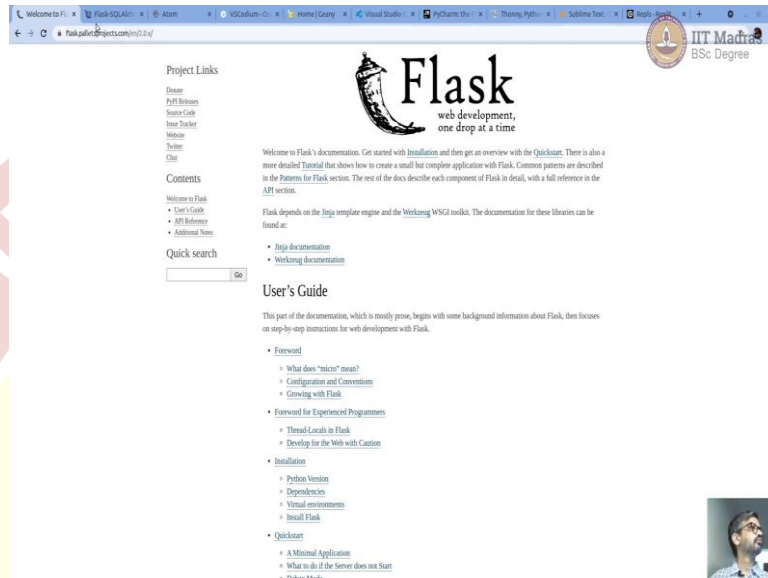


IIT Madras
ONLINE DEGREE

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

(Refer Slide Time: 00:15)



Welcome to the Modern Application Development screencast in this short screencast, we will learn how to use SQL alchemy inside a flask web app using an extension called flask SQL alchemy.

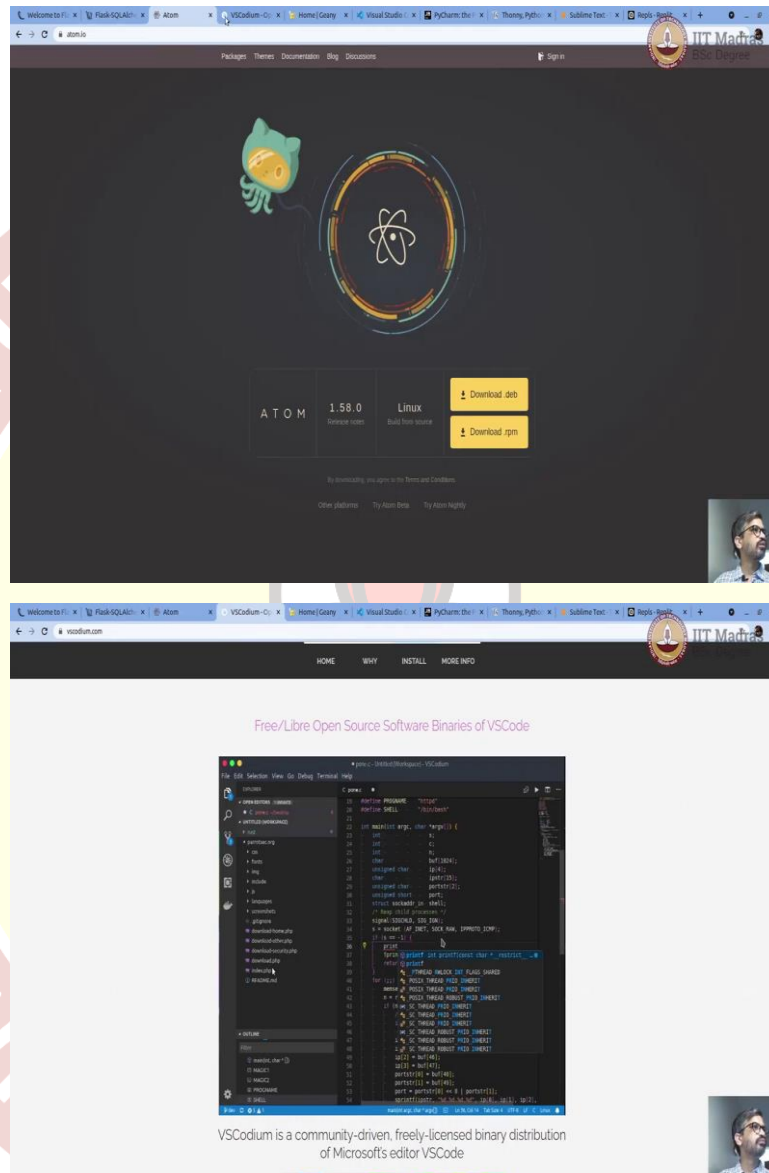
(Refer Slide Time: 00:31)



This one here. Before we start open the following applications on your Ubuntu desktop so that you can work along with me, a browser. Chrome is fine, or you can use Firefox. Text

editor. In this case, we are not going to use a text editor our prefer use some kind of IDE, which has, syntax highlighting some, help regarding coding project management and stuff like that.

(Refer Slide Time: 01:08)

