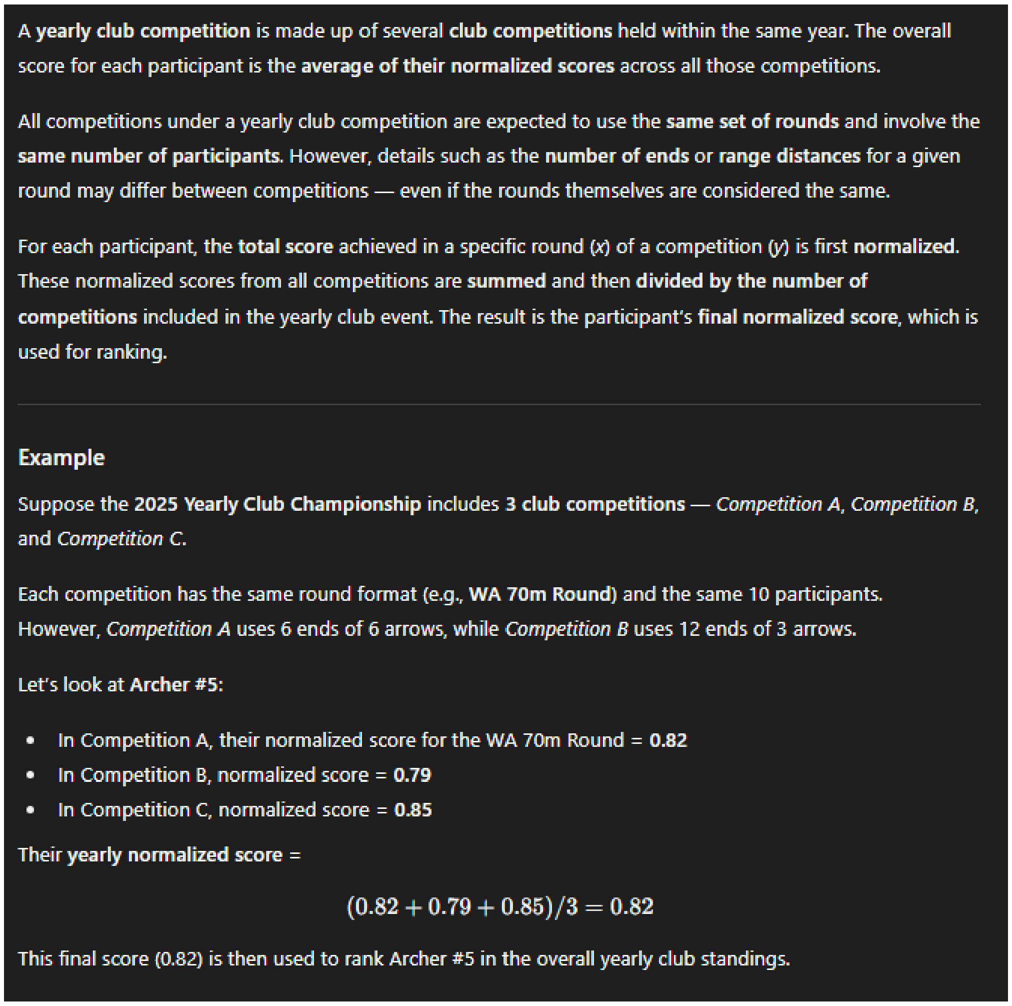
–Autralian_Archery_Club_Mangement_Project–

I: Terminology to know:

1. **Event**: a general term for a yearly club championship or a club competition.
2. **AAF:** Australia Archery Federation.
3. **Yearly Club Championship (also called yearly club competition):** is an event that is made up of more than one club competitions. The photo is verified to be correct by me and what I meant



A **yearly club competition** is made up of several **club competitions** held within the same year. The overall score for each participant is the **average of their normalized scores** across all those competitions.

All competitions under a yearly club competition are expected to use the **same set of rounds** and involve the **same number of participants**. However, details such as the **number of ends** or **range distances** for a given round may differ between competitions — even if the rounds themselves are considered the same.

For each participant, the **total score** achieved in a specific round (*x*) of a competition (*y*) is first **normalized**. These normalized scores from all competitions are **summed** and then **divided by the number of competitions** included in the yearly club event. The result is the participant's **final normalized score**, which is used for ranking.

Example

Suppose the **2025 Yearly Club Championship** includes **3 club competitions** — *Competition A*, *Competition B*, and *Competition C*.

Each competition has the same round format (e.g., **WA 70m Round**) and the same 10 participants. However, *Competition A* uses 6 ends of 6 arrows, while *Competition B* uses 12 ends of 3 arrows.

Let's look at **Archer #5**:

- In Competition A, their normalized score for the WA 70m Round = **0.82**
- In Competition B, normalized score = **0.79**
- In Competition C, normalized score = **0.85**

Their **yearly normalized score** =

$$(0.82 + 0.79 + 0.85)/3 = 0.82$$

This final score (0.82) is then used to rank Archer #5 in the overall yearly club standings.

