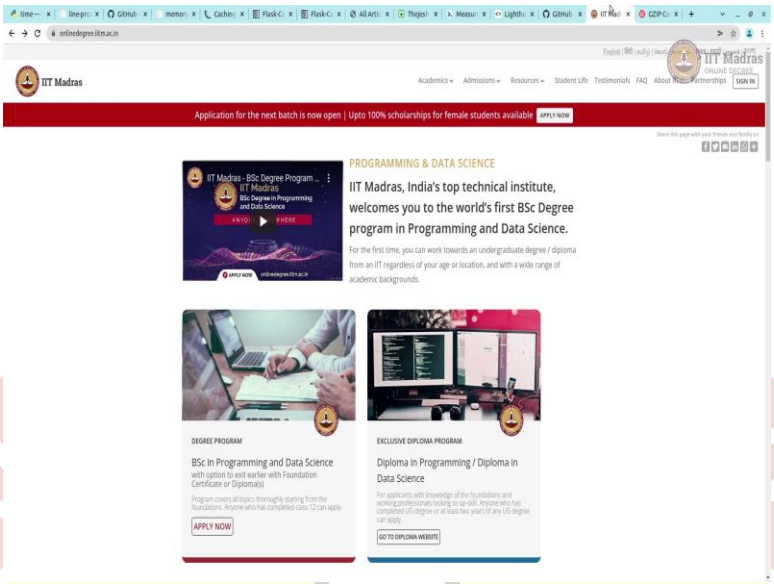
I will, just quickly change the import so that they work this application dot comes from utils. And this comes from data. This comes from data as well, anything else, nothing else here. And here main dot app here, this comes from data and application dot jobs, import task. We are books same here application dot jobs. Your data, rest seems okay. Can do similarly, refactoring in other things to just to make it clean and fast?

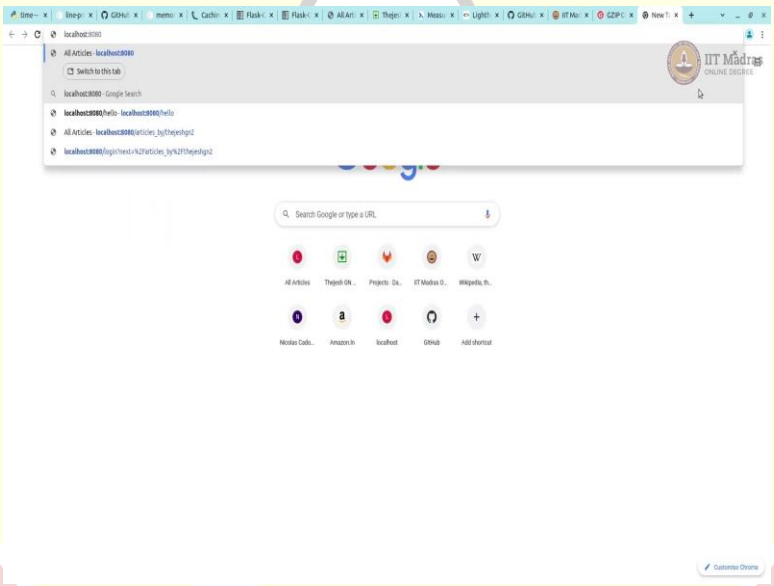
Cleaner rather than faster because I think if it is cleaner, it becomes easier to debug and improve it upon in the future. Here it is jobs, utils seems to be okay. We just have one more file left which is the main dot py. This is correct. This is correct. This will be data. Here are some of the imports. That is it, I think we have reorganized, it is pretty much clean.

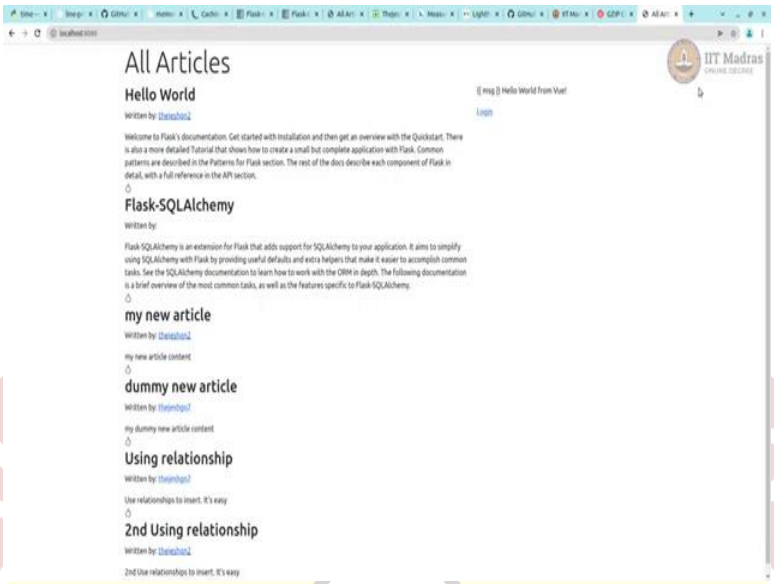
(Refer Slide Time: 05:43)





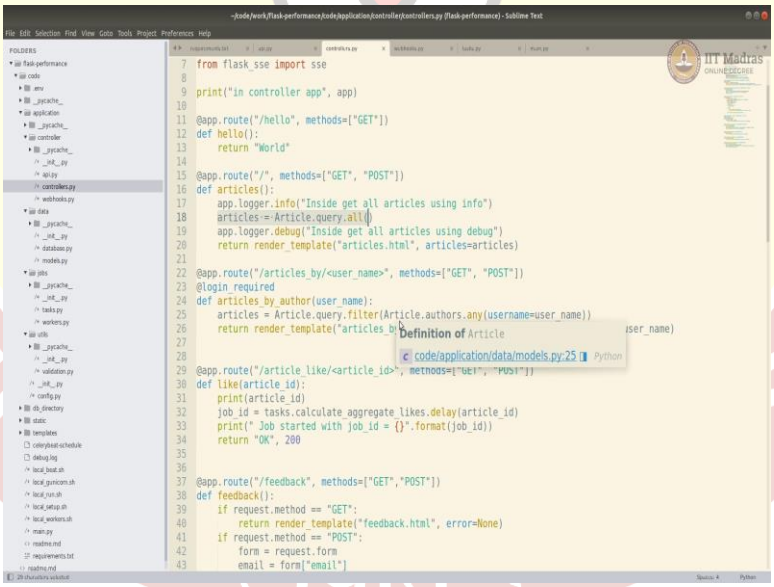

IIT Madras
Application for the next batch is now open | Upto 100% scholarships for female students available
APPLY NOW
PROGRAMMING & DATA SCIENCE
IIT Madras, India's top technical institute, welcomes you to the world's first BSc Degree program in Programming and Data Science.
For the first time, you can work towards an undergraduate degree / diploma from an IIT regardless of your age or location, and with a wide range of academic backgrounds.
DEGREE PROGRAM
BSc in Programming and Data Science
with option to exit earlier with Foundation Certificate or Diploma(s)
APPLY NOW
EXCLUSIVE DIPLOMA PROGRAM
Diploma in Programming / Diploma in Data Science
GO TO DIPLOMA WEBSITE


Search Google or type a URL



So, let us start it have already set up the environment. You can see here, consider dot enb, if you want to consider type it again. So, I will just run it just to check whether all the changes are okay, it seems to be okay, let is just open our page and see if it loads. So the restructuring is fine.

(Refer Slide Time: 06:18)

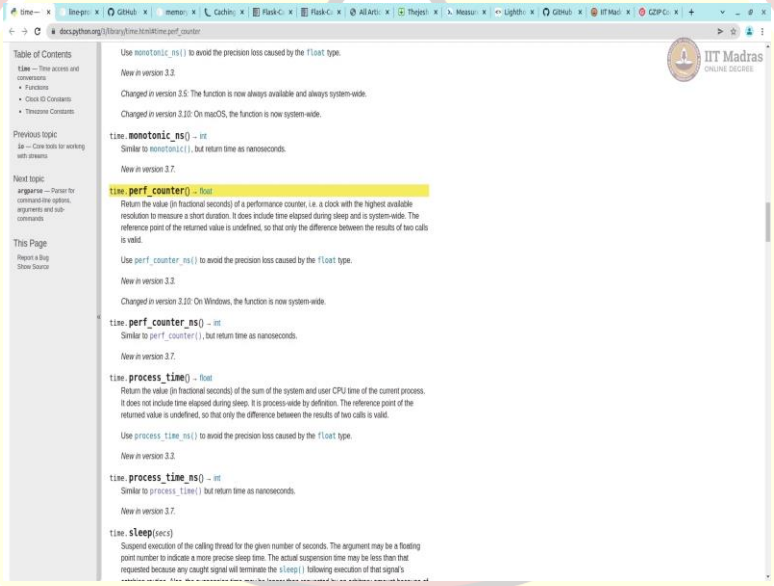


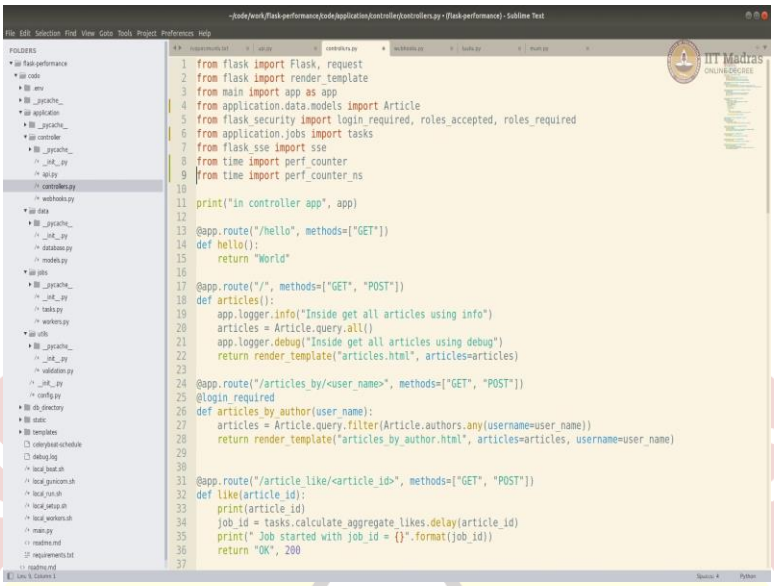
Now, we want to see how we can improve this application from two points of view. One is from the front end point of view and another one is the back end point of view. Back in, there are two parts, mostly the processing that we are going to do and the queries that we are going to run. For example, let us take a controller, here is a controller, there are three things that are happening here, one we are getting, when the user accesses slash, it is going to come to article.

And it is going to do some processing if there are loops or anything like that, it is going to do a query, then it is going to render the template and then return to the user. Now, if the articles are not going to change very often, we do not need to run this for every user all the time. Or, for example, we do not need to run at least this, if you want to change some rendering, for example, if they are rendering as username, that you do not want to do, then we do not we can want to render this, but we do not want to repeat this.

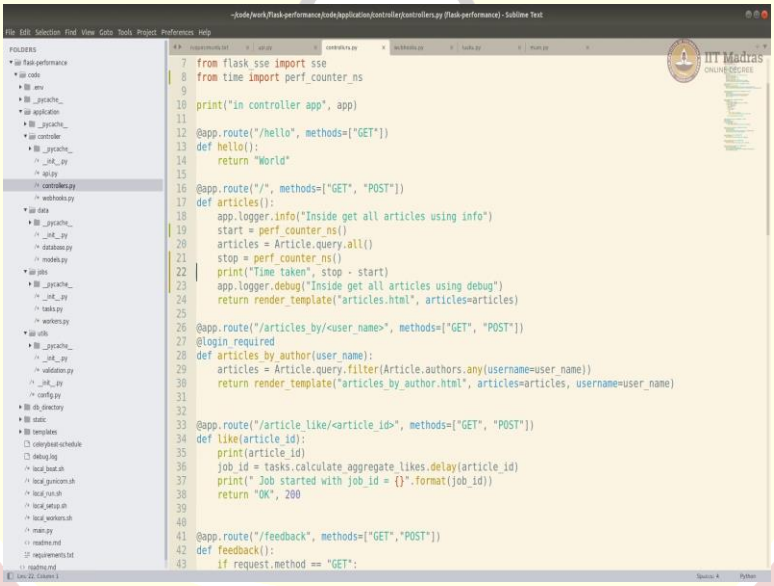
So, you can actually, reduce the process at various levels. So, you first we will start with measuring, and then we will see how to reduce it. Let us start measuring.

(Refer Slide Time: 07:44)





```
from flask import Flask, request
from flask import render_template
from main import app as app
from application.data.models import Article
from flask_security import login_required, roles_accepted, roles_required
from application.jobs import tasks
from flask_sse import sse
from time import perf_counter
from time import perf_counter_ns

print("in controller app", app)

@app.route("/hello", methods=["GET"])
def hello():
    return "World"

@app.route("/", methods=["GET", "POST"])
def articles():
    app.logger.info("Inside get all articles using info")
    articles = Article.query.all()
    app.logger.debug("Inside get all articles using debug")
    return render_template("articles.html", articles=articles)

@app.route("/articles_by/<user_name>", methods=["GET", "POST"])
@login_required
def articles_by_author(user_name):
    articles = Article.query.filter(Article.authors.any(username=user_name))
    return render_template("articles_by_author.html", articles=articles, username=user_name)


@app.route("/article_like/<article_id>", methods=["GET", "POST"])
def like(article_id):
    print(article_id)
    job_id = tasks.calculate_aggregate_likes.delay(article_id)
    print(" Job started with job_id = {}".format(job_id))
    return "OK", 200
```



```
from flask_sse import sse
from time import perf_counter_ns

print("in controller app", app)

@app.route("/hello", methods=["GET"])
def hello():
    return "World"

@app.route("/", methods=["GET", "POST"])
def articles():
    app.logger.info("Inside get all articles using info")
    start = perf_counter_ns()
    articles = Article.query.all()
    stop = perf_counter_ns()
    print("Time taken", stop - start)
    app.logger.debug("Inside get all articles using debug")
    return render_template("articles.html", articles=articles)

@app.route("/articles_by/<user_name>", methods=["GET", "POST"])
@login_required
def articles_by_author(user_name):
    articles = Article.query.filter(Article.authors.any(username=user_name))
    return render_template("articles_by_author.html", articles=articles, username=user_name)


@app.route("/article_like/<article_id>", methods=["GET", "POST"])
def like(article_id):
    print(article_id)
    job_id = tasks.calculate_aggregate_likes.delay(article_id)
    print(" Job started with job_id = {}".format(job_id))
    return "OK", 200


@app.route("/feedback", methods=["GET","POST"])
def feedback():
    if request.method == "GET":
```

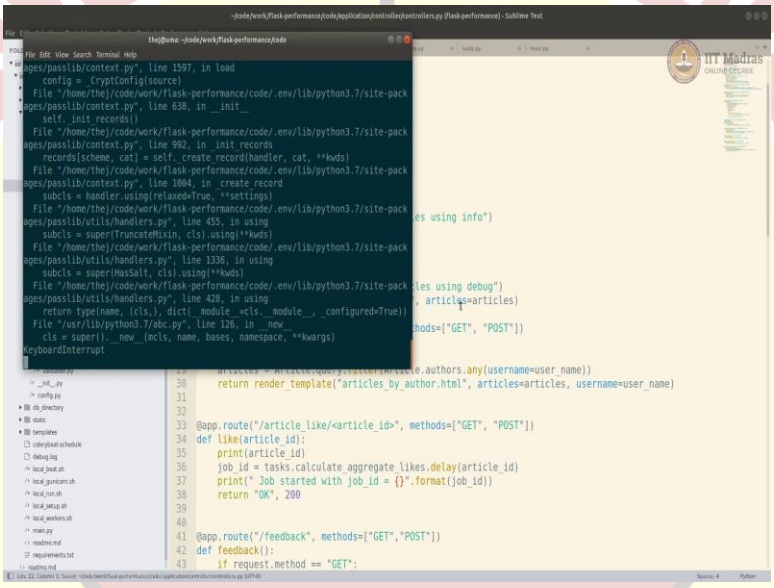
For measuring, we are going to use a simplest function called perf counter ns, it is a built in python function that we can use. It starts a timer, and then you can start or stop the timer whenever you want and measure the difference. Let us start doing that. Now I am just going to add here, I am just going to use the nanosecond one and not the second one.

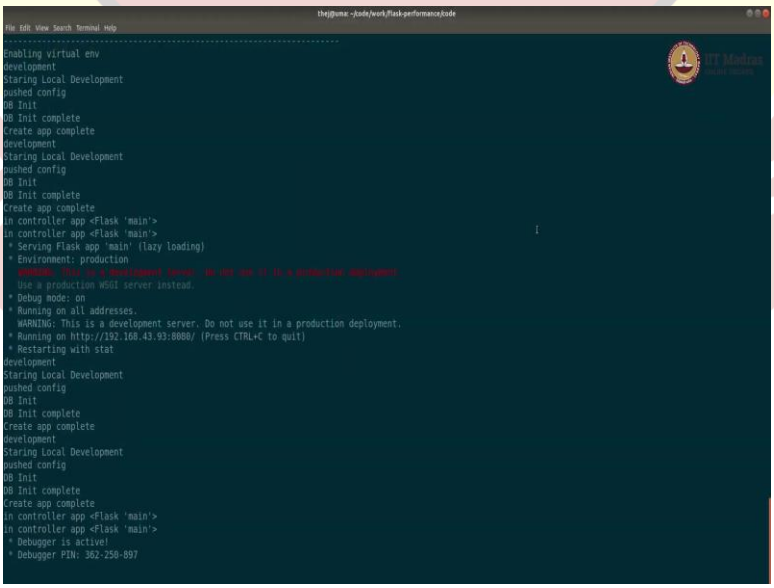
So, only this is enough. So, let us say we want to start, we want to improve this, let us assume that this is the one which is taking a lot of time, we want to improve this. So, let us add a start pointer, and then we will add a stop pointer, stop and then we will print it. Print Time taken stop minus start to the end minus start, let us say how much time it takes.

Here we do not have many articles. So, it might take a quite less time. But, I assume if you are running a media organization where you are at 10s of 1000s of articles, then this query would be taking a lot of time because it is trying to query all over the whole table. Even if you have, some constraint, it is going to run over the table and then reduce and do all of those.

So, generally, listing is one of the areas that we concentrate when you are improving the performance. So, we will see how much time it takes.

(Refer Slide Time: 09:45)





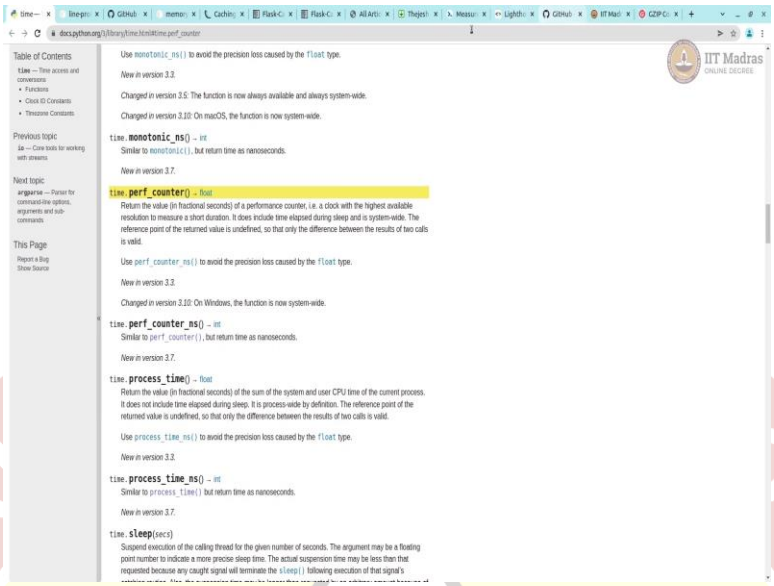


Table of Contents

time — Time access and conversions

- Functions
- Clock ID Constants
- Timezone Constants

Previous topic

io — Core tools for working with streams

Next topic

argparse — Parser for command-line options, arguments and sub-commands

This Page

Report a Bug
Show Source

Use monotonic_ns() to avoid the precision loss caused by the float type.

