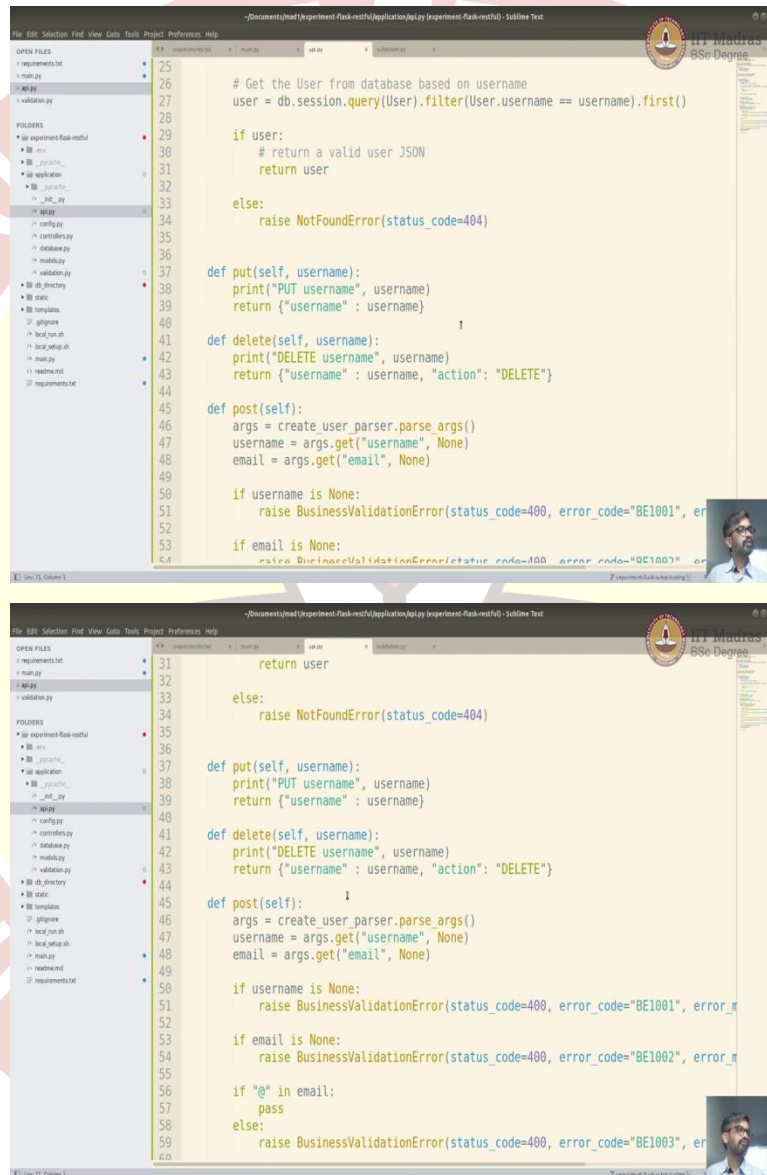


**IIT Madras**  
ONLINE DEGREE

(Refer Slide Time: 0:15)



DB Browser for SQLite - /home/ibag/Documents/Used/Experiment-Flask-restful/db\_browser/restful.db.sqlite3

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

Table: article\_authors

article_id	user_id
1	1
2	1
3	2
4	3
5	4
6	5
7	6
8	2

Go to: 1

Mode: Text

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

Plot

Columns	X	Y1	Y2	Axis Type
Row #				Numeric
rowid				Numeric
article_id				Numeric
user_id				Numeric

Line type: Steplight Point shape: None

Plot SQL Log DB Schema Remote

DB Browser for SQLite - /home/ibag/Documents/Used/Experiment-Flask-restful/db\_browser/restful.db.sqlite3