Formula to calculate a normalized score for a participant X in a round Y in a competition Z, let call it the normalized score A.
B = actual score attained from a round
C = max score attained from a round (max score = total number of arrows used in the row * 10)
Then A = B/C. For ex: B = 180, C = 360 → A = 0.5

4. **Club competition**: An event where there are more than 2 rounds to compete and this is where recorders configure the round using round options available form *round* table and add to a club competition through *event_context* table . For example, recorders may add round A (Male under 21 Long Bow) and round B

(female, under 18 Recurve Bow) to design the competition. We cannot compare scores between different rounds, a competition will have 2 gold medals, one for each category.

5. **Round**: is where participants compete and get ranking for that category in a competition. AAF members have rights to officially add new rounds to the *round* table so that recorders can have more round options to choose when building a club competition. However, we still do not know how many ranges that round in that club competition have.
6. **Equivalent Round**: 2 rounds are considered equivalent if meeting both 2 criteria: (1) sharing the same category_id. (2) being officially added by one of AAF members with specified valid starting date and valid ending date.
7. **Category**: the combination of discipline, age division, and equipment defines a category of a round.
8. **Range:** is where participants can see to determine distance, target face diameter AAF members have the rights to officially add new ranges to *range* table so that recorders can have more range options to choose when building a round in a club competition**.** However, we still do not know how many ends in that range in that round in that competition are there.
9. **End:** is a record in *event_context* table giving the score from 1st to 6th arrow being shot, we have the end_order attribute in event context to help recorders specify how many ends are there for that range in that round in that competition. For example: if recorders specify 5 rows (end order goes from 1 to 5) with same yearl_club_championship_id, club_competition_id, round_id, and range_id, we have 5 ends.

II. Background about Actor:

The web is designed for 4 types of actors (**Admin, AAF Member, Recorder, and Archer**), here are their rights:

1. **Admin** has nothing to do with an archery :

   1.1 **Adding, modifying** account in account table.

   1.2 **Reactivating** and **Deactivating** account.

   1.3 **Review** reports in account_report table to make decision to deactivate or not.

Note: To be a member of Admin, you need to contact DB team

2. **AAF Member** are the people who have rights to add new options, and recorders will design an event using given options provided by them :

   2.1 **Add** new round in *round* table.

   2.2 **Add** new range in *range* table.

   2.3 **Add** new target face in *target_face* table.

   2.4 **Add** new discipline in *discipline* table.

   2.5 **Add** new age_division in *age_division* table.

   2.6 **Add** new equipment in *equipment* table.

   2.7 **Select** a round to be equivalent with another round in *round_equivalent* table.

Note: To be a member of AAF, you need to contact DB team

3. **Recorder** are the people who design an event from options provided by AAF, responsible for the whole yearly club championship or club competition:

   3.1 **Creating** a new yearly club championship in *yearly_club_championship* table, then becoming the creator.
   3.2 **Creating** a new club competition in *club_competiton* table, then becoming a creator.
   3.3.Given forms submitted on *competition_request_form* table applied to "**participating**" by archers or "**recording**" by other recorders, the creator can **make decisions** to **accept** or **refuse**
   a) an recorder to be a member to record in that yearly club championship or that club competition
   b) an archer to participate in that round in that competition if that competition is not a part of a yearly club championship.
   c) an archer to participate in that round in all competitions that make up the whole yearly club championship.

   3.3. **Designing** and **structuring** an event by working with Yearly Club Championship, Competition, Round, Range, End by **viewing, adding, deleting,** and **modifying.**
   3.4 **Scheduling** round for a club competition by working with *round_schedule* table
   3.5 **Specifying** a group of clubs allowed to compete by working with the *eligible_club_member* table and the *eligible_group_of_club* table.
   3.6. **Viewing**, **adding, deleting,** and **modifying** participants and participant score for that round in that competition where the recorders have been registered.

Note: To be a member of Recorder, you need to choose option " recorder" when signing up

4. **Archer** are people who can compete and can only exclusively either join one or create one club:

   3.1. **Viewing** all currently available events and their information (Round, Range, End) linked with the events in the hierarchy through *event_context* table.
   3.2. **Viewing, modifying** his or her own personal (*both competition and practice),* "**modify"** does not apply if his or her own personal score has been verified as "eligible" from Recorder.
   3.3. **Viewing** another participant's personal *(only competition)*, and community's score *(only competition)* for that round in that yearly club championship or a club competition he or she has been registered.
   3.4 **Creating** his or her own club or **Joining** another club. If already creating one club, he or she cannot join another club; if already joining another club not created by his or her, he or she cannot create a new club or join a second club. Overall, only one club is allowed
   3.5 **Making decisions** to accept or refuse other archers to be club members of club created by an archer through club enrollment forms submitted in *club_enrollment* table.

Note: To be a member of Archer, you need to choose option "archer" when signing up

III. Web Page:

This section lists the name of each page and brief description of each page, we expect team members to read brief descriptions carefully, look at the database and see how members can code, work with the database to build the page.

