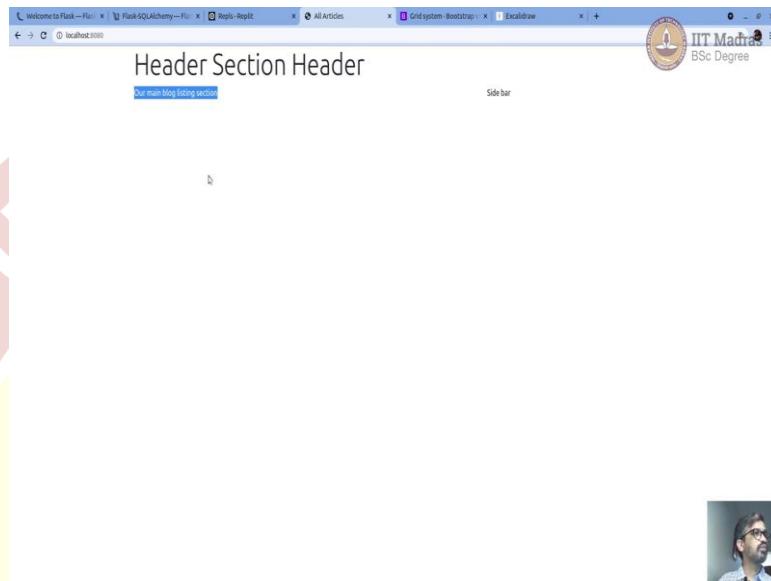


# IIT Madras

## 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)

A screenshot of a Sublime Text editor window. The title bar shows 'File', 'Edit', 'Selection', 'Find', 'View', 'Goto', 'Tools', 'Project', 'Preferences', 'Help'. The status bar shows 'IIT Madras BSc Degree'. The code editor displays the following HTML code:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title> All Articles</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7mX95PdgYtMZZMECAngseQB83DfGTwi0iMj1NaevhAn4FJkqJByhZMI3AhU" crossorigin="anonymous">
</head>
<body>
<div class="container">
<div class="jumbotron">
<h1 class="display-4">Header Section Header</h1>
</div>
<div class="row">
<div class="col-md-8">
{{ articles|join|safe}}
</div>
<div class="col-md-4">
Side bar
</div>
</div>
</div>
</body>
</html>
```



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)



```
-/Documents/Mad/experiment-flask-sqlalchemy/templates/articles.html - (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
• all experiment-flask-sqlalchemy
• experiment-flask-sqlalchemy
• templates
  • articles.html
    • main.py
  requirements.txt
  testdb.sql3
articles.html
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title> All Articles</title>
6     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7mX95PdgyTmZZMECAngseOB83DfGTwi0iMjiWaeVhAn4FJkqJByhZMI3AhilU" crossorigin="anonymous">
7   </head>
8   <body>
9     <div class="container">
10       <div class="jumbotron">
11         <h1 class="display-4">Header Section Header</h1>
12       </div>
13
14       <div class="row">
15         <div class="col-md-8">
16           {% for article in articles %}
17             |
18             {% endfor %}
19           </div>
20         <div class="col-md-4">
21           Side bar
22         </div>
23       </div>
24
25     </div>
26   </body>
27 </html>
```

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)

The screenshot shows the DB Browser for SQLite interface. On the left, there's a table named 'article' with three columns: 'article\_id', 'title', and 'content'. The data in the table is as follows:

article_id	title	content
1	Hello World	This is my first article.
2	H2 content	
3	my new article	my new article content
4	dummy new article	my dummy new article content
5	Using relationships	Use relationships to insert. It's easy
6	2nd Using relationship	2nd use relationships to insert. It's easy

On the right, there is a 'Plot' window with a scatter plot. The x-axis is labeled 'X' and the y-axis is labeled 'Y1'. There are six data points plotted at coordinates (1, 1), (2, 2), (3, 3), (4, 4), (5, 5), and (6, 6). The plot also includes a legend and some axis settings.

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

(Refer Slide Time: 01:12)

The screenshot shows the Sublime Text editor with an HTML file named 'articles.html' open. The file contains the following Jinja2 templating code:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>All Articles</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7z8iDqEEqWZLAFqjJ4OYQZBkZl2f4Mq+uLZtCnqDZGZLJUdHJ" crossorigin="anonymous">
</head>
<body>
<div class="container">
<div class="jumbotron">
<h1 class="display-4">Header Section Header</h1>
</div>
<div class="row">
<div class="col-md-8">
{% for article in articles %}
{{ article["title"]}}
{{ article["content"]}}
{% endfor %}
</div>
<div class="col-md-4">
Side bar
</div>
</div>
</div>
</body>
</html>
```



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)

A screenshot of the Sublime Text code editor. The file is named "articles.html" and contains the following HTML code:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>All Articles</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7mX95PdgY7mZZMECAngseOB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3Ahlu" crossorigin="anonymous">
</head>
<body>
<div class="container">
<div class="jumbotron">
<h1 class="display-4">Header Section Header</h1>
</div>
<div class="row">
<div class="col-md-8">
<% for article in articles %>
<div>
<h2>{{ article["title"] }}</h2>
<{{ article["content"] }}<br>
</div>
<% endfor %>
</div>
<div class="col-md-4">
Side bar
</div>
</div>
</div>
</body>
</html>
```

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)

Header Section Header

Hello World

This is my first article.

h2

my new article

my new article content

dummy new article

my dummy new article content

Using relationship

Use relationships to insert. It's easy

2nd Using relationship

2nd Use relationships to insert. It's easy

IIT Madras  
BSc Degree



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)

```
<meta charset="utf-8">
<title> All Articles</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowI0iMjiaeVhAn4FJkqJByhZMI3AhU" crossorigin="anonymous">
</head>
<body>
<div class="container">
<div class="jumbotron">
<h1 class="display-4">Header Section Header</h1>
</div>
<div class="row">
<div class="col-md-8">
{% for article in articles %}
<div>
<h2>{{ article["title"] }}</h2>
<div>
{{ article["content"] }}
</div>
</div>
{% endfor %}
</div>
<div class="col-md-4">
Side bar
</div>
</div>

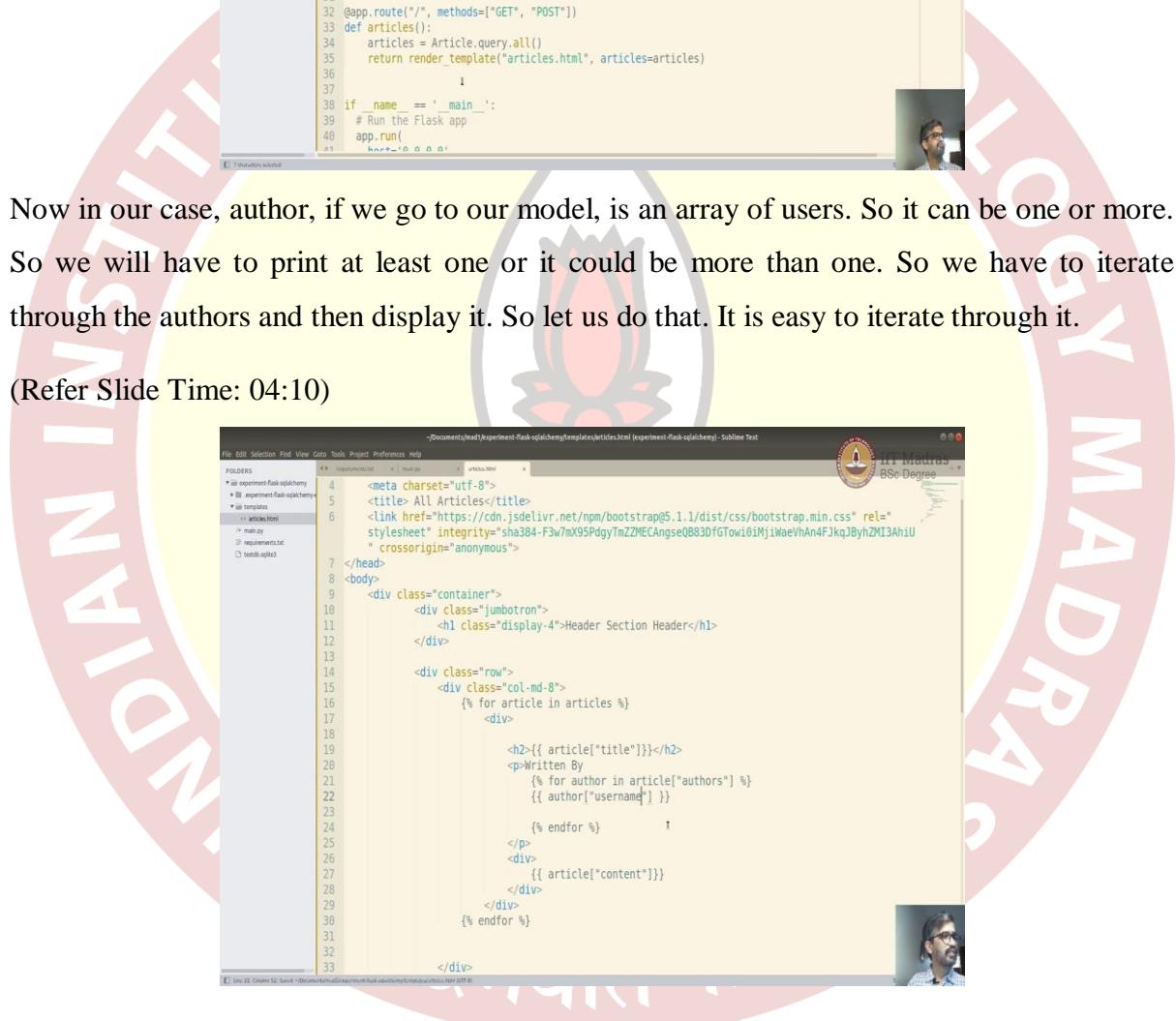
```

IIT Madras  
BSc Degree



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)



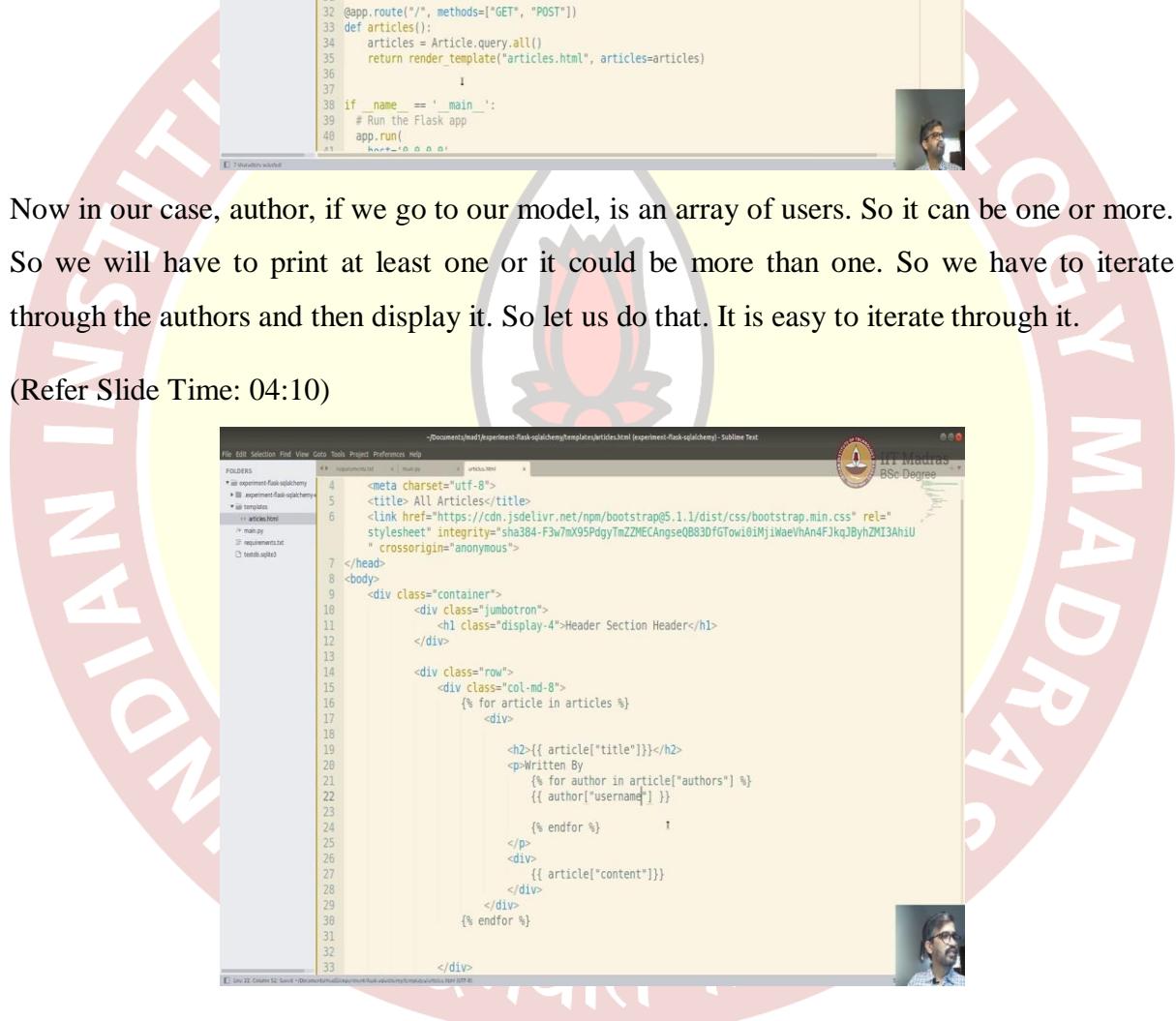
```
-[Documents]\mad\experiment-flask-sqlalchemy\main.py (experiment-flask-sqlalchemy) - Sublime Text
IIT Madras
BSc Degree

FOLDERS
* all experiment-flask-sqlalchemy
* all experiment-flask-sqlalchemy
* all templates
  * articles.html
  * main.html
requirements.txt
tests.sqlite3

10 db = SQLAlchemy()
11 db.init_app(app)
12 app.app_context().push()
13
14 class User(db.Model):
15     __tablename__ = 'user'
16     user_id = db.Column(db.Integer, autoincrement=True, primary_key=True)
17     username = db.Column(db.String, unique=True)
18     email = db.Column(db.String, unique=True)
19
20 class Article(db.Model):
21     __tablename__ = 'article'
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=False)
30     article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, nullable=False)
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
38 if __name__ == '__main__':
39     # Run the Flask app
40     app.run(
        host='0.0.0.0',
        port=5000,
        debug=True)
```

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.

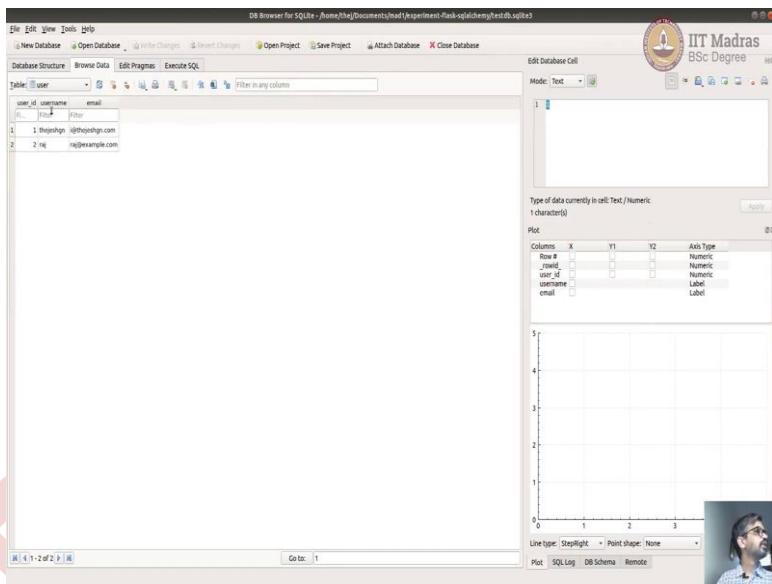
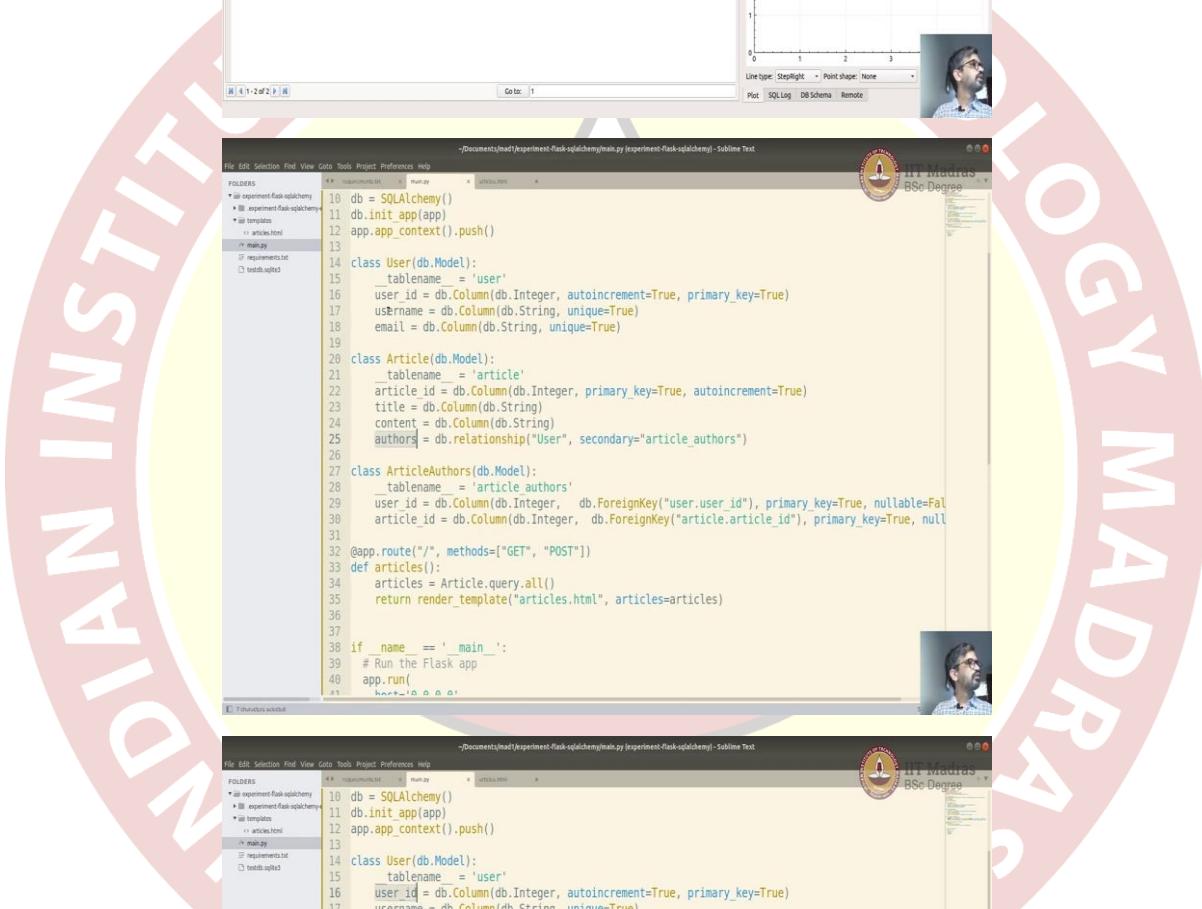
(Refer Slide Time: 04:10)



