Unity

First Steps:

Add “namespace Polybius {“ to the top of any script, after usings and before the class declaration. This will allow you to use a “PolybiusManager.function();”, in any part of the script, without having to connect PolybiusManager via a public gameobject field or whatnot. PolybiusManager will provide the core data structures and associated functions (updating data from server, sending data to server, etc…). When adding to this data structure, please only add if the data has to interface with the server or the scope expands over the entire app (such as user settings).

Data Structure

PolybiusManager:

Variables:

* username
  + string
  + Contains the user’s username
* email
  + string
  + Contains the user’s email
* userID
  + int
  + Contains user’s unique ID
* messages
  + List
  + Contains all unread messages from the server

Methods:

* getMessages()
  + public static void
  + updates messages list with unread messages from server
* clearMessages()
  + public static void
  + clears message list, updates server if need be

Database

The Users database contains all information relevant to the users of Polybius. In this database there are two different tables that hold user data and messages. Below is the variables that are in the tables and their associated order.

Users Database

Table userdata:

* username
  + Type: Text
  + This column is a text field that contain the user’s username.
* password
  + Type: Text
  + This text field is the password of the user. This gets encrypted through the server transfer.
* email
  + Type: Text
  + This column is a text field that contain the user’s email.
* dob
  + Type: Date
  + This corresponds to the date of birth of a given user.
* id
  + Type: Integer
  + This corresponds to the entry of a user in the database. This is automatically calculated when the user is initially inserted into the database and does not need to be set.
* isonline
  + Type: Tinyint/Boolean
  + This value is set to True when a person logs into their account and set to false when they logout.
* currentgame
  + Type: int
  + This identifies which games a user is currently playing. If the value is zero, then the user is not currently in any game mode.
* friends
  + Type: Text
  + This is a text field that hold a list of the name of the user’s friends.
* statistics
  + Type: Text
  + This text fields holds pre-calculated statistics in the form of text.
* notifications
  + Type: Text
  + Holds notifications of type text that has the current notifications that need to be sent to the user.
* Private
  + Type: Int
  + Holds a value that determines if a user has a profile that is public. 1 is private 0 is public.

Table msgs:

* id
  + Type: int
  + This is the id of each message that gets sent. This id is used to see the order of the messages that each user has received.
* sender
  + Type: Text
  + This is the username of the sender of the corresponding message
* receiver
  + Type: Text
  + This is the username of the receiver of the corresponding message
* time
  + Type: Timestamp
  + The set time that a message is sent. This is set automatically when the message is sent to the database.
* message
  + Type: Text
  + This contains the contents of what the sender sends to the receiver. This is in the format of text.

Table rooms:

* roomName
  + Type: Text
  + Contains the player set name of the current game room.
* roomType
  + Type: Int
  + Contains the type of minigame being played.
* roomID
  + Type: Int
  + Unique ID of each room that is generated on creation.
* usersInRoom
  + Type: Text
  + A string of text that contains all the usernames of the players currently connected lobby.
* gameType
  + Type: Text
  + A string of text that contains the name of the minigame being played.
* latcord
  + Type: Float
  + Contains the latitude coordinate of the location of the room
* longcord
  + Type: float
  + Contains the longitude coordinate of where the room is located

Reporting Database

Table userfeedback:

* id
  + Type: Int
  + This is an automatically set int value that corresponds to the given entry into the feedback table.
* user
  + Type:text
  + This is a set value that is the name of the user who is sending in feedback.
* feedback
  + Type:text
  + This is the written response from the user detailing their issues that we should take care of.

Table userreport:

* id
  + Type: Int
  + This is an automatically set int value that corresponds to the given entry into the report table
* reporteduser
  + Type:text
  + This is the name of the user that is being reported.
* Reporter
  + Type:text
  + This is the name of the user that is doing the reporting.
* Report
  + Type:text
  + This is the field where the reason for the report is being made.