National University of Singapore
School of Computing
CS3216: Software Development on Evolving Platforms
AY2011/2012, Semester 1

# Assignment 1: **Life of a Facebook Application**

Issue date: 8 August 2011
Due date: **20 August 2011, 23:59 (mid-assignment submmission)**
**1 September 2011, 23:59 (final submission)**

## General Overview

In this assignment, you will learn about the development life cycle of a Facebook Application. You will be guided[1] to write your (probably first) Facebook application. Before we begin, you might want to decide which programming language to use. If you have not done so, you are encouraged to login into and explore your AWS account. You should have set up PHP (or the programming language of your choice) and MySQL and be able to access your content online. Remember, these may take a day or two to get right. You definitely do not want to leave it to the last week.

We **strongly** suggest PHP; it might not be the nicest language, but it is certainly easy to learn and easy to use with Facebook.

This assignment is highly open-ended. We provide milestones (in the form of aspirations) so that we can grade your application in a consistent way, even though everyone would be building different apps. This assignment is also designed to introduce you to the various elements of Facebook application development and the milestones are there to ensure that you learn about the elements in a structured way.

However, you have a lot of freedom to express your creativity. You are free to develop any application you choose. However, if some of the proposed milestones do not make sense for the application you intend to build, please submit a petition at least one week before the assignment is due. Your petition may or may not be approved. Furthermore, while the aspirations may be easy to meet, simply meeting them will not gain you maximum credit. We expect quality submissions, not run-of-the-mill work.

The high-level goal of this assignment is simple: a working Facebook application that is able to store persistent users' state and that allows users to interact with each other. Please read through the whole assignment once to get a sense of what is required before starting work on it. A detailed grading scheme is attached at the very end.

Yes, we know you could flip to the last page right now to see it, but hopefully putting it at the end will make some of you read the entire assignment first before worrying about the grading.

### A Word of Caution

Before we begin, there's something you should know. Facebook is a very dynamic platform and things are liable to change at any time. As you read this, Facebook's

[1] guide: to direct, supervise, or influence usually to a particular end. (Merriam-Webster) So expect to spend some time exploring on your own.

engineers are changing the way their SDKs (Software Development Kits) work. More specifically, they are adopting the OAuth 2.0 protocol, which affects how users authenticate with Facebook and how applications retrieve data from their platform.

These changes have implications for the code examples we have provided and to some of the information included in this write-up. We tried to reduce your workload by converting all the examples to use the latest protocol but there are bugs and issues with it which Facebook has not fixed. Fortunately, developers have until the 1st of October, 2011, to migrate to the new protocol.

Therefore, in this assignment, we will continue using the older version of the protocol. Your assignment submission should not be affected since it is due on September. However, your tutors will continue to keep an eye on Facebook to see how it evolves. If necessary, we will release another document later in the semester to detail the changes and to keep you in the loop.

Think of this as an opportunity to work with an actively changing platform; there are valuable lessons to be learnt on project and risk management.

With that out of the way, let's begin...

| | |
|---|---|
| **Reminder:** | Please read the entire assignment before starting. |

## Phase 0

> *"If you wait to do everything until you're sure it's right, you'll probably never do much of anything."*
>
> —Win Borden

After meeting with the wonderful fellow students in CS3216, you finally decided that it is time to do something. You have decided to make your inspiration a reality! While you are at it, we are here to guide you.

To embark on this journey, you have to first install the Facebook Developer application.

| | |
|---|---|
| **Aspiration 0:** | Install the Facebook Developer application from `http://developers.facebook.com/apps`. *(Not graded.)* |

**Note**: During this process, Facebook may require you to verify your identity and confirm that you are not a spammer. You can choose to do so by either entering

a one-time code, which they will send you via SMS, or by entering your credit card number.

Next, you have to decide what kind of application you want to create. You can choose to create: (i) an IFrame application, (ii) a standalone web application that connects with Facebook, or (iii) both! An IFrame application is one that loads within the main Facebook website while a standalone application is a separate website that connects to Facebook to retrieve information about users and post on their walls. An example of the former is *CityVille*[2] and an example of the latter is *JFDI Academy*[3], the app used by students in CS1101S.

If you have a good reason to, you could even develop both types of apps for your project. However, do note that execution matters! You do not automatically get extra points for choosing to do both. If you choose to develop both an IFrame and a standalone application, and if your execution is poor, you will get lower grades compared to a team that only developed one type of application and did it well. You have been cautioned! (But of course, if you decide to do both and your execution is fantastic, you can expect a better grade).

| | |
|---|---|
| **Aspiration 1:** | Choose to do an IFrame application or a standalone application or both. Choose wisely and justify your choice to us with a short write-up. |

To begin the journey of your new Facebook Application, go to the Facebook Developer page and click the button that says “Create New App” located at the top right.

Before you do anything else, click on the link that says “Facebook Terms” (the Terms of Service: `http://www.facebook.com/terms.php`) and read the terms of service. No, we mean it, read them! You are going to write an application for Facebook. Some of you may even decide to monetize your applications in the future. It is a very good idea to read and understand the terms of service before you write something that is inappropriate. Remember, if you violate any of the terms, you risk having your application deleted.

(Scenario: imagine you have just finished your final project, you have even reached that coveted 1 million eyeballs that would have given you an easy A+, yet, on the day before project submission, you received an e-mail from Facebook telling you that your application breached its ToS and had to be deleted... we do not need to tell you the rest of this story, do we?)

## Phase 1: Baby

*“Babies are always more trouble than you thought – and more wonderful.”*

—Charles Osgood

[2] `http://apps.facebook.com/cityville/`
[3] `http://jedi.ddns.comp.nus.edu.sg`

After you have read the ToS, it is time to name your Facebook application. By now, you should already have an idea of what your application will do, so pick a reasonable name.

Out of ideas? Check out http://www.whatalovelyn :p

| **Aspiration 2:** | Your new baby needs a name! Give it one! *(Not graded.)* |
|---|---|

Once you have a name for your new application and once you agree to the Facebook Terms (you have read it, haven't you?), click on "Continue". You may need to complete a captcha before Facebook creates your application.

Next, you need to provide Facebook details about your application. Explore the categories on the left to see what kind of details are required. If you need information about any of the fields, simply move your cursor over the **[?]** symbol next to the field names. If you make changes to any of the fields, do remember to click on "Save Changes" once you are done.

Let's go through some of the more important fields now:

- **About** > **Basic Info** > **App Info:**

  **App ID:** This uniquely identifies your application on Facebook.

  **App Secret:** This is the key with which you would access the Facebook APIs within your application. Do NOT share this with anyone. If you do, they can use it together with your App ID and make API requests without you knowing. Reset your App Secret key immediately once you have reason to believe that it has been compromised.

- **About** > **Basic Info** > **Basic Info, App Images, Contact Info:**

  These sections essentially describe your application. These fields would be important when you are working on the next aspiration. We leave it to you to read the descriptions of the various fields before filling them up. However, of particular note is the **Privacy Policy URL**.

  Facebook requires that you create a privacy policy, specifying what kind of details you collect about your users and what you intend to do with those details. You then have to provide a link to this privacy policy in the *Privacy Policy URL* field. We have attached a privacy policy template in the appendix for you to use with your apps. You could also write your own if you wanted to. For another example of a privacy policy, see the Facebook Privacy Policy at http://www.facebook.com/policy.php.

- **About** > **Roles** > **Developers:** Add your teammates as developers here[4].

- **About** > **Advanced** > **Authentication:**

  **Deauthorize Callback:** The URL that Facebook would ping when a user removes your app and deauthorizes it. See the **App Deauthorization** section in http://developers.facebook.com/docs/authentication/.

[4] You can give different users different roles. For an explanation of these roles, see http://developers.facebook.com/docs/ApplicationSecurity/.