```
-[Documents]\mad\experiment-flask-sqlalchemy\templates\articles.html (experiment-flask-sqlalchemy) - Sublime Text
IIT Madras
BSc Degree

FOLDERS
* all experiment-flask-sqlalchemy
* all experiment-flask-sqlalchemy
* all templates
  * articles.html
  * main.html
requirements.txt
tests.sqlite3

4 <meta charset="utf-8">
5 <title> All Articles </title>
6 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-FxwvDABoRzT00q/l1kjZqqlJWYkHn4FJkqJBhZM13AhU" crossorigin="anonymous">
7 </head>
8 <body>
9   <div class="container">
10    <div class="jumbotron">
11      <h1 class="display-4">Header Section Header</h1>
12    </div>
13
14    <div class="row">
15      <div class="col-md-8">
16        {% for article in articles %}
17          <div>
18            <h2>{{ article["title"] }}</h2>
19            <p>Written By
20              {% for author in article["authors"] %}
21                {{ author["username"] }}
22              {% endfor %}
23            </p>
24            <div>
25              {{ article["content"] }}
26            </div>
27          {% endfor %}
28        </div>
29      {% endfor %}
30    </div>
31
32  </div>
```



```
->Documents\mad\experiment-flask-sqlalchemy\main.py (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
• all experiment-flask-sqlalchemy
• III experiment-flask-sqlalchemy
• III templates
• III articles.html
• main.py
requirements.txt
testdb.sqlite3
10 db = SQLAlchemy()
11 db.init_app(app)
12 app.app_context().push()

13 class User(db.Model):
14     __tablename__ = 'user'
15     user_id = db.Column(db.Integer, autoincrement=True, primary_key=True)
16     username = db.Column(db.String, unique=True)
17     email = db.Column(db.String, unique=True)

18 class Article(db.Model):
19     __tablename__ = 'article'
20     article_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
21     title = db.Column(db.String)
22     content = db.Column(db.String)
23     authors = db.relationship("User", secondary="article_authors")

24 class ArticleAuthors(db.Model):
25     __tablename__ = 'article_authors'
26     user_id = db.Column(db.Integer, db.ForeignKey("user.user_id"), primary_key=True, nullable=False)
27     article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, nullable=False)

28 @app.route("/", methods=["GET", "POST"])
29 def articles():
30     articles = Article.query.all()
31     return render_template("articles.html", articles=articles)

32 if __name__ == '__main__':
33     # Run the Flask app
34     app.run(
35         host='0.0.0.0',
36         port=5000,
37         debug=True)
```

```
->Documents\mad\experiment-flask-sqlalchemy\main.py (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
• all experiment-flask-sqlalchemy
• III experiment-flask-sqlalchemy
• III templates
• III articles.html
• main.py
requirements.txt
testdb.sqlite3
10 db = SQLAlchemy()
11 db.init_app(app)
12 app.app_context().push()

13 class User(db.Model):
14     __tablename__ = 'user'
15     user_id = db.Column(db.Integer, autoincrement=True, primary_key=True)
16     username = db.Column(db.String, unique=True)
17     email = db.Column(db.String, unique=True)

18 class Article(db.Model):
19     __tablename__ = 'article'
20     article_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
21     title = db.Column(db.String)
22     content = db.Column(db.String)
23     authors = db.relationship("User", secondary="article_authors")

24 class ArticleAuthors(db.Model):
25     __tablename__ = 'article_authors'
26     user_id = db.Column(db.Integer, db.ForeignKey("user.user_id"), primary_key=True, nullable=False)
27     article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, nullable=False)

28 @app.route("/", methods=["GET", "POST"])
29 def articles():
30     articles = Article.query.all()
31     return render_template("articles.html", articles=articles)

32 if __name__ == '__main__':
33     # Run the Flask app
34     app.run(
35         host='0.0.0.0',
36         port=5000,
37         debug=True)
```

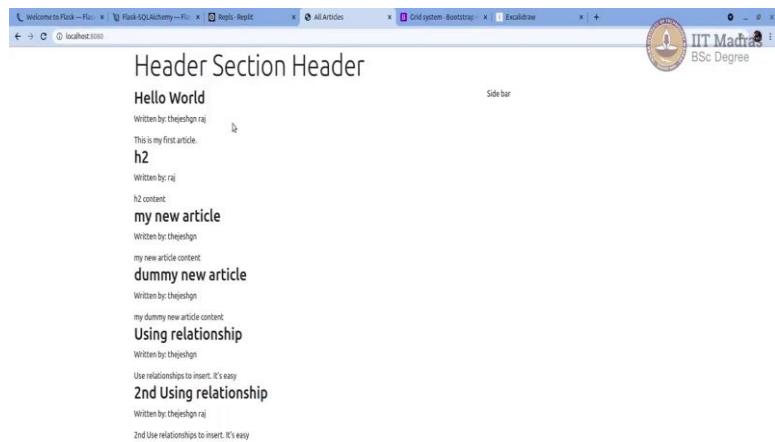


Let us do one. Let us put the author names in inside the P tag, below it. Let us say written by. So there can be more than one so even that will be a For loop. So For loop will be article authors, around article authors. So written by looping through authors from article and I am getting author for each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

```
-/Documents/med/experiment-flask-sqlalchemy/templates/articles.html - (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
• all experiment-flask-sqlalchemy
• all experiment-flask-sqlalchemy
• all templates
  • articles.html
    • main.py
    • requirements.txt
    • tests.sql3
4   <meta charset="utf-8">
5   <title> All Articles</title>
6   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7X95PdgY7mZZMECAngseQB83DfGTow101Mj1iaeVhAn4FJkqJByhZMI3AhU" crossorigin="anonymous">
7   </head>
8   <body>
9     <div class="container">
10       <div class="jumbotron">
11         <h1 class="display-4">Header Section Header</h1>
12       </div>
13
14       <div class="row">
15         <div class="col-md-8">
16           {% for article in articles %}
17             <div>
18               <h2>{{ article["title"] }}</h2>
19               <p>Written by:
20                 {% for author in article["authors"] %}
21                   {{ author["username"] }}&nbsp;
22
23                   {% endfor %}
24               </p>
25               <div>
26                 {{ article["content"] }}
27               </div>
28             {% endfor %}
29           </div>
30
31
32
33       </div>
```

Header Section Header

Hello World  
Written by: thejeshgn, raj.  
This is my first article.

**h2**  
Written by: raj.  
h2 content:

**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



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)

```
-Documents\mad\experiment-flask-sqlalchemy\templates\articles.html - Sublime Text
IIT Madras
BSc Degree

File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  • all experiment-flask-sqlalchemy
  • all experiment-flask-sqlalchemy
  • all templates
    • articles.html
      • main.py
      • requirements.txt
      • testdb.sqlite
4   <meta charset="utf-8">
5   <title> All Articles</title>
6   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7mX95PdgTmZZMECngseQB83DfGTow101MjiiaeVhAn4FJkqJByhZMI3AhU" crossorigin="anonymous">
7   </head>
8   <body>
9     <div class="container">
10       <div class="jumbotron">
11         <h1 class="display-4">Header Section Header</h1>
12       </div>
13
14       <div class="row">
15         <div class="col-md-8">
16           {% for article in articles %}
17             <div>
18
19               <h2>{{ article["title"] }}</h2>
20               <p>Written by:
21                 {% for author in article["authors"] %}
22                   {{ author["username"] }}
23                 {% if not loop.last %}
24                   |
25                 {% endif %}
26
27                 {% endfor %}
28               </p>
29               <div>
30                 {{ article["content"] }}
31               </div>
32             </div>
33           {% endfor %}

```



The screenshot shows a web browser window with multiple tabs open. The main content area displays a list of articles under a header titled "Header Section Header". Each article entry includes the title, author (written by: thejeshgn), and content. The articles listed are "Hello World", "my new article", "dummy new article", "Using relationship", and "2nd Using relationship". The browser's address bar shows "localhost:8000". A sidebar on the right contains the IIT Madras logo and the text "IIT Madras BSc Degree".

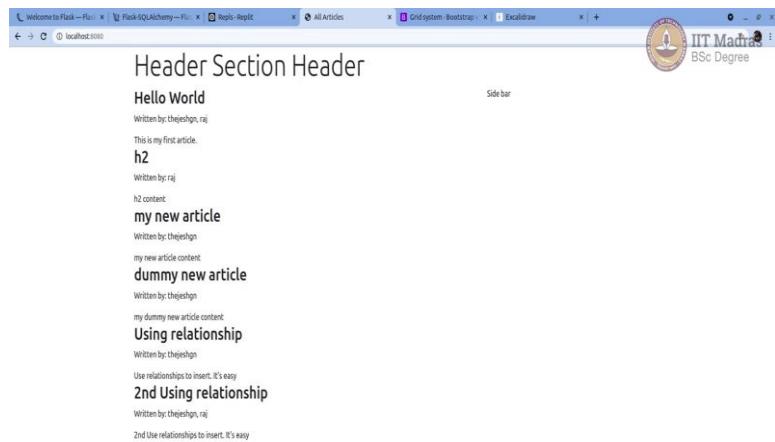


We can, let us do that. We can do if not last, if not the last item, then we can put the comma. So if this is not the last item in the loop then put a comma. Now you can see that because this is the last item in the loop, it is not putting comma otherwise it is putting comma. But there is like a space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

(Refer Slide Time: 07:16)

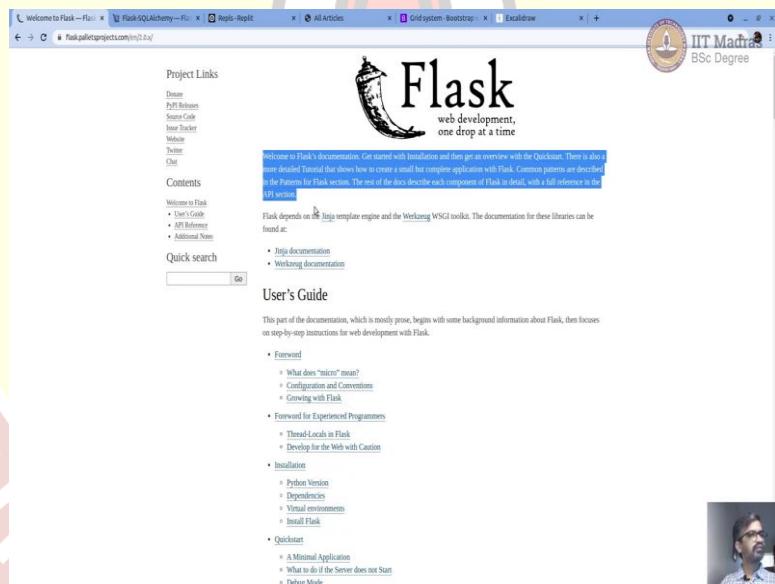
The screenshot shows the Sublime Text editor with the file "articles.html" open. The code is an HTML template for displaying a list of articles. It includes a header section with a jumbotron and a row section with a column for each article. The code uses Jinja2 templating syntax, including loops and conditionals. The browser tab above the editor shows "welcome to Flask - File - flask-sqlalchemy - File - Apps - Replic - All Articles - Gridsystem - Bootstrap - Exceldraw - Side bar". The status bar at the bottom indicates "line 23, column 31, Sublime - Documents\mad\experiment-flask-sqlalchemy\templates\articles.html (experiment-flask-sqlalchemy) - Sublime Text".





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)



The image shows a composite screenshot of two windows. The left window is 'DB Browser for SQLite' showing a table of articles with columns article\_id, title, and content. The right window is a Flask application's homepage displaying a list of articles with titles like 'Hello World', 'h2 content', 'my new article', etc., and a sidebar with a user profile picture.

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)

welcome to Flask — File — flask-SQLAlchemy — File — Repls — Replics — All Articles — Gridsystem - Bootstrap — ExcelDraw — IIT Madras BSc Degree

**Flask SQLAlchemy**

Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simply 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.

**Requirements**

Our Version	Python	Flask	SQLAlchemy
2.x	2.7, 3.4+	0.12+	0.8+ or 1.0.10+ w/ Python 3.7
3.0+ (in dev)	2.7, 3.5+	1.0+	1.0+

**User Guide**

- Quickstart
  - A Minimal Application
  - Simple Relationships
  - Road to Enlightenment
- Introduction into Contexts
- Configuration
  - Configuration Keys
  - Connection URI Format
  - Using custom Metadatas and naming conventions
  - Timers
- Declaring Models
  - Simple Example
  - One-to-Many Relationships
  - Many-to-Many Relationships
- Select, Insert, Delete
  - Inserting Records
  - Deleting Records
  - Querying Records

Go

Project Links

- Donate to Pallets
- Website
- PyPI releases
- Source code
- Issue tracker

Contents

Flask-SQLAlchemy

- Requirements
- User Guide
- API Reference
- Additional Information

Quick search

DB Browser for SQLite - /home/hkg/Documents/mediawiki/experiment-flask-sqlalchemy/testdb.sqlite

IIT Madras BSc Degree

Mode: Text

1 Plus SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simply using SQLAlchemy with Flask by providing useful defaults and extra helpers that make it easier to accomplish common tasks.  
2 See the SQLAlchemy documentation to learn how to work with the ORM in depth.  
3 This following document is a brief overview of the most common tasks, as well as the features specific to Flask-SQLAlchemy.

Type of data currently in cell: Text / Numeric  
445 character(s)

Plot

Columns X Y1 Y2 Axis Type  
Row #    Numeric  
\_rowid\_    Numeric  
article\_id    Numeric  
title    Label  
content    Label

5  
4  
3  
2  
1

Line type: Step/light Point shape: None  
Plot SQL Log DB Schema Remote

DB Browser for SQLite - /home/hkg/Documents/mediawiki/experiment-flask-sqlalchemy/testdb.sqlite

IIT Madras BSc Degree

Mode: Text

1 Flask-SQLAlchemy

Type of data currently in cell: Text / Numeric  
16 character(s)

Plot

Columns X Y1 Y2 Axis Type  
Row #    Numeric  
\_rowid\_    Numeric  
article\_id    Numeric  
title    Label  
content    Label

5  
4  
3  
2  
1

Line type: Step/light Point shape: None  
Plot SQL Log DB Schema Remote

DB Browser for SQLite - /home/hkg/Documents/mediawiki/experiment-flask-sqlalchemy/testdb.sqlite

IIT Madras BSc Degree

Mode: Text

1 Flask-SQLAlchemy

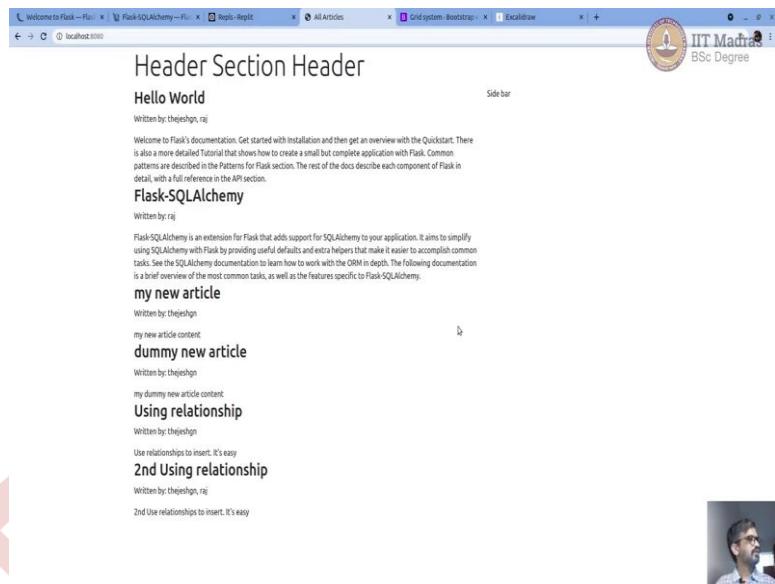
Type of data currently in cell: Text / Numeric  
16 character(s)

Plot

Columns X Y1 Y2 Axis Type  
Row #    Numeric  
\_rowid\_    Numeric  
article\_id    Numeric  
title    Label  
content    Label

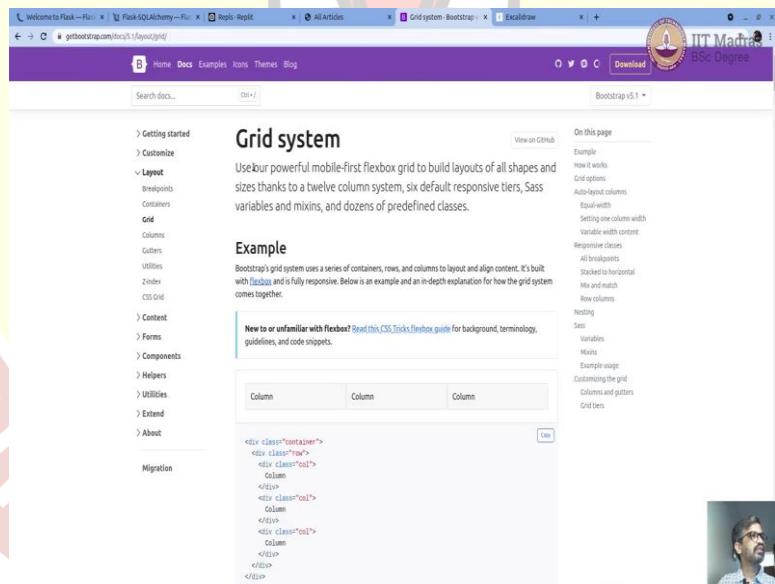
5  
4  
3  
2  
1

Line type: Step/light Point shape: None  
Plot SQL Log DB Schema Remote



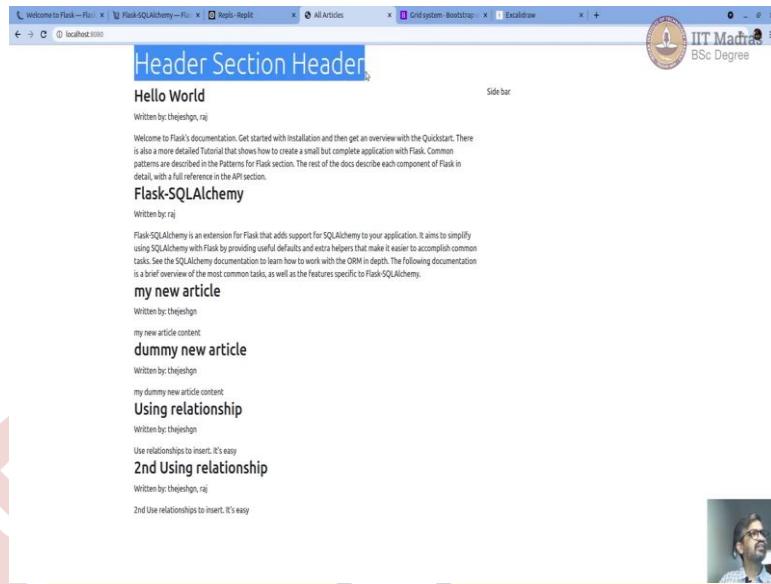
Let us add some more content to some other things just to make it little more useful content instead of dummy content. And give a title also, meaningful title. I have saved it, let us come back here and see. So now you can see that there is, there is some decent looking blog listing.

(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)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>All Articles</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7M95PdgTmZZMEAngse0B83DfGTow1o1Mj1iaeVhAn4FJkqJByhZMI3AhU" crossorigin="anonymous">
</head>
<body>
<div class="container">
<div class="jumbotron">
<h1 class="display-4">Header Section Header
</div>
<div class="row">
<div class="col-md-8">
{% for article in articles %}
<div>
<h2>{{ article["title"] }}</h2>
<p>Written by:
    {% for author in article["authors"] %}
    {{ author["username"] }}{% if not loop.last %},{% endif %}
    {% endfor %}
</p>
<div>
    {{ article["content"] }}
</div>
</div>
{% endfor %}
</div>
</div>
</div>