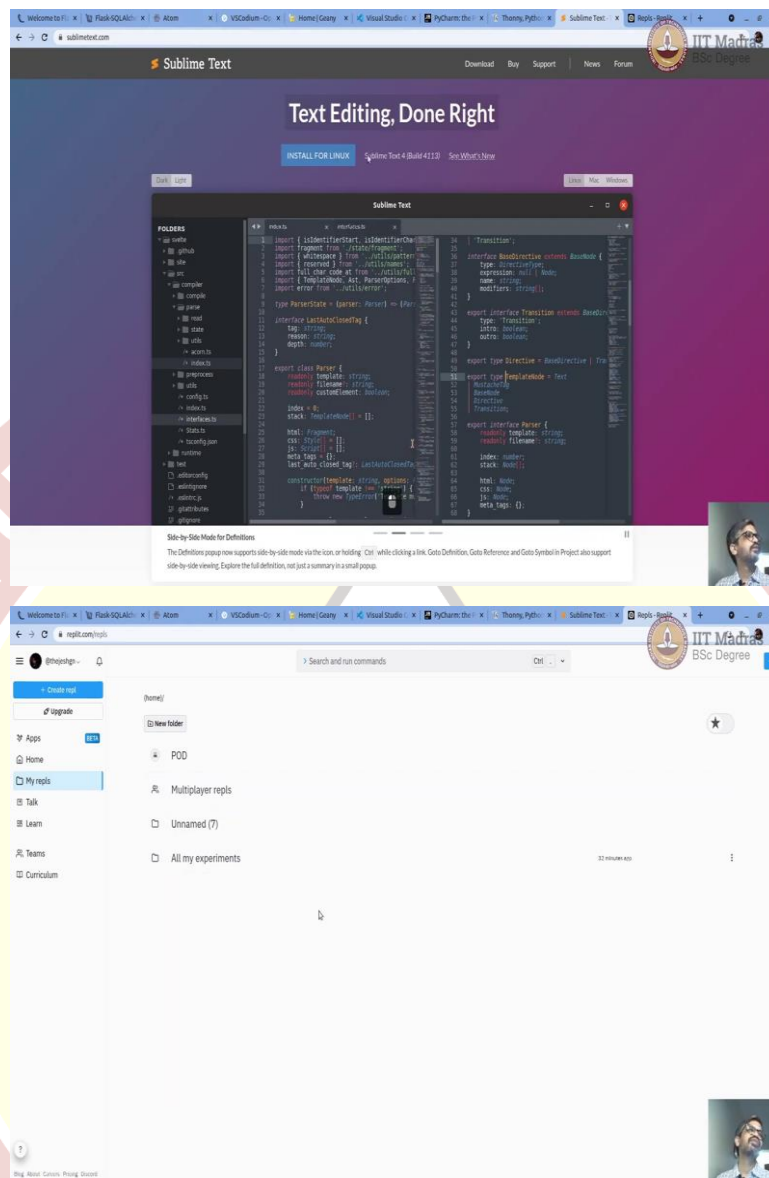




So, we could use one of the available open-source IDEs like Atom or VSCodium or Geany, Visual Studio, Pycharm is not open source but if you want there is a free Community Edition that you can use, or there is a professional version if you are ready to pay. There is Thonny again now open source, Sublime Text, it is not open source, it is a paid version.

And then, of course, if you do not want to install any of this, you can use an online editor or an IDE like Replit. We are going to use one of these. And then also we are going to repeat the same thing using Replit.

(Refer Slide Time: 01:54)



I am going to use something called Sublime Text, and then I am going to repeat quickly using ((1:57)) so you can all do it on ((1:59)) too. There is not much different, but I will just show it to you. Let I am just going to close this.

(Refer Slide Time: 02:13)

The image is a composite of three screenshots related to a Flask web development project, overlaid with a large, semi-transparent watermark of the Indian Institute of Technology Madras logo.

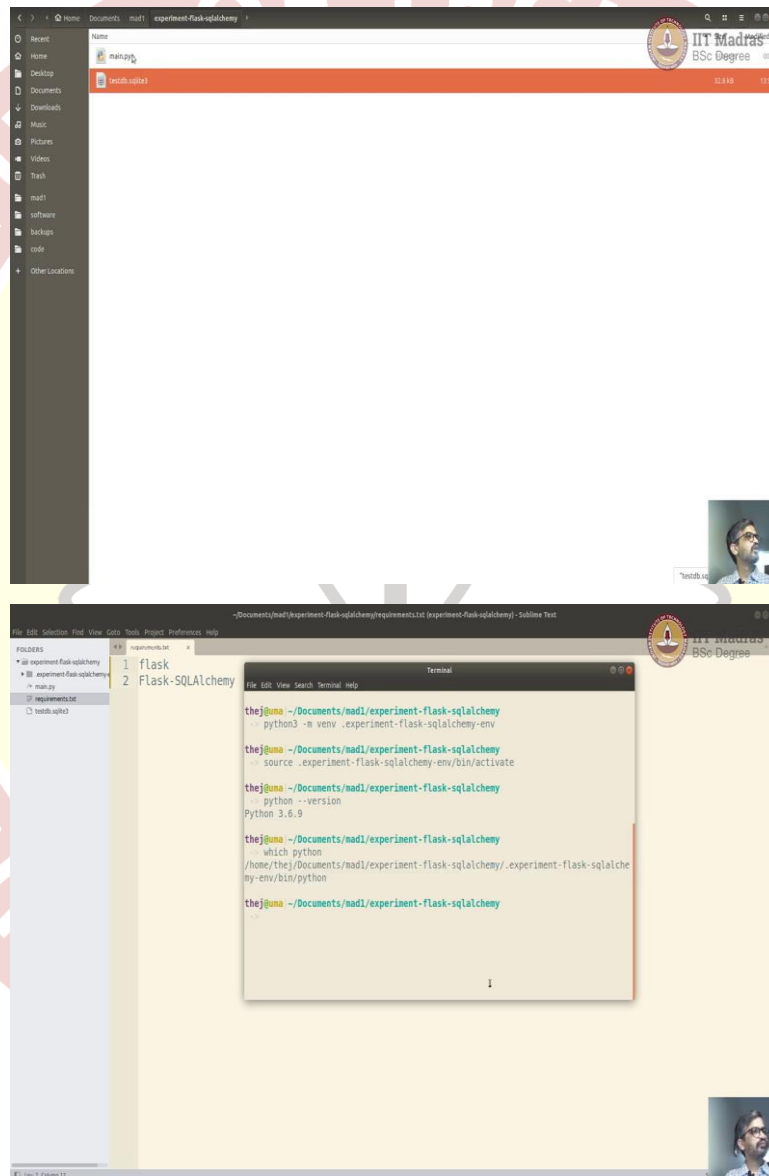
Top Screenshot: Flask Documentation
This is a screenshot of the Flask documentation website. The browser address bar shows "Welcome to Flask - Flask-SQLAlchemy - Flask-SQLAlchemy - Flask-SQLAlchemy". The page features the Flask logo and the tagline "web development, one drop at a time". The main content area is titled "User's Guide" and includes sections for "Project Links", "Contents", "Quick search", and "User's Guide". The "User's Guide" section lists various topics such as "Forward", "What does 'micro' mean?", "Configuration and Conventions", "Growing with Flask", "Forward for Experienced Programmers", "Themed Locals in Flask", "Developing for the Web with Caching", "Installation", "Python Version", "Dependencies", "Virtual environments", "Install Flask", "Quickstart", "A Minimal Application", "What to do if the Server does not Start", and "Debug Mode".

