# Phase 3: Child

**Aspiration 6:** Share with us some queries (at least 3) in your application that require database access. Provide the actual SQL queries you use and explain how it works.

## Query 1

```
INSERT INTO issues_fb (IssueName, startDate, endDate, description, dataSource, rsvp, idToken, lastUpdated, location) VALUES (<PARAMETERS HERE>) ON DUPLICATE KEY UPDATE IssueName = (CASE WHEN (lastUpdated != VALUES(lastUpdated)) THEN VALUES(IssueName) ELSE IssueName END), startDate = (CASE WHEN (lastUpdated != VALUES(lastUpdated)) THEN VALUES(startDate) ELSE startDate END), endDate = (CASE WHEN (lastUpdated != VALUES(lastUpdated)) THEN VALUES(endDate) ELSE endDate END), description = (CASE WHEN (lastUpdated != VALUES(lastUpdated)) THEN VALUES(description) ELSE description END), rsvp = (CASE WHEN (lastUpdated != VALUES(lastUpdated)) THEN VALUES(rsvp) ELSE rsvp END), lastUpdated = (CASE WHEN (lastUpdated != VALUES(lastUpdated)) THEN VALUES(lastUpdated) ELSE lastUpdated END), location = (CASE WHEN (lastUpdated != VALUES(lastUpdated)) THEN VALUES(location) ELSE location END)
```

This inserts fields of fb events that a user has some sort of link to into the mySQL database, but only when said events have not been inserted. If they are, it does nothing, but if they are inserted and lastUpdated time is different, it means that the FB events' parameters have changed and the query will update the fields accordingly.

## Query 2

```
Partial query that is included in by other queries:
STR_TO_DATE(<PARAMETERS>.split[0], '%Y-%m-%dT%H:%i:%s')
```

This basically converts FB's DateTime format into mySQL's DateTime format. Of note is that we ignore GMT offsets and assume it to be relative to local, wherever that is, because FB events are inconsistent in their inclusion of GMT/Timezone.

Planendar – the only organizer app you ever need

## Query 3

```
select issueID from issues where (startDate >= D1 and startDate <= D2) or (endDate >= D1 and endDate <= D2) or (startDate <= D1 and endDate >= D2) union select issueID from issuesmeta where (repeatstart >= D1 and repeatstart <= D2) or (repeatend >= D1 and repeatend <= D2)or (repeatstart <= D1 and (repeatend >= D1 or repeatend = null) and (ceil((D1-repeatstart)/repeat_interval) <= floor((D2-repeatstart)/repeat_interval) or ceil((D1-repeatstart+((select endDate from issues where issues.issueID = issuesmeta.issueID) - (select startDate from issues where issues.issueID = issuesmeta.issueID)))/repeat_interval) <= floor((D2-repeatstart+((select endDate from issues where issues.issueID = issuesmeta.issueID) - (select startDate from issues where issues.issueID = issuesmeta.issueID)))/repeat_interval) or (floor((D1-repeatstart)/repeat_interval) + ((select endDate from issues where issues.issueID = issuesmeta.issueID) - (select startDate from issues where issues.issueID = issuesmeta.issueID))/repeat_interval > (D2-repeatstart)/repeat_interval)));
```

This query seeks to return events that occur within a given range; by occurring we mean that as long as any part of an event touches/is relevant to the given range it will be returned. The second part of the union essentially does the same thing for repeatable events – it takes into account the repeatable aspects.

**Aspiration 7:** Show us some of your most interesting Graph queries. Explain what they are used for. (2-3 examples)

## Example 1

```
/{userId}/events/[attending OR created OR maybe OR not_replied OR declined]?fields=name,start_time,end_time,description,rsvp_status,updated_time,admins,location
```

This is the Graph query that enables our core functionality. Events from up to 2 weeks back will be pulled to our database when the permission is obtained. Also, this pulls all the relevant information we need for our app; of note is that admins is considered a separate object and returns an array of event hosts under an additional data wrapper. Even more interesting is that if a page created the event, then for admins (aka event hosts) it will return null; only user-hosts will be returned! #TIL moment indeed.

### Example 2

```
/{userId}/groups?fields=name,description,administrator
```

The intended purpose for pulling user groups is to enable permission sharing in the future. Users can set their calendar to be only visible to certain groups (e.g. if they want to hide events from other friends or their parents). By default, these groups will only be the Facebook groups unless they create or modify their groups on Planendar. This query pulls the relevant information we need; of note is to also to know if the user is an administrator of the group or not – we use this in our internal permissions system.