```

सद्विर्भवति कर्मजा



Let us, because it is all articles I am just going to put the same header there too, in the jumbotron. 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)

```

File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  • all experiment Flask-SQLAlchemy
  • all experiment-flask-sqlalchemy
  • all templates
  • articles.html
  • main.py
  • requirements.txt
  • testdb.sqlite

main.py
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=False)
30     article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, nullable=False)
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     )

```



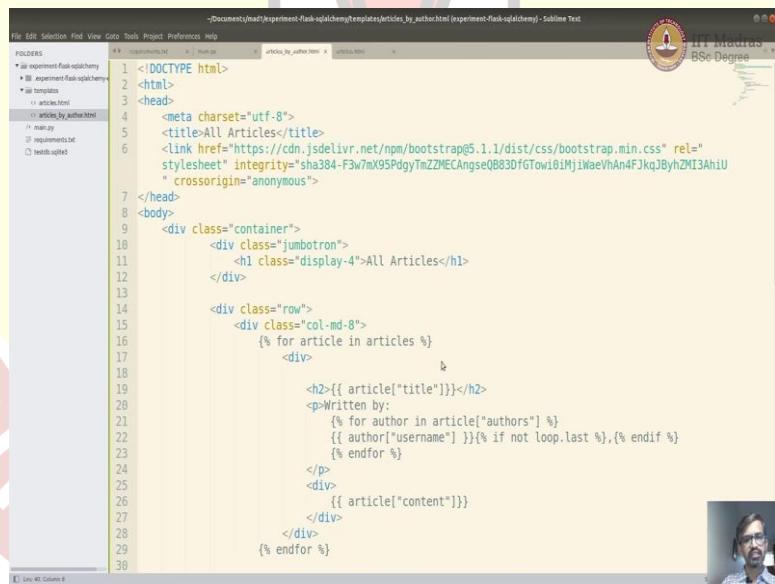
So for that, we have to have some format where we can pass the user name to the articles listing and then filter it by that username, for which we will write another model, we will not use the same model. Just to keep it simple we will add another model, and we will call it get Articles by username or 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 template 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)



The screenshot shows a Sublime Text editor window with the file 'articles\_by\_author.html' open. The code is an HTML template with Jinja2 syntax. It includes a header section with meta charset and title, a link to a Bootstrap CSS file, and a main body section. The body starts with a jumbotron containing a heading 'All Articles'. Below it is a row of columns where each column contains an article's title, author information (username), and content. The code uses loops and conditionals to structure the data. The Sublime Text interface shows a sidebar with project files and a status bar indicating the file is saved.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>All Articles</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7mXGj5DyIqB0WtRdiQO9jwHsz0GvUZKuOkfklQSsPQ4XsJ3MjW0zJ4" crossorigin="anonymous">
</head>
<body>
<div class="container">
<div class="jumbotron">
<h1 class="display-4">All Articles</h1>
</div>
<div class="row">
<div class="col-md-8">
<% for article in articles %>
<div>
<h2>{{ article["title"] }}</h2>
<p>Written by:<br/>
<% for author in article["authors"] %>
{{ author["username"] }}<% if not loop.last %>,<% endif %>
<% endfor %>
</p>
<div>
{{ article["content"] }}
</div>
</div>
<% endfor %>
</div>
</div>

```

INDIAN INSTITUTE

LOGY MADA

```
->Documents\madi\experiment-flask-sqlalchemy\main.py (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find Goto Tools Project Preferences Help
FOLDERS
  • all experiment-flask-sqlalchemy
  • all experiment-flask-sqlalchemy
  • all templates
    • articles.html
    • articles_by_author.html
  • main.py
  • requirements.txt
  • tests.sqlite3
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

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=False)
    article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, nullable=False)

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

@app.route("/articles_by<user_name>", methods=["GET", "POST"])
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)

if __name__ == '__main__':
    # Run the Flask app
    app.run(
        host='0.0.0.0',
        debug=True,
        port=8080
    )
```



```
->Documents\madi\experiment-flask-sqlalchemy\main.py (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find Goto Tools Project Preferences Help
FOLDERS
  • all experiment-flask-sqlalchemy
  • all experiment-flask-sqlalchemy
  • all templates
    • articles.html
    • articles_by_author.html
  • main.py
  • requirements.txt
  • tests.sqlite3
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

authors = db.relationship("User", secondary="article_authors")

class ArticleAuth(db.Model):
    tablename = 'article_authors'
    user_id = db.Column(db.Integer, db.ForeignKey("user.user_id"), primary_key=True, nullable=False)
    article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, nullable=False)

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

@app.route("/articles_by<user_name>", methods=["GET", "POST"])
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)

if __name__ == '__main__':
    # Run the Flask app
    app.run(
        host='0.0.0.0',
        debug=True,
        port=8080
    )
```



```
->Documents\madi\experiment-flask-sqlalchemy\main.py (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find Goto Tools Project Preferences Help
FOLDERS
  • all experiment-flask-sqlalchemy
  • all experiment-flask-sqlalchemy
  • all templates
    • articles.html
    • articles_by_author.html
  • main.py
  • requirements.txt
  • tests.sqlite3
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

authors = db.relationship("User", secondary="article_authors")

class ArticleAuth(db.Model):
    tablename = 'article_authors'
    user_id = db.Column(db.Integer, db.ForeignKey("user.user_id"), primary_key=True, nullable=False)
    article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, nullable=False)

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

@app.route("/articles_by<user_name>", methods=["GET", "POST"])
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)

if __name__ == '__main__':
    # Run the Flask app
    app.run(
        host='0.0.0.0',
        debug=True,
        port=8080
    )
```





The screenshot shows a Sublime Text editor window with a terminal window open below it. The terminal window displays the command 'python main.py' and its output. The output includes several warning messages from SQLAlchemy regarding track modifications and deployment environments. A small portrait of a man with glasses is visible in the bottom right corner.

```

File Edit Selection Find Goto Tools Project Preferences Help
FOLDERS
  • all experiment-flask-sqlalchemy
  • all experiment-flask-sqlalchemy
  • all templates
    • articles.html
    • articles_by_author.html
  • main.py
  • resources/sqlite
  • tests.sqlite

-/Documents/nadl/experiment-flask-sqlalchemy/main.py (experiment-flask-sqlalchemy) - Sublime Text
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=False)
30     article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, nullable=False)
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=8088
49     )

```

```

File Edit View Search Terminal Help
-/Documents/nadl/experiment-flask-sqlalchemy/main.py (experiment-flask-sqlalchemy) - Terminal
python main.py
/home/thej/Documents/nadl/experiment-flask-sqlalchemy/experiment-flask-sqlalchemy/_init_.py:873: FSDeprecationWarning: SOLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
  'SOLALCHEMY TRACK MODIFICATIONS adds significant overhead and '
WARNING: This is a development server. Do not use it in a production deployment.
  * Serving Flask app 'main' (lazy loading)
  * Environment: production
  * Debug mode: on
  * Running on all addresses.
  * WARNING: This is a development server. Do not use it in a production deployment
  * Running on http://192.168.0.209:8088/ (Press CTRL+C to quit)
  * Restarting with stat

```

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)



The screenshot shows a Sublime Text editor window with a terminal window open below it. The terminal window displays the command 'python main.py' and its output. The output includes several warning messages from SQLAlchemy regarding track modifications and deployment environments. A small portrait of a man with glasses is visible in the bottom right corner.

```

File Edit Selection Find Goto Tools Project Preferences Help
FOLDERS
  • all experiment-flask-sqlalchemy
  • all experiment-flask-sqlalchemy
  • all templates
    • articles.html
    • articles_by_author.html
  • main.py
  • resources/sqlite
  • tests.sqlite

-/Documents/nadl/experiment-flask-sqlalchemy/main.py (experiment-flask-sqlalchemy) - Sublime Text
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=False)
30     article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, nullable=False)
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=8088
49     )

```

```

File Edit View Search Terminal Help
-/Documents/nadl/experiment-flask-sqlalchemy/main.py (experiment-flask-sqlalchemy) - Terminal
python main.py
/home/thej/Documents/nadl/experiment-flask-sqlalchemy/experiment-flask-sqlalchemy/_init_.py:873: FSDeprecationWarning: SOLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
  'SOLALCHEMY TRACK MODIFICATIONS adds significant overhead and '
WARNING: This is a development server. Do not use it in a production deployment.
  * Serving Flask app 'main' (lazy loading)
  * Environment: production
  * Debug mode: on
  * Running on all addresses.
  * WARNING: This is a development server. Do not use it in a production deployment
  * Running on http://192.168.0.209:8088/ (Press CTRL+C to quit)
  * Restarting with stat

```

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.

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 User relationships to insert. It's easy

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: thejeshgn

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

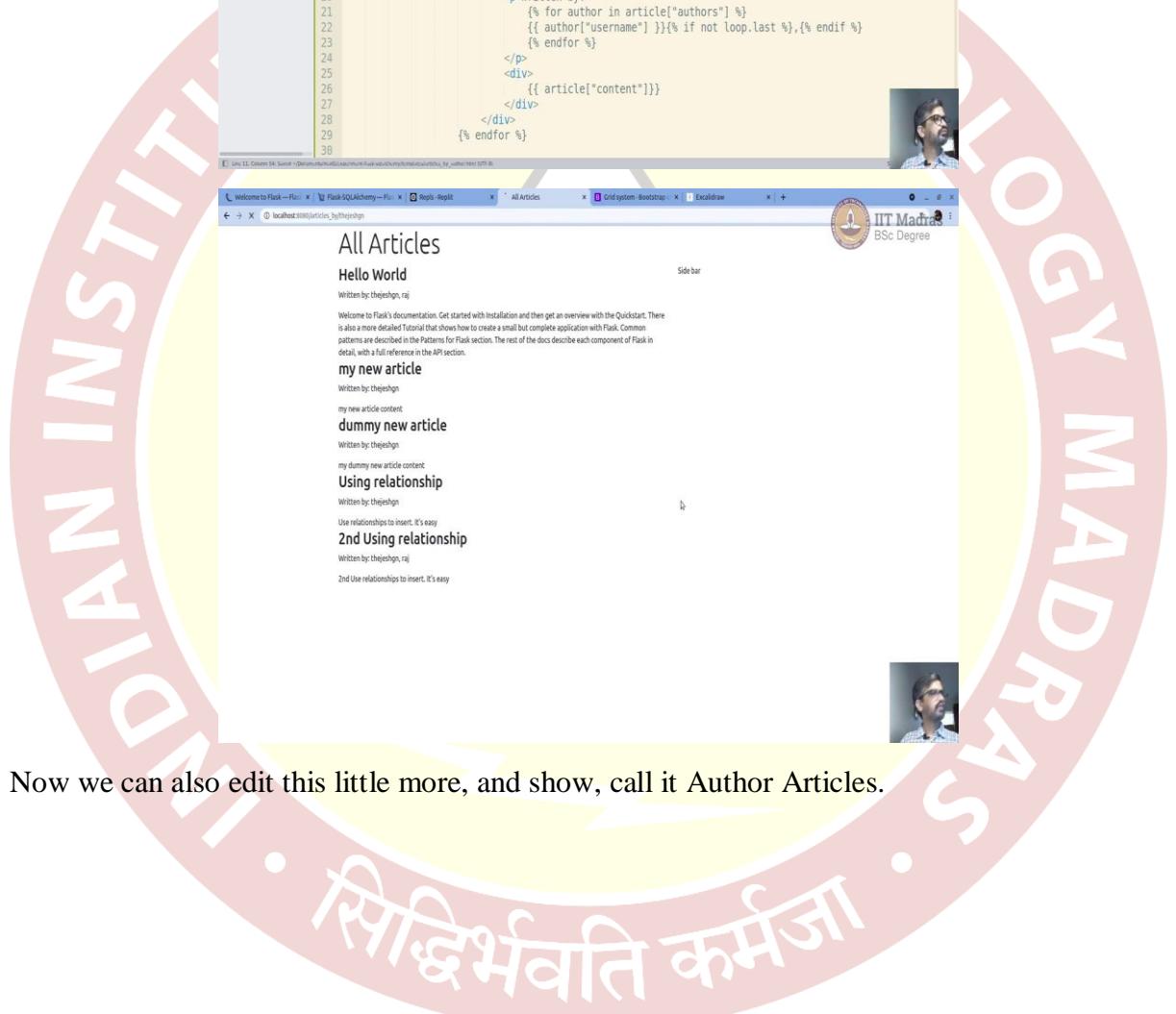
Use relationships to insert. It's easy

2nd Using relationship  
Written by: thejeshgn, raj

2nd User relationships to insert. It's easy

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)



The image shows two screenshots of a computer interface. The top screenshot is a Sublime Text editor window titled "articles\_by\_author.html". It displays an HTML file with code for rendering a list of articles. The code includes imports, a DOCTYPE declaration, and a template for displaying articles. The bottom screenshot is a web browser window titled "All Articles" showing the rendered HTML output. The browser has tabs for "Welcome to Flask", "Flask-SQLAlchemy", "Articles", "Grid system: Bootstrap", and "ExcelDraw". The page content lists several articles: "Hello World" (written by rhejeshn, raj), "my new article" (written by rhejeshn), "dummy new article" (written by rhejeshn), "my dummy new article content", "Using relationship" (written by rhejeshn), "Use relationships to insert. It's easy", "2nd Using relationship" (written by rhejeshn, raj), and "2nd Use relationships to insert. It's easy". Each article entry includes a thumbnail image of a person.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>All Articles</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7m95PdgYmZZMECAngseQB830fGTwi0iMjiHaeVhAn4FJkqJBhZMI3Ahlu" crossorigin="anonymous">
</head>
<body>
<div class="container">
<div class="jumbotron">
<h1 class="display-4">Author Articles</h1>
</div>
<div class="row">
<div class="col-md-8">
{% for article in articles %}
<div>
<h2>{{ article["title"] }}</h2>
<p>Written by:<br/>
    {{ author["username"] }}{{ loop.last }}<br/>
    {% endif %}<br/>
</p>
<div>
<{{ article["content"] }}</div>
</div>
{% endfor %}
</div>
</div>
</div>

```

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

(Refer Slide Time: 13:44)

All Articles

Hello World  
Written by: theejohn, 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 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: theejohn

my new article content.

dummy new article  
Written by: theejohn

my dummy new article content

Using relationship  
Written by: theejohn

Use relationships to insert. It's easy

2nd Using relationship  
Written by: theejohn, raj

2nd Use relationships to insert. It's easy

IIT Madras  
BSc Degree



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.