Middle Screenshot: File Explorer
This is a screenshot of a Windows File Explorer window. The address bar shows the path "Home > Documents > mad1 > experiment-flask-sqlalchemy". The left sidebar shows the "mad1" folder selected. The main pane displays the contents of the "experiment-flask-sqlalchemy" folder, which includes a file named "main.py" and a subfolder named "testdb.sqlite3".

Bottom Screenshot: Database Browser
This is a screenshot of a database browser application, specifically "DB Browser for SQLite". The address bar shows the path "DB Browser for SQLite - /home/hey/Documents/mad1/experiment-flask-sqlalchemy/testdb.sqlite3". The left sidebar shows the "Database Structure" tab selected, displaying a list of tables and their schemas. The main pane shows the "Edit Database Cell" dialog box, which is currently empty. The bottom right corner of the window shows a plot area with a grid and axes labeled "X", "Y1", and "Y2".

So, and then you need a terminal, internal terminal is fine. And then we just need to create a project. And we are creating a project called experiment-flask-sqlalchemy I have already created a project. And what I have done is I have copied the database, which we will use last time, in our SQL alchemy experiment, you can see that here, I opened it in dB browser for SQL, you can open the browser and see it is the same data, article article_authors and user, we are going to use the same database for this experiment too.

(Refer Slide Time: 03:02)



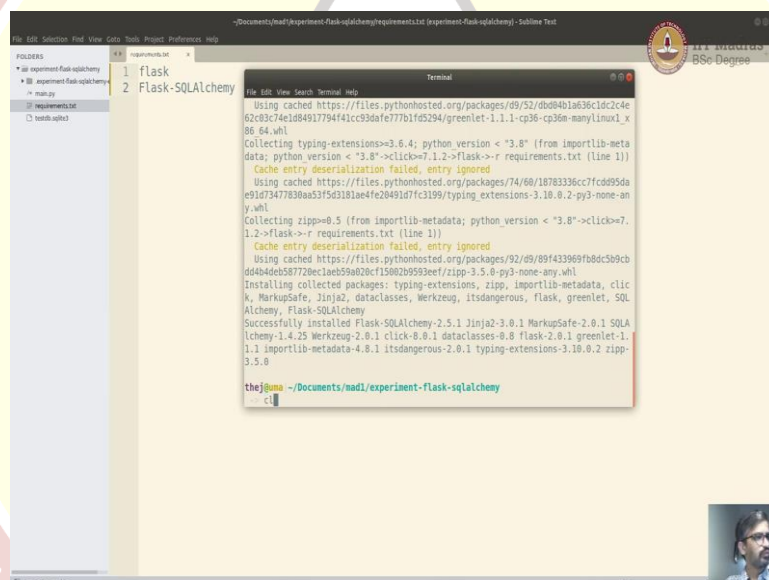
And then I have created a file called main.py. currently empty. I am just going to open that in my Sublime Text, you can see that it has another folder, and it has main.py and testdb.sqlite3. Like, as usual, we will start with creating a requirements.txt file .txt file, I am just

going to add only two requirements, flask and flask SQL alchemy. This is the extension of flask that we are going to use.

Now, we are going to go to command line and we will start with as usual creating the virtual environment. Now, for that, we have to run the command. Let me just paste that command for quicker access. Here, I am creating virtual environment called this is the same folder of this project. I am already in this folder. You can see I am already in that folder.

It will take a minute; you can see it getting created. Now we are going to enable it which is done by sourcing it. Sourcing dot experiment bin activate. Sourced, I can just test it python -- version Python3, which Python should show the path. Which is into the current folder, Virtual Environment. Now virtual environment is created, and it is enabled.

