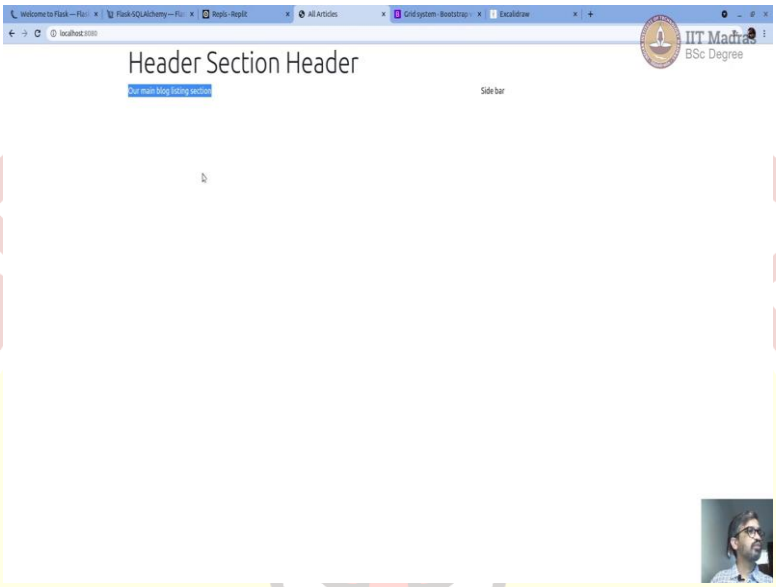
IIT Madras

ONLINE DEGREE

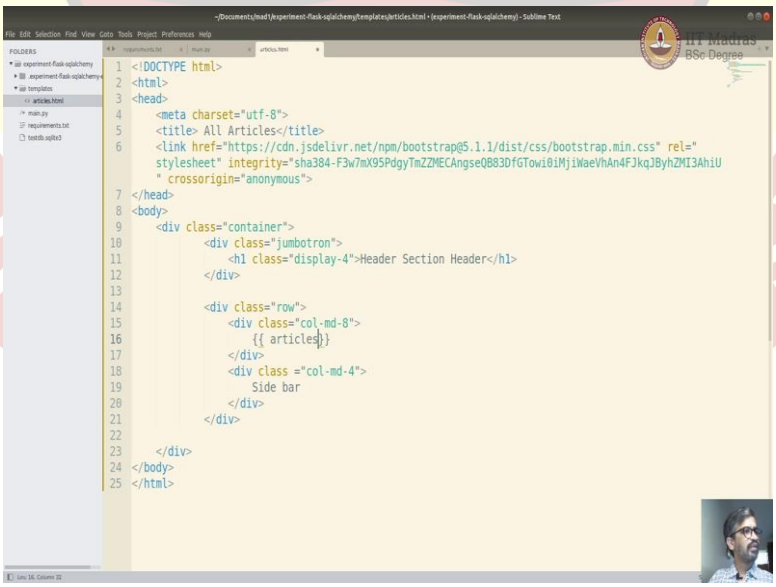
Modern Application Development - I
Professor Thejesh G N
Software Consultant
Indian Institute of Technology Madras
Introduction to Flask-SQLAlchemy - II

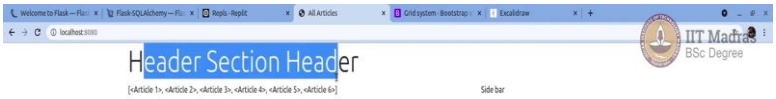
(Refer Slide Time: 00:19)



Now, in this board, block section, we want a list of blocks with their content.

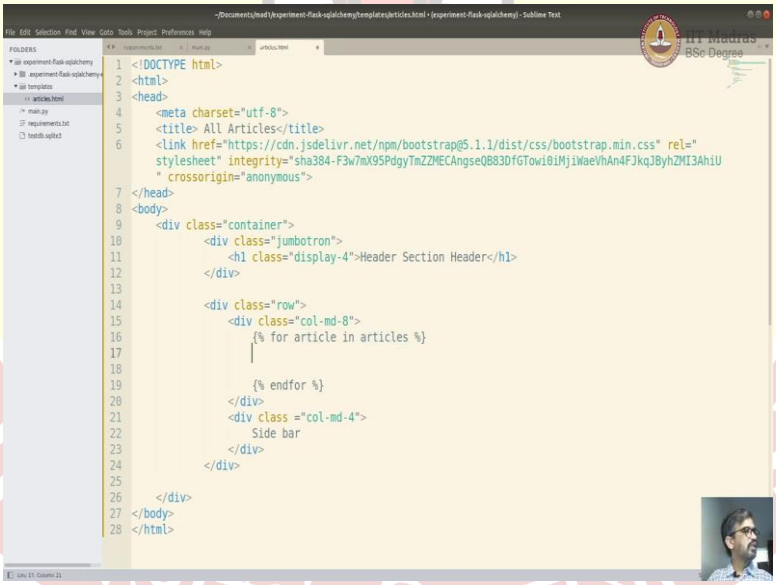
(Refer Slide Time: 00:25)





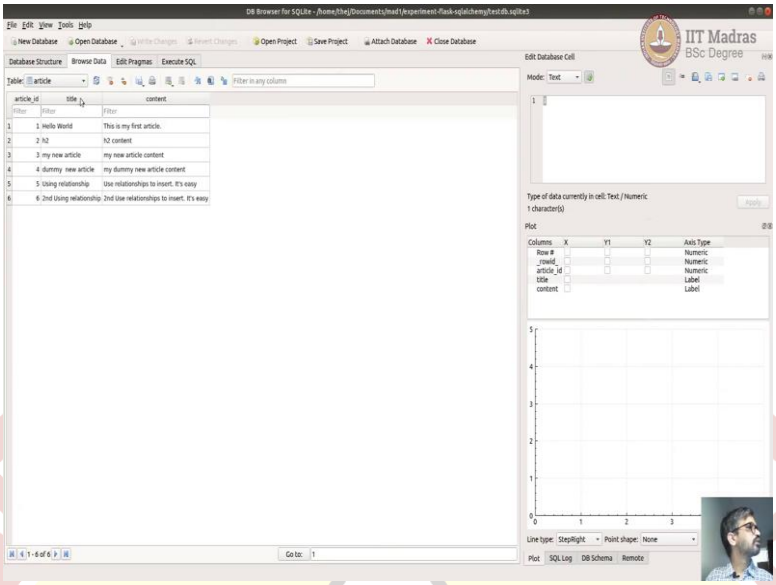
So let us, how do we do that? That should come from this our articles. Like if I make it like this you can see that there is a list. Now I have to iterate through that list and display everything.

(Refer Slide Time: 00:45)



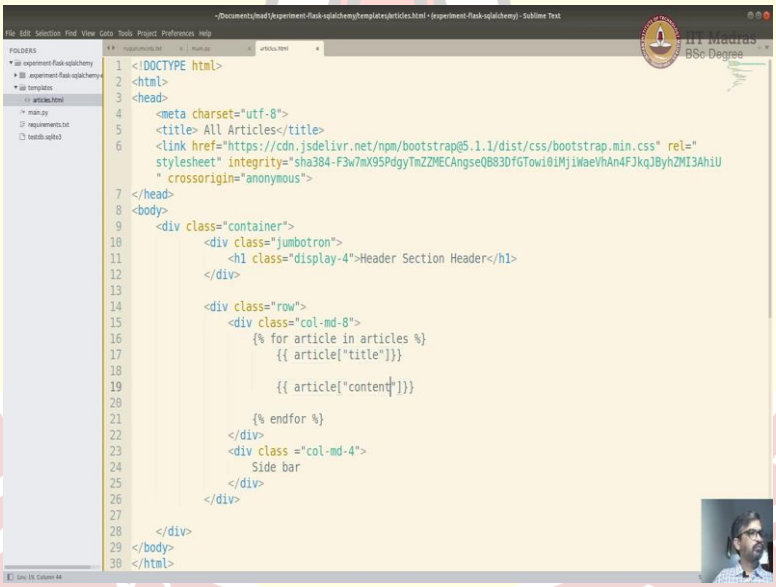
Now that is done by iterating through articles. You would already seen it. You can just use a For loop for the (())(0:53), for loop and then and iterate through the list. Now we are iterating through articles and getting an article. Now I can just print article title and article body.

(Refer Slide Time: 01:10)



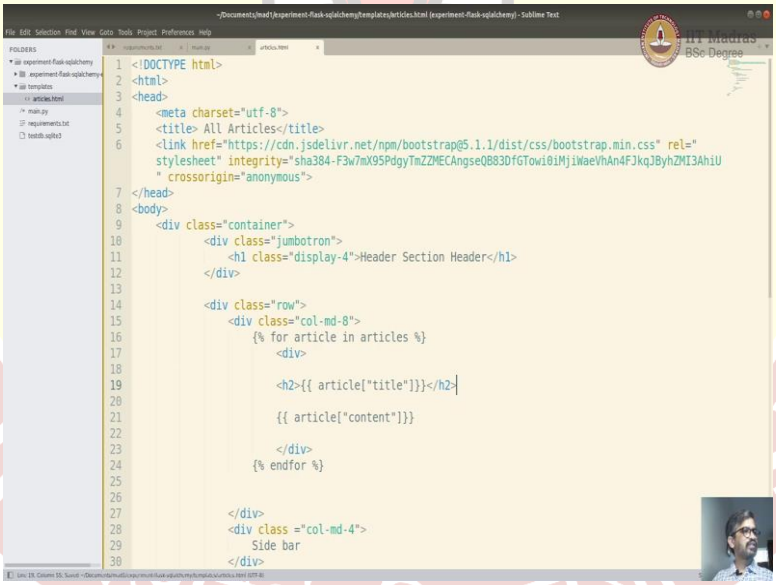
Let us see in the column name. So it is title and content.

(Refer Slide Time: 01:12)



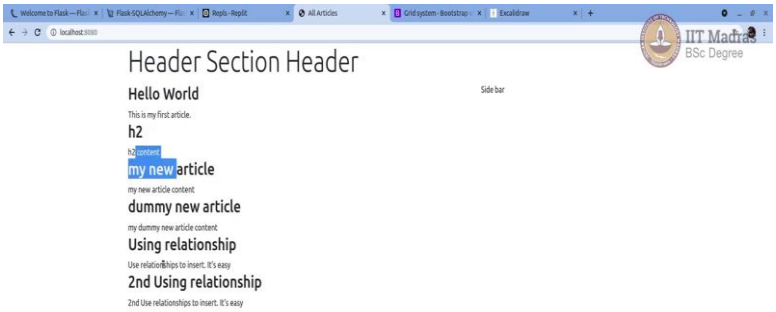
So I can just do title, actually it will be articles title, so article title, and then article content. Let me just go back here, refresh this. So it is coming but they are all coming in one line. It is, it is not great. So, let us add some design.

(Refer Slide Time: 01:45)



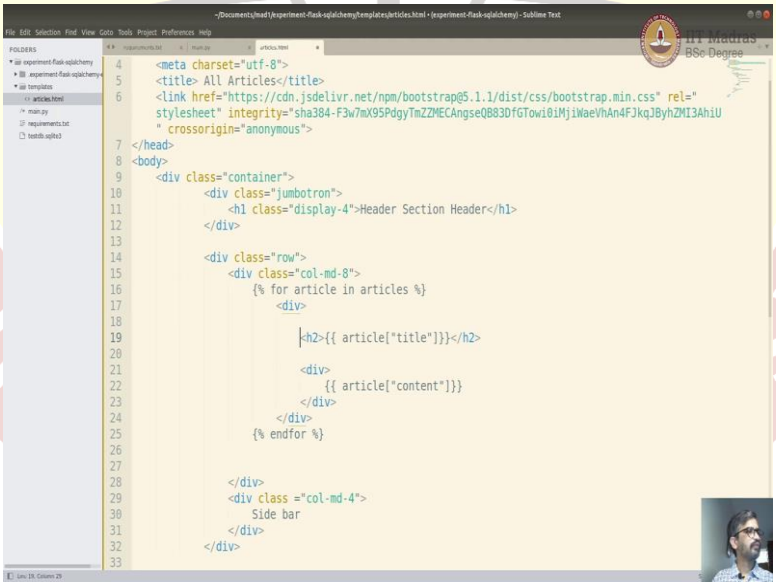
Let us say I want the whole thing to be within a div just to cover this whole section if I want to at some point, give some style, I would write, actually keep, want to keep it within a div. So I will add a div just before an article starts, actually article does not start here, article starts here. So be a div. Inside the div, we have our title and content. Let us say we want the title, make the title as h2, header. Now let us see how it looks.

(Refer Slide Time: 02:30)



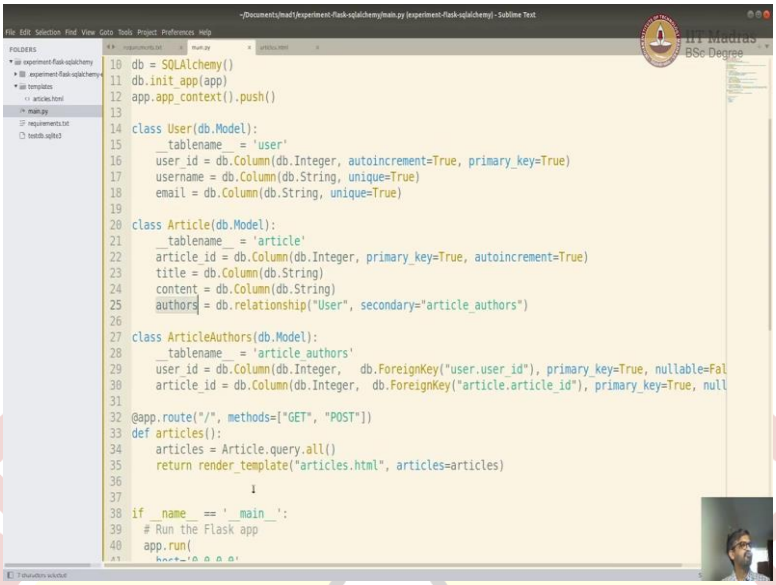
Now we can see it is being iterating and there is title of each of the article and then the content of the article. Now, even now it does not have like a great design for content. I would rather want the content to be in another div so at some point I can give it some design or styling. So I would like to keep that inside one div.

(Refer Slide Time: 02:48)



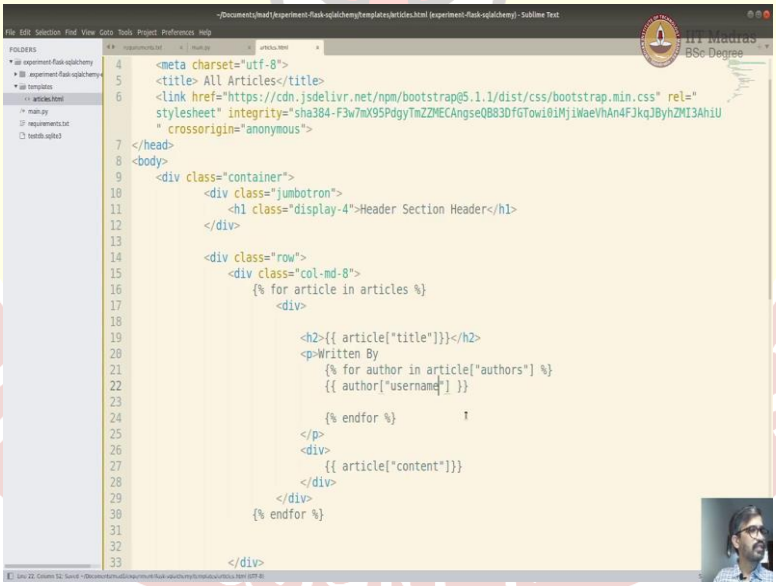
Now we can, at some point we can style this div. We can change the background or we can add another, other style contents. Just going to format it a bit for easy, easy reading. Now after the title is displayed, I want to display the author of this article, who has written this.

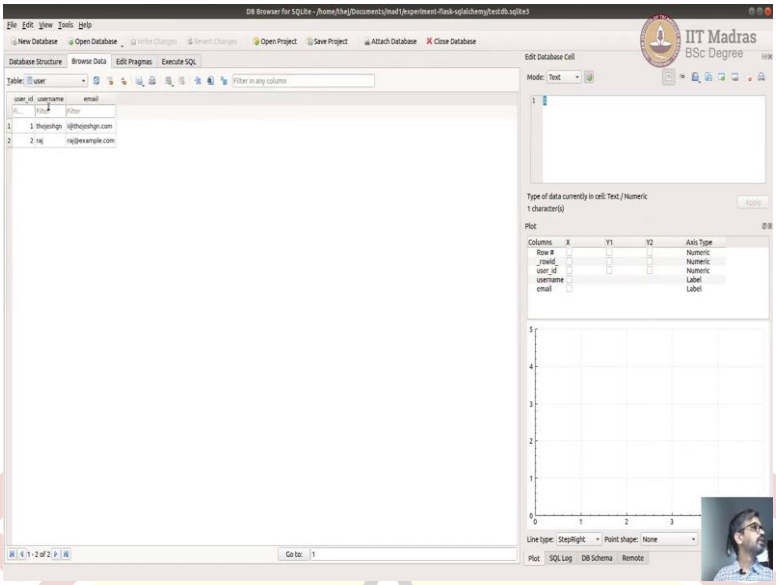
(Refer Slide Time: 03:50)

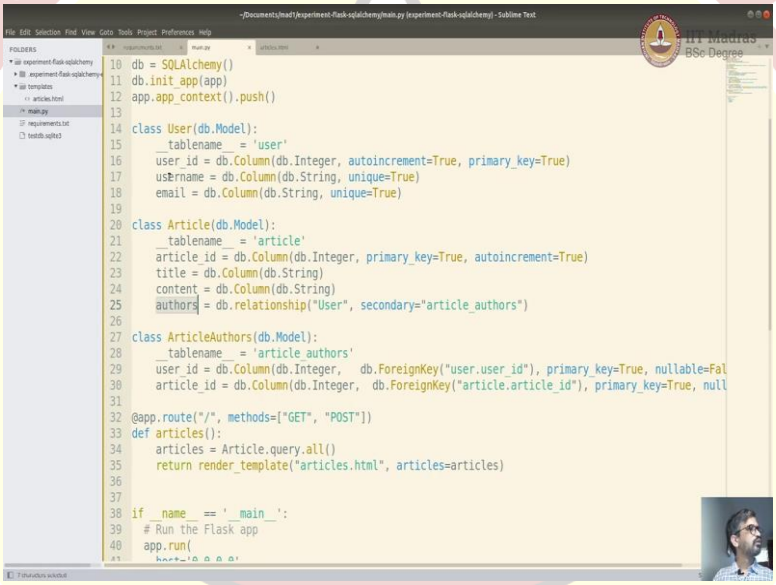


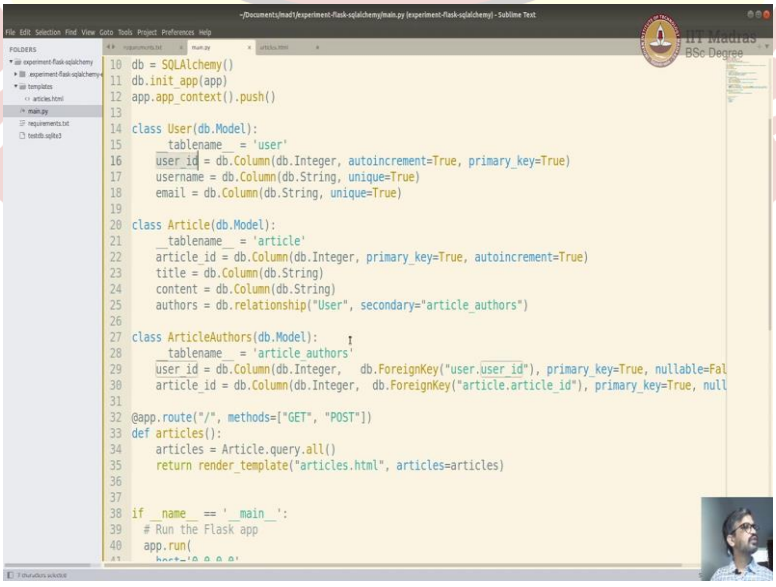
Now in our case, author, if we go to our model, is an array of users. So it can be one or more. So we will have to print at least one or it could be more than one. So we have to iterate through the authors and then display it. So let us do that. It is easy to iterate through it.