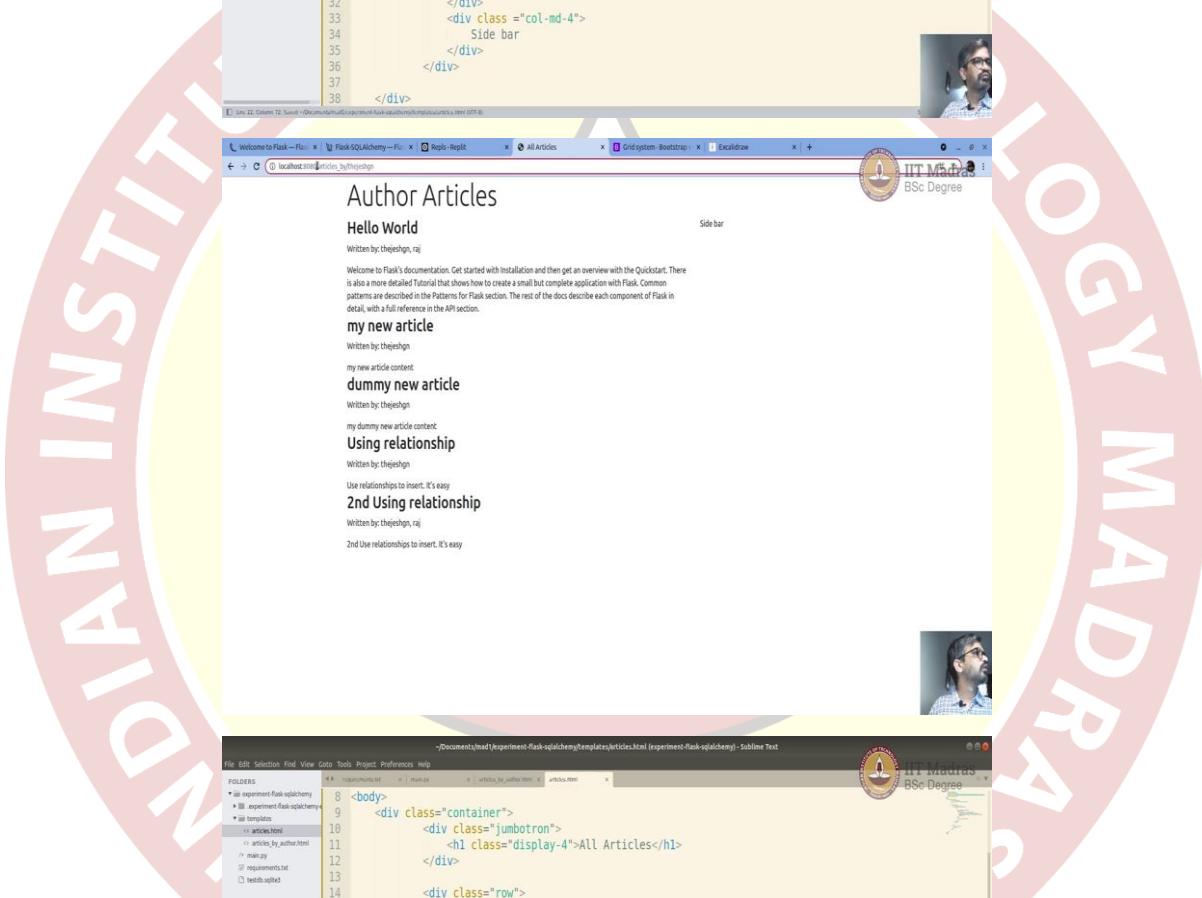
(Refer Slide Time: 13:54)

```
</body>
<div class="container">
    <div class="jumbotron">
        <h1 class="display-4">All Articles</h1>
    </div>

    <div class="row">
        <div class="col-md-8">
            {% for article in articles %}
                <div>
                    <h2>{{ article["title"] }}</h2>
                    <p>Written by:<br/>
                        {% for author in article["authors"] %}
                            {{ author["username"] }}{% if not loop.last %}, {% endif %}
                        {% endfor %}
                    </p>
                    <div>
                        {{ article["content"] }}
                    </div>
                {% endfor %}
            </div>
            <div class="col-md-4">
                Side bar
            </div>
        </div>
    </div>
</body>
```

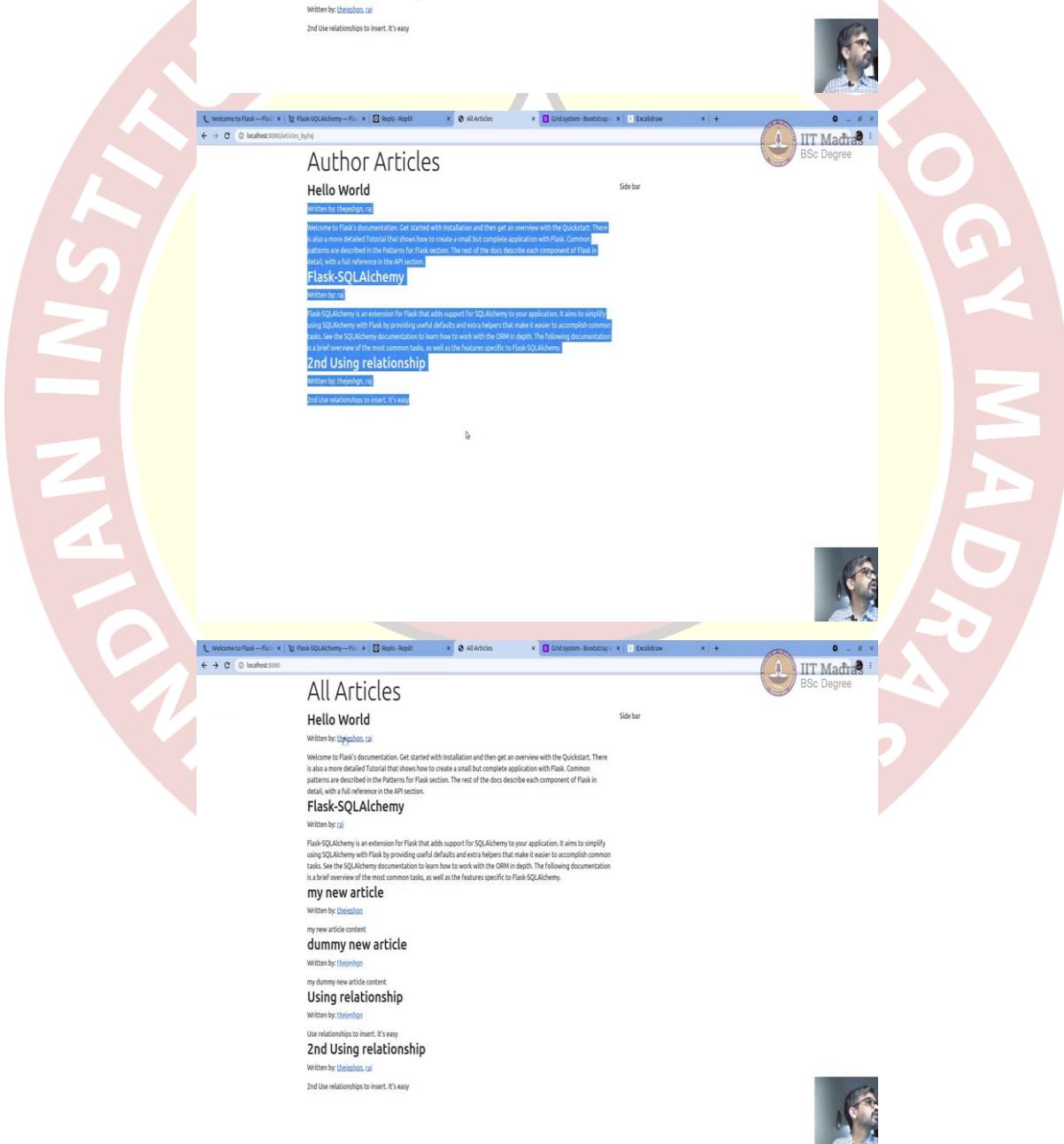
IIT Madras  
BSc Degree





```
-/Documents/Net/Flask-SQLAlchemy/templates/articles.html (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find Goto Tools Project Preferences Help
FOLDERS
  * all experiment Flask-SQLAlchemy
  * experiment Flask-SQLAlchemy
  * templates
    * articles.html
      * articles_by_author.html
      * main.py
      * requirements.txt
      * tests.sqlitedb
8 <body>
9   <div class="container">
10     <div class="jumbotron">
11       <h1 class="display-4">All Articles</h1>
12     </div>
13
14     <div class="row">
15       <div class="col-md-8">
16         {% for article in articles %}
17           <div>
18
19             <h2>{{ article["title"] }}</h2>
20             <p>Written by:<br>
21               {% for author in article["authors"] %}
22                 <a href="#">{{ author["username"] }}</a>{% if not loop.last %}, {%endif%}
23               {% endfor %}
24             </p>
25             <div>
26               {{ article["content"] }}
27             </div>
28           </div>
29         {% endfor %}
30
31       </div>
32       <div class="col-md-4">
33         Side bar
34       </div>
35     </div>
36
37   </div>
38
E Line 22, Column 72. Swind - /Documents/Net/Flask-SQLAlchemy/templates/articles.html (experiment-flask-sqlalchemy) - Sublime Text
```





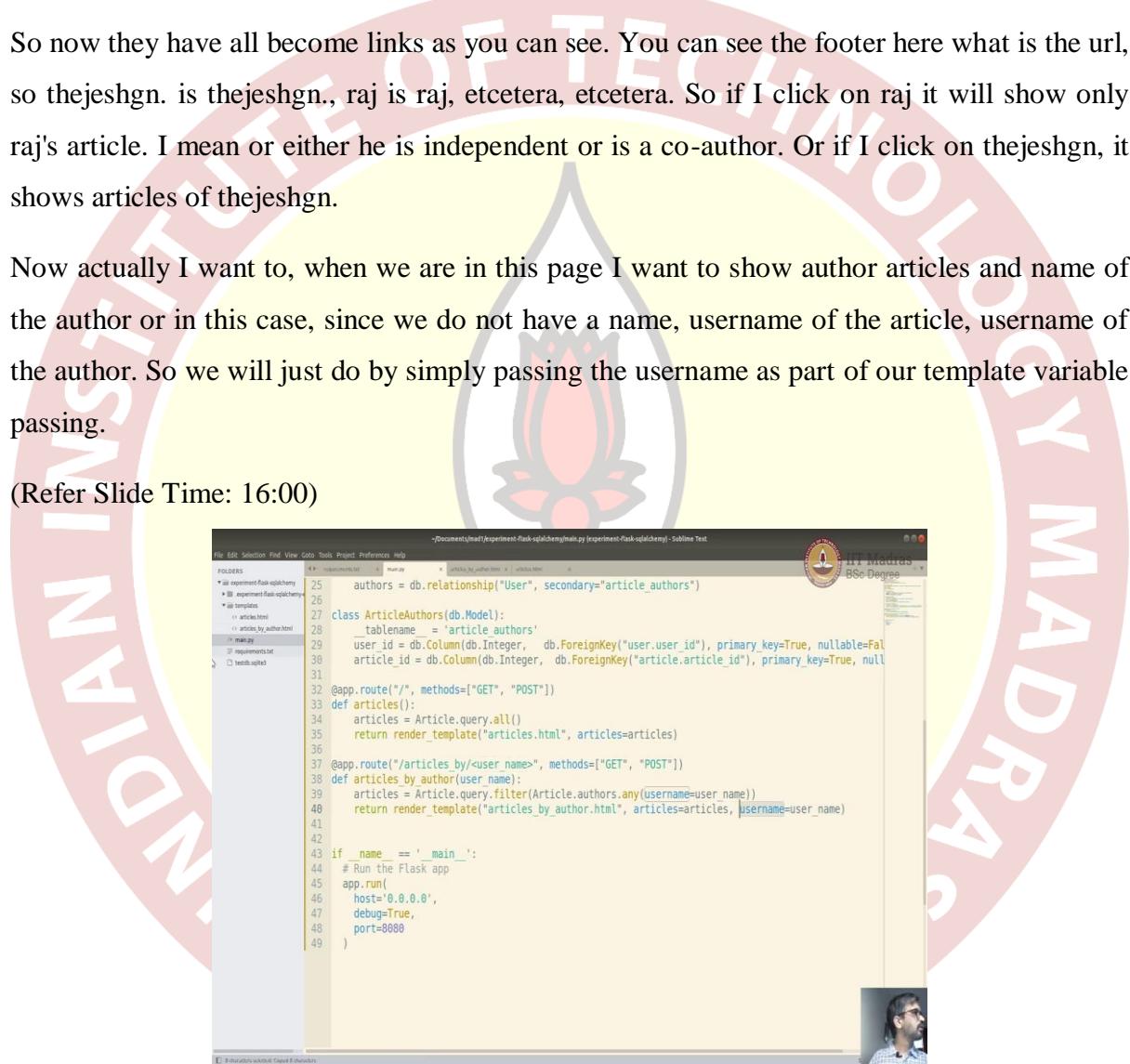
So that is done by going to this, making this art author name as clickable link. I am going to make it like a href. And then link will look like, for example, thejeshgn. So it will look like.

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)



```
->/documents/main/experiment/flask-sqlalchemy/main.py experiment-flask-sqlalchemy - Sublime Text
File Edit Selection Find View Go To Tools Project Preferences Help
FOLDERS
  • i experiment-flask-sqlalchemy
    • i experiment-flask-sqlalchemy
      • i __init__.py
        • i article.html
        • i articles_by_author.html
      • i main.py
    • requirements.txt
  • testdb.sqlite3

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=False)
30   article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, nullable=False)
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, user_name=user_name)
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   )
```

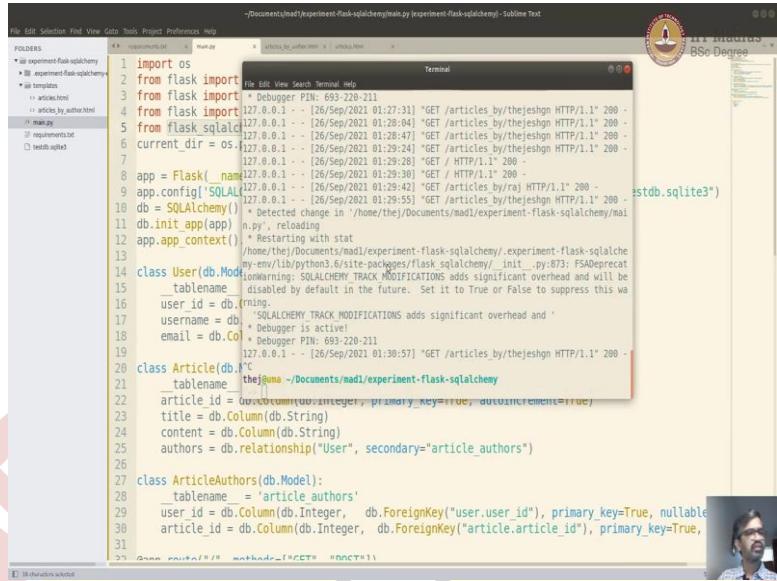




So `username=username`. And then we will use this `username` passed in our `articles_by_author.html`. Articles, Author Articles. That should work. There you go. Now you can also play around with sidebar and do many things. You can also fly here, and, or you can make it like a icon here, if it is clicks 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 repl.it.