(Refer Slide Time: 05:01)



```
File Edit View Search Terminal Help
--Documents\mad1\experiment-flask-sqlalchemy\requirements.txt (experiment-flask-sqlalchemy) - Sublime Text

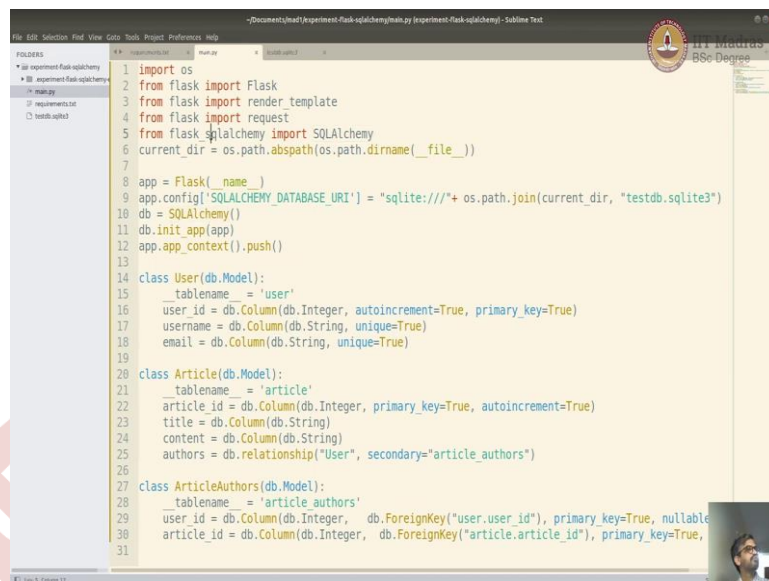
1 flask
2 Flask-SQLAlchemy

File Edit View Search Terminal Help
Terminal
Using cached https://files.pythonhosted.org/packages/d5/52/db04b1a6361dc2c4e62c83c74e1d84917794f41cc93d4fe777b1fd5294/greenlet-1.1.1-cp36-cp36m-manylinux1_x86_64.whl
Collecting typing-extensions>=3.6.4; python_version < "3.8" (from importlib-metadata; python_version < "3.8")>->click=7.1.2->flask->-r requirements.txt (line 1))
Cache entry deserialization failed, entry ignored
Using cached https://files.pythonhosted.org/packages/74/60/1878336cc7fcd895dae91d73477830a53f5d3181ae4fe20491d7fc3199/typing_extensions-3.10.0.2-py3-none-any.whl
Collecting zipp>=0.5 (from importlib-metadata; python_version < "3.8")>->click=7.1.2->flask->-r requirements.txt (line 1))
Cache entry deserialization failed, entry ignored
Using cached https://files.pythonhosted.org/packages/92/d9/89f433969fb8dc509cbdd404deb587720e1aeb59a028cf15002b9593eef/zipp-3.5.0-py3-none-any.whl
Installing collected packages: typing-extensions, zipp, importlib-metadata, click, MarkupSafe, Jinja2, dataclasses, Werkzeug, itsdangerous, flask, greenlet, SQLAlchemy, Flask-SQLAlchemy
Successfully installed Flask-SQLAlchemy-2.5.1 Jinja2-3.0.1 MarkupSafe-2.0.1 SQLAlchemy-1.4.25 Werkzeug-2.0.1 click-8.0.1 dataclasses-0.8 flask-2.0.1 greenlet-1.1.1 importlib-metadata-4.8.1 itsdangerous-2.0.1 typing-extensions-3.10.0.2 zipp-3.5.0

thej@ms ~$ pip install -r requirements.txt
thej@ms ~$ python --version
Python 3.10.0
```

Now, let us install everything that is there in the requirements.txt, there will be pip3 install - r requirements.txt. We will take a minute to install the things, it is installed.

(Refer Slide Time: 05:32)



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

Now, we will start with our main.py. I am just going to remove the default statement that I had written. Now we are going to go back and create our simple flask application. Now I am going to import, flask related modules. Flask related stuff. Now, we are going to use the same SQLite alchemy extension, so I am going to import that too. Now we are done with the import, I am going to create a flask app that we are done this before.

So, I am just doing it. And then there is one change that we need to do here, because we want to instantiate our db through SQL alchemy, which is what we had done before using the SQL alchemy. But here, we are going to do it through the extension a little more easier. That is been done by setting app.config['SQL_ALCHEMY_DATABASE_URI'], set this to :///, and name of this. Just going to copy paste.

But, I mean, I do not want it to be like this, I wanted to use it in the current folder. So, I am just going to create a current folder variable by reading, dynamically, what is the current folder, and then, I am going to use that. So, I am going to import OS. And get the current directory, this is how you get the current directory.

Now, I am just going to attach current directory. And path of SQLite3, just want to remove this extra plus. Now, this is setup, I am going to initialize the db and then, set it in the flask app, so here is setting up the db by calling this and then initializing the db app by passing a flask then pushing it to context.

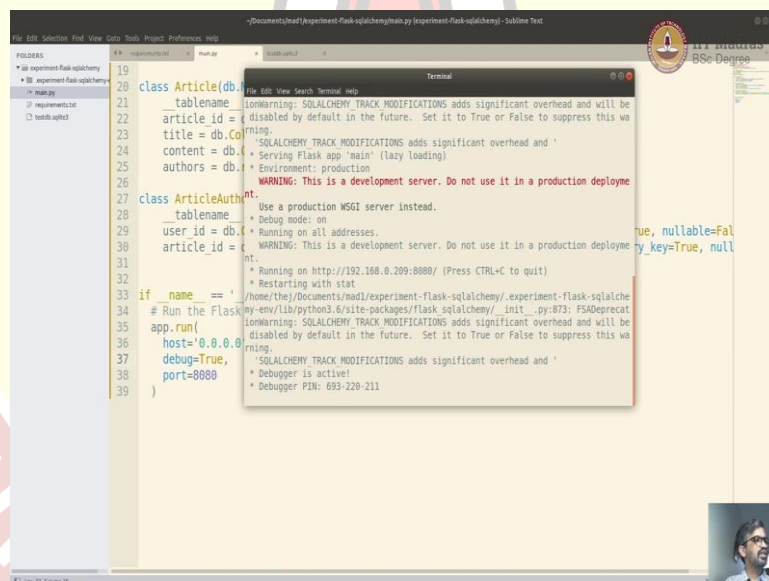
Now what it does is it actually creates this variable, which is pointing to this database, and then initializes everything and we are set. Now again, like last time we had to write model.

Now instead of importing the base model from SQL alchemy, we are going to import the base model everything from flask SQL alchemy, that is the probably the only difference and otherwise, it remains the same.

I am just going to copy paste it to make it quick. Here it is, the only difference is you can see here I am referring everything with respect to db, which is SQL alchemy, which is flask SQL alchemy here. Same thing, integer comes from db, column comes from db. Same thing with, foreign key comes from db, etc, rest of them are the way you set up relationships set up other things will all remain the same.

We are just using now from flask SQL alchemy, instead of directly using from SQL alchemy. That is, it. That is the only difference. Right now, we are set up all the models and everything. Just going to write the main class to start the flask app. Here. What I am doing is I am binding into the local and then enabling the debug and then setting the port to 8080.