New in version 3.3.

Changed in version 3.5: The function is now always available and always system-wide.

Changed in version 3.10: On macOS, the function is now system-wide.

time.**monotonic_ns**() → int
Similar to monotonic(), but return time as nanoseconds.

New in version 3.7.

time.**perf_counter**() → float
Return the value (in fractional seconds) of a performance counter, i.e. a clock with the highest available resolution to measure a short duration. It does include time elapsed during sleep and is system-wide. The reference point of the returned value is undefined, so that only the difference between the results of two calls is valid.

Use perf_counter_ns() to avoid the precision loss caused by the float type.

New in version 3.3.

Changed in version 3.10: On Windows, the function is now system-wide.

time.**perf_counter_ns**() → int
Similar to perf_counter(), but return time as nanoseconds.

New in version 3.7.

time.**process_time**() → float
Return the value (in fractional seconds) of the sum of the system and user CPU time of the current process. It does not include time elapsed during sleep. It is process-wide by definition. The reference point of the returned value is undefined, so that only the difference between the results of two calls is valid.

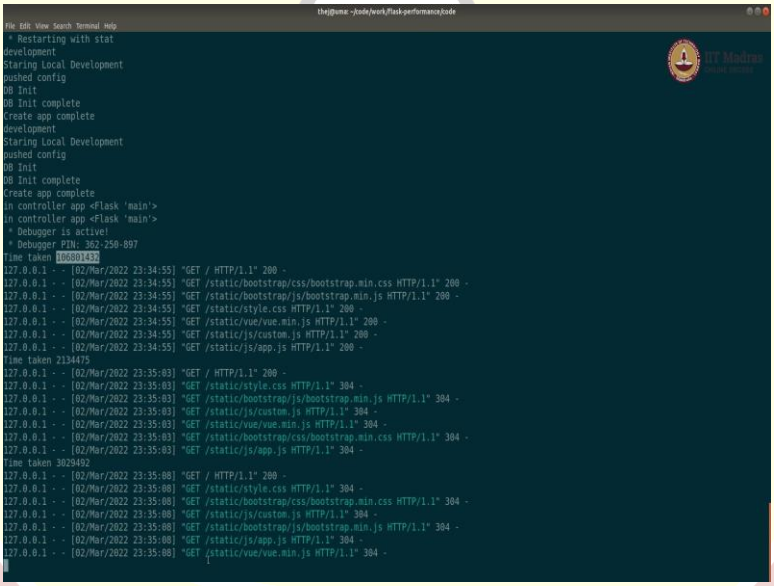
Use process_time_ns() to avoid the precision loss caused by the float type.

New in version 3.3.

time.**process_time_ns**() → int
Similar to process_time() but return time as nanoseconds.

New in version 3.7.

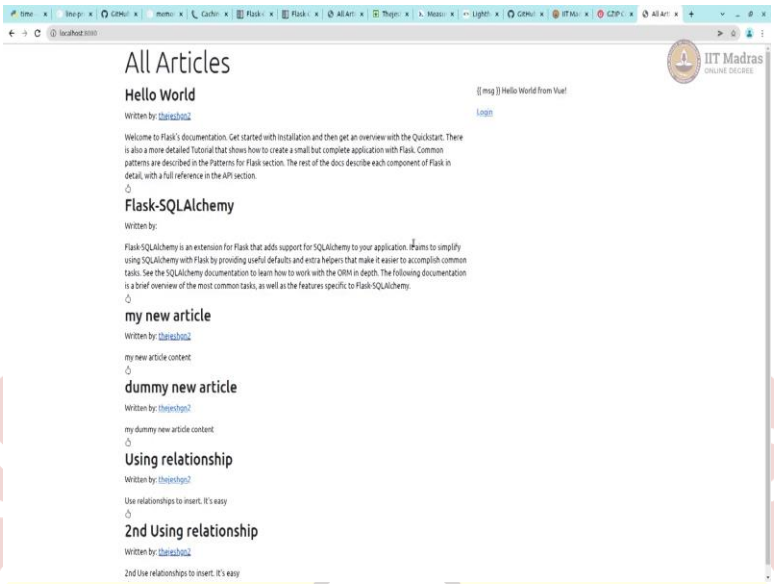
time.**sleep**(secs)
Suspend execution of the calling thread for the given number of seconds. The argument may be a floating point number to indicate a more precise sleep time. The actual suspension time may be less than that requested because any caught signal will terminate the sleep() following execution of that signal's

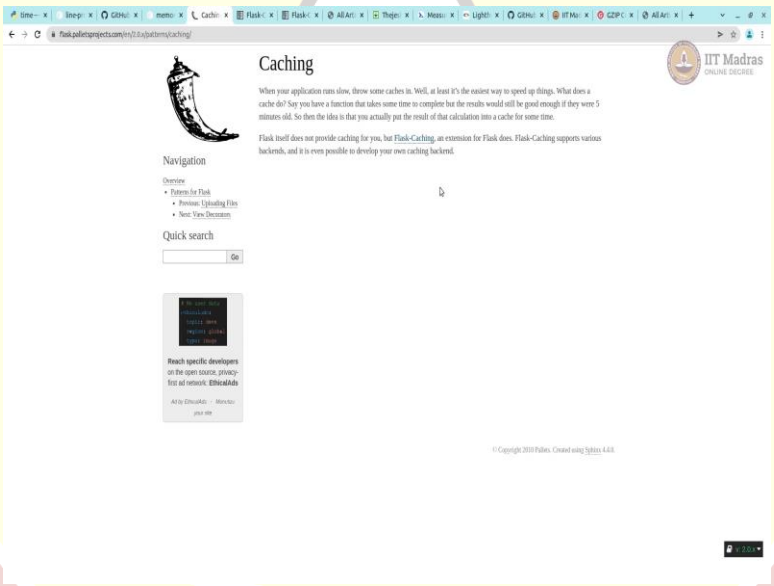


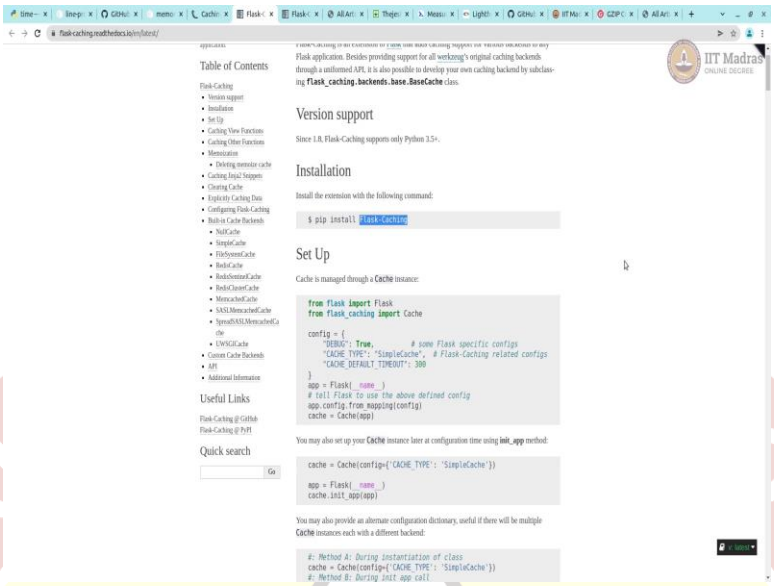
thej@umu: ~/code/work/flask-performance/code

File Edit View Search Terminal Help

```
 * Restarting with stat
development
Staring Local Development
pushed config
DB Init
DB Init complete
Create app complete
development
Staring Local Development
pushed config
DB Init
DB Init complete
Create app complete
in controller app <Flask 'main'>
in controller app <Flask 'main'>
 * Debugger is active!
 * Debugger PIN: 362-250-897
Time taken 106801432
127.0.0.1 - - [02/Mar/2022 23:34:55] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [02/Mar/2022 23:34:55] "GET /static/bootstrap/css/bootstrap.min.css HTTP/1.1" 200 -
127.0.0.1 - - [02/Mar/2022 23:34:55] "GET /static/bootstrap/js/bootstrap.min.js HTTP/1.1" 200 -
127.0.0.1 - - [02/Mar/2022 23:34:55] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 - - [02/Mar/2022 23:34:55] "GET /static/vue/vue.min.js HTTP/1.1" 200 -
127.0.0.1 - - [02/Mar/2022 23:34:55] "GET /static/js/custom.js HTTP/1.1" 200 -
127.0.0.1 - - [02/Mar/2022 23:34:55] "GET /static/js/app.js HTTP/1.1" 200 -
Time taken 2134475
127.0.0.1 - - [02/Mar/2022 23:35:03] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [02/Mar/2022 23:35:03] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [02/Mar/2022 23:35:03] "GET /static/bootstrap/js/bootstrap.min.js HTTP/1.1" 304 -
127.0.0.1 - - [02/Mar/2022 23:35:03] "GET /static/js/custom.js HTTP/1.1" 304 -
127.0.0.1 - - [02/Mar/2022 23:35:03] "GET /static/vue/vue.min.js HTTP/1.1" 304 -
127.0.0.1 - - [02/Mar/2022 23:35:03] "GET /static/bootstrap/css/bootstrap.min.css HTTP/1.1" 304 -
127.0.0.1 - - [02/Mar/2022 23:35:03] "GET /static/js/app.js HTTP/1.1" 304 -
Time taken 3029492
127.0.0.1 - - [02/Mar/2022 23:35:08] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [02/Mar/2022 23:35:08] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [02/Mar/2022 23:35:08] "GET /static/bootstrap/css/bootstrap.min.css HTTP/1.1" 304 -
127.0.0.1 - - [02/Mar/2022 23:35:08] "GET /static/js/custom.js HTTP/1.1" 304 -
127.0.0.1 - - [02/Mar/2022 23:35:08] "GET /static/bootstrap/js/bootstrap.min.js HTTP/1.1" 304 -
127.0.0.1 - - [02/Mar/2022 23:35:08] "GET /static/js/app.js HTTP/1.1" 304 -
127.0.0.1 - - [02/Mar/2022 23:35:08] "GET /static/vue/vue.min.js HTTP/1.1" 304 -
```


All Articles
Hello World
Written by: thejeshgn2
Welcome to Flask's documentation. Get started with Installation and then get an overview with the Quickstart. There is also a more detailed Tutorial that shows how to create a small but complete application with Flask. Common patterns are described in the Patterns for Flask section. The rest of the docs describe each component of Flask in detail, with a full reference in the API section.
Flask-SQLAlchemy
Written by:
Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simplify using SQLAlchemy with Flask by providing useful defaults and extra helpers that make it easier to accomplish common tasks. See the SQLAlchemy documentation to learn how to work with the ORM in depth. The following documentation is a brief overview of the most common tasks, as well as the features specific to Flask-SQLAlchemy.
my new article
Written by: thejeshgn2
my new article content
dummy new article
Written by: thejeshgn2
my dummy new article content
Using relationship
Written by: thejeshgn2
Use relationships to insert. It's easy
2nd Using relationship
Written by: thejeshgn2
2nd Use relationships to insert. It's easy
{{ msg }} Hello World from Vue!
Login
IIT Madras
ONLINE DEGREE


Caching
When your application runs slow, throw some caches in. Well, at least it's the easiest way to speed up things. What does a cache do? Say you have a function that takes some time to complete but the results would still be good enough if they were 5 minutes old. So then the idea is that you actually put the result of that calculation into a cache for some time.
Flask itself does not provide caching for you, but Flask-Caching, an extension for Flask does. Flask-Caching supports various backends, and it is even possible to develop your own caching backend.
Navigation
Overview
Patterns for Flask
Previous: Uploading Files
Next: View Decorators
Quick search
Go
Reach specific developers
on the open source, privacy-
first ad network: EthicalAds
IIT Madras
ONLINE DEGREE



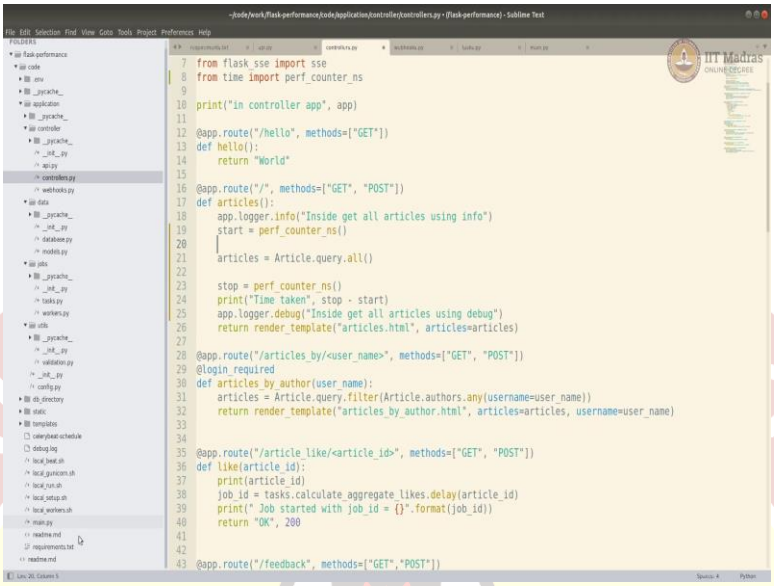
Let us stop this starting in, and let us refresh this and not close the page. So, here it is, so it is taking, some nanoseconds. And if I refresh, it is roughly changing. But it is taking somewhat similar time, it changes sometimes based on, whether the code is already compiled, it is in the memory of the based on that. But it is taking, what we call significant amount of time.

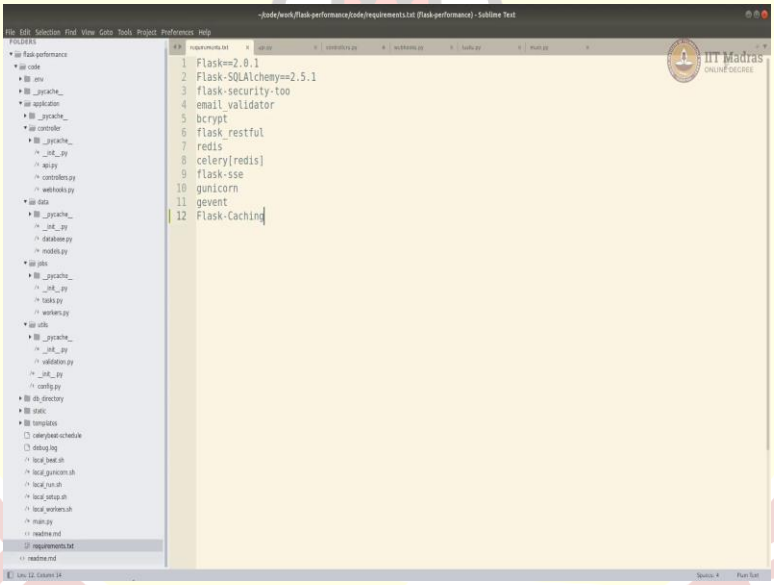
Because we know that it is entering here, running this query, and all of this. What if, what if the articles do not change often? Let us say all our editors upload the article in the morning, and we do not change articles during the day, and only the evening, they will upload again. So, we could literally cache this for a whole day without querying the database, every time the user accesses.

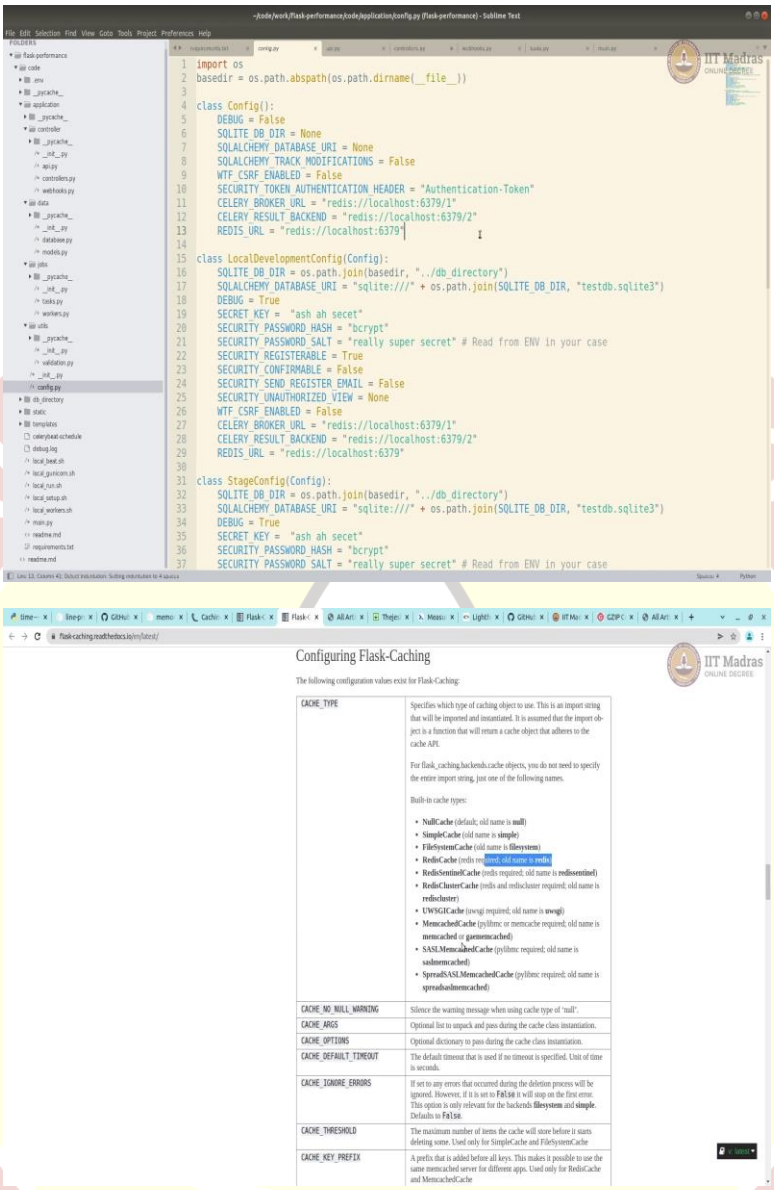
So, if in case, if this whole thing came from memory, it would have been really fast instead of hitting a database. Now, here, we are using a SQLite. So, it is on the same machine. And it is relatively fast, because I am using an SSD and also I have data in that, very less data compared to any media organization. But assume a case where you are hitting a Postgres database on a different machine.

And there is a network called network query, getting the data and doing the stuff, then actually optimizing this becomes very important. Let us see how to optimize this if you want to optimize this. Basically, I do not want to get this every time, I want to get it only when required, or cache it for time that I do not want to query database all the time. To cache this, we are going to use a library called flask caching it is one of the recommended ones by the flask project itself, you can go check out. Check the project, can install it by pip installing Flask-caching.