**Sandbox Mode:** Developer mode, you may want to enable this option for now (Disable it before you publish your app).

- **About** > **Advanced** > **Migrations:**

  **OAuth Migration:** This indicates if we're ready to migrate to the new OAuth protocol. Make sure you set this to `Disabled`.

- **(For Standalone Apps) Web** > **Web** > **Website Settings:**

  **Site URL:** This is the URL where your standalone app is located (such as `http://ec2-174-129-70-144.compute-1.amazonaws.com/some-dir/`). If you do not fill up this field, your standalone app would not be able to use the Facebook API.

  **Site Domain:** This specifies the domain your application lives in. Facebook uses this a security feature to make sure the user is authenticated correctly and is directed to the right application. If your Site URL is `http://ec2-174-129-70-144.compute-1.amazonaws.com/some-dir/`, then your Site Domain would be `ec2-174-129-70-144.compute-1.amazonaws.com`.

- **(For IFrame Apps) On Facebook** > **Canvas Settings** > **Canvas**

  **Canvas Page:** This is your application page (you know, those `http://apps.facebook.com/some-app`). Use something that is both related to your app and easy to remember.

  **Canvas URL:** The URL that will be accessed by Facebook and shown to users when they go to the canvas page. If you are using AWS, your callback url should look something like this **`http://ec2-174-129-70-144.compute-1.amazonaws.com/some-dir/`**. You should not refer to a page directly (e.g. `http://[your_host]/some-dir/index.php`) because if the user attempts to access the page `http://apps.facebook.com/some-directory/sayhello.php`, Facebook will attempt to access `http://[your_host]/some-dir/index.phpsayhello.php`, which is not what you would usually want.

  **Secure Canvas URL:** From the 1st of October, Facebook requires you provide a link to direct users to when they visit your IFrame app using HTTPS instead of HTTP. You can leave this blank for now while we figure out the logistics of providing SSL certificates.

- **(For IFrame Apps) On Facebook** > **Canvas Settings** > **Discovery:**

  **Social Discovery:** If you enable this, Facebook with help you promote your application by automatically publishing feeds on your users' profiles. This will help their friends to discover your app. These feeds are published when users install your app or when other such key events occur. In most cases, you want this feature to be enabled.

- **(For IFrame Apps) On Facebook** > **Canvas Settings** > **Page Tabs:**

  Facebook provides a feature to allow your application to be used within the context of a Facebook Page. If you use this feature, anyone who creates a Facebook Page can integrate your app into their page. This is an optional feature and we won't dwell on this any further. If you want more information, see `http://developers.facebook.com/docs/guides/canvas/#tabs`.

Remember that you can find out more about each of the fields by moving your mouse cursor over the **[?]** symbol. Do also consult the documentation at http://developers.facebook.com/docs

| | |
|---|---|
| **Aspiration 3:** | Give your newborn application some love. Go to the **About** category and fill the page with as much appropriate information as you can. And yes, we expect a cute application icon! |

We've finally gotten that out of the way. Did you remember to click "Save Changes"?.

## Sidetrack: Software development models and Facebook

Okay, you are now perched on the threshold of the birth of your amazing Facebook application, but before you get down and dirty, we'd like to take some time to talk you through some higher level things that you should keep solidly in mind during your Facebook development experience.

A software development model is a model of the process through which your software (in our context, your Facebook application) gets developed. A common model you may have heard of is the **Waterfall model**[5], where your team goes through a sequence of well-defined stages from planning and design, through development and testing and finally to release in one full linear process. What we'd like to drill into you at this point, before you actually start laying out your plans, is this:

**DO NOT do Waterfall**.

Specifically, this means don't start with drafting out a large and ambitious 'mother-of-all-designs' and dividing the rest of your time into slowly and methodically implementing that design. This is why Facebook is interesting - the platform is such that traditional software development models (like Waterfall) do not work well, because applications on Facebook tend to be relatively small scale, and tend to enjoy very intimate and rapid interaction with their users.

In such an environment, a much more sensible model is **rapid deploy** and **incremental improve**. This means you start off by rapidly completing a small but workable application with your key ideas in place, then push it out immediately for users to try it out. Then keep an open channel of communication with your users (use Facebook's forum functionality and your application page's wall and Facebook's user feedback/rating mechanism). Note that this does not mean just sitting there and reading your users' responses but interacting with them, asking them for suggestions and feedback, and actively responding to their issues and fixing bugs they report. Then, *incrementally improve your application along the direction of your user feedback*. Your users know best whether they are happy with your app, so if they tell you that this feature is difficult to use or that having such and such a functionality would be nice, you better be busy taking notes and incorporating them into your application's roadmap.

This is part of the story of we want you to experience while developing on Facebook – a unique and successful relationship with real users.

[5] See http://en.wikipedia.org/wiki/Waterfall_model

## Phase 2: Toddler

> *"If you give a hacker a new toy, the first thing he'll do is take it apart to figure out how it works."*
>
> —Jamie Zawinski

Creating Facebook applications is similar to developing regular web applications. This was not the case a year ago, where students had to grapple with concepts such as FBJS (Facebook Javascript - a sandboxed version of Javascript). Now you have to grapple with different things and at the rate Facebook is changing, there'll be newer things to learn in a few months. The Facebook environment is very dynamic and things change often; be prepared to learn and re-learn (and curse like a sailor when things break).

The first thing you should be aware of is the types of Facebook APIs. There are two types currently: Graph and FQL. As we will explore them in greater detail later in this assignment, we won't go through their differences now. Just be aware of their existence.

The next thing you should be aware of is the SDKs (Software Development Kits) Facebook provides: the PHP SDK and the Javascript SDK. These SDKs help in integrating your app with the Facebook platform and in using the APIs. You can use either one for your apps but if you want to use both at the same time, it is easier to do so in IFrame apps than in standalone apps[6].

The PHP SDK makes API calls from the web server your application is hosted in while the Javascript SDK makes API calls from the web browser. The question you should immediately ask is: "Why should I care where the API calls are made from, so long as they are actually made?".

To begin answering this question, we need to understand how a web request cycle works. When a user loads your app, a web request is sent to your server. Your server picks up the request, figures out what the user is looking for and executes the appropriate PHP script. During execution, the script may make API calls and wait for the responses. Once Facebook sends all required API responses, the script finishes execution and the final result is sent to the user.

Let's suppose that each API call takes 300 ms on average to complete, especially because your web server is located far away from Facebook's API servers. Now, if your PHP scripts make ten API calls, one after another, the total execution time would be at least 3 s. Therefore, each time your user loads your web application or clicks on a link, she has to wait at least 3 s for a response. Do you see where this is going? The user will start complaining that your app is "laggy" and she may quit using your app altogether.

You have several ways of remedying this issue. Firstly, you can optimize your scripts so that you don't make redundant API requests. For example, if you use a result from an API call in a loop, you could make the API call before the loop, cache the result and use that in the loop instead of making the API call within the loop. Secondly, you could modify your application, remove less-important features and only send API calls that are absolutely necessary for your application to function.

[6] After reading this handout, take a look at the addendum handout available in IVLE. It describes the issues of using the PHP SDK with standalone apps.

The third alternative is use the Javascript SDK and make the API calls from the user's browsers. For example, you may want to display the user's profile picture in your application. Instead of retrieving the link to the picture using the PHP SDK, you can retrieve the link with the Javascript SDK. This way, your web server can send responses to users quicker since it makes less API calls.

By now, you should have figured out the answer to the question earlier: performance. Your applications may gain in performance by offloading some API calls to the Javascript SDK. In general, application performance is a complex issue and optimizations vary from case to case but it does not hurt to be informed. Do note however that as Donald Knuth[7] would say, "premature optimization is the root of all evil" – focus on the features first without being overly worried about performance.

In this phase, we will learn how to create IFrame and standalone applications that integrate with Facebook. We will be using both PHP and Javascript SDKs. As stated earlier, detailed discussion on the API would be left to a later section of this assignment.