(Refer Slide Time: 09:45)



The screenshot shows a code editor with two files: `requirements.txt` and `main.py`. The `main.py` file contains the following code:

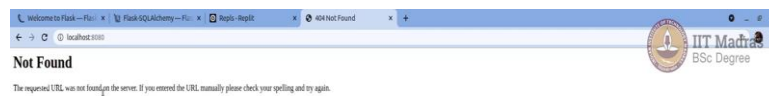
```
19 class Article(db.Model):
20     __tablename__ = 'article'
21     article_id = db.Column(db.Integer, primary_key=True)
22     title = db.Column(db.String(200))
23     content = db.Column(db.String(200))
24     authors = db.Column(db.String(200))
25
26
27 class ArticleAuth(db.Model):
28     __tablename__ = 'article_auth'
29     user_id = db.Column(db.Integer, primary_key=True)
30     article_id = db.Column(db.Integer, primary_key=True)
31
32
33 if __name__ == '__main__':
34     # Run the Flask app
35     app = Flask(__name__)
36     app.run(host='0.0.0.0', debug=True, port=8080)
```

The terminal window shows the output of running the application:

```
WARNING: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be
disabled by default in the future. Set it to True or False to suppress this wa
rning.
SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and '
Serving Flask app 'main' (lazy loading)
 * Environment: production
WARNING: This is a development server. Do not use it in a production environme
nt.
 * Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
WARNING: This is a development server. Do not use it in a production environme
nt.
 * Running on http://192.168.0.209:8080/ (Press CTRL+C to quit)
 * Restarting with stat
/home/thej/Documents/mad/experiment-flask-sqlalchemy/experiment-flask-sqlalche
my-env/lib/python3.6/site-packages/flask_sqlalchemy/_init_.py:873: FSADeprecat
ionWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be
disabled by default in the future. Set it to True or False to suppress this wa
rning.
 * SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and '
 * Debugger is active!
 * Debugger PIN: 693-220-211
```

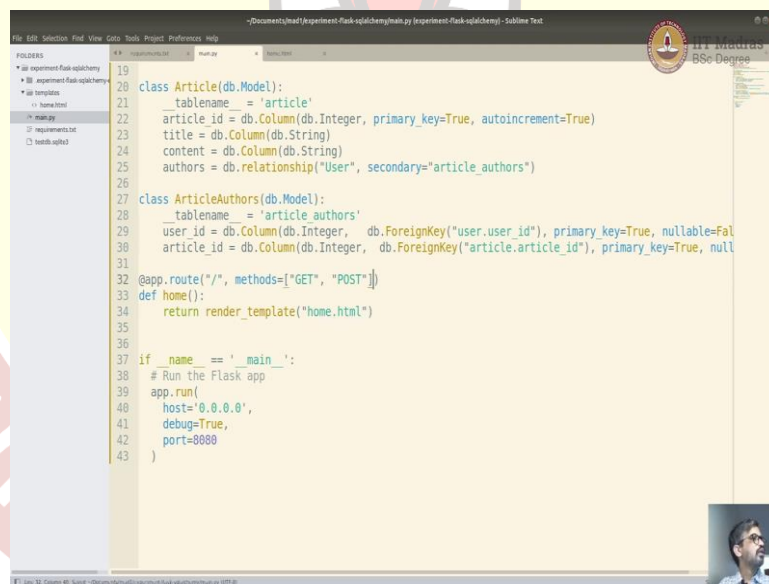
Now I should be able to run it, nothing will happen. But at least it should run. So let us see. So, it is running. I mean nothing will happen.

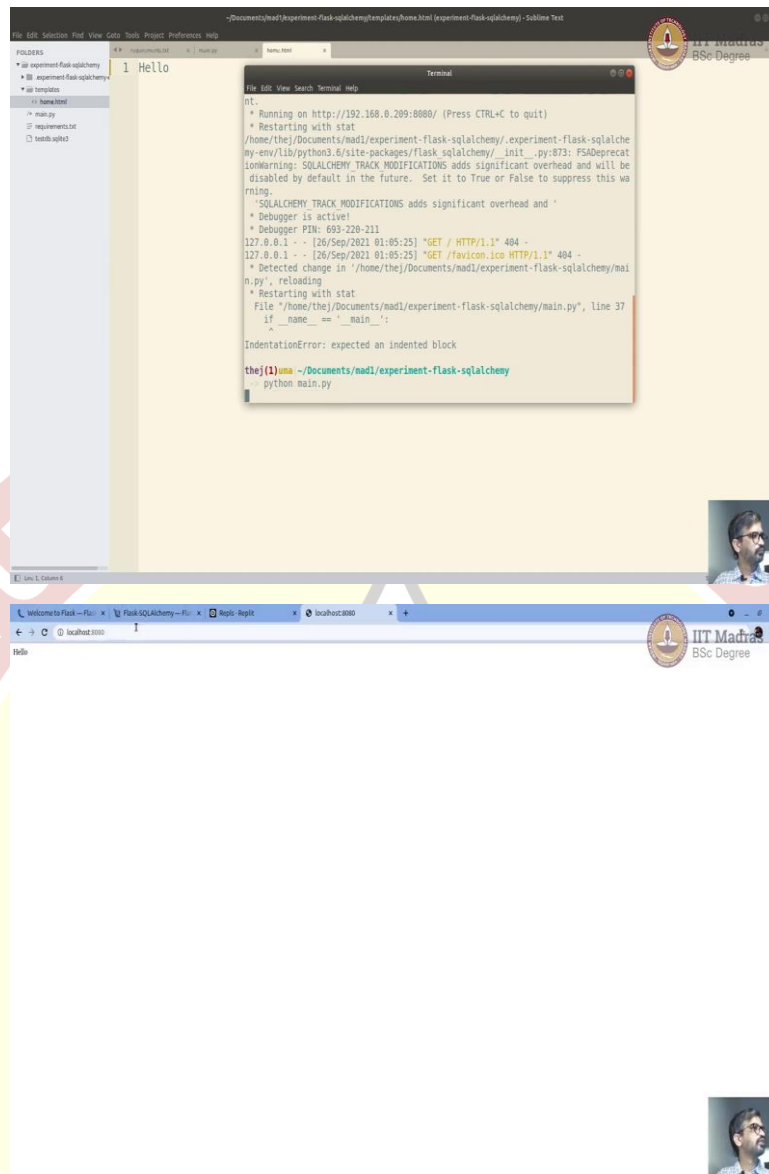
(Refer Slide Time: 10:04)



I think if I go to this path which is my localhost. It might throw 404 because there is no control setup but it is running.