1. **Sign Up / Log in Page:** used to register new accounts for the application and choose one out of 2 roles (archer or recorder). use email and password to log in.

2. **Admin Page:**
   - used to manage user accounts.
   - view account reports in account_report table and make decisions.
   - admin can sort accounts or reports using parameters.

3. **Event Page:**
   - used to browse through events and get information to submit club competition request forms to be a participant in that round in that competition as an archer.
   - used to submit forms to be a member to record in that yearly club championship or a club competition as a recorder. A creator of an event can make decisions to accept or not.

4. **Score Tracking Page:** used by participants to fill out scores and used by recorders to check arrow-by-arrow scores.

5. **Performance Page:** used to browse through personal, others' personal, and community performance.
   - For each round in a club competition, using total score, ranking, percentile for each participant for that round in the event.
   - For same round played by all competitions that form part of a yearly club championship, using ranking, normalized average score (seeing how well an archer performs on that round during a yearly club championship).
   - For category, using *category_rating_percentile* table to get each archer percentile for that category.

6. **Category Page:** used to browse through types of bow and observe which round uses that bow; used to browse through disciplines and read descriptions to understand what that discipline is for; and used to see all official available age divisions. used to add options for recorders to select when designing competition.

7. **Club Page:**

- used to browse information from a club to find a suitable club to enrol as an archer; used to create a club as an archer.
- used to view members of a club and information of a club in “my club” tab, if an archer is the creator, then the archer has the right to kick members out. Archers can find a club to write a club enrollment here.
- used by archers who are creators of clubs to make decisions to accept by setting status to “eligible”.

8. **Chatbot Assistant Page:** Chat GPT- like interface that answers questions based on data in the database, the data AI can access depends on the role of the account. There can be multiple conversations, accounts can choose to delete conversations.

9. **My Connection Page:** used to search all accounts available by a searching bar, can sort accounts based on 4 types of actors or friends only. Accounts can send friend requests with a short introduction message like in Zalo. Clicking gives a place to let two people chat with each other. If one of two accounts blocks another, then both accounts cannot send messages to each other.

10. **My Friend Request Page**: used to see friend requests being sent, being received. Accounts can accept friend requests if they update the status of the form to be “eligible”.

11. **Group Page:** used to search all groups available by a searching bar, or sort groups an account has joined, account can leave a group, account creator can kick other group members. For a club that an account has joined, an account can see group chat and send messages to the group.

IV. Detailed Guideline per page:

1. **Sign Up / Login Page:** nothing to add more to be more specific.
2. **Admin Page:**

   There will be following tabs:

   ### 3.1. Tab 1: Dashboard (selectable mode)

   Fetch necessary information about the tables in the Database to calculate:

   1) Total number of accounts
   2) Total number of deactivated accounts
   3) Total number of accounts where role = “admin”
   4) Total number of accounts where role = “australia_archery_federation”
   5) Total number of accounts where role = “recorder”
   6) Total number of accounts where role = “archer”

   Using st.metric() to display

3.2. Tab 2: Browse Account (selectable mode)

Firstly, let admin configure filter using input widgets on (email_address, full_name, age, role, country) then click “run” button, all config information for filtering will passed to a supabase.rpc(para_1, para_2,.., para_n) made by us, the rpc gets called and returns the filtered accounts, then we use df.dataframe() to display the filtered accounts to web. There are options to displaying only a group of activated or deactivated accounts, or displaying only group of one out of 4 actors (Admin, AAF Member, Recorder, Archer)

3.3. Tab 3: Update Account (updatable mode)

In case someone wants to change their account information and forgets what their account_id is, then let admin search for an account_id using two parameters (email_address, date_of_birth) to fetch a unique row of account from account table to display in df.data_editor() widget, after admin is done with modifying information, admin clicks an “updating” button to confirm the update back to the database.

3.4 Tab 4: Account Report Review (updatable mode)

Fetching all account reports where status == “pending”, admin can view the content of each account report form and also view use st.pdf() to view attached evidence file fetched from file url saved in superbase storage.
changes status to

1) **“eligible”** → run supabase.___ to update the deactivate field in account table == true, delete the account report in report table, delete the file in superbase storage.
2) **“Ineligible”** → run superbase.__ to delete the account report in the report table, delete the file in superbase storage.