### Facebook PHP SDK

The attractiveness of using PHP to develop Facebook applications is due to the PHP SDK. It lets you access both types of the APIs and returns the results of API calls in nice PHP arrays.

Download the PHP SDK from: `https://github.com/facebook/php-sdk/`. Once you finish downloading it, extract the archive into any directory and you should see the following file structure:

```
facebook-php-sdk/
   | changelog.md
   | readme.md
   + examples
   - src
     | base_facebook.php
     | facebook.php
     | fb_ca_chain_bundle.crt
   + tests
```

The files you really need are located in the `src` directory. It contains the main library files and the certificate that the SDK uses to send API requests via SSL. The `tests` directory contains some scripts that verify the SDK's functionality and the `examples` directory contains an example of how the SDK is used. You don't have to explore them now because we will soon show you how you can use the SDK. `readme.md` and `changelog.md` are merely text files that explain what the SDK is and what has been updated in the SDK since the last version. You can read them later.

Now, go to your application directory on your web server. Create a new directory named `php-sdk` and copy all the files from the `src` directory of the Facebook PHP SDK to new directory. The final layout should look something as follows:

[7] `http://en.wikipedia.org/wiki/Donald_Knuth`

```
your-app-dir/
   - php-sdk
     | base_facebook.php
     | facebook.php
     | fb_ca_chain_bundle.crt
```

You are now ready to use the PHP SDK in your application. But let's explore the Javascript SDK first.

### Facebook Javascript SDK

The Javascript SDK is used in standalone applications mainly for its authentication mechanism. It allows users to login to your app using their Facebook accounts. You don't need it in IFrame apps since Facebook handles user authentication directly in IFrame apps. As highlighted earlier, the Javascript SDK is also used to make API calls from the client's browser.

If you wish, you can download the Javascript SDK source from: https://github.com/facebook/connect- You can view the internals of the SDK and look at examples of how you can use the SDK with other Javascript libaries such as JQuery (this might prove to be handy in the later portion of this assignment).

However, to actually **use** the Javascript SDK, you do not have to download anything manually. Instead, you only need to add the following chunk of markup to your page to use it.

```
01: <div id="fb-root"></div>
02: <script type="text/javascript" src="http://connect.facebook.net/en_US/all.js"></script>
03: <script type="text/javascript">
04:   FB.init({
05:    appId  : 'YOUR APP ID',
06:    status : true,  // check login status
07:    cookie : true,  // enable cookies to allow the server to access the session
08:    xfbml  : true,  // parse XFBML
09:    oauth  : false  // disable OAuth since we're not ready for it
10:   });
11: </script>
```

The `div` tag in the first line is important because the Javascript SDK uses it to perform some initialization. The id of that `div` has to be *fb-root*. It is in the second line where we actually include the Javascript SDK. This is an external Javascript file which the browser will download from Facebook directly before executing it. Once that is done, you will gain access to the `FB` object. This is the main object defined by the Javascript SDK and you will need it to access **any** of the SDK's functionality.

For example, to initialize the SDK, you need the code from lines 4 to 10. Here, you're calling the `init` method of the `FB` object[8].

As you can see, when calling `FB.init`, you need to provide certain parameters. You can see all the possible parameters in the documentation but we'll go through some of them here:

[8] Documentation on `FB.init` is available at https://developers.facebook.com/docs/reference/javascript/FB.init/

- **appId**

  Your application id, provided by Facebook. Note that unlike the PHP SDK (which you'll see in action later), you do not specify your secret key here because if you did, anyone would be able to retrieve it simply by viewing the source of your page.

- **cookie**

  Set this to `true` to ask the SDK to create cookies on the user's browser when she is logged in to Facebook. These cookies will then be used by both the PHP and Javascript SDKs to obtain the necessary authorizations to make API calls.

- **xfbml**

  Some of the common elements you see on Facebook, such as the Login button and the Like button[9], can be created with Facebook's custom markup language called XFBML. By setting this parameter to `true`, you're asking the SDK to convert all XFBML tags to the appropriate HTML form so that the elements can be displayed on the browser.

- **logging**

  Set this to `true` if you want the SDK to log debug messages and if you want to use the `FB.log` method[10] to log your own messages. These debug messages will show up on your browser's Javascript console if it has one. On Firefox, you can use the Firebug extension to view the console. Google Chrome has one built-in.

Don't worry too much if you don't completely understand how this works. We're going to start playing with some IFrame and standalone applications and you'll figure everything out soon enough.

## Integrating IFrame Apps

It's finally time to get down and dirty with code. Let's begin integrating an IFrame application with Facebook. Go to your application directory on your web server and create a new file named `index.php`. Your app directory should look as follows now:

```
your-app-dir/
   | index.php
   - php-sdk
     | base_facebook.php
     | facebook.php
     | fb_ca_chain_bundle.crt
```

Now, open the file `iframe_app.php`. It's located in the assignment 1 zip file. Copy that code and paste it into index.php.

The code should be self-explanatory. There is a small section in the code that deals with permissions which you have not learned about yet. We'll explore it later when

[9] These are also known as Social Plugins. See: https://developers.facebook.com/docs/plugins/

[10] FB.log is currently undocumented but it's very simple to use: `FB.log(''Debug message'')`.

we visit the APIs. Once you place your app id, secret key, and canvas URL in the code, visit your application and you should see the text "Hello World". You should also see your name and some information about you. If it does not work, make sure that you have configured your app correctly on Facebook's developer application.

This sample also shows how easy it is to use both SDKs in IFrame apps. This should be enough to get you started. Let's look at standalone apps next.

### Integrating Standalone Apps

In this section, we'll learn how to write simple standalone apps. Create another application directory, and create `index.php` in the new directory. Next, open the file `standalone_js_app.php` and paste the code there into index.php. Don't be afraid of its length. It's only longer than the IFrame version because we are required to construct the entire web page without any help from Facebook.

Try it out now. You may want to see what happens when you log in and log out from the application.

The comments in the code should help you figure out what's going on. But there are two things we want to highlight. Firstly, we are using only the Javascript SDK here. If you want to know how to use the PHP SDK with standalone apps, do look at the addendum to this assignment. It's in IVLE. This addendum is just a guide and does not contain any aspirations. Read it after you finish reading this document.

The next thing we want to highlight is educational. When creating a web application, you may often find that you need to control the visibility of certain elements. For example, you wouldn't want to show the "login" link when the user is already logged in since that would just confuse the user. In our sample app, we demonstrate two common techniques of accomplishing this. The first technique is to use Javascript and CSS. Look at the login and logout link. By default, their CSS properties are set so that they are invisible. Then, we ask the Javascript SDK if the user is logged in. When we get our response, we make the appropriate link visible. The second technique is to use PHP. Look at PHP code that displays the Login button only when it detects the user is not logged in. There are differences between both techniques and we could just state what they are, but that's no fun. So we leave it to you to figure out and tell us.

| | |
|---|---|
| **Bonus:** | What are the pros and cons of each method of visibility control? When should one use the Javascript method and when should one use the PHP method? (1%) |

**Aspiration 4:** Integrate your application with Facebook. If you are developing an IFrame app, then users should be able to visit your app and at least see their name (retrieved using the API) on the page. Similarly, if you are developing a standalone app, users should be able to login to your app using their Facebook account and see their own name appearing.

## Phase 3: Child

> *You can learn many things from children. How much patience you have, for instance.*
>
> —Franklin P. Jones

Now that you are familiar with the basic Facebook building blocks, it is time to graduate to the more complicated toys. In this phase, we will learn how to store persistent data! To be more precise, we are going to store data about our application's users in a MySQL database.

Well, actually any SQL-based database system will do.

### An overview of relational databases

A relational database is a type of database that models stored data as tables with columns and rows. It is called "relational" because you can link a table to another table through *foreign keys*.

