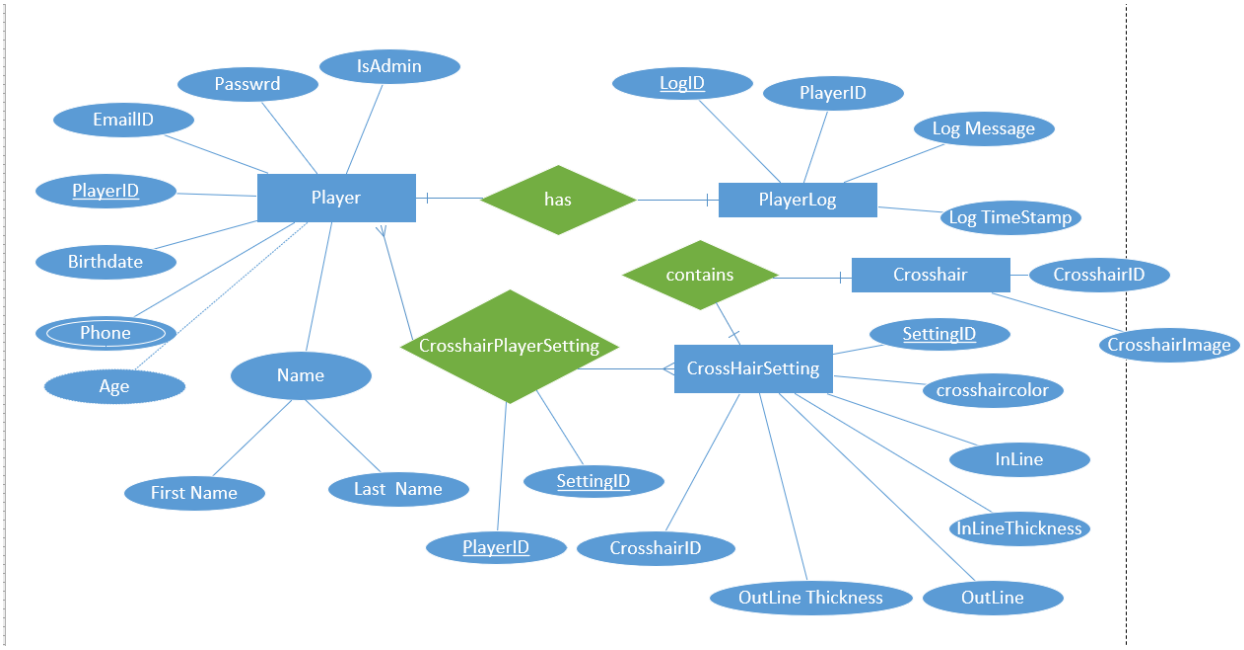
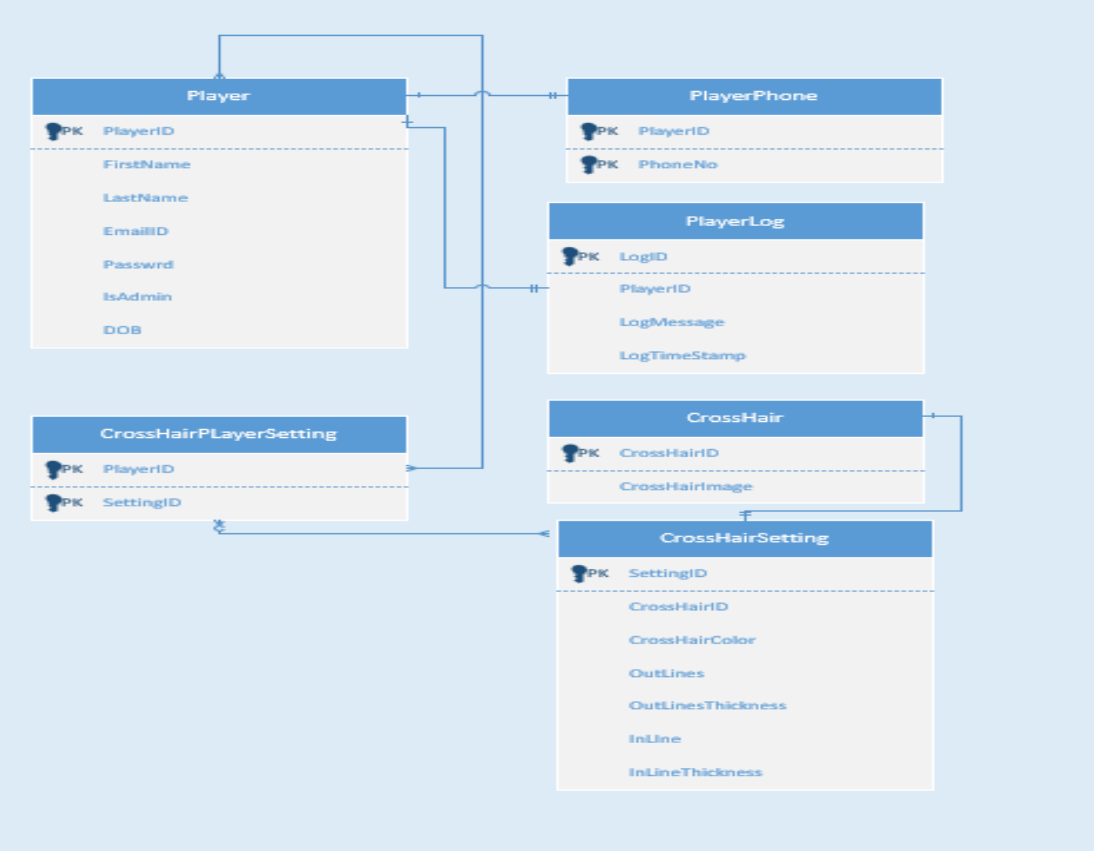
Valorant Database Management System

