Aditi Pratap Patil
Ankita Udaykumar Kulkarni

REST API for each service

UsersMicroservice.py:

- **createUser(username, email, password)**

  A new user can be created in this service, it takes username, email and password from the user and saves it in the User table in database.

  HTTP Method: POST

  Status code: Success-201

  Failure- 409 Conflict

  Endpoint: /creatUser

  Sample JSON request:

  {

  “usernameAPI”: “ankita”,

  “emailAPI”: “ankita@gmail.com”,

  “passwordAPI”: “ankita@123”

  }

  Success Response:

  {

  “message”: “User created”,

  “status_code”: “201”

  }

  Failure Response:

  {

  “message”: “username / email_ /password_ is not in request”,

  “status_code”: “409”,

  “error”: “Conflict”

}

- authenticateUser(username, hashed_password)

  This service provides user authentication. When the user enters the username and password, the password is hashed using hashlib in python flask. The entered password is then verified with the hashed password saved against that user in the database. After verifying the hashed passwords, we get a response- true if passwords match and false if any of the character does not match the password in the database.

  HTTP Method: GET

  Status code: Success-200 OK

  Failure- 400 Bad Request, 500 Internal Server Error

  Endpoint: /authenticateUser

  Sample JSON request:

  {

  “usernameAPI”: “ankita”,

  “passwordAPI”: “ankita@123”

  }

  Success Response:

  {

  “message”: “True”,

  “status_code”: “200”

  }

  Failure Response:

  {

  “message”: “username / email_ /password_ is not in request”,

  “status_code”: “400”,

  “error”: “Bad Request”

  },

{

“message”: “False”,

“status_code”: “500”,

“error”: “Internal Server Error”

}

- addFollower(username, usernameToFollow)

  addFollower adds the follower username against the username. That is when a particular username wants to add a follower their username is saved in usernameAPI and the person who the user wants to follow their username is saved in usernamToFollow in the follower table in database.

  HTTP Method: POST

  Status code: Success-201

  Failure- 409 Conflict

  Endpoint: /addFollower

  Sample JSON request:

  {

  “usernameAPI”: “ankita”,

  “UsernameFollowingAPI”: “om”

  }

  Success Response:

  {

  “message”: “om started following ankita”,

  “status_code”: “201”

  }

  Failure Response:

{

“message”: “username / email_ /password_ is not in request”,

“status_code”: “409”,

“error”: “Conflict”

}

- removeFollower(username, usernameToRemove)
- removeFollower deletes the follower username against the username. That is when a particular username wants to remove a follower their follower data is removed in follower table in database.

  HTTP Method: DELETE

  Status code: Success-200 OK

  Failure- 409 Conflict

  Endpoint: /addFollower

  Sample JSON request:

{

“usernameAPI”: “ankita”,

“UsernameFollowingAPI”: “om”

}

  Success Response:

{

“message”: “om started unfollowing ankita”,

“status_code”: “200”

}

  Failure Response:

{

“message”: “username / email_ /password_ is not in request”,

“status_code”: “409”,

“error”: “Conflict”

}

TimelineMicroservice.py:

- getUserTimeline(username)

This service shows what a user has posted recently. For a particular username, the tweet table will be checked for a tweet against that username in the database and then this service will give the recent tweets by that username.

HTTP Method: GET

Status code: Success-200 OK

Failure- 500 Internal server error, 405 Method not allowed, 404 Not Found

Status code: Success-201

Failure- 409 Conflict

Endpoint: /getUserTimeline

Sample JSON request:

{

“usernameAPI”: “ankita”,

}

Success Response:

{

“message”: “hola”,

“status_code”: “201”

}

Failure Response:

{

“message”: “username is not in request”,

“status_code”: “409”,

“error”: “Conflict”

}

- getPublicTimeline()

  This service will check the tweet table in the database and return all recent posts by all the users.

  HTTP Method: GET

  Status code: Success-201

  Endpoint: /getPublicTimeline

  Sample JSON request:

  Success Response:

  [

  {

  “message”: “[(hola), (hello)] ”

  “status_code”: “201”

  }

  ]

- getHomeTimeline(username)

  This service allows a user to view all the recent tweets from all the users that this user follows.

  HTTP Method: GET

  Status code: Success-200 OK

  Failure- 409 Conflict

  Endpoint: /getHomeTimeline

Sample JSON request:

{

“usernameAPI”: “ankita”,

}

Success Response:

{

“message”: “[(Namaste), (Sasriyakal)] ”,

“status_code”: “201”

}

Failure Response:

{

“message”: “username is not in request”,

“status_code”: “409”,

“error”: “Conflict”

}

- postTweet(username, text)

  This service allowed the user to post a new tweet. This tweet is stored in the tweets table against the username in the database.

  HTTP Method: POST

  Status code: Success-201

  Failure- 409 Conflict

  Endpoint: /postTweet

  Sample JSON request:

  {

  “usernameAPI”: “ankita”,

“tweetAPI”: “I am a girl!”

}

Success Response:

{

“message”: “I am a girl!”,

“status_code”: “201”

}

Failure Response:

{

“message”: “Username does not exist. Create a user first to proceed”,

“status_code”: “409”,

“error”: “Conflict”

}