## Phase 4: Teen

**Aspiration 9:** Your application should include the Like button for your users to click on. Convince us why you think that is the best place you should place the button.

The Like button is placed in the footer under the Social column. It is meant to be inconspicuous and hidden from the user's view as part of the minimalistic UI adopted. Through Dynamic HTML, the Like button is always hidden from view until the user scrolls down. This makes the UI less cluttered and in turn, more focused on the core functionality – the tasks and calendar functions.

This does not mean that we compromise on promotion of Planendar. Through the various user interactions such as creating or updating an event, we offer a non-intrusive method of sharing their cool stuff with their friends. Our Open Graph queries serve as our best promotion tool, sending customized information that tells more about our application than the Like button.

**Aspiration 11:** Explain how you handle a user's data when he removes your application. Convince us that your approach is necessary and that it is the most logical. Do also include an explanation on whether you violate Facebook's terms and conditions.

The user's current data will still be stored in our database upon removal of our application.

From the user perspective, Planendar is meant to be social. Sharing is encouraged and we expect the links to the user's calendar to be available to his friends. As such, even if he removes our application, his friends should still have access to his shared links.

The user can remove his data from our database by explicitly informing us that he wishes to delete all his data.

## Facebook's terms and conditions

We do not violate Facebook's terms and conditions. Facebook's data use policy states that users should contact the application to delete their information if they want. (https://www.facebook.com/about/privacy/your-info-on-other)

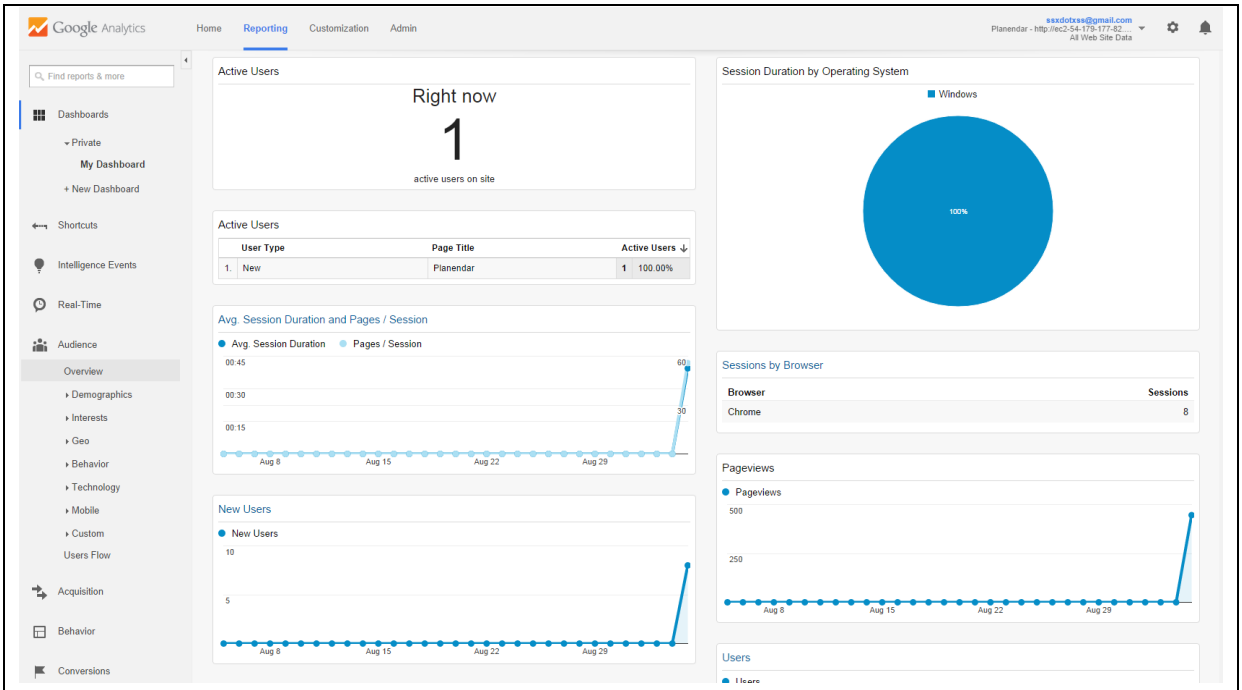
Under Facebook platform policy (https://developers.facebook.com/policy), we must comply with local laws and regulations, which would refer to the PDPA.