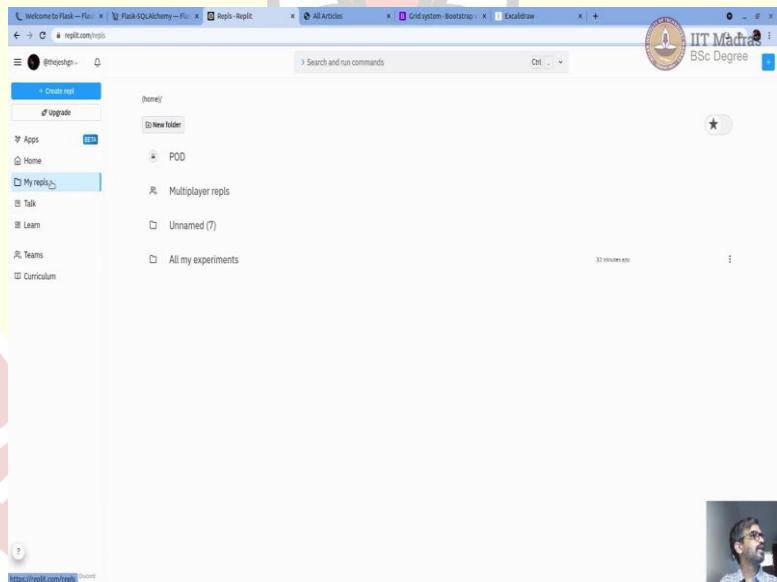
(Refer Slide Time: 17:15)

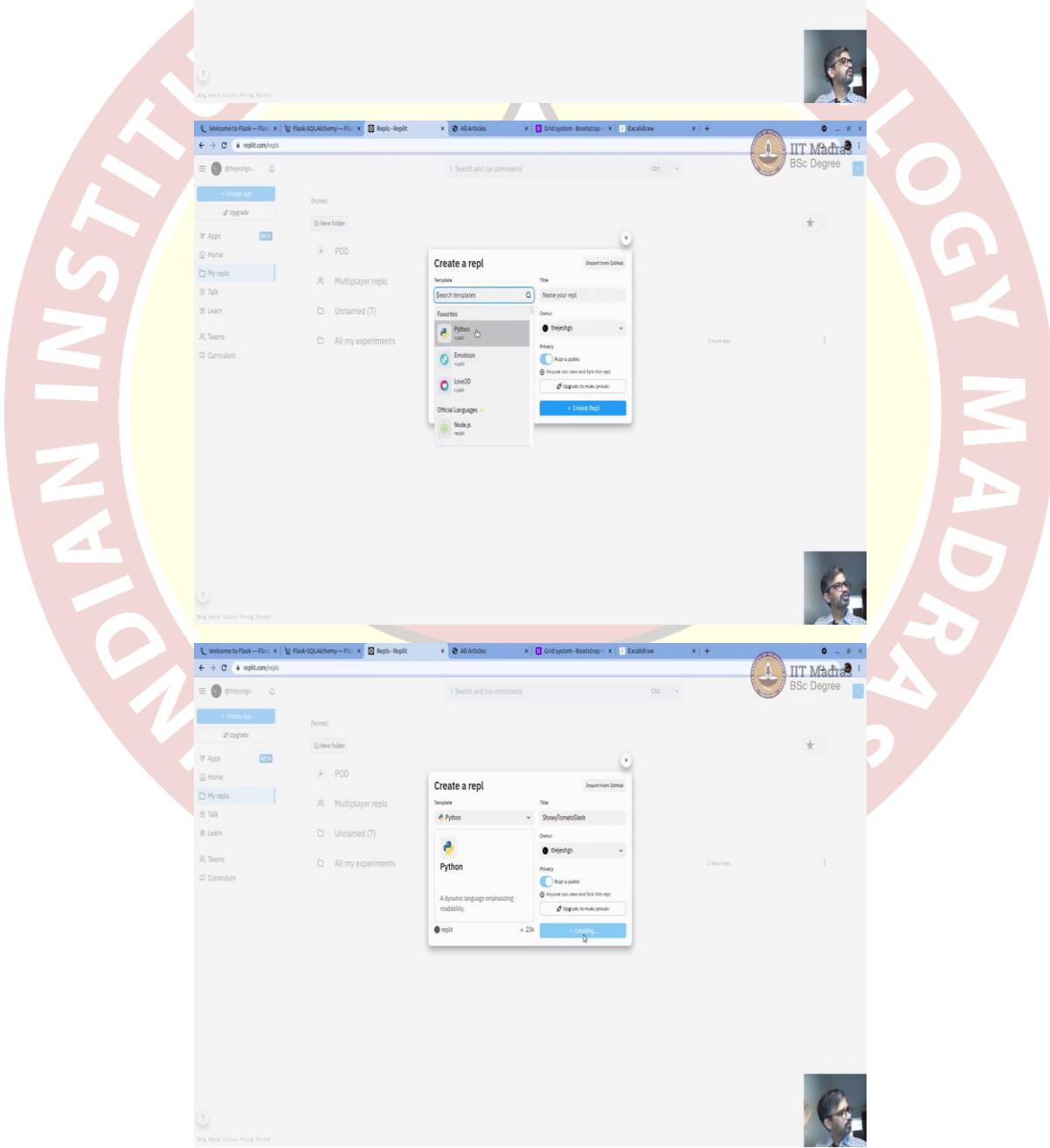


```
File Edit Selection Find View Go Tools Project Preferences Help
FOLDERS
    experiment-flask-sqlalchemy
        __init__.py
        articles.html
        articles_by_author.html
        main.py
        requirements.txt
        testdb.sqlite3
main.py
1 import os
2 from flask import *
3 from flask import *
4 from flask import *
5 from flask_sqlalchemy import *
6 current_dir = os.path.dirname(__file__)
7
8 app = Flask(__name__)
9 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///testdb.sqlite3'
10 db = SQLAlchemy()
11 db.init_app(app)
12 app.app_context().push()
13 class User(db.Model):
14     __tablename__ = 'users'
15     user_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
16     username = db.Column(db.String(80), unique=True)
17     email = db.Column(db.String(120), unique=True)
18     def __repr__(self):
19         return '' % self.username
20 class Article(db.Model):
21     __tablename__ = 'articles'
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=False)
30     article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, nullable=False)
31
32 app.run(debug=True, port=5000)
33
```

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.

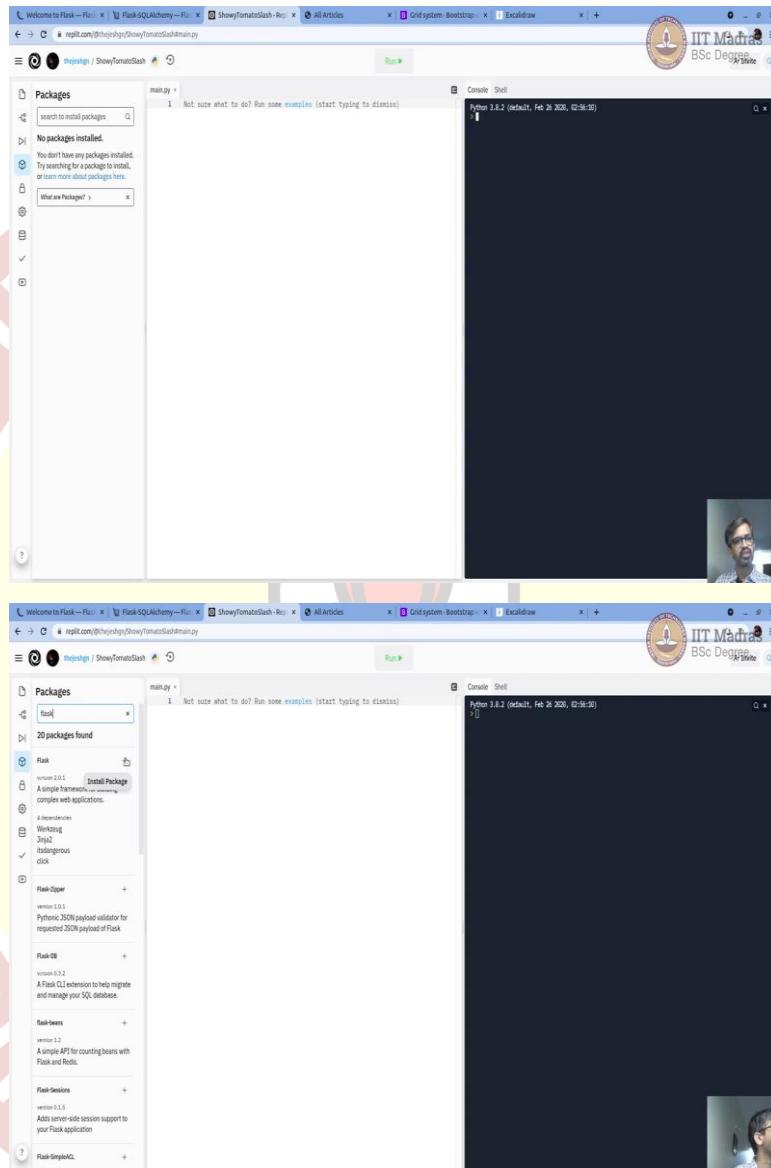
(Refer Slide Time: 17:22)

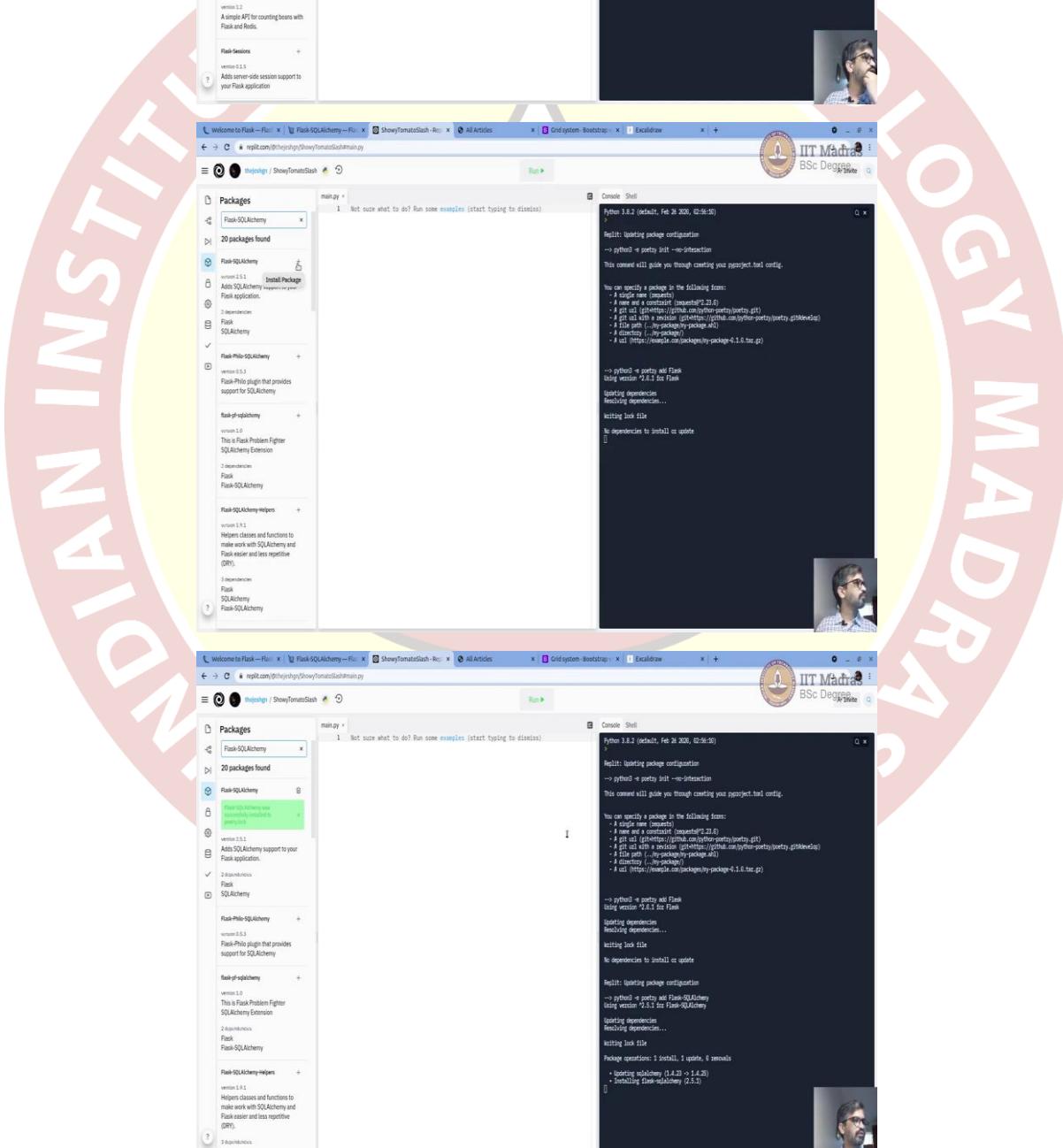




So once you log into Replit, go to My repls, create a new repl here using create repl. And choose python. And give some name. I am just going to go with the default name. And create repl.

(Refer Slide Time: 17:48)





```

IIT Madras
BSc Dev
Private

welcome to Flask — File 1 Flask-SQLAlchemy — File 2 ShowyTomatoDash — Run All Articles Gridsystem - Bootstrap Exceldraw
← → ⌂ repl.com/thejeshys/ShowyTomatoDash/main.py
thejeshys / ShowyTomatoDash
Run ▶

Files
  main.py
  Add folder
  templates
  Poetry files
    poetry.lock
    pyproject.toml

Packager files
  poetry.lock
  pyproject.toml

main.py
1 Not sure what to do? Run some examples (start typing to dismiss)

Console Shell
Python 3.8.2 (default, Feb 26 2020, 02:06:30)
[PyPy 5.11.0+poetry]
→ poetry config
→ poetry init -n interaction
This command will guide you through creating your pyproject.toml config.

You can specify a package in the following forms:
- A single name (depends)
- A git url & revision (git+https://github.com/your-poetry/poetry@v1.0.0)
- A git url with a revision (git@github.com:your-poetry/poetry.git@v1.0.0)
- A git url (git@github.com:your-poetry/poetry.git)
- A directory (./my-project)
- A url (http://example.com/packages/by-package-4.0.0.tart.gz)

→ poetry add Flask
Using version "2.1.1" for Flask
Upgrading dependencies...
Resolving dependencies...
Writing lock file
No dependencies to install or update

Poetry: Updating package configuration
→ poetry add Flask-SQLAlchemy
Using version "2.1.1" for Flask-SQLAlchemy
Upgrading dependencies...
Resolving dependencies...
Writing lock file
Package operations: 1 install, 1 update, 0 remove
  - Upgrading 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)

The image shows three screenshots of a computer interface, likely a virtual machine or a terminal session, demonstrating the setup of a Flask application. The top screenshot shows a Sublime Text editor with an HTML file named 'articles.html'. The code in the file includes Jinja2 templating, such as loops and conditionals, to render articles. The middle and bottom screenshots show a web browser window for 'Flask-SQLAlchemy' with the URL 'http://127.0.0.1:5000>ShowyTomatoDash'. The browser displays a simple 'Not sure what to do? Run some examples (start typing to dismiss)' message. To the right of the browser window is a terminal window titled 'Console Shell' showing the output of running 'python3 -m flask run'. The terminal output includes the Flask startup message, the result of running 'pipenv shell', and the command to run 'python3 app.py'. The terminal also shows the user navigating through a 'pyproject.toml' file and running 'poetry install'.

```
<body>
  <div class="container">
    <div class="jumbotron">
      <h1 class="display-4">All Articles</h1>
    </div>

