Contents

[Useful links 2](#_Toc485482993)

[Timeline 3](#_Toc485482994)

[Standards to follow 4](#_Toc485482995)

[Add Client - Donna 5](#_Toc485482996)

[Training Session & Synching with Calendar - Kim 9](#_Toc485482997)

[Workout Design - Deng 11](#_Toc485482998)

[Notes - Xian 12](#_Toc485482999)

[Additional Notes 13](#_Toc485483000)

# Useful links

Google-Udacity Video Course:

https://classroom.udacity.com/courses/ud851

# Timeline

|  |  |
| --- | --- |
| 3-June | Start of project – requirements gathering |
| 10-June | Training |
| 17-June | Training |
| 24-June | Start assigned tasks |
| 1-July | Continue with tasks |
| 8-July | Integrate/Test |
| 15-July | Fix bugs/Add any missing features |
| 22-July | Continue debugging |

# Standards to follow

Include referential integrity when adding all tables

The id column for all tables must have the same name and must be autonumber. Never display the id in the front end. The field length for all ID values in the database must be 3.

Create extensive user documentation

* include your name as the author for all files that you have created
* if you edit any files for which you are not the author, comment the parts you have changed and add your name to the maintenance history
* update the version number each time a file is changed
* Update the maintenance history when making any changes. Mention the date, your name, and what you changed/added.
* Files name must use small letter and an underscore must be used instead of a space
* Functionality and design (of elements such as deleting, adding, viewing) should be consistent. Therefore, before implementation, check the description for other tasks to see if you are being consistent.

# Add Client - Donna

client – ID(3), firstName(20), lastName(20), dob, weight(3), height(3), waist(3), phoneID(3), addressID(3), emailID(30), gender (Boolean)

phone - ID, number (10)

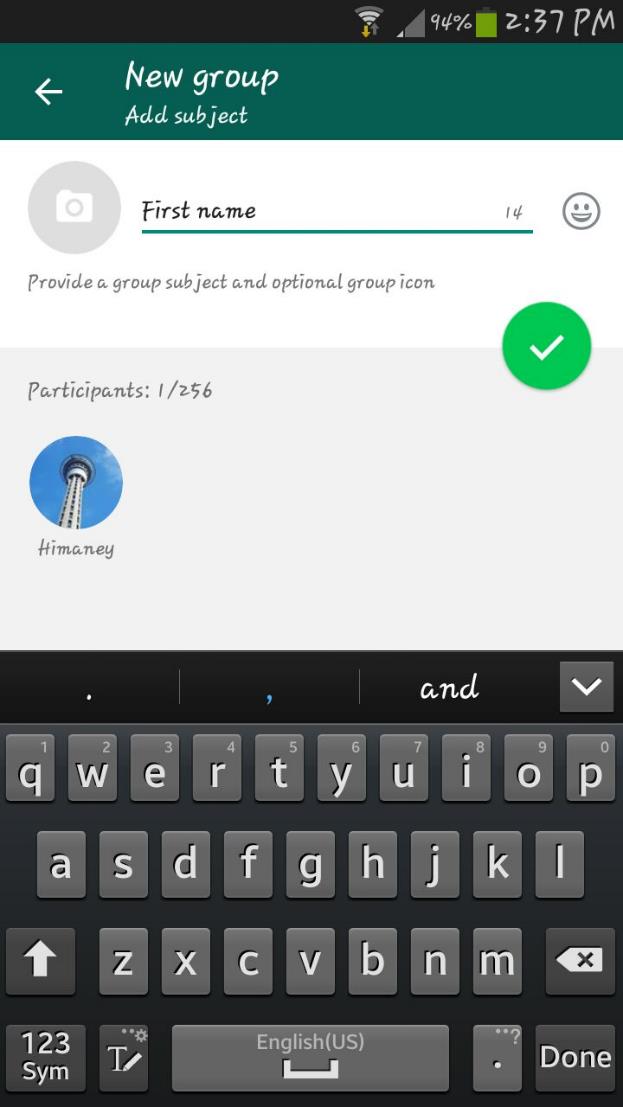
address - ID, line1 (15), line2 (15), line 3 (15)

email - ID, email (20)