In this section, we will be going through simple relational database concepts. There are some other more advanced concepts that we'll go through in the SQL workshop.

A database application may store several *databases*. Hence, while each application will usually use its own database, several applications may share the same database application running on the same server (e.g. if you and a friend each have a blog, even if each blog needs 1 database you could still house both blogs on the same MySQL instance).

Visualizing a database at highest level, we think about a *schema*, which is basically a blueprint of the database's tables, their structural details and the relationships between them. Some database applications allow multiple schema in a single database, MySQL only allows one, and one is really enough for our purposes, so we won't go further on that.

No, in fact, we will never discuss schema again after this.

Within a schema resides two things: *tables* and *relations*. Tables contain one or more columns each. For example, we can imagine a `students` table containing 5 columns: `matric_no`, `name`, `address`, `phone`, `birthdate`. Each column has a *type* that you need to specify (e.g. `name` is of type text, `birthdate` is of type date). Actual data will then be simply stored as rows in the table. Each row needs to be uniquely identifiable. If two rows happen to be completely identical, you will

run into trouble trying to update or delete them since there is no way to pinpoint exactly which one you mean. Thus, we usually have a column (or a set of columns in combination) that we require to be unique for each row. We call this the *primary key*. In the `students` table example, the `matric_no` column is an excellent candidate for primary key since no two students share the same matric number. MySQL (and any other proper database system) will prevent you from inserting a row if there already exists another row with an identical primary key.

Relations indicate relationships between tables. For example, suppose we add a `home_faculties` table containing two columns: `matric_no` and `faculty` - a simple mapping of student to faculty. We can link this table to the previous `students` table using the `matric_no` column, which both tables share. We say that `matric_no` in the `home_faculties` table is a *foreign key* that *references* the `matric_no` column in the `students` table. Note that a foreign key column set must reference a primary key column set of another table. Note also that our two-table setup allows students to become members of two faculties (eg. when doing double degrees).

This is a small space to attempt a proper introduction to relational database concepts. For better or further understanding, you might want to look up additional descriptions online. Eg. at Wikipedia. Oh, and the workshop may help if you have not already used MySQL.

| | |
|---|---|
| **Bonus:** | What is the primary key of the `home_faculties` table? (0.5%) |

Other important concepts include *indexed columns* (that makes searching within a column really fast), *unique keys* (enforce uniqueness for non-primary key columns), and relations cascading (where deleting a student from `students` table can automatically update/delete all entries in other tables that reference this table). We may cover these in the workshop.

After this section (and maybe after sitting through our awesome introductory MySQL workshop), you should be ready to produce a schema for your application. Do consider how efficient your schema will be. Specifically, think about the number of queries required to accomplish common tasks and the number of tables accessed to complete a single user query. Your schema should be graphical, and should indicate clearly the table names, column names/types, primary keys, and relationships.

You should remember that as a rule of thumb, database schemas should be planned with a **design once use forever** principle in mind. You should spend a good amount of thinking on a good schema design, after which you should almost never need to touch the design again.

No idea how to draw it out? Think simple. Annotated boxes and arrows are fine - just make sure the design is clearly communicated. Of course, we also accept proper entity-relationship diagrams.

| | |
|---|---|
| **Aspiration 5:** | Draw the database schema of your application. |

## SQL: Querying the database

SQL is a standard language designed to manage a database and to retrieve or store data in a database. In addition to SQL, most database systems will have several

additional SQL-like commands that are used to perform specific administrative tasks like adding new users or modifying passwords.

The MySQL workshop docs provide details on commands you can use to create and alter databases and tables, and also commands you can use to insert, update, delete, and retrieve rows from tables. We call the former 'data definition language' (DDL), and the latter 'data manipulation language' (DML). You should **never** ever call DDL from publicly accessible pages (that includes your application pages that can be accessed from Facebook).

## Accessing MySQL from PHP

In PHP, `mysql_connect` is used to connect to a database, and `mysql_close` is used to disconnect from a database. For example:

```
<?php
  // Open a database connection
  $con = mysql_connect("db_hostname", "userid", "password");
  if (!$con) {
    die('Could not connect: ' . mysql_error());
  }

  // Some code to access the database and for processing
  mysql_select_db("db_name", $con);
  mysql_query("SELECT OR OTHER QUERY HERE", $con);
  ...

  // Close the database connection
  mysql_close($con);
?>
```

MySQL queries are handled using the command `mysql_query(query string, db variable)`[11].

Note that the example above is far from optimal; if the code that access the database throws an error, the connection will not be closed.

If the language of your choice supports RAII (Resource Acquisition Is Initialization) pattern, please use it.

> **Aspiration 6:** Tell us some user queries (at least 3) in your app that needs database access. Provide the actual SQL queries you use and explain how it works.

## Facebook API: Graph

It's time to begin discussing the Facebook API. We'll start with the Graph API. At the core of Facebook is the social graph[12], a data structure where objects on Facebook (such as posts, comments, images, user profiles, and fan pages) are uniquely

[11] `mysql_query` and the rest of PHP's MySQL API can be found documented here: http://php.net/manual/en/book.mysql.php

[12] Just in case you are not aware, the name "graph" comes from the mathematical concept. See http://en.wikipedia.org/wiki/Graph_(mathematics).

identified. Objects have connections between them if they have some relation. For example, a user in the social graph would be linked with the posts she makes. The Graph API is an attempt by Facebook to expose this social graph to developers in a clear and consistent manner.

The key idea you have to internalize is that every object in the graph is associated with at least one unique identifier. Some objects have more than one identifier. For example, you could refer to a user using either her Facebook id or using her Facebook username. However, each identifier is associated with one (and only one) object. You would never encounter a case where a single Facebook id is associated with two different users.

When you pass an identifier to the Graph API, it will return some basic information about the object. The exact information you get depends on the kind of object the identifier is referring to. Thanks to the design of the Graph API, retrieving additional information about an object or retrieving related objects is a simple case of extending your query in a natural manner.

Let's look at some examples. We'll begin by trying the Graph API on our dear Prof. Ben. Try visiting the following URL in your browser: `http://graph.facebook.com/benlwl`. Your should see something as follows:


```
{
   "id": "642742165",
   "name": "Ben Leong",
   "first_name": "Ben",
   "last_name": "Leong",
   "link": "http://www.facebook.com/benlwl",
   "username": "benlwl",
   "gender": "male",
   "locale": "en_US"
}
```


Facebook has returned some basic information about Prof. Ben in the JSON format[13]. If you need only certain fields, then you would add the `fields` parameter followed by a comma-separated list of fields you are interested in. For example, if you only want the name of Prof. Ben and the link to his profile, you would visit: `http://graph.facebook.com/benlwl?fields=name,link`. This would return you the following:


```
{
   "name": "Ben Leong",
   "link": "http://www.facebook.com/benlwl",
   "id": "642742165"
}
```


Information about users is not the only thing available with the Graph API. Try visting the following URL for information about the CS3216 Facebook group: `http://graph.facebook.com/216503318372716`. You should be able to see information such as the owner of the group (who, incidentally, is none other than our dear professor). The question to ask now is, how did I get the identifier of the CS3216 group? Well, I searched for it using the Graph API. You can search the Facebook platform with the following query:

[13] JSON is a compact format for representing data and exchanging them between different services. See `http://en.wikipedia.org/wiki/JSON`. You'll be using them in the second assignment too.

`http://graph.facebook.com/search?q=DESCRIPTION&type=OBJECT_TYPE`.
To search for the CS3216 Facebook Group, you would use this search query:
`http://graph.facebook.com/search?q=CS3216&type=group`.

```
{
   "data": [
      {
         "version": 1,
         "name": "CS3216",
         "id": "216503318372716"
      }
   ]
}
```

As there is only one group named CS3216, there is only one result from the search. The id specified there is the identifier of the CS3216 group.