ER Diagram



Relation Schema Diagram



Tables and Stored Procedure and Triggers

```
mysql> use valorantcrosshair
Database changed
mysql> show tables
    -> ;
+----------------------------+
| Tables_in_valorantcrosshair |
+----------------------------+
| crosshair                  |
| crosshairplayersetting     |
| crosshairsetting           |
| player                     |
| playerlog                  |
| playerphone                |
+----------------------------+
```

```
mysql> desc player;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| PlayerID  | int          | NO   | PRI | NULL    | auto_increment |
| FirstName | varchar(255) | NO   |     | NULL    |                |
| LastName  | varchar(255) | NO   |     | NULL    |                |
| EmailID   | varchar(255) | NO   |     | NULL    |                |
| DOB       | date         | NO   |     | NULL    |                |
| Passwrd   | varchar(255) | NO   |     | NULL    |                |
| ISAdmin   | tinyint(1)   | NO   |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
7 rows in set (0.02 sec)
```

```
mysql> desc PlayerPhone
    -> ;
+----------+--------+------+-----+---------+-------+
| Field    | Type   | Null | Key | Default | Extra |
+----------+--------+------+-----+---------+-------+
| PlayerID | int    | NO   | PRI | NULL    |       |
| PhoneNo  | bigint | NO   | PRI | NULL    |       |
+----------+--------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
mysql> desc PlayerLog;
+--------------+--------------+------+-----+-------------------+-------------------+
| Field        | Type         | Null | Key | Default           | Extra             |
+--------------+--------------+------+-----+-------------------+-------------------+
| LogID        | int          | NO   | PRI | NULL              | auto_increment    |
| PlayerID     | int          | YES  | MUL | NULL              |                   |
| LogMessage   | varchar(255) | NO   |     | NULL              |                   |
| LogTimestamp | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+--------------+--------------+------+-----+-------------------+-------------------+
4 rows in set (0.00 sec)
```

```
mysql> desc crosshair;
+----------------+------------+------+-----+---------+----------------+
| Field          | Type       | Null | Key | Default | Extra          |
+----------------+------------+------+-----+---------+----------------+
| CrosshairID    | int        | NO   | PRI | NULL    | auto_increment |
| CrosshairImage | mediumblob | NO   |     | NULL    |                |
+----------------+------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)
```

```
mysql> desc crosshairsetting;
+---------------------+-------------+------+-----+---------+----------------+
| Field               | Type        | Null | Key | Default | Extra          |
+---------------------+-------------+------+-----+---------+----------------+
| SettingID           | int         | NO   | PRI | NULL    | auto_increment |
| CrosshairID         | int         | YES  | MUL | NULL    |                |
| CrosshairColor      | varchar(10) | NO   |     | NULL    |                |
| OutLines            | tinyint(1)  | YES  |     | NULL    |                |
| OutLinesThickness   | tinyint     | YES  |     | NULL    |                |
| InnerLines          | tinyint(1)  | YES  |     | NULL    |                |
| InnerLinesThickness | tinyint     | YES  |     | NULL    |                |
+---------------------+-------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

```
mysql> desc crosshairplayersetting;
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| SettingID | int  | NO   | PRI | NULL    |       |
| PlayerID  | int  | NO   | PRI | NULL    |       |
+-----------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
DELIMITER //

CREATE PROCEDURE GetPlayerCredentialsByEmail(IN p_email VARCHAR(255))
BEGIN
    SELECT PlayerID, EmailID, passwrd,ISAdmin,FirstName
    FROM player
    WHERE EmailID = p_email;
END //

DELIMITER ;
```

```
DELIMITER //

CREATE PROCEDURE InsertCrosshairAndSetting(
    IN p_CrosshairImage MEDIUMBLOB,
    IN p_CrosshairColor VARCHAR(10),
    IN p_OutLines TINYINT,
    IN p_OutLinesThickness TINYINT,
    IN p_InnerLines TINYINT,
    IN p_InnerLinesThickness TINYINT
)
BEGIN
    DECLARE v_CrosshairID INT;

    -- Insert data into Crosshair table
    INSERT INTO Crosshair (CrosshairImage) VALUES (p_CrosshairImage);
    SET v_CrosshairID = LAST_INSERT_ID();

    -- Insert data into CrosshairSetting table
    INSERT INTO CrosshairSetting (CrosshairColor, OutLines, OutLinesThickness, InnerLines, InnerLinesThickness, CrosshairID)
    VALUES (p_CrosshairColor, p_OutLines, p_OutLinesThickness, p_InnerLines, p_InnerLinesThickness, v_CrosshairID);
END //

DELIMITER ;
```

```
--Create Player and Phone NUmber during Signin
DELIMITER //
CREATE PROCEDURE AddPlayerAndPhones(
    IN p_FirstName VARCHAR(255),
    IN p_LastName VARCHAR(255),
    IN p_EmailID VARCHAR(255),
    IN p_DOB DATE,
    IN p_Passwrd VARCHAR(255),
    IN p_ISAdmin TINYINT,
    IN p_PhoneNumbersCSV VARCHAR(255)
)
BEGIN
    DECLARE newPlayerID INT;


-- Create a temporary table to store phone numbers
    CREATE TEMPORARY TABLE IF NOT EXISTS TempPhoneNumbers (
     PlayerID INT,
        PhoneNo BIGINT
 );
```

```sql
 CREATE TEMPORARY TABLE IF NOT EXISTS Numbers (
    digit INT
);

INSERT INTO Numbers (digit) VALUES (0), (1), (2), (3), (4), (5);

-- Insert data into the Player table
INSERT INTO Player (FirstName, LastName, EmailID, DOB, Passwrd, ISAdmin)
VALUES (p_FirstName, p_LastName, p_EmailID, p_DOB, p_Passwrd, p_ISAdmin);

-- Get the last inserted PlayerID
SET @newPlayerID = LAST_INSERT_ID();

INSERT INTO TempPhoneNumbers (PlayerID, PhoneNo)
SELECT @newPlayerID AS PlayerID,
       CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(p_PhoneNumbersCSV, ',', Numbers.digit+1), ',', -1) AS SIGNED) AS PhoneNo
FROM Numbers
WHERE Numbers.digit < (LENGTH(p_PhoneNumbersCSV) - LENGTH(REPLACE(p_PhoneNumbersCSV, ',', '')) + 1);

-- Insert data into the PlayerPhone table from the temporary table
INSERT INTO PlayerPhone (PlayerID, PhoneNo)
SELECT PlayerID, PhoneNo FROM TempPhoneNumbers;


 -- Drop the temporary table
 DROP TEMPORARY TABLE IF EXISTS TempPhoneNumbers;

 DROP TEMPORARY TABLE IF EXISTS Numbers;

 END //
 DELIMITER ;