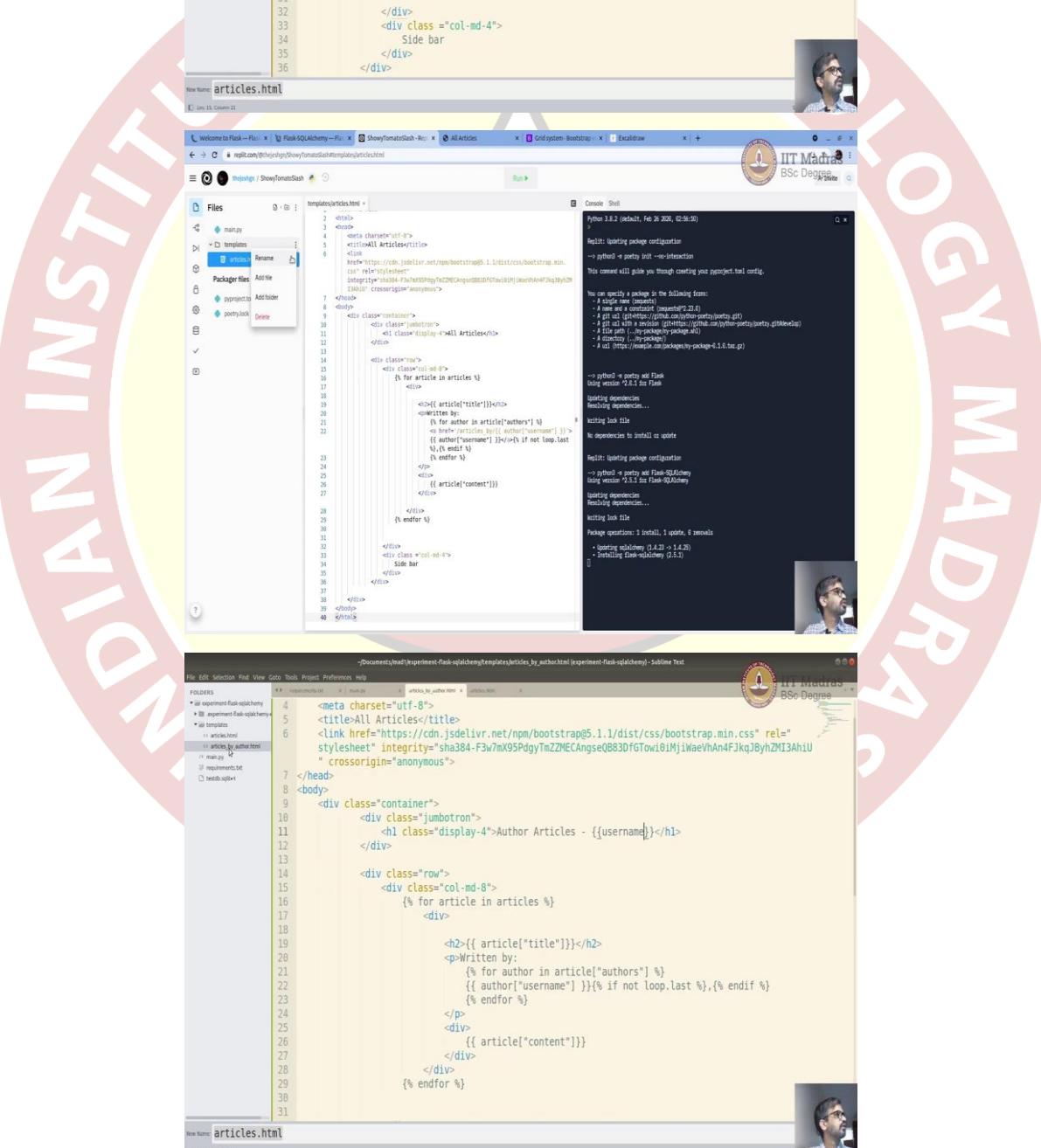
    <div class="row">
      <div class="col-md-8">
        {% for article in articles %}
          <div>
            <h2>{{ article["title"] }}</h2>
            <p>Written by:<br>
              {% for author in article["authors"] %}<br>
                <a href="/articles_by/{{ author["username"] }}"/>{{ author["username"] }}</a>{% if not loop.last %},&nbsp;{% endif %}<br>
              {% endfor %}
            </p>
            <div>
              {{ article["content"] }}
            </div>
          </div>
        {% endfor %}
      </div>
      <div class="col-md-4">
        Side bar
      </div>
    </div>
  </div>
```

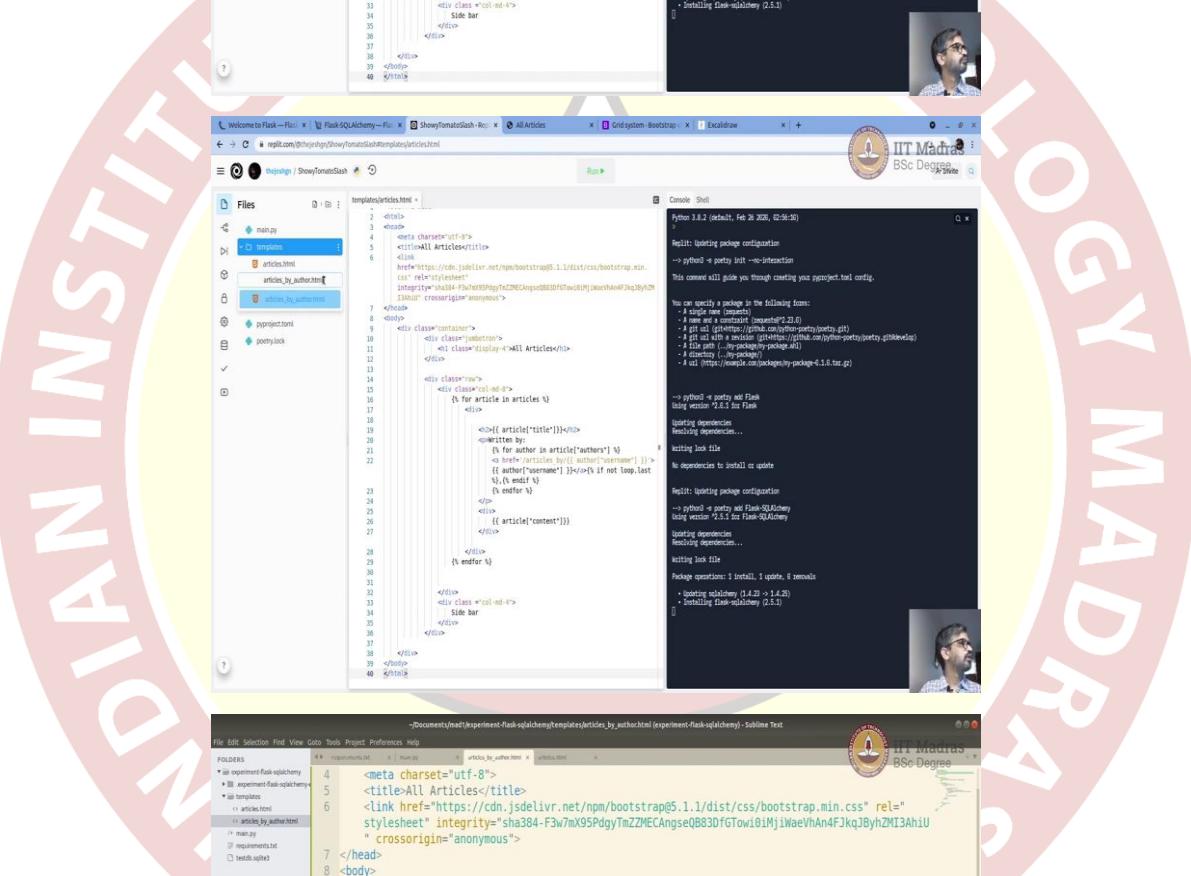
```
Welcome to Flask -- Flask-SQLAlchemy -- File -- ShowyTomatoDash -- Run -- All Articles -- Grid system: Bootstrap -- Exceldraw -- IIT Madras BSc Degree
```

```
File Edit Selection Find View Tools Project Preferences Help
```

```
File Edit Selection Find View Tools Project Preferences Help
```

```
File Edit Selection Find View Tools Project Preferences Help
```





```
-Documents\me\experiment-flask-sqlalchemy\templates\articles_by_author.html (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find Goto Tools Project Preferences Help
FOLDERS
• all experiment-flask-sqlalchemy
• experiment-flask-sqlalchemy
• templates
• articles.html
• articles_by_author.html
• main.py
• requirements.txt
• testutils3
4 <meta charset="utf-8">
5 <title>All Articles</title>
6 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-FwzYrHbtw宝玉ZPlQNw8vNQ6RQDgkUZG+XwWQFyHbJ4uZP+q79wZBwAk" crossorigin="anonymous">
7 </head>
8 <body>
9     <div class="container">
10        <div class="jumbotron">
11            <h1>{{ article["display"] }} All Articles - {{username}}</h1>
12        </div>
13        <div class="row">
14            <div class="col-md-8">
15                <div>
16                    {% for article in articles %}
17                        <div>
18                            <h2>{{ article["title"] }}</h2>
19                            <p>Written by:</p>
20                            {% for author in article["authors"] %}
21                                <p>{{ author["username"] }}</p>
22                            {% if not loop.last %}
23                                <br>
24                            {% endif %}
25                            {{ article["content"] }}
26                        </div>
27                    {% endfor %}
28                </div>
29            </div>
30            <div class="col-md-4">
31                <h3>Side Bar</h3>
32            </div>
33        </div>
34    </div>
35 </body>
36 </html>
```

```
welcome to Flask -- File -- ShowyTomasDash -- Run -- ShowyTomasDash -- All Articles -- Gridsystem - Bootstrap -- Exceldraw -- IIT Madras BSc Degree
File Edit Selection Find Goto Tools Project Preferences Help
Files
• main.py
• templates
• articles.html
• Packager files
• pyproject.toml
• poetry.lock
• Delete
1 templates\articles.html =
2 <meta charset="utf-8">
3 <title>All Articles</title>
4 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" integrity="sha384-FwzYrHbtw宝玉ZPlQNw8vNQ6RQDgkUZG+XwWQFyHbJ4uZP+q79wZBwAk" crossorigin="anonymous">
5 </head>
6 <body>
7     <div class="container">
8         <div class="jumbotron">
9             <h1>{{ article["display"] }} All Articles</h1>
10            </div>
11            <div class="row">
12                <div class="col-md-8">
13                    <div>
14                        {% for article in articles %}
15                            <div>
16                                <h2>{{ article["title"] }}</h2>
17                                <p>Written by:</p>
18                                {% for author in article["authors"] %}
19                                    <p>{{ author["username"] }}</p>
20                                {% if not loop.last %}
21                                    <br>
22                                {% endif %}
23                                {{ article["content"] }}
24                            </div>
25                        {% endfor %}
26                    </div>
27                </div>
28                <div class="col-md-4">
29                    <h3>Side Bar</h3>
30                </div>
31            </div>
32        </div>
33    </body>
34 </html>
```

```
Python 3.8.2 (default, Feb 28 2020, 02:56:30)
[Clang 10.0.0 (clang-1000.11.46.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
> python3 > poetry init -n interaction
This command will guide you through creating your pyproject.toml config.
You can specify a package in the following form:
  • A single name (empty)
  • A name & a constraint (e.g. mypackage@1.2.0)
  • A git url (git@github.com:myname/myrepo.git)
  • A file path (./my-project)
  • A directory (./my-project)
  • A directory (http://example.com/packages/my-package-4.1.6.tar.gz)

> python3 > poetry add Flask
Using version '2.1.5' for Flask
Upgrading dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update

Rebuilt: Updating package configuration
> python3 > poetry run flask-SQLAlchemy
Using version '2.5.1' for flask-SQLAlchemy
Upgrading dependencies
Resolving dependencies...
Writing lock file
Package operations: 1 install, 1 update, 0 remove
  • Upgrading sqlalchemy (1.4.23 -> 1.4.23)
  • Installing flask-SQLAlchemy (2.5.1)

IIT Madras BSc Degree
```



```
Welcome to Flask -- File -- ShowyTomasDash -- Run -- ShowyTomasDash -- All Articles -- Gridsystem - Bootstrap -- Exceldraw -- IIT Madras BSc Degree
File Edit Selection Find Goto Tools Project Preferences Help
Files
• main.py
• templates
• articles.html
• articles_by_author.html
• pyproject.toml
• poetry.lock
1 templates\articles.html =
2 <meta charset="utf-8">
3 <title>All Articles</title>
4 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" integrity="sha384-FwzYrHbtw宝玉ZPlQNw8vNQ6RQDgkUZG+XwWQFyHbJ4uZP+q79wZBwAk" crossorigin="anonymous">
5 </head>
6 <body>
7     <div class="container">
8         <div class="jumbotron">
9             <h1>{{ article["display"] }} All Articles - {{username}}</h1>
10            </div>
11            <div class="row">
12                <div class="col-md-8">
13                    <div>
14                        {% for article in articles %}
15                            <div>
16                                <h2>{{ article["title"] }}</h2>
17                                <p>Written by:</p>
18                                {% for author in article["authors"] %}
19                                    <p>{{ author["username"] }}</p>
20                                {% if not loop.last %}
21                                    <br>
22                                {% endif %}
23                                {{ article["content"] }}
24                            </div>
25                        {% endfor %}
26                    </div>
27                </div>
28                <div class="col-md-4">
29                    <h3>Side Bar</h3>
30                </div>
31            </div>
32        </div>
33    </body>
34 </html>
```

```
Python 3.8.2 (default, Feb 28 2020, 02:56:30)
[Clang 10.0.0 (clang-1000.11.46.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
> python3 > poetry init -n interaction
This command will guide you through creating your pyproject.toml config.
You can specify a package in the following form:
  • A single name (empty)
  • A name & a constraint (e.g. mypackage@1.2.0)
  • A git url (git@github.com:myname/myrepo.git)
  • A file path (./my-project)
  • A directory (./my-project)
  • A directory (http://example.com/packages/my-package-4.1.6.tar.gz)

> python3 > poetry add Flask
Using version '2.1.5' for Flask
Upgrading dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update

Rebuilt: Updating package configuration
> python3 > poetry run flask-SQLAlchemy
Using version '2.5.1' for flask-SQLAlchemy
Upgrading dependencies
Resolving dependencies...
Writing lock file
Package operations: 1 install, 1 update, 0 remove
  • Upgrading sqlalchemy (1.4.23 -> 1.4.23)
  • Installing flask-SQLAlchemy (2.5.1)

IIT Madras BSc Degree
```



```
-Documents\me\experiment-flask-sqlalchemy\templates\articles_by_author.html (experiment-flask-sqlalchemy) - Sublime Text
File Edit Selection Find Goto Tools Project Preferences Help
FOLDERS
• all experiment-flask-sqlalchemy
• experiment-flask-sqlalchemy
• templates
• articles.html
• articles_by_author.html
• main.py
• requirements.txt
• testutils3
4 <meta charset="utf-8">
5 <title>All Articles</title>
6 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-FwzYrHbtw宝玉ZPlQNw8vNQ6RQDgkUZG+XwWQFyHbJ4uZP+q79wZBwAk" crossorigin="anonymous">
7 </head>
8 <body>
9     <div class="container">
10        <div class="jumbotron">
11            <h1>{{ article["display"] }} Author Articles - {{username}}</h1>
12        </div>
13        <div class="row">
14            <div class="col-md-8">
15                <div>
16                    {% for article in articles %}
17                        <div>
18                            <h2>{{ article["title"] }}</h2>
19                            <p>Written by:</p>
20                            {% for author in article["authors"] %}
21                                <p>{{ author["username"] }}</p>
22                            {% if not loop.last %}
23                                <br>
24                            {% endif %}
25                                {{ article["content"] }}
26                            </div>
27                        {% endfor %}
28                    </div>
29                </div>
30            </div>
31        </div>
32    </div>
33 </body>
34 </html>
```



```

Flask -- File 1 Flask SQLAlchemy -- File 2 ShowyTomasDash - Run All Articles Gridsystem - Bootstrap 1 Exceldraw
welcome to Flask -- File 1
Flask SQLAlchemy -- File 2
ShowyTomasDash - Run
All Articles
Gridsystem - Bootstrap 1
Exceldraw
IIT Madras
BSc Dev
Project
thehttps://ShowyTomasDash
Run
Files
main.py
templates
articles.html
articles_by_author.html
Packager files
poetry.lock
poetry.toml
Console Shell
Python 3.8.2 (default, Feb 26 2020, 02:06:30)
[Pyo] Replit: Updating package configuration
-> poetry & 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 (repects)
  * A git url or a tag (repects)
  * A file path (repects)
  * A git url with a revision (git+https://github.com/pyfun-poetry/poetry@gh0st)
  * A directory (repects)
  * A direct URL (repects)
-> python3 & poetry init Flask
Using version 2.1.1 for Flask
Upgrading dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update
Replit: Updating package configuration
-> poetry & poetry add Flask-SQLAlchemy
Using version 2.5.5 for Flask-SQLAlchemy
Upgrading dependencies
Resolving dependencies...
Writing lock file
Package operations: 1 install, 1 update, 0 remove
  * Upgrading sqlalchemy (1.4.23 -> 1.4.23)
  * Installing flask-sqlalchemy (2.5.5)

```

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.

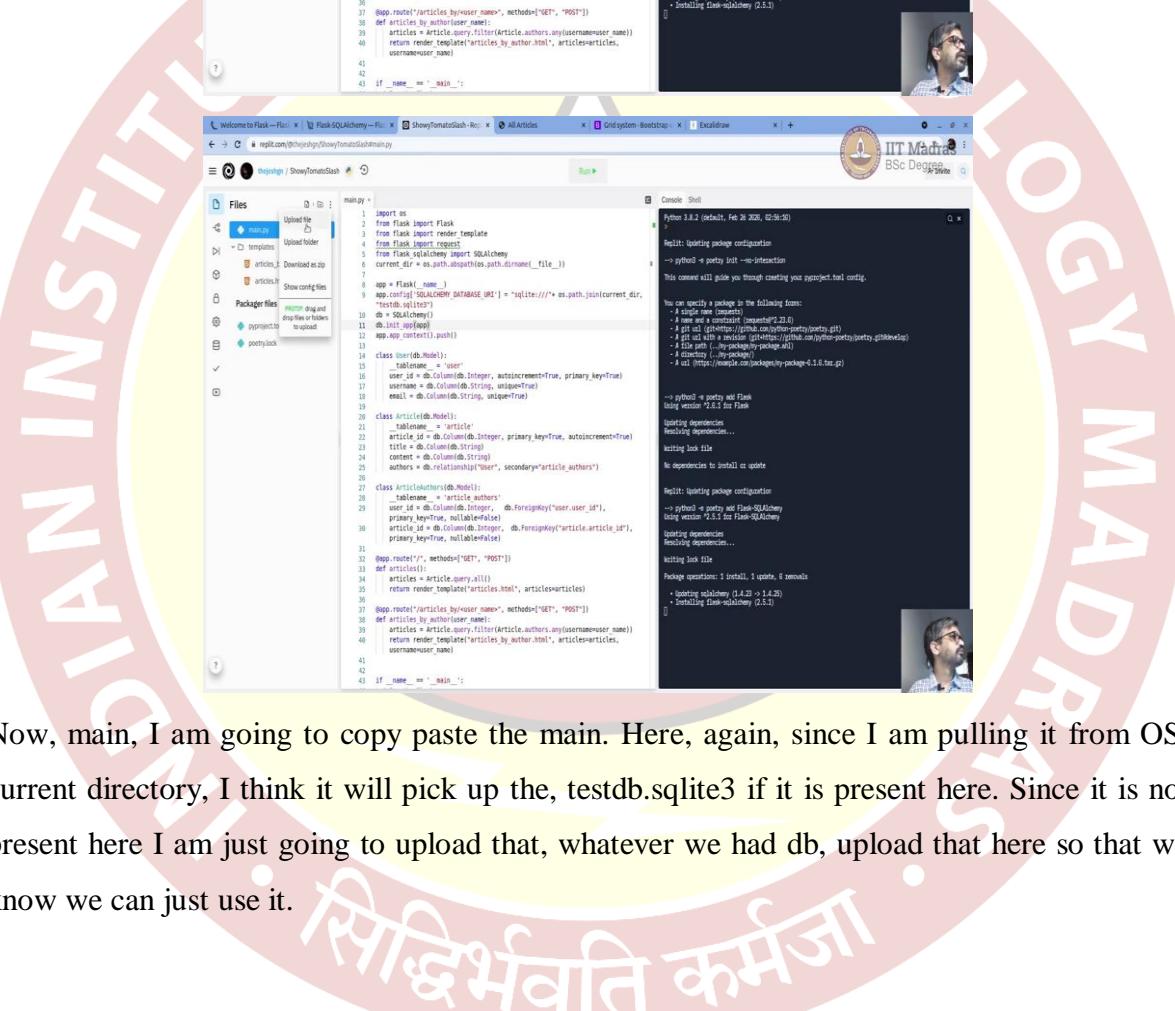
(Refer Slide Time: 20:36)