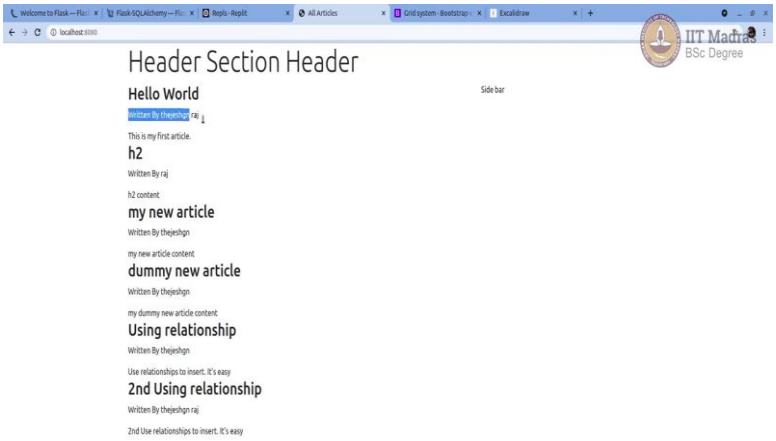
(Refer Slide Time: 04:10)




DB Browser for SQLite - /home/thej/Documents/mad1/experiment-flask-sqlalchemy/test.db.sqlite3
File Edit View Tools Help
New Database
Open Database
Write Changes
Revert Changes
Open Project
Save Project
Attach Database
Close Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table: user
Filter in any column
user_id username email
1 1 thejeshgn i@thejeshgn.com
2 2 raj raj@example.com
Edit Database Cell
Mode: Text
Type of data currently in cell: Text / Numeric
1 character(s)
Apply
Plot
Columns X Y1 Y2 Axis Type
Row # Numeric
rowid Numeric
user_id Numeric
username Label
email Label
Line type: StepRight
Point shape: None
Plot
SQL Log
DB Schema
Remote
Go to: 1
IIT Madras
BSc Degree


~/Documents/mad1/experiment-flask-sqlalchemy/main.py (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
experiment-flask-sqlalchemy
templates
articles.html
main.py
requirements.txt
testdb.sqlite3
db = SQLAlchemy()
db.init_app(app)
app.app_context().push()
class User(db.Model):
__tablename__ = 'user'
user_id = db.Column(db.Integer, autoincrement=True, primary_key=True)
username = db.Column(db.String, unique=True)
email = db.Column(db.String, unique=True)
class Article(db.Model):
__tablename__ = 'article'
article_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
title = db.Column(db.String)
content = db.Column(db.String)
authors = db.relationship("User", secondary="article_authors")
class ArticleAuthors(db.Model):
__tablename__ = 'article_authors'
user_id = db.Column(db.Integer, db.ForeignKey("user.user_id"), primary_key=True, nullable=Fal
article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, null
@app.route("/", methods=["GET", "POST"])
def articles():
articles = Article.query.all()
return render_template("articles.html", articles=articles)
if __name__ == '__main__':
# Run the Flask app
app.run(
IIT Madras
BSc Degree

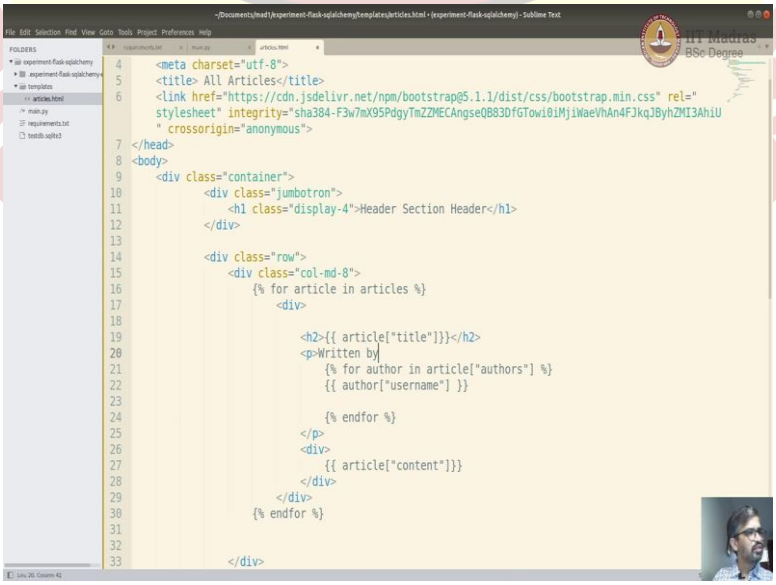

~/Documents/mad1/experiment-flask-sqlalchemy/main.py (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
experiment-flask-sqlalchemy
templates
articles.html
main.py
requirements.txt
testdb.sqlite3
db = SQLAlchemy()
db.init_app(app)
app.app_context().push()
class User(db.Model):
__tablename__ = 'user'
user_id = db.Column(db.Integer, autoincrement=True, primary_key=True)
username = db.Column(db.String, unique=True)
email = db.Column(db.String, unique=True)
class Article(db.Model):
__tablename__ = 'article'
article_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
title = db.Column(db.String)
content = db.Column(db.String)
authors = db.relationship("User", secondary="article_authors")
class ArticleAuthors(db.Model):
__tablename__ = 'article_authors'
user_id = db.Column(db.Integer, db.ForeignKey("user.user_id"), primary_key=True, nullable=Fal
article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, null
@app.route("/", methods=["GET", "POST"])
def articles():
articles = Article.query.all()
return render_template("articles.html", articles=articles)
if __name__ == '__main__':
# Run the Flask app
app.run(
IIT Madras
BSc Degree

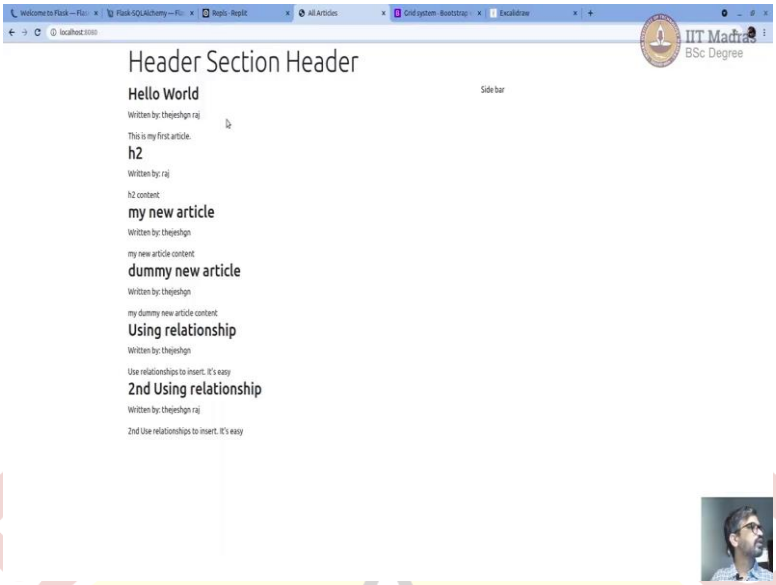


Let us do one. Let us put the author names in inside the P tag, below it. Let us say written by. So there can be more than one so even that will be a For loop. So For loop will be article authors, around article authors. So written by looping through authors from article and I am getting author for each of those. And this will be end for. Now you can print that.

Author, so what is available for us? It will be username or email. Here if I check main.py, username is username, and user_id is user_id, and email is email. So let us take username and display it. So that is how this is displayed. Let us see. Now it shows written by thejeshgn and raj, raj and raj, etcetera, etcetera, but this is not good. We want it to be like little more readable, make it like comma separated with space and stuff like that.

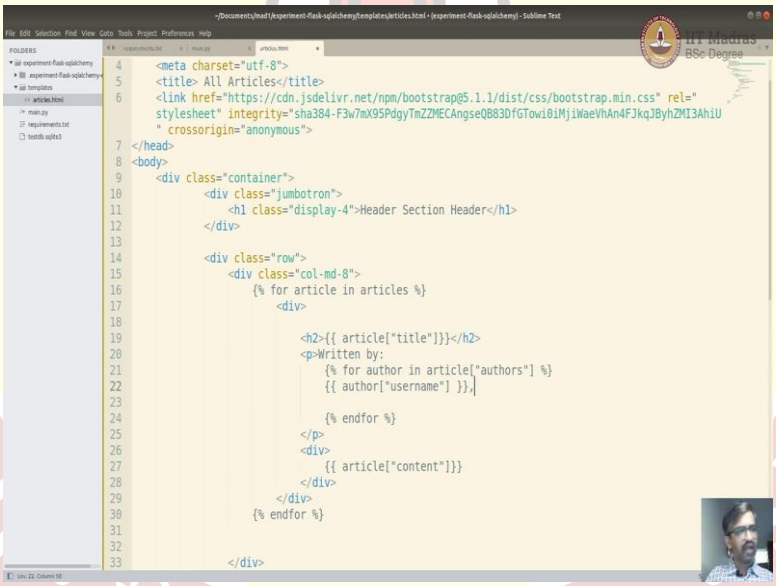
(Refer Slide Time: 05:45)

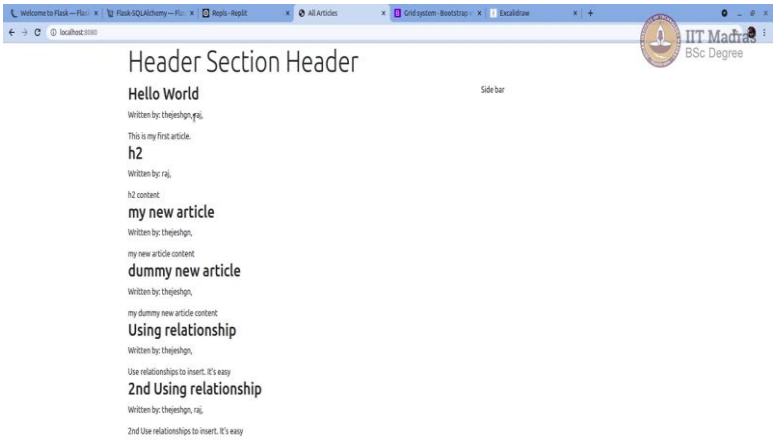




So do little more formatting, and I will also make it small characters, make it like this. That looks little better.

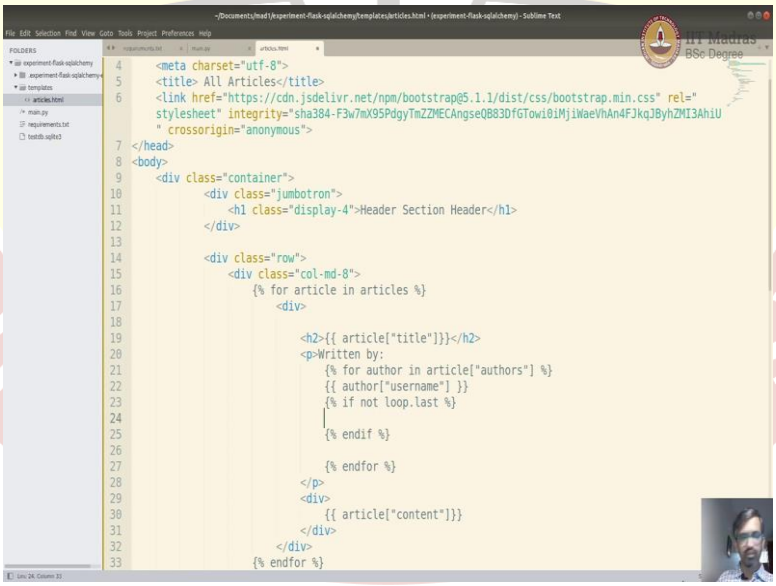
(Refer Slide Time: 05:56)

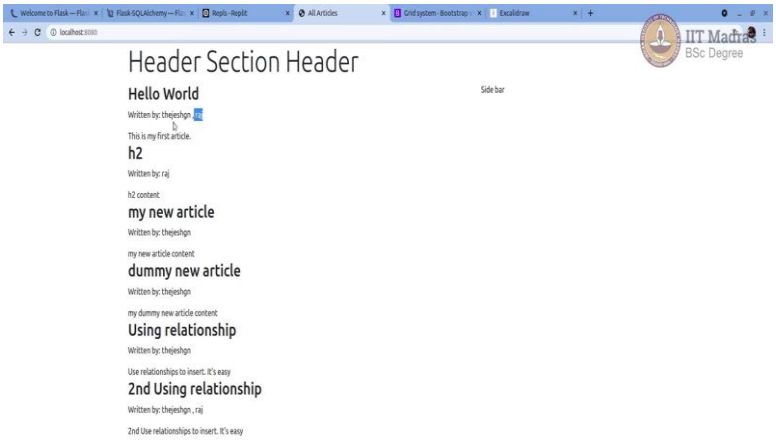




Now let us make it a comma separated. Now, if I add a comma here, that should work but there is a problem. Comma comes even at the end even if there is one user or even if there are two users, even at the last user, there is a comma. I wanted rather it to be not to have anything and I wanted to be thejeshgn, Raj. That is it. Nothing else. So there is condition that you can check in For loop that if it is not the last and then you can use that to put the comma.

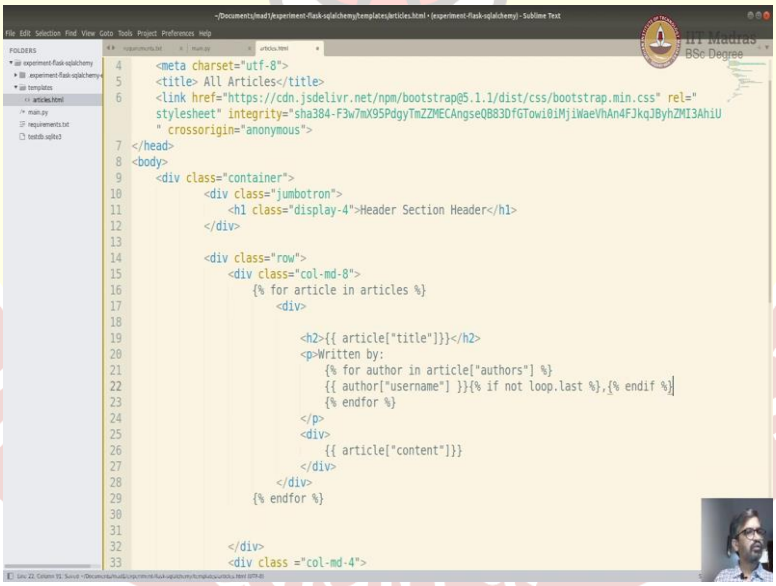
(Refer Slide Time: 06:40)

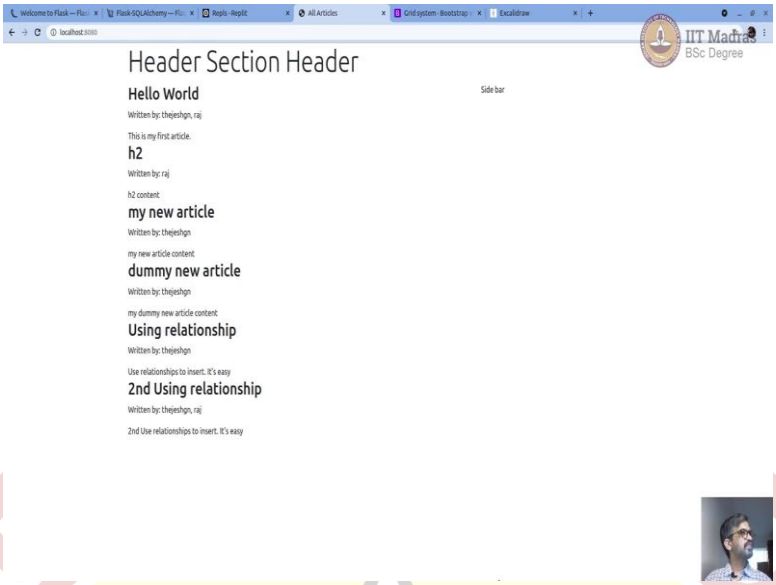




We can, let us do that. We can do if not last, if not the last item, then we can put the comma. So if this is not the last item in the loop then put a comma. Now you can see that because this is the last item in the loop, it is not putting comma otherwise it is putting comma. But there is like a space here that is because there is a space here.