(Refer Slide Time: 12:08)







13:30

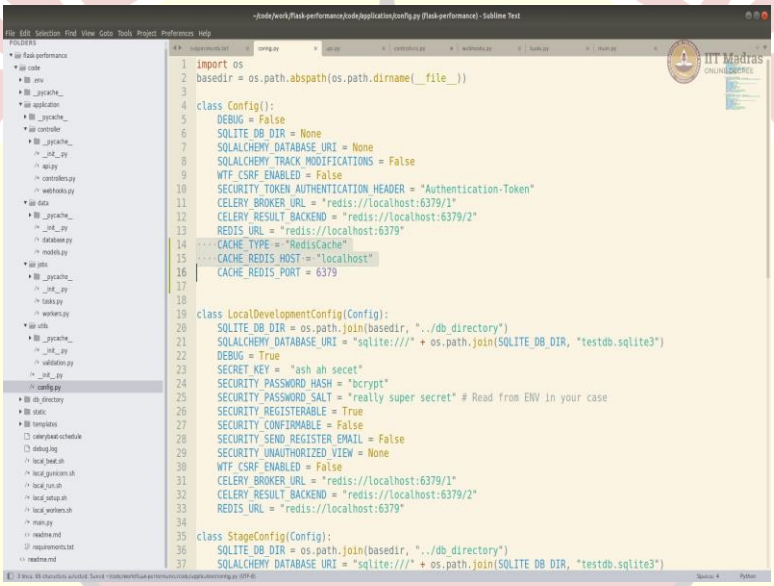
I have already added to requirements dot txt here. And I have already installed it, so I do not need to do any further installation, go ahead and install on your local machine. Once you install, we need to do some setup, flask cache in, supports multiple types of caching, you can make it null, which means no caching. And there is simple caching, Redis caching in and various kinds of caching.

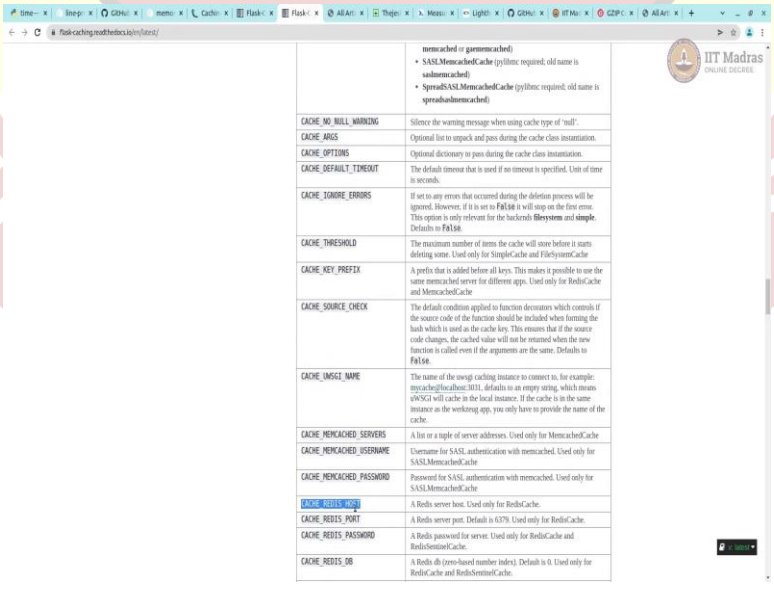
We could use local file system caching in the sense you read from the database, write it to local file and read it from there and display to the user every time you want to. Or you could use Redis

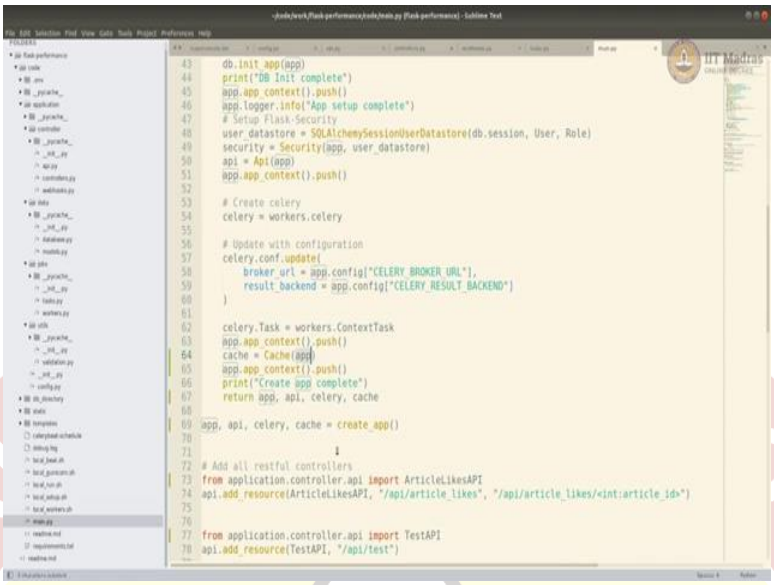
because very since Redis is a fast in-memory key value database, it could be much much faster than actually calling the database.

Similarly, you can use memcached. In this case, we are going to use a Redis Cache, because we already have the Redis running and we are going to use the same Redis of for many things it is some multipurpose tool. So, we are going to set up cache type here. And since you are going to use Redis, you are going to set it up Redis cache in the config.

(Refer Slide Time: 13:31)







And then since we are going to use Redis, you are going to set up the host. And we had to set up the port, host will be local host because I am running on the same machine, most of the time, if you want to use something as a cache, you would probably want to run it on the same machine. To reduce the network I am going to copy paste the same into the local development and staging.

So, now this is we have added the configuration, but we have to enable caching for our application. And that can be done with like any other app installation we do, we will also do cache installation, cache initialization. So, import cache, then similar to how we do other app initialization and we will also do cache. So, it is just we will do it at the end.

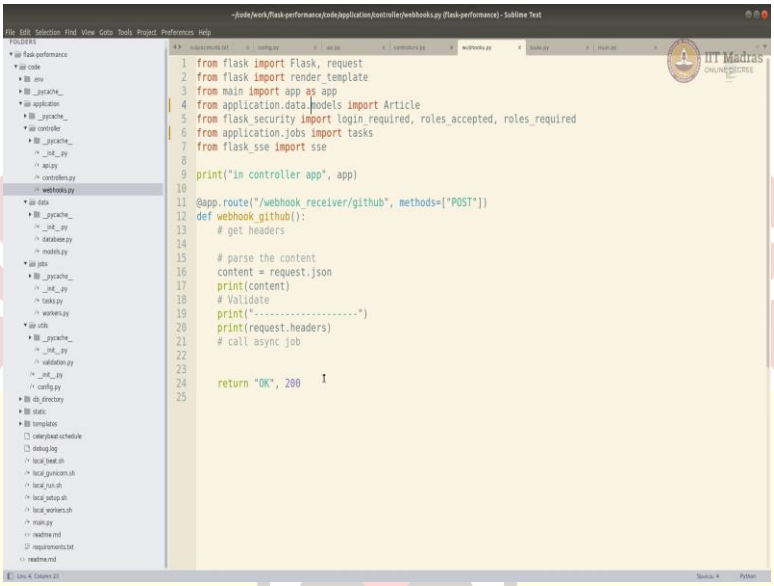
Then just push the context again, and return cache, because we are going to use it under for it multiple places, and I am going to setup the cache null in the beginning, or none. And then initialize it. And this will use all the things from the config, and then create it. This seems to be done. Now we want to do it here.

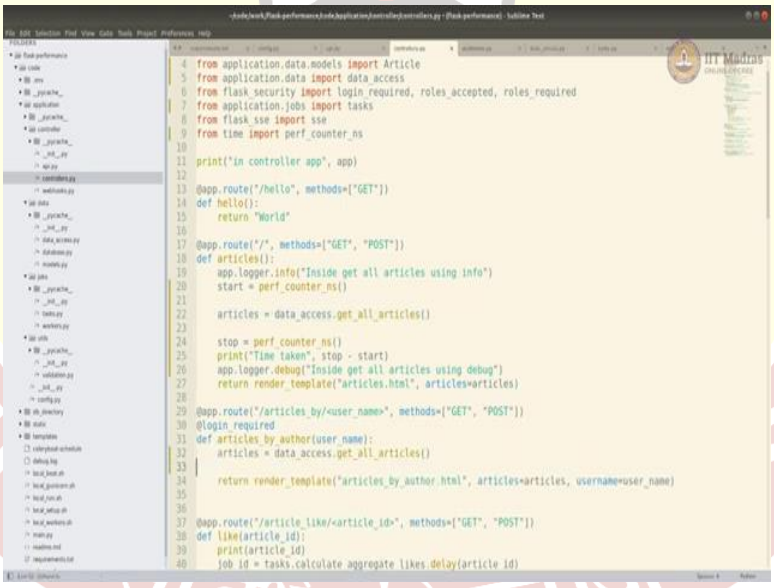
Now, usually, we actually try to keep the access functions or methods in a separate module, or in a separate file. And so that it can be reused across different calls. For example, I had articles get if I had a function called get articles that can be used here, that can be used in an API that can be used in web books, all of that will use the same function and it can be reused.

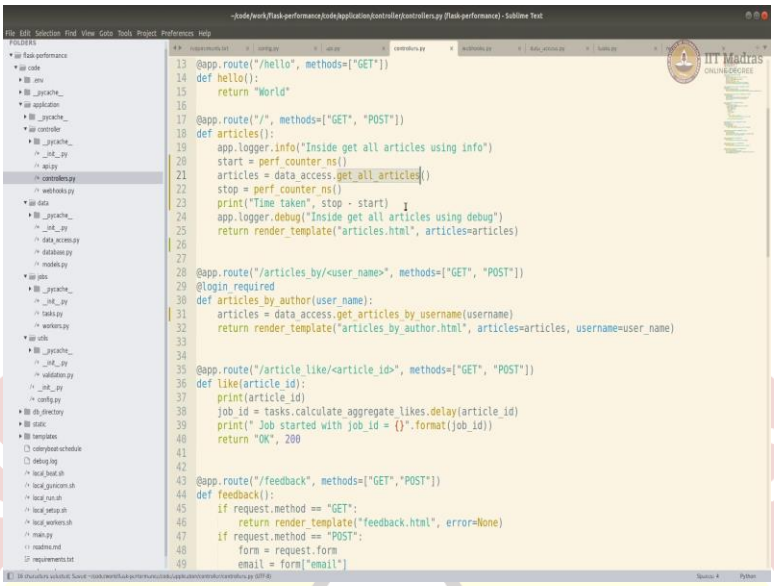
And it is very useful, because then you can apply your caching, your authorization or any other things that you want to do at one place. And it applies, regardless from where it is accessed. So,

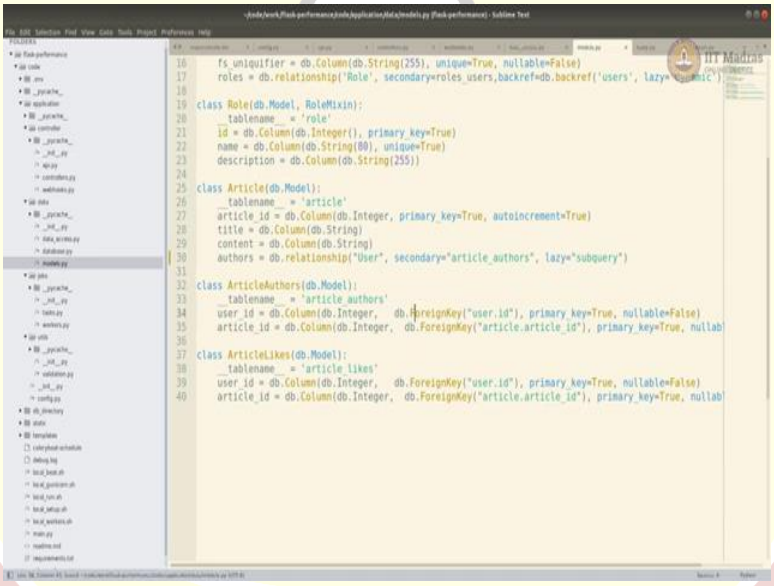
what we will do is we will create a file called data access dot pie, and I will add some functions to it. So, that it can be used by our controllers.

(Refer Slide Time: 16:31)









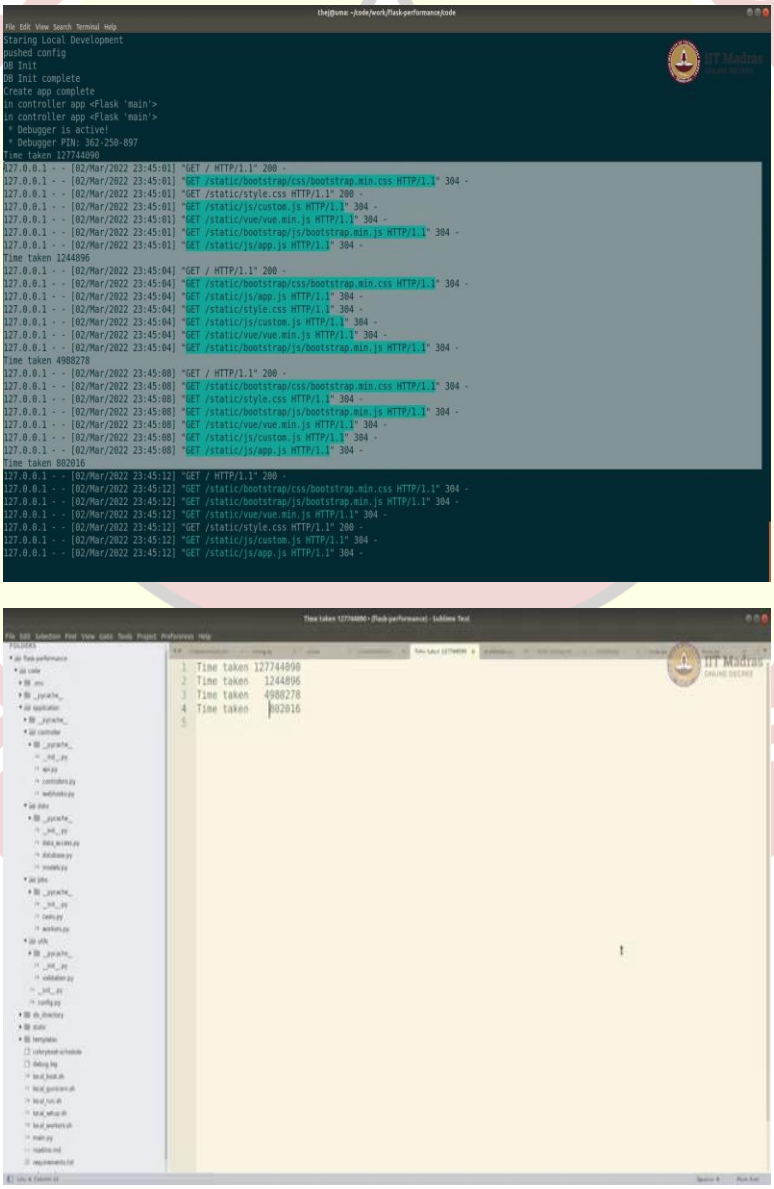
First thing, I will add article and article likes, and then I will add a function called get articles. And it is going to do one simple thing return the articles. Similarly, I will add another function get articles by user name. That also we use in one of our controllers, and I return the articles filtered by the username. Now, let us take this get all articles and use it in the controller.

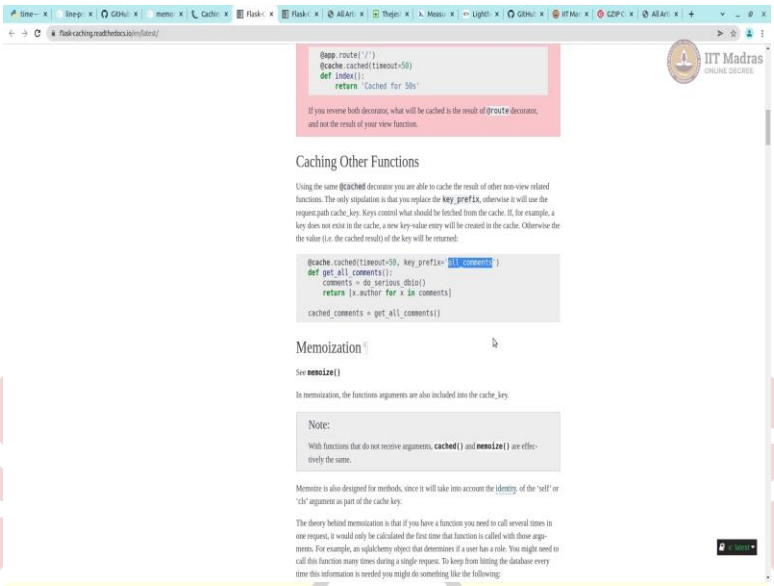
Instead of this I can just get our articles, before that just do from data, data access spelling correct, access correct. Now, we will call this. Similarly instead of this could use and get articles by username and pass the username. So, now this is all done. So, you are start is the end is there data accesses there. Now, we just want to cache this is that can be done very easily here.

So, we want to cache the output of this, whenever this gets called, we want the cached or to be returned. And that can be done very easily by using a cache. So, this says, cache it for at least 50 microseconds or milliseconds. And the cache prefixes get all articles. So, now one of the things that you might have to do is to change your models, because I think since it is in time the lazy dynamic will not work for the article author scating.

So, you might have to change it to get it using sub-query so that it gets everything and caches it. So, that can be done just by changing lazy is equal to sub-query this. So, now we are ready.

(Refer Slide Time: 19:47)





```
@app.route('/')
@cache.cached(timeout=50)
def index():
    return 'Cached for 50s'
```

If you reverse both decorator, what will be cached is the result of `@route` decorator, and not the result of your view function.

Caching Other Functions

Using the same `@cached` decorator you are able to cache the result of other non-view related functions. The only stipulation is that you replace the `key_prefix`, otherwise it will use the request.path cache_key. Keys control what should be fetched from the cache. If, for example, a key does not exist in the cache, a new key-value entry will be created in the cache. Otherwise the the value (i.e. the cached result) of the key will be returned:

```
@cache.cached(timeout=50, key_prefix='all_comments')
def get_all_comments():
    comments = do_serious_dbio()
    return [x.author for x in comments]

cached_comments = get_all_comments()
```

Memoization

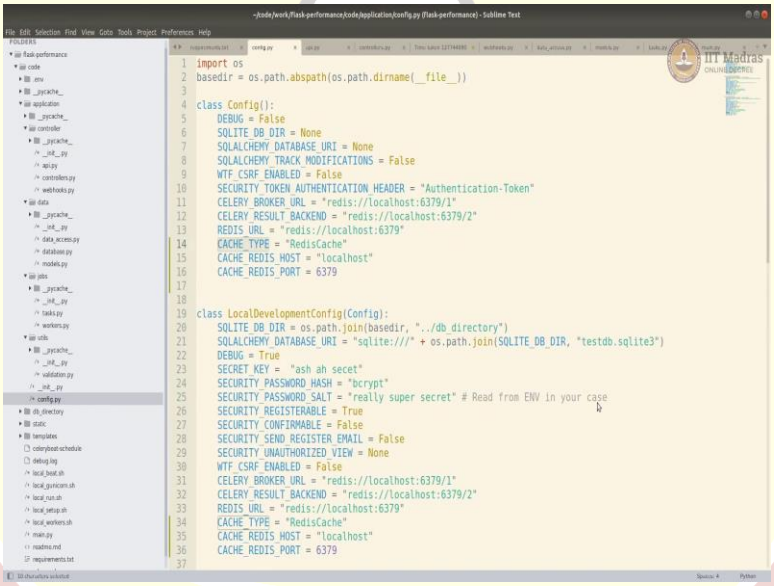
See **`memoize()`**

In memoization, the functions arguments are also included into the cache_key.