 DELIMITER //
 CREATE TRIGGER ChangeToAdminLog
 AFTER UPDATE ON Player
 FOR EACH ROW
 BEGIN
     -- Check if the ISAdmin column is updated to 1 (admin)
     IF NEW.ISAdmin = 1 AND OLD.ISAdmin = 0 THEN
         -- Insert a log entry into PlayerLog
         INSERT INTO PlayerLog (PlayerID, LogMessage)
         VALUES (NEW.PlayerID, 'Player changed to admin');
     END IF;
 END //
 DELIMITER ;
```

UI Code – main.py

```
import streamlit as st
from datetime import datetime, timedelta
import bcrypt
import mysql.connector
from mysql.connector import Error
from io import BytesIO
from PIL import Image
import pandas as pd
# Function to create MySQL connection
def create_connection():
    return  mysql.connector.connect(
        host='localhost',
        user='root',
        password='Nitish080603',
        database='ValorantCrossHair'
)
#sign in
def authenticate(email, password):
    try:
        connection = create_connection()
        if connection:
            cursor = connection.cursor()
            cursor.callproc("GetPlayerCredentialsByEmail", (email,))
            # Fetch the result set from the stored procedure
            result = []
            for result_cursor in cursor.stored_results():
                result.extend(result_cursor.fetchall())
            if result:
```

```
            hashed_password = result[0][2]  # Assuming hashed password is at index 4, adjust accordingly
            if bcrypt.checkpw(password.encode('utf-8'), hashed_password.encode('utf-8')):
                st.session_state.PlayerID = result[0][0]
                st.session_state.FirstName = result[0][4]
                if result[0][3] ==  1:
                    st.session_state.IsAdmin = True
                else:
                    st.session_state.IsAdmin = False
                return True
            else:
                st.error("Authentication Failed - Try again")
                return False
        else:
            st.error("Not able to get user credentials. Try later")
            return False


    except Error as e:
        st.error(f"Error authenticating user: {e}")
        return False
    finally:
        cursor.close()
        connection.close()
#sign in
#sign up
def hash_password(password):
    # Generate a salt and hash the password
    salt = bcrypt.gensalt()
    hashed_password = bcrypt.hashpw(password.encode('utf-8'), salt)
    return hashed_password
```

```
def calculate_age(birthdate):
    today = datetime.today()
    birthdate = datetime.strptime(birthdate, "%Y-%m-%d")
    age = today.year - birthdate.year - ((today.month, today.day) < (birthdate.month, birthdate.day))
    return age
def insert_player_data(first_name, last_name, email, dob, hashed_password, is_admin, phone_string):
    try:
        # Establish a connection to the MySQL server
        connection = create_connection()

        # Create a MySQL cursor
        cursor = connection.cursor()

        # Insert data into the table
        cursor.callproc("AddPlayerAndPhones", (first_name, last_name, email, dob, hashed_password,
is_admin, phone_string))

        # Commit the transaction
        connection.commit()

        st.success("Player data inserted successfully!")

    except Exception as e:
        st.error(f"Error: {e}")

    finally:
        # Close the cursor and connection
        cursor.close()
        connection.close()
```

```
#sign up
#Dual sign in and register option
def main():
  st.title("Valorant Crosshair Setting Database")
  selected_option = st.radio("Choose an option", ["Sign In", "Sign Up"])
  if selected_option == "Sign In":
    st.subheader("Sign In Form")
    email = st.text_input("Email")
    password = st.text_input("Password", type="password")

    if st.button("Sign In"):
      if authenticate(email, password):
        st.session_state.is_authenticated = True
        st.experimental_rerun()
  elif selected_option == "Sign Up":
    phone_numbers = []
    st.subheader("Sign Up Form")
    fname = st.text_input("First Name")
    lname = st.text_input("Last Name")
    email = st.text_input("Email")
    password = st.text_input("Password", type="password")
    if password:
      # Hash the entered password
      hashed_password = hash_password(password)

    # Set the maximum allowed date to today
    max_date = datetime.today()

    # Set the minimum allowed date to 100 years ago from today
```

```
    min_date = max_date - timedelta(days=365 * 100)

    # Display the date input field with the specified range
    dob = st.date_input("Date of Birth:", min_value=min_date, max_value=max_date)

    if dob:
        age = calculate_age(str(dob))
        st.success(f"Age: {age} years")

    # Use a slider to control the number of phone number input fields
    num_phone_numbers = st.slider("Number of Phone Numbers", min_value=1, max_value=5, value=1)

    # Create a list to store phone numbers
    phone_numbers = []

    # Display phone number input fields based on the slider value
    for i in range(num_phone_numbers):
        phone_numbers.append(st.text_input(f"Phone Number {i + 1}:"))
    phone_string = ", ".join(phone_numbers)
    is_admin = st.checkbox("Admin")
    if st.button("Sign Up"):
        # Implement your sign-up logic here
        insert_player_data(fname, lname, email, dob, hashed_password, is_admin, phone_string)
        st.success(f"Sign Up clicked! First Name: {fname}, Last Name : {lname},  Email: {email},
                Password: {hashed_password}, admin: {is_admin}, phone: {phone_string}")
#Dual sign in and register option
```

```
#option 2  - admin -- start
def update_admin_privilege_sql(player_id, is_admin):
    try:
        connection = create_connection()
        cursor = connection.cursor()
        query = "UPDATE Player SET ISAdmin = %s WHERE PlayerID = %s"
        cursor.execute(query, (is_admin, player_id))
        connection.commit()
        st.success("Admin privilege updated successfully.")
    except Error as e:
        st.error(f"Error: {e}")
    finally:

            cursor.close()
            connection.close()
def get_all_players():
    try:
        connection = create_connection()
        cursor = connection.cursor(dictionary=True)
        query = "SELECT PlayerID, FirstName, LastName FROM Player where IsAdmin = 0"
        cursor.execute(query)
        players = cursor.fetchall()
        return players
    except Error as e:
        st.error(f"Error: {e}")
    finally:

            cursor.close()
            connection.close()
```

```
def update_access_previlage():


    # Get all players
        players = get_all_players()
        if players :
        # Create a dropdown to select a player
            selected_player = st.selectbox("Select a player:", [f"{player['PlayerID']}: {player['FirstName']}
{player['LastName']}" for player in players])


        # Extract PlayerID from the selected player string
            player_id = int(selected_player.split(":")[0])


        # Create a checkbox to update admin privilege
            is_admin = st.checkbox("Grant Admin Privilege")


        # Button to update admin privilege
            if st.button("Update Admin Privilege"):
                update_admin_privilege_sql(player_id, int(is_admin))
        else:
            st.error("No Player to change privilage")
#option 2  - admin -- end