```

File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
all experiment Flask-SQLAlchemy
all templates
articles.html
articles_by_author.html
main.py
requirements.txt
tests.apptests
Sublime Text: /Documents/main/experiment/Flask-SQLAlchemy/main.py (experiment/Flask-SQLAlchemy)
IIT Madras
BSc Degree
1 import os
2 from flask import Flask
3 from flask import render_template
4 from flask import request
5 from flask_sqlalchemy import SQLAlchemy
6 current_dir = os.path.abspath(os.path.dirname(__file__))
7
8 app = Flask(__name__)
9 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///+' + os.path.join(current_dir, "testdb.sqlite3")
10 db = SQLAlchemy()
11 db.init_app(app)
12 app.app_context().push()
13
14 class User(db.Model):
15     __tablename__ = 'user'
16     user_id = db.Column(db.Integer, autoincrement=True, primary_key=True)
17     username = db.Column(db.String, unique=True)
18     email = db.Column(db.String, unique=True)
19
20 class Article(db.Model):
21     __tablename__ = 'article'
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=False)
30     article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"), primary_key=True, nullable=False)

New Name: articles_by_author.html
49 lines, 1891 characters selected

```



welcome to Flask — File — flask-SQLAlchemy — ShowyTomatoDash — Run — All Articles — Gridsystem - Bootstrap — Exceldraw — IIT Madras BSc Degree

**Files**

```

1 import os
2 from flask import Flask
3 from flask import render_template
4 from flask import request
5 from flask_sqlalchemy import SQLAlchemy
6 current_dir = os.path.abspath(os.path.dirname(__file__))
7
8 app = Flask(__name__)
9 app.config['SQLALCHEMY_DATABASE_URI'] = "sqlite:////* os.path.join(current_dir,
10 db = SQLAlchemy()
11 db.init_app(app)
12 app.app_context().push()
13
14 class User(db.Model):
15     __tablename__ = 'user'
16     id = db.Column(db.Integer, autoincrement=True, primary_key=True)
17     username = db.Column(db.String, unique=True)
18     email = db.Column(db.String, unique=True)
19
20 class Article(db.Model):
21     __tablename__ = 'article'
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.id"),
30                         primary_key=True, nullable=False)
31     article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"),
32                         primary_key=True, nullable=False)
33
34 @app.route('/', methods=['GET', 'POST'])
35 def articles():
36     articles = Article.query.all()
37     return render_template('articles.html', articles=articles)
38
39 @app.route('/articles/by/user_name', methods=['GET', 'POST'])
40 def articles_by_username():
41     articles = Article.query.filter(Article.authors.any(username=user_name))
42     return render_template('articles_by_author.html', articles=articles,
43     user_name=user_name)
44
45 if __name__ == '__main__':
46     app.run()

```

**Console - Shell**

```

Python 3.8.2 (default, Feb 28 2020, 02:56:30)
[Clang 10.0.0 (clang-1000.11.46.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
--> poetry run flask db init
Flask: Updating package configuration
--> poetry run flask db migrate
This command will guide you through creating your pyproject.toml config.
You can specify a package in the following forms:
  * A single name (depends)
  * A single name & constraint (depends["name"]>=2.2.0)
  * A file path (depends["path/to/package"])
  * A git url (depends["git+https://github.com/your-project/your-git-repo"])
  * A file url (depends["http://your-project.com/your-project"])
  * A directory (depends["my-package"])
  * A URL (http://example.com/packages/my-package-4.1.6.tar.gz)

--> poetry run flask db upgrade
Using version 2.1.5 for Flask-SQLAlchemy
Upgrading dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update

Flask: Updating package configuration
--> poetry run flask db migrate
Using version 2.1.5 for Flask-SQLAlchemy
Upgrading dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update

Flask: Updating package configuration
--> poetry run flask db upgrade
Using version 2.1.5 for Flask-SQLAlchemy
Upgrading dependencies
Resolving dependencies...
Writing lock file
Package operations: 1 install, 1 update, 0 remove
  * Upgrading sqlalchemy (1.4.20 -> 2.1.5)
  * Installing flask-sqlalchemy (2.5.3)

--> poetry run flask run
Using version 2.1.5 for Flask
Upgrading dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update

Flask: Updating package configuration
--> poetry run flask db migrate
Using version 2.1.5 for Flask-SQLAlchemy
Upgrading dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update

Flask: Updating package configuration
--> poetry run flask db upgrade
Using version 2.1.5 for Flask-SQLAlchemy
Upgrading dependencies
Resolving dependencies...
Writing lock file
Package operations: 1 install, 1 update, 0 remove
  * Upgrading sqlalchemy (1.4.20 -> 2.1.5)
  * Installing flask-sqlalchemy (2.5.3)

--> poetry run flask run
Using version 2.1.5 for Flask
Upgrading dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update

```

**Files**

```

1 import os
2 from flask import Flask
3 from flask import render_template
4 from flask import request
5 from flask_sqlalchemy import SQLAlchemy
6 current_dir = os.path.abspath(os.path.dirname(__file__))
7
8 app = Flask(__name__)
9 app.config['SQLALCHEMY_DATABASE_URI'] = "sqlite:////* os.path.join(current_dir,
10 db = SQLAlchemy()
11 db.init_app(app)
12 app.app_context().push()
13
14 class User(db.Model):
15     __tablename__ = 'user'
16     id = db.Column(db.Integer, autoincrement=True, primary_key=True)
17     username = db.Column(db.String, unique=True)
18     email = db.Column(db.String, unique=True)
19
20 class Article(db.Model):
21     __tablename__ = 'article'
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.id"),
30                         primary_key=True, nullable=False)
30     article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"),
31                         primary_key=True, nullable=False)
32
33 @app.route('/', methods=['GET', 'POST'])
34 def articles():
35     articles = Article.query.all()
36     return render_template('articles.html', articles=articles)
37
38 @app.route('/articles/by/user_name', methods=['GET', 'POST'])
39 def articles_by_username():
40     articles = Article.query.filter(Article.authors.any(username=user_name))
41     return render_template('articles_by_author.html', articles=articles,
42     user_name=user_name)
43
44 if __name__ == '__main__':
45     app.run()

```

**Console - Shell**

```

Python 3.8.2 (default, Feb 28 2020, 02:56:30)
[Clang 10.0.0 (clang-1000.11.46.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
--> poetry run flask db init
Flask: Updating package configuration
--> poetry run flask db migrate
This command will guide you through creating your pyproject.toml config.
You can specify a package in the following forms:
  * A single name (depends)
  * A single name & constraint (depends["name"]>=2.2.0)
  * A file path (depends["path/to/package"])
  * A git url (depends["git+https://github.com/your-project/your-git-repo"])
  * A file url (depends["http://your-project.com/your-project"])
  * A directory (depends["my-package"])
  * A URL (http://example.com/packages/my-package-4.1.6.tar.gz)

--> poetry run flask db upgrade
Using version 2.1.5 for Flask-SQLAlchemy
Upgrading dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update

Flask: Updating package configuration
--> poetry run flask db migrate
Using version 2.1.5 for Flask-SQLAlchemy
Upgrading dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update

Flask: Updating package configuration
--> poetry run flask db upgrade
Using version 2.1.5 for Flask-SQLAlchemy
Upgrading dependencies
Resolving dependencies...
Writing lock file
Package operations: 1 install, 1 update, 0 remove
  * Upgrading sqlalchemy (1.4.20 -> 2.1.5)
  * Installing flask-sqlalchemy (2.5.3)

--> poetry run flask run
Using version 2.1.5 for Flask
Upgrading dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update

Flask: Updating package configuration
--> poetry run flask db migrate
Using version 2.1.5 for Flask-SQLAlchemy
Upgrading dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update

Flask: Updating package configuration
--> poetry run flask db upgrade
Using version 2.1.5 for Flask-SQLAlchemy
Upgrading dependencies
Resolving dependencies...
Writing lock file
Package operations: 1 install, 1 update, 0 remove
  * Upgrading sqlalchemy (1.4.20 -> 2.1.5)
  * Installing flask-sqlalchemy (2.5.3)

--> poetry run flask run
Using version 2.1.5 for Flask
Upgrading dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update

```

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)

The image shows two screenshots of a computer interface, likely a terminal or code editor, demonstrating the creation of a Flask-SQLAlchemy application.

**Screenshot 1:** This screenshot shows a code editor with a file named `main.py`. The code imports Flask and SQLAlchemy, defines a database URL, and sets up the application. It includes a route for displaying articles by user name and a route for rendering the article template. A conditional check at the bottom runs the application if the script is executed directly.

```
1 import os
2 from flask import Flask
3
4 from flask import render_template
5
6 from flask_sqlalchemy import SQLAlchemy
7
8 current_dir = os.path.abspath(os.path.dirname(__file__))
9
10 app = Flask(__name__)
11
12 app.config['SQLALCHEMY_DATABASE_URI'] = "sqlite:///{}".format(current_dir + "/testdb.sqlite")
13 db = SQLAlchemy(app)
14
15 app.app_context().push()
16
17 class User(db.Model):
18     __tablename__ = 'user'
19     id = db.Column(db.Integer, autoincrement=True, primary_key=True)
20     username = db.Column(db.String, unique=True)
21     email = db.Column(db.String, unique=True)
22
23 class Article(db.Model):
24     __tablename__ = 'article'
25     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
26     title = db.Column(db.String)
27     content = db.Column(db.String)
28     author_id = db.Column(db.Integer, db.ForeignKey("user.id"), primary_key=True, nullable=False)
29     author = db.relationship("User", backref="articles")
30
31 def articles_by_author_name(username):
32     return Article.query.filter(Article.authors.any(username=username))
33
34 @app.route('/articles_by_user_name', methods=['GET', 'POST'])
35 def articles_by_username():
36     articles = Article.query.filter(Article.authors.any(username=username))
37     return render_template('articles_by_author.html', articles=articles, username=username)
38
39 if __name__ == '__main__':
40     app.run()
41
42 # Run the Flask app
43 app.run()
```

**Screenshot 2:** This screenshot shows a terminal window running a command to update package configurations. The command is `python3 -m poetry init --no-interaction`. The terminal also displays help text for specifying a package.json.

```
Python 3.8.2 (default, Feb 26 2020, 02:56:10)
> Rerun: 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.json in the following formats:
- A single name request(s)
- A name and a constraint (scope@^2.23.0)
- A git url (git+https://github.com/python-poetry/poetry.git)
- A tar ball with an revision (git+https://github.com/python-poetry/poetry.git@develop)
- A file path (.../my-package/my-package.whl)
- A zip file (.../my-package.zip)
- A url (https://example.com/packages/my-package-0.1.0.tar.gz)

--> python3 -m poetry add flask
Using version "0.8.0" for flask
Updating dependencies
Resolving dependencies...
Writing lock file
No dependencies to install or update

Rerun: Updating package configuration
--> python3 -m poetry add Flask-SQLAlchemy
Using version "0.5.3" 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 (0.5.3)
> [ ]
```

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)

```

main.py
6 current_app = os.path.abspath(os.path.dirname(__file__))
7
8 app = Flask(__name__)
9 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///{}'.format(os.path.join(current_dir, 'testdb.sqlite'))
10 db = SQLAlchemy()
11 db.init_app(app)
12 app.app_context().push()
13
14 class User(db.Model):
15     __tablename__ = 'user'
16     id = db.Column(db.Integer, autoincrement=True, primary_key=True)
17     username = db.Column(db.String, unique=True)
18     email = db.Column(db.String, unique=True)
19
20 class Article(db.Model):
21     __tablename__ = 'article'
22     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 ArticleAuthor(db.Model):
28     __tablename__ = 'article_authors'
29     user_id = db.Column(db.Integer, db.ForeignKey('user.user_id'), primary_key=True, nullable=False)
30     article_id = db.Column(db.Integer, db.ForeignKey('article.article_id'), primary_key=True, nullable=False)
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_name():
39     articles = Article.query.filter(Article.authors.any(username=user_name))
40     return render_template('articles_by_author.html', articles=articles, username=user_name)
41
42
43 if __name__ == '__main__':
44     app.run(debug=True, host='0.0.0.0', port=8080)

```

```

main.py
4 from flask import request
5 from flask_sqlalchemy import SQLAlchemy
current_dir = os.path.abspath(os.path.dirname(__file__))
6
7 app = Flask(__name__)
8 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///{}'.format(os.path.join(current_dir, 'testdb.sqlite'))
9 db = SQLAlchemy()
10 db.init_app(app)
11 app.app_context().push()
12
13 class User(db.Model):
14     __tablename__ = 'user'
15     id = db.Column(db.Integer, autoincrement=True, primary_key=True)
16     username = db.Column(db.String, unique=True)
17     email = db.Column(db.String, unique=True)
18
19 class Article(db.Model):
20     __tablename__ = 'article'
21     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
22     title = db.Column(db.String)
23     content = db.Column(db.String)
24     authors = db.relationship('User', secondary='article_authors')
25
26 class ArticleAuthor(db.Model):
27     __tablename__ = 'article_authors'
28     user_id = db.Column(db.Integer, db.ForeignKey('user.user_id'), primary_key=True, nullable=False)
29     article_id = db.Column(db.Integer, db.ForeignKey('article.article_id'), primary_key=True, nullable=False)
30
31 @app.route('/', methods=['GET', 'POST'])
32 def articles():
33     articles = Article.query.all()
34     return render_template('articles.html', articles=articles)
35
36 @app.route('/articles/by-user-name', methods=['GET', 'POST'])
37 def articles_by_author_name():
38     articles = Article.query.filter(Article.authors.any(username=user_name))
39     return render_template('articles_by_author.html', articles=articles, username=user_name)
40
41
42 if __name__ == '__main__':
43     app.run(debug=True, host='0.0.0.0', port=8080)

```

```

main.py
9 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///{}'.format(os.path.join(current_dir, 'testdb.sqlite'))
10 db = SQLAlchemy()
11 db.init_app(app)
12 app.app_context().push()
13
14 class User(db.Model):
15     __tablename__ = 'user'
16     id = db.Column(db.Integer, autoincrement=True, primary_key=True)
17     username = db.Column(db.String, unique=True)
18     email = db.Column(db.String, unique=True)
19
20 class Article(db.Model):
21     __tablename__ = 'article'
22     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 ArticleAuthor(db.Model):
28     __tablename__ = 'article_authors'
29     user_id = db.Column(db.Integer, db.ForeignKey('user.user_id'), primary_key=True, nullable=False)
30     article_id = db.Column(db.Integer, db.ForeignKey('article.article_id'), primary_key=True, nullable=False)
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_name():
39     articles = Article.query.filter(Article.authors.any(username=user_name))
40     return render_template('articles_by_author.html', articles=articles, username=user_name)
41
42
43 if __name__ == '__main__':
44     app.run(debug=True, host='0.0.0.0', port=8080)

```

The image shows two screenshots of a development environment. The top screenshot shows a terminal window with Python code for a Flask-SQLAlchemy application. The code defines a database setup with a User model and an Article model. It includes routes for listing all articles and filtering articles by author. A warning message from the terminal indicates that 'ALCHEMY\_TRACK\_MODIFICATIONS' is set to True, which adds significant overhead and will be disabled by default in the future. The bottom screenshot shows a browser window displaying the 'All Articles' page, which lists several articles. The browser's status bar shows the URL as 'http://ShowToonDash.theoryg.com/repl'. The terminal window below the browser shows the command 'Ctrl+C' being entered.

