National University of Singapore
School of Computing
CS3216: Software Development on Evolving Platforms
AY2009/2010, Semester 2

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

Issue date: 11 January 2010
Due date: **21 January 2010 (mid-assignment submmission)**
28 January 2010 (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 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 suggest PHP; it might not be the nicest language, but it is certainly easy to learn.

This assignment is highly open-ended. We provide milestones (in the form of aspirations) so that we could grade your application in a consistent way, even though you may be building different apps. Also, this assignment is 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 will 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. But before you begin, please read through the whole assignment once to get a sense of what is required. 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.

Let's begin...

Reminder:	Please read the entire assignment before starting.

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

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://www.facebook.com/developers/`. *(Not graded.)*

Now you are ready to go! To begin the journey of your new Facebook Application, go to the Facebook Developer page and click the button that says "Set Up New Application".

Before you do anything else, click on the link that says "Facebook Platform" (the Terms of Service: `http://www.facebook.com/developers/tos.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 might 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?)

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 (think of it as 'prenatal education').

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'[2], where your team essentially goes

[2] See `http://en.wikipedia.org/wiki/Software_development_process#Waterfall_processes`

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 user's 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*. It is your users that know best what would make them happy, so if your users tell you that this feature is difficult to use or that having such and such a functionality would be nice, you'd better be busy taking notes and incorporating them into your application's roadmap.

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

Phase 1: Baby

> *"Babies are always more trouble than you thought – and more wonderful."*
>
> —Charles Osgood

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

Out of ideas? Check out `http://www.whatalovelyname.com/`. d:

Aspiration 1:	Your new baby needs a name! Give it one! *(Not graded.)*

Once you have a decent name for your new application, explore more options on the left menu. Most fields are self-explanatory. We talk through some notable fields below.

- **Basic** $>$ **Developers:** Add your teammates here.

- **Authentication** > **Post-Authorize/Post-Remove:** They are what they said they are. Post-Remove/Post-Authorize URL are basically a simple callback URLs that Facebook notifies when a user removes/authorizes your application, i.e. these are the pages that are called by Facebook whenever a user adds or removes your application. For example, you may want to perform some database cleanup when a user removes your application. This URL allows you to do that.
- **Profiles** > **Tab , Info , Publisher** These allow you to integrate with Facebook profile pages. You might want to find out what they are and express your creativity here. We will look at them in greater depth later.
- **Canvas** > **Canvas Callback URL** is probably the most important field. It is 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 attempted 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.
- **Canvas** > **Canvas Page URL:** This is your application page (you know, those `http://apps.facebook.com/some-app/`). Be sure to select the Render Method as 'FBML' - this makes life generally easier [3]. You can pick the other option if you are confident that you know what you're doing.
- **Advanced** > **Sandbox Mode:** Developer mode, you may want to Enable this option for now (Disable it before you publish your app).

  If you want to read more about the rest of the options, you can take a look at `http://wiki.developers.facebook.com/index.php/Creating_your_first_application`. Don't expect too much. The wiki page basically rephrases the grey help text on the page into full sentences and the information is somewhat outdated when we last checked.

Aspiration 2:	Give your newborn application some love. Make sure that you fill the page with as much appropriate information as you can. And yes, we expect a cute application icon!

Phew! Now that we have gotten that one out of the way, click submit.

You have read the ToS, haven't you?

Setting up canvas page

Using an FTP client (or any alternative that you are comfortable with), create a directory (with the same name as the one specified in the Callback URL) in your document

[3] If you are sure that you want to use iframe instead, still pick FBML, you can include an iframe from FBML.

root directory[4]. Create a file `index.php` in the new directory, with the following content:

```
<?php echo "<p>Hello World!</p>"; ?>
```

Visit your application canvas and make sure that 'Hello World!' appears. Congratulations, you have just hooked up your application with Facebook.

Aspiration 3:	Hook up the canvas page with the server. *(Not graded.)*

Notes: If you are familiar with PHP or web programming, you might be wondering why we did not output the standard skeleton (the `<html>...</html>`). This is because the output is slapped on the canvas page (after some pre-processing by the Facebook Platform). The canvas page already contains the skeleton including the Facebook menu, chats and navigation bar.

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

Facebook application is different from standalone web applications. It is like a widget. Facebook needs to give your application enough flexibility to become useful but also needs to *sandbox* you. It needs to insulate you from other applications and itself. If it does not provide a good sandbox, application could disrupt the entire profile's layout, access other applications, or, worse, steal user's data from Facebook (authentication cookies are a big target of most XSS[5] and XSRF[6] attacks).

Thus, Facebook came up with a set of specialized languages that your application can use, mainly FBML (contains a subset of HTML and Facebook's own markup), FBJS (a sandboxed version of Javascript), FQL (a SQL-like query language to retrieve user data), and the Facebook API itself (to interact with Facebook's user data). In this phase, we will play with FBML and the API. We will leave FQL and FBJS to the later sections of this assignment.

[4] By default your AWS website document root is at /home/webuser/helloworld/htdocs/. One way you can change this is by modifying the Apache DocumentRoot directive in /home/webuser/helloworld/conf/httpd.conf.

[5] Cross-site scripting. http://en.wikipedia.org/wiki/Cross-site_scripting

[6] Cross-site request forgery. http://en.wikipedia.org/wiki/Cross-site_request_forgery

Facebook API

The Facebook API is simply a REST-like (stateless) web service. Your application will query the API for user data that belongs to Facebook (such as name, birthday, friends list, etc.). The API is meant to provide restricted access to user data. Users must give explicit permission before you can get at their information (through "authorization" or adding of the application).