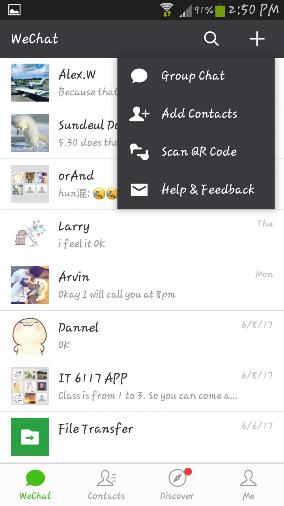
registration - ID, clientID, date, time

Features to be included:

1. All fields must use place holders (fig 1)
2. Ability to add the client’s picture (fig 1)
3. The right side must show the remaining characters (fig 1)
4. The main page should display a list of all the clients (fig 2)
5. The list should be scrollable if needed
6. There should be an add (+) icon on the top-right. When clicked, shows a drop-down list (fig 2)
7. For now, the drop down will only have one item: Add client (fig 2)
8. Clicking on add client will open a form to add a client. This form will have the features mentioned in figure one.
9. Use a linear, clean layout for the form. As shown in figure 3. (Use appropriate fields – number, date, text, option button, etc.)
10. When a client is clicked from the main page, then their profile is displayed in a categorized manner (shown in figure 4). The first banner should show the profile image and the name just as it is shown in figure 4. When this banner is clicked, it loads a new screen to edit the image, first name, and last name. (Implement the adding on an image similar to how it is done in WhatsApp. Exclude, the web option.)
11. The second banner should group: dob, weight, height, waist, gender
12. The third banner (heading: Contact) should group: phone, address, email
13. All banners should be clickable and must then load a screen that allows editing of only those fields.
14. It should be possible to delete clients from the main page by holding on their name. This will bring up a list. One of the entries in this list will be delete client. Will ask for a confirmation before deleting. The list will only have one item for now. (This feature is similar to how deleting a conversation works in WeChat).
15. The list of clients must be ordered by showing the most recently added client first.
16. It should only be possible to submit the form to add a new client when all mandatory fields have been filled. Mandatory fields are: firstname (there is only one).
17. Edit the second banner to also display the client’s age.
18. Edit the form so that the user can also enter the age of the client. However, do create a new column in the database table. Save the age in the form of a year of birth. (But if a dob has been entered then simply use that to calculate age, i.e. the user either enter the DOB or the age so we save the DOB or the YOB respectively.)
19. It should be possible to undo the delete action. To implement this, you may have to add a new Boolean column to the table. (The undo icon should only be visible after a client has been deleted.)



**Figure 1 – Screen capture of WhatsApp when making a new group.**



**Figure 2 – Screen capture of WeChat when clicking the add icon**



Figure 3 – Screen capture from WeChat when editing remarks

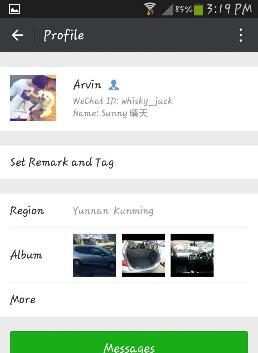


Figure 4 – We Chat when viewing a profile

# Training Session & Synching with Calendar - Kim

client – ID, name

schedule - ID, startDate, endDate, startTime, endTime, clientID, mon, tue, wed, thur, fri, sat, sun,

Features to be included:

1. Create a test app that has the following function. You add a date and time value (hardcoded) and click a button. When you click the button, it takes this date and time value and enters it into the Google Calendar app installed on the same phone. (Use the Google Calendar APIs)
2. Create another test app that has the following function. You click on a button and it transfers to the Google Calendar app. From the google app you choose a date, time and make a booking. After you submit this booking, the screen returns to the test app and the data chosen from the google app is displayed in the test app. (Similar to how the ANZGo app transfers control to a messaging app when the share icon beside the A/C# is clicked.)
3. First create a simple client table with ID, and name (only 2 fields)
4. Create a simple form with one field to enter a customer id and one button to submit. When you submit this form, it takes the user id and creates a new form with heading: Training Schedule. This is the main form for the task.
5. The main form will allow the user to enter the start and end dates, the start and end times, and the days when the training will recur. (The customer id is taken from the previous form). Save this data to the table. Do not work on making the form user friendly right now. That will be done on a later task. (Do not focus on the design of this form. All forms will be redesigned by one person at the end of the project. Keep it simple and follow the images above.)
6. It should be possible to synchronize the schedule when it is created with the user’s Google Calendar. Therefore, as soon as you submit the Training Schedule form, it will display a pop-up, and ask the user, “Do you wish to add this schedule to your calendar?”
7. You do not have to worry about displaying, editing, or deleting existing schedules. Only work on adding.
8. Download the Alarm Clock Plus App for Android and view how they use a form to input the time and the days. Follow their design. Use calendars to input the start and end dates. Make the form fields as user friendly as possible. If you have other ideas, then please share with the team
9. When saving the schedule, you should check for a clash. If there is a clash then ask the user to pick a different time.
10. Remove the end date field from the form and replace with duration (in weeks). The back end will remain the same. (Do the same for the end time. Replace it with a session duration.)
11. Add a check box beside the start date field (label – Use personal calendar). If this check box is not selected then the user can enter the start date into the field by directly using the app. If it is selected then clicking on the start date field will transfer control to Google Calendar. If the check box is enable then it should stay enabled always and vice-versa.
12. When you click the start date while the check box is enable, it should switch to google calendar and allow the user to use the Google interface to choose date and time. You will have to use the Google APIs to transfer the value of the date and time from google to our Training App.

# Workout Design - Deng

Create page with one button | Design Workout |

Click button opens form to design workout

Should be able to create a list of "Focus Areas"

These focus areas should be saved in a table with priority

For future designs, exisiting FAs should be available.

focusArea - id, activityName (unique constraint), frequency

client\_focusArea - id, clientID, focusAreaid

//maybe combine both list and input field ...

//Later feature - save workout profile to use for other clients ... allowing editing

//make simple customer table with id and name

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Notes - Xian

noteType - id, noteType (15) (unique constraint).

1. Create a form to add a type of note to the noteType table.
2. Create a page that will list all the note types in a page.
3. Do not focus on the design of these pages. Someone else will work on the design at the end.
4. Each of note types will also be a button. When you click the note type it will open a form to add a new note under that note type. (Just ensure that a new form opens, you do not have to add any content to these forms right now.)
5. There should be three note types by default: General, Session, Check-in
6. It should be possible to delete a note type (Use the code from deleting a client to make sure you do it in the same manner)
7. It should not be possible to create more than 5 note types. If the user tries to create a sixth, display a popup saying, “You cannot create more than five types of notes.”
8. It should not be possible to delete the general note type.

note - id, noteText (255), noteTypeID, date, time, clientID, prioritized (boolean)

1. Create a form to add a note to the note table. Fields – Note text, date, time. (For now, use default hardcoded values for noteTypeID, clientID, and prioritized = false) (Date and time are calculated, not entered.) The form’s design should be similar to that of WeChat for sending a chat message.
2. Above the form, all existing notes should be visible (just as sent messages are visible in a chat history)
3. Sort the notes by date (same as it is done in messaging apps)
4. It should be possible to click and hold the notes. This will bring up a list (similar to the client list). The list should contain an option – <star icon>
5. When a note is marked as important, change the value of prioritized to true.
6. If a prioritized note is pressed then the star icon will be replaced with a crossed-out star icon (see WhatsApp)
7. Add another option to the list – Delete (See delete function for client. Implement the same way.)
8. Add a filer icon to the top right. When this is clicked, it displays a list. (See the add icon in client list page. Implement the same way)
9. The list has the following option – View All, General, Session, Check-In. Clicking on one of these, filters the list of notes below.
10. When a note is marked as important, a star icon should appear beside it, within the same text bubble. (See how it is done in WhatsApp)
11. There should a star icon on the top-right, beside the filter icon. This button can be toggled. If on, then only important notes will be shown. If off, then all notes will be shown.
12. Add a scroll function (see client list)
13. Categorize the notes into weeks similar to how it is done in chatting apps based on days. (Week 1 <date>, Week 2 <date>, etc.)
14. So far the note-type has been hardcoded. Refactor so that by default, the filter option is selected to view all and when a note is entered at this time, it is a general note.
15. If the filter is set to Session then the note type will be session. Implement a similar functionality for the other two.
16. Modify the filter button so that the list shows all existing note types. (Max = 5, Min = 1)
17. Different note types should use different colors for their bubbles.
18. Add an undo button to undo a delete operation. The undo icon should only be visible after a note has been deleted. (See undo delete function for client. Implement the same way.)

# Additional Notes