## PDPA

We also do not violate the PDPA (and by extension, Facebook's terms and conditions).

We will not use the data for any other purposes except what the user has consented to. For example, we will not share any links to anybody, and after the user has removed Planendar, the only people with access to the user's data would be those that has the link already.

**Aspiration 12:** Embed Google Analytics on all your pages and give us a screenshot of the report.




http://54.179.177.82/images/ganalytic.png

**Aspiration 13:** Describe 3 user interactions in your application and show us that you have thought through those interactions. You can even record gifs to demonstrate that interaction! It would be great if you could also describe other alternatives that you decided to discard, if any.

## Interaction 1 – Drag and Drop

For easy updating of events, user can drag events around the calendar and even to the tasks bar! This drag and drop functionality aims to improve user-friendliness of modifying events, and ultimately, encourage users to use Planendar more by placing tentative events in as well.

DragnDrop.gif

## Interaction 2 – Create Task or Event

Users can create tasks or events on Planendar. Typically, one will add 2 buttons (clutter) or add an extra field in the Creation form (extra click). We chose neither.

We hide the 2 buttons from the user when unnecessary in a dropdown menu that only shows itself when the user needs it.

Create.gif

### Interaction 3 – Mouseover for Details

We believe that it should be easy to view details about your events at a glance. Just hover over each event and the user can immediately view all information about their event.

Hover.gif

**Aspiration 14:** Show us an interesting DOM manipulation through jQuery that occurs in your application.

```
// To vary min-height for calendar on load or resize
  $(window).on('load resize', function(){
     $('.minheight').css( "min-height", function( index ) {
       var index = $(window).height();
       return index - 120;
       });
  });
```

Using 3 jQueries, ".on();" ".css();" ".height()", we set a minimum height for our calendar such that the footer really stays at the bottom even when you resize it.

## Phase 5: Young Adult

**Aspiration 15:** Implement at least one Story involving an Action and Object in your application. Describe why you think it will create an engaging experience for the users of your application.

### Create a Calendar

This is the first Story user will encounter from the use of Planendar. It is posted whenever the user creates a task or event on Facebook.

While there is no differentiation now, when fully implemented with unique links to their calendar, it will tell their friends automatically whenever an event is created on their calendar.

### Update a Calendar

This informs the user's Facebook friends whenever he changes any details on his calendar.

When fully implemented, users will automatically tag their invited friends in this Story update to keep everyone up to date with their meeting dates.

## Optional Aspirations

**Aspiration 16:** Describe 2 or 3 animations used in your application and explain how they add value to the context in which they are applied. (Optional)

### Taskbar as a Delete button

When users try to drag events around, they will notice that the taskbar turns red and a rubbish bin logo appears.

This is an animation to make apparent that they can delete their event by dragging it to the rubbish bin as and when they wish.

### Agenda/Week/Month Tabs

Clicking each of the tabs will change the view of the calendar.

It reduces the clutter in the UI and maximizes the space usage of Planendar

### Informative Popovers

Popovers explaining some usage can be used to tell users about certain functions.

When users first use Planendar, it may not have occurred to them how useful the taskbar may be. The popover seeks to address this by explaining to the users the use of task.

**Aspiration 17:** Describe how AJAX is utilised in your application, and it improves the user experience. (Optional)

AJAX is utilised throughout the program in tandem with AngularJS two-way data binding. When bound data is modified through the UI (dragging/dropping calendar items, creating new events or tasks etc), AngularJS recognizes the data as dirty and, through the IssueControllerService controller, sends an AJAX message to the server to modify and/or retrieve data about the user's events. Instead of using XML to send the data, however, we have chosen to use JSON. This is so that the data can be parsed easily on the client side.

In this manner, the user will experience less page loading time - the request for the user's data can be done while the page has already loaded. At the same time, the user will be able to make multiple changes to the data in real time without waiting for the server's database to respond. The result is a very responsive UI, which in turn makes for a more streamlined user experience.

**Aspiration 18:** Tell us how you have made use of any existing jQuery plugins to enhance your application functionality. Impress us further by writing your own reusable jQuery plugin. (Optional)

Semantic UI is the core of Planendar UX.

It has a grid collection that allows for easy formatting of the UI. Semantic UI also has added functionality for Mobile View, which we has also implemented to show a more suitable view on a small screen.

When Semantic UI is insufficient, it is supplemented with Angular.js, which gives our website most of the custom functionalities you see such as Drag and Drop and dynamic HTML.