#option 1 - admin --start
def create_crosshair_setting():
    # Crosshair Input
        st.header('Crosshair Information')
        crosshair_image = st.file_uploader('Upload Crosshair Image', type=['png', 'jpg', 'jpeg'])
        if crosshair_image is not None:
            crosshair_image_content = crosshair_image.read()
```

```
    st.image(crosshair_image, caption='Uploaded Crosshair Image', width=100)

# Crosshair Setting Input
st.header('Crosshair Setting Information')
color_options = ["Red", "Green"]
crosshair_color = st.selectbox("Select a color", color_options)
outlines = st.checkbox('Enable Outlines')
outlines_thickness = st.slider('Outlines Thickness', min_value=0, max_value=2, step=1)
inner_lines = st.checkbox('Enable Inner Lines')
inner_lines_thickness = st.slider('Inner Lines Thickness', min_value=0, max_value=2, step=1)

# Button to save data
if st.button('Save Data'):
    # Connect to MySQL
    connection = create_connection()
    cursor = connection.cursor()

    try:
        # Call the stored procedure
        cursor.callproc('InsertCrosshairAndSetting', (crosshair_image_content, crosshair_color,
int(outlines), outlines_thickness, int(inner_lines), inner_lines_thickness))

        # Commit the changes
        connection.commit()

        st.success('Data saved successfully!')
    except mysql.connector.Error as err:
        st.error(f"Error: {err}")
```

```
        finally:
            # Close the connection
            cursor.close()
            connection.close()
#option 1 - admin --start


#option 4 -  Delete crosshair setting --start
# Function to fetch crosshair settings with images from the database
def get_crosshair_settings():
    try:
        connection = create_connection()


        if connection.is_connected():
            cursor = connection.cursor(dictionary=True)
            cursor.execute("SELECT s.SettingID, s.CrosshairColor, s.OutLines, s.OutLinesThickness, s.InnerLines,
"
                           "s.InnerLinesThickness, c.CrosshairImage "
                           "FROM CrosshairSetting s "
                           "JOIN Crosshair c ON s.CrosshairID = c.CrosshairID")
            settings = cursor.fetchall()
            return settings
    except Error as e:
        st.error(f"Error: {e}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
# Function to delete a crosshair setting based on SettingID
def delete_crosshair_setting(setting_id):
```

```
    try:
        connection = create_connection()

        if connection.is_connected():
            cursor = connection.cursor()
            cursor.execute("DELETE FROM CrosshairSetting WHERE SettingID = %s", (setting_id,))
            connection.commit()
            st.success("Crosshair setting deleted successfully!")
    except Error as e:
        st.error(f"Error: {e}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
#starting
def delete_crosshair_Setting_option():
    # Fetch crosshair settings with images
    settings = get_crosshair_settings()

    # Display crosshair settings with images in a table
    for setting in settings:
        # Convert binary image data to Image object
        image = Image.open(BytesIO(setting['CrosshairImage']))
        # Display the image
        st.image(image, caption=f"SettingID: {setting['SettingID']}", width=100)

    # Dropdown to select a row for deletion
    selected_setting_id = st.selectbox("Select a row to delete", [setting['SettingID'] for setting in settings])
```

```
    # Delete button
    if st.button("Delete Selected Row"):
        delete_crosshair_setting(selected_setting_id)
        # Reload the page after deletion to reflect the changes
        st.experimental_rerun()
#option 4 -  Delete crosshair setting --end


#option 5
# Function to get player information for a given SettingID using a subquery
def get_players_for_setting(setting_id):
    connection = create_connection()
    cursor = connection.cursor(dictionary=True)
    cursor.execute(
        f"SELECT * FROM Player "
        f"WHERE PlayerID IN (SELECT PlayerID FROM CrosshairPlayerSetting WHERE SettingID =
{setting_id});"
    )
    players = cursor.fetchall()
    cursor.close()
    return players
def display_player_crosshair_setting():



    connection = create_connection()
    # Get distinct SettingIDs from CrosshairSetting table
    cursor = connection.cursor(dictionary=True)
    cursor.execute("SELECT DISTINCT SettingID FROM CrosshairSetting;")
    setting_ids = [row['SettingID'] for row in cursor.fetchall()]
    cursor.close()
```

```
    # Select a SettingID from the dropdown
    selected_setting_id = st.selectbox("Select a SettingID", setting_ids)

    # Display players for the selected SettingID
    if st.button("Show Players"):
        players = get_players_for_setting(selected_setting_id)
        st.subheader(f"Players for SettingID: {selected_setting_id}")
        if players:
            st.table(players)

        else:
            st.warning("No players found for the selected SettingID.")
#option 5
#option 6
# Function to get player count for each SettingID
def get_player_counts_by_setting():
    connection = create_connection()
    cursor = connection.cursor(dictionary=True)
    cursor.execute(
        "SELECT SettingID, COUNT(DISTINCT PlayerID) AS PlayerCount "
        "FROM CrosshairPlayerSetting GROUP BY SettingID;"
    )
    player_counts = {row['SettingID']: row['PlayerCount'] for row in cursor.fetchall()}
    cursor.close()
    return player_counts
def display_Cross_hair_setting_usage():
    # Get player counts for each SettingID
   player_counts = get_player_counts_by_setting()
```

```
  # Display player counts in a table
  st.table(
      data={
          'SettingID': list(player_counts.keys()),
          'PlayerCount': list(player_counts.values())
      }
  )
#option 6
#option 4
# Function to get all non-admin players
def get_non_admin_players():
    connection = create_connection()
    cursor = connection.cursor(dictionary=True)
    cursor.execute("SELECT PlayerID, FirstName, LastName FROM Player WHERE ISAdmin = 0;")
    players = cursor.fetchall()
    cursor.close()
    return players
def delete_player(player_id):
    connection = create_connection()
    cursor = connection.cursor()
    cursor.execute("DELETE FROM Player WHERE PlayerID = %s;", (player_id,))
    connection.commit()
    cursor.close()
def delete_non_admin_player():
    # Get all non-admin players
    players = get_non_admin_players()

    # Display non-admin players in a table
```

```
    st.table(players)

    # Dropdown to select a player for deletion
    selected_player_id = st.selectbox("Select a player to delete", [player['PlayerID'] for player in players])

