Project #1 – Redis / Key-Value Stores
Due Date: Sunday, April 27th, 2025

Assume a Teams/Zoom-like environment but for physical meetings. In other words, the meetings take place in some location with lat and long (x,y) during a time interval (t1, t2) and has participants. Meeting Administrators (MA) create meetings. When a meeting starts, it has a chat room, i.e. participants can post messages.

Assume a relational database of:

- user (email, name, age, gender)
- meeting (meetingID, title, description, t1, t2, lat, long, participants)
  participants is of text datatype and has a comma-separated list of emails
- log (email, meetingID, timestamp, action)
  action is an integer and can be 1 (join_meeting), 2 (leave_meeting) or 3 (time_out)

Scheduler: a meeting becomes “active” when it is due (scan database every 1 minute to “activate”/”deactivate” meetings). When a meeting becomes active, it is inserted to Redis and all management is done in Redis.

Use Python/Java that connects to Redis to implement the following functions (endpoints):

* A user with email e and location (x,y) wants to see the active events near him/her that can join. Write a function that gets (e, x, y) and returns a list of meeting IDs that are active (i.e. in Redis), e is in the Participants list of the meeting, and (x,y) is within 100m of the location of the event.

* A user with email e joins a meeting with ID m. Write a a function that gets (e, m) and does all the appropriate bookkeeping; it also updates the Log table (join_meeting).

* A user with email e leaves a meeting with ID m. Write a a function that gets (e, m) and does all the appropriate bookkeeping; it also updates the Log table (leave_meeting).

* Given a meeting with ID m, write a function that returns a list of emails of the participants that have joined the meeting.

* Write a function that returns a list of meeting IDs of all active meetings.

* Given a meeting with ID m, write a function that ends the meeting: does all proper bookkeeping in Redis and inserts to Log for each remaining participant the action time_out.

* Given a user with email e and a text s, write a function that posts s to the chat room of the meeting that user e has joined.

* Given a meeting with ID m, write a function that returns a list of posts of m’s chat room in chronological order

* Given a user with email e, write a function that returns a list of all messages that user e has posted in the chat room of the meeting s/he has joined.

* Function: show for an active meeting and a user his/her chat messages

-----

It would be nice if there was an interface to:

(a) create/delete users and meetings
(b) show/test the functions