(Refer Slide Time: 10:12)

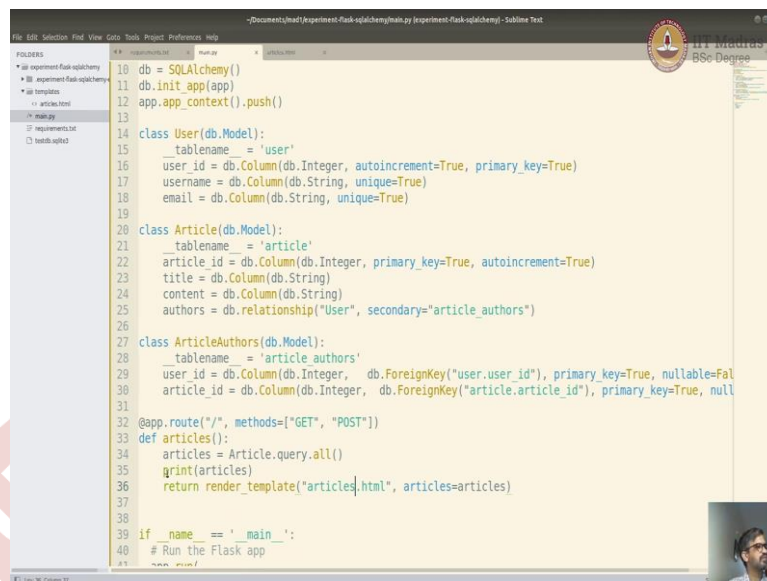




Now, let us just set up one hello world thing as usual, we always used to do hello world. So, and then we can change that, home, and then this points to base, I am going to route it to base, and I am going to create a template folder. Like last time, like in the flask app, we are creating the template folder to keep all our templates. A new folder, templates, by default, flask creates all of its templates from templates folder.

Hence doing that, I am going to create a new file inside that, I am going to call it home. html. I am not going to do much, I am just going to test it. So, I am just going to put Hello, so here, I am just going to render that, this is just for testing that everything is working.

(Refer Slide Time: 11:58)



```
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     print(articles)
36     return render_template("articles.html", articles=articles)
37
38
39 if __name__ == '__main__':
40     # Run the Flask app
```

The setup is working. So at least now, when I go to slash(/), it should go to hello.html and print hello. Let us see, run it. Go back here, refresh Hello. So now our setup is working basic flask setup is working. Now, we have not yet connected and done anything with respect to the database.

We are not done anything, we have set up the model, but we are not done anything. Now, this is articles and uses. It is like a blog or an article or a new site. So, the homepage, actually, I wanted it to be a list of articles. I do not want it to be some Hello message, I actually wanted it to be a list of articles displayed.

So, let us say instead of home, let it call it articles. And also create a template called articles instead of home.html. So, I am going to just rename it to articles.html. So here, how are we going to get articles, we need to get articles from article table, article model. So, it is quite straightforward.

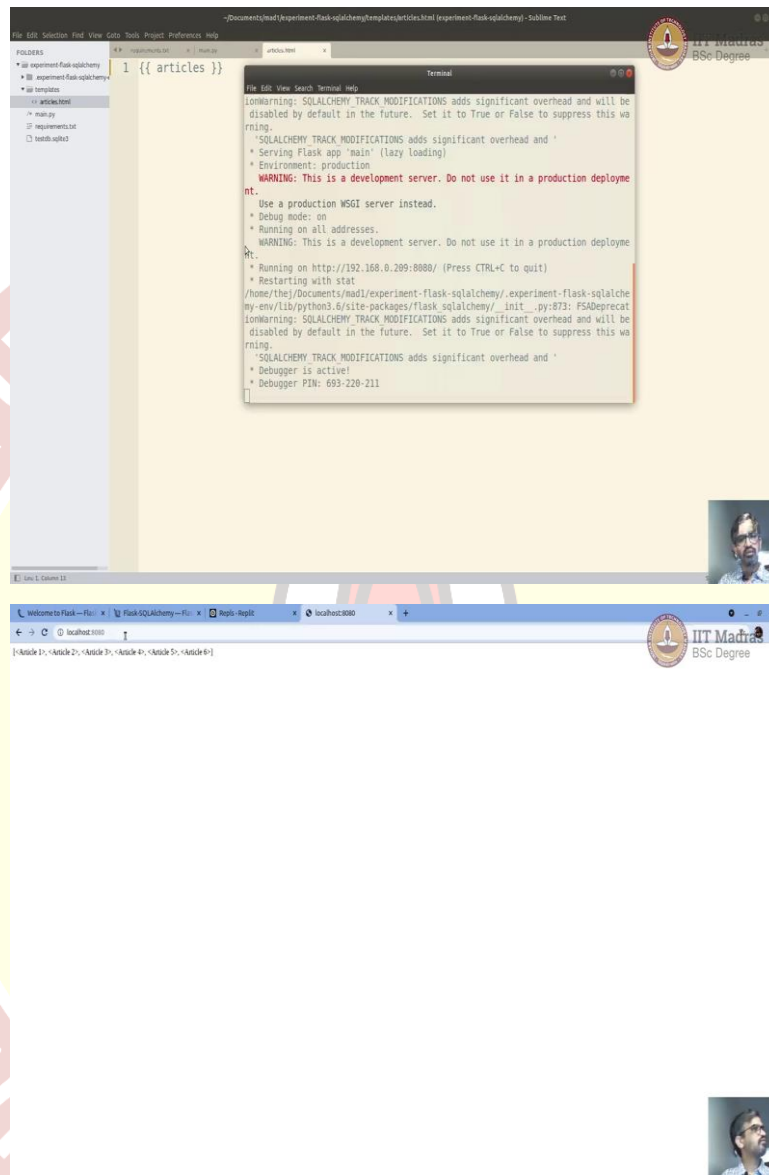
So, we are just going to get all the articles from this table. So that straightforward articles=Article.query.all() this statement of SQL alchemy will give you all the articles from this article model.