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

Table: article\_authors

article_id	user_id
1	1
2	1
3	2
4	3
5	4
6	5
7	6
8	2

Go to: 1

Mode: Text

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

Plot

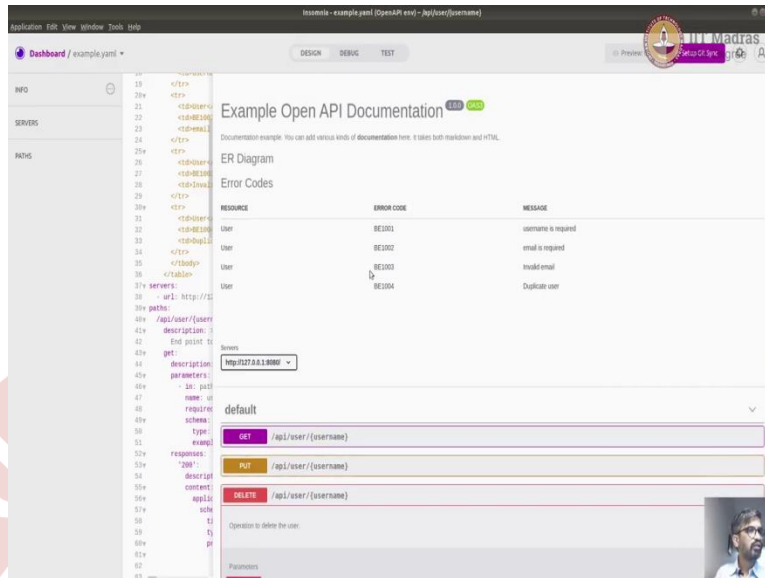
Columns	X	Y1	Y2	Axis Type
Row #				Numeric
rowid				Numeric
article_id				Numeric
user_id				Numeric

Line type: Steplight Point shape: None

Plot SQL Log DB Schema Remote







Similarly, you can write, delete and PUT, for example, PUT will get first and then do the saving, updating and saving. Let us try that. Or we can try delete, delete also will check if other dependencies are not there, based on no other dependencies, it tries to delete in our database, we could see that we cannot really delete a user, if there is no article associated with it, there is a conflict.

So, either you can check it at the database level by, you can try to delete it, and it might throw a database exception. And then you can tell it to now you can return a response saying, you cannot delete because there are articles for this user. Or you can always say that, please delete article before you delete the user.

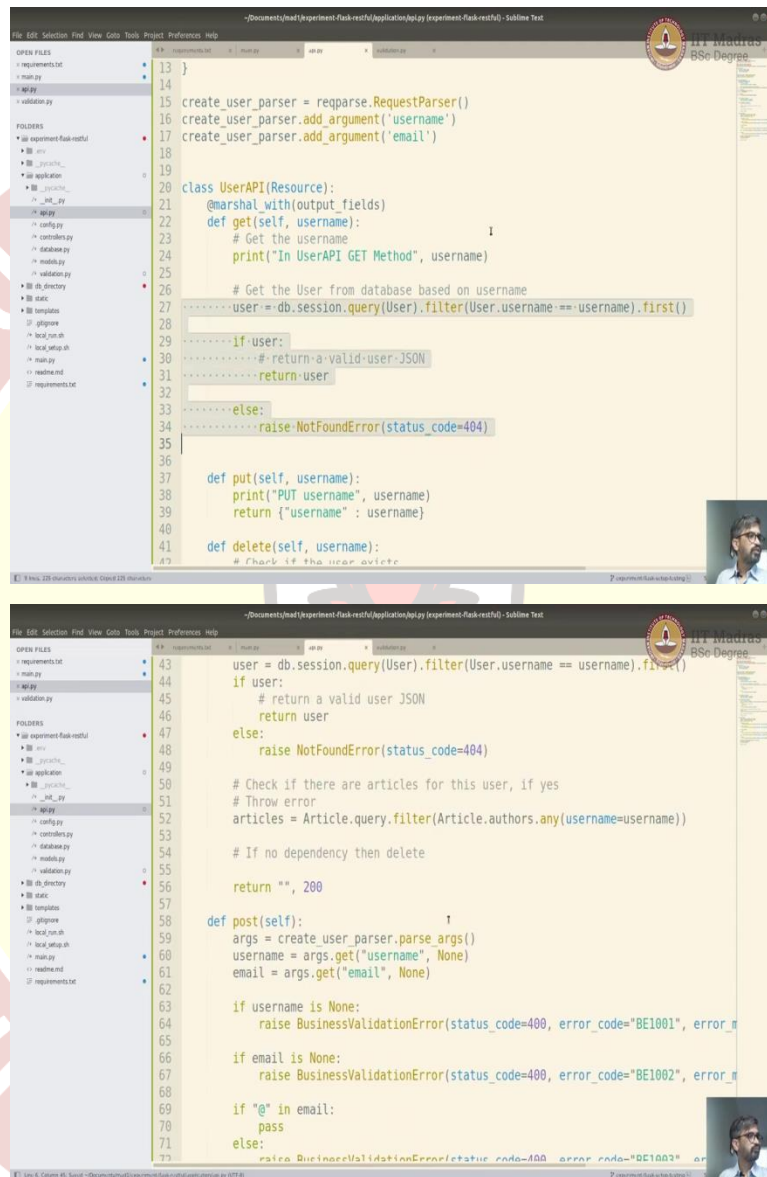
Or the other way is to, you could if the API wanted, you can also delete a user along with deleting his articles as a default thing. So, depends on how the API's are defined, what is the requirement of the API, or you can also check first here, instead of depending on the database, you can find first if there are articles, then say that you cannot delete it.