Now, information about an object alone is all nice and well but sometimes we want to retrieve other objects related to a particular object. In such cases, instead of using the `fields` parameter, you would simply append the name of the related objects to your Graph API query. For example, if you want to look at user reviews of CityVille, you would visit `http://graph.facebook.com/291549705119/reviews`. Observe the `/reviews` appended to the end.

The Graph API is very extensive and we cannot completely explore it here. You are **strongly** encouraged to look at the Graph API documentation located at `http://developers.facebook.com/docs/reference/api/`. The documentations clearly lists all the information you can retrieve with the Graph API. Also, try playing with the Graph API explorer at `http://developers.facebook.com/tools/explorer`.

There is one final note we wish to make before moving on. So far, in this section, we've only accessed the Graph API via our browser. But you would be using the PHP or Javascript SDKs when developing your applications. When accessing the Graph API using the SDKs, just specify the identifer directly and optionally, add the *fields* parameter; you do not have to add `http://graph.facebook.com`.

For example, here is how you would retrieve information about Prof. Ben:

```
// In PHP SDK
$facebook->api("/benlwl");

// In Javascript SDK
<script type="text/javascript">
    FB.api('/benlwl', function(response) {
        // Do something when the API returns
    });
</script>
```

In closing, remember to look at the documentation and play with the Graph API explorer tool. If you have questions, ask your tutors or post in the IVLE forums.

### Facebook API: FQL

Let's discuss FQL (Facebook Query Language) now, a SQL-like language that you can use to query Facebook's data. It lets you request complicated sets of data in an elegant manner. For instance, if you want to get the pictures of a person's friends, you would use the following query:

```
SELECT pic FROM user WHERE uid IN
  (SELECT uid2 from friend WHERE uid1 = $facebook->user)
```

The query behaves the way you'd expect regular SQL queries to behave. Using FQL, we can minimize the amount of back and forth API calls between the application and the Facebook server (it also reduces the amount of data transferred). We specify exactly what we want and Facebook's servers will process the request. (It is likely that they can do this faster than us. Leave Facebook to do what it's good at: churning social network data).

Working at FQL's level of abstraction, all the data available from Facebook is stored in a series of tables (conceptually similar to SQL tables), such as `user` and `friend`. The complete list of table is available here:

http://developers.facebook.com/docs/reference/fql/

Each of these tables have columns like SQL tables and FQL gives you a subset of SQL syntax to query them. The user table, for example, has columns `uid`, `first_name`, `pic` and so on.

Here's how you would make an FQL query using the two different SDKs.

```
// In PHP
$query_string = "SELECT ...";
$result = $facebook->api(array(
             'method' => 'fql.query',
             'query' => $query_string,
          ));

// Using Javascript SDK
<script type="text/javascript">
var query_string = "SELECT ....";

FB.api(
{
    method: 'fql.query',
    query: query_string
},
function(response) {
    // Do something when the query returns
}

);
</script>
```

An FQL query is restricted to `SELECT ... FROM ... WHERE ...` syntax only. Only 1 table may be specified in the FROM field (i.e. no implicit table joins). Furthermore, the `WHERE` clause must limit the query to a subset of `uid` (you can't specify `WHERE 1` for example). You may use the `WHERE...IN...` form instead to perform pseudojoins. Do note FQL does not support `INSERT` statements, unlike regular SQL. If you want to add data into the Facebook Platform, such as feeds, use the Graph API; FQL only allows you to read data.

WHERE...IN... is actually query in query and not exactly a join.

Refer to the documentation for examples of FQL queries. The best way to learn is to just experiment with these queries in your application.

## Facebook API and Permissions

When we talked about the FQL and Graph APIs earlier, we only went through really simple examples. But if you want to use any of the more powerful features, you need

to get the user's permission. For example, if you want to retrieve the photos uploaded by a particular user, you need to request the *user_photos* permission. If you wanted to know the user's relationship status, you would request the *user_relationships* permission. You can look at all the available permissions at:

`http://developers.facebook.com/docs/reference/api/permissions/`.

You should try your best to only request a small number of permissions that are absolutely necessary. That way, a user is more likely to grant you those permissions. If you go overboard and request too many, the user will hesitate and may just avoid your app altogether. As a developer, you should follow certain codes of conduct. When users give you permissions, they are actually trusting you. Don't betray that trust and don't misuse the power granted to you. And if users don't give you permissions, your app should continue being functional instead of completely breaking down.

Once you decide which permissions you need, simply modify the `fb_perms_required` variable in the sample IFrame and standalone application code to start requesting them from users.

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

## Phase 4: Teen

> *"The ultimate metric that I would like to propose for user friendliness is quite simple: if this system was a person, how long would it take before you punched it in the nose ?"*
>
> —Tom Carey.

Teen phase is filled with socializing and self-importance. We are going to explore the use of feeds to advertise your apps, employ Google Analytics as a measure of self-importance, and polish your apps to improve user experience.

### Feeds

Feeds can be great at keeping users happy and at advertising your application to non-users. Users will love feeds from your application to brag about something they did (like in JFDI Academy, where CS1101S students will publish feeds to announce to the world that they levelled up). Non-users will see your application feeds on their friends' profile and they might get intrigued enough to try out your application.

Unfortunately, feeds are doubled-edged. Use them poorly and you may cause your application more harm than good. Do you remember all those annoying applications that kept spamming you with feeds you don't even care about? Avoid that. If you keep

spamming your users, they will eventually stop publishing anything from your app. Remember, a feed should make sense and add real value to the user experience.

There are two ways to publish feed stories. The first way is to use the Graph API. By issuing a post request to `/PROFILE_ID/feed` with the appropriate parameters, you can directly publish a feed to the user's profile[14]. You will need the `publish_stream` permission to use this API. This is how you would call the API using the PHP SDK:

```
$params = array(
    'message' => 'Post Message',
    'link' => 'http://www.example.org',
    'picture' => 'http://www.example.org/some-image.png'
    'name' => 'Post Name',
    'caption' => 'Post Caption',
    'description' => 'Post Description',
);
$post_id = $facebook->api("/me/feed", "POST", $params);
```

The second way of publishing a feed story is more polite. In this method, you will show her a feed dialog and provide her the choice to either publish the feed or ignore it[15]. The user also has the ability to customize the message in the feed. Another benefit with this method of publishing feeds is that you do not request the `publish_stream` permission (or any other permissions). Here's some sample code of how to pop up the feed dialog to the user using the Javascript SDK[16]:

```
<script type="text/javascript>
FB.ui(
  {
    method: 'feed',
    name: 'Feed dialog name',
    link: 'http://www.example.org',
    picture: 'http://www.example.org/example.png',
    caption: 'Feed caption',
    description: 'Feed description',
    message: 'Feed message'
  },
  function(response) {
      if (response && response.post_id) {
          // Post published
      } else {
          // Post not published
      }
  }
);
</script>
```

**Aspiration 8:** We want some feeds! **BUT** remember to put thought into this. Nuisance feeds will not only earn you no credit but may incur penalization!

[14] See https://developers.facebook.com/docs/reference/api/user/ for more information.

[15] Documentation for this method is located at https://developers.facebook.com/docs/reference/dialogs/feed/

[16] Sample code adapted from https://developers.facebook.com/docs/reference/javascript/FB.ui/

## The Like Button

One of the interesting things that Facebook has inflicted upon the world is the Like button. It became very popular shortly after it was released. Ordinary people use it to express their interest in something and to share cool stuff with their friends and companies use to it to promote their products and services. We'll show you how to place Like buttons in your app now. Then it'll be your job to think about using them in creative and meaningful ways.