3. Event Page:

Tab 1: Event Browse (Viewable):

Section 1: Let accounts, regardless of actors, configure filter using input widgets on (option = [“yearly club championship”, “club competition”], date range, category, club eligibility) → click “apply filter” button to pass config information to supabase.rpc() → the rpc is executed → returns the filtered records, what are the records? There are 2 cases:

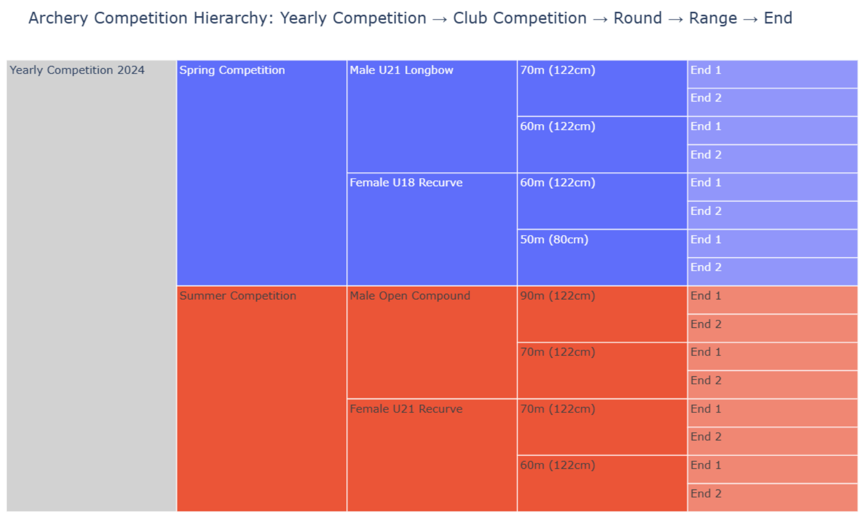
Case 1: records are rows from *yearly_club_championship* table if filter chooses option “yearly club championship”.

Case 2: records are rows following from *club_competition* table if filter chooses option “club competition”) and records satisfy the following criteria: records are club competitions that are not linked or form part of any club championship.

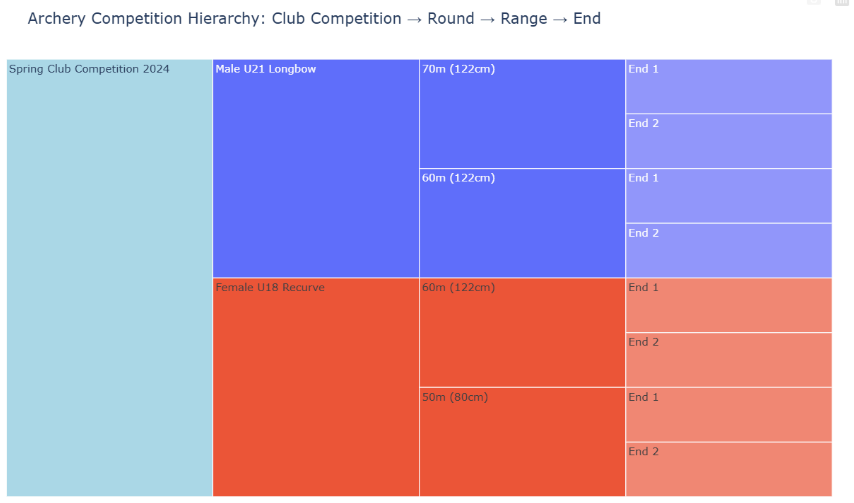
The returned records are displayed using st.dataframe().

Section 2: users get the id of a yearly club championship or a club competition from section 1, then let users input that id into a search bar →

the id is passed to supabase.rpc() as an argument ⟶ the the rpc returns hierarchical data → the hierarchical data then is manipulated using pandas and displayed using plotly.express.icicle(). The .icicle() makes the chart interactive, viewers can zoom in or zoom out per each level in the hierarchy, viewers can see more information when hovering on each block, see [Icicle charts in Python](#).
The deepest children are always End.
If the flow belongs to case 1, then here is an example of an icicle chart.



If flow belongs to case 2, here is an example of an icicle chart.

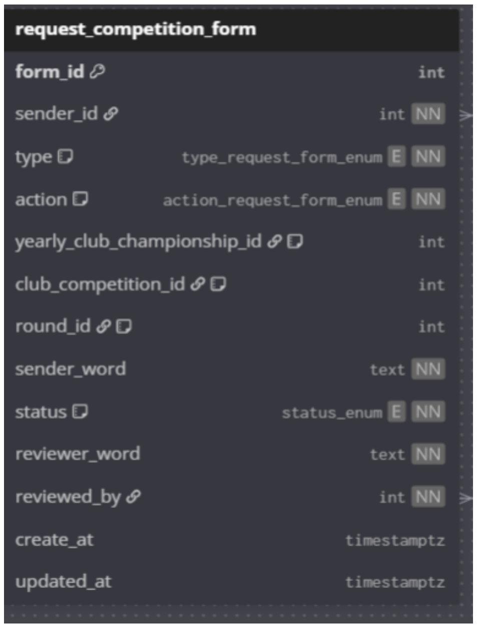


Section 3: Club Eligibility
Simply displaying a list of a group of clubs eligible for the yearly club championship (case 1) or the club competition (case 2)

Tab 2: Event Enrollment / Withdraw Form

Tab 2.1: Write Form

Only archers and recorders are able to write request competition forms and submit them. Only able to “submit” to participate as a competitor if the archer belongs to a club whose club_id can be tracked using *eligible_group_of_club* and *eligible_club_member* tables. Archers cannot apply to action “recording”, and Recorders cannot apply to action “participating”, while other 2 actors (admin and AFF member) cannot access this tab.



request_competition_form		
form_id	int	
sender_id	int	NN
type	type_request_form_enum	E NN
action	action_request_form_enum	E NN
yearly_club_championship_id	int	
club_competition_id	int	
round_id	int	
sender_word	text	NN
status	status_enum	E NN
reviewer_word	text	NN
reviewed_by	int	NN
create_at	timestamptz	
updated_at	timestamptz	

Tab2.2: Review Form

There is only one role and one criteria to satisfy to review the forms to make a decision to accept or refuse, that role is Recorder and the criteria is that the recorder must be the creator of that yearly club championship or a club competition.