```

current_dir = os.path.abspath(os.path.dirname(__file__))

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///{}'.format(
    os.path.join(current_dir, "testdb.sqlite3"))
db.init_app(app)
app.app_context().push()

class User(db.Model):
    __tablename__ = 'user'
    user_id = db.Column(db.Integer, primary_key=True,
                        autoincrement=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=False)
    article_id = db.Column(db.Integer, db.ForeignKey(
        "article.article_id"), primary_key=True, nullable=False)

@app.route('/', methods=['GET', 'POST'])
def articles():
    articles = Article.query.all()
    return render_template('articles.html',
                          articles=articles)

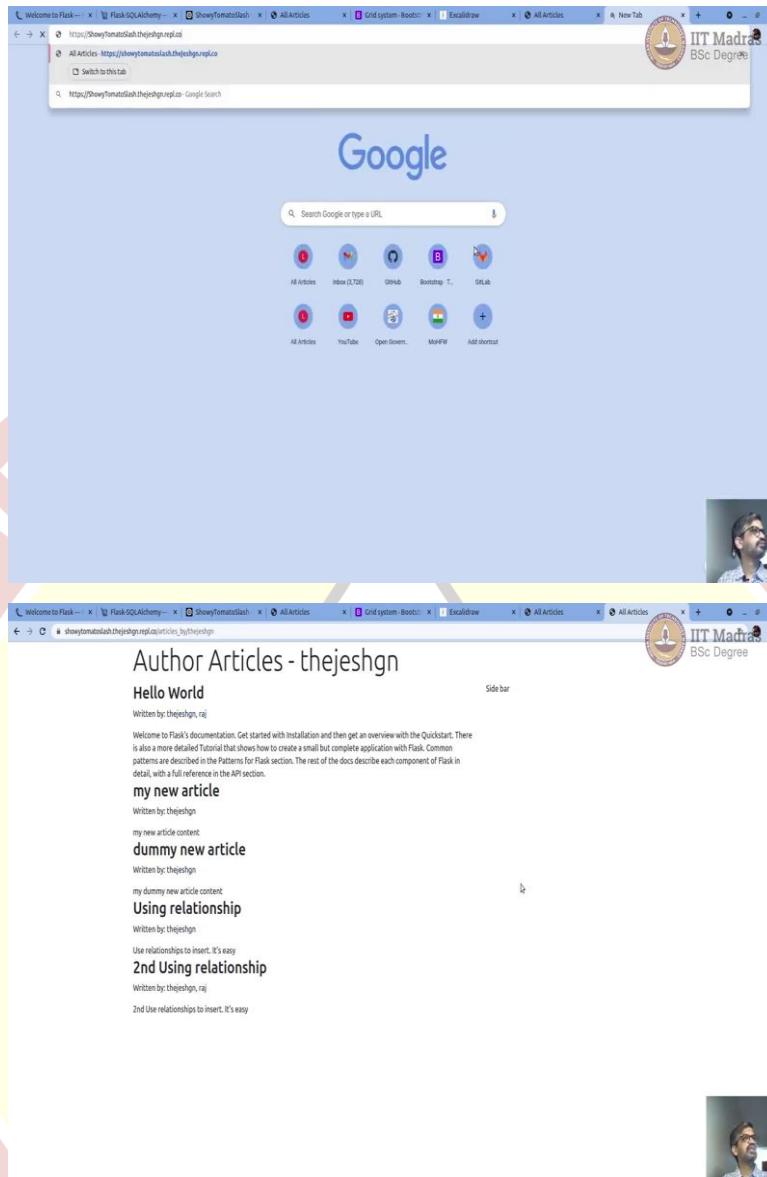
@app.route('/articles/by/', methods=['GET', 'POST'])
def articles_by_author(user_name):
    articles = Article.query.filter(Article.authors.any(
        username=user_name))
    return render_template('articles_by_author.html',
                          articles=articles,
                          username=user_name)

if __name__ == '__main__':
    db.create_all()
    app.run()

```

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)

**Flask-SQLAlchemy**

The screenshot shows three browser tabs for the Flask-SQLAlchemy application:

- File Structure:** Shows the project directory structure with files like main.py, models.py, and various templates.
- Code Editor:** Displays the main.py file containing the application setup and routes. It includes imports for Flask, SQLAlchemy, and various models. The code defines a database URI, creates a db object, and sets up the application context. It defines User and Article models with their respective attributes and relationships. Routes are defined for articles by user and articles by author.
- Terminal:** Shows the command line interface with the Flask application running. It displays log messages related to SQLAlchemy tracing and modification events. It also shows a screenshot of a video player showing a man speaking.

```

main.py
1 import os
2 from flask import Flask
3 from flask import render_template
4 from flask import request
5
6 if __name__ == '__main__':
7     app = Flask(__name__)
8     app.config['SQLALCHEMY_DATABASE_URI'] = "sqlite:///./+os.path.join(current_dir, 'testdb.sqlite')"
9     db = SQLAlchemy()
10    db.init_app(app)
11    db.app_context().push()
12
13    class User(db.Model):
14        __tablename__ = 'user'
15        user_id = db.Column(db.Integer, autoincrement=True, primary_key=True)
16        username = db.Column(db.String, unique=True)
17        email = db.Column(db.String, unique=True)
18
19    class Article(db.Model):
20        __tablename__ = 'article'
21        article_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
22        title = db.Column(db.String)
23        content = db.Column(db.String)
24        authors = db.relationship('User', secondary='article_authors')
25
26    class ArticleAuthors(db.Model):
27        __tablename__ = 'article_authors'
28        user_id = db.Column(db.Integer, db.ForeignKey("user.user_id"),
29                           primary_key=True, nullable=False)
30        article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"),
31                           primary_key=True, nullable=False)
32
33    @app.route('/', methods=['GET', 'POST'])
34    def articles_by_user():
35        articles = Article.query.filter(Article.authors.any(username=user_name))
36        return render_template('articles.html', articles=articles,
37                              username=username)
38
39    @app.route('/articles_by_user_name', methods=['GET', 'POST'])
40    def articles_by_author():
41        articles = Article.query.filter(Article.authors.any(username=user_name))
42        return render_template('articles_by_author.html', articles=articles,
43                              username=username)
44
45    if __name__ == '__main__':
46        app.run(
47            host='0.0.0.0',
48            debug=True,
49            port=8080)
50
51
52
All Articles
Hello World
Written by thejeshgn [ai]
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 thejeshgn [ai]
Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simplify using SQLAlchemy
Console Shell
http://virtualenv/python3.8/lib/python3.8/site-packages/flask_sqlalchemy/_init_.py:872: Q * deprecationwarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
  warnings.warn(DeprecationWarning)
* Serving Flask app "main" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
For a production WSGI server instead:
+ gunicorn: http://gunicorn.org/
+ uWSGI: http://uwsgi.io/
+ Flask-socketio: http://flask-socketio.readthedocs.io/en/latest/
+ Debug mode: on
+ Running on all addresses
+ Environment: production
+ Running on http://172.16.0.2:8080/ (Press CTRL-C to exit)
+ Browsing with std
http://virtualenv/python3.8/lib/python3.8/site-packages/flask_sqlalchemy/_init_.py:872: FSA * deprecationwarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
  warnings.warn(DeprecationWarning)
* Debugger is active!
+ Debugger FDN: 303-866-061
+ 127.0.0.1 - [26/Sep/2021:20:08:34] "GET / HTTP/1.1" 200 -
  127.0.0.1 - [26/Sep/2021:20:08:34] "GET / HTTP/1.1" 200 -
  127.0.0.1 - [26/Sep/2021:20:08:39] "GET / HTTP/1.1" 200 -
  127.0.0.1 - [26/Sep/2021:20:08:40] "GET /favicon.ico HTTP/1.1" 404 -


```

**All Articles**

The screenshot shows three browser tabs for the Flask-SQLAlchemy application:

- File Structure:** Shows the project directory structure with files like main.py, models.py, and various templates.
- Code Editor:** Displays the main.py file containing the application setup and routes. It includes imports for Flask, SQLAlchemy, and various models. The code defines a database URI, creates a db object, and sets up the application context. It defines User and Article models with their respective attributes and relationships. Routes are defined for articles by user and articles by author.
- Terminal:** Shows the command line interface with the Flask application running. It displays log messages related to SQLAlchemy tracing and modification events. It also shows a screenshot of a video player showing a man speaking.

```

main.py
1 import os
2 from flask import Flask
3 from flask import render_template
4 from flask import request
5
6 if __name__ == '__main__':
7     app = Flask(__name__)
8     app.config['SQLALCHEMY_DATABASE_URI'] = "sqlite:///./+os.path.join(current_dir, 'testdb.sqlite')"
9     db = SQLAlchemy()
10    db.init_app(app)
11    db.app_context().push()
12
13    class User(db.Model):
14        __tablename__ = 'user'
15        user_id = db.Column(db.Integer, autoincrement=True, primary_key=True)
16        username = db.Column(db.String, unique=True)
17        email = db.Column(db.String, unique=True)
18
19    class Article(db.Model):
20        __tablename__ = 'article'
21        article_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
22        title = db.Column(db.String)
23        content = db.Column(db.String)
24        authors = db.relationship('User', secondary='article_authors')
25
26    class ArticleAuthors(db.Model):
27        __tablename__ = 'article_authors'
28        user_id = db.Column(db.Integer, db.ForeignKey("user.user_id"),
29                           primary_key=True, nullable=False)
29        article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"),
30                           primary_key=True, nullable=False)
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_user_name():
39        articles = Article.query.filter(Article.authors.any(username=user_name))
40        return render_template('articles.html', articles=articles,
41                              username=username)
42
43    if __name__ == '__main__':
44        app.run(
45            host='0.0.0.0',
46            debug=True,
47            port=8080)
48
49
50
All Articles
Hello World
Written by thejeshgn [ai]
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 thejeshgn [ai]
Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simplify using SQLAlchemy
Console Shell
http://virtualenv/python3.8/lib/python3.8/site-packages/flask_sqlalchemy/_init_.py:872: Q * deprecationwarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
  warnings.warn(DeprecationWarning)
* Serving Flask app "main" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
For a production WSGI server instead:
+ gunicorn: http://gunicorn.org/
+ uWSGI: http://uwsgi.io/
+ Flask-socketio: http://flask-socketio.readthedocs.io/en/latest/
+ Debug mode: on
+ Running on all addresses
+ Environment: production
+ Running on http://172.16.0.2:8080/ (Press CTRL-C to exit)
+ Browsing with std
http://virtualenv/python3.8/lib/python3.8/site-packages/flask_sqlalchemy/_init_.py:872: FSA * deprecationwarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
  warnings.warn(DeprecationWarning)
* Debugger is active!
+ Debugger FDN: 303-866-061
+ 127.0.0.1 - [26/Sep/2021:20:08:34] "GET / HTTP/1.1" 200 -
  127.0.0.1 - [26/Sep/2021:20:08:34] "GET / HTTP/1.1" 200 -
  127.0.0.1 - [26/Sep/2021:20:08:39] "GET / HTTP/1.1" 200 -
  127.0.0.1 - [26/Sep/2021:20:08:40] "GET /favicon.ico HTTP/1.1" 404 -


```

**All Articles**

The screenshot shows three browser tabs for the Flask-SQLAlchemy application:

- File Structure:** Shows the project directory structure with files like main.py, models.py, and various templates.
- Code Editor:** Displays the main.py file containing the application setup and routes. It includes imports for Flask, SQLAlchemy, and various models. The code defines a database URI, creates a db object, and sets up the application context. It defines User and Article models with their respective attributes and relationships. Routes are defined for articles by user and articles by author.
- Terminal:** Shows the command line interface with the Flask application running. It displays log messages related to SQLAlchemy tracing and modification events. It also shows a screenshot of a video player showing a man speaking.

```

main.py
1 import os
2 from flask import Flask
3 from flask import render_template
4 from flask import request
5
6 if __name__ == '__main__':
7     app = Flask(__name__)
8     app.config['SQLALCHEMY_DATABASE_URI'] = "sqlite:///./+os.path.join(current_dir, 'testdb.sqlite')"
9     db = SQLAlchemy()
10    db.init_app(app)
11    db.app_context().push()
12
13    class User(db.Model):
14        __tablename__ = 'user'
15        user_id = db.Column(db.Integer, autoincrement=True, primary_key=True)
16        username = db.Column(db.String, unique=True)
17        email = db.Column(db.String, unique=True)
18
19    class Article(db.Model):
20        __tablename__ = 'article'
21        article_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
22        title = db.Column(db.String)
23        content = db.Column(db.String)
24        authors = db.relationship('User', secondary='article_authors')
25
26    class ArticleAuthors(db.Model):
27        __tablename__ = 'article_authors'
28        user_id = db.Column(db.Integer, db.ForeignKey("user.user_id"),
29                           primary_key=True, nullable=False)
29        article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"),
30                           primary_key=True, nullable=False)
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_user_name():
39        articles = Article.query.filter(Article.authors.any(username=user_name))
40        return render_template('articles.html', articles=articles,
41                              username=username)
42
43    if __name__ == '__main__':
44        app.run(
45            host='0.0.0.0',
46            debug=True,
47            port=8080)
48
49
50
All Articles
Hello World
Written by thejeshgn [ai]
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 thejeshgn [ai]
Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simplify using SQLAlchemy
Console Shell
http://virtualenv/python3.8/lib/python3.8/site-packages/flask_sqlalchemy/_init_.py:872: Q * deprecationwarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
  warnings.warn(DeprecationWarning)
* Serving Flask app "main" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
For a production WSGI server instead:
+ gunicorn: http://gunicorn.org/
+ uWSGI: http://uwsgi.io/
+ Flask-socketio: http://flask-socketio.readthedocs.io/en/latest/
+ Debug mode: on
+ Running on all addresses
+ Environment: production
+ Running on http://172.16.0.2:8080/ (Press CTRL-C to exit)
+ Browsing with std
http://virtualenv/python3.8/lib/python3.8/site-packages/flask_sqlalchemy/_init_.py:872: FSA * deprecationwarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
  warnings.warn(DeprecationWarning)
* Debugger is active!
+ Debugger FDN: 303-866-061
+ 127.0.0.1 - [26/Sep/2021:20:08:34] "GET / HTTP/1.1" 200 -
  127.0.0.1 - [26/Sep/2021:20:08:34] "GET / HTTP/1.1" 200 -
  127.0.0.1 - [26/Sep/2021:20:08:39] "GET / HTTP/1.1" 200 -
  127.0.0.1 - [26/Sep/2021:20:08:40] "GET /favicon.ico HTTP/1.1" 404 -


```

The image shows two screenshots of a computer screen displaying a development environment. On the left is a code editor showing a Python file named `main.py` which contains the code for a Flask application using SQLAlchemy. On the right is a web browser window showing the application's homepage titled "All Articles" with a single post "Hello World". Below the browser is a terminal window showing logs of HTTP requests from IP address 172.28.0.1.

```

main.py
1 import os
2 from flask import Flask
3 from flask import render_template
4 from flask import request
5 from flask_sqlalchemy import SQLAlchemy
6 current_dir = os.path.abspath(os.path.dirname(__file__))
7
8 app = Flask(__name__)
9 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:////* os.path.join(current_dir,
10                                     'testdb.sqlite3')')
11 db = SQLAlchemy()
12 app.init_app(db)
13 app.app_context().push()
14
15 class User(db.Model):
16     __tablename__ = 'user'
17     user_id = db.Column(db.Integer, autoincrement=True, primary_key=True)
18     username = db.Column(db.String, unique=True)
19     email = db.Column(db.String, unique=True)
20
21 class Article(db.Model):
22     __tablename__ = 'article'
23     article_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
24     title = db.Column(db.String)
25     content = db.Column(db.String)
26     author_id = db.Column(db.Integer, db.ForeignKey('user.article_id'),
27                          primary_key=False)
28
29 class ArticleAuthor(db.Model):
30     __tablename__ = 'article_authors'
31     user_id = db.Column(db.Integer, db.ForeignKey("user.user_id"),
32                        primary_key=True, nullable=False)
33     article_id = db.Column(db.Integer, db.ForeignKey("article.article_id"),
34                           primary_key=True, nullable=False)
35
36     def __init__(self, user_id, article_id):
37         self.user_id = user_id
38         self.article_id = article_id
39
40     def __repr__(self):
41         return f'{self.user_id} - {self.article_id}'
42
43 if __name__ == '__main__':
44     app.run()

```

All Articles

Hello World

Written by: thegeekz.ca

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. 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: thegeekz.ca

Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simplify using SQLAlchemy with Flask.

Console Shell

```

/opt/virtualenv/python3/lib/python3.8/site-packages/flask_sqlalchemy/_init_.py:872: Q * DeprecationWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be d
ebugged by default in the future. Set it to True or False to suppress this warning.
  * Serving Flask app "main" (lazy loading)
  * Environment: production
    WARNING: This is a development server. Do not use it in a production deployment.
    * Debug mode: on
      WARNING: This is a development server. Do not use it in a production deployment.
    * Running on http://127.0.0.1:8080/ (Press CTRL+C to exit)
    * Restarting with stat
    /opt/virtualenv/python3/lib/python3.8/site-packages/flask_sqlalchemy/_init_.py:872: FSA
* DeprecationWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be d
ebugged by default in the future. Set it to True or False to suppress this warning.
  * Debugger PIN: 201-466-863
172.28.0.1 - - [25/Sep/2023 20:08:34] "GET / HTTP/1.1" 200 -
172.28.0.1 - - [25/Sep/2023 20:08:34] "GET / HTTP/1.1" 200 -
172.28.0.1 - - [25/Sep/2023 20:08:37] "GET / HTTP/1.1" 200 -
172.28.0.1 - - [25/Sep/2023 20:08:40] "GET /favicon.ico HTTP/1.1" 404 -

```

All Articles

Hello World

Written by: thegeekz.ca

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. 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: thegeekz.ca

Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simplify using SQLAlchemy with Flask.

Console Shell

```

/opt/virtualenv/python3/lib/python3.8/site-packages/flask_sqlalchemy/_init_.py:872: Q * DeprecationWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be d
ebugged by default in the future. Set it to True or False to suppress this warning.
  * Serving Flask app "main" (lazy loading)
  * Environment: production
    WARNING: This is a development server. Do not use it in a production deployment.
    * Debug mode: on
      WARNING: This is a development server. Do not use it in a production deployment.
    * Running on http://127.0.0.1:8080/ (Press CTRL+C to exit)
    * Restarting with stat
    /opt/virtualenv/python3/lib/python3.8/site-packages/flask_sqlalchemy/_init_.py:872: FSA
* DeprecationWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be d
ebugged by default in the future. Set it to True or False to suppress this warning.
  * Debugger PIN: 201-466-863
172.28.0.1 - - [25/Sep/2023 20:08:34] "GET / HTTP/1.1" 200 -
172.28.0.1 - - [25/Sep/2023 20:08:34] "GET / HTTP/1.1" 200 -
172.28.0.1 - - [25/Sep/2023 20:08:37] "GET / HTTP/1.1" 200 -
172.28.0.1 - - [25/Sep/2023 20:08:40] "GET /favicon.ico HTTP/1.1" 404 -

```

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 repl.it using the same code without making any changes as such. Thank you. Bye.