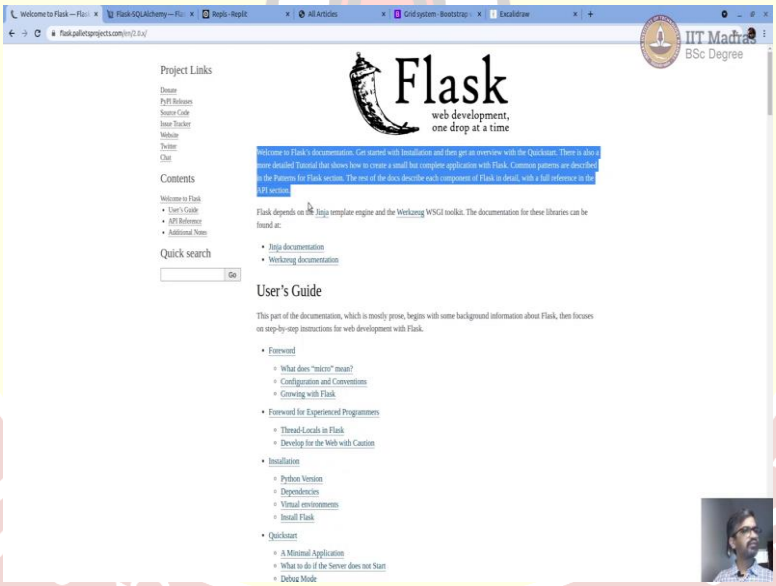
(Refer Slide Time: 07:16)

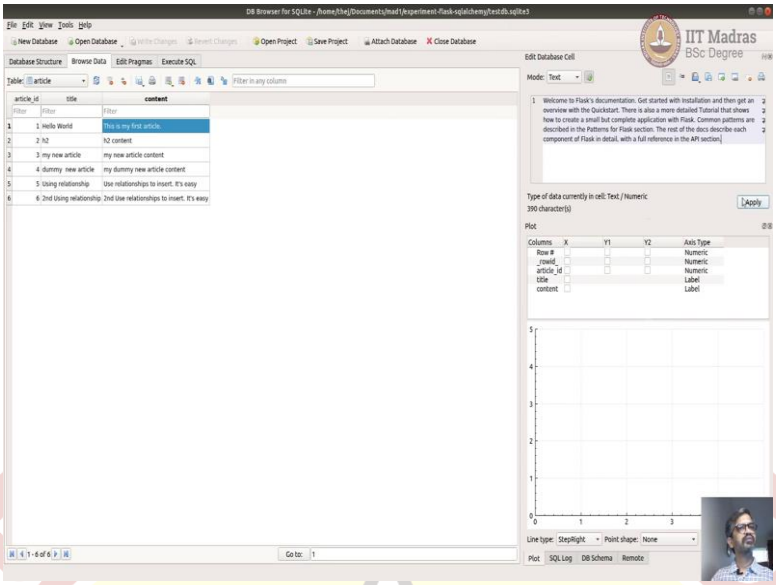


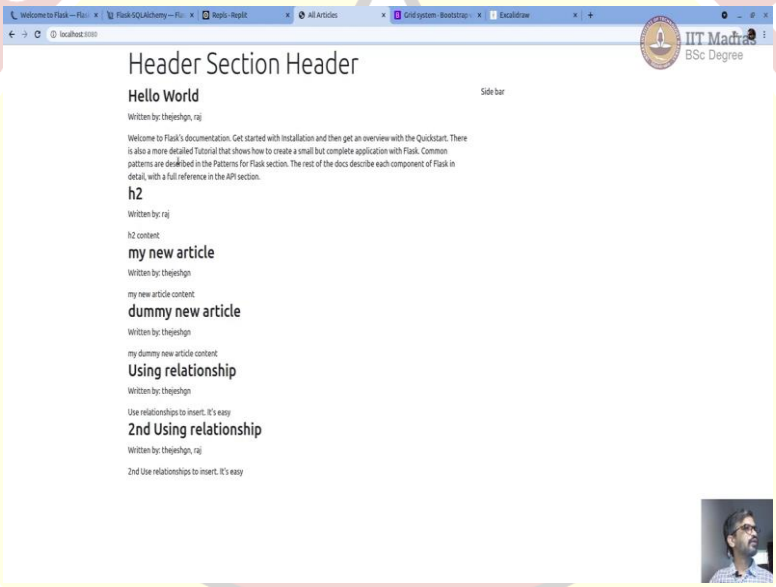


So let us bring it to the same line because it is putting a slash. Now should be fine. There you go.

(Refer Slide Time: 07:35)

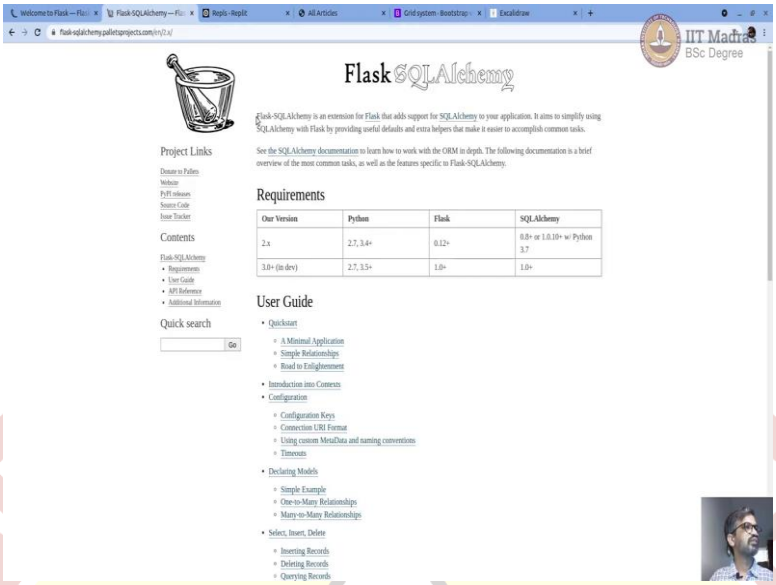


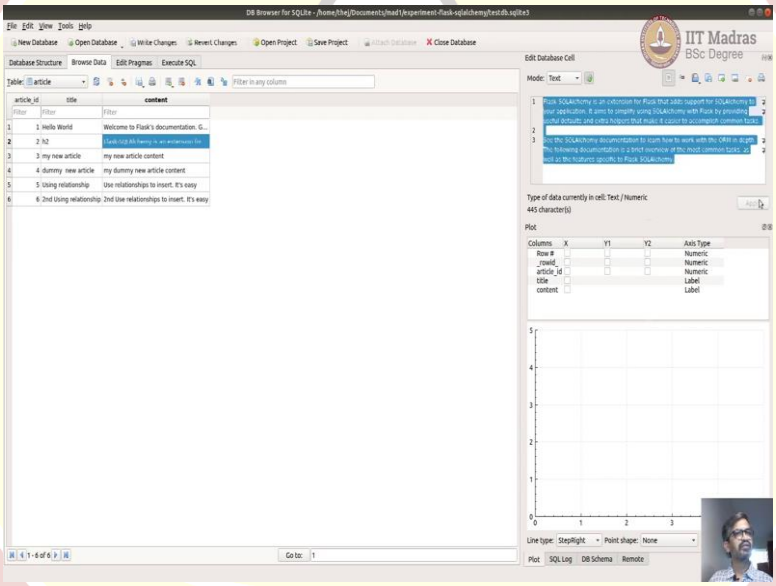


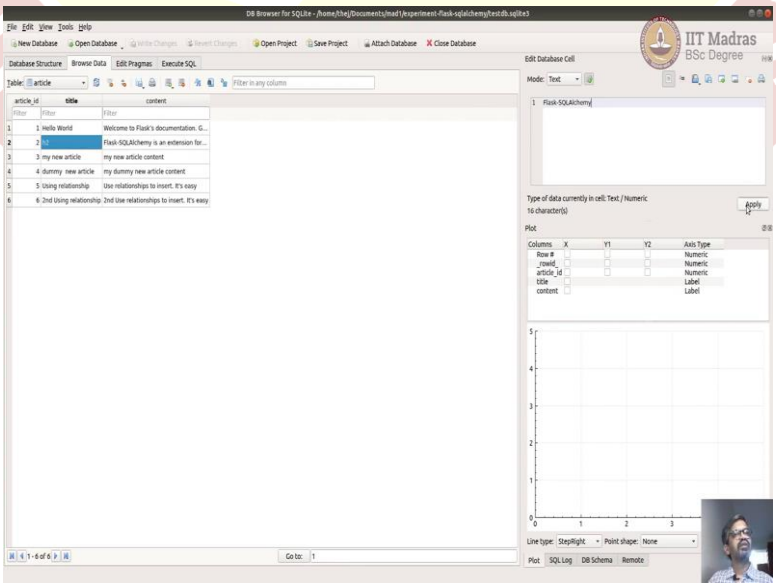


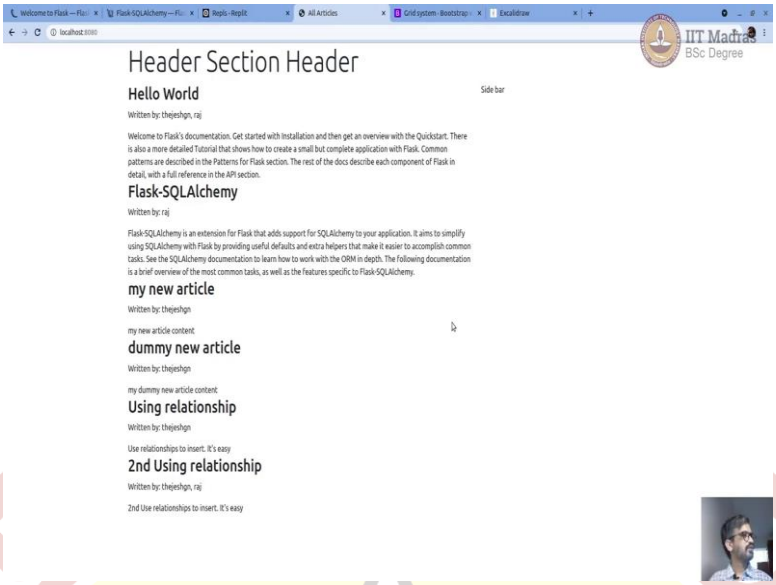
Let us put some more content to this article. I am just going to copy paste some text so it looks better. I am just going to go back to my article and then just put some content. Let us see how it looks. There you go. You can see that it stops at the main section of our blog listing and it does not go to the sidebar part.

(Refer Slide Time: 08:03)



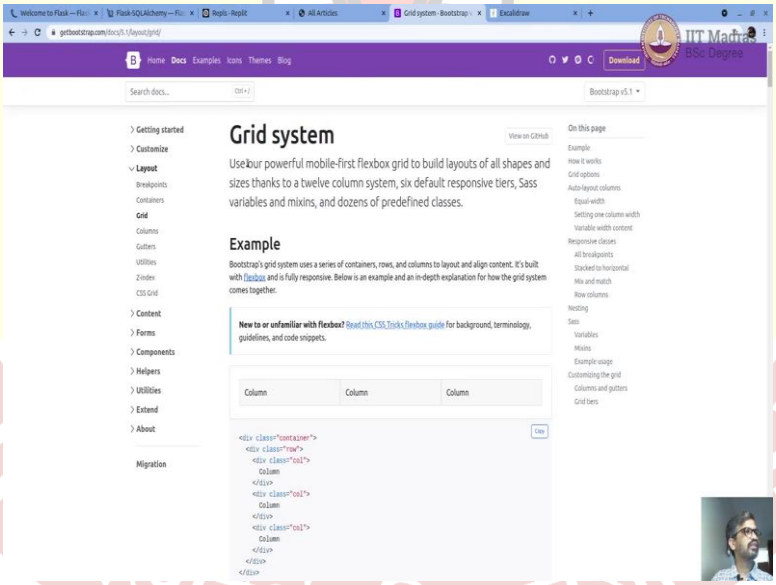






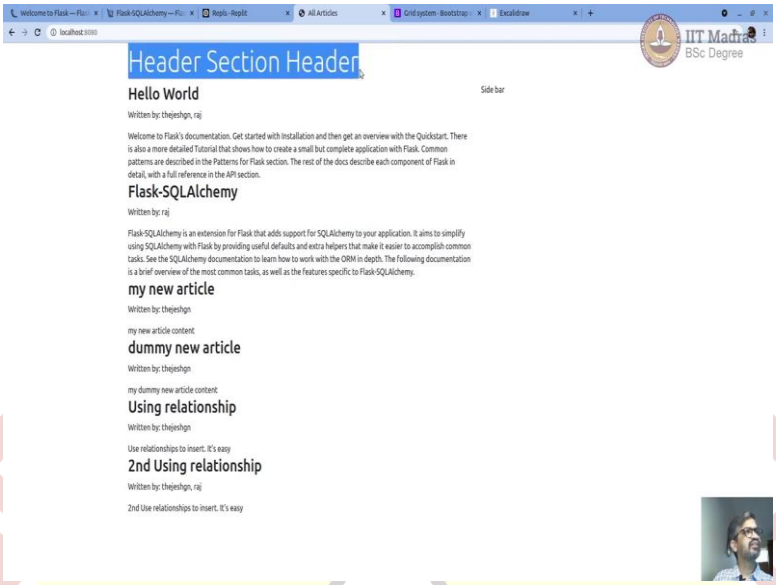
Let us add some more content to some other things just to make it little more useful content instead of dummy content. And give a title also, meaningful title. I have saved it, let us come back here and see. So now you can see that there is, there is some decent looking blog listing.

(Refer Slide Time: 08:49)



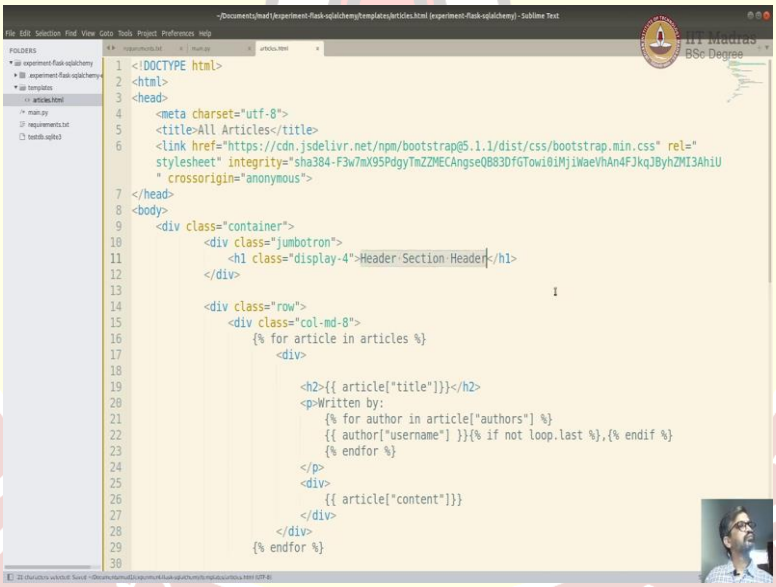
Now we can also add space between these two, our by adding it like breaking or we can give like a buffer space around the layout and stuff like that. You can do many more things. I am not just going to go there I am just going to leave at this because that is virtually, like impossible to cover. You can play with this.

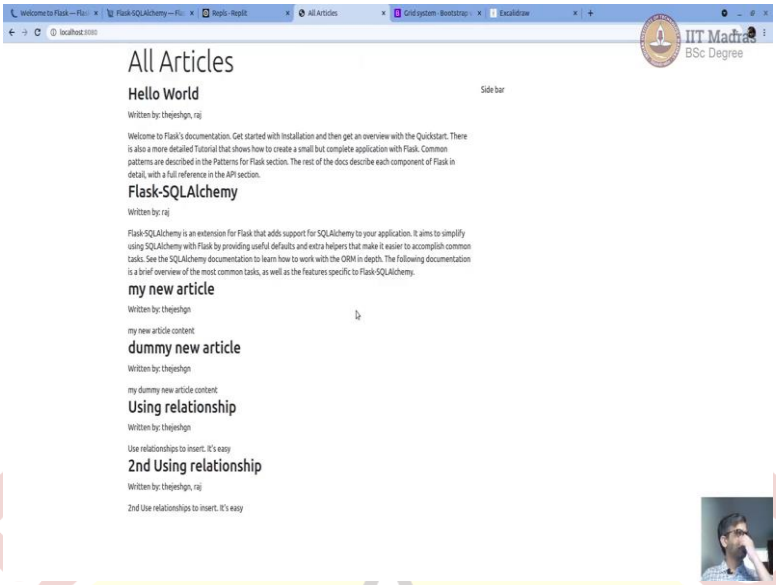
(Refer Slide Time: 09:00)



I am going to replace this header by actually writing our own header.

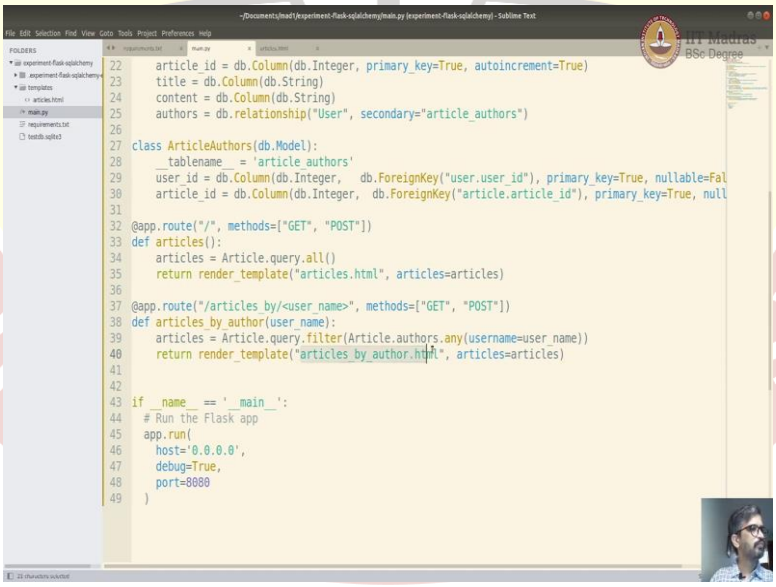
(Refer Slide Time: 09:09)





Let us, because it is all articles I am just going to put the same header there too, in the jumbo tron. That is correct. Now I want to add a feature where if I click this user name, I want to show the list of articles only by that user. It is like a interesting way to look at it. If you click, like, on raj, it shows a list of articles by raj, if you click on thejeshgn. it should have the list of articles by thejeshgn.

(Refer Slide Time: 09:55)



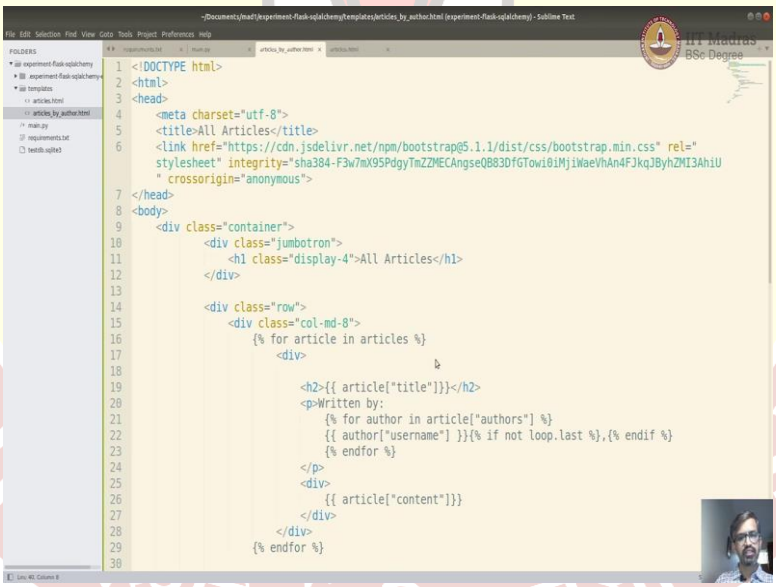
So for that, we have to have some format where we can pass the user name to the articles listing and then filter it by that username, for which we will write another model, we will not use the same model. Just to keep it simple we will add another model, and we will call it get Articles by username or author. And it takes user name. And let us set up the path or routing, we can give whatever want to but I will just give it Articles by to make it meaningful.