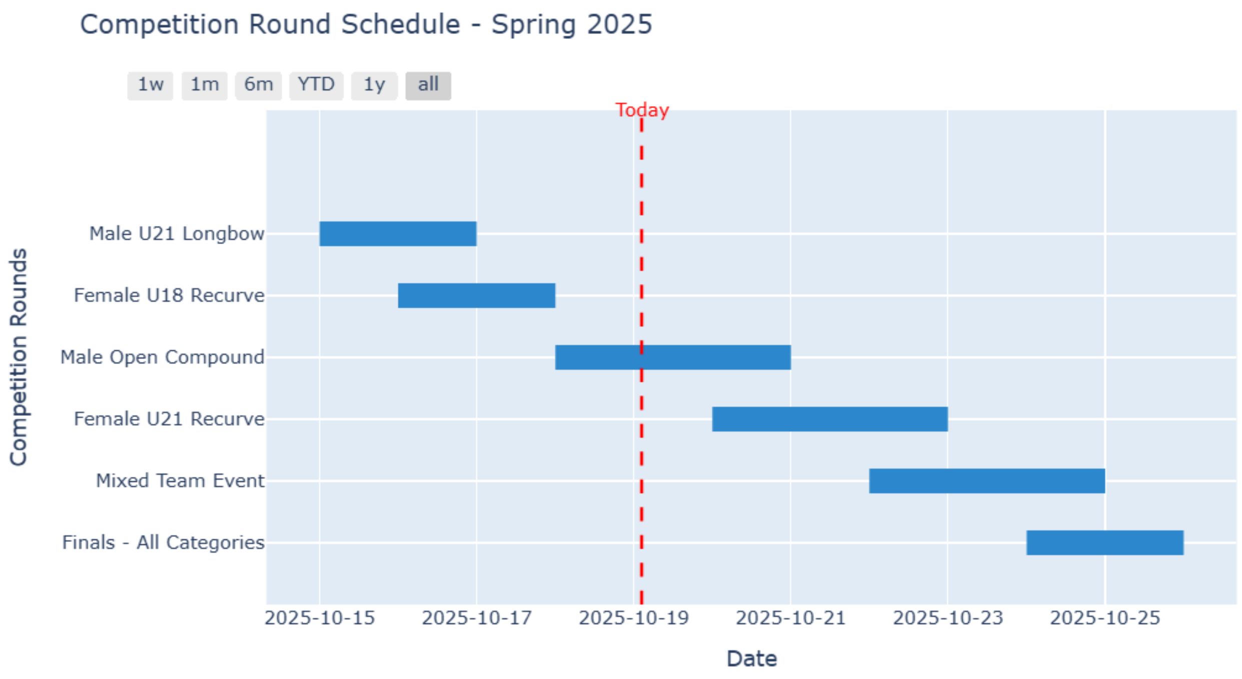
Let accounts (any actor) use input widgets to config filter on status [“pending”, “in progress”, “eligible”, “ineligible”], type [“participating”, “recording”], action [“enrol”, “withdraw”] → click “apply filter” button to pass config information to supabase.rpc() → the rpc is executed → returns the filtered forms, there are 2 cases to display the filtered forms:

Case1: if account role = archer or non-creator recorder, then use st.dataframe() to simply display records (forms).

Case2: if account’s role = recorder and recorder is a creator, then use st.data_editor() to not only display records (forms) but also enable accounts to update status of forms to accept or refuse → after editing one forms or multiple forms by updating status, click the “confirm” button → in order to push the update to database, we scan for edited records (forms) by comparing new modified dataframe of records from dataframe of original records initially displayed after applying filter → get form_id and new status of all records that have been edited → passes to supabase.__ to update the database.

Tab 3: Event Schedule

Let users, any actor, input a club competition id→ click "search" → pass to superbase.__
to get filtered records (rounds of that club competition id) from *round_schedule* table → use plotly.figurefactory and plotly.graphobjects to generate a gantt chart from filtered records.



Tab 4: Event Management

Tab 4.1: Creating Event

Only the Recorder is allowed to access this tab.
Firstly, Recorders choose an option among two choices [" yearly club championship", "club competition"] to start building an event → If option = "yearly club championship", then starting from yearly club championship, add club competitions → add rounds per club competition → add ranges for each round in a club competition, does the same for other club competitions → use st.number_input() to let recorders specify number of ends for a range.
To guarantee consistency, all club competitions that make up a whole yearly club championship must have same participants and same rounds.
If option = "club competition", then does the same process but starting from club competition.