Let us do that. Let us say get user name, let us say check. If the user exists first, then check if there are articles for this user, if yes, throw error if no dependency, then delete. In this case, we would not be returning anything we just returning, empty, and what could be the status message, can check here go to delete. Everything was well, we will return 200.

So, you could do I mean, I do not think they have defined error messages or here. But if they have defined, then you could throw but I am going to define one end throw, the most probably

new if you are running an assignment or, getting real documentation, you would have got that also.

(Refer Slide Time: 3:16)



```
13 }
14
15 create_user_parser = reqparse.RequestParser()
16 create_user_parser.add_argument('username')
17 create_user_parser.add_argument('email')
18
19
20 class UserAPI(Resource):
21     @marshal_with(output_fields)
22     def get(self, username):
23         # Get the username
24         print("In UserAPI GET Method", username)
25
26         # Get the User from database based on username
27         user = db.session.query(User).filter(User.username == username).first()
28
29         if user:
30             # return a valid user JSON
31             return user
32         else:
33             raise NotFoundError(status_code=404)
34
35
36
37 def put(self, username):
38     print("PUT username", username)
39     return {"username": username}
40
41 def delete(self, username):
42     # Check if the user exists
```

```
43 user = db.session.query(User).filter(User.username == username).first()
44 if user:
45     # return a valid user JSON
46     return user
47 else:
48     raise NotFoundError(status_code=404)
49
50 # Check if there are articles for this user, if yes
51 # Throw error
52 articles = Article.query.filter(Article.authors.any(username=username))
53
54 # If no dependency then delete
55
56 return "", 200
57
58
59 def post(self):
60     args = create_user_parser.parse_args()
61     username = args.get("username", None)
62     email = args.get("email", None)
63
64     if username is None:
65         raise BusinessValidationError(status_code=400, error_code="BE1001", error_message="username is required")
66
67     if email is None:
68         raise BusinessValidationError(status_code=400, error_code="BE1002", error_message="email is required")
69
70     if "@" in email:
71         pass
72     else:
73         raise BusinessValidationError(status_code=400, error_code="BE1003", error_message="email is not valid")
```



```
File Edit Selection Find View Goto Tools Project Preferences Help
-Documents\mad\experiment-flask-restful\application\api.py (experiment-flask-restful) - Sublime Text

OPEN FILES
- requirements.txt
- main.py
- api.py
- validation.py

FOLDERS
- experiment-flask-restful
  - api
  - application
  - _api.py
  - api.py
  - controllers.py
  - database.py
  - models.py
  - validation.py
- db_directory
- static
- templates
- gunicorn
- local_run.sh
- main.py
- realtime.md
- requirements.txt

1 from flask_restful import Resource
2 from flask_restful import fields, marshal_with
3 from flask_restful import reqparse
4
5 from application.database import db
6 from application.models import User, Article
7 from application.validation import NotFoundError, BusinessValidationError
8
9 output_fields = {
10     "user_id": fields.Integer,
11     "username": fields.String,
12     "email": fields.String
13 }
14
15 create_user_parser = reqparse.RequestParser()
16 create_user_parser.add_argument('username')
17 create_user_parser.add_argument('email')
18
19
20 class UserAPI(Resource):
21     @marshal_with(output_fields)
22     def get(self, username):
23         # Get the username
24         print("In UserAPI GET Method", username)
25
26         # Get the User from database based on username
27         user = db.session.query(User).filter(User.username == username).first()
28
29         if user:
30             # return a valid user JSON
```