The easiest way of adding a Like button is to use the `fb:like` tag. This is how you would place it in your app:

```
<fb:like href="http://www.example.org/the-link-to-like"
    width="450" show_faces="true" font=""></fb:like>
```

If you want to use the `fb:like` tag, you need to use the Javascript SDK. Alternatively, you could use the following:

```
<iframe src="http://www.facebook.com/plugins/like.php?app_id=YOUR_APP_ID&
                href=THE_LINK_TO_LIKE&layout=standard&
                width=450&show_faces=true&action=like&
                colorscheme=light&font&height=80"
        scrolling="no" frameborder="0" style="border:none; overflow:hidden; width:450px; height:80px;"
        allowTransparency="true"></iframe>
```

This method loads an IFrame (not to be confused with the Facebook IFrame app) containing the Like button.

Do refer to the documentation[17]. It explains how to customize the Like button and how to also include a Share button to let users easily share things they find cool.

> **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.

## Spreading Your Application

Many applications have a page that lets you invite your friends to use the application. You should have one too. This feature leverages the nature of Facebook and helps increase the number of users you have. You could even reward your users and their friends for sending invitations and for accepting them.

If you are using the Javascript SDK, you can pop up a dialog to the user and ask her to select the friends she wants to invite[18]. If you are not using the Javascript SDK, you can direct the user to a page where she can select the friends to invite[19].

[17] The Like button documentation is located at https://developers.facebook.com/docs/reference/plugins/like/

[18] Refer to http://developers.facebook.com/docs/reference/dialogs/requests/ to see how

[19] See the Requests section in http://developers.facebook.com/docs/guides/canvas/

Once the user is done sending the invites, Facebook will provide you the IDs of those invited.

Note that in both the above examples, Facebook automatically detects whether the user's friends are already users of your application or not and splits them into different categories, making it easy for your users to know who to invite. However, you can also check whether a user's friends are users of your application via the FQL API. Refer to the documentation and find out how[20].

What? You didn't think we'll tell you how to do everything, did you? :)

> **Aspiration 10:** First, include the feature to let users invite their friends to use your application. Next, create a page showing either (a) all friends who are users of your application; or (b) all friends (or some sensible subset) who are not users of your application.

## Facebook and Privacy

Now that you're creating a social application, it's time to talk about a very important topic: privacy of users. Quite clearly, the Internet has become a huge part of people's lives. These people usually have a reasonable expectation that their personal data is safe and won't be shared with anyone else without their consent. When a company fails to meet that expectation, things become ugly.

Facebook has violated the privacy of its users many times in the past. For example, there was an incident where this guy was purchasing a ring for his wife as a Christmas gift. It turns out that the website he was buying the ring from was using the Facebook Beacon, a feature that allowed site owners to publish their users' actions on Facebook easily. When the guy bought the ring, a feed was posted on his wall without his consent or knowledge, announcing his purchase to everyone. Needless to say, there was a massive backlash and Facebook removed the Facebook Beacon feature[21].

Facebook has since adopted more privacy-concious practices. As a developer, you cannot just access your users' data willy-nilly; you need to ask them for permission first. And even if you have permissions, you can't use the data as you please because you are bound to terms and conditions of Facebook.

When a user removes your application, you get notified via the **Deauthorize Callback** URL. The question is, what do you do with the user's data when she removes your application? Do you delete the data or do you still keep it in your database? If you decide to keep the data, are you violating Facebook's Terms and Conditions?

[20] Hint: See `http://developers.facebook.com/docs/reference/fql/user/`

[21] See `http://www.washingtonpost.com/wp-dyn/content/article/2007/11/29/AR2007112902503.html` for a story on this incident

**Aspiration 11:** Explain how you handle a user's data when she 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 condition.

## Google Analytics

Web developers are often obsessed with the usage statistics of their applications. You should be too! Let's use Google Analytics to track usage of your application. It's a very popular tool that offers detailed analysis of your users. It shows you which countries your users are using your application from, what browsers they are using, what their screen resolutions are, and so on.

These information will help you greatly in making your users happy. Knowing where most of your users are from will tell you where you should locate your servers. Knowing the screen resolution of your users lets you determine the above-the-fold screen real-estate (the parts of your pages that are viewable directly without scrolling down, assuming a maximized standard browser window in the chosen resolution). Knowing the breakdown of the browsers used by users would be useful in determining whether you need to spend more time testing your features in those browsers.

To add Google Analytics, start by exploring the Google Analytics website[22] and create an account. You might also want to take a look at the Social Tracking feature for additional metrics on your users[23].

Do note that Google Analytics is not the only tool to get metrics on your users. You can also use Facebook Insights[24], the analytics tool designed to work with Facebook applications. In certain cases, Facebook Insights can provide more information than Google Analytics about your application. We focus on Google Analytics here because it's more widely applicable but we want you to be aware of alternatives.

**Aspiration 12:** Embed Google Analytics on all your pages and give us a screenshot of the report. Note that this means you have to install Analytics at least 48 hours before submission deadline as Analytics only updates the report once per day.

## User experience

A good application, web or otherwise, provides a good user experience. Some companies even perform user experience research to determine whether changes to an

[22] http://www.google.com/analytics/
[23] https://code.google.com/apis/analytics/docs/tracking/gaTrackingSocial.html
[24] https://www.facebook.com/insights/

application's UI affect users in a good way. We certainly don't expect that level of focus, but we do hope that you are keeping the user experience prominently in mind when you design your application.

User experience is usually judged based on how easy and, more importantly, how intuitive the user interface is from the users' point of view. A good-looking UI helps too (though a pretty UI that is a pain to use is much worse than an okay-looking UI that has pleasant interaction). This section reminds you to consider each and every interaction your application executes with the user and make them friendly. An interaction is the complete sequence of steps users need to perform through your application's UI to accomplish a single coherent operation (e.g. posting photos in the Photos application or deleting wall posts).

And forces you!

User interactions that you expect to be frequently used should be the easiest to use and the easiest to find. A to-do list application that asks you to go through a sub-menu and several pages of wizard probably won't gain much traction. Oh, and first impressions matter too, so if your application requires elaborate steps to install, please spend some time to make those steps bearable and pleasant for the user.

If you have carefully considered the design of your application, this section should provide free marks for your team and would serve as a reminder that user experience plays an important role in how successful your application will be.

**Aspiration 13:** Describe 2-3 user interactions in your application and show us that you have thought through those interactions. It would be great if you could also describe other alternatives that you decided to discard, if any.

## JQuery

And here we come to a really fun part of this assignment. This subsection and the next three subsections will deal exclusively with JQuery, along with some animation and AJAX.

Most modern web applications use tonnes of Javascript. With Javascript you can create animations, you can modify the loaded page on the fly, you can even send a hidden request to your server to fetch new data. The language itself is very pretty (though it can get ugly. Real ugly). It is dynamically-typed, with very flexible object-oriented support and tt supports functions as first-class objects.

Of course, the problem with Javascript is that browsers handle DOM structure and CSS differently. We suggest following W3C standards and testing your pages in at least IE8 and Firefox 4.

However, developing in Javascript causes a few problems to developers. Firstly, different browsers and different versions of these browsers have slight deviations from the published standards. Therefore, developers may need to prepare different sets of code to cater to different browsers. As applications get more complex, it becomes increasingly painful to write many lines of code just to get a simple task done.

This has led to the development of Javascript libraries which serve as an abstraction layer for common web scripting. These libraries try to automatically handle browser-quirks, leaving you free to implement your app. Examples of Javascript libraries

are Prototype, JQuery, ExtJS etc. In this assignment, we will show you how to use JQuery[25].

To start using JQuery, you can download the code from `http://code.jquery.com/jquery-1.6.2.min.js`. This is the main library file of JQuery and in order to use JQuery, you need to include this file in the `<head>` of the document that you are working on.

Actually, browsers keep two trees, the DOM tree and the rendering tree. Updating the DOM tree may cause the browsers to update the rendering tree, causing user-visible changes.