Tab 4.2: Modifying Existing Event

Only recorders who are creators of events can use this tab and view events created by them, and they can only edit events in which starting datetime has not passed now().

Firstly, creator recorders configure filters to search for a yearly club championship or a club competition. The config filter let users choose an option among two choices [" yearly club championship", "club competition"] → assume users know the id of yearly club championship if option = " yearly club championship" or club competition if option = "club competition" from Tab 1 (Event Browse), users input the id using st.text input() → that id will be used as an argument in supabase.__ to display filtered records from *event_context* table using dt.data_editor → editing → click the "confirm" button → in order to push the update to database, we scan for edited records by comparing new modified dataframe of records from dataframe of original records initially displayed after applying filter → get id and new status of all records that have been edited → passes to supabase.__ to update the database.

Tab 4.3: Scheduling An Event

Only recorders who are creators of events can use this tab to schedule events created by them, and they can only schedule events in which starting datetime has not passed now(). Meanwhile, other actors cannot access this site.
Firstly, creator recorders configure filter→ pass to superbase() to get filtered records from *round_schedule* table → displayed records using st.data_editor() → editing → click the "confirm" button → in order to push the update to database, we scan for edited records by comparing new modified dataframe of records from dataframe of original records initially displayed after applying filter → get id and new status of all records that have been edited → passes to superbase.__ to update the database.

4. Score Tracking Page:

Only accessed by Archer and Recorder.

4.1 Tab 1: Score Tracking For Archer

For Archer, using input widgets to enter club_competition_id, round_id of the club competition they have been applied for. No need to enter participating_id as st.session.state() automatically stores that→ pass those id (participating_id, club_competition_id, round_id) to superbase.__ to retrieve only records in that matches with (club_competition_id, round_id, participating_id)→ returned filtered records from *participating* table → using st.data_editor() to display and let archers edit them, only allowed to edit score field from 1st to 6th arrows, cannot edit score field once status has been set to "eligible" by Recorder → Click "confirm update" button → using the approach (scanning) mentioned above to push the update to the database.

4.2 Tab 2: Score Tracking for Recorder

For Recorder, using input widgets to enter club_competition_id, round_id (optional, leave "empty" if want to display all rounds in a club competition), participating_id (optional, leave "empty" if want to see all participants → pass inputs to superbase.__ retrieve only records in that matches with the filter configuration → returned filtered records from *participating* table → using st.data_editor() to display and let archers edit them, only allowed to edit status field and score field from 1st to 6th arrows → Click “confirm update” button → using the approach (scanning) mentioned above to push the update to the database.

5. Performance Page:

All actors can access this page.

Tab 1: View sum score

Users need to enter information to configure the score viewing option.

The configuration is as follows: Getting chosen option for viewing score [“per end”, “per range”, “per round”, “average for round in a yearly club championship”]. The “average for round in a yearly club championship” option simply means a yearly normalized average score for the same round played in multiple club competitions in a yearly club championship (see formula in part I “Terminology”)

If users choose among 3 options [“per end”, “per range”, “per round”] → they need to enter participating_id (optional, leave “empty” if wanting to see scores with type = "competition" of all participants), club_competition_id (also allowing club_competition_id that forms part of a yearly club championship).

Else if users choose option [“average for round in a yearly club championship”] → then they need to enter participating_id (optional, leave “empty” if wanting to see scores with type = "competition" of all participants), yearly_club_championship. → clicking “apply score configuration” → Passing the score configuration to superbase.__ to apply filter, groupby, pivot table to retrieve records → displaying records using st.dataframe()

Tab 2: Ranking

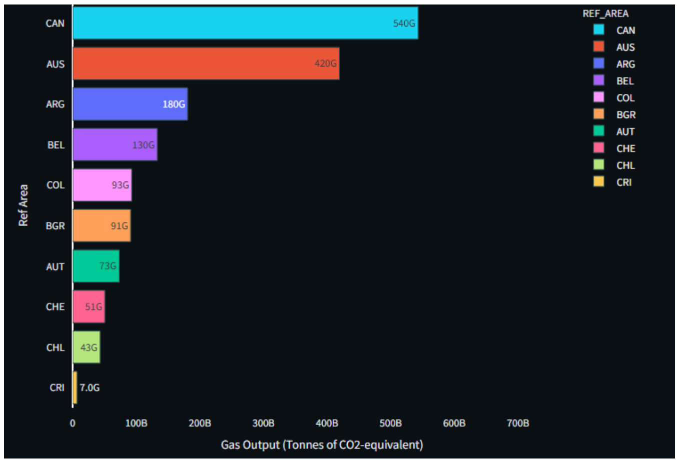
Only apply ranking for participants in rounds in yearly club championships and club competitions where today has passed “year” field in *yearly_club_championship* table or “date_end” in *club_competition* table to guarantee correctness.