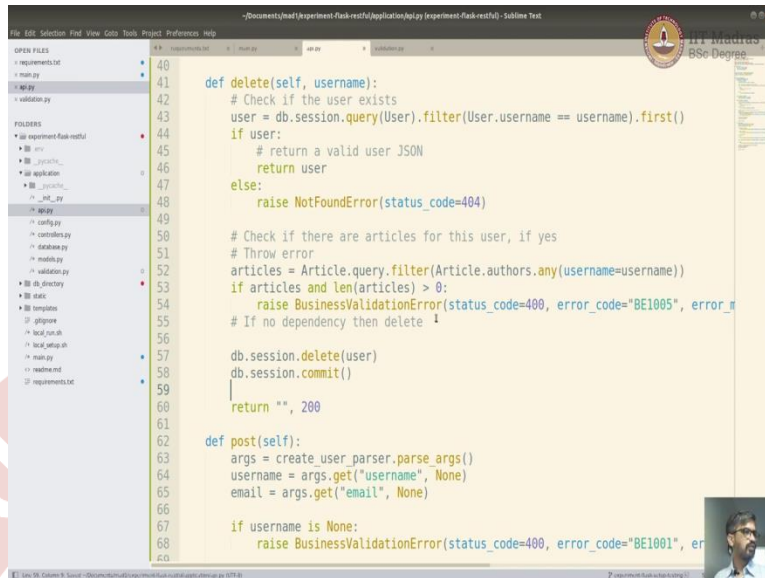
```
43 user = db.session.query(User).filter(User.username == username).first()
44 if user:
45     # return a valid user JSON
46     return user
47 else:
48     raise NotFoundError(status_code=404)
49
50 # Check if there are articles for this user, if yes
51 # Throw error
52 articles = Article.query.filter(Article.authors.any(username=username))
53 if articles and len(articles) > 0:
54     raise BusinessValidationError(status_code=400, error_code="BE1005", error_message="User has articles")
55 # If no dependency then delete
56
57 return "", 200
58
59 def post(self):
60     args = create_user_parser.parse_args()
61     username = args.get("username", None)
62     email = args.get("email", None)
63
64     if username is None:
65         raise BusinessValidationError(status_code=400, error_code="BE1001", error_message="Username is required")
66
67     if email is None:
68         raise BusinessValidationError(status_code=400, error_code="BE1002", error_message="Email is required")
69
70     if "@" in email:
71         pass
72     else:
73         raise BusinessValidationError(status_code=400, error_code="BE1003", error_message="Invalid email format")
```

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

सिद्धिर्भवति कर्मजा







```
def delete(self, username):
    # Check if the user exists
    user = db.session.query(User).filter(User.username == username).first()
    if user:
        # return a valid user JSON
        return user
    else:
        raise NotFoundError(status_code=404)

    # Check if there are articles for this user, if yes
    # Throw error
    articles = Article.query.filter(Article.authors.any(username=username))
    if articles and len(articles) > 0:
        raise BusinessValidationError(status_code=400, error_code="BE1005", error_message="User has articles")
    # If no dependency then delete
    db.session.delete(user)
    db.session.commit()
    return "", 200

def post(self):
    args = create_user_parser.parse_args()
    username = args.get("username", None)
    email = args.get("email", None)

    if username is None:
        raise BusinessValidationError(status_code=400, error_code="BE1001", error_message="Username is required")
```

So, you can check if the user exists, which is same as this call? If there is no user, then you can throw the same error. Straight forward that way that is first step, just reputation. Now the second thing is we want to check if there are articles for this user. That is just a query on the articles table. I will let me just paste that for you.

So, that it is easy. So, I am just going to get all the articles for this user `article.query.filter`, you could use `db.session article (())(04:06)`, or you can query this way. Both ways are valid article input this article from model. Article 0 no, just checking can throw an error I just want to throw an arrow like a business error.

It is say it is BE1005 cannot delete users as they are articles written by this user. In this case, it could be like that. Or it could be that, you might want to write a transaction where you delete the articles first of this user and then delete the user. But here, I am just going to go in a safe way, where I do not want to delete anything.

I just want to warn the user that you are going to try delete, you are trying to delete a user who has articles so I am not allowing it to delete it. So, if there is no dependency, then comes the delete, we can just check how do we delete by going to a flask, SQL alchemy, just check, delete, just removing from the session and deleting.

So, we can just do. And `(())(06:09)` delete the user that committed. So, this is again, a very simple way of deleting the user. If there are no dependencies, and there could be more and more conditions. You could have many things to check here. But this is the most impressed.

(Refer Slide Time: 6:32)

The top screenshot displays the DB Browser for SQLite application. The 'Table: user' is selected, showing 6 rows of data. The 'Edit Database Call' panel on the right shows the 'user' table selected. The bottom screenshot shows the Insomnia API client interface. The 'default' REST client is selected, showing a 'GET /api/user/{username}' endpoint. The 'Execute' button is highlighted, and the 'Responses' section shows a successful 200 response with the message 'Successfully Defined'.

id	username	email
1	1	@example.com
2	2	@example.com
3	3	@example.com
4	4	@example.com
5	5	@example.com
6	6	@example.com

default

GET /api/user/{username}

PUT /api/user/{username}

DELETE /api/user/{username}

Operation to delete the user.