    # Delete button
    if st.button("Delete Player"):
        delete_player(selected_player_id)
        st.success(f"Player with PlayerID {selected_player_id} deleted successfully.")
#option 4
#log out
def logout():
    # Reset the authentication status
    st.session_state.is_authenticated = False
    st.session_state.IsAdmin = False
    st.session_state.PlayerID = 0
    st.session_state.FirstName =''
    # Redirect to the login page
    st.experimental_rerun()
    main()
#log out
#admin menu - start
def admin_page(PlayerID,FirstName):
    st.title(f"Welcome Admin {FirstName}")

    # Add menu bar items for the authenticated page
    menu_option = st.sidebar.radio("Menu", ["Create Cross hair Setting", "Update Player Access Privilage", "Delete Non Admin Player", "Delete cross hair settings","Display Player Cross hair Setting","Display Cross hair setting usage","Log out"])
```

```
    if menu_option == "Create Cross hair Setting":
        create_crosshair_setting()
    elif menu_option == "Update Player Access Privilage":
       update_access_previlage()
    elif menu_option == "Delete Non Admin Player":
         delete_non_admin_player()
    elif menu_option == "Delete cross hair settings":
         delete_crosshair_Setting_option()
    elif menu_option == "Display Player Cross hair Setting":
         display_player_crosshair_setting()
    elif menu_option == "Display Cross hair setting usage":
         display_Cross_hair_setting_usage()
    elif menu_option == "Log out":
         logout()
#admin menu - end
#user option1 - start
def assign_crosshair_setting_sql(setting_id,PlayerID):
    try:
        connection = create_connection()
        if connection.is_connected():
            cursor = connection.cursor()
            query = f"INSERT INTO CrosshairPlayerSetting (SettingID, PlayerID) VALUES ({setting_id},
{PlayerID})"
            cursor.execute(query)
            connection.commit()
            st.success("Crosshair setting assigned successfully!")
    except Error as e:
        st.error(f"Error: {e}")
    finally:
```

```
        if connection.is_connected():
            cursor.close()
            connection.close()
def assign_crosshair_setting(PlayerID):
    # Fetch crosshair settings with images
    settings = get_crosshair_settings()
    # Display crosshair settings with images in a table
    for setting in settings:
        # Convert binary image data to Image object
        image = Image.open(BytesIO(setting['CrosshairImage']))
        # Display the image
        st.image(image, caption=f"SettingID: {setting['SettingID']}", width=100)

    # Dropdown to select a row for deletion
    selected_setting_id = st.selectbox("Assign crosshair to me ", [setting['SettingID'] for setting in settings])
    # Delete button
    if st.button("Assign crosshair setting"):
        assign_crosshair_setting_sql(selected_setting_id,PlayerID)

#user option 1 - end
#user option2 - start
def get_player_settings(PlayerID):
    try:
        connection = create_connection()

        if connection.is_connected():
            cursor = connection.cursor()

            # Query to get crosshair settings for the selected player
```

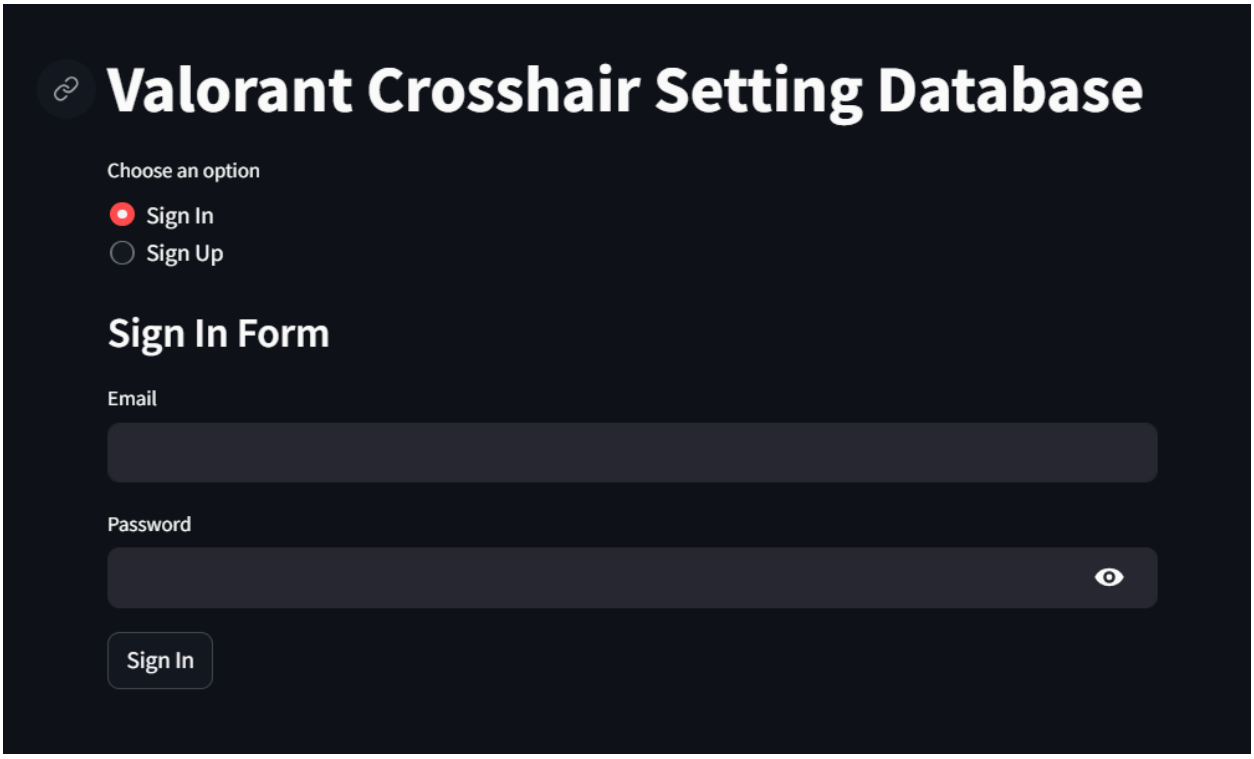
```
        query = f"SELECT cs.SettingID, cs.CrosshairColor, cs.OutLines, cs.OutLinesThickness, " \
            f"cs.InnerLines, cs.InnerLinesThickness, c.CrosshairImage " \
            f"FROM CrosshairPlayerSetting cps " \
            f"INNER JOIN CrosshairSetting cs ON cps.SettingID = cs.SettingID " \
            f"INNER JOIN Crosshair c ON cs.CrosshairID = c.CrosshairID " \
            f"WHERE cps.PlayerID = {PlayerID}"

        cursor.execute(query)
        result = cursor.fetchall()

        return result

    except Error as e:
        st.error(f"Error connecting to database: {e}")
        return None

