FE

Android/JRE

Application

Google Map API

① Create New Account (FR + User Versions)
- Send Valid New Account Info to BE
- Show Success + Errors

② Login + Logout (User + FR version)
- Send Valid Login info for comparison
- if success, receive and store login key
- if failure, no login key + login error
- periodically send login key to BE for verification
- if success, no change
- if failure, logout
- logout: delete local key + Tell BE
- timeout: automatic logout due to idling or network issues

③ Send user location to BE to end up in DB at some interval (user version)

④ Receive user locations from BE (FR version)

BE

Hosting System (Cloud or Local)

Python Environment

Flask Framework

System Application

MySQL Python Connector Library + Key Generator

① Create New Account Info (User + FR version)
- (Flask) Receive + Save New Account Info from FE
- (Python Conn. + MySQL) Send New Account Info to DB (User + FR version)
  - MySQL: (Table "User_Info" xor "FR_Info") New Row, Insert Info (w/ blank location Column)
  - Python Conn: Send MySQL code to DB to be executed
- (Flask/HTTP?/Py. Conn.) Try/catch + return error codes to FE

② Login + Logout (User + FR version)
- (Flask) Receive + Save FE Login attempt Info
- (Python conn. + MySQL) Retrieve + Save BE Login info with matching username
  - MySQL: (Table "User_Info" xor "FR_Info") Query Login info
  - Python Conn.: Retrieve + Save login info w/ matching username
- (Flask/HTTP?/KeyGen/MySQL/Py. Conn.) Verify Login info + grant key if verified (otherwise send error code)
  - Compare FE Login attempt info w/ BE Login info.
  - (Flask/HTTP?) If not equivalent, delete locally saved FE + BE login info and send error code to FE
  - (Flask/KeyGen/Py Conn./MySQL) If equivalent, delete locally saved FE + BE login info except DB username, Generate key, and send key to FE + DB
    - MySQL: (Table "User_key" xor "FR_Key") New Row, Insert key + username
    - PyConn: Send MySQL code to DB to be executed
    - Flask: Send key to FE; delete locally saved DB username
- (Flask/HTTP?/MySQL/PyConn.) Periodically verify the FE login state via key and username. Logout if invalid.
  - Flask: Receive and save FE key and username.
  - MySQL: (Table "User_key" xor "FR_key") Query given username and key (from the FE)
  - Pycon: Retrieve and save the DB version of the key and username
  - Compare the FE and DB versions of the key and username
  - (Flask/HTTP?) If not equivalent, delete all locally saved key and username data, Send
  - (Flask) If equivalent, delete all locally saved key and username data
- (Flask/HTTP?/MySQL/PyConn) FE-started logout
  - Flask: Receive Logout notice. Receive + Save FE username and key before FE key deletion. Send back okay for FE logout
  - MySQL: (Table "User_key" xor "FR_key") Query + delete row containg DB key/username equivalent to FE version
  - PyConn.: Send MySQL Code to DB to be executed
  - Delete all locally stored key/username data
  - (Flask/HTTP?) Send error codes to FE

③ Sending user location to DB (user version) [requires login state]
- (Flask) Receive emergency notice from FE. Receive + store location data and username.
- (MySQL/PyConn.) Overwrite location data in DB
  - MySQL: (Table "User Info") Overwrite "location" column of row of the username matching the FE username
  - PyConn: Send MySQL code to DB to be executed
- repeat every 30 seconds (for large scale) or 5 seconds (for small scale)

④ retrieving user location from DB (FR version) [requires login state]
- (Flask) Receive Ready notice from FE. Receive and Store location data and username.
- Find conversion for X number of miles to magnitude of longitude/latitude as bounds for a query search
- (MySQL/PyConn.) Find all victims within a specified radius and save their data in the system as multiple arrays
  - MySQL: (Table "User_Info") Query identity, medical, location (excluding login data)
  - PyConn.: Retrieve and save user data
- (Flask) Pass user data to FE. Once passed delete all locally stored user data.

- (Flask) Pass user data to FE. Once passed delete all locally stored user data.

Friday, November 2, 2018 4:19 PM