Parameters

NAME DESCRIPTION

username \* NAME Description

Executing

Responses

Code Description Links

200 Successfully Defined No links

400 Bad request

application/json

Example below Schema

the@joma: ~/Documents/mad1/experiment-flask-restful

```
File Edit View Search Terminal Help
* Restarting with stat
Starting Local Development
* Debugger is active!
* Debugger PIN: 138-015-846
* Detected change in '/home/thej/Documents/mad1/experiment-flask-restful/application/api.py', reloading
* Restarting with stat
Starting Local Development
* Debugger is active!
* Debugger PIN: 138-015-846
* Detected change in '/home/thej/Documents/mad1/experiment-flask-restful/application/api.py', reloading
* Restarting with stat
Starting Local Development
Traceback (most recent call last):
  File "/home/thej/Documents/mad1/experiment-flask-restful/main.py", line 28, in <module>
    from application.api import UserAPI
  File "/home/thej/Documents/mad1/experiment-flask-restful/application/api.py", line 57
    return "", 200
    ^
IndentationError: expected an indented block
```

the@joma:~/Documents/mad1/experiment-flask-restful

code	description	status	parameters
45v		200	Successfully Deleted.
46v			
47	name: u	400	Bad request.
48	required		
49v	schema:		
50	type:		
51	example:		
52v			
53v	'200':		
54	description:		
55v	content:		
56v	applic:		
57v	sch:		
58	ti:	404	User does not exist.
59	ti:		
60v	pr:	500	Internal Server Error.
61v			
62			
63			

the@joma: ~/Documents/mad1/experiment-flask-restful

```
File Edit View Search Terminal Help
IndentationError: expected an indented block
the@joma:~/Documents/mad1/experiment-flask-restful$ sh local_run.sh
Welcome to the setup. This will setup the local virtual env.
And then it will install all the required python libraries.
You can rerun this without any issues.
-----
Enabling virtual env
Starting Local Development
* Serving Flask app 'main' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  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 deployment.
* Running on http://192.168.0.209:8080/ (Press CTRL+C to quit)
* Restarting with stat
Starting Local Development
* Debugger is active!
* Debugger PIN: 138-015-846
```

the@joma:~/Documents/mad1/experiment-flask-restful

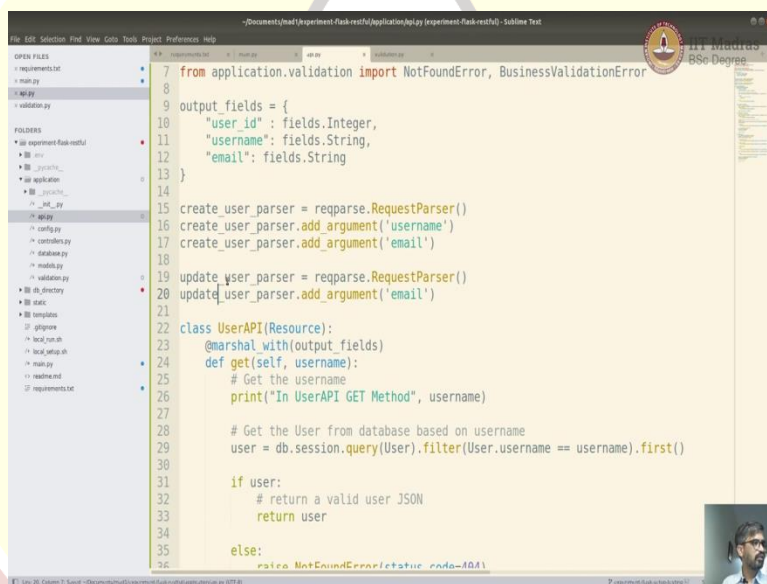
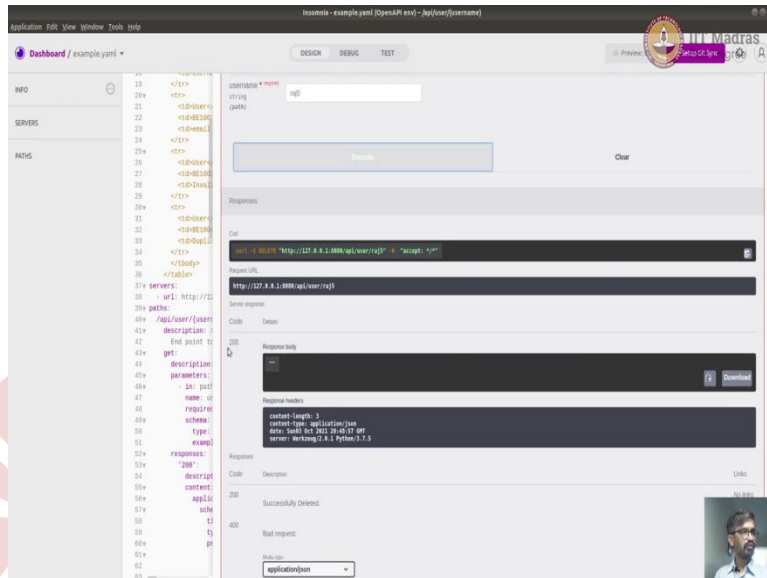
code	description	parameters	Responses
45v			
46v			
47	name: u		
48	required		
49v	schema:		
50	type:		
51	example:		
52v			
53v	'200':		
54	description:		
55v	content:		
56v	applic:		
57v	sch:		
58	ti:		
59	ti:		
60v	pr:		
61v			
62			
63			











So, let us check for this condition. I think, for example, user 1 has articles. So, like if I try to delete him, delete that it should say that you cannot delete in separate execute. Type 0 failed to fetch. Let us start again. There was an error in the previous one, shut down the server. So, usually just check when you are changing. Just check whether the server is restarted.