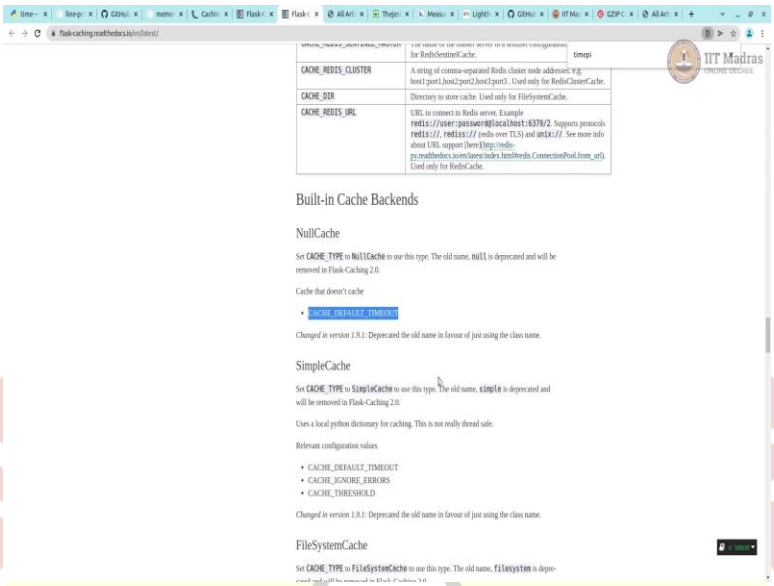
> **Note:**
>
> With functions that do not receive arguments, **`cached()`** and **`memoize()`** are effectively the same.

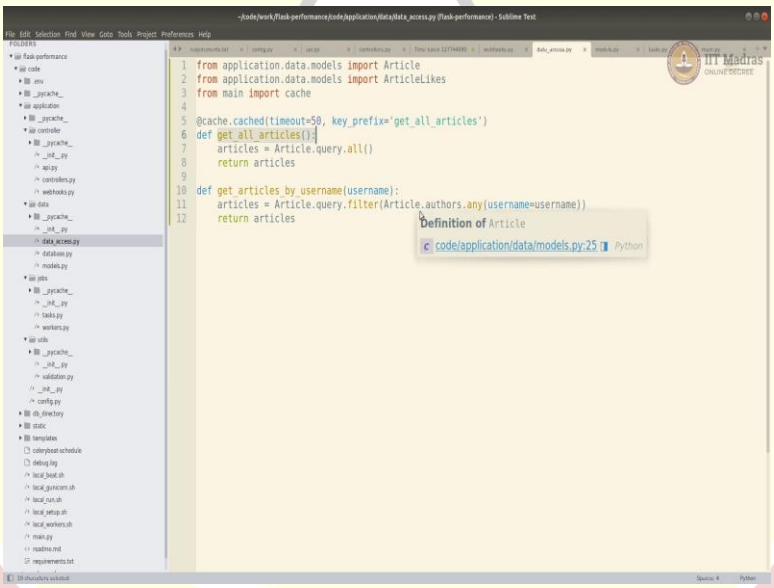
Memoize is also designed for methods, since it will take into account the `identity` of the 'self' or 'cls' argument as part of the cache key.

The theory behind memoization is that if you have a function you need to call several times in one request, it would only be calculated the first time that function is called with those arguments. For example, an sqlalchemy object that determines if a user has a role. You might need to call this function many times during a single request. To keep from hitting the database every time this information is needed you might do something like the following:



```
import os
basedir = os.path.abspath(os.path.dirname(__file__))

class Config():
    DEBUG = False
    SQLITE_DB_DIR = None
    SQLALCHEMY_DATABASE_URI = None
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    WTF_CSRF_ENABLED = False
    SECURITY_TOKEN_AUTHENTICATION_HEADER = "Authentication-Token"
    CELERY_BROKER_URL = "redis://localhost:6379/1"
    CELERY_RESULT_BACKEND = "redis://localhost:6379/2"
    REDIS_URL = "redis://localhost:6379"
    CACHE_TYPE = "RedisCache"
    CACHE_REDIS_HOST = "localhost"
    CACHE_REDIS_PORT = 6379


class LocalDevelopmentConfig(Config):
    SQLITE_DB_DIR = os.path.join(basedir, "../db_directory")
    SQLALCHEMY_DATABASE_URI = "sqlite:///" + os.path.join(SQLITE_DB_DIR, "testdb.sqlite3")
    DEBUG = True
    SECRET_KEY = "ash ah secet"
    SECURITY_PASSWORD_HASH = "bcrypt"
    SECURITY_PASSWORD_SALT = "really super secret" # Read from ENV in your case
    SECURITY_REGISTERABLE = True
    SECURITY_CONFIRMABLE = False
    SECURITY_SEND_REGISTER_EMAIL = False
    SECURITY_UNAUTHORIZED_VIEW = None
    WTF_CSRF_ENABLED = False
    CELERY_BROKER_URL = "redis://localhost:6379/1"
    CELERY_RESULT_BACKEND = "redis://localhost:6379/2"
    REDIS_URL = "redis://localhost:6379"
    CACHE_TYPE = "RedisCache"
    CACHE_REDIS_HOST = "localhost"
    CACHE_REDIS_PORT = 6379
```





Let us check whether this works, controller. Let us check if this works start this, took that much, that much, that much. So, let us check the difference, this is the first time axis. This is the subsequent accesses. So, you can see how small they are compared to the first time axis, at least 10 to more than 10,100 times smaller. It is a factor of that so it is significantly faster.

Now, if you had a much bigger, much bigger data, or much bigger database, it, this could have been this first number would have been much bigger. This first number would have been much bigger, this would be somewhat some similar, because it is still reading from memory. What we are using here is the flask caching cached here, this one, and key prefixes.

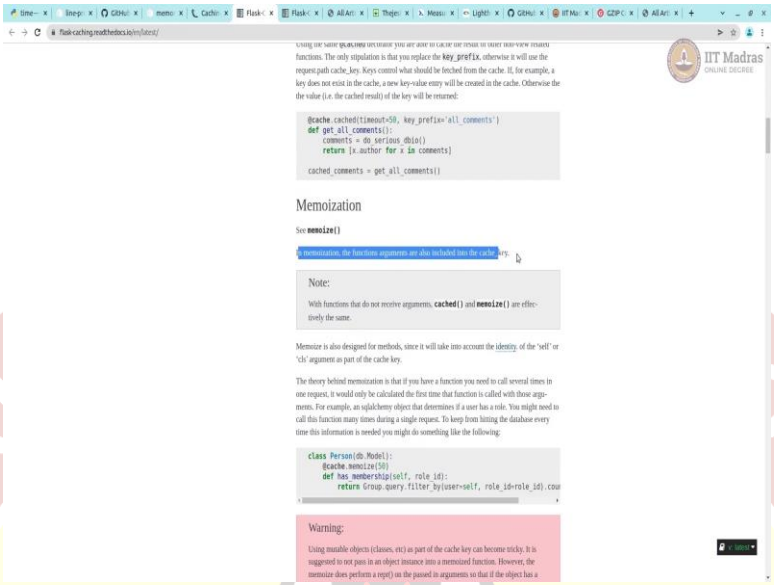
All comments, and usually you keep it unique. So, that if you are, you do not want two functions to return the same thing if you have the same, prefix, or if they have the same name. So, ideally, you would want the key prefix to be separate. So, that, the keys are unique, and are cached properly. The other one to remember is the timeout.

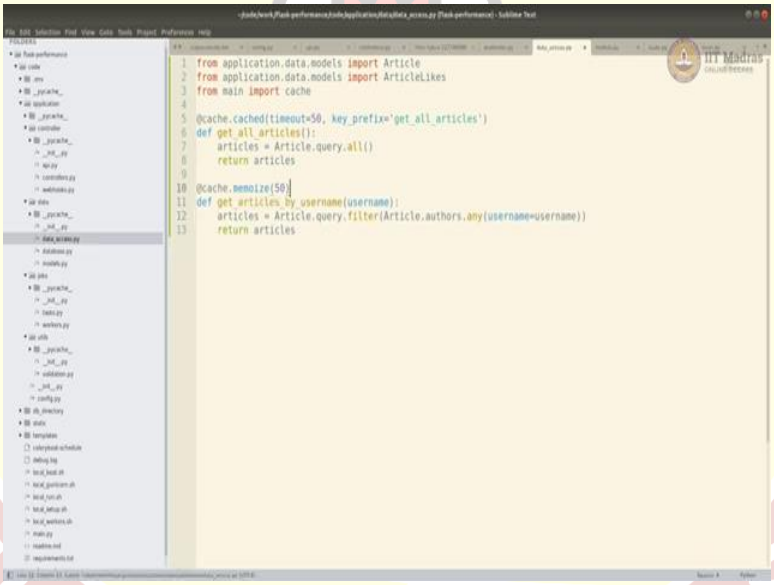
And if it automatically timed out, we could also define a standard timeout in our config. We are not defined here. But we can default timeout, we can define a default timeout, we are default timeout. Cache default timeout, we could use this could keep it to 300 milliseconds or something. And then you can overwrite based on the requirement of that specific function. You could also do this for this. But here is a small change.

This has to have for user name. So, this does not have any input to the output always remains the same, regardless of what is the user. What is the user name, or what is a, who is the user or regardless of whether he is accessing from the first page last page, whatever page, unless you have a page initial as part of this, but we do not have it.

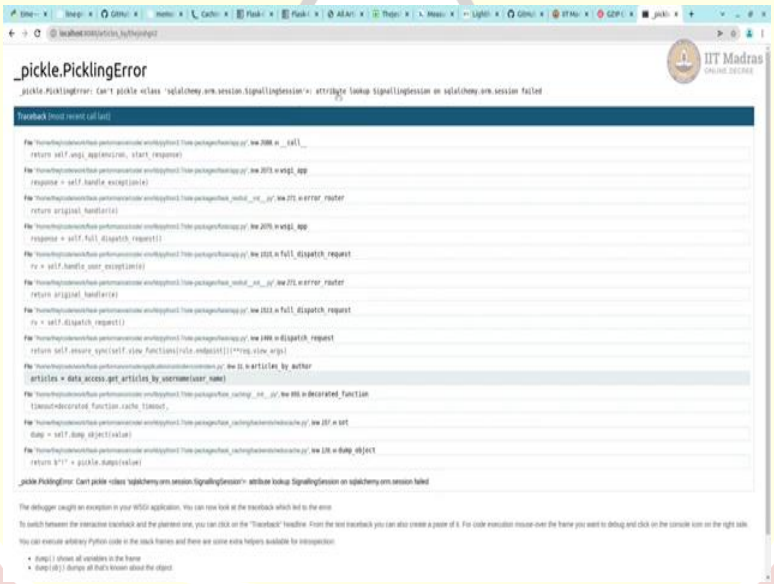
But in this, it differs because this is articles by user name. This is usually getting called when we access this link here, and this could be different users. And we should cache it differently for different users. Otherwise, sometimes, if we do not do it properly, for all users, it will show this same content, which is wrong. That is why we are going to use this different caching way to do it. It is called memorize.

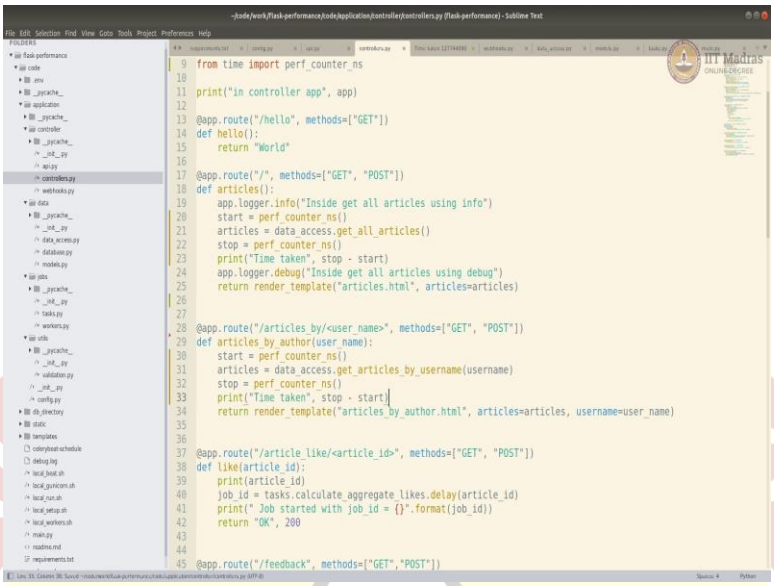
(Refer Slide Time: 23:47)



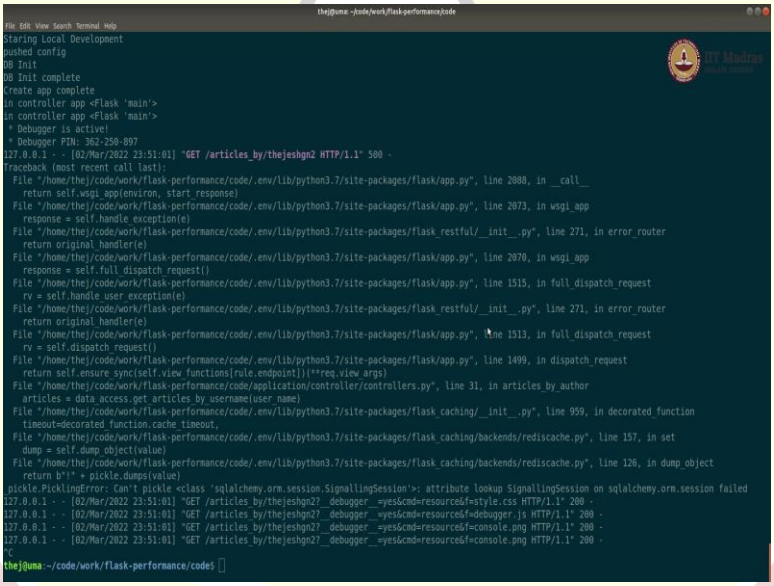



NameError
NameError: name 'username' is not defined
Traceback (most recent call last)
NameError: name 'username' is not defined
The debugger caught an exception in your WSGI application. You can now look at the traceback which led to the error.
To switch between the interactive traceback and the plaintext one, you can click on the "Traceback" headline. From the text traceback you can also create a paste of it. For code execution mouse-over the frame you want to debug and click on the console icon on the right side.
You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:
dump() shows all variables in the frame
dump(obj) dumps all that's known about the object


_pickle.PicklingError
_pickle.PicklingError: Can't pickle <class 'sqlalchemy.orm.session.SignallingSession'>: attribute lookup SignallingSession on sqlalchemy.orm.session failed
Traceback (most recent call last)
_pickle.PicklingError: Can't pickle <class 'sqlalchemy.orm.session.SignallingSession'>: attribute lookup SignallingSession on sqlalchemy.orm.session failed
The debugger caught an exception in your WSGI application. You can now look at the traceback which led to the error.
To switch between the interactive traceback and the plaintext one, you can click on the "Traceback" headline. From the text traceback you can also create a paste of it. For code execution mouse-over the frame you want to debug and click on the console icon on the right side.
You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:
dump() shows all variables in the frame
dump(obj) dumps all that's known about the object



```
from time import perf_counter_ns

print("in controller app", app)

@app.route("/hello", methods=["GET"])
def hello():
    return "World"

@app.route("/", methods=["GET", "POST"])
def articles():
    app.logger.info("Inside get all articles using info")
    start = perf_counter_ns()
    articles = data_access.get_all_articles()
    stop = perf_counter_ns()
    print("Time taken", stop - start)
    app.logger.debug("Inside get all articles using debug")
    return render_template("articles.html", articles=articles)


@app.route("/articles_by/<user_name>", methods=["GET", "POST"])
def articles_by_author(user_name):
    start = perf_counter_ns()
    articles = data_access.get_articles_by_username(username)
    stop = perf_counter_ns()
    print("Time taken", stop - start)
    return render_template("articles_by_author.html", articles=articles, username=user_name)


@app.route("/article_like/<article_id>", methods=["GET", "POST"])
def like(article_id):
    print(article_id)
    job_id = tasks.calculate_aggregate_likes.delay(article_id)
    print(" Job started with job_id = {}".format(job_id))
    return "OK", 200


@app.route("/feedback", methods=["GET","POST"])
```



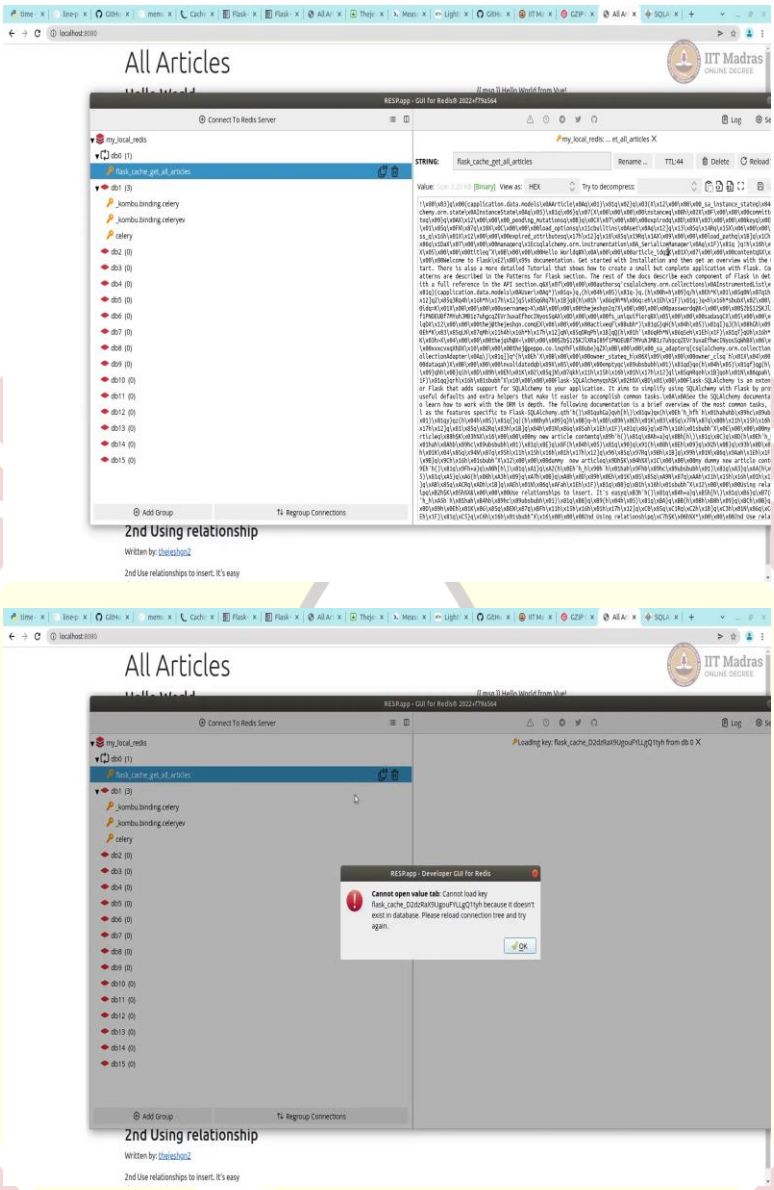
```
Staring Local Development
pushed config
DB Init
DB Init complete
Create app complete
in controller app <Flask 'main'>
in controller app <Flask 'main'>
 * Debugger is active!
 * Debugger PIN: 362-250-897
127.0.0.1 - - [02/Mar/2022 23:51:01] "GET /articles_by/thejeshgn2 HTTP/1.1" 500 -
Traceback (most recent call last):
  File "/home/thej/code/work/flask-performance/code/.env/lib/python3.7/site-packages/flask/app.py", line 2088, in __call__
    return self.wsgi_app(environ, start_response)
  File "/home/thej/code/work/flask-performance/code/.env/lib/python3.7/site-packages/flask/app.py", line 2073, in wsgi_app
    response = self.handle_exception(e)
  File "/home/thej/code/work/flask-performance/code/.env/lib/python3.7/site-packages/flask_restful/__init__.py", line 271, in error_router
    return original_handler(e)
  File "/home/thej/code/work/flask-performance/code/.env/lib/python3.7/site-packages/flask/app.py", line 2070, in wsgi_app
    response = self.full_dispatch_request()
  File "/home/thej/code/work/flask-performance/code/.env/lib/python3.7/site-packages/flask/app.py", line 1515, in full_dispatch_request
    rv = self.handle_user_exception(e)
  File "/home/thej/code/work/flask-performance/code/.env/lib/python3.7/site-packages/flask_restful/__init__.py", line 271, in error_router
    return original_handler(e)
  File "/home/thej/code/work/flask-performance/code/.env/lib/python3.7/site-packages/flask/app.py", line 1513, in full_dispatch_request
    rv = self.dispatch_request()
  File "/home/thej/code/work/flask-performance/code/.env/lib/python3.7/site-packages/flask/app.py", line 1499, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**req.view_args)
  File "/home/thej/code/work/flask-performance/code/application/controller/controllers.py", line 31, in articles_by_author
    articles = data_access.get_articles_by_username(user_name)
  File "/home/thej/code/work/flask-performance/code/.env/lib/python3.7/site-packages/flask_caching/__init__.py", line 959, in decorated_function
    timeout=decorated_function.cache_timeout,
  File "/home/thej/code/work/flask-performance/code/.env/lib/python3.7/site-packages/flask_caching/backends/rediscache.py", line 157, in set
    dump = self.dump_object(value)
  File "/home/thej/code/work/flask-performance/code/.env/lib/python3.7/site-packages/flask_caching/backends/rediscache.py", line 126, in dump_object
    return b'!' + pickle.dumps(value)
pickle.PicklingError: Can't pickle <class 'sqlalchemy.orm.session.SignallingSession'>: attribute lookup SignallingSession on sqlalchemy.orm.session failed
127.0.0.1 - - [02/Mar/2022 23:51:01] "GET /articles_by/thejeshgn2?__debugger__=yes&cmd=resource&f=style.css HTTP/1.1" 200 -
127.0.0.1 - - [02/Mar/2022 23:51:01] "GET /articles_by/thejeshgn2?__debugger__=yes&cmd=resource&f=debugger.js HTTP/1.1" 200 -
127.0.0.1 - - [02/Mar/2022 23:51:01] "GET /articles_by/thejeshgn2?__debugger__=yes&cmd=resource&f=console.png HTTP/1.1" 200 -
127.0.0.1 - - [02/Mar/2022 23:51:01] "GET /articles_by/thejeshgn2?__debugger__=yes&cmd=resource&f=console.png HTTP/1.1" 200 -
^C
thej@uma:~/code/work/flask-performance/code$
```

