By: Jordan Lederer and Saad Saeed

BE DB-interaction core model:

(Brackets indicate places for user input and are not syntax)

1. Create New Account

-FE sends account info to BE (FE has already completed form validation)

-BE locally saves account info

-BE checks username uniqueness

-- MySQL: “SELECT [[username]] FROM [User\_Info] WHERE [[[username] =[“[...]”]]];”

---DB Responses: “No Results”/ “[...]”

-- If DB response is “[...]”, stop and return to FE a “username is already in use” error

-- If DB response is “No Results”, continue

-BE pushes account info to DB and return success message to FE

--MySQL: “INSERT INTO [User\_Info] ([column name],...) VALUES ([value],...);”

-BE deletes locally saved account info

2. Initial Authentication

-FE sends username and password (login info) to BE

-BE locally saves username and password (login info)

-BE checks login info to either give or deny login status

--BE checks for similar login info in DB

---MySQL: “SELECT [[username], [password]] FROM [User\_Info] WHERE [[[username]=”[...]”]] AND [[password]=“[…]”]];”

----DB Response: “[...]” / “No Results”

---Responses (BE-> FE): “Invalid Credentials” (if DB response is “No Results”) / “User Found, Login Successful” (if DB response is “[...]”)

--If response is “Invalid credentials”, BE sends a login error to FE

--If response is “User Found, Login successful”, BE calls key generator and sends key to be saved in FE and DB

---MySQL: “INSERT INTO [User\_Key] (username, key) VALUES (“[...]”,

“[...]”);”

-BE deletes locally saved login info

3. Periodic Key Check

-Periodically, BE commands FE to send the key and username in use by FE. BE stores this locally temporarily.

- BE searches the DB key table for an identical set of key and username

--MySQL: “SELECT username, key FROM [User\_Key] WHERE [[[username] = “[…]”] AND [[key]=”[…]”]];”

---DB Responses: “No Results” / “[…]”

--Responses (BE): “Re-login required” ( if DB Response is “No Result”) / “Valid login state, no action required” (if DB response is “[…]”)

--If response is “Valid login state, no action required”, no change occurs to the FE or DB

--If response is “Login required”, BE sends command to FE to automatically initiate logout

-BE deletes locally saved username and key after this periodic check

4. Logout

-Wait for either user to initiate logout process from the FE or for BE to send a command to FE to automatically initiate logout

-FE initiates logout process

-FE sends the username and key to BE

-FE deletes all local username and key information

-BE deletes rows with the same username info from the key table

--MySQL: “DELETE FROM [User\_Key] WHERE [[[username] = “[…]”] AND [[key] = “[...]”]];”

-BE deletes locally saved username and key

5. Store User Location (User version only)

-Authentication required

-User activates emergency mode

-FE sends username to BE. BE locally stores this username.

-Loop begins, ending when the user deactivates emergency mode:

--FE sends location data to BE at some time interval. BE locally saves the location data.

--BE sends location data to DB as data is received

---MySQL: “UPDATE [User\_Info] SET location\_longitude = [...] WHERE [[[username] = “[...]”]];”

---MySQL: “UPDATE [User\_Info] SET location\_latitude = [...] WHERE [[[username] = “[...]”]];”

-Emergency mode deactivated on FE, loop ends

-After some time period after emergency mode is deactivated, location data remaining in DB is deleted

---MySQL: “UPDATE [User\_Info] SET location\_longitude = [NULL] WHERE [[[username] = “[...]”]];”

---MySQL: “UPDATE [User\_Info] SET location\_longitude = [NULL] WHERE [[[username] = “[...]”]];”

-BE deletes locally saved username and location data

6. Retrieve user location (FR version only)

-Authentication required

-FR activates retrieval mode

-FR FE sends FR username (with, only in a real-world use-case, the FR FE location and additional location bounds, based on FR FE location, to be used to narrow query results to nearby victims)

-Loop begins, ending with FR deactivating retrieval mode

--FR FE sends username to BE

--BE retrieves and locally saves identity (excluding login information), medical, and location data (that of every user in demo , and that of nearby users in a real-world scenario)

--- MySQL: “SELECT [[[identity\_columns], ...], [[medical\_columns], ...], [[location\_columns], ...]] FROM [User\_Info];” (replace the “;” at the end with “WHERE (location\_longitude <= ([FR + LG\_bound])) AND (location\_latitude <= ([FR + LT\_bound])) AND (location\_longitude >= ([FR – LG\_bound])) AND (location\_latitude >= ([FR – LT\_bound]));” for real-world scenarios only)

---DB Responses: “[...]”

--BE sends info to FR FE

-(Only in real-world scenarios: FR FE stores list of nearby users and their info initially created by BE so BE doesn’t have to repetitively retrieve the unchanging medical and identity info from the DB and keep sending that unchanging info to FR FE. Victims are added to a FR FE list as they enter specified range of FR and are removed from this FR FE list as they leave range of FR. The FR FE list is deleted directly after retrieval mode ends along with all locally saved location data in FR FE.)

-Loop ends as retrieval mode is deactivated on FR FE

-FR FE deletes all locally saved victim info

-BE deletes locally saved username along with all other locally saved victim info

-------------------------------------------------------------------------------------------------------------------------------

DB-building MySQL syntax to be used: (Brackets indicate places for user input and are not syntax)

- “CREATE DATABASE [DB\_name];”

- “USE [DB\_Name];”  
- “CREATE TABLE [Table\_Name] ( [[column\_name] [column\_datatype]([column\_datatype\_size]) NOT NULL],[[column\_name] [column\_datatype]([column\_datatype\_size])],..., PRIMARY KEY ([[column\_name)]);”

- “CREATE TABLE [Table\_Name] ( [[column\_name] [column\_datatype]([column\_datatype\_size]) NOT NULL],[[column\_name] [column\_datatype]([column\_datatype\_size])],..., CONSTRAINT [constraint\_name] PRIMARY KEY ([[column\_name)], [column name], ...]);”

- “ALTER TABLE [table\_name] ADD [column\_name] [column\_datatype]([column\_datatype\_size]) NOT NULL;”

- “ALTER TABLE [table\_name] DROP [column\_name];”  
- “ALTER TABLE [table\_name] CHANGE [column\_name] [column\_datatype]([column\_datatype\_size]) NOT NULL;”

- “ALTER TABLE [table\_name] CHANGE [column\_name] [column\_datatype]([column\_datatype\_size]);”  
- “ALTER TABLE [table\_name] MODIFY [column\_name] [column\_datatype]([column\_datatype\_size]) NOT NULL;”

-“ALTER TABLE [table\_name] MODIFY [column\_name] [column\_datatype]([column\_datatype\_size]);”

- “INSERT INTO [Table\_Name] ([[column\_name], ...]) VALUES ([[value], ...]);”

- “SELECT [ \* or [[column name], ...]] FROM [Table\_Name] WHERE [[[column\_name] = [...]] AND ... OR....];”

- “DELETE FROM [Table\_Name] WHERE [[[column\_name] = [...]] AND ... OR ...];”

- “ALTER TABLE [Table\_Name] ADD CONSTRAINT [constraint\_name] PRIMARY KEY ([column\_name])”;