If not just restart it. It execute trying to run throwing some error object of type JSON up I am trying to return the user is well it here and I do not have to do that. I just copy pasted so then I will raise an error if not I do this, if there is no dependency then delete. Now this should work because I think I had a bug here because I copy pasted it. Let us go here.

Let us go to here raj5, execute, got deleted return 200. Let us see here. Refresh this raj5 is gone. That is our simple delete functionality. By API can similarly do put, I am just going to leave that for, for your experimentation. But in case of PUT, you are going to return the object again. So, you might want to use marshal output, and then return the whole object.

In this case, we have written a create user parser, we will just write the update user parser just have email update, we are not going to update the user name through this will be update, update. So, I am going to use this to parser in input.

```
File Edit Selection Find View Go Tools Project Preferences Help
-Document: /root/.kaggle/kaggle-api/Application.py (Application.py) - Sublime Text

OPEN FILES
- requirements.txt
- main.py
- Application.py
- validation.py

FOLDERS
- /usr/experiment/Task-001
- /src
- /src/_pyutils_
- /src/application
- /src/_pyutils_
- /src/_src_py
- /src/pyutils
- /src/constants.py
- /src/database.py
- /src/models.py
- /src/validation.py
- /src/db_directory
- /src/status
- /src/templates
- /src/generator
- /src/test_data.py
- /src/test_setup_db
- /src/main.py
- /src/testcase.py
- /src/requirements.txt

if user:
    # return a valid user JSON
    return user

else:
    raise NotFoundError(status_code=404)

def put(self, username):
    args = update_user_parser.parse_args()
    email = args.get("email", None)

    # Check if the user exists
    user = db.session.query(User).filter(User.username == username).first()
    if user is None:
        raise NotFoundError(status_code=404)

    return {"username" : username}

def delete(self, username):
    # Check if the user exists
    user = db.session.query(User).filter(User.username == username).first()
    if user is None:
        raise NotFoundError(status_code=404)

    # Check if there are articles for this user, if yes
    # Throw error
    articles = Article.query.filter(Article.authors.any(username=username)).first()
    print(articles)
```

```
def put(self, username):
    args = update_user_parser.parse_args()
    email = args.get("email", None)

    if email is None:
        raise BusinessValidationError(status_code=400, error_code="BE1002", error_message="email is required")

    if "@" in email:
        pass
    else:
        raise BusinessValidationError(status_code=400, error_code="BE1003", error_message="Invalid email")

    user = db.session.query(User).filter(User.email == email).first()

    if user:
        raise BusinessValidationError(status_code=400, error_code="BE1006", error_message="Duplicate email")

    # Check if the user exists
    user = db.session.query(User).filter(User.username == username).first()
    if user is None:
        raise NotFoundError(status_code=404)

    return {"username": username}
```

```
args()

or(status_code=400, error_code="BE1002", error_message="email is required")

or(status_code=400, error_code="BE1003", error_message="Invalid email")

filter(User.email == email)

or(status_code=400, error_code="BE1006", error_message="Duplicate email")

filter(User.username == username).first()

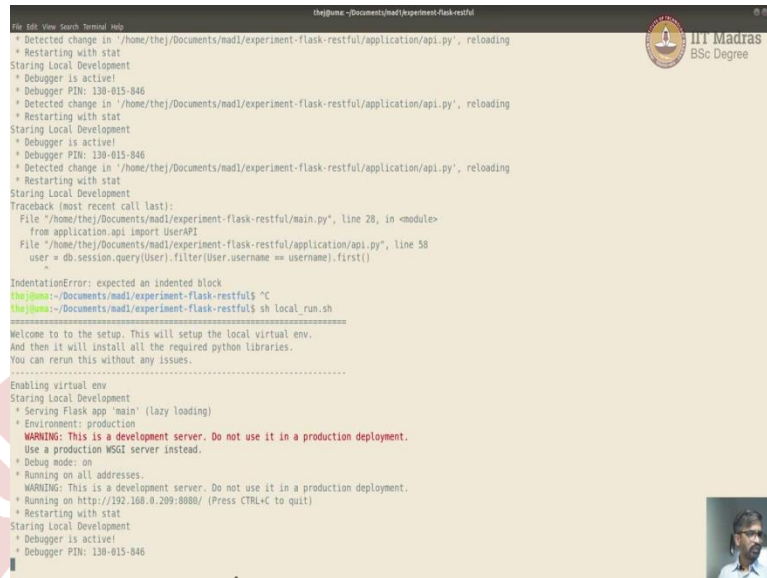
code=404)
```











```
File Edit View Search Terminal Help
/home/thej/~/Documents/mad/experiment-flask-restful/application/api.py', reloading
* Restarting with stat
Starting Local Development
* Debugger is active!
* Debugger PIN: 138-815-846
/home/thej/~/Documents/mad/experiment-flask-restful/application/api.py', reloading
* Restarting with stat
Starting Local Development
* Debugger is active!
* Debugger PIN: 138-815-846
/home/thej/~/Documents/mad/experiment-flask-restful/application/api.py', reloading
* Restarting with stat
Starting Local Development
Traceback (most recent call last):
  File "/home/thej/~/Documents/mad/experiment-flask-restful/main.py", line 28, in <module>
    from application.api import UserAPI
  File "/home/thej/~/Documents/mad/experiment-flask-restful/application/api.py", line 58
    user = db.session.query(User).filter(User.username == username).first()
    ^
IndentationError: expected an indented block

thejames@kali:~/Documents/mad/experiment-flask-restful$ sh local_run.sh
=====
Welcome to the setup. This will setup the local virtual env.
And then it will install all the required python libraries.
You can rerun this without any issues.
=====
Enabling virtual env
Starting Local Development
* Serving Flask app "main" (Lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  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 deployment.
* Running on http://192.168.0.209:8080/ (Press CTRL+C to quit)
* Restarting with stat
Starting Local Development
* Debugger is active!
* Debugger PIN: 138-815-846
```

