Contents

[**Become a user account requirement:** 1](#_Toc212055162)

[(Some future projects could use the same design and data elements to login) 1](#_Toc212055163)

[Some ideas to share with you on how to accommodate and capture user information: 1](#_Toc212055164)

[**Become a Tenure program membership requirement:** 3](#_Toc212055165)

[(Some future projects might have different membership requirement and capture different data elements) 3](#_Toc212055166)

[Some ideas to share with you on how to accommodate and capture membership information: 4](#_Toc212055167)

[**Determine the queue order requirement:** 4](#_Toc212055168)

[Payment requirement 4](#_Toc212055169)

[Payout requirement 4](#_Toc212055170)

I listed what I think the db schema on data to capture, and rules to accommodate to meet the business requirement. Please fee free to provide your feedback and suggestion.

## **Become a user account requirement:**

## (Some future projects could use the same design and data elements to login)

1. An active user must have either a valid email or phone number to create an account.
2. Same phone number or email address cannot be used by more than one person to create an account.
3. A user can only have one email address and one phone number in the system. (let’s keep it simple for now)
4. A user can update its email, phone number and first/last name after login account creation. (**Roge**r: Do we need to keep a history of changes to these data here?)
5. A user must be at least 18 years old.
6. A user can have multiple addresses of different types but each address type should only have one active record:

For example, of address types:

* 1. Permanent address - US address format for now (Required)
  2. Credit card payment address. (If user uses a credit card that has different address that is not permanent address)
  3. Mailing address (Future?? Skip now to keep it simple)

1. After successfully creating an account, a user can always log in to the account to view the person’s user dashboard, regardless if the person is an active member of the Tenure program (please also see block login below). Not active member use cases could be as below:
   1. Use case#1: A user has not paid initial membership annual fee.
   2. Use case#2: A user has stopped paying annual membership at some points.
   3. Use case#3: A user won the payout and decided not to continue to be a member of the program.
2. Admin/support team can disable a user account (block login) if there is fault detected or user created multiple accounts by mistake.

### Some ideas to share with you on how to accommodate and capture user information:

I highly recommend you guys to create a lookup table for all of the type or status fields, and then enforce the codes to be used on the values table as foreign key. So that you can make sure only the correct type or status are being used for the records in the value table.

As I mentioned earlier, there are many ways to implement a solution. Each has its advantages and disadvantages. My recommendation is to keep it simple and then adjust as needed, my comment is as below.

For example:

#1 If you capture all of the user bio information, email and phone (minus addresses, keep address in a separate tables) in the same table

**Advantages**: when you insert/query/update a user information, it is faster and less complicated since everything is in the same row. You can use database schema rule such as checking uniqueness to a combination of fields due to all fields are in the same row of a table.

You would have to do that unique check in the app business layer if you keep the contact information in a separate table like what you have there now.

**Disadvantages**: In the future, if there is a need to have both work and home email address or phones them the table could get very large and requires changes to the schema.

#2 if you keep the contact information in a separate table like what you have there now.

Advantages: In the future, if there is a need to have both work and home email address or phones then you just need to add the new type code to the type lookup table and the email and phone to the contact table for the new types.

**Disadvantages:**  The steps to phone or email address update. Or to return a single records of a user containing all bio, email and phone numbers if there are 100k users in the system.

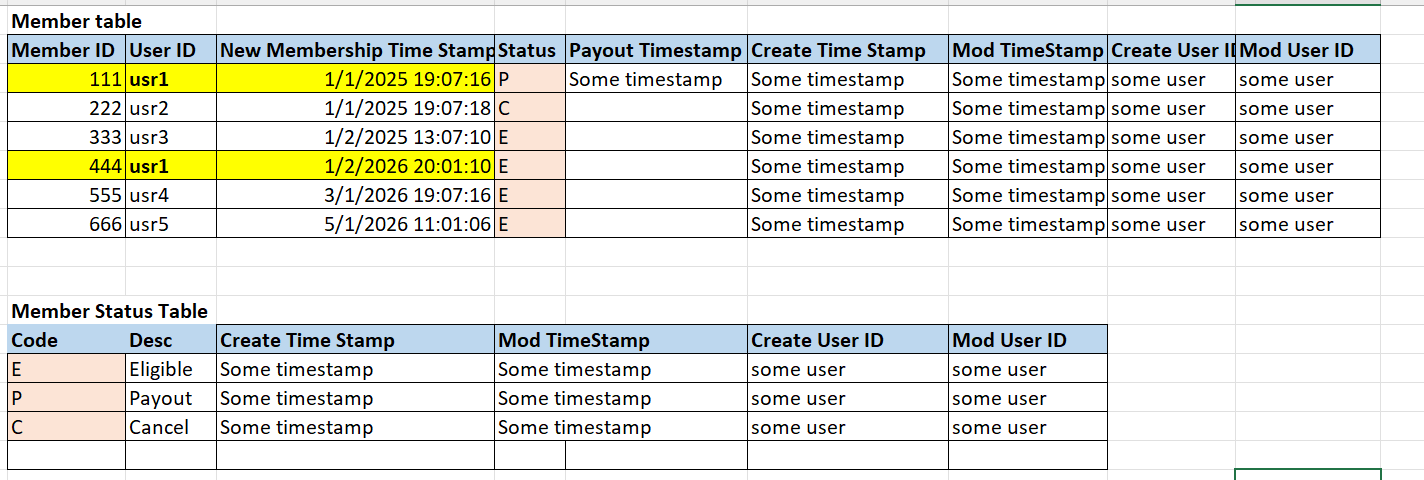
## **Become a Tenure program membership requirement:**

## (Some future projects might have different membership requirement and capture different data elements)

1. An account’s user becomes an active **Member** once the person has paid in full “first annul+ first month” fees. The person becomes a membership timestamp needs to be stored in the member’s table and the value should **never** be updated. A unique member ID should be generated for this membership timestamp. Each member timestamp should be unique in the entire member table, meaning no two users can have the same member’s timestamp as this value will be used to determine queue priority.
2. An account user is not an active **Member** and is not eligible for the payout queue calculation order. If
   1. The person stops paying the annual or monthly fee.
   2. The person chooses to leave the program.
   3. Admin disables the person membership.
3. A user could have multiple member records:
   1. The person stops paying the annual or monthly fee but after sometime, decides to pay and join again. If joining again, the person will have a new member record with a new member id, an active status and a new “member” timestamp (queue order will be reset by using this timestamp). The previous member record will remain in-active and has its previous original member timestamp.
   2. Each **Member** record can only have **one and only one** payout event. Once a payout goes to a member record of a person for a given member timestamp. The member record should have a “payout” status therefore is no longer be eligible to be used for future payout. The person can decide to pay and join again. If joining again, the person will have a new member id, an active status record and a new “member timestamp” (queue order will be reset by using this timestamp). The previous member record will remain in-active and its own old become member timestamp.

### Some ideas to share with you on how to accommodate and capture membership information:

In the example below usr1 user has a payout member record and decides to join and paid on 1/2/2026 again.



## **Determine the queue order requirement:**

1. The queue order will be re-calculated dynamically every time a person drop from the program or won the payout or became a new member. (Scheduled job or trigger by the drop/add/payout events?? TBD)

2. Each active member will have a queue number assignment dynamically based on various criteria as below:

1) First In first out by active member timestamp.

More to come….

## Payment requirement

More to come

## Payout requirement

More to come