Youth Club Attendance System Building Report

**Requirements analysis**

By analysing the description of the scenario, there are mainly two types of user of this APP, which are young members and adult members. Functions for each type of member are listed below.

General:

1. No login checking requirements.

For Youth:

1. Display all activities that include young members.

2. Displayed activities should be selectable for detailing all young members involved.

3. Mark the attendance status in a particular activity.

For Adult

1. Select name to login.

2. Display all members in a group.

3. Display attendance status of one member in all involved activities.

4. Display attendance status of all involved members in one activity.

5. Attendance status can be modified.

6. Display detail information of one particular group member.

7. Member details can be modified.

8. Add a new member for one group.

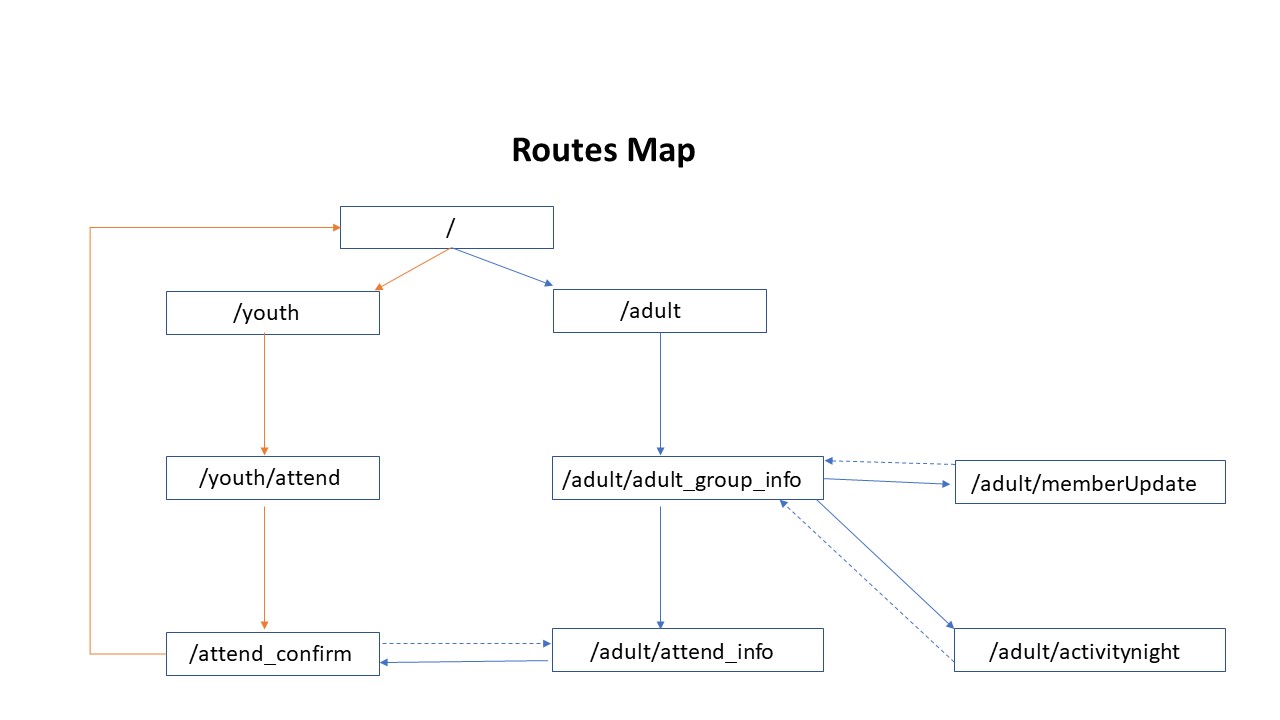
9. Display all the activities of one group.

10.Add a new activity for one group.

To summarise, two main interfaces for two types of users are needed. The functions for adult members could be categorised into three parts to deal with different information. Furthermore, all information is within a group. Therefore, a main interface for adult users based on their group showing the information is reasonable.

**Function implementing**

All the routes for implementing the above functions are displayed in the sketch below.



**Youth part:**

The route (/youth) displays all the activities that includes young members. Users could click on the hyperlink to select the event they should attend.