So, let us go back to the documentation, see what memorize means. Here, so in memorization, the function arguments are also included in the cache key. So, along with their name, it is the function parameters like whatever the actual value of username is also used as a key. So, that if, indirectly it is like for example, the key is like, roughly, it would be like this.

Roughly the key would be like this, and hence that would be unique for each of the query. And this, we do not need to do and the memorize automatically handles it. And 50 is a timeout period. So, let us try that also. Let us go back and add the similar time out for that. Timers for the same, going to remove login required for now?

Now let us say we click on this user name is not defined made is this. This is user underscore name, stop this, I think it is similar. So, the way caching does is it actually takes the output and pickle sit and stores in the Redis. So, sometimes, if the object cannot be pickled and put in, then there will be an issue. So, let us see, what is the issue here?

(Refer Slide Time: 26:39)



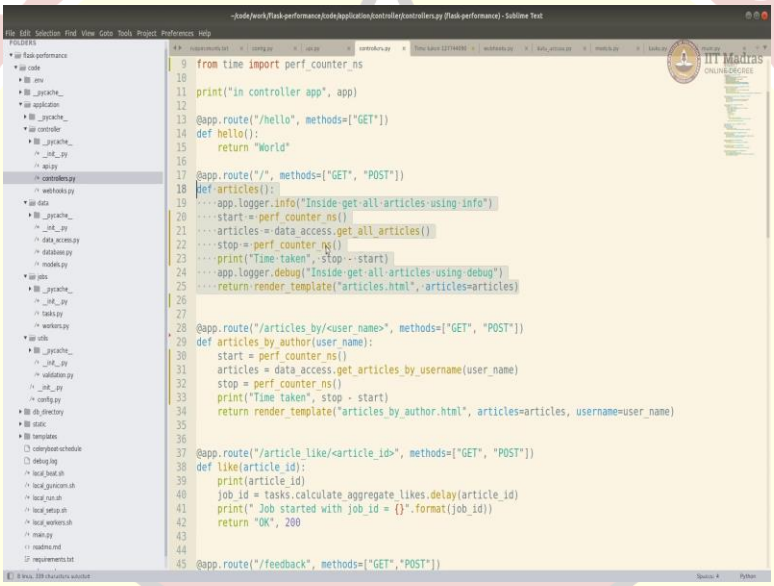
There was a small bug here. That is because in that access, we are actually returning the, the session and not the actual or it response of the query. So, I am just going to do dot all. So, it gets all the objects from the database, runs the queries and get dot all and then actually returns it. So, that result can be cached and not the reference. So, this will be actually the query. Now let us see if this works. Let us restart the server.

And there it works. And you can actually see it in the. See it in the Redis to by default reduce flash cache uses the database 0, if you do not give any, you can give the specific one, you can

refresh this to see. For example, this is the current one and it is been cached with a TTL, I think, which is reducing, if you keep reloading it, will keep changing reducing.

Once it expense it is actually query second from the database and get it. Let us go back to the homepage. And see, you can refresh this, you can see that, flash cache and get all articles is there. And it is quickly it is getting expired, the previous one got expired, and then this one is still valid for another 44 seconds. And even that is gone now. So, that is how you actually get the caching working on the backend in the database part.

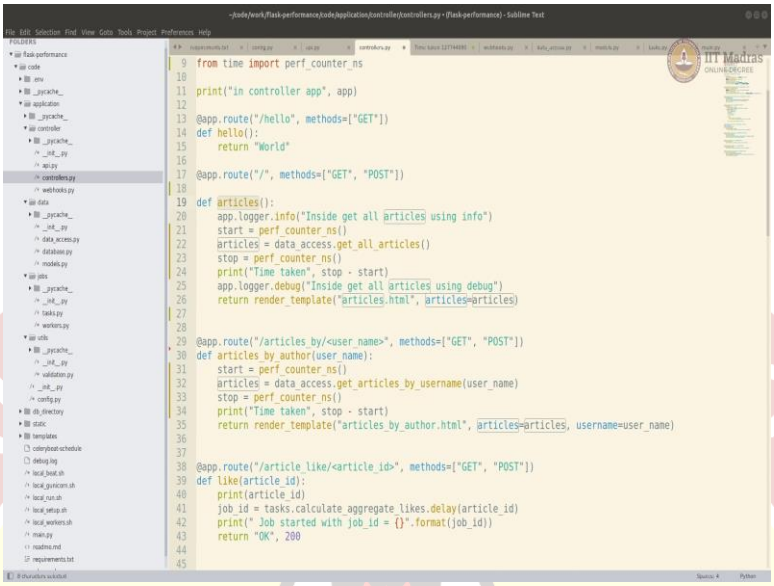
(Refer Slide Time: 28:38)

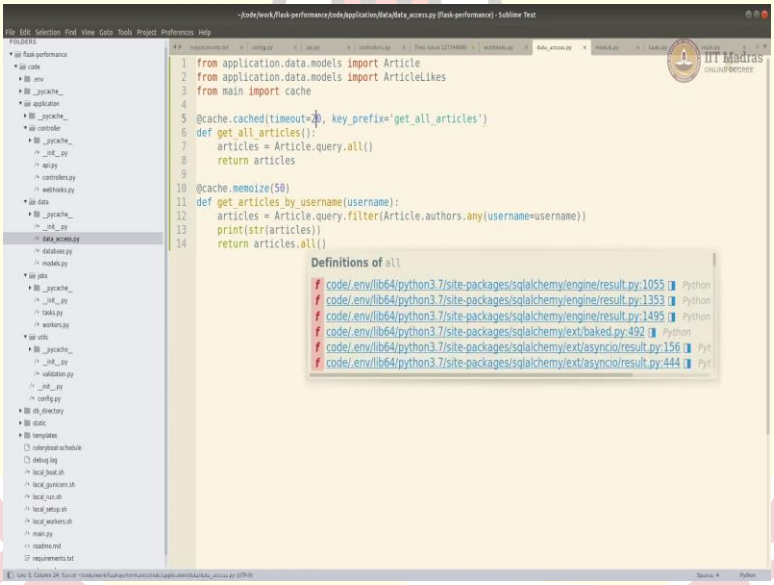


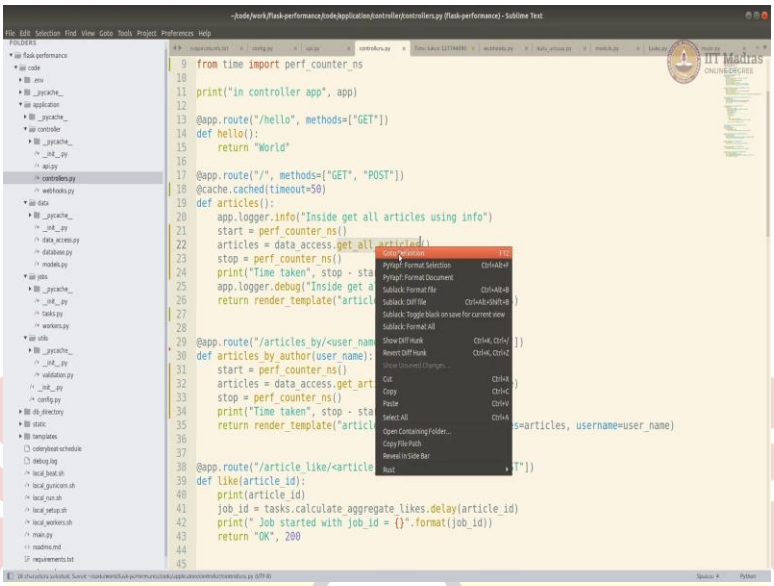
Now, you could also say that, this whole thing does not change that often. If the database, values are not changing. And if it is, it same for logged in users logged out users and all of it and you do not want to change in your this part. You want to respond with the same values regardless then you can cache it at the whole controller level. You can cache it at this level.

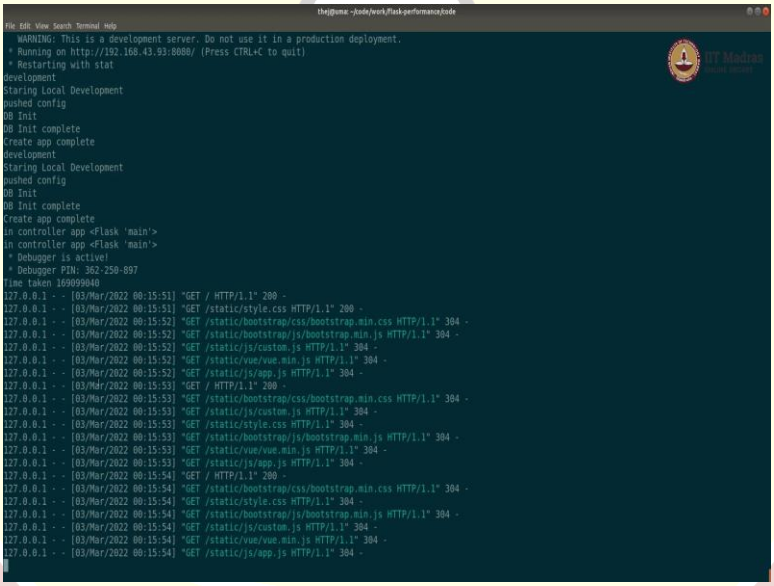
This will be really useful in an API where you want, the response is going to be same. And you can cache the whole response so that you will avoid database calling, marshaling, demarshaling all of that effort and just respond quickly. It can work authentication. No, it does not like it does not mean that just pick just for an example, I removed, authentication for one of the thing, because I did not want to log in. Otherwise, the caching also works with authentication.

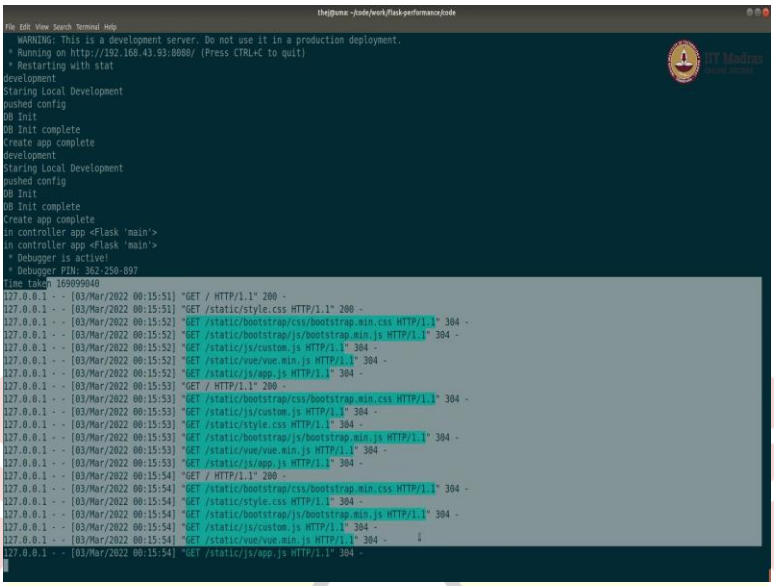
(Refer Slide Time: 29:40)

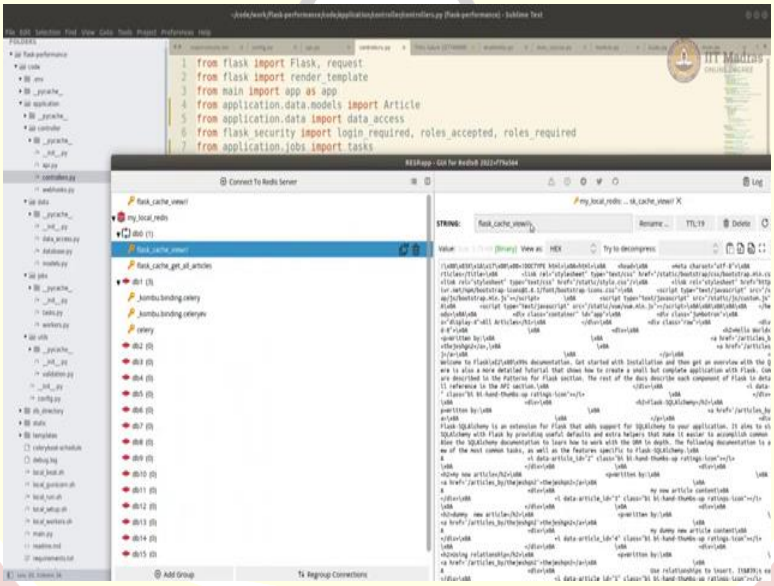





~/code/work/flask-performance/code/application/controller/controllers.py (flask-performance) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
flask-performance
code
.env
__pycache__
application
__pycache__
controller
__pycache__
__init__.py
api.py
controllers.py
webhooks.py
data
__pycache__
__init__.py
data_access.py
database.py
models.py
jobs
__pycache__
__init__.py
tasks.py
workers.py
utils
__pycache__
__init__.py
validation.py
__init__.py
config.py
db_directory
static
templates
celerybeat-schedule
debug.log
local_beat.sh
local_gunicorn.sh
local_run.sh
local_setup.sh
local_workers.sh
main.py
readme.md
requirements.txt
from time import perf_counter_ns
print("in controller app", app)
@app.route("/hello", methods=["GET"])
def hello():
return "World"
@app.route("/", methods=["GET", "POST"])
@cache.cached(timeout=50)
def articles():
app.logger.info("Inside get all articles using info")
start = perf_counter_ns()
articles = data_access.get_all_articles()
stop = perf_counter_ns()
print("Time taken", stop - sta
app.logger.debug("Inside get a
return render_template("articl
@app.route("/articles_by/<user_nam
def articles_by_author(user_name):
start = perf_counter_ns()
articles = data_access.get_art
stop = perf_counter_ns()
print("Time taken", stop - sta
return render_template("articl
@app.route("/article_like/<article
def like(article_id):
print(article_id)
job_id = tasks.calculate_aggregate_likes.delay(article_id)
print(" Job started with job_id = {}".format(job_id))
return "OK", 200
Goto Definition F12
PyYapf: Format Selection Ctrl+Alt+F
PyYapf: Format Document
Sublack: Format file Ctrl+Alt+B
Sublack: Diff file Ctrl+Alt+Shift+B
Sublack: Toggle black on save for current view
Sublack: Format All
Show Diff Hunk Ctrl+K, Ctrl+
Revert Diff Hunk Ctrl+K, Ctrl+Z
Show Unsaved Changes...
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Select All Ctrl+A
Open Containing Folder...
Copy File Path
Reveal in Side Bar
Rust
IIT Madras
ONLINE DEGREE


thej@sma: ~/code/work/flask-performance/code
File Edit View Search Terminal Help
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://192.168.43.93:8080/ (Press CTRL+C to quit)
* Restarting with stat
development
Staring Local Development
pushed config
DB Init
DB Init complete
Create app complete
development
Staring Local Development
pushed config
DB Init
DB Init complete
Create app complete
in controller app <Flask 'main'>
in controller app <Flask 'main'>
* Debugger is active!
* Debugger PIN: 362-250-897
Time taken 169099040
127.0.0.1 - - [03/Mar/2022 00:15:51] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [03/Mar/2022 00:15:51] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 - - [03/Mar/2022 00:15:52] "GET /static/bootstrap/css/bootstrap.min.css HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:52] "GET /static/bootstrap/js/bootstrap.min.js HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:52] "GET /static/js/custom.js HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:52] "GET /static/vue/vue.min.js HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:52] "GET /static/js/app.js HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:53] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [03/Mar/2022 00:15:53] "GET /static/bootstrap/css/bootstrap.min.css HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:53] "GET /static/js/custom.js HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:53] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:53] "GET /static/bootstrap/js/bootstrap.min.js HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:53] "GET /static/vue/vue.min.js HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:53] "GET /static/js/app.js HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:54] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [03/Mar/2022 00:15:54] "GET /static/bootstrap/css/bootstrap.min.css HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:54] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:54] "GET /static/bootstrap/js/bootstrap.min.js HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:54] "GET /static/js/custom.js HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:54] "GET /static/vue/vue.min.js HTTP/1.1" 304 -
127.0.0.1 - - [03/Mar/2022 00:15:54] "GET /static/js/app.js HTTP/1.1" 304 -
IIT Madras
ONLINE DEGREE