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
def unassign_crosshair_setting_sql(setting_id,PlayerID):
    try:
        connection = create_connection()

        if connection.is_connected():
            cursor = connection.cursor()
            query = f"DELETE FROM CrosshairPlayerSetting WHERE SettingID = {setting_id} AND PlayerID = {PlayerID}"
            cursor.execute(query)
```

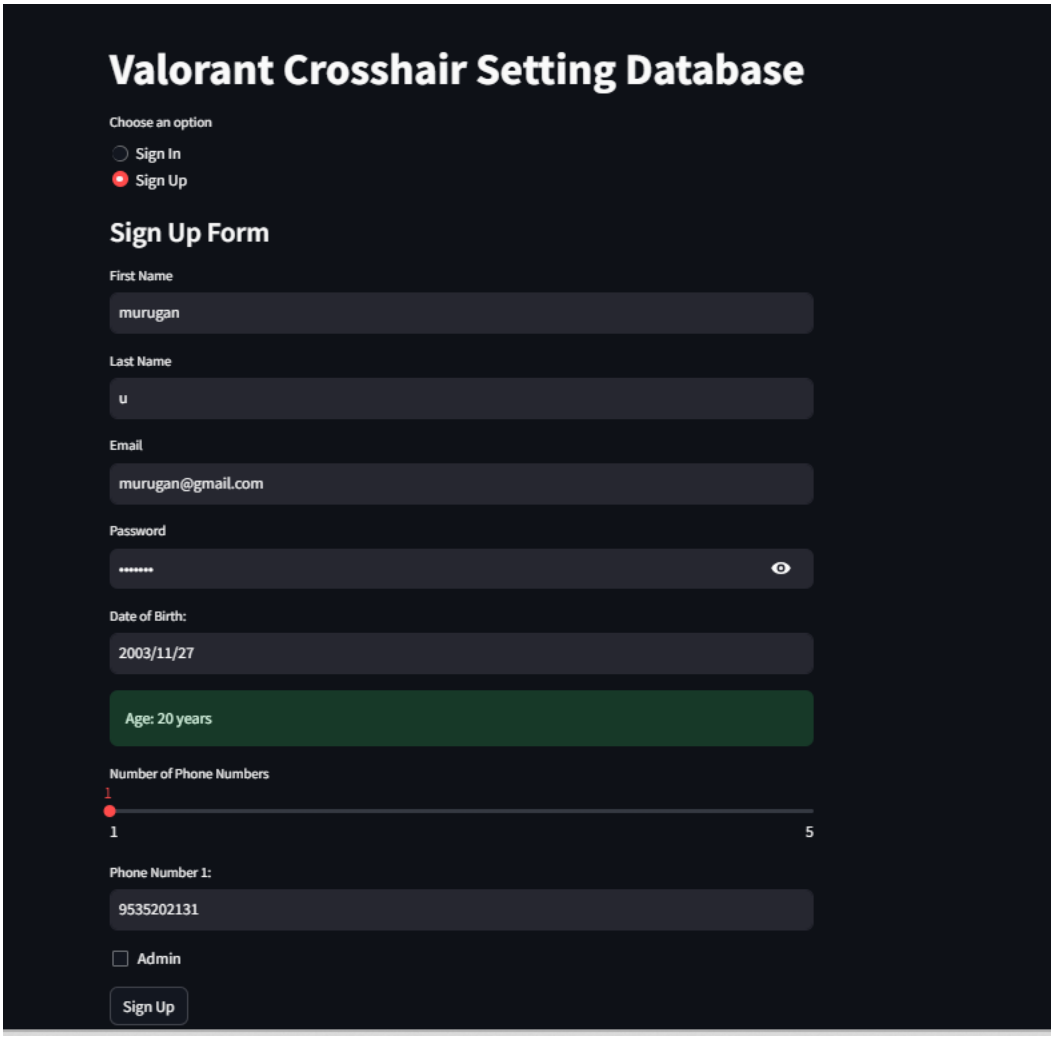
```
        connection.commit()
        st.success("Crosshair setting unassigned successfully!")
    except Error as e:
        st.error(f"Error: {e}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
def unassign_crosshair_setting(PlayerID):

    # Fetch crosshair settings with images
    player_settings = get_player_settings(PlayerID)
    # Display crosshair settings with images in a table
    if player_settings:
        st.subheader("Crosshair Settings to Unassign:")



        # Display the crosshair settings in a table
        df_settings = pd.DataFrame(player_settings, columns=["SettingID", "CrosshairColor", "OutLines",
"OutLinesThickness", "InnerLines", "InnerLinesThickness", "CrosshairImage"])



        # Display the crosshair image
        for _, row in df_settings.iterrows():
            st.image(row["CrosshairImage"], caption=f"Setting ID: {row['SettingID']}", width=50)