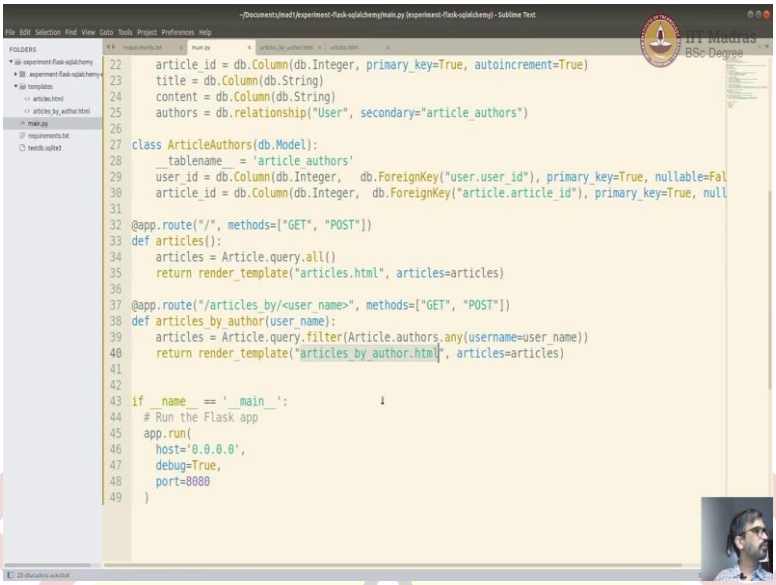
I mean even though we can give anything it is always useful to give some useful name or meaningful name. And it takes one parameter as part of its get called username. It takes username and the same thing is passed here. Now we have to modify our articles query to filter by username.

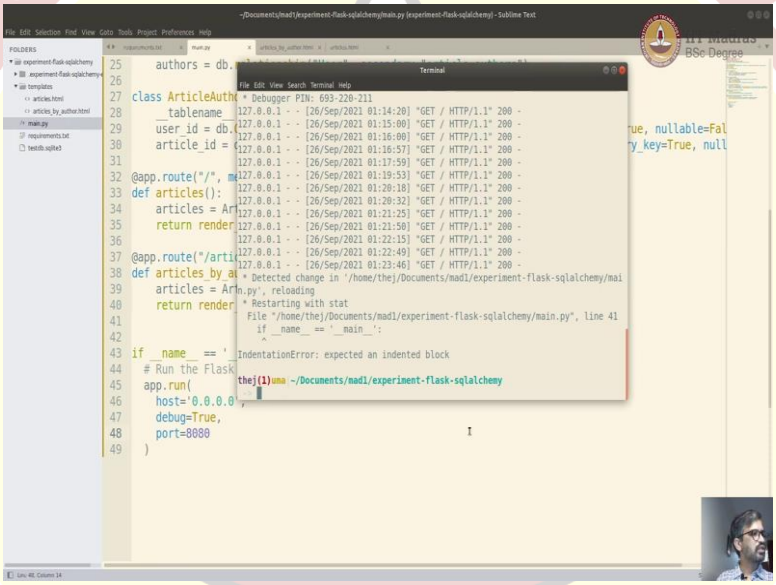
So it is actually quite straightforward, again. So it is article query and then filter, filter by author name with, has a username as username, if any of them. Give me all of them which has username as username, username being here is this. This is the column name and this is the variable. Actually you can just, to make it clear for you, you can even make it like this. This.

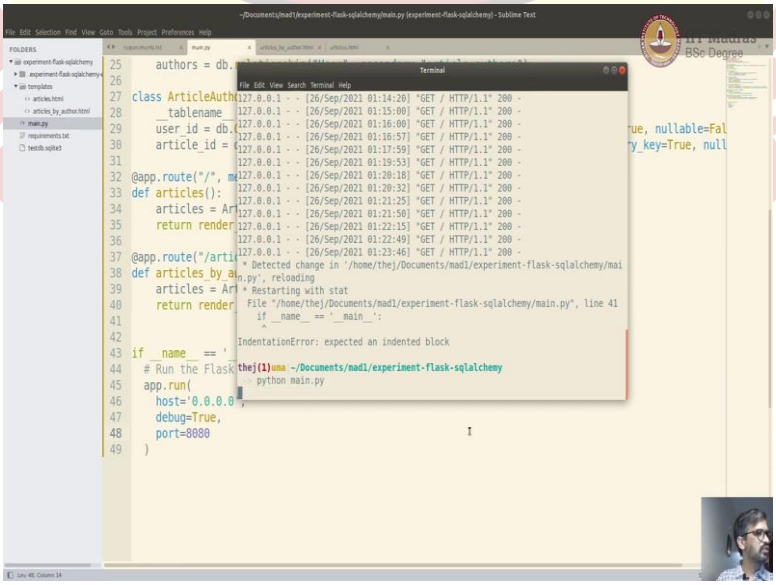
Now, we have got all the filtered articles. You want to return to, like, a template. I am going to create a separate template. Though we could have used the same template I am going to create a separate temple called Author Articles or Articles by an Author. So it is a separate template. So let us create this template.

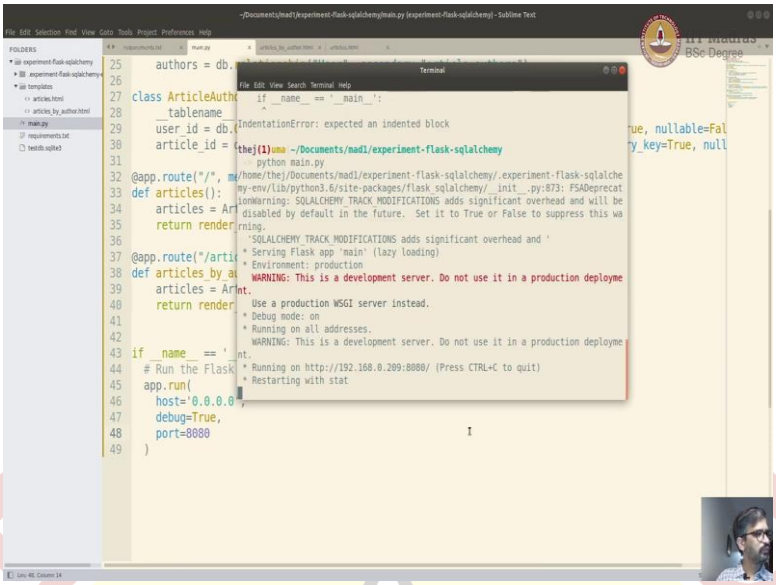
(Refer Slide Time: 12:13)




~/Documents/mad1/experiment-flask-sqlalchemy/main.py (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
experiment-flask-sqlalchemy
templates
articles.html
articles_by_author.html
main.py
requirements.txt
testdb.sqlite3
22 article_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
23 title = db.Column(db.String)
24 content = db.Column(db.String)
25 authors = db.relationship("User", secondary="article_authors")
26
27 class ArticleAuthors(db.Model):
28 __tablename__ = 'article_authors'
29 user_id = db.Column(db.Integer, db.ForeignKey("user.user_id"), primary_key=True, nullable=Fal
30 article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, null
31
32 @app.route("/", methods=["GET", "POST"])
33 def articles():
34 articles = Article.query.all()
35 return render_template("articles.html", articles=articles)
36
37 @app.route("/articles_by/<user_name>", methods=["GET", "POST"])
38 def articles_by_author(user_name):
39 articles = Article.query.filter(Article.authors.any(username=user_name))
40 return render_template("articles_by_author.html", articles=articles)
41
42
43 if __name__ == '__main__':
44 # Run the Flask app
45 app.run(
46 host='0.0.0.0',
47 debug=True,
48 port=8080
49)
IIT Madras
BSc Degree

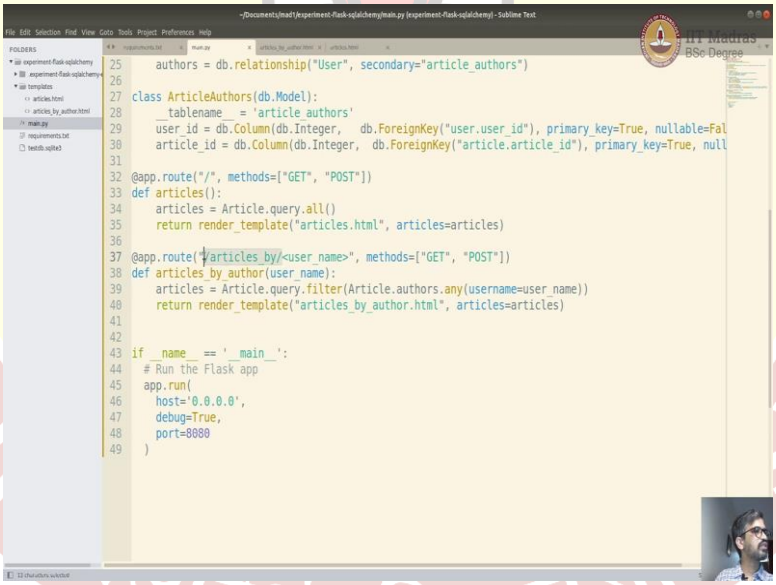

~/Documents/mad1/experiment-flask-sqlalchemy/main.py (experiment-flask-sqlalchemy) - Sublime Text
Terminal
File Edit View Search Terminal Help
* Debugger PIN: 693-220-211
127.0.0.1 - - [26/Sep/2021 01:14:20] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:15:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:16:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:16:57] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:17:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:19:53] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:20:18] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:20:32] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:21:25] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:21:50] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:22:15] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:22:49] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:23:46] "GET / HTTP/1.1" 200 -
* Detected change in '/home/thej/Documents/mad1/experiment-flask-sqlalchemy/main.py', reloading
* Restarting with stat
File "/home/thej/Documents/mad1/experiment-flask-sqlalchemy/main.py", line 41
if __name__ == '__main__':
^
IndentationError: expected an indented block
thej(1)uma ~/Documents/mad1/experiment-flask-sqlalchemy
IIT Madras
BSc Degree

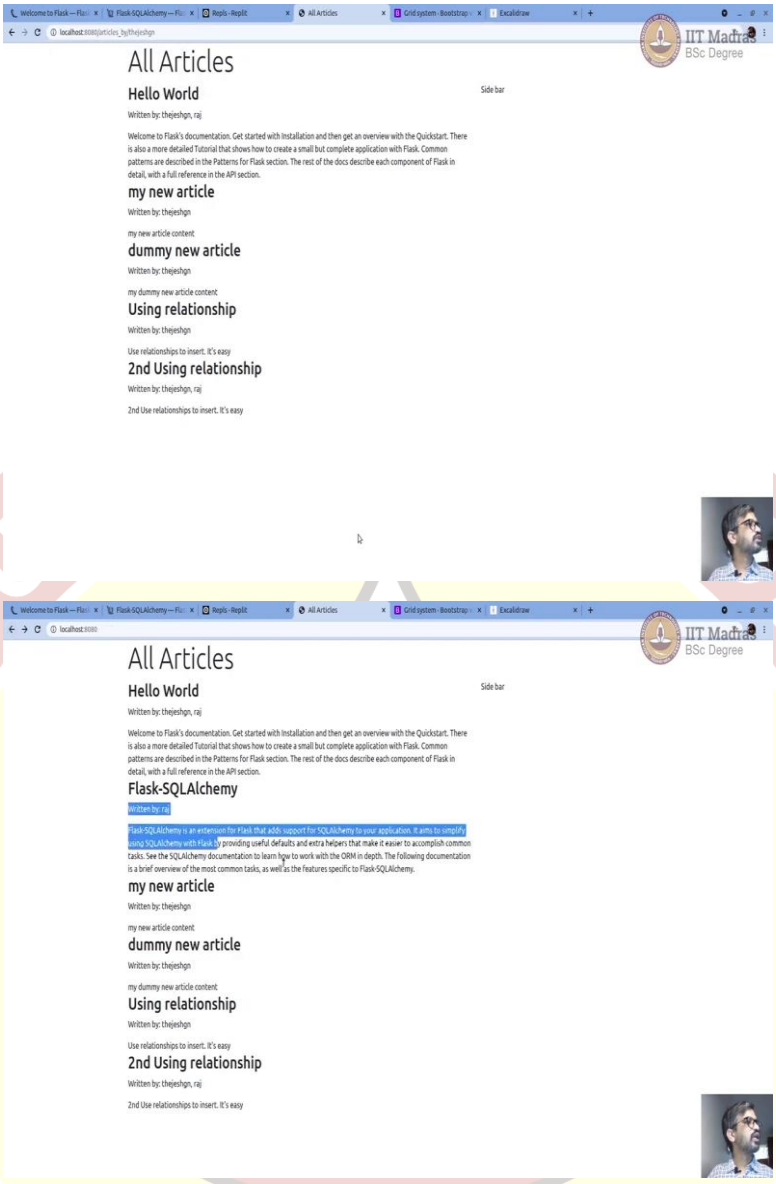

~/Documents/mad1/experiment-flask-sqlalchemy/main.py (experiment-flask-sqlalchemy) - Sublime Text
Terminal
File Edit View Search Terminal Help
127.0.0.1 - - [26/Sep/2021 01:14:20] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:15:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:16:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:16:57] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:17:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:19:53] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:20:18] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:20:32] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:21:25] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:21:50] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:22:15] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:22:49] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Sep/2021 01:23:46] "GET / HTTP/1.1" 200 -
* Detected change in '/home/thej/Documents/mad1/experiment-flask-sqlalchemy/main.py', reloading
* Restarting with stat
File "/home/thej/Documents/mad1/experiment-flask-sqlalchemy/main.py", line 41
if __name__ == '__main__':
^
IndentationError: expected an indented block
thej(1)uma ~/Documents/mad1/experiment-flask-sqlalchemy
python main.py
IIT Madras
BSc Degree



The content will be somewhat similar so I am going to copy it and paste the template. Let us see how it looks first. Let us save this. Let us go back to main.py. Yes, it has been set up. Let us restart this.

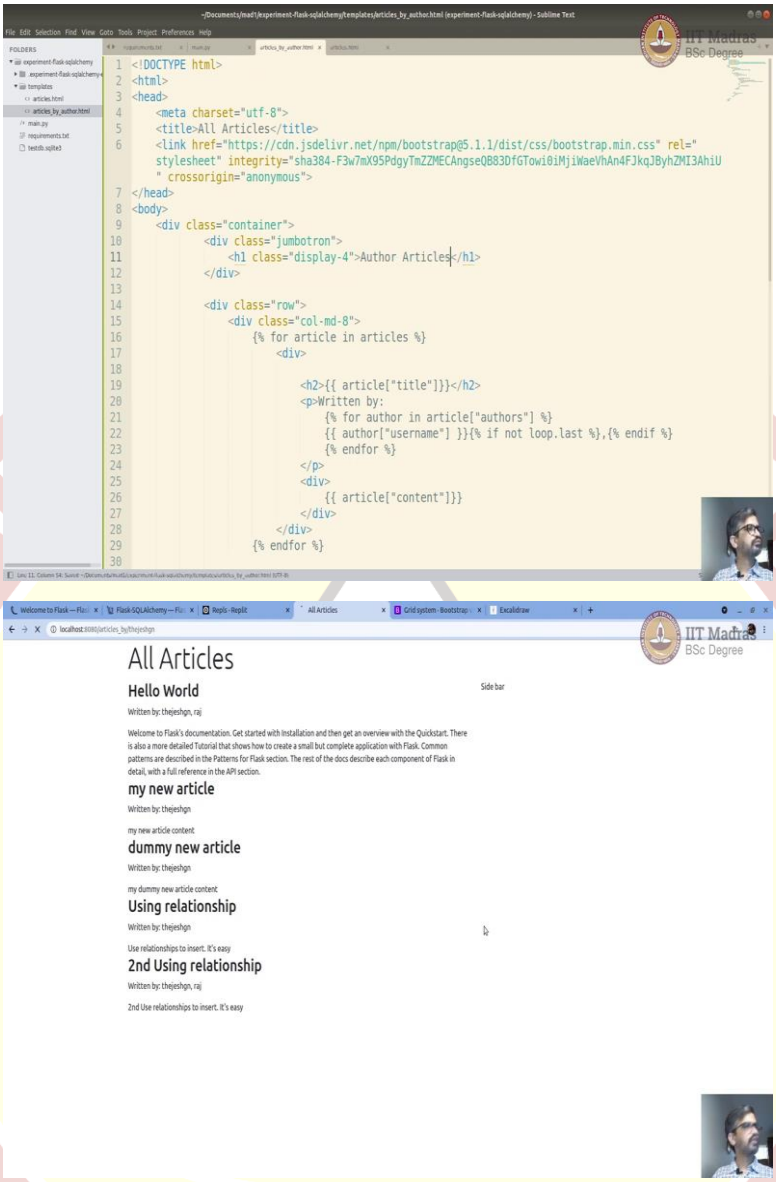
(Refer Slide Time: 12:42)





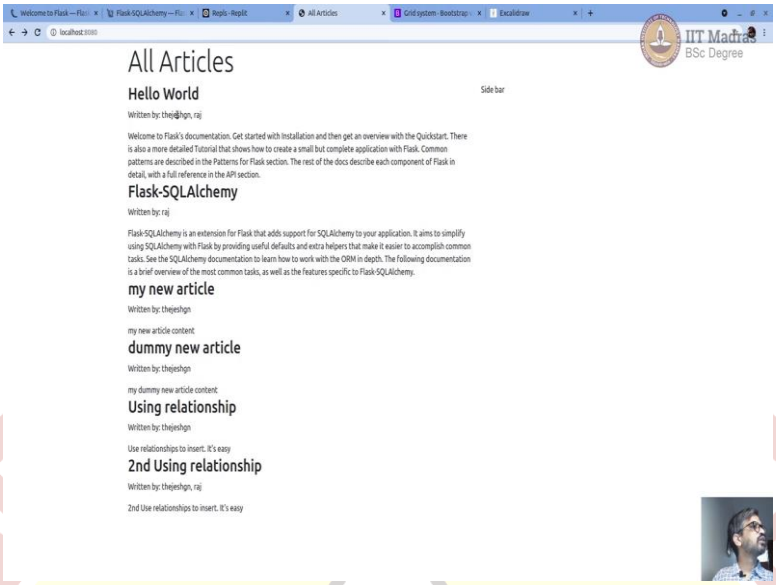
Now if I go through /articles by thejeshgn, it should show list of articles by thejeshgn. So here it will be articles, give a username, thejeshgn. So yeah it is happening. It is, you can see only articles by thejeshgn. In the other one there was also an article by raj. And there we have.

