A separate file that lists all of the use cases and the queries executed by them (with brief explanation). This should be well organized and readable. It should be detailed enough to give readers a good idea of how your application works, without making them dig through all the code.

Features:

- View public info (Select interests to see groups that match said interest):
    -

Query	Explanation
SELECT * FROM interest	Select all the interests found on the interest table
SELECT group_name, description FROM about NATURAL JOIN a_group WHERE category = %s AND keyword = %s	Select all the groups that have the interest that the user selected

    -
- View public info (Events happening in the next three days)
    -

Query	Explanation
SELECT title, description, start_time, end_time, location_name, zipcode FROM `an_event` WHERE start_time <= DATE_ADD(CURRENT_TIMESTAMP , INTERVAL 3 DAY) AND start_time >= CURRENT_TIMESTAMP ORDER BY an_event.start_time ASC	Selects events that are happening in the next three days

- Login:
    -

Query	Explanation
SELECT * FROM member WHERE username = %s and password = md5(%s)	Checks to see if username and password exists in the database

- View my upcoming events:
    -

Query	Explanation
SELECT title, description, start_time, end_time, location_name, zipcode FROM sign_up NATURAL JOIN an_event WHERE username = %s AND start_time >= DATE_ADD(CURRENT_DATE, INTERVAL 1 DAY) AND start_time <= DATE_ADD(CURRENT_DATE, INTERVAL 3 DAY) ORDER BY an_event.start_time ASC	Selects all events that are happening in the next three days
SELECT title, description, start_time, end_time, location_name, zipcode FROM sign_up NATURAL JOIN an_event WHERE username = %s AND DATE(start_time) = CURRENT_DATE() ORDER BY an_event.start_time ASC	Select all events that are happening today

- Sign up:
    -

Query	Explanation
INSERT INTO sign_up (event_id, username) VALUES (%s, %s)	When the user signs up for the event, the new information is added onto the sign_up table

- Search for events of interest:
    -

Query	Explanation
SELECT * FROM about NATURAL JOIN organize NATURAL JOIN an_event WHERE (category, keyword) IN (SELECT category, keyword FROM interested_in WHERE username = %s) AND event_id NOT IN (SELECT event_id FROM sign_up WHERE username = %s)	Selects events that the user are interested in and that the user has not signed up for yet

- Create an event:
    -

Query	Explanation
SELECT * FROM belongs_to NATURAL JOIN a_group WHERE username = %s AND authorized = 1	Select groups that the user is authorized to create events for
INSERT INTO an_event (title, description, start_time, end_time, location_name, zipcode) VALUES (%s, %s, %s, %s, %s, %s)	After the user provides information of the new event, the an_event table is updated to reflect the new event
INSERT INTO organize (event_id, group_id) VALUES (%s, %s)	After the user provides information about the new event, the organize table is updated to reflect the new event
INSERT INTO sign_up (event_id, username) VALUES (%s, %s)	After the user provides information about the new event, the sign_up table is updated to reflect the new table

- Rate an event:
    -

Query	Explanation
SELECT * FROM an_event	Select all events that have been created
SELECT * FROM an_event NATURAL JOIN sign_up WHERE username = %s AND event_id = %s AND start_time <= CURDATE()	Select events that a user has signed up for, selected to rate, and have also already begun
UPDATE \`sign_up\` SET \`rating\` =%s WHERE \`event_id\` = %s AND \`username\` = %s	If a user has signed up for the event he/she wants to rate and has also already begun, sign_up table is updated to have a user/event rating.

- See average ratings:
    -

Query	Explanation
SELECT title, group_id, AVG(rating) as rate FROM sign_up NATURAL JOIN an_event as e NATURAL JOIN organize WHERE group_id IN	Select event name and its rating average, grouped by groups that a user belongs to and has occurred over the past three days.

(SELECT group_id FROM belongs_to WHERE username = %s) AND start_time >= DATE_SUB(CURRENT_DATE, INTERVAL 3 DAY) AND start_time < CURRENT_DATE GROUP BY e.event_id	

- See friends’ events:
    -

Query	Explanation
SELECT event_id, title, username FROM sign_up NATURAL JOIN an_event WHERE username IN (SELECT friend_of FROM friend WHERE friend_to = %s)	Select events and usernames where the username is a friend to the current user

- Logout:
    -

Query	Explanation
No queries needed	N/A

- Join group:
    -

Query	Explanation
SELECT DISTINCT group_id, group_name, description, creator FROM belongs_to NATURAL JOIN a_group WHERE group_id NOT IN (SELECT group_id FROM belongs_to WHERE username = %s)	Selects all the groups that the user is not already part of
INSERT INTO belongs_to (group_id, username, authorized) VALUES (%s, %s, %s)	Inserts the username and group ID in the belongs_to table in order to signify that the user is now part of the group. The user by default does not have authorization to post events (when they are joining)

- Create group:

-

Query	Explanation
INSERT INTO a_group (group_name, description, creator) VALUES (%s, %s, %s)	Inserts a new group into the a_group table based on the a parameters provided by the user
INSERT INTO belongs_to (group_id, username, authorized) VALUES (last_insert_id(), %s, %s)	The user is inserted into the belongs_to table as an authorized member for the group they just created

- Grant ability to create event:
  -

Query	Explanation
SELECT distinct group_id, group_name FROM a_group NATURAL JOIN belongs_to where username = %s and authorized = %s	Displays all the groups the user is an administrator of
SELECT firstname, lastname, username FROM member NATURAL JOIN belongs_to where belongs_to.group_id = %s and authorized != %s	Displays all the members in the group the user selects in the previous page
UPDATE belongs_to SET authorized = %s WHERE belongs_to.username = %s and belongs_to.group_id = %s	Changes the authorized attribute for the selected member

- Make friends:
  -

Query	Explanation
SELECT username from member where zipcode = (Select zipcode from member where username = %s) and username != %s and username not in (Select friend_to from friend where friend_of = %s)	Selects all the members from the members group who live in the same zipcode and are not already friends with the user
INSERT INTO friend (friend_of, friend_to) VALUES (%s, %s)	Inserts the selected member in the friend table

- Register:
    -

Query	Explanation
SELECT * FROM member WHERE username = %s	Checks if the username is already in the database
INSERT INTO member VALUES(%s, md5(%s), %s, %s, %s, %s)	Insert insert the new registered info onto the database

- Post in event:
    -

Query	Explanation
SELECT * FROM an_event NATURAL JOIN sign_up WHERE username = %s	Select events that a user has signed up for