The route (/youth/attend) displays all young members involved in one activity, also contains a form to collect information of attendance status. To facilities the data enquiry from database, a VIEW named events is created in the database through SQL benchwork. SQL statement is

CREATE view events

AS

SELECT activitygroup.groupname, activitygroup.groupid, currentgroupmember.memberid,currentgroupmember.firstname, currentgroupmember.familyname,currentgroupmember.dateofbirth, activitynight.activitynightid, activitynight.nighttitle, activitynight.activitynightdate, activitynight.description

FROM activitygroup

LEFT JOIN activitynight ON activitygroup.groupid = activitynight.groupid

LEFT JOIN currentgroupmember ON activitygroup.groupid = currentgroupmember.groupid

ORDER BY activitygroup.groupname;

The route (/attend\_confirm) deals with data come from both (/youth/attend) and (/adult/attend\_info), which are differentiated by respective variables. It inserts the collected information into table attendance by in database. For young users, this function returns to a template youth\_confirm.html, while for adult users, it returns to the (/adult/attend\_info). In order to use INSERT ON CONFLICT enquiry in database for updating or adding information, a UNIQUE constraint has been added into the table attendance (activitynightid,memberid) .

**Adult part:**

The route (/adult) is the first page for adult members, which demonstrates all current adult members name and groups for selecting. The group is expected to be linked with members name, but this function is failed to implement, which needs more study. To facilities the data enquiry from database, a VIEW named currentGroupMember is created in the database through SQL benchwork. SQL statement is:

CREATE VIEW currentgroupmember

AS

SELECT member.memberid, member.firstname, member.familyname, member.dateofbirth, groupmember.groupid, activitygroup.groupname

FROM member

INNER JOIN groupmember ON member.memberid = groupmember.memberid

INNER JOIN activitygroup on groupmember.groupid = activitygroup.groupid

WHERE groupmember.leftdate IS NULL;

The route (/adult/group\_info) is the main interface for adult users, which contains a list of group member information and a list of group activity information. There are five hyperlinks on the page which direct to five functions:

1. Edit and update the attendance status of one member in different activities.

2. Edit and update the attendance status of all the member associated within one activity.

3. Update the detail information for a group member.

4. Add a new member in group.

5. Add a new activity information for group.

Function 1 and function 2 shares the route (/adult/attend\_info) and are distinguished by sending different variables through the url. The view function adult\_attend\_info() selects information from database according to the different variable and returns to template adult\_attend\_form.html separately, which is in order to display information grouped by members or events. User selected information would be sent to route (/attend\_confirm) by POST method for inserting into database, and then returns to (/adult/attend\_info) displaying attendance status sorted by members or activities. For also displaying the members who have left the group, a view named allmember is created in database, the SQL statement is:

CREATE view allmember

AS

SELECT member.memberid, member.firstname, member.familyname, groupmember.joineddate, groupmember.leftdate, groupmember.groupid, activitygroup.groupname, member.dateofbirth, member.adultleader

FROM member

INNER JOIN groupmember ON member.memberid = groupmember.memberid

INNER JOIN activitygroup ON groupmember.groupid = activitygroup.groupid;

Function 3 and function 4 use the same route (/adult/memberUpdate) and are distinguished by variables, which is the same logic with the implementation of function 1 and function 2. However, user inputs are sent back to same route by POST method for inserting into database. For the use of INSERT ON CONFLICT enquire, UNIQUE constraints have been added into the table groupmember (activitynightid, memberid) and table member (memberid). This route redirects to (/adult/group\_info) after inserting information into database.

Function 5 is achieved though the route(/adult/activitynight) and redirect back to (/adult/group\_info) after inserting new activities into table activitynight.

**Deficiencies**

1. Due to the limited time, form validation has not been implemented, several function were failed to added. Another reason is that unfamiliar with Javascript, Bootstrap, jinjia and insufficient practice of html building. Web layout is unstable, would vary with browser zooming.

2. Redundance Python code, unfamiliar with Python functions and methods, the logic is straight and simple and insufficient of logically express. Data type converting between database and python is tricky to deal with.