(Refer Slide Time: 13:28)



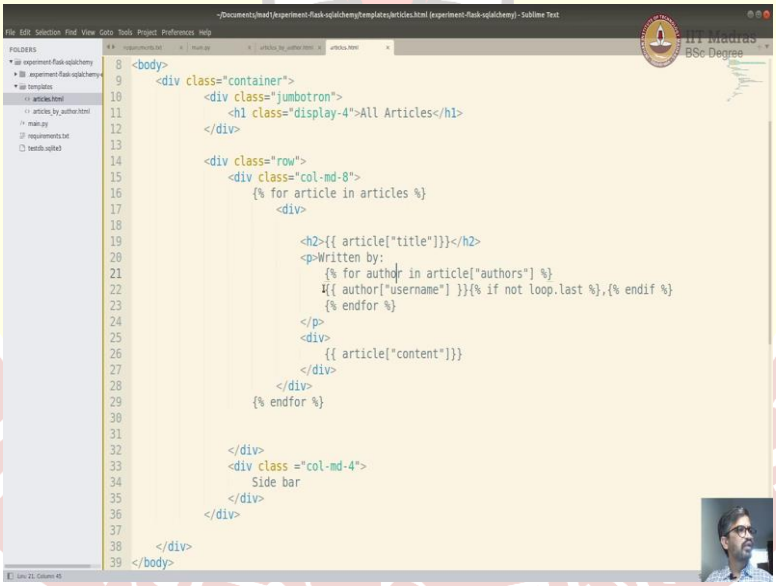
Now we can also edit this little more, and show, call it Author Articles.

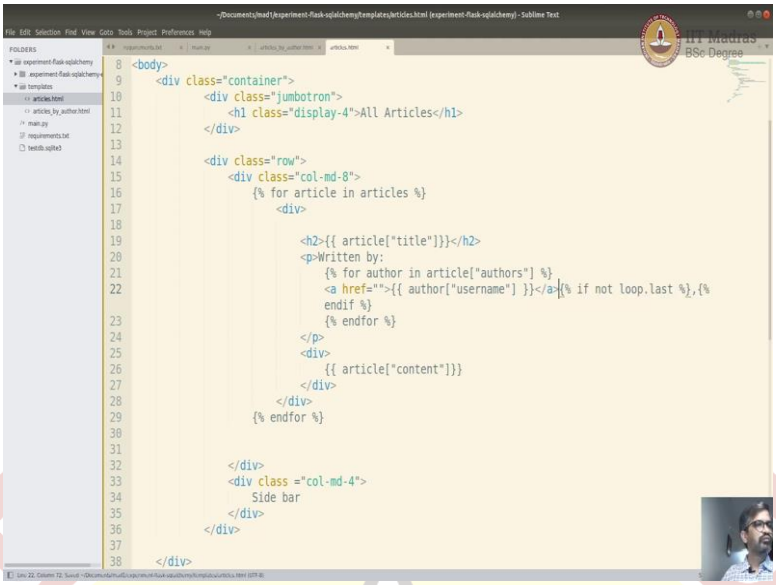
(Refer Slide Time: 13:44)



Now in the main page I want this to be clickable link, and take it to that page so we have to make it like an href.

(Refer Slide Time: 13:54)




~/Documents/mad1/experiment-flask-sqlalchemy/templates/articles.html (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
experiment-flask-sqlalchemy
templates
articles.html
articles_by_author.html
main.py
requirements.txt
testdb.sqlite3
<body>

<h1 class="display-4">All Articles</h1>

{% for article in articles %}

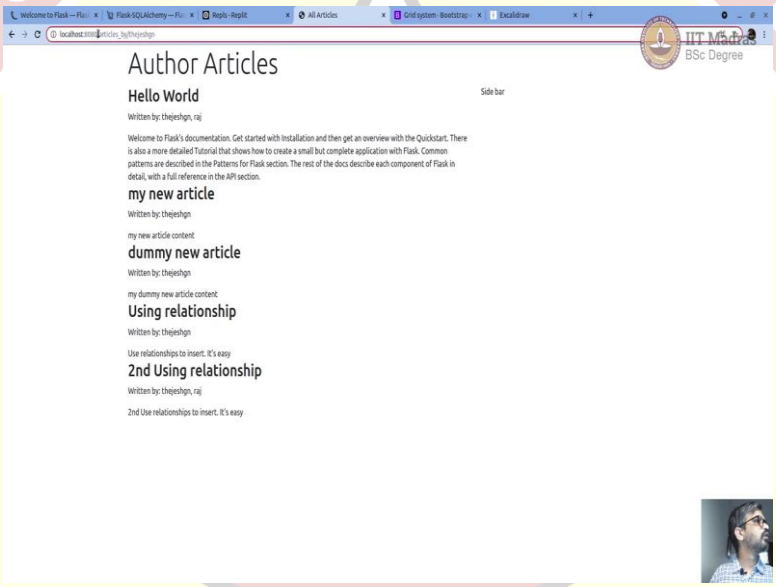
<h2>{{ article["title"]}}</h2>
Written by:
{% for author in article["authors"] %}
{{ author["username"] }}{% if not loop.last %},{%
endif %}
{% endfor %}

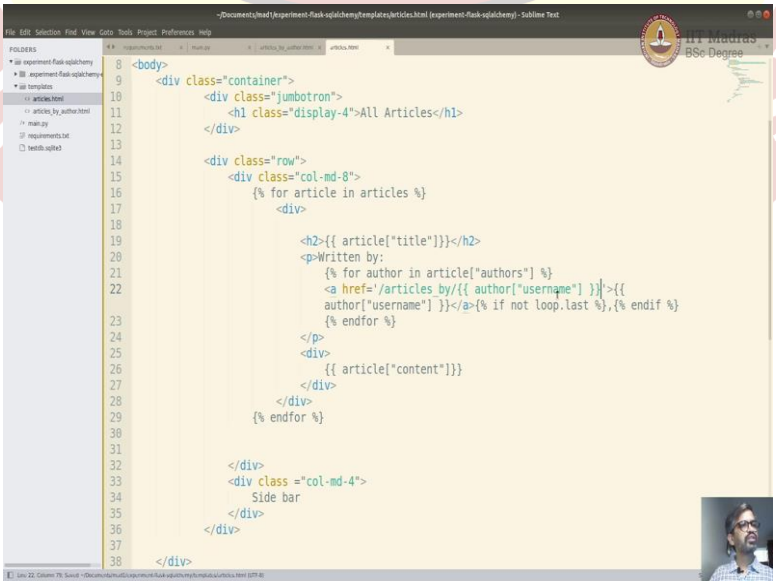
{{ article["content"]}}

{% endfor %}

Side bar

IIT Madras
BSc Degree


Welcome to Flask — Flas
Flask-SQLAlchemy — Flas
Repls - Replit
All Articles
Grid system · Bootstrap
Excalidraw
localhost:8000/articles_bythejeshgn
Author Articles
Hello World
Written by: thejeshgn, raj
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.
my new article
Written by: thejeshgn
my new article content
dummy new article
Written by: thejeshgn
my dummy new article content
Using relationship
Written by: thejeshgn
Use relationships to insert. It's easy
2nd Using relationship
Written by: thejeshgn, raj
2nd Use relationships to insert. It's easy
Side bar
IIT Madras
BSc Degree


~/Documents/mad1/experiment-flask-sqlalchemy/templates/articles.html (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
experiment-flask-sqlalchemy
templates
articles.html
articles_by_author.html
main.py
requirements.txt
testdb.sqlite3
<body>

<h1 class="display-4">All Articles</h1>

{% for article in articles %}

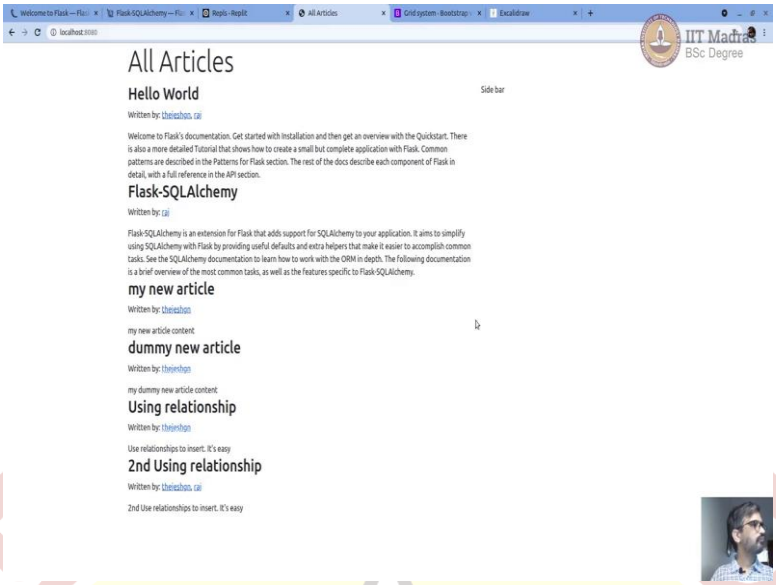
<h2>{{ article["title"]}}</h2>
Written by:
{% for author in article["authors"] %}
{{
author["username"] }}{% if not loop.last %},{% endif %}
{% endfor %}

{{ article["content"]}}

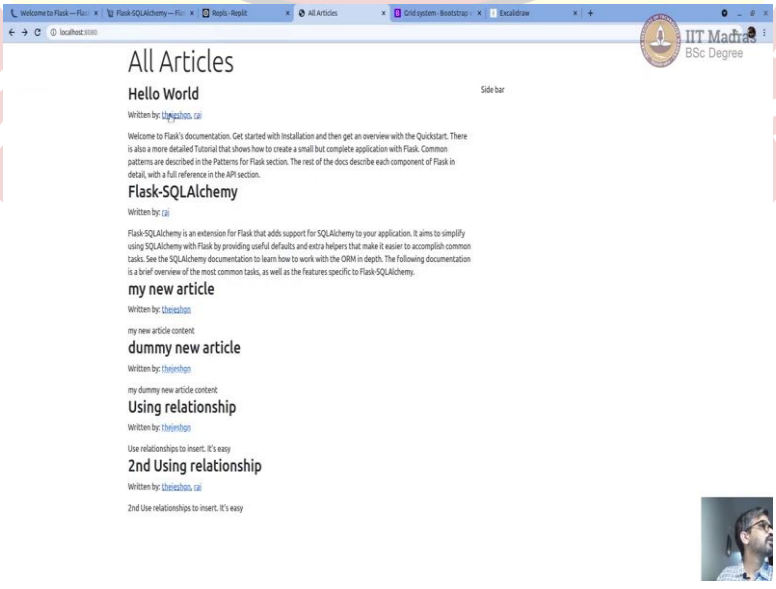
{% endfor %}

Side bar

IIT Madras
BSc Degree


All Articles
Hello World
Written by: thejeshgn, raj
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: raj
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: thejeshgn
my new article content
dummy new article
Written by: thejeshgn
my dummy new article content
Using relationship
Written by: thejeshgn
Use relationships to insert. It's easy
2nd Using relationship
Written by: thejeshgn, raj
2nd Use relationships to insert. It's easy
Side bar
IIT Madras
BSc Degree


Author Articles
Hello World
Written by: thejeshgn, raj
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: raj
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.
2nd Using relationship
Written by: thejeshgn, raj
2nd Use relationships to insert. It's easy
Side bar
IIT Madras
BSc Degree


All Articles
Hello World
Written by: thejeshgn, raj
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: raj
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: thejeshgn
my new article content
dummy new article
Written by: thejeshgn
my dummy new article content
Using relationship
Written by: thejeshgn
Use relationships to insert. It's easy
2nd Using relationship
Written by: thejeshgn, raj
2nd Use relationships to insert. It's easy
Side bar
IIT Madras
BSc Degree

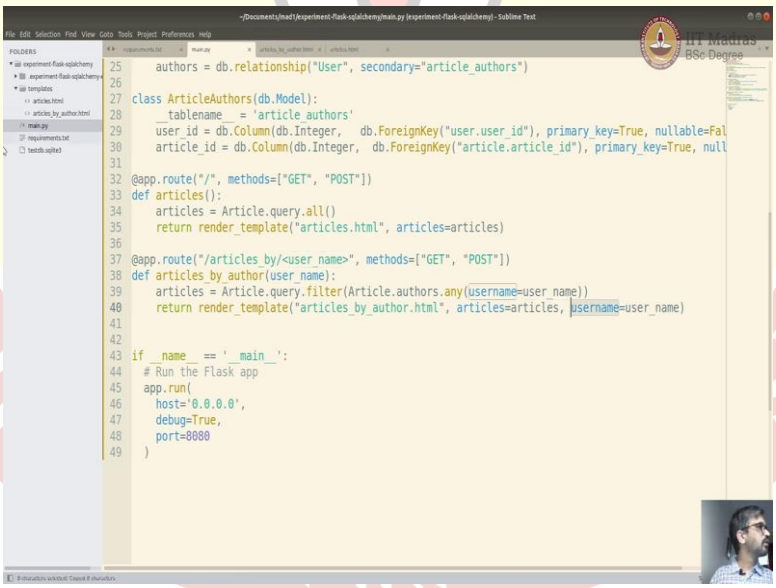
So that is done by going to this, making this art author name as clickable link. I am going to make it like a href. And then link will look like, for example, thejeshgn. So it will look like.

This just same as our routing but instead of this thejeshgn, it will be the name of, name of the specific author. I have to take it from a variable. This is the value actually so I am going to take this and paste it here. So it should actually give me articles by whichever is the username. Now if I refresh, oh, okay, I should go to the main page. There you go.

So now they have all become links as you can see. You can see the footer here what is the url, so thejeshgn. is thejeshgn., raj is raj, etcetera, etcetera. So if I click on raj it will show only raj's article. I mean or either he is independent or is a co-author. Or if I click on thejeshgn, it shows articles of thejeshgn.

Now actually I want to, when we are in this page I want to show author articles and name of the author or in this case, since we do not have a name, username of the article, username of the author. So we will just do by simply passing the username as part of our template variable passing.

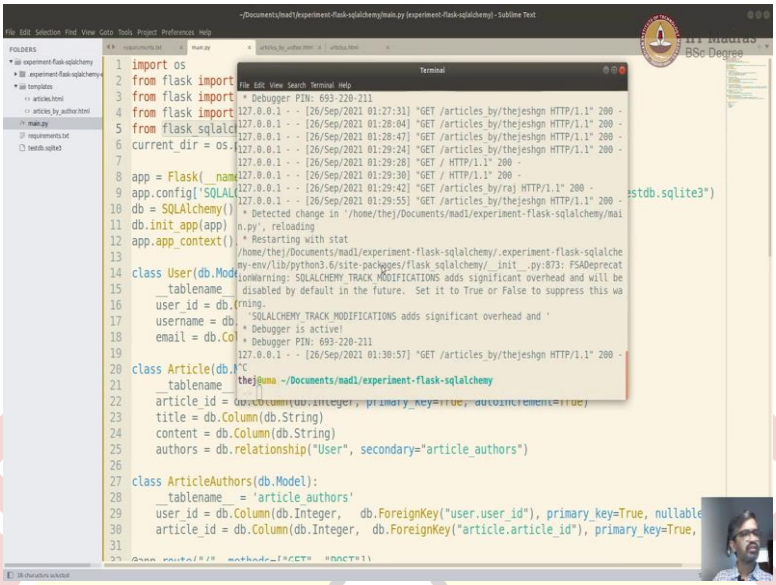
(Refer Slide Time: 16:00)



So username=username. And then we will use this username passed in our articles by author. Articles, Author Articles. That should work. There you go. Now you can also play around with sidebar and do many things. You can also fly here, and, or you can make it like a icon here, if it is clicks back and you go back to the main page etcetera, etcetera. But this should tell you how to use FlaskSQLAlchemy with Flask, and develop a simple application.

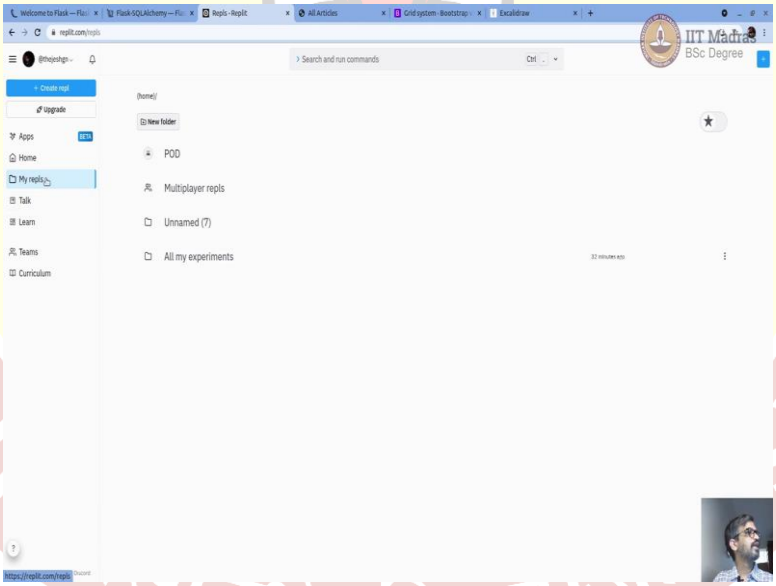
I have not done many other parts. You can write the post part where you can add an article, etcetera, etcetera. You can try that on your own. Now we have done this using local. Let us try and do the same thing on replit.