        # Dropdown to select a SettingID for deletion
        selected_setting_id = st.selectbox("Select a SettingID to Unassign", df_settings["SettingID"])
        # Delete button
```

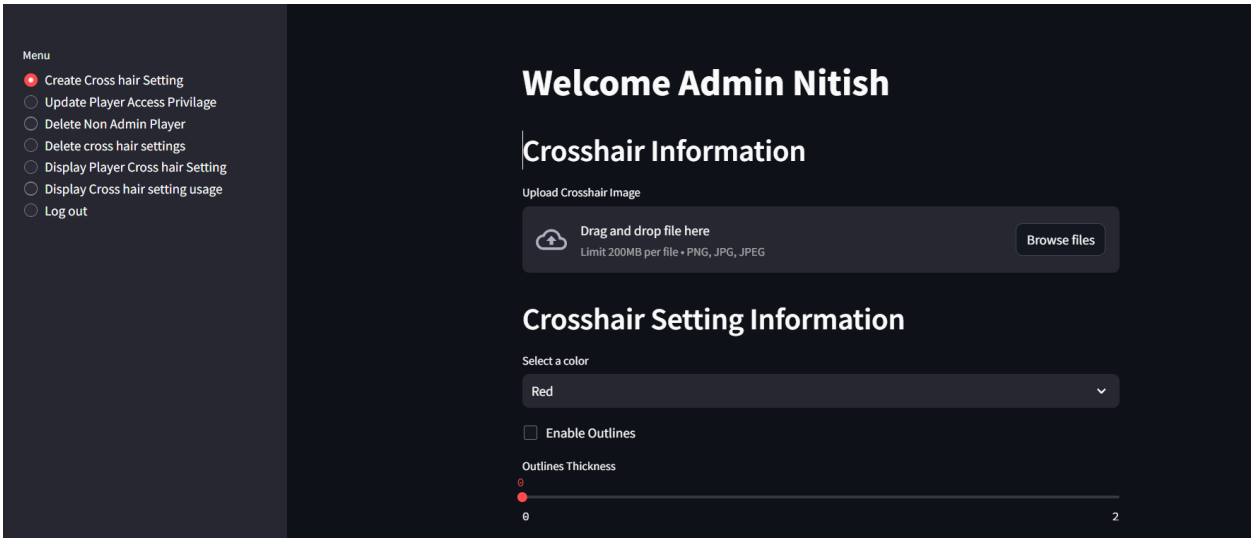
```
        if st.button("Unassign"):
            # Call the function to delete the selected SettingID from the CrosshairPlayerSetting table
            unassign_crosshair_setting_sql(selected_setting_id, PlayerID)
            # Reload the page after deletion to reflect the changes
            st.experimental_rerun()
#user option 2 - end
def user_page(PlayerID,FirstName):
    st.title(f"Welcome User {FirstName}")
    st.title(f"{PlayerID}")
    # Add menu bar items for the authenticated page
    menu_option = st.sidebar.radio("Menu", ["Assign crosshair setting", "Unassign crosshair setting","Log
out"])
    if menu_option == "Assign crosshair setting":
        assign_crosshair_setting(PlayerID)
    elif menu_option == "Unassign crosshair setting":
        unassign_crosshair_setting(PlayerID)
    elif menu_option == "Log out":
         logout()


if __name__ == "__main__":
    # Initialize session state
    if "is_authenticated" not in st.session_state:
        st.session_state.is_authenticated = False