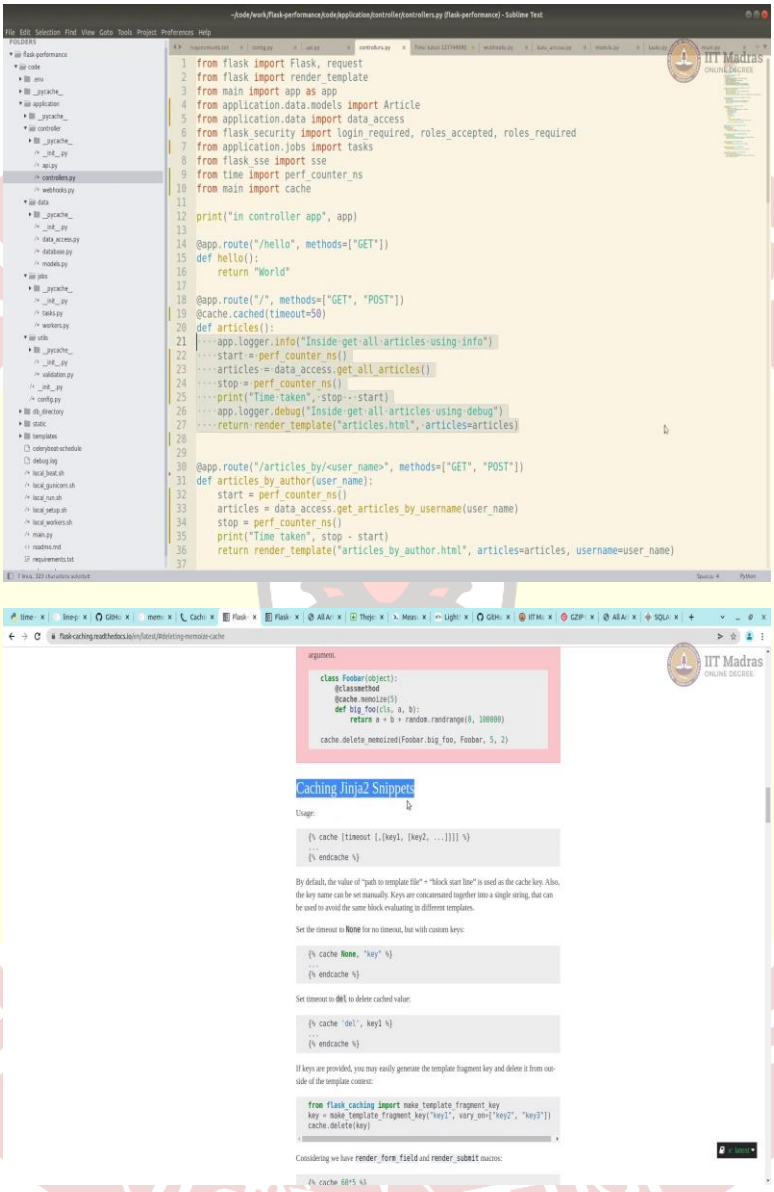


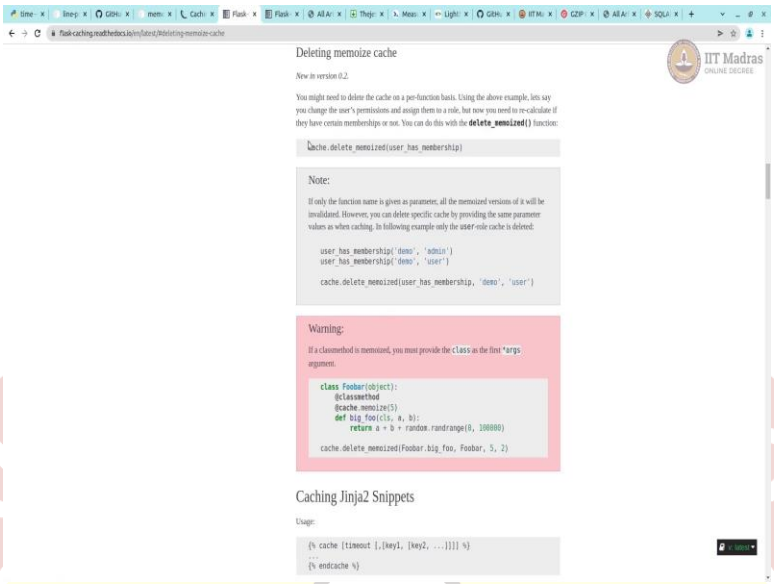
Now let us see here. Let us add a caching at the whole articles level. Let us check whether how to add it. So, it would be just that, so let us see if it caches for let us say, this is cache for 50 and let us say this is cache for much smaller time. You could be 20. And it imported in the controllers. Start this; now let me just go back to the browser. And if we do this, and this, this, you can see that the first time we see that time taken has been printed.

The rest of the times no time taken has been printed that because it does not even enter this. It responds directly from here, we can even see that the cache, you can see, let us refresh this and enable auto refresh as well see you can see the whole view has been slash view. That it is

showing slash the whole view, all HTML has been cached. So, that the whole time the takes to run this, render this, all of that gets saved, because we are just loading it from the cache.

(Refer Slide Time: 31:36)





Deleting memoize cache

New in version 0.2.

You might need to delete the cache on a per-function basis. Using the above example, lets say you change the user's permissions and assign them to a role, but now you need to re-calculate if they have certain memberships or not. You can do this with the **delete_memoized()** function:

```
cache.delete_memoized(user_has_membership)
```

Note:

If only the function name is given as parameter, all the memoized versions of it will be invalidated. However, you can delete specific cache by providing the same parameter values as when caching. In following example only the **user**-role cache is deleted:

```
user_has_membership('demo', 'admin')
user_has_membership('demo', 'user')

cache.delete_memoized(user_has_membership, 'demo', 'user')
```

Warning:

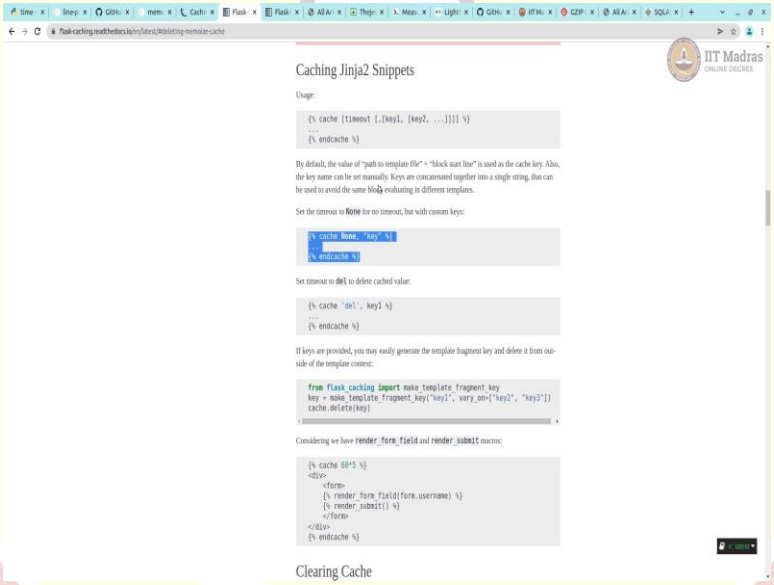
If a classmethod is memoized, you must provide the `class` as the first `*args` argument.

```
class Foobar(object):
    @classmethod
    @cache.memoize(5)
    def big_foo(cls, a, b):
        return a + b + random.randrange(0, 100000)

cache.delete_memoized(Foobar.big_foo, Foobar, 5, 2)
```

Caching Jinja2 Snippets

Usage:

```
{% cache [timeout [,[key1, [key2, ...]]]] %}
...
{% endcache %}
```



Caching Jinja2 Snippets

Usage:

```
{% cache [timeout [,[key1, [key2, ...]]]] %}
...
{% endcache %}
```

By default, the value of "path to template file" + "block start line" is used as the cache key. Also, the key name can be set manually. Keys are concatenated together into a single string, that can be used to avoid the same block evaluating in different templates.

Set the timeout to **None** for no timeout, but with custom keys:

```
{% cache None, "key" %}
...
{% endcache %}
```

Set timeout to **del** to delete cached value:

```
{% cache 'del', key1 %}
...
{% endcache %}
```

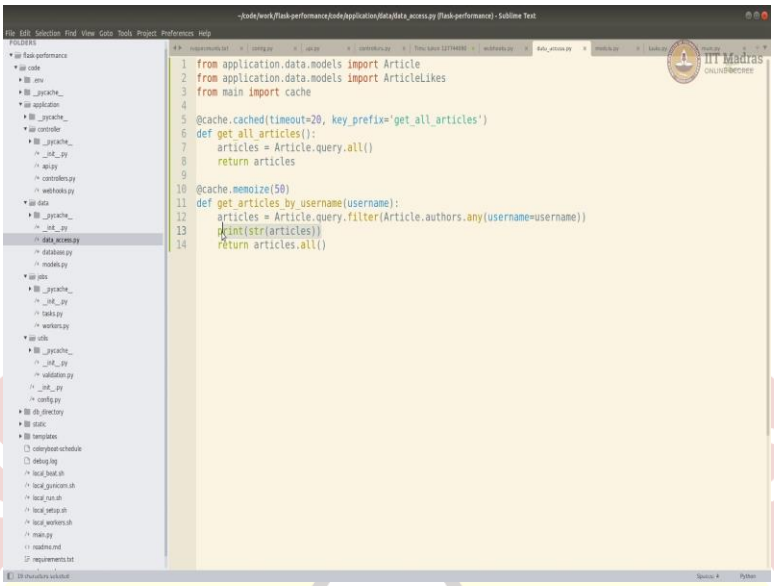
If keys are provided, you may easily generate the template fragment key and delete it from outside of the template context:

```
from flask_caching import make_template_fragment_key
key = make_template_fragment_key("key1", vary_on=["key2", "key3"])
cache.delete(key)
```

Considering we have `render_form_field` and `render_submit` macros:

```
{% cache 60*5 %}
<div>
    <form>
    {% render_form_field(form.username) %}
    {% render_submit() %}
    </form>
</div>
{% endcache %}
```

Clearing Cache

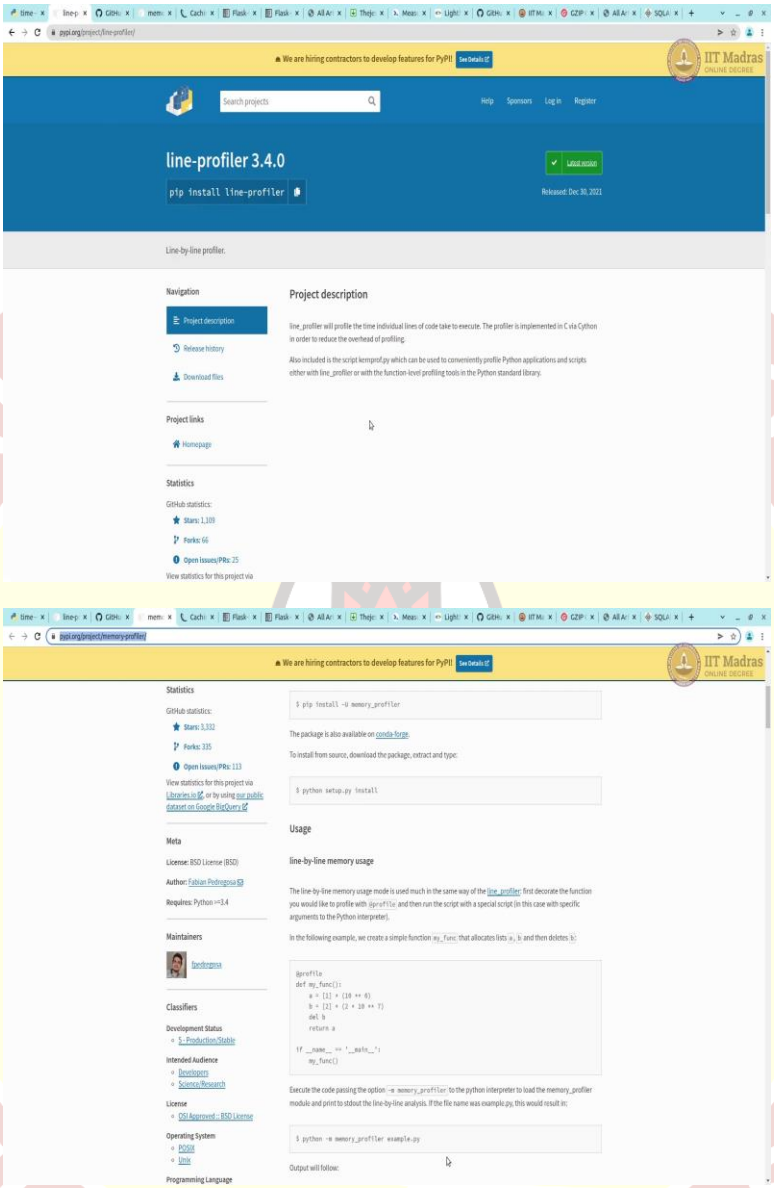


Now, you can also time out the caches when you want, if you want to immediately clear, for some reason, for example, you have a long term cache for get reading the things. But when you post something, you want to expire, the cache. That also you can do by expiring, the cache or deleting the cache, basically, you could it same function as deleting this from here and that would expire, the cache. Or you could have a function called expire or delete, let us just check, you can delete the memorized cache, or you can delete by key as well. Both of them will work.

Now, you can also use caching within Jinja2 template if you want to cache just like how in controller, the cache only small part of it. If this was not used, we are caching only this part. Similarly, within the whole article dot HTML, if you want to cache only small parts of it, you could cache that as well, you can see that, in the tutorial, I am not going to show how to do that, you can see it in the tutorial how to do it.

Now this is one step. And the most often used step to improve the performance. You also saw that I am printing the queries here, when I ran, and you could use explain on the database, to get explanation on the query, how it runs, and then you can add indexes and stuff. I am not going to take you through that because I think you would have learned that as part of you DBMS. Similarly, you could also do a memory.

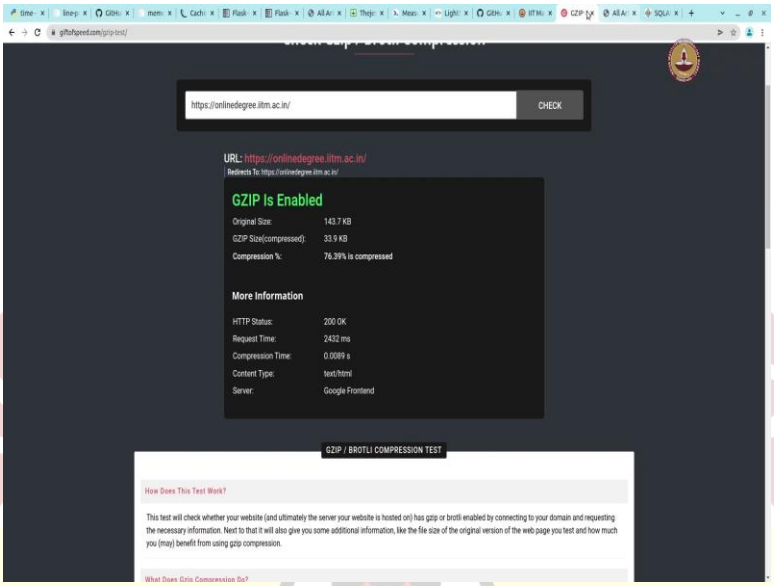
(Refer Slide Time: 33:27)

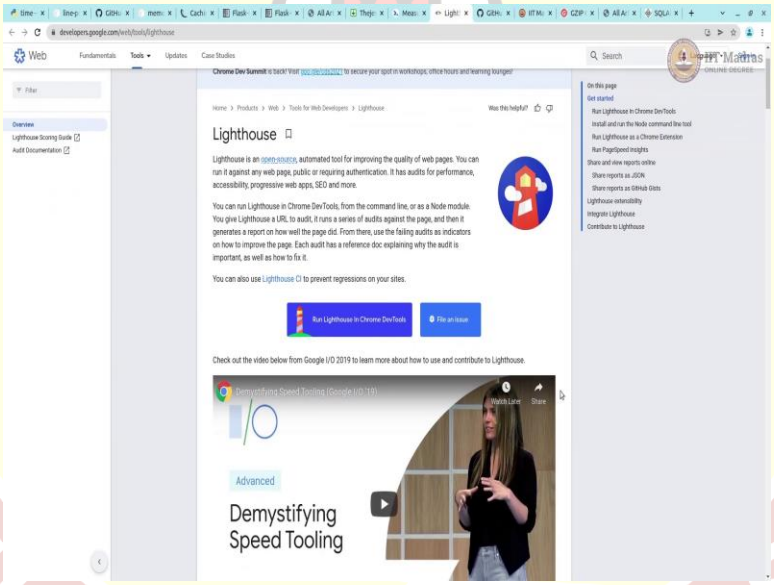


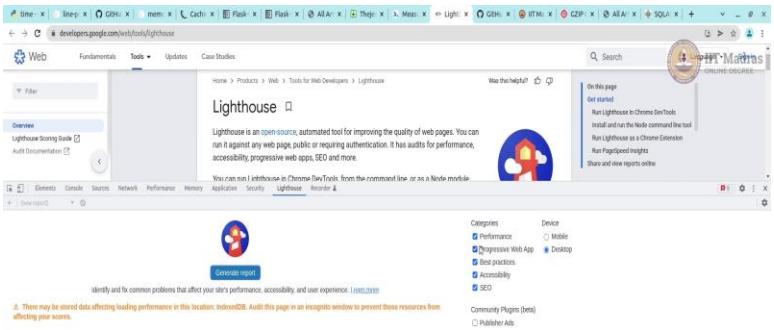
Or line profiling, line profiling or memory profiling at the Python level to check, how much memory it is getting used, are there any objects that is not been garbage collected, etcetera, etcetera. To improve the performance of the Python program, this is similar to any other, Python program it does not have to be just for flask.

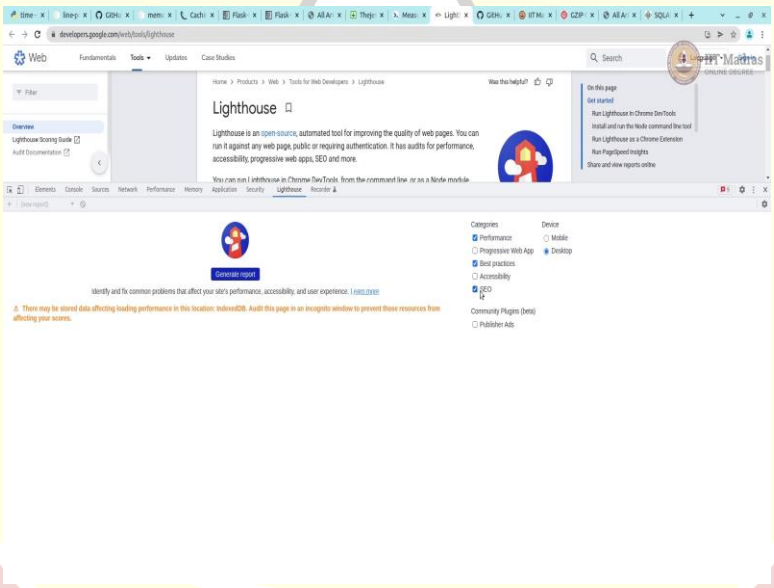
If you can do this with your regular Python program, you can do in a similar way for flask app 2. There, I am going to just stop the backend optimization, and I am going to show how to measure the front end performance using something called Lighthouse.