Let users choose one among 3 options *[“view ranking in a round in a club competition”, “view ranking in the same rounds played by multiple club competition in a yearly club championship”, “view global rating percentile per category”].*

If users **choose option 1** “view ranking in a round in a club competition:

Let users input club_competition_id (also allowing club_competition_id that forms part of a yearly club championship), round_id → click “confirm” button → passing those id to superbase.__ → getting filtered records from the *participating* table → use pandas to manipulate data (records) → use plotly.express.bar() to generate a horizontal bar chart where the highest value is displayed in the top and lowest value is displayed at the bottom. Each bar is the total score in that round attained by a participant.

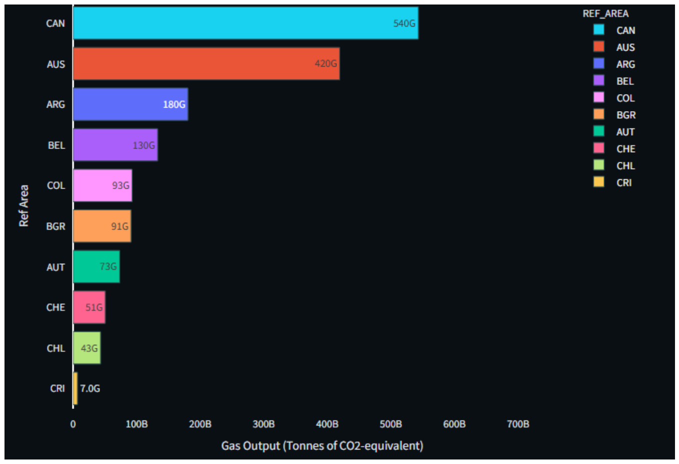
Here is an example of a sorted horizontal bar chart:



If users choose **option 2** "view ranking in the same rounds played by multiple club competitions in a yearly club championship".

Let users input a yearly_club_championship_id, a round_id → click "confirm" button → passing those id to superbase.__ → getting filtered records from the *participating* table → use pandas to manipulate data (records) → use plotly.express.bar() to generate a horizontal bar chart where the highest value is displayed in the top and lowest value is displayed at the bottom. Each bar is a yearly normalized average score for the same round played in multiple club competitions in a yearly club championship. (see formula in part I "Terminology")

Here is an example of a sorted horizontal bar chart



If users choose **option 3** *"view global rating percentile per category".*

Let users input archer_id, category_id → click "confirm" button → passing those id to superbase.__ → getting filtered records from the *participating* table and *category_rating_percentile* tables → use pandas to manipulate data (records) → use plotly.express.area() or plotly.express.histogram() to generate area line chart to show distribution of archers' c_(archer_id). The formula for calculating percentile of an archer in that category is similar to percentile calculated from global category-specific elo rating in chess. For ex: in chess, we have 50 percentile in bliz chess, 10 percentile in classic chess ,... categories.

90 percentile means the archer id performs better than 90% of archers played in that category

formula:

n_(archer_id): number of rounds of that category that were played by the archer id

a_(i): max score for round i that uses that category

b_(archer_id, i): actual score achieved by that archer_id after finishing round i that uses that category

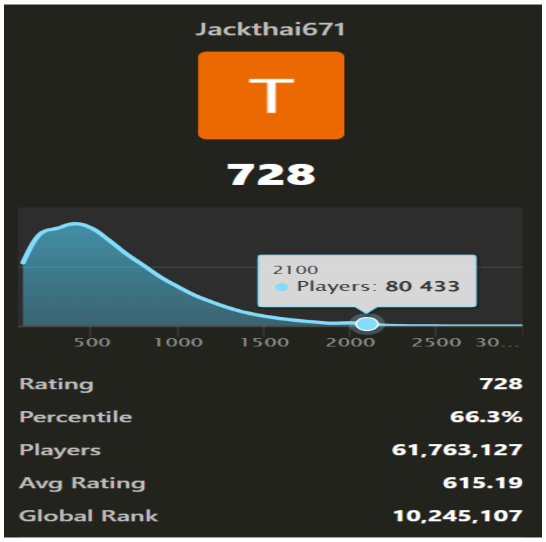
c_(archer_id) = average of sum of b_(archer_id, i)/a_(i) where i goes from 0 to n of the archer id

then we have a list d = [c_(1), c_(2), ... c_(m)] where m is the total number of archers

then we make a d_sorted = [c_(2),.., c_(m), c_(1)] if c_(2) is lowest, c_(m) is secondly highest, and c_(1) is highest]

percentile of x player is c_(x) = ((index of c_(x) in d_sorted + 1) / m) * 100

note that we plus 1 because index starts from 0



Rating	728
Percentile	66.3%
Players	61,763,127
Avg Rating	615.19
Global Rank	10,245,107

6. Category Page:

If role = “AAF member”

Tab 1: Round (insertable)

Add new round in *round* table.