    if not st.session_state.is_authenticated:
        main()
    elif not st.session_state.IsAdmin:
        user_page(st.session_state.PlayerID,st.session_state.FirstName)
    elif st.session_state.IsAdmin:
```

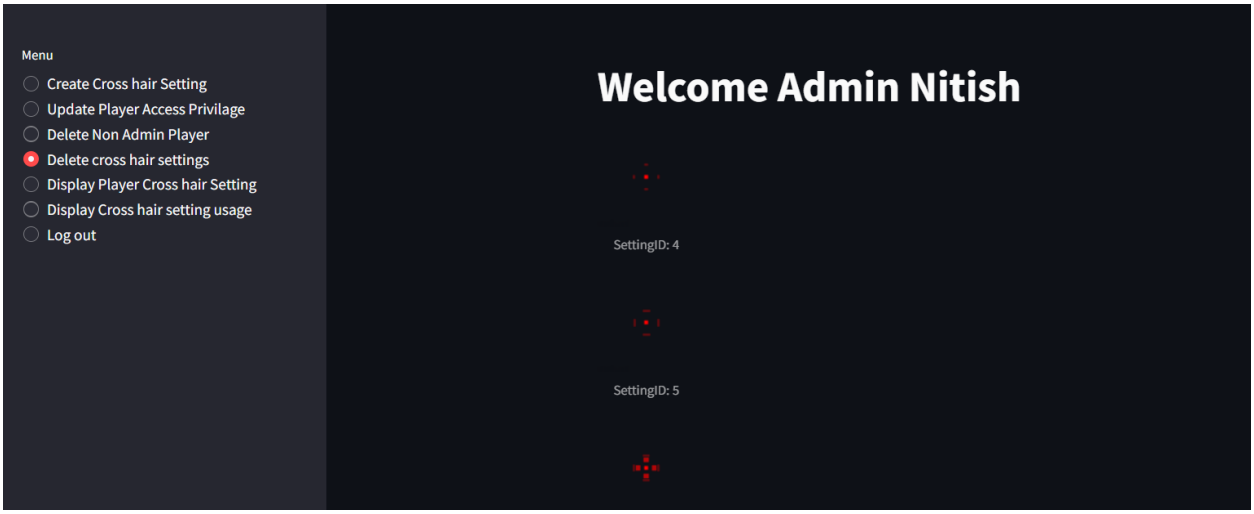
```
admin_page(st.session_state.PlayerID,st.session_state.FirstName)
```

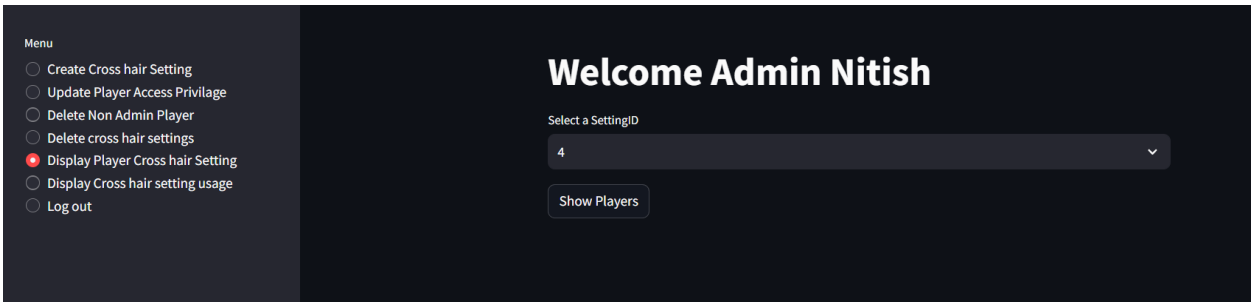
UI Screen Shot

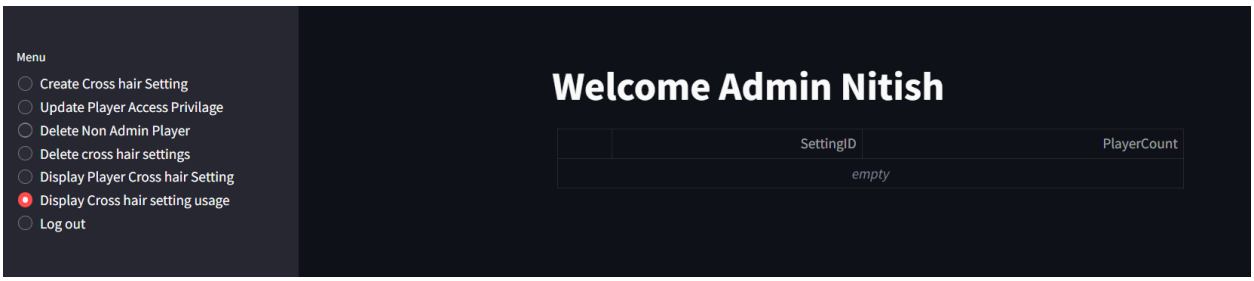


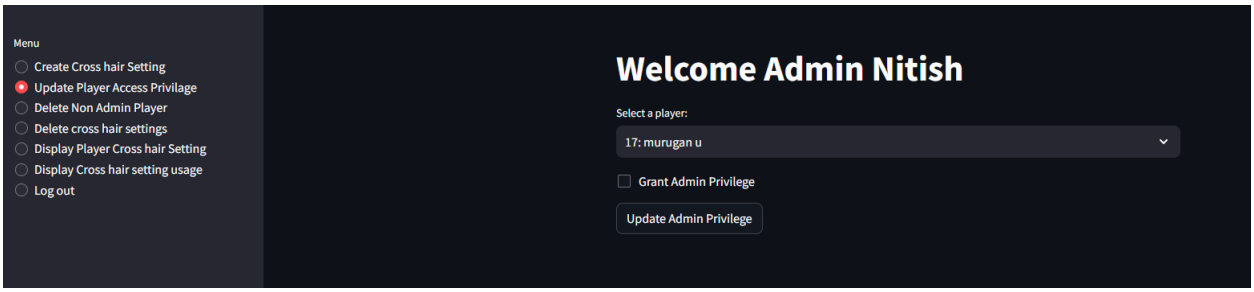

Valorant Crosshair Setting Database
Choose an option
Sign In
Sign Up
Sign Up Form
First Name
murugan
Last Name
u
Email
murugan@gmail.com
Password
Date of Birth:
2003/11/27
Age: 20 years
Number of Phone Numbers
1
1
5
Phone Number 1:
9535202131
Admin
Sign Up

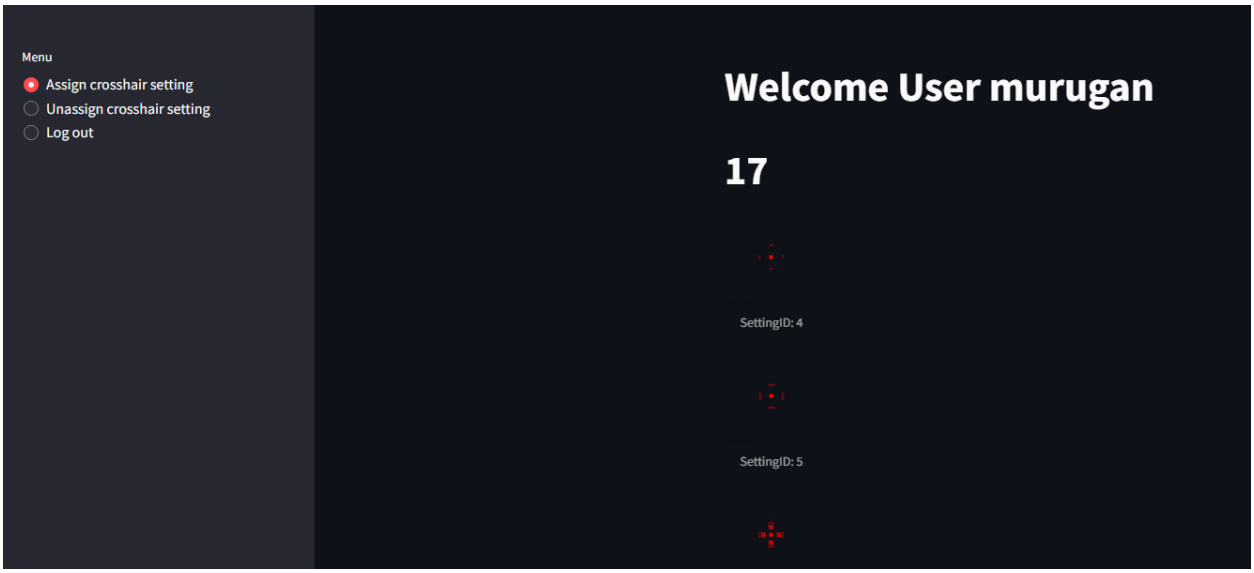

Menu
Create Cross hair Setting
Update Player Access Privilage
Delete Non Admin Player
Delete cross hair settings
Display Player Cross hair Setting
Display Cross hair setting usage
Log out
Welcome Admin Nitish
Crosshair Information
Upload Crosshair Image
Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG
Browse files
Crosshair Setting Information
Select a color
Red
Enable Outlines
Outlines Thickness
0
0
2


Menu
Create Cross hair Setting
Update Player Access Privilage
Delete Non Admin Player
Delete cross hair settings
Display Player Cross hair Setting
Display Cross hair setting usage
Log out
Welcome Admin Nitish
SettingID: 4
SettingID: 5


Menu
Create Cross hair Setting
Update Player Access Privilage
Delete Non Admin Player
Delete cross hair settings
Display Player Cross hair Setting
Display Cross hair setting usage
Log out
Welcome Admin Nitish
Select a SettingID
4
Show Players


Menu
Create Cross hair Setting
Update Player Access Privilage
Delete Non Admin Player
Delete cross hair settings
Display Player Cross hair Setting
Display Cross hair setting usage
Log out
Welcome Admin Nitish
SettingID
PlayerCount
empty


Menu
Create Cross hair Setting
Update Player Access Privilage
Delete Non Admin Player
Delete cross hair settings
Display Player Cross hair Setting
Display Cross hair setting usage
Log out
Welcome Admin Nitish
Select a player:
17: murugan u
Grant Admin Privilege
Update Admin Privilege


Menu
Assign crosshair setting
Unassign crosshair setting
Log out
Welcome User murugan
17
SettingID: 4
SettingID: 5