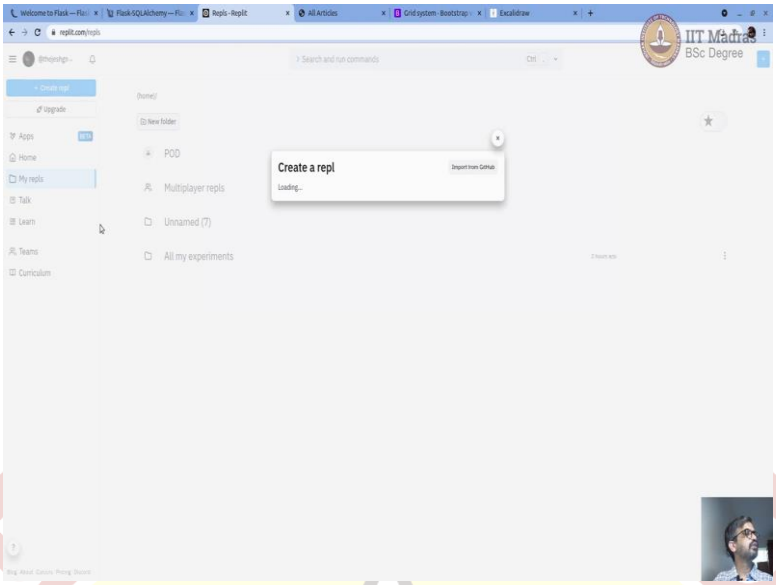
(Refer Slide Time: 17:15)

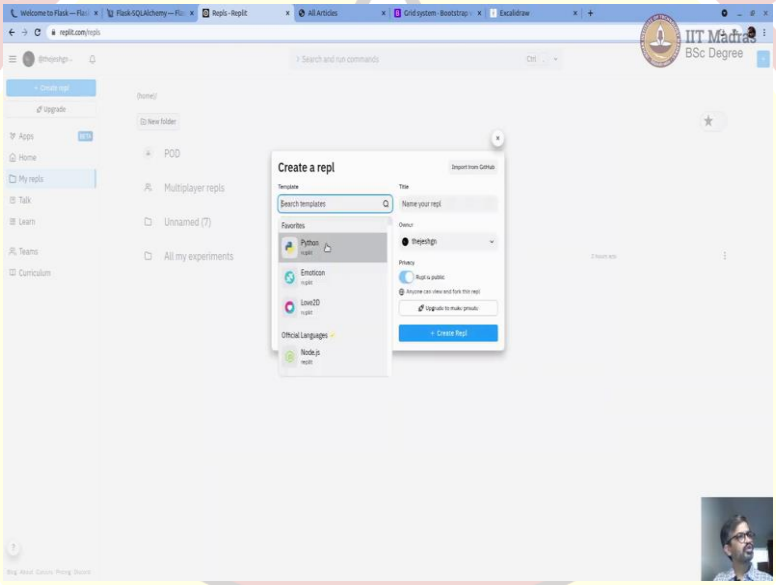


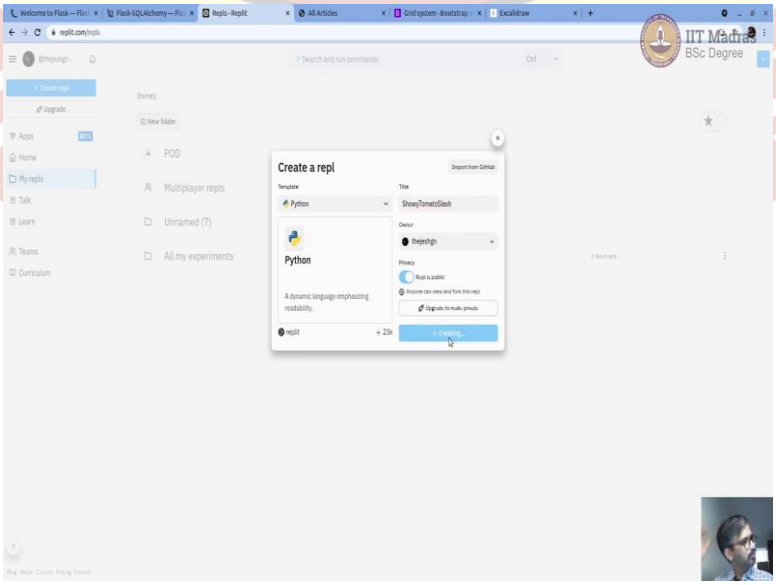
Just quickly, I am going to code the whole thing, I am just going to copy paste. But I am going to show it to you how.

(Refer Slide Time: 17:22)



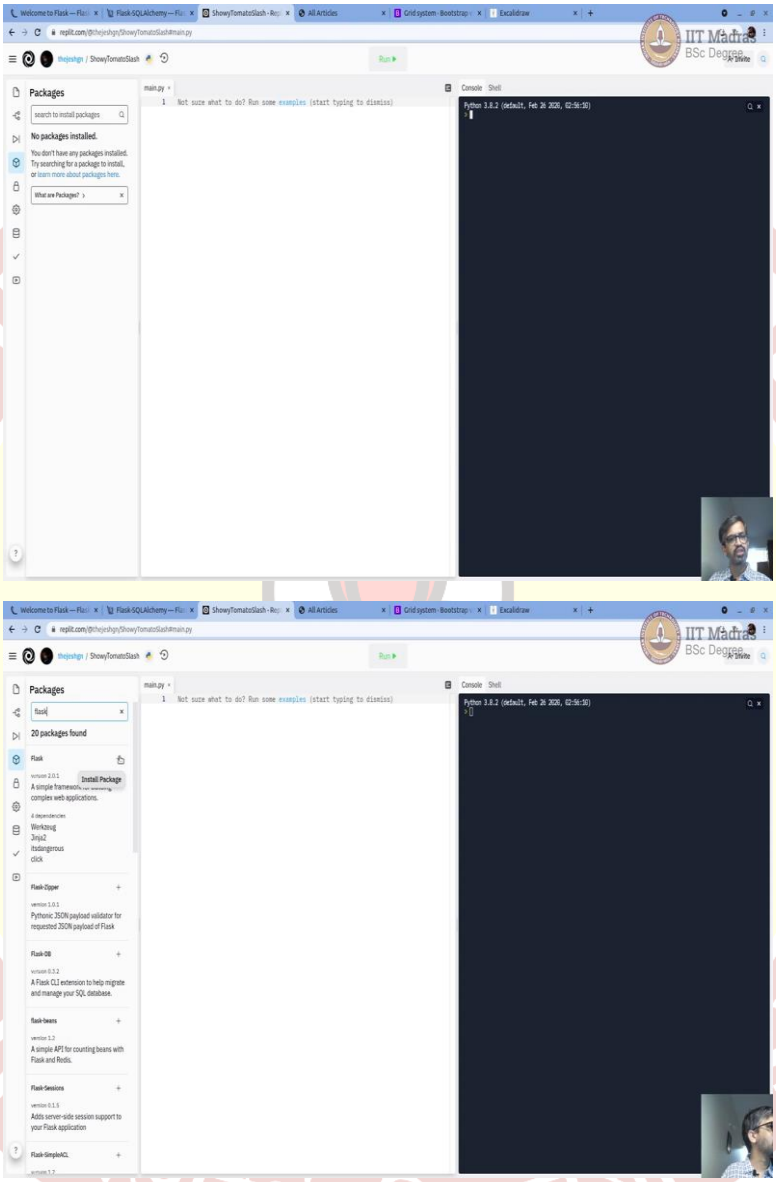

Create a repl
Loading...
Import from GitHub
POD
Multiplayer repls
Unnamed (7)
All my experiments

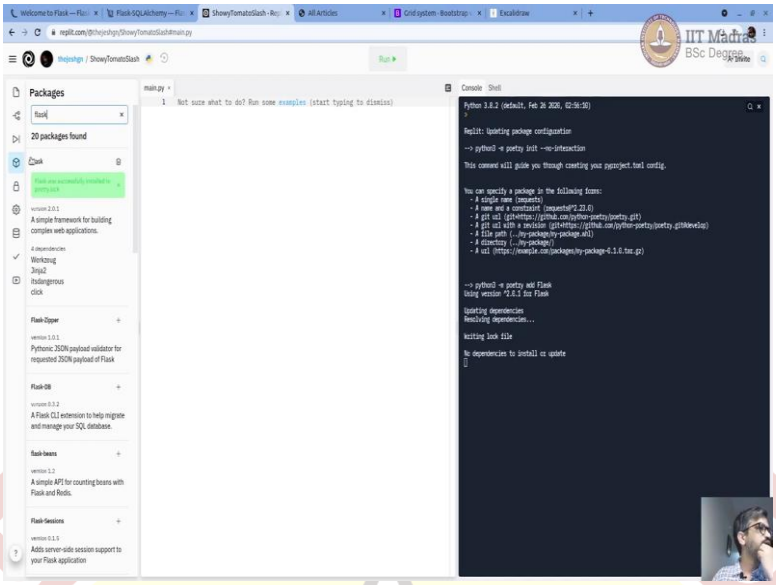

Create a repl
Import from GitHub
Template
Search templates
Favorites
Python
Emoticon
Love2D
Official Languages
Node.js
Title
Name your repl
Owner
thejeshgn
Privacy
Repl is public
Anyone can view and fork this repl
Upgrade to make private
Create Repl
POD
Multiplayer repls
Unnamed (7)
All my experiments

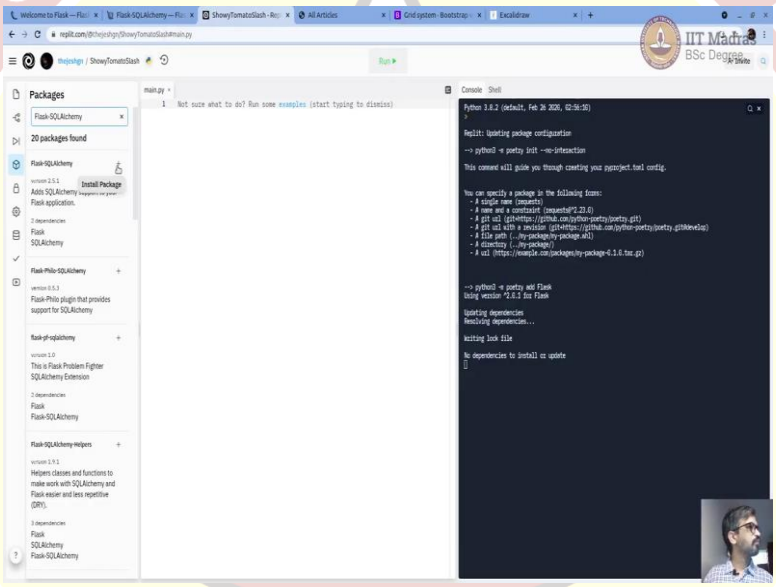

Create a repl
Import from GitHub
Template
Python
Title
ShowyTomatoSlash
Owner
thejeshgn
Privacy
Repl is public
Anyone can view and fork this repl
Upgrade to make private
Python
A dynamic language emphasizing readability.
replit
+ 23k
Creating...
POD
Multiplayer repls
Unnamed (7)
All my experiments

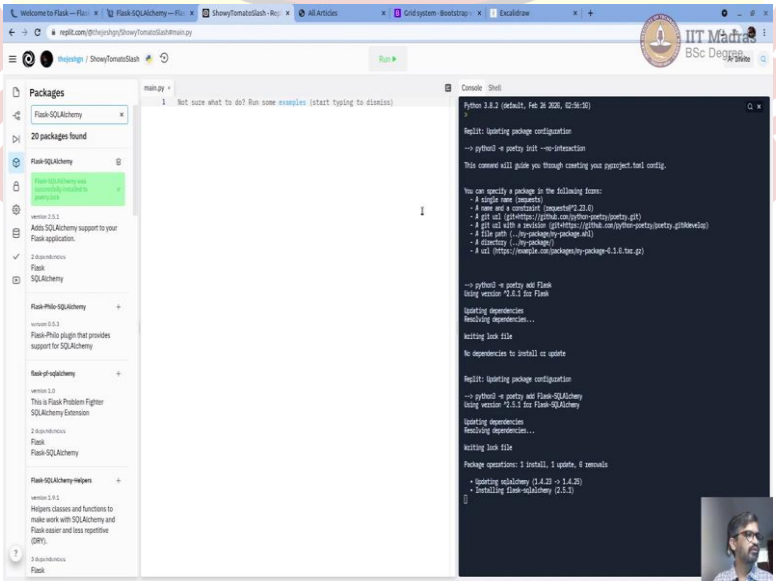
So once you log into Replit, go to My repls, create a new repl here using create repl. And choose python. And give some name. I am just going to going to go with the default name. And create repl.

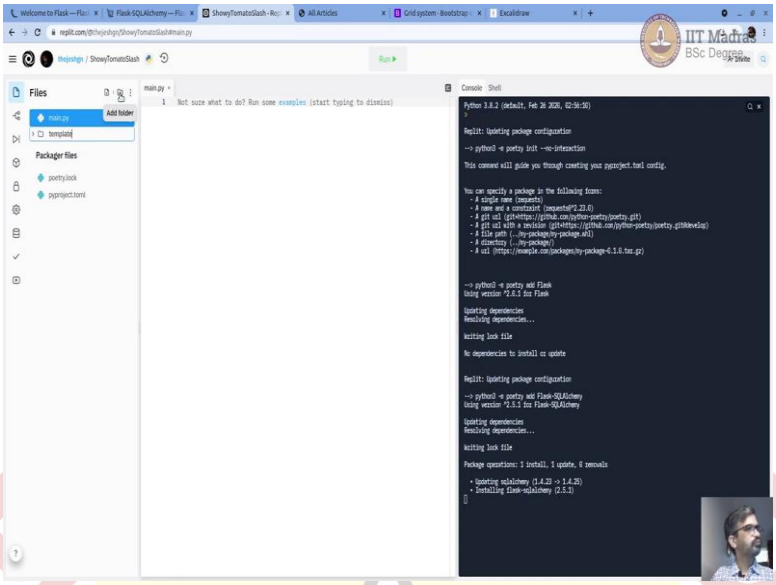
(Refer Slide Time: 17:48)




Packages
flask
20 packages found
Flask
Flask-Zipper
Flask-DB
flask-beans
Flask-Sessions
main.py
Not sure what to do? Run some examples (start typing to dismiss)
Console
Shell
Python 3.8.2 (default, Feb 26 2020, 02:56:10)
Replit: Updating package configuration
--> python3 -m poetry init --no-interaction
This command will guide you through creating your pyproject.toml config.
You can specify a package in the following forms:
- A single name (requests)
- A name and a constraint (requests@^2.23.0)
- A git url (git+https://github.com/python-poetry/poetry.git)
- A git url with a revision (git+https://github.com/python-poetry/poetry.git#develop)
- A file path (../my-package/my-package.whl)
- A directory (../my-package/)
- A url (https://example.com/packages/my-package-0.1.0.tar.gz)
--> python3 -m poetry add Flask
Using version ^2.0.1 for Flask
Updating dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update


Packages
Flask-SQLAlchemy
20 packages found
Flask-SQLAlchemy
Install Package
Flask-Philo-SQLAlchemy
flask-pf-sqlalchemy
Flask-SQLAlchemy-Helpers
main.py
Not sure what to do? Run some examples (start typing to dismiss)
Console
Shell
Python 3.8.2 (default, Feb 26 2020, 02:56:10)
Replit: Updating package configuration
--> python3 -m poetry init --no-interaction
This command will guide you through creating your pyproject.toml config.
You can specify a package in the following forms:
- A single name (requests)
- A name and a constraint (requests@^2.23.0)
- A git url (git+https://github.com/python-poetry/poetry.git)
- A git url with a revision (git+https://github.com/python-poetry/poetry.git#develop)
- A file path (../my-package/my-package.whl)
- A directory (../my-package/)
- A url (https://example.com/packages/my-package-0.1.0.tar.gz)
--> python3 -m poetry add Flask
Using version ^2.0.1 for Flask
Updating dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update


Packages
Flask-SQLAlchemy
20 packages found
Flask-SQLAlchemy
Flask-Philo-SQLAlchemy
flask-pf-sqlalchemy
Flask-SQLAlchemy-Helpers
main.py
Not sure what to do? Run some examples (start typing to dismiss)
Console
Shell
Python 3.8.2 (default, Feb 26 2020, 02:56:10)
Replit: Updating package configuration
--> python3 -m poetry init --no-interaction
This command will guide you through creating your pyproject.toml config.
You can specify a package in the following forms:
- A single name (requests)
- A name and a constraint (requests@^2.23.0)
- A git url (git+https://github.com/python-poetry/poetry.git)
- A git url with a revision (git+https://github.com/python-poetry/poetry.git#develop)
- A file path (../my-package/my-package.whl)
- A directory (../my-package/)
- A url (https://example.com/packages/my-package-0.1.0.tar.gz)
--> python3 -m poetry add Flask
Using version ^2.0.1 for Flask
Updating dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update
Replit: Updating package configuration
--> python3 -m poetry add Flask-SQLAlchemy
Using version ^2.5.1 for Flask-SQLAlchemy
Updating dependencies
Resolving dependencies...
Writing lock file
Package operations: 1 install, 1 update, 0 removals
- Updating sqlalchemy (1.4.23 -> 1.4.25)
- Installing flask-sqlalchemy (2.5.1)

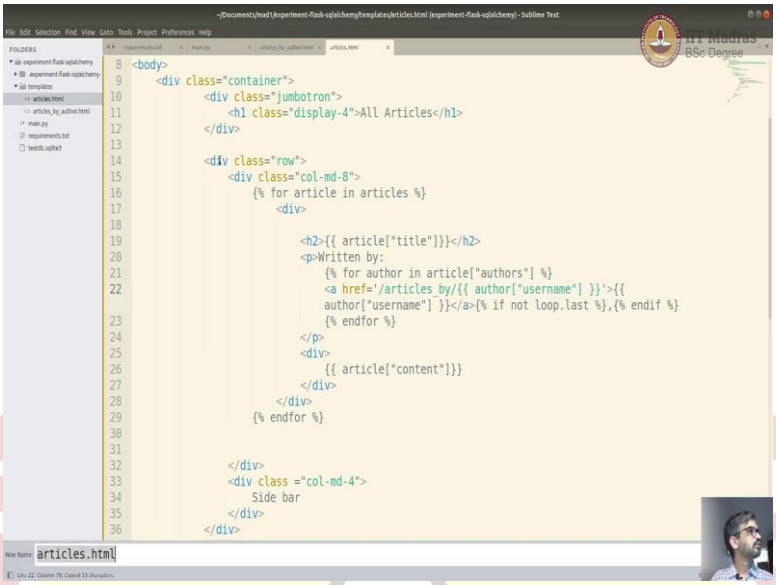


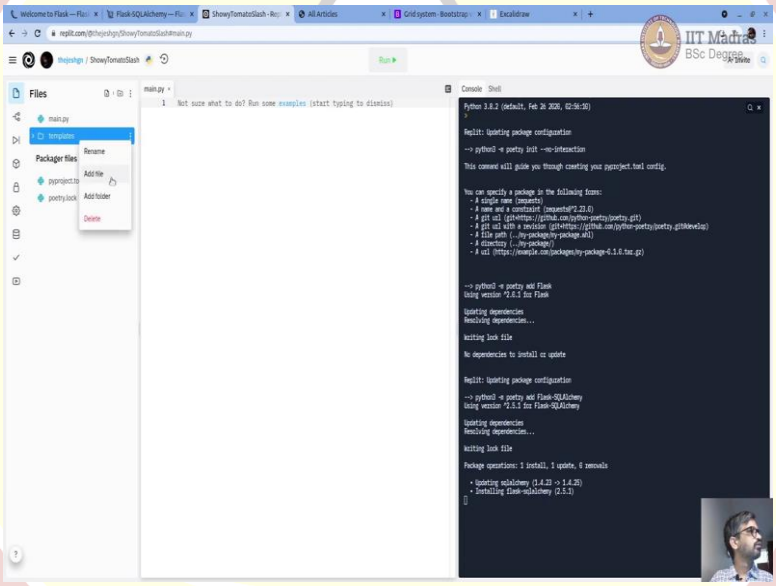
Now it will create a repl and open it. So here, you can, we need to add packages. It is called packages here. What we installed using requirements, we are going to use it, do it using packages here. The first one is Flask. I am gonna search for Flash and click on this plus to install it. It will take a while, it will install Flask. It is installed successfully. Now I am going to paste Flask-SQLAlchemy. Same as our requirements.txt, if you had seen. Same, same this. And click on plus again. So it will install again.