One last thing before we move on to discuss JQuery: let's introduce ourselves to DOM. That's the Document Object Model. Think of a HTML document as a tree. Each HTML tag and contiguous text contained in the document is a node in the tree, and each nested tag is a child node of its parent tag. The tag attributes become attributes of the node. This is the DOM tree. Javascript provides you methods to traverse and manipulate this tree; you can add, remove, or modify any nodes in the tree (as long as they result in a valid document). This explanation may sound a little fluffy, but the main gist is that changes in the DOM tree are in turn reflected in the actual displayed HTML page. The DOM thus ends up being a useful abstract model through which you could program dynamic web content in a systematic and elegant manner.

In JQuery, the most fundamental operation is selecting a part of the document. This is done with the `$()` construct. Assuming that you have a DOM object with class *sample_class* , you can access the object via the following command:

```
$(".sample_class")
```

Take note of the `.` in front of *sample_class*. In JQuery, `.` refers to the class of the object. In other words, the above code is accessing the object with class *sample_class*. Likewise, `#` refers to the id of the object. So, if the DOM object's id is *sample_id*, you can access it via the following command:

```
$("#sample_id")
```

This is the basic method of accessing individual DOM elements. There are also other ways of accessing DOM elements. For more information, refer to `http://api.jquery.com/category/traversing`.

Traditionally, to run some piece of Javascript code after the page has loaded completely (including all of its images), we would rely on the *onload* handler. We will first define the code that needs to be executed and place it in a function:

```
function sample_function() {
    //some code
}
```

Then, we will attach this function to the *onload* event by modifying the HTML `<body>` tag:

```
<body onload="sample_function();">
```

[25] `http://jquery.com/`

This causes our code to run after the page is completely loaded. However, there are drawbacks to this approach. For one, the code lies within the HTML. This tight coupling of structure and function clutters the code, thereby requiring the same function calls to be repeated over many different pages. In the case of other events such as mouse clicks, function calls will be made over every instance of an element on a page. Adding new behaviors would then require alterations in two different places, increasing chances of making errors.

To avoid this pitfall, jQuery allows us to schedule function calls for firing once the DOM is loaded, without waiting for images, with the `$(document).ready()` [26] construct.

Thus, the structure for JQuery file will be as follows:

```
$(document).ready(function(){
    // Your code here
});
```

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

## Phase 5: Young Adult

> *I look back five years ago, when I thought I was an adult and knew everything about the world, and I realize I knew nothing.*
>
> —Neve Campbell

### JQuery Animation (Optional)

**This section is optional. Completing milestone(s) described in this section *may* contribute to the 30% coolness factor.**

Our warnings at the section on Feeds apply here too. Animating your pages may spice up your applications, possibly making it easier and funner to use, and may even earn you cookies (and coolness points!). But! Too much of it (or even small amounts of badly done ones) can backfire furiously. Exercise your better judgement before indiscriminately adding animations to everything in sight, just because you know how to do it.

JQuery provides several methods to add basic animations to your application. [27] The most basic animation you will need to know is *show()* and *hide()*.

Assuming you have the following DOM element:

You've seen blogs where everything seems to be moving, cursors turning into pixie dust, mp3s playing on the background... Yeah, those may be great for a small, special group, but if you plan to cater to a more general audience...
Animation is normally triggered by certain event such as mouse click. JQuery library provide a compehensive list of event handlers See
http://api.jquery.com/c

[26] For more info, please refer to http://docs.jquery.com/How_jQuery_Works
[27] See http://api.jquery.com/category/effects/

```
<div id="sample_id" class="sample_class"> This is a div </div>
```

To hide this element, you will execute the following JQuery code:

```
$(".sample_class #sample_id").hide();
```

To show this element in 1 sec, you will execute the following JQuery code:

```
$(".sample_class #sample_id").show(1000);
```

The above two methods are very simple effects bundled in the JQuery core. To create your own customised animations with desired effects, JQuery provides us with the `.animate(Properties, [Duration], [Easing], [Complete])` method. This method accepts four arguments:

- **Properties** Mapping of CSS properties such as opacity, height, width for the animation
- **Duration (optional)** You can use strings such as ”Fast”, ”Slow” or number (in terms of milliseconds) to control the duration of the animation
- **Easing (optional)** This specifies the speed at which the animation progresses at different points within the animation
- **Complete (optional)** The function to callback when the animation is completed

If you give the `.animate()` method a deeper thought, you will realise that `.show()` and `.hide()` methods can also be achieved by passing the appropriate values to `.animate()`. That's right! The `.show()` and `.hide()` methods are actually shortcut methods of `.animate()`. If you look through the list of animation methods in JQuery, you will encounter a few other similar shortcut methods. So, think through your application needs before designing your customised animation. Quite often, the animation effect you need may already been provided for by the JQuery library.

| | |
|---|---|
| **Aspiration 15:** | Describe 2 to 3 uses of animations in your application. Explain what kind of value do they add to the context to which they are applied. Highlight your more innovative animations. We would be interested to know! *(Optional)* |

## AJAX (Optional)

**This section is optional. Completing milestone(s) described in this section *may* contribute to the 30% coolness factor.**

AJAX stands for Asynchronous Javascript and XML. Despite the name you almost never need to get your hands dirty manipulating XML when using AJAX. When we discuss AJAX calls, what we're really talking about is an asynchronous phone-home (a la ET.) to the server, usually to notify the server that something has happened, or to fetch some data from the server.

True to what we said, we will not deal with the XML part of AJAX at all in this section.

### Asynchronous vs. Synchronous

A synchronous request to the server almost invariably involves loading a new page from the server, or reloading the current page. Clicking on a hyperlink makes a synchronous request. On the other hand an asynchronous request does not (usually) involve any reloading or redirection. The user may continue viewing and interacting with the present page as the sending of a request and the reception of the response occurs in the background - hence 'asynchronous'. It may create a more "application"-like experience for your users. We shall not make any assertion that asynchronous calls are faster or more responsive as it turns out that improper usage of asynchronous calls may make your application seem less responsive.

Another benefit with asynchronous calls is that the server need not send an entire HTML page back as the response. It only needs to send the necessary data (if any). For example, in a normal (non AJAX-empowered) blog, when you submit a comment, the server will eventually send the entire blog page back to your browser with your new comment processed in - hence you see the page reload. On the other hand when you submit a wall post on Facebook (which is AJAX-powered), the server only sends the contents of your wall post back after processing, and Javascript on the Facebook page dynamically adds your comment to the page without reload. This is what makes asynchronous calls faster (or at least, appear faster).

Note that a (not always undesirable) side effect of AJAX calls is that it is in itself completely invisible to the user - the browser does not give off any visual cues when your script fires off an AJAX request. Sometimes this is what you want. Other times, it may create the false impression that your application failed to respond to a user event, especially if your AJAX call was triggered by a user event (eg, clicking the 'submit' button on a wall post). For such cases, you may want to create your own visual cue - design a loading indicator and display it while your AJAX transaction is taking place, and remove it when it's done.

### AJAX in JQuery

JQuery provides some APIs to easily make AJAX requests. Refer to the documentation at `http://api.jquery.com/category/ajax/`.

In this section we will be discussing some of the most common functions that you will likely use in this assignment.

To improve page loading speed, a common technique we usually use is to include only essential information in the web page while loading everything else in the background or when an event triggers. JQuery provides us with the `.load()` method to implement this technique easily.

Assuming you have a `<div>` with the id *sample_id*, you may wish to load certain content into this element. Suppose the content resides in *sample.html* on the server, you can execute the following:

```
$("div#sample_id").load("sample.html",
    function(){ alert("The data is loaded.");});
```

The above code instructs the `div` with the id *sample_id* to load the file *sample.html*. Upon completion of the loading, the callback function will be called to notify the user.

