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 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 | Select event name and its rating average, grouped by groups that a user belongs to and has occurred over the past three days. |

* 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 |