So, let us, even before we go there, we let us check here. Parse this value similar to how we are done here. So, I am just going to copy here. So, I am not going to get user name, this is not create user, this is update user parse. Now if email is valid, and then do the other checkings, we will have to do that again here. Let us do that, which is similar to our posts.

So, again, I am going to copy paste. So, email is none, then it will throw an error because it needs a valid email. Again, valid email but there is one more condition here say if there is already in user with the same email id, we do not want to update it. So, we want to check whether there is another user in the same email id.

So, let us just check that. If we can find another user with user.email=email, then, it will become a duplicate user. If user and will throw an error. We do not want to use the same email id. I do not think we have an error code defined for that we can just do 5 this is we can do it as 6. And just say duplicate email.

And if then let us continue. After, that will check whether this specific user by this username exists. If, not will throw an error. This will be dot first. Correct. I will check if this user exists. If, not will throw an error. If everything is we will do user model update user.email = email past. And then the rest of them is same.

Let us just confirm updating the record, inserting querying, deleting, inserting or updating a same as far as I know, it is just add that user just similar to this, this user and then commit, but here we

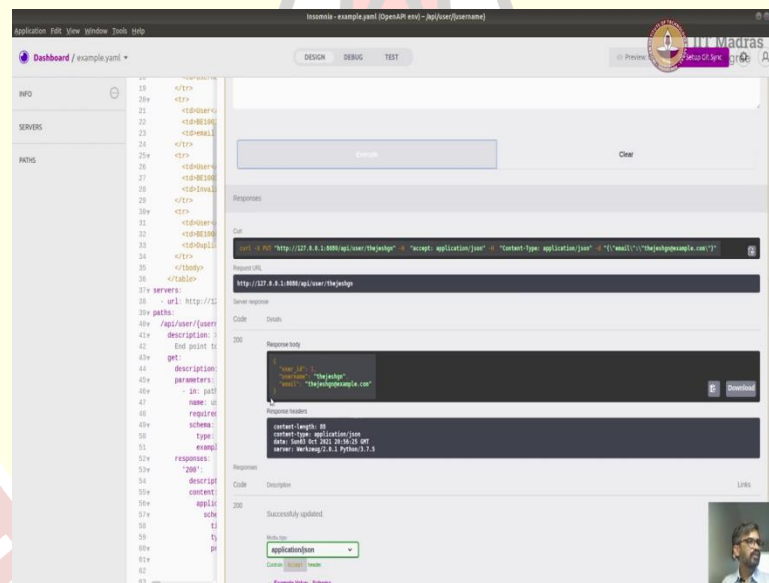


are going to return the user updated user. Hence we are going to the marshal it. Similar, output. So, let us stop and restart, see if there is any errors seem to be no error.