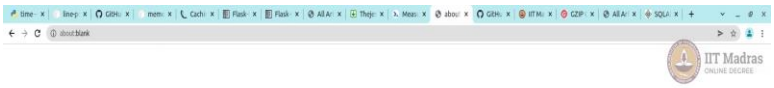
(Refer Slide Time: 34:14)

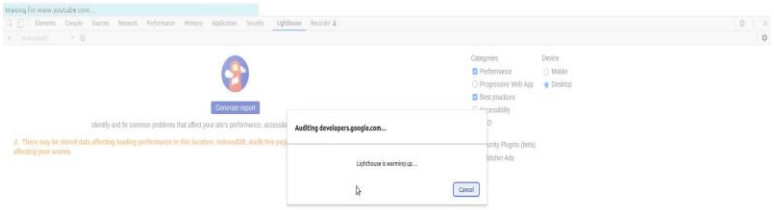


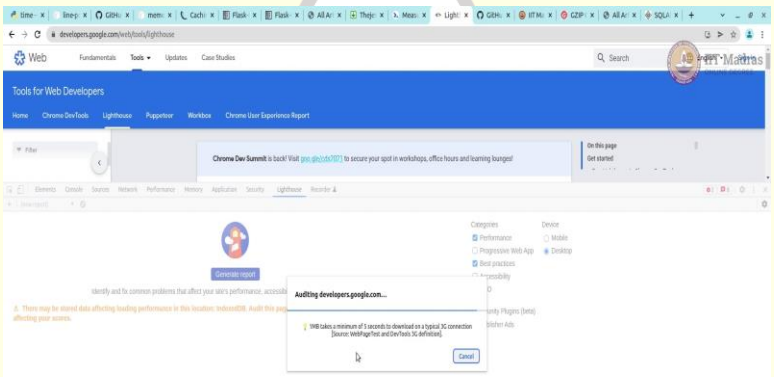


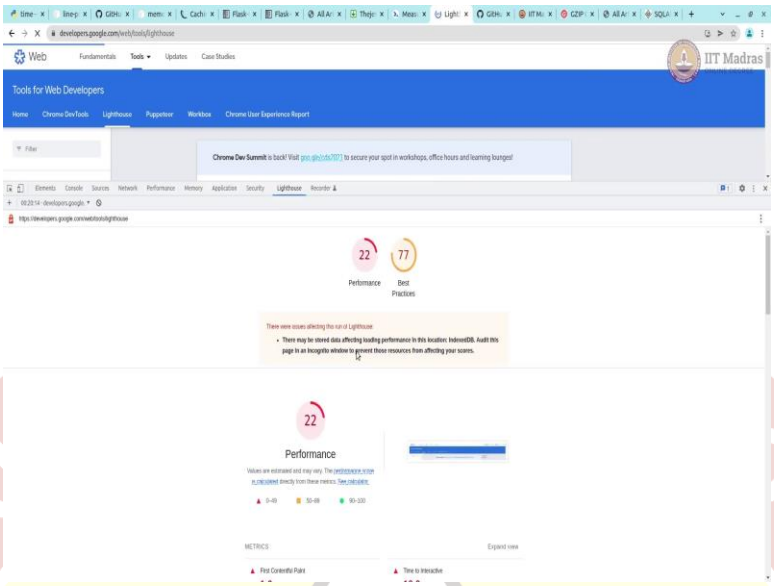

developers.google.com/web/tools/lighthouse
Web
Fundamentals
Tools
Updates
Case Studies
Search
Lighthouse
Lighthouse is an open-source, automated tool for improving the quality of web pages. You can run it against any web page, public or requiring authentication. It has audits for performance, accessibility, progressive web apps, SEO and more.
Generate report
Categories
Performance
Progressive Web App
Best practices
Accessibility
SEO
Device
Mobile
Desktop
Community Plugins (beta)
Publisher Ads


developers.google.com/web/tools/lighthouse
Web
Fundamentals
Tools
Updates
Case Studies
Search
Lighthouse
Lighthouse is an open-source, automated tool for improving the quality of web pages. You can run it against any web page, public or requiring authentication. It has audits for performance, accessibility, progressive web apps, SEO and more.
Generate report
Categories
Performance
Progressive Web App
Best practices
Accessibility
SEO
Device
Mobile
Desktop
Community Plugins (beta)
Publisher Ads


about:blank


Auditing developers.google.com...
Lighthouse is warming up...
Cancel

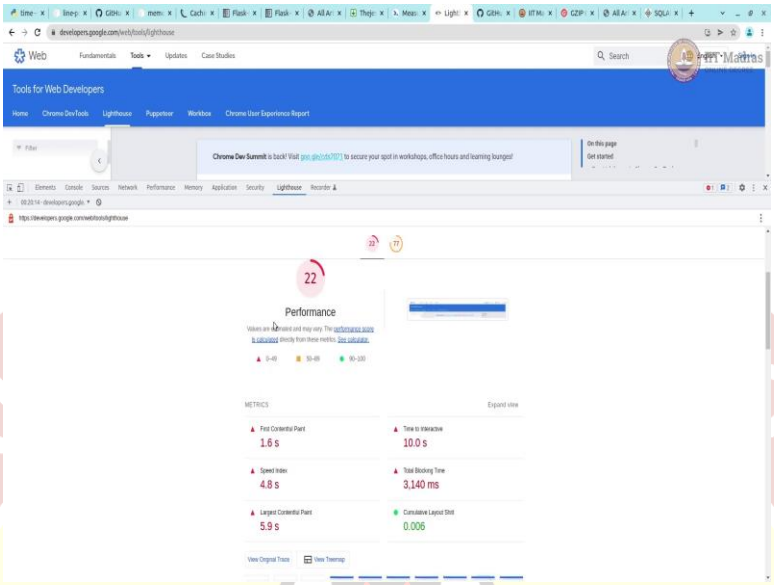

developers.google.com/web/tools/lighthouse
Tools for Web Developers
Auditing developers.google.com...
Cancel

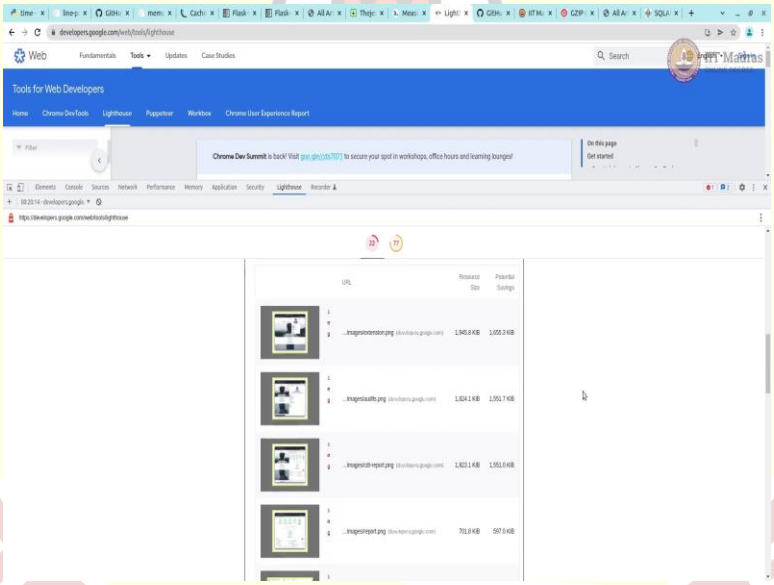


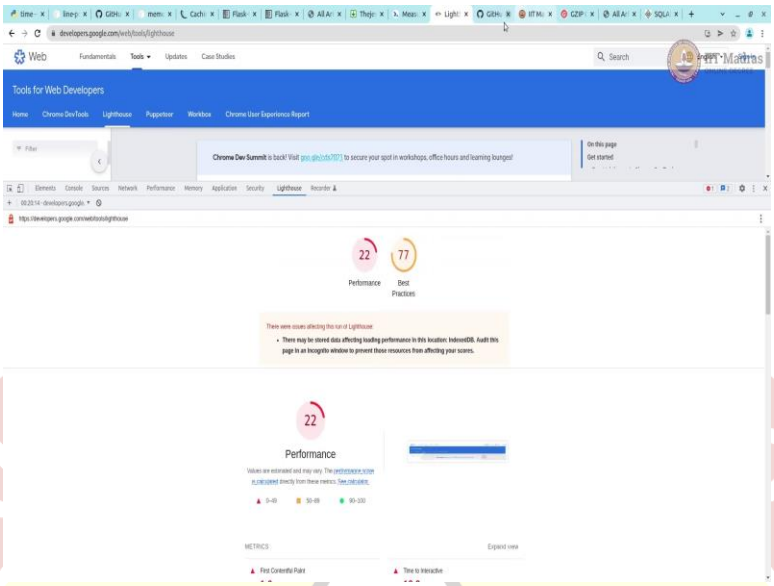
Now lighthouse is a project. Maybe I am not let us is an open source project by Google or for measuring performance. It is also been integrated into the Chrome. So, if you are in right click and click on Inspect, you can go to the lighthouse and it will show Chrome. And then you can choose the categories, I will probably choose.

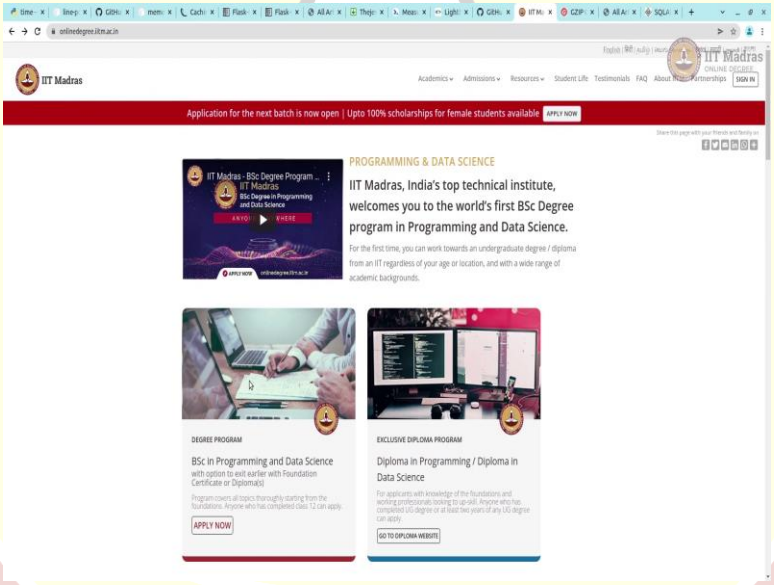
You can do Progressive Web App checks, Performance checks, Best practices, Accessibility, and SEO checks. I am just going to check because there is this is not a, I am going to just check these three. Click on this and generate report will take couple of minutes to run, gives you a report or some instructions on how to improve any performance related issue or the best practice related issue. It is done. So, here you can see the performance is 22 and best practice is 77.

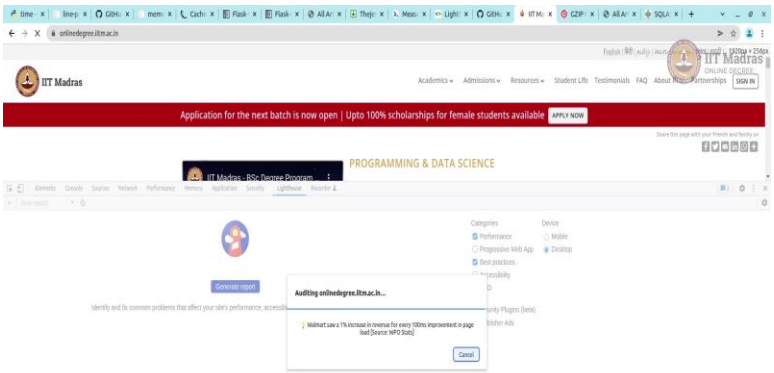
(Refer Slide Time: 35:35)

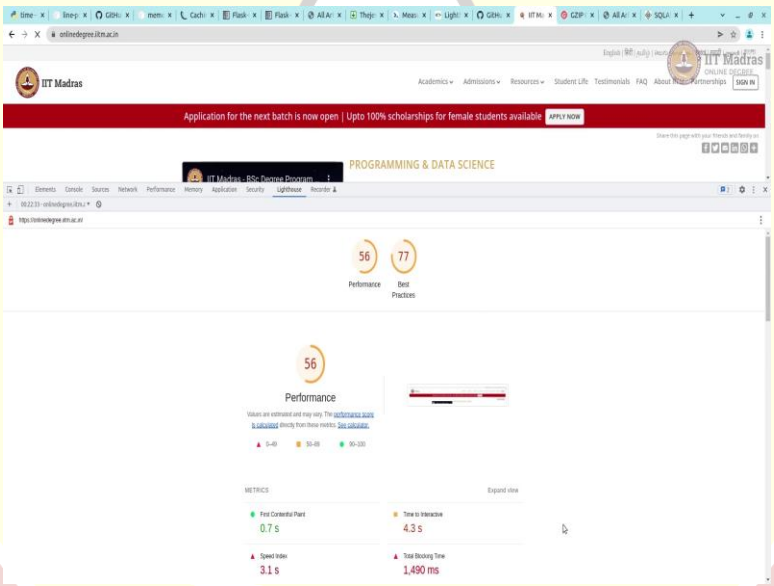


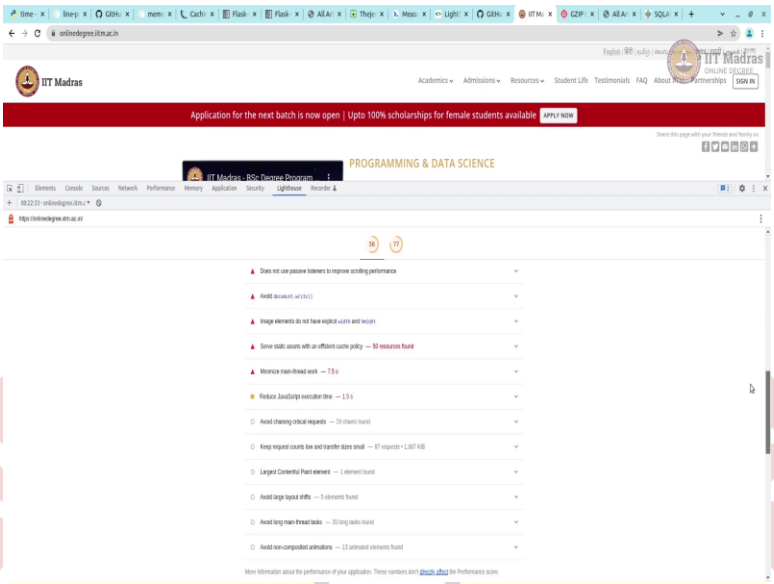


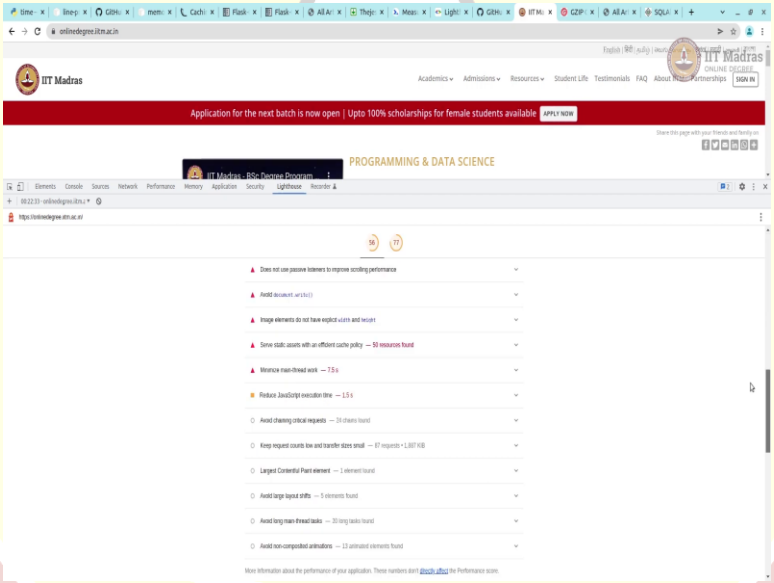

developers.google.com/web/tools/lighthouse
Web
Fundamentals
Tools
Updates
Case Studies
Search
Tools for Web Developers
Home
Chrome DevTools
Lighthouse
Puppeteer
Workbox
Chrome User Experience Report
Chrome Dev Summit is back! Visit goo.gle/cds2017 to secure your spot in workshops, office hours and learning lounges!
On this page
Get started
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
https://developers.google.com/web/tools/lighthouse
22
77
Performance
Best Practices
There were issues affecting this run of Lighthouse
There may be stored data affecting loading performance in this location: IndexedDB. Audit this page in an incognito window to prevent those resources from affecting your scores.
22
Performance
METRICS
Expand view
First Contentful Paint
Time to Interactive

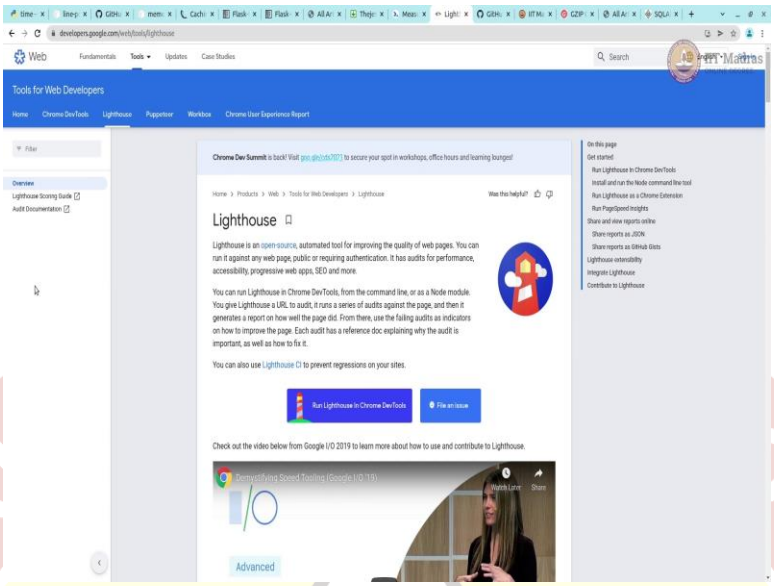

onlinedegree.iitm.ac.in
IIT Madras
Academics
Admissions
Resources
Student Life
Testimonials
FAQ
Partnerships
SIGN IN
Application for the next batch is now open | Upto 100% scholarships for female students available
APPLY NOW
IIT Madras - BSc Degree Program ...
IIT Madras
BSc Degree in Programming and Data Science
PROGRAMMING & DATA SCIENCE
IIT Madras, India's top technical institute, welcomes you to the world's first BSc Degree program in Programming and Data Science.
For the first time, you can work towards an undergraduate degree / diploma from an IIT regardless of your age or location, and with a wide range of academic backgrounds.
DEGREE PROGRAM
BSc in Programming and Data Science
with option to exit earlier with Foundation Certificate or Diploma(s)
APPLY NOW
EXCLUSIVE DIPLOMA PROGRAM
Diploma in Programming / Diploma in Data Science
GO TO DIPLOMA WEBSITE


IIT Madras
Application for the next batch is now open | Upto 100% scholarships for female students available
APPLY NOW
PROGRAMMING & DATA SCIENCE
Auditing onlinedegree.iitm.ac.in...
Cancel


IIT Madras
Application for the next batch is now open | Upto 100% scholarships for female students available
APPLY NOW
PROGRAMMING & DATA SCIENCE
56
Performance
77
Best Practices
56
Performance
METRICS
First Contentful Paint
0.7 s
Time to Interactive
4.3 s
Speed Index
3.1 s
Total Blocking Time
1,490 ms


onlinedegree.iitm.ac.in
IIT Madras
Academics
Admissions
Resources
Student Life
Testimonials
FAQ
About Us
Partnerships
SIGN IN
Application for the next batch is now open | Upto 100% scholarships for female students available
APPLY NOW
PROGRAMMING & DATA SCIENCE
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Does not use passive listeners to improve scrolling performance
Avoid document.write()
Image elements do not have explicit width and height
Serve static assets with an efficient cache policy — 50 resources found
Minimize main-thread work — 7.5 s
Reduce JavaScript execution time — 1.5 s
Avoid chaining critical requests — 26 chains found
Keep request counts low and transfer sizes small — 87 requests • 1,807 KiB
Largest Contentful Paint element — 1 element found
Avoid large layout shifts — 5 elements found
Avoid long main-thread tasks — 30 long tasks found
Avoid non-composited animations — 13 animated elements found


onlinedegree.iitm.ac.in
IIT Madras
Academics
Admissions
Resources
Student Life
Testimonials
FAQ
About Us
Partnerships
SIGN IN
Application for the next batch is now open | Upto 100% scholarships for female students available
APPLY NOW
PROGRAMMING & DATA SCIENCE
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Does not use passive listeners to improve scrolling performance
Avoid document.write()
Image elements do not have explicit width and height
Serve static assets with an efficient cache policy — 50 resources found
Minimize main-thread work — 7.5 s
Reduce JavaScript execution time — 1.5 s
Avoid chaining critical requests — 26 chains found
Keep request counts low and transfer sizes small — 87 requests • 1,807 KiB
Largest Contentful Paint element — 1 element found
Avoid large layout shifts — 5 elements found
Avoid long main-thread tasks — 30 long tasks found
Avoid non-composited animations — 13 animated elements found



It also shows, if it is in red, it is actually quite bad, yellow is and green is great, shows some of the values how much each one, took and what it means like the image size is big, you could reduce the image size to improve the loading time, for example, that is one of the inputs that you can use. Format of the images, for example, I think most of them are using png here.