The above method provides an elegant way of pulling in fully-formed HTML on demand. However, there are times where we want to retrieve some raw data and process it before rendering it. JQuery provides a few methods to do this. One such method is to use `$.getJSON()` to get the data in JSON format.

Assuming we have a JSON file known as *"sample.json"*, the following code will allow us to retrieve the file from the server

```
$.getJSON('sample.json', function(data) {
    //data processing code here
});
```

The above code will request the JSON file *sample.json* from the server. Upon receiving the data, the second argument, which is the callback function, will be executed.

So far, we have discussed two ways of retrieving data from the server. There are instances where we need to send data to server for processing. JQuery offers two methods of sending data to the server: `$.get()` and `$.post()`.

Assuming we wish to send a variable *sample_variable* with the value 1000 to the web page *sample.php*, we will execute the code as follows:

```
$.post('sample.php', {sample_variable: 1000}, function(data) {
    //process the server response here
});
```

The first argument is the web page you wish to send the data to. The second argument is in a JSON format, indicating the data you wish to send to the server. The third argument is a callback function meant for processing the response received from the server.

The syntax for `$.get()` and `$.post()` is very similar. One of the main differences is that GET places its arguments in the query string portion of the URL, whereas POST requests do not. However, this difference is not very obvious when you are performing AJAX call. Thus the main reason for you to choose one method over the other, is the size of data you wish to send over. GET usually has a more stringent limit on the data size.

| | |
|---|---|
| **Aspiration 16:** | Describe any cool utilization of AJAX in your application. *(Optional)* |

## JQuery Plugins (Optional)

**This section is optional. Completing milestone(s) described in this section *may* contribute to the 30% coolness factor.**

One of the more powerful aspects of JQuery is its ability to incorporate plugins developed by external developers.

To use a plugin, you need to first include the plugin in the `<head>` of the document. You must make sure that the core JQuery file is included before the plugins. See the following:

```
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8"/>
<script src="jquery.js" type="text/javascript"></script>
<script src="plug-in.js" type="text/javascript"></script>
<script src="your_application.js" type="text/javascript"></script>
</head>
```

After the inclusion of the plugin file, you can include your application JQuery file that makes use of the plugins.

There are hundreds of plugins available to aid you in your development. You can search for appropriate plugin via `http://plugins.jquery.com/`.

> **Aspiration 17:** Tell us how you have made use of any existing JQuery plugins to enhance your application functionality. *(Optional)*

## Grading Scheme

The grading of the assignment is divided into two components: satisfying the aspirations (70%) and "coolness" factor (30%). Not including Aspiration 0, there are **17 aspirations** aspirations in total: three are optional (Aspirations 15, 16, and 17) and one is not graded (Aspiration 2). That leaves you with 13 compulsory aspirations. Therefore, we'll adopt the following breakdown: Aspirations 1, 3-12, and 14, are worth 5% each while Aspiration 13 is worth 10%.

The bonus aspirations and the optional aspirations will contribute towards the remaining 30%.

This assignment has 2 key deliverables, a mid-assignment milestone and the final application itself.

**Mid-assignment Milestone (due 20 August 2011):**

- A short write-up on your application idea and execution plans (see Aspiration 1). Restriction: no longer than a double-sided A4 sheet of paper, 12 pt. font, Georgia or equivalent.
- Make sure you have completed Aspiration 3 (and yes, we do want a cute icon!).
- Application URL. Your application must already be online. Things need not be perfect but a fair portion of the functionality must be working. This is a sanity check for you because you don't have much time between the mid-assignment milestone and the final submission.
- A SQL schema (for Aspiration 5). While your SQL schema may change on the second week, we will grade the SQL schema you submit here. Be sure to plan well.
- Answers to the bonus aspirations (if you decide to attempt them).

**Application Submission (due 1 September 2011):**

- Completion of all compulsory aspirations (up to Aspiration 14). Submit your answers in a write-up and categorize your answers by the aspirations they belong to.
- Another separate write-up pitching your application to the teaching staff, i.e. convince us that your application is so good that it deserves all 30% of the "coolness" factor points. Restriction: no longer than 2 A4 sides, 12pt. font, Georgia or equivalent.

## Mode of Submission

**For the mid-assignment write-up**, please submit a single document, uploaded to IVLE workbin. If you use an online collaboration tool such as Google Docs, you still want to download a PDF copy and upload it to IVLE.

**For the application submission**, please provide a gzipped/bzip2ed tarball (yes, we do want to test your UNIX skills as well) containing:

1. A `README` file containing the matriculation number and name for each member of your group. Also list contributions made by each members. Make sure that your application name is clearly written in the `README` file. You may also provide a list of changes that you have made to your original idea submitted in the mid-assignment write-up.
2. A write-up (in PDF format) containing your answers to all compulsory aspirations that require written answers.
3. Source code: place the source code in a sub-directory called `src`. Make sure that the directory structure remains intact (yes, we still want to test your UNIX skills).
4. Proof of working application: either publish your application publicly and provide a link to the main page of your application in your `README` file, or add all the teaching team members as co-developers of your application.

Name the tarball `life-[MatricNo1]-...-[MatricNoN].tar.(gz|bz2)` and upload it to IVLE workbin. In addition, upload your one-(or two-)page pitch of your applicattion to the same directory separately (not in the tarball).

| | |
|---|---|
| **Important Note:** | Your tarball should expand to one and only one directory, in which your `README`, write-up and `src` directory should reside. |

As a reminder, there is a total of **17 aspirations** (excluding Aspiration 0) in this assignment. 3 of these aspirations are optional and Aspiration 2 is not graded.

Clarifications and questions related to this assignment may be directed to the IVLE Forum under the header 'Assignment 1: Life of a Facebook Application'.

Good luck and have fun!

# Appendix - Privacy Policy

This Privacy Policy describes how users' personal information is handled in order to engage in the services available on our application. It applies generally to web pages where this policy appears in the footer. By accepting the Privacy Policy, you express consent to our collection, storage, use and disclosure of your personal information as described in this Privacy Policy. This Privacy Policy is effective upon acceptance for new users and is otherwise effective on August 08, 2011.

## Definitions

1. References to "Our", "We", "Us" and **[Your Application Name]** shall be references to **[Your Application]**
2. References to "You", "Users" and "Your" shall mean references to user(s) visiting this web site, as the context requires.

## Information Collection

Browsing our websites does not require your identities to be revealed. However, under the following circumstances, you are not anonymous to us.

## User

We will ask for your personal information. The personal information collected includes but not restricting to the following:

1. Private information such as name and birthdate
2. Contact information such as email address, mobile number and physical address
3. Additional information which we may ask for if we believe the site policies are violated

Once you log into the account, your identity will be revealed to us.

## Information Usage

The primary purpose in collecting personal information is to provide the users with a smooth and customized experience.

We will use the information collected for the following purposes

1. To provide its intended services
2. To resolve disputes, and troubleshoot problems and errors
3. To assist in law enforcement purposes and prevent/restrict the occurrences of potentially illegal or prohibited activities

## Disclosure of information

We may share information with governmental agencies or other companies assisting us in fraud prevention or investigation. We may do so when:

1. permitted or required by law; or,
2. trying to protect against or prevent actual or potential fraud or unauthorized transactions; or,
3. investigating fraud which has already taken place.

The information is not provided to these companies for marketing purposes.

## Usage of Cookies

Cookies are small files placed in your computer hard drives. We use it to analyse our site traffic. We have also used cookies to maintain your signed in status when you login to our websites.

## Commitment to Data Security

Your personally identifiable information is kept secure. Only authorized employees, agents and contractors (who have agreed to keep information secure and confidential) have access to this information. All emails and newsletters from this site allow you to opt out of further mailings.

## Changes to the policies

We reserved the rights to amend this Privacy Policy at any time. Upon posting of new policies, it will take immediate effect. We may notify you should there be any major changes to the policies.