Tab 2: Equivalent Round (insertable)

Choose a round y to be equivalent with another round x in the round_equivalent table from what datetime to what datetime. Note that round y is selected from a filtered set of rounds where category_id of round y = category_id of round x.

Tab 3: Range (insertable):

Add new range in *range* table.

Tab 4: Target Face (insertable)

Add new target face in *target_face* table.

Tab 5: Discipline (insertable)

Add new discipline in *discipline* table.

Tab 6: Age Division (insertable)

Add new age_division in *age_division* table**.**

Tab 7: Equipment (insertable)

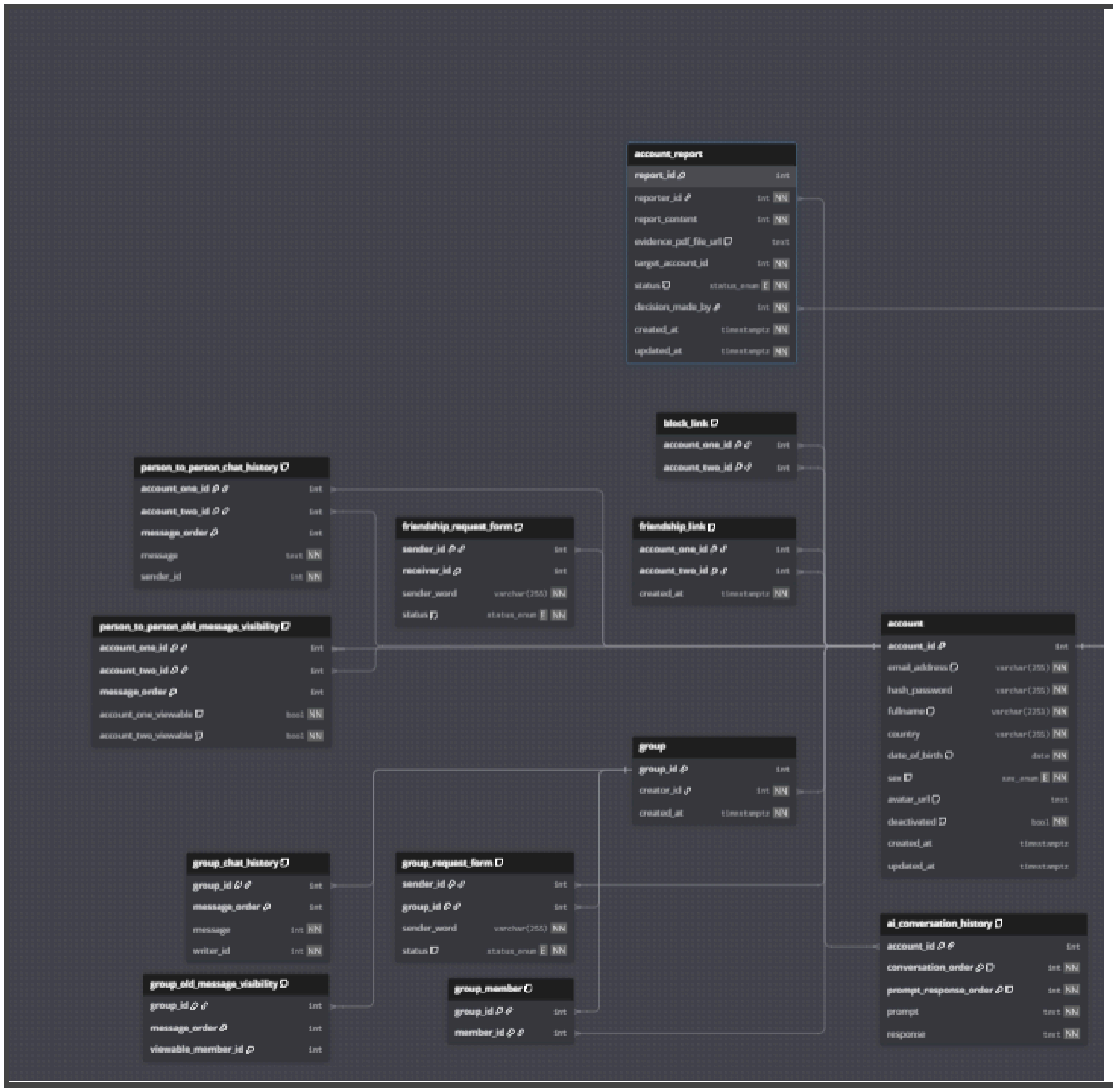
Add new equipment in *equipment* table.

Else:

There are also 7 tabs displaying records using st.dataframe() to let users view only. Also rendering Equipment pictures from superbase storage

VI: Warning Notes:

The ERD and Physical Implementation in Supabase strictly prohibit deleting a record after inserting the record in all tables related to archery, the exception tables in which records can be deleted safely and still guarantee data consistency are the tables used in creating friend link, block link, group creation, message history, message visibility, chatting.



Records in tables (except *account* table) appeared in the photo can be deleted safely