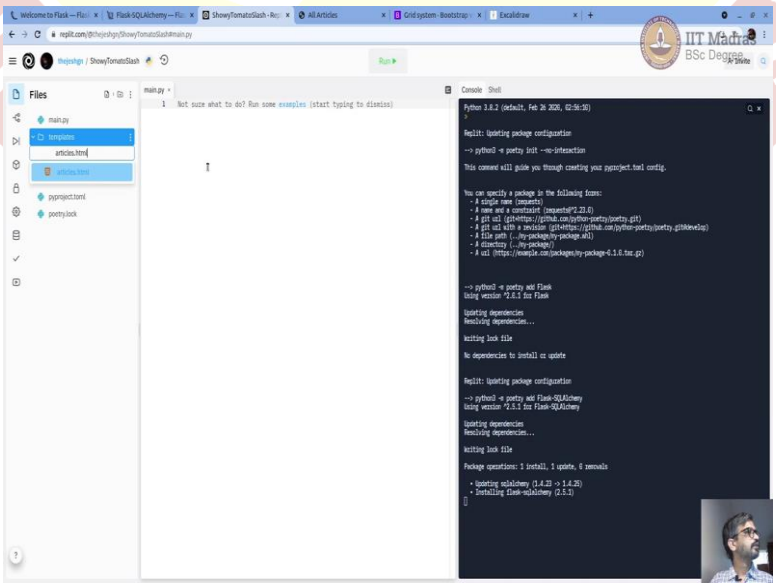
It will take a while because it has some dependencies to install. So we will just wait for a couple of, maybe a minute or two. And you can see the status of installation on the right side. Instead of pip and requirements.txt it uses something called poetry. At this point you do not have to worry about it but you should just know that it uses poetry, and stuff like that, and a very high level.

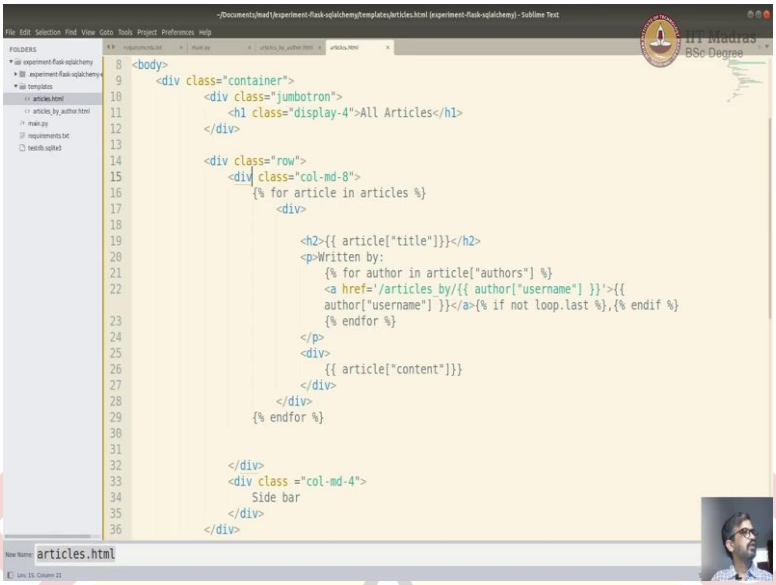
Now it is installed. Let me go back to our files. These are all the files we have, main.py but we do not have other folders and files. So I am just going to create a folder called templates where we are going to put all our templates.

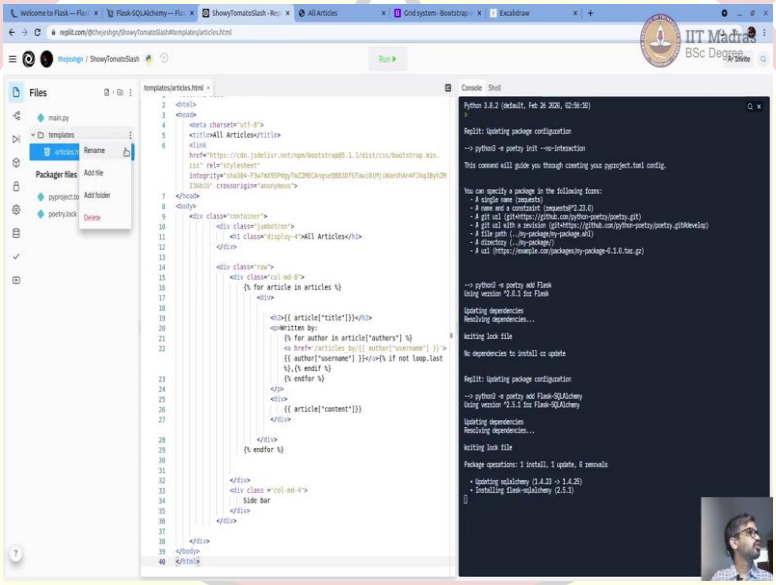
(Refer Slide Time: 19:41)

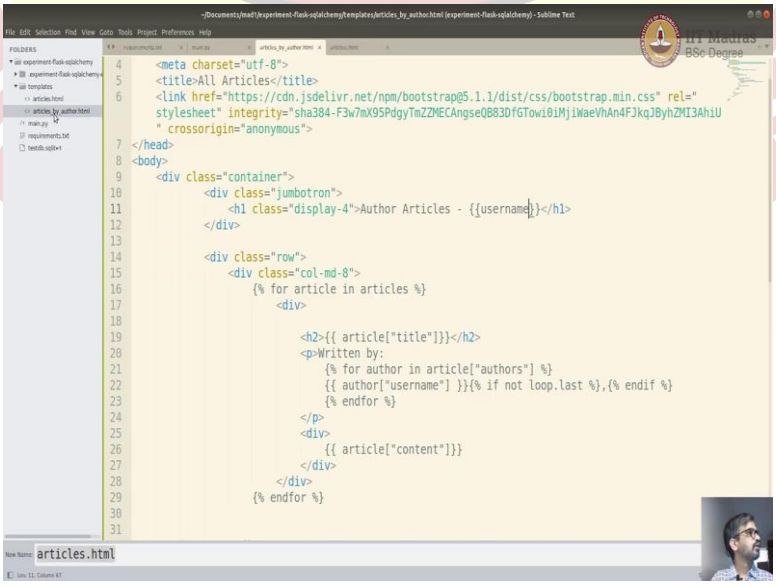


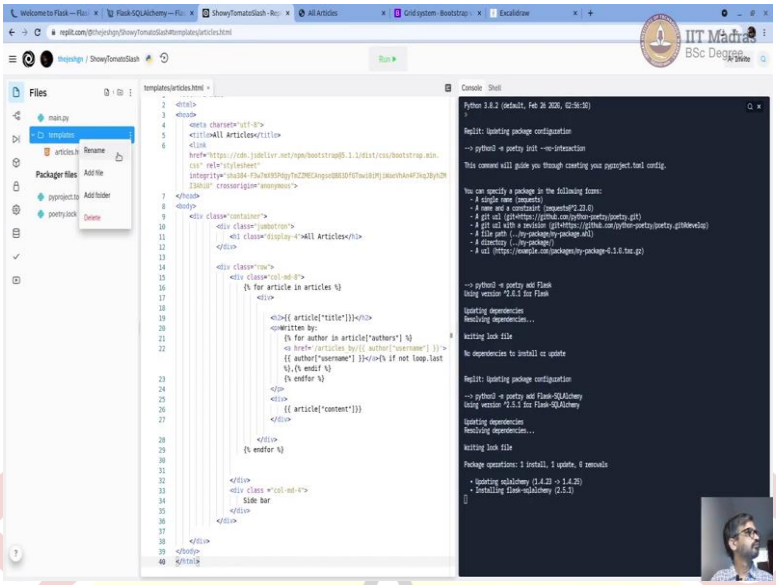


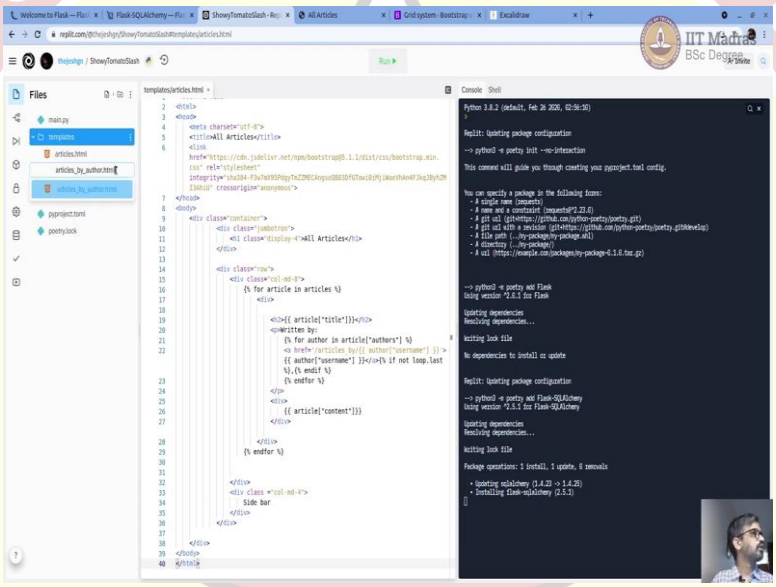


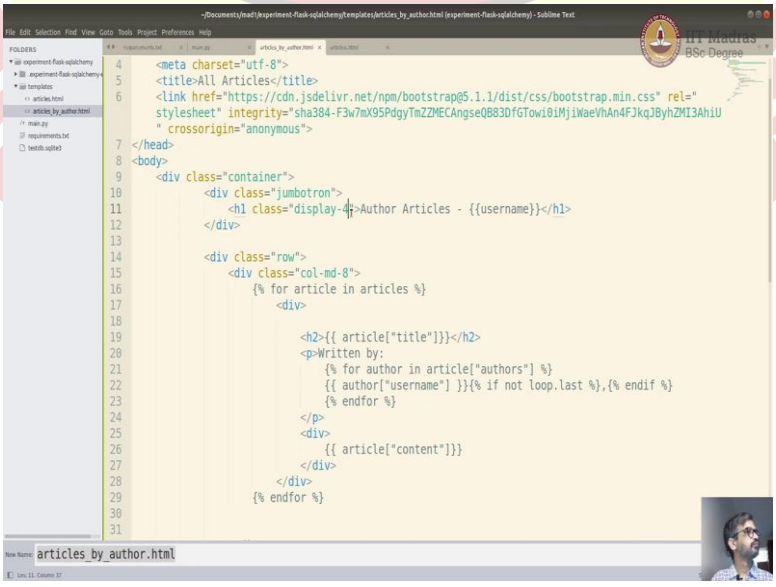

articles.html

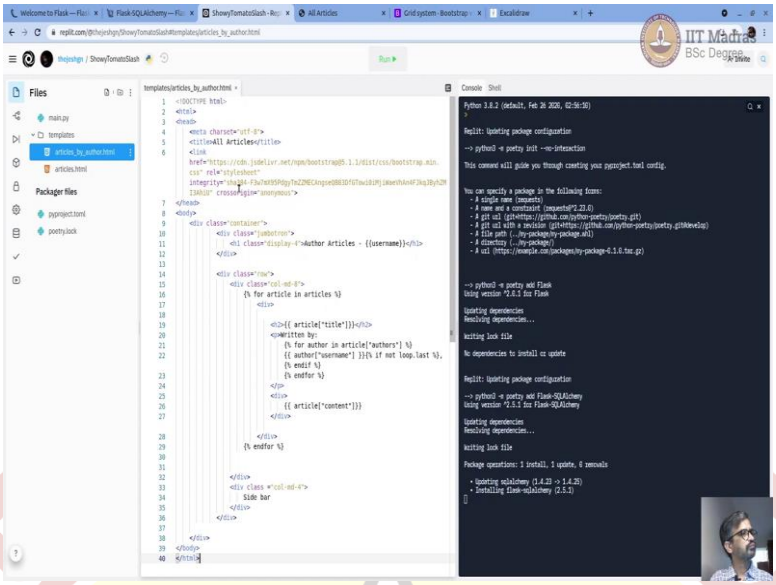



articles.html


templates/articles.html
Console
Shell
Files
main.py
templates
articles.html
articles_by_author.html
pyproject.toml
poetry.lock
Rename
Add file
Add folder
Delete
Packager files
Replit: Updating package configuration
--> python3 -m poetry init --no-interaction
This command will guide you through creating your pyproject.toml config.
--> python3 -m poetry add Flask
Using version ^2.0.1 for Flask
Updating dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update
--> python3 -m poetry add Flask-SQLAlchemy
Using version ^2.5.1 for Flask-SQLAlchemy
Updating dependencies
Resolving dependencies...
Writing lock file
Package operations: 1 install, 1 update, 0 removals
• Updating sqlalchemy (1.4.23 -> 1.4.25)
• Installing flask-sqlalchemy (2.5.1)


templates/articles.html
Console
Shell
Files
main.py
templates
articles.html
articles_by_author.html
pyproject.toml
poetry.lock
Replit: Updating package configuration
--> python3 -m poetry init --no-interaction
This command will guide you through creating your pyproject.toml config.
--> python3 -m poetry add Flask
Using version ^2.0.1 for Flask
Updating dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update
--> python3 -m poetry add Flask-SQLAlchemy
Using version ^2.5.1 for Flask-SQLAlchemy
Updating dependencies
Resolving dependencies...
Writing lock file
Package operations: 1 install, 1 update, 0 removals
• Updating sqlalchemy (1.4.23 -> 1.4.25)
• Installing flask-sqlalchemy (2.5.1)

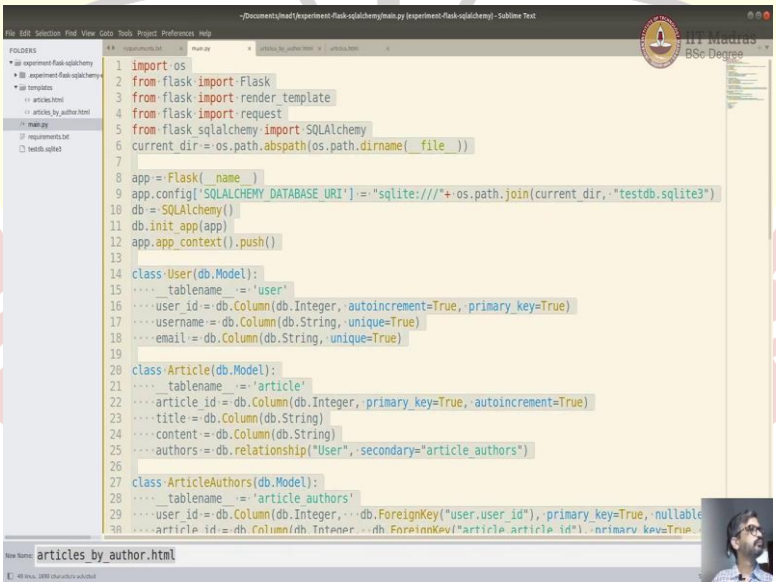

FOLDERS
articles_by_author.html
<h1 class="display-4">Author Articles - {{username}}</h1>
<p>Written by:
{% for article in articles %}
{% for author in article["authors"] %}
{{ author["username"] }}{% if not loop.last %},{% endif %}
{% endfor %}
{{ article["content"]}}
{% endfor %}
New Name: articles_by_author.html

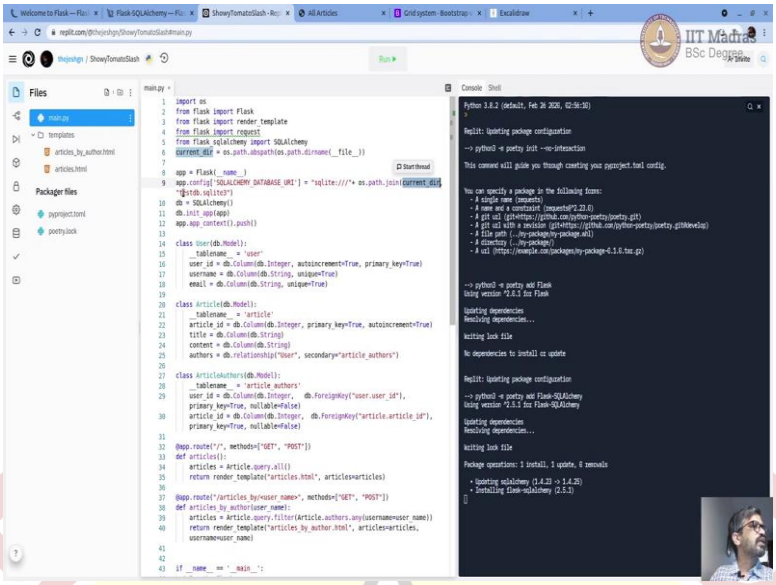


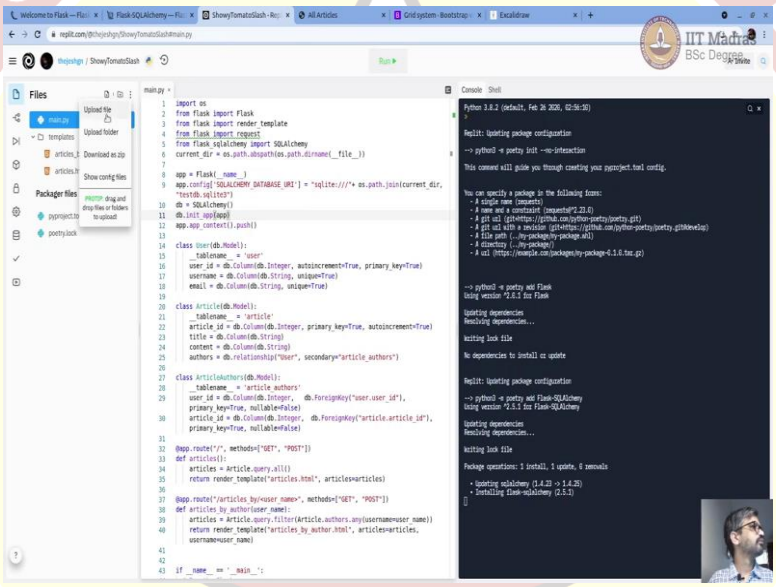
And our first template was Article. So I am going to create, under template, a new, add file, articles, and then copy paste the same thing here.

So I am just going to be pasting the same thing. It should not matter. Like I told you it is very straightforward. Add files. Then create another template called Articles by Author. So now we have both templates.