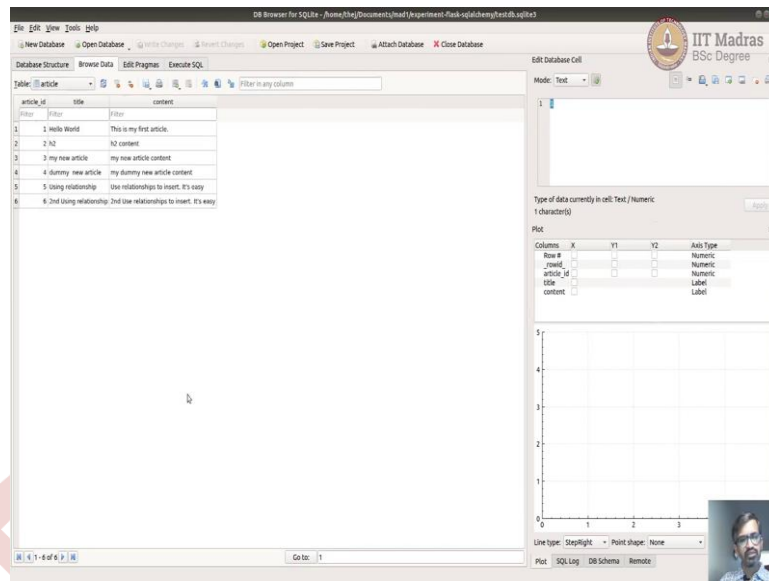
Or rather, all the rows from this article model, there are other ways to write it as well, you can also use, for example, other way of querying, which is using the db.session.query(), and then .all, but I think this is more intuitive. So let us use this. Now we have got all the articles.

We can print them, but it does not make any sense. We need to use them in the template. So, I am going to pass it to the template as articles. I am passing it to the template testing as article

so we renamed the template, so let it be articles.html, now it is going there. And let us see whether we are getting it there. Let us not print here.

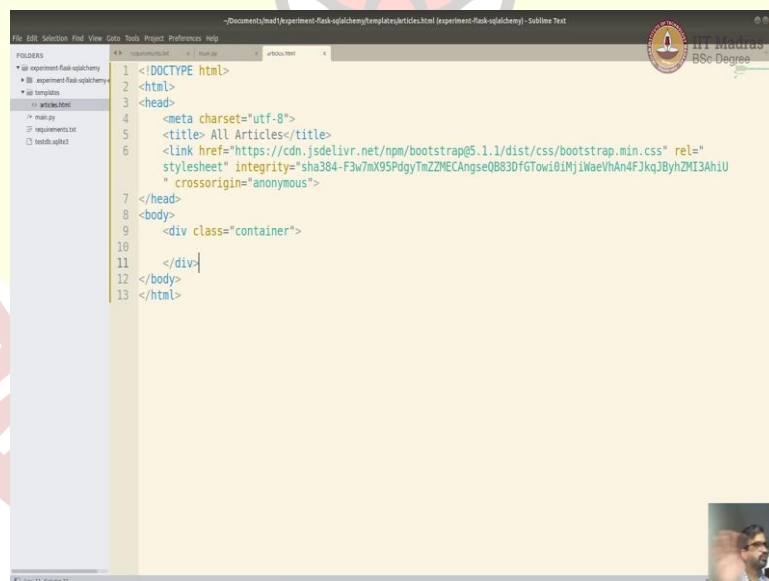
(Refer Slide Time: 14:15)

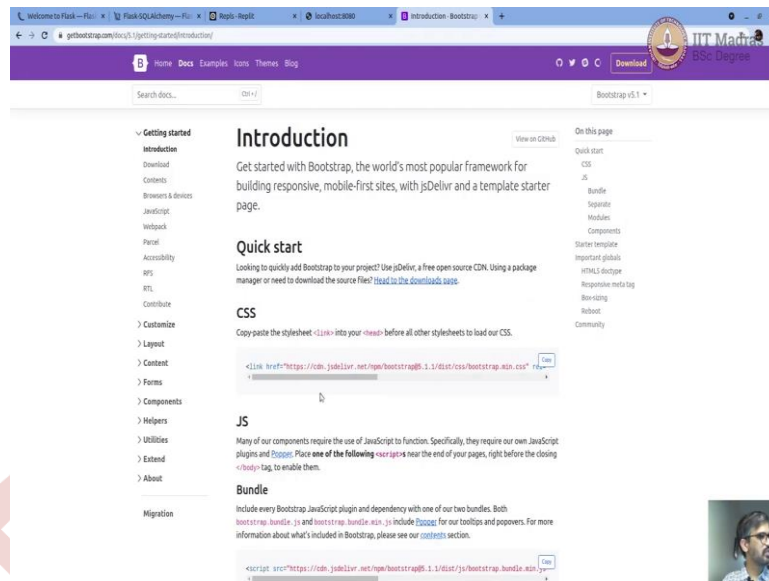




Let us just do just simple Jinja. I am just directly print articles. This is just debugging, making sure things are working. So, this is going to start window refresh. So, I am going to get an array of six articles. Let us go to articles table, there are six articles. So now we somehow need a way to display these articles. So, you create HTML.

