Jalen Jackson
CPSC 449
Project 2 - Microblog Microservices

Users API Operations

`all_users()`

SQL Query	Selects all columns in Users Table
URL Endpoint	http://127.0.0.1:5000/

HTTP Verb	**HTTP Status Code**	**HTTP Response to Status Codes**
GET	200 OK	Returns all users in the database using the **SQL Query**
POST	201 CREATED	1) Encrypt `password` by generating a password hash 2) Runs `create_user(username, email, password)`

`create_user(username, email, password)`

SQL Query	1) Create user with `username`, `email`, and a `password` in JSON
	{
	"username": `username`, "email": `email`, "password": `password`
	}
	2) Insert this user into the Users table in the database

`user(username)`

URL Endpoint	http://127.0.0.1:5000/<string:`username`>

HTTP Verb	**HTTP Status Code**	**HTTP Response to Status Codes**
GET	• 200 OK	Run `user_by_username(username)` query

`user_by_username(username)`

SQL Query	1) Selects all columns from user with `username` in Users table

Jalen Jackson
CPSC 449
Project 2 - Microblog Microservices

authenticate_user(username)

URL Endpoint	http://127.0.0.1:5000/<string:username>/authenticate?password=suppliedPassword ● suppliedPassword is a parameter submitted in the URL

HTTP Verb	**HTTP Status Codes**	**HTTP Response to Status Codes**
GET	● 302 FOUND ● 303 NOT FOUND ● 409 CONFLICT	● suppliedPassword **=** password of user(username) ● suppliedPassword **!=** password of user(username) ● suppliedPassword **not passed** in as parameter in URL

following(username)

URL Endpoint	http://127.0.0.1:5000/<string:username>/following?username=removeFollower ● removeFollower is username parameter submitted in the URL

HTTP Verb	**HTTP Status Codes**	**HTTP Response to Status Codes**
GET	200 OK	Run show_following(username)
POST	201 CREATED	Run add_follower(username, usernameToAdd)
DELETE	206 PARTIAL CONTENT	Run remove_follower(username, usernameToRemove)

show_following(username)

SQL Query	1) Merge Relationships & Users Table on followed names & username 2) Select follower names, followed names, usernames, & emails from Table (1) 3) Get Relationships where the name of the follower is username in Table (2) 4) Select usernames & emails from Table(3)

add_follower(username, usernameToAdd)

SQL Query	1) Add username & usernameToAdd to Relationships table
	{
	"username": usernameToAdd
	}

`remove_follower(username, usernameToRemove)`

SQL Query	1. Remove `username` & `usernameToAdd` relationship out of Relationships table

Timelines API Operations

`get_public_timeline()`

URL Endpoint	http://127.0.0.1:5001/

HTTP Verb	**HTTP Status Codes**	**HTTP Response to Status Codes**
GET	200 OK	Run `public_timeline()` query

`public_timeline()`

SQL Query	1) Select author names, created timestamps, and tweet texts from Posts table 2) Sort rows from Table (1) by most recent time in created timestamp 3) Show 25 rows in Table (2)

`get_home_timeline(username)`

URL Endpoint	http://127.0.0.1:5001/<string:`username`>/home

HTTP Verb	**HTTP Status Codes**	**HTTP Response to Status Codes**
GET	200 OK	Run `home_timeline()` query

`home_timeline(username)`

SQL Query	1) Merge Relationships & Posts Table on followed names & author name 2) Select these columns from Table (1): a) Follower names b) Followed names c) Post's author name d) Post created timestamp e) Post's tweet 3) Get Relationships where follower name is `username` in Table (2) 4) Select post's author_name, created timestamp, & tweet from Table (3)

	5) Sort rows from Table (1) by most recent time in created timestamp 6) Show 25 rows in Table (2)

get_user_timeline(username)

URL Endpoint	http://127.0.0.1:5001/<string:username>/user

HTTP Verb	**HTTP Status Codes**	**HTTP Response to Status Codes**
GET	200 OK	Run `user_timeline(username)` query
POST	201 CREATED	Run `post_tweet(username)` query

`user_timeline(username)`

SQL Query	1) Select author names, created timestamps, and tweet texts from Posts table 2) Get Posts where the author name of a post is `username` 3) Sort rows from Table (2) by most recent time in created timestamp 4) Show 25 rows in Table (3)

`post_tweet(username)`

SQL Query	1) Create `username`'s post by typing in the tweet's text in JSON
	{
	"tweet": <text>
	}
	2) Insert it to the Posts table in the database