(Refer Slide Time: 20:36)

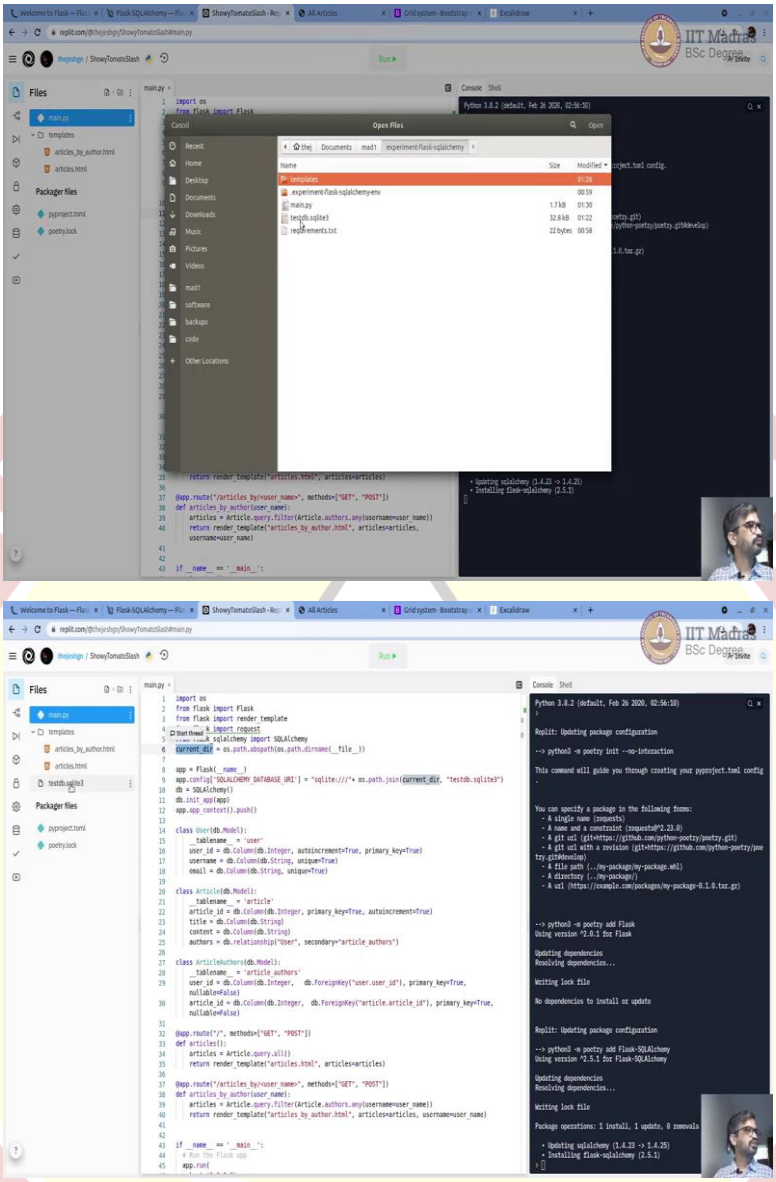






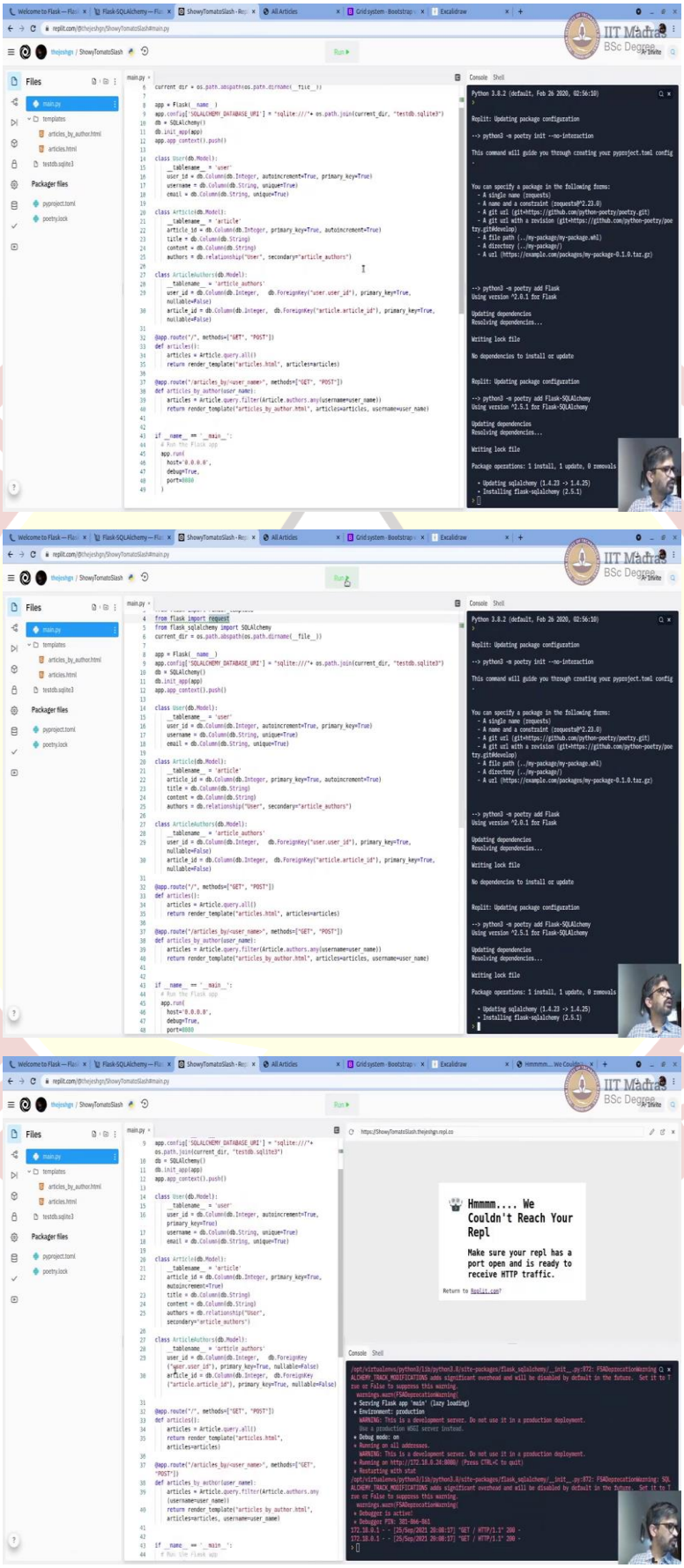
Now, main, I am going to copy paste the main. Here, again, since I am pulling it from OS, current directory, I think it will pick up the, testdb.sqlite3 if it is present here. Since it is not present here I am just going to upload that, whatever we had db, upload that here so that we know we can just use it.

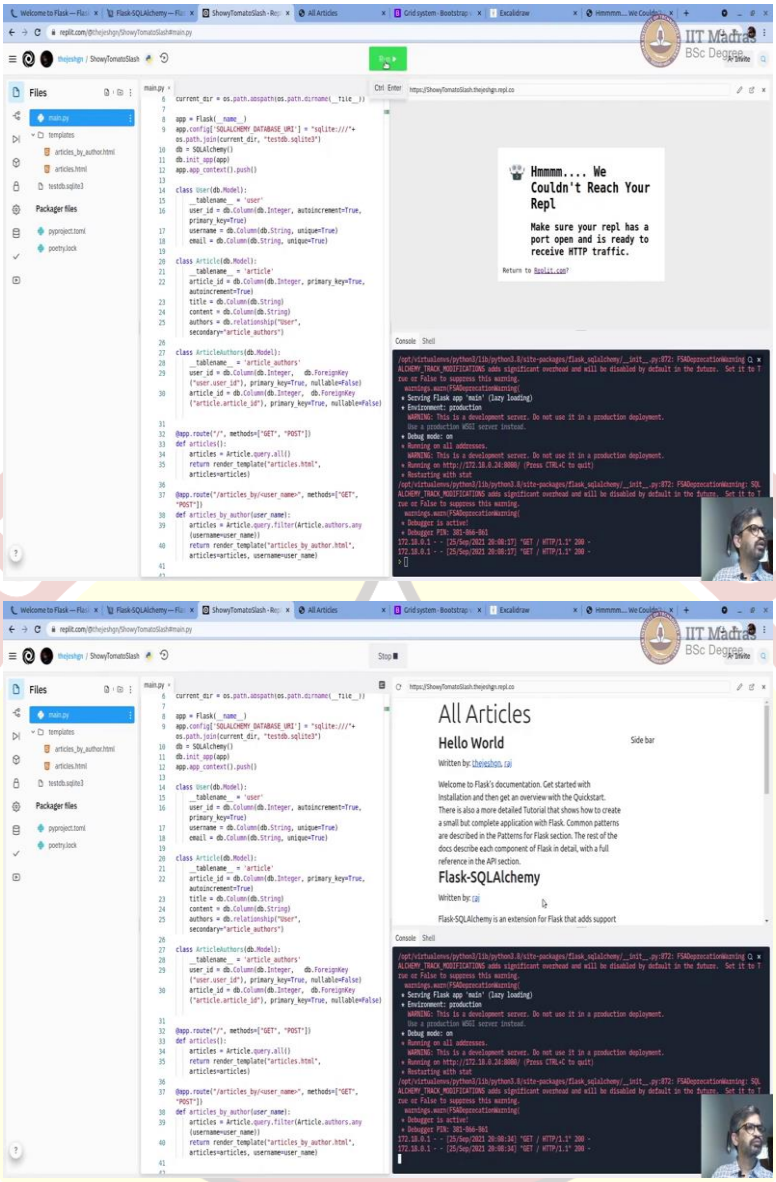
(Refer Slide Time: 21:07)



Just selecting the same db, I am opening it. So it will, it will take a second but then it has uploaded. So now you can see that it has uploaded, and we are using the same thing. Since we are using the current directory and actually dynamically it should not matter, it should still use this testdb.

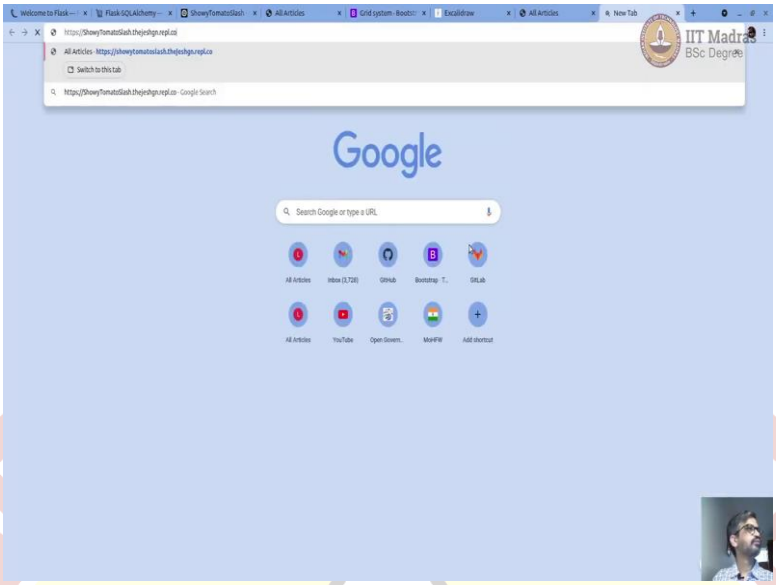
(Refer Slide Time: 21:35)





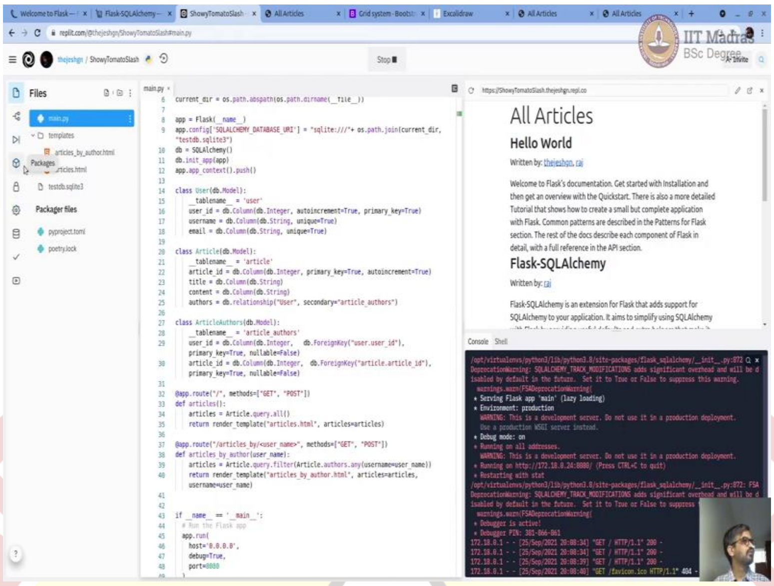
I think now we are set. We can just try to run it. You can just click on Run. It seems to be some network issue. So I am just going to stop and start again. Just click on Run. There you go. It has started. You can see the console here, what it is doing. And it automatically opens up the browser and shows you.

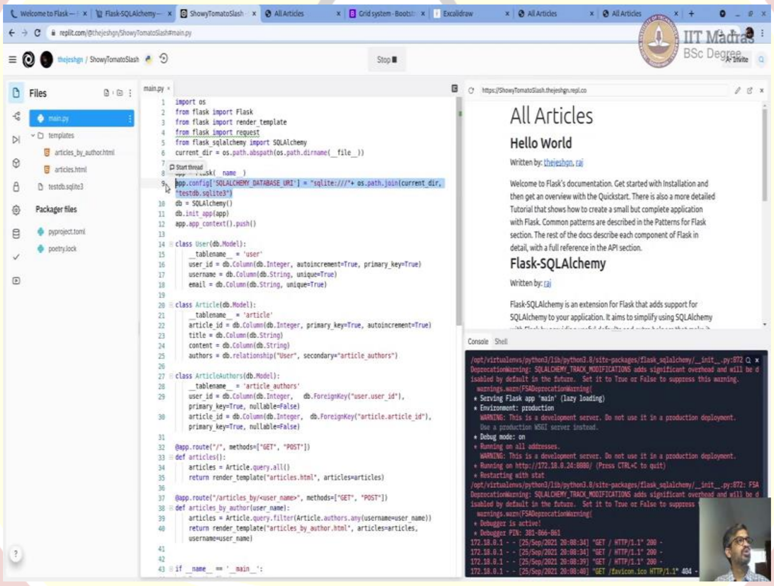
(Refer Slide Time: 22:13)

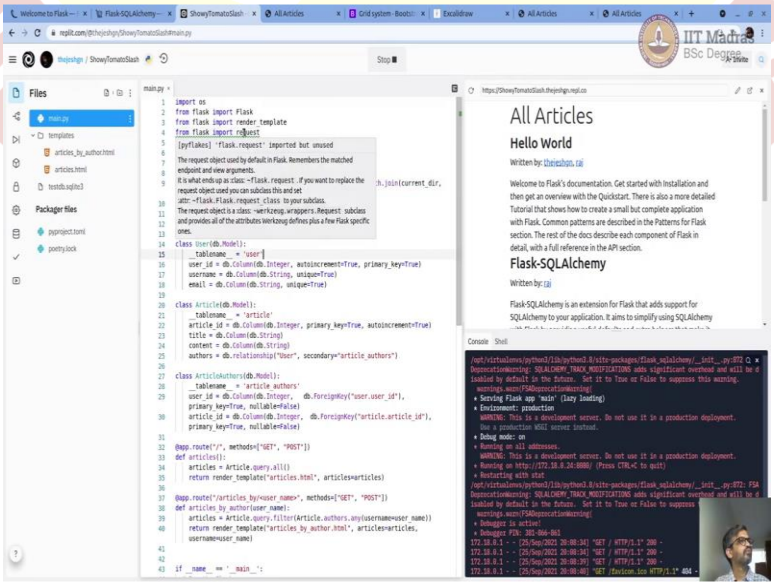


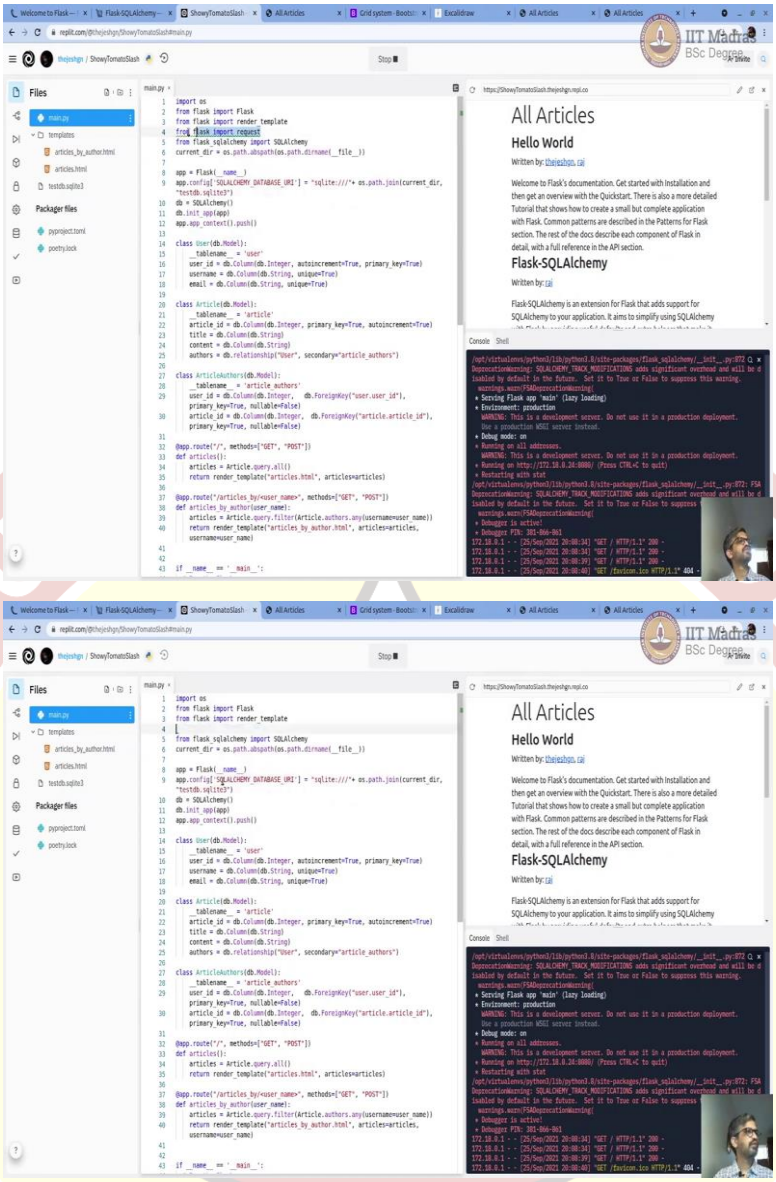
Now you can also run it separately by copying it and running it in a separate window. There you go. It is running on repl now. You can also click on the links, it takes you to the link section, etcetera, etcetera. You can do these things.

(Refer Slide Time: 22:27)









So like I told you, straight forward, now just add the packages using packages part, this section, and the rest of them remain same. And whenever you, whenever, when you are using local variables try to make it dynamic as much as possible. Where this has unused package, so no, wait, no specific error but you do not need to import if you are not using it.

That is it, actually. I think you learned how to use SQLAlchemy using a Flask extension called Flask-SQLAlchemy, and templates. And all should be able to run on replit using the same code without making any changes as such. Thank you. Bye.