Can let us go back to her. So, it takes the one email, let me try out user name, let me just try and use an email that is given here. Christian, let us make that. Let not give an email, let us save throws an error. It is throw an email is required, then let us give an existing email of some other user, same user does not matter.

It should throw an error that duplicate email, duplicate email through an error duplicate email. Now, if we put an non existing user thejeshgn123, through an error there 2, throwing in a duplicate email. But regardless, if we give a valid email 2 let us say, I was giving Thejeshgn123 for this user, but this user does not exist you want to this is unique and valid email.

(Refer Slide Time: 16:44)



Application: 560 View Window Tools Help

Example Open API Documentation

Documentation example. You can add various kinds of documentation here. It takes both markdown and HTML.

ER Diagram

Error Codes

RESOURCE	ERROR CODE	MESSAGE
User	BE1001	username is required
User	BE1002	email is required
User	BE1003	invalid email
User	BE1004	Duplicate user

default

GET /api/user/{username}

PUT /api/user/{username}

DELETE /api/user/{username}

POST /api/user

DB Browser for SQLite - /home/389/Desktop/Used/Apprentice/flash-res/js/DB\_browser/index.html

File Edit View Tools Help

New Database Open Database Recent Changes Open Project Save Project Attach Database Close Database

Database Structure Browser Data Edit Pragma Execute SQL Filter in any column

article_id	title	content
1	hello World	Welcome to Flash's documentation. Get st...
2	Flash-SQLAlchemy	Flash-SQLAlchemy is an extension for Fla...
3	my new article	my new article content
4	dummy new article	my dummy new article content
5	Using relationship	Use relationships to insert. It's easy
6	2nd using relationship	2nd use relationships to insert. It's easy

1 of 6

Go to: 1

Edit Database Cell

Mode: Text

Type of data currently in cell: Text / Numeric

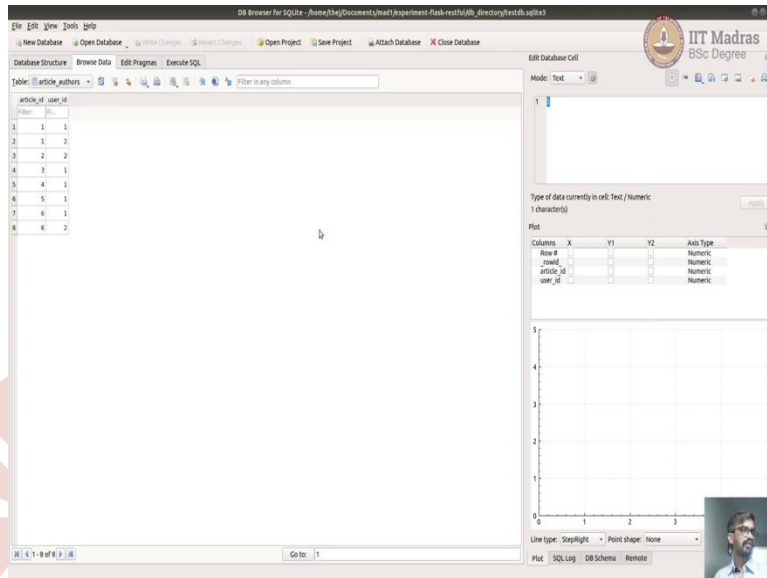
1 character(s)

Plot

Columns	X	Y1	Y2	Axis Type
Row #				
article_id				Numeric
title				Numeric
content				Label

Line type: Steplight Point shape: None

Plot SQL Log DB Schema Remote



So, choose not found this user is not found. So, which is the error that we had expected here it is, in our costal resources not found. Now, the last validation that we want to check is we want to update if everything goes well. So, now what we can do is we can just update this Thejeshgn@example.com.

This is a valid user, a valid email id also unique email id not used by anyone. So, now let us if I execute, it should run and return the response which is an updated response you can see that it is written the user id user name as Thejeshgn email thejeshgn@example.com it is just in the day ways if I refresh by doing Ctrl 5 or Ctrl R you can see that the email has been update.

So, you can do much more validation or in a different way but this is how you generally write and this is a simpler way of writing it of course, you can had more customization more custom validators a different type of error (18:04) some of those error it not defined extra, ideally I would stick to what they have sent usually the documentation will be full complete this documentation is not complete because not all the errors are mentioned here.

Usually all the errors will be mentioned and all the cases will be mentioned so it is crystal clear as to what to implement. That is all for the rest of the tables I will leave it to you how to implement for example how to insert all of the credit into articles or sent authors as part of articles so that you can insert into article authors, et cetera, et cetera.

You can probably first write the documentation for yourself, get it validated by your one of your peer or one of your course instructor and then try it out to implement yourself just as an experiment. That is all from me today. Thank you so much. Bye