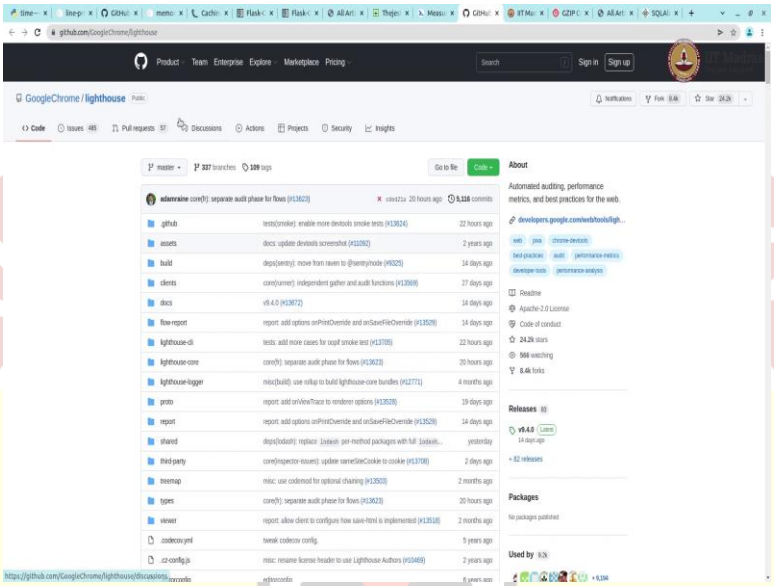
You could use other formats like, WebP, especially when you are delivering to Chrome to improve the speed. Many other I think you can go through them and see to work on some of them, you do not need to work on all of them to work, you can work on some of the time, which actually brings a larger change.

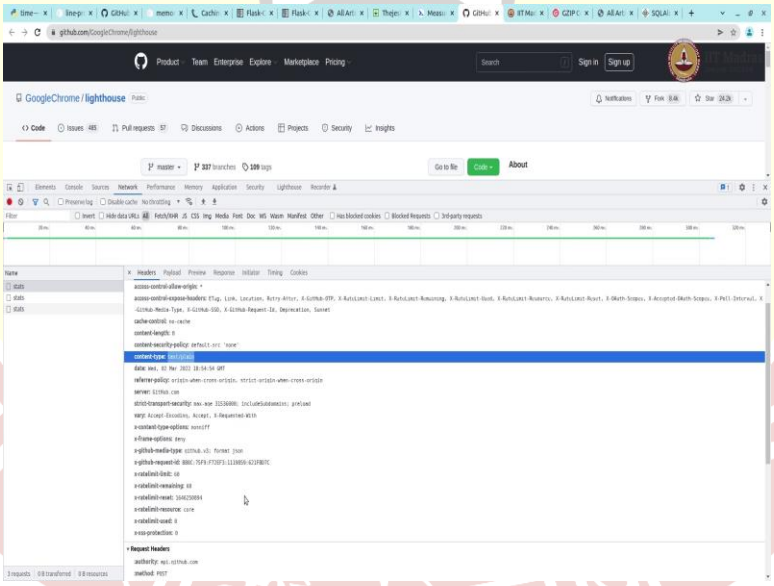
And you can improve this, for example, in our IIT degree site go to the lighthouse, I am just going to say the same thing, click on Generate Report, you can say that performance is 56. And the best practice is 77. Sometimes it also depends on the speed of the Internet. I mean, if your speed is or the browser is slow, it could reduce.

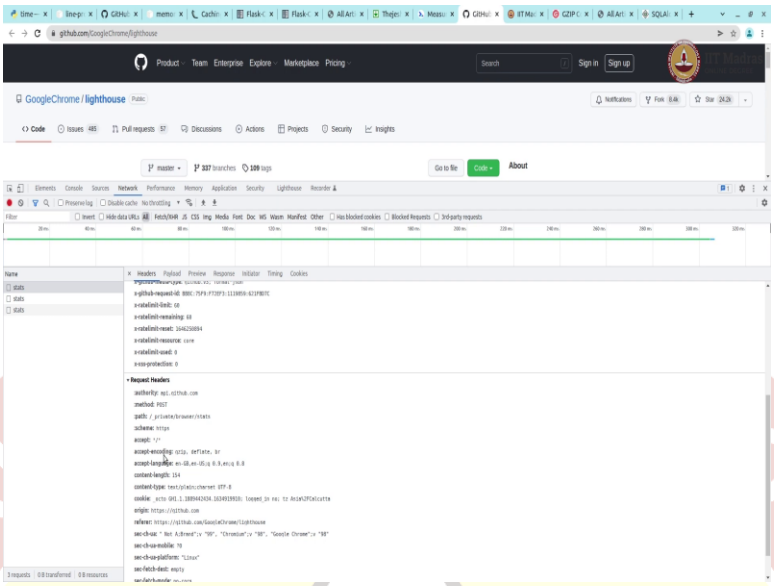
You can run it multiple times to take an average of how the performance is, it does shows like, you could I am using third party, for example, YouTube Embedded and I could reduce it to make it faster, things like that. Some of them, you cannot do anything, but some of the others things you can work on them to improve, the tips are actually good, you should go through them and improve. If you are interested in accessibility, you should run the accessibility test as well.

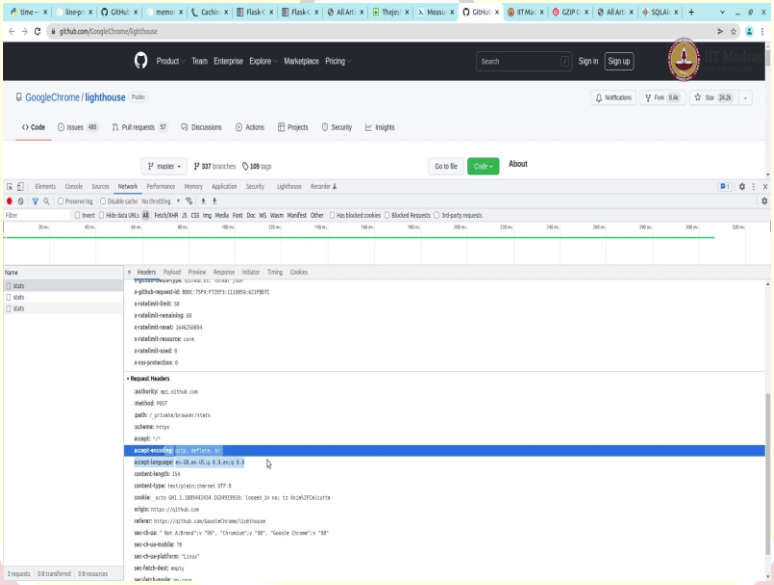
The lighthouse site itself is actually quite useful, you can check the scoring guide and tips how to improve the, scoring and that is very useful.

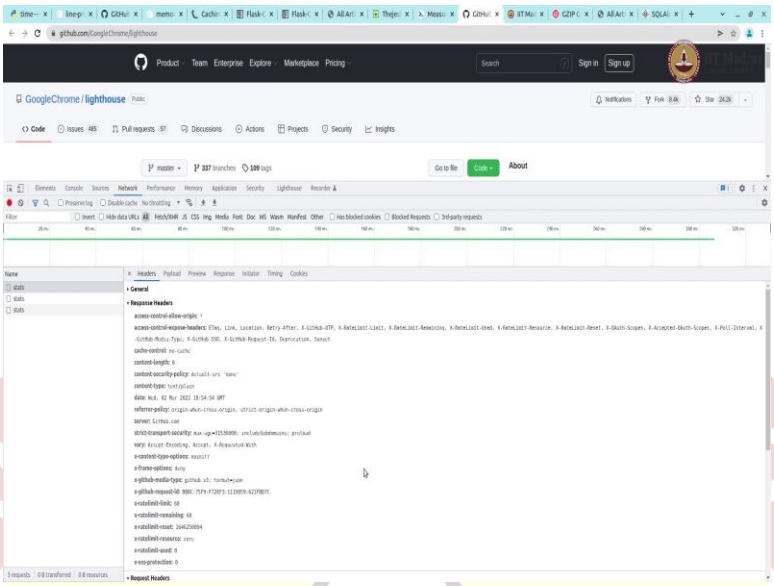
(Refer Slide Time: 38:01)







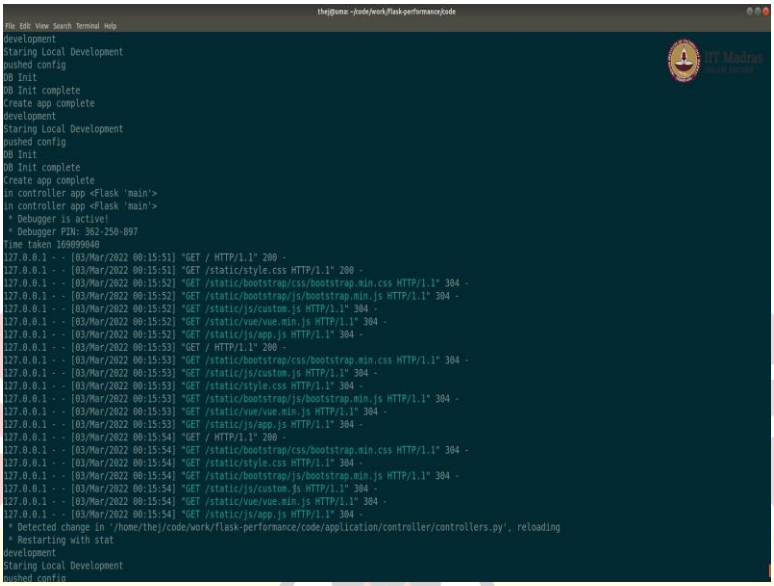


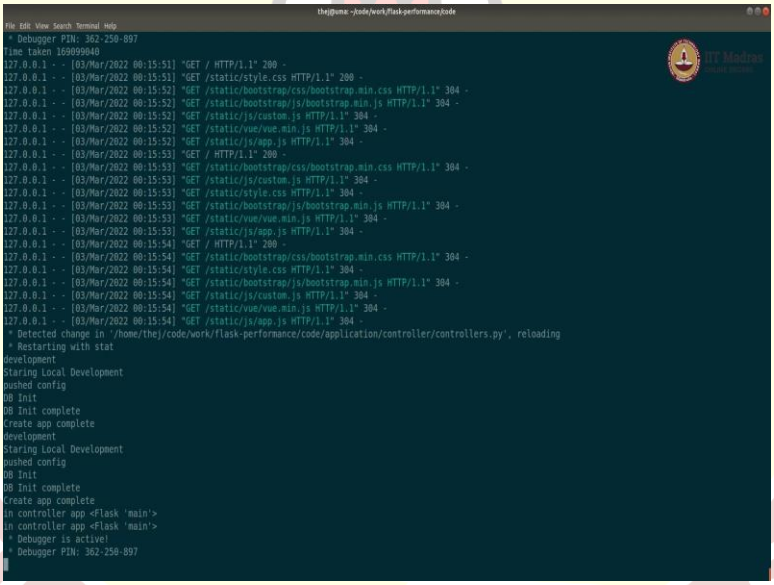


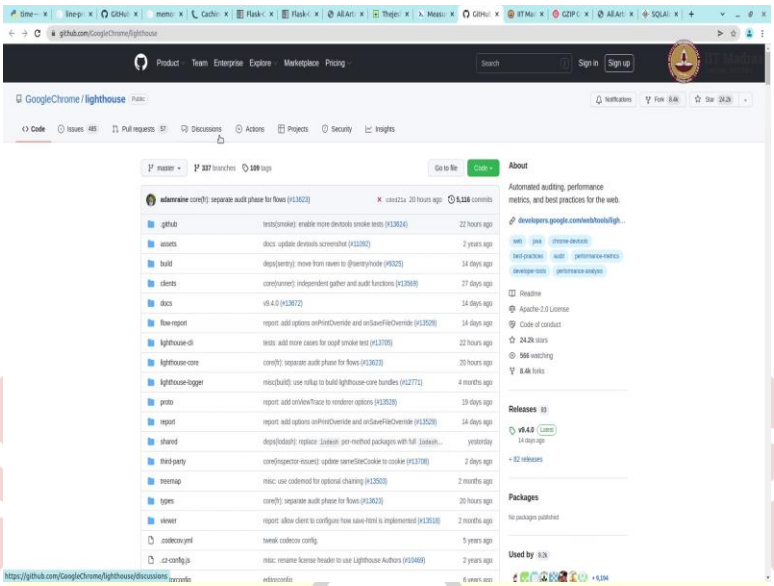
The other thing I would want you to look at this whether the gzipped caching is enabled for any site for most of the requests, you can look at the network. For example, if we look at the network, it shows multiple response headers, and also shows whether the gzipped is enabled on an disabled. gzipped is a way of compressing the content in text plain or text, HTML, compressed on the server side and sent gzipped basically.

And send to the client and the client like browser like Chrome can decrypt it and show to the user; that way the bandwidth that is used on the transport is very less and it will be much faster. So, it can, you can, when you send a request, you can say that or your browser actually sees that it can accept gzipped. And most of the time servers will respond with a gzipped content.

(Refer Slide Time: 39:15)






GoogleChrome / lighthouse
Code
Issues 485
Pull requests 57
Discussions
Actions
Projects
Security
Insights
About
Automated auditing, performance metrics, and best practices for the web.
Releases 83
Packages
Used by 9.2k

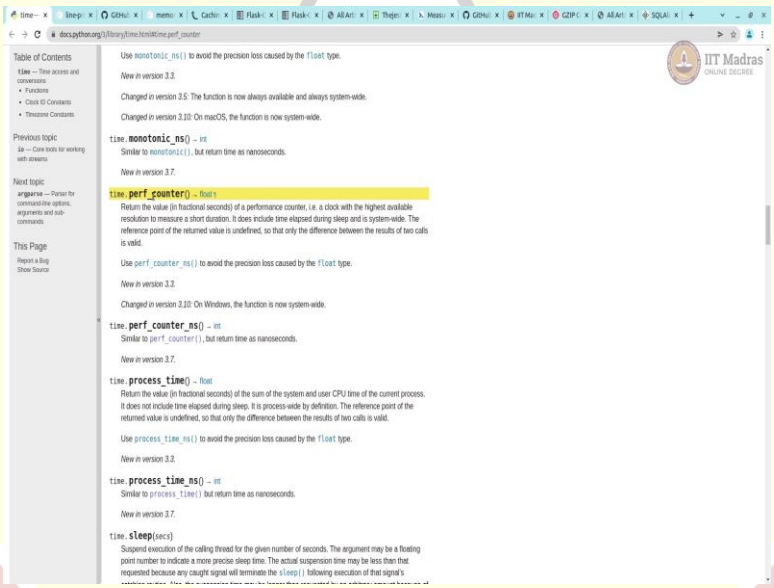
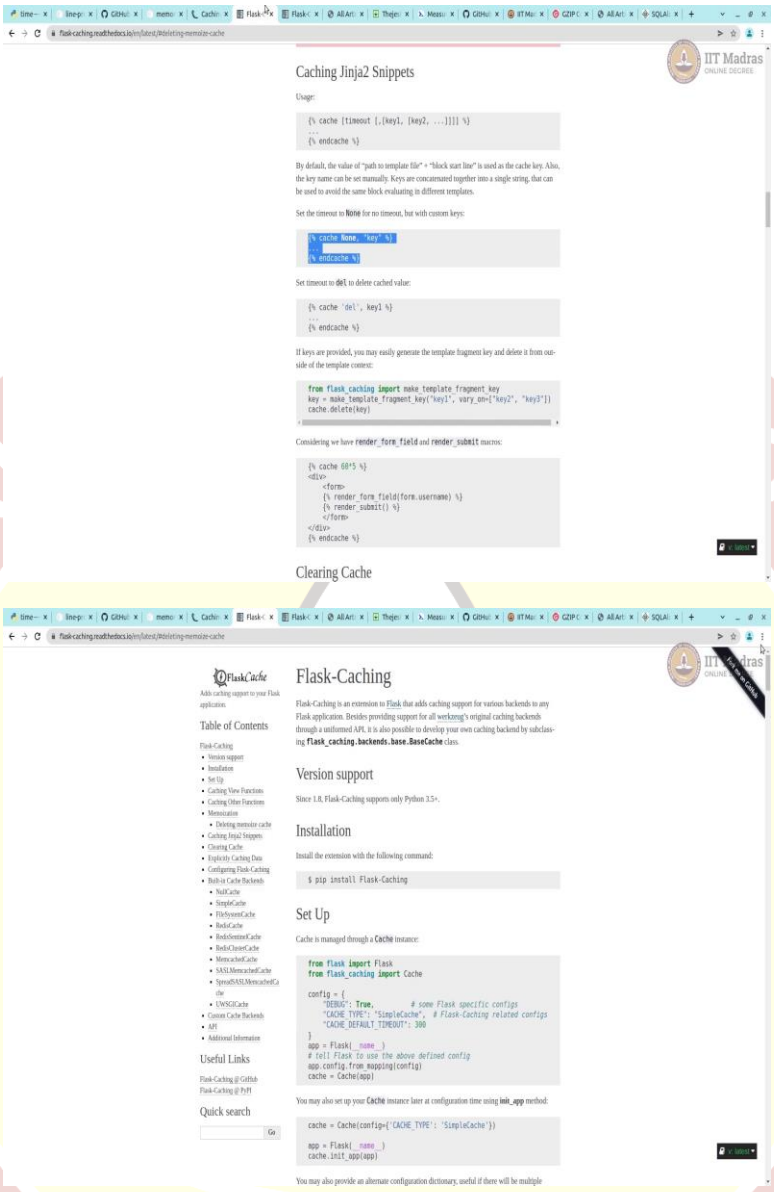


Table of Contents
Previous topic
Next topic
This Page
time.monotonic_ns() → int
time.perf_counter() → float
time.perf_counter_ns() → int
time.process_time() → float
time.process_time_ns() → int
time.sleep(secs)



This may not work with, our development server but it will surely work with the proxy like, nginx that we shown in the last deployment process and then you can enable the gzipped. Those are the few things that you can do to improve the performance. And again, this is like much vast subject and we can keep looking at more and more ways to improve the performance.

The first thing is to identify where the bottleneck and I would like you to add, Performance Counters everywhere that is possible and use cache content as much as possible. Because it is, one, it is easy to implement and easy to use. And two, it brings a lot of value in the sense that it makes your application really fast without much effort. Thank you for watching, that is all.