(Refer Slide Time: 14:50)

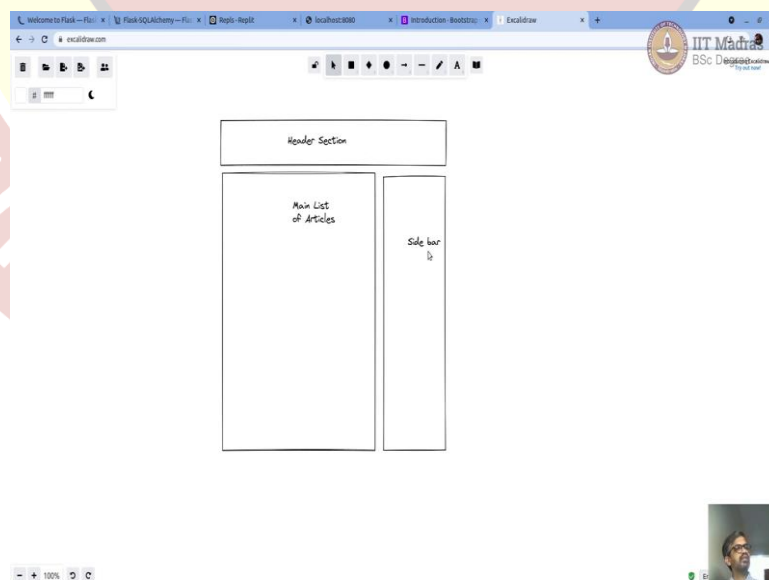


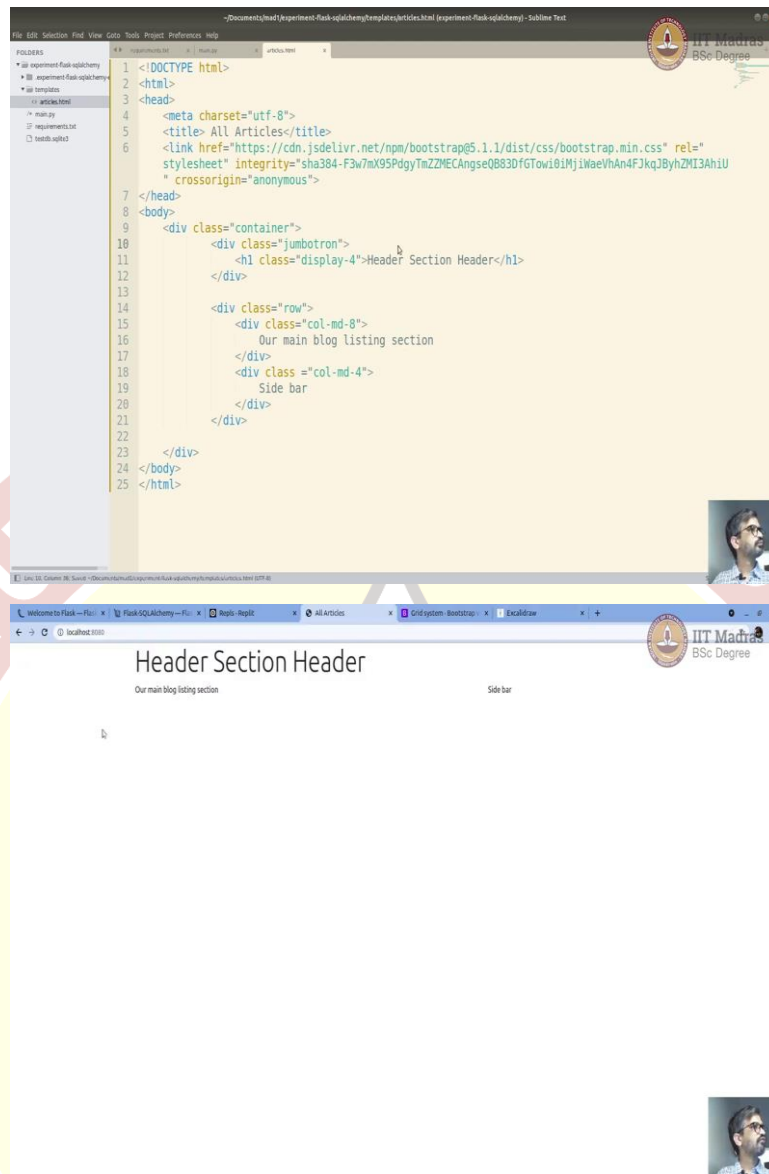


So here I have created a template HTML base template and, everything minimal tags that are required, I am just going to put title as all articles. Now, I do not want to design every part of CSS by myself, I want to use bootstrap. So, I am going to get bootstrap. Get started, and I am going to use that CSS line from here.

You are going to use that CSS, like in bootstrap. The first thing that we want to create is a container. Let us put a container, container div here, that is just done by just adding a div container like this.

(Refer Slide Time: 16:01)





So now I want to divide the section of the page into two parts or like this, this is the main list of articles. I am just doing a wireframe here. And, I want to have a sidebar here. And I want a top header here, like a header section, just displaying something this is my simple bootstrap. So, I am just this whole thing will be in the container, we are going to add this part, this part and this part, for the header section, we can add something called jumbotron. This is also from the bootstrap. So, this is the one this is the header section. So, I am just going to make it like a header section here.

We can rename it later and within the rest I want to divide into these two things. Now that you can do that by lay outing. I am not going to go into much of it, you can look into a bootstrap tutorial, or you can look into this area here. Now I am going to create a row and then create two columns, since the whole width is 12, I am going to create for the main

section, column width of 8 and for the sidebar column width of 4 that is also done by first creating the row.

Let us do a class row. And then within the row, I want to divide that into two parts. Which is one is, of column length 8 and then other one is column length 4, here our main blog listing section, this is our side bar, Let us see how it looks, I am Just want to save it. Do started just in case and go back to localhost 8080 and refresh it. So, you can see that there is header section, there is main blog listing section and the sidebar, I am not added the border, but, you do not need to, but you can add it as well. If you want to.