The queries and responses take the form of XML[7]. This is like a language we use to communicate with each other, but it has a much stricter (and restricted) sets of grammar and vocabulary. The API itself is very diverse and covers a lot of stuff. You can find the full API reference at this page: `http://wiki.developers.facebook.com/index.php/API`.

And no, it is not written in PHP; the entire Facebook is rumoured to be written in C++, with PHP acting as simple interface between user and actual backend.

The Wiki page provides good details on how to call each API method. Most of the example requests are written for the PHP client library. All the client libraries roughly work in the same way: they provide a native API for developers to use and deal with all the heavy-lifting (constructing the request and translating the response). You can test API methods using the API Test Console located here: `http://developers.facebook.com/tools.php?api`.

Well, after all that long-winded talk about the API, here comes the exciting part, your first trip using the API. We will go with the PHP library here, but you are free to try your hand at the others. We will explore two methods provided in the library that makes your life easier (they are not exactly API-related, but they will give you a feel on how easily we can use the API). The two are `require_login` and `require_install`.

Get the Facebook PHP library[8] by visiting this page:

`http://developers.facebook.com/get_started.php`

And clicking on 'PHP Client Libraries' in the box on the right. You are welcome to read the content on this page for a good introduction too.

After downloading the client library, extract the tarball into your application directory. You should end up with this directory structure:

```
your-app-dir/
  | index.php
  + facebook-platform
      + php
          | facebook.php
          | facebookapi_php5_restlib.php
          | facebook_desktop.php
          | ...
```

You can ignore `/footprints` folder for now. It contains the sample application.

Let's modify our `index.php` to request for authorization.

[7] Not that it's the best choice, XML, in my (Chris) opinion, is too heavy a harness for this task

[8] The wiki page provides other options: `http://wiki.developers.facebook.com/index.php/PHP`

```
<?php
// We first need to get the library for further usage.
require_once "facebook-platform/php/facebook.php";

// Create a Facebook object using API and secret key.
$appapikey = "<your app API key>";
$appsecret = "<your app secret key>";
$facebook = new Facebook($appapikey, $appsecret);

// Require user to login to the application before viewing canvas.
$user_id = $facebook->require_login();
echo "<p>Hello, your user id is: " . $user_id . "</p>";
?>
```

You can always find your API and secret keys via the Facebook developer application (look for your application under 'My Applications' on the right column and click it).

Check out your canvas page.

Note: As a best practice, you would want to extract application constants (in this case `$appapikey` and `$appsecret`) into its own separate PHP file.

API methods are contained in `facebookapi_php5_restlib.php` file. Each of the PHP methods has the same name as the API name (with `.` replaced with `_`. The file also contains javadoc-style documentation that can help you figure out how to call a particular API. Check it out!

The API itself can be called like this (this one sets the contents of the profile box for a particular user):

```
$facebook->api_client->profile_setFBML(<..>);
```

Aspiration 4:	Your application should be able to require the user to login - ie your application should ask the users to allow it access to their data the first time they use it.

Facebook Markup Language (FBML)

This is one exciting and fun "toy" because this is the language used to display stuff on the canvas and profile page. It is very easy too. The only needed ingredients to use FBML is your creativity, the Facebook wiki, and a good debugging tool (Firebug[9] and Drosera[10] comes to mind).

You should realize by now how important the Facebook wiki is. This is the way of the IT and software engineering world today - a textbook can become obsolete in

[9] A Firefox add-on. https://addons.mozilla.org/en-US/firefox/addon/1843

[10] Installed by default with Safari Nightlies/Webkit. http://webkit.org/blog/61/introducing-drosera/

no time, whereas online resources (especially authoritative wikis) adapt and update themselves fairly quickly. The ability to dig for desired information in wikis and source for online docs and materials is becoming an essential skill today.

All right, moving on, FBML is basically a subset of HTML baked with some of Facebook's own markups mixed in. Facebook's server takes the FBML you write, processes them and outputs HTML for your browser.

Unfortunately HTML/CSS is not exact science. With a good array of browsers and rendering engines available today, what works on one browser may look a little off on another browser, and be completely distorted in yet another. This is the curse of web development, and we owe it to our users to ensure cross-browser compatibility to the best of our abilities. **Hint:** Stick to the standards! And check out `http://www.quirksmode.org/`.

Facebook provides a Test Console (`http://developers.facebook.com/tools.php?fbml`) and FBML wiki page (`http://wiki.developers.facebook.com/index.php/FBML`).

Fire up the test console and let's try the following (click 'Preview' to show the results):

```
<fb:header>Hello World!</fb:header>

<p>Hello, <em><fb:name uid="<your user id here>" firstnameonly="true"
  useyou="false" linked="true" /></em>!</p>

<p>Here is your profile picture:</p>
<fb:profile-pic uid="<your user id here>" size="s" />
```

A failsafe way to find your user id is to look at your profile picture url. Depending on your browser you can right click your profile picture and copy link url, or hover your mouse over the picture and look at your browser status bar. The number string at the end of the url, after the `&id=`, is your user id e.g. `http://www.facebook.com/album.php?profile=1&id=`**`630308427`**

Tags like `<p>` and `<em>` you will recognize as plain old HTML. However, all the tags prefixed with `fb:` are FBML-specific. In the example above, we used two such tags, `fb:name`[11] and `fb:profile-pic`[12]. The wiki pages for these (and all the other) FBML tags are very informative and sufficient. So you should be referring yourself to them now to check out what the two tags are all about. The pages tell you what the tag does, what attributes it accepts (in the case of `fb:name`, it accepts `uid`, `firstnameonly`, `useyou`, among others), and also provides an example.

Attributes may be *required* or *optional*. You have to specify all required attributes. It will give you an error otherwise (try removing the `uid` attribute from the example above). Unspecified optional attributes, however, will simply be assigned a default value.

The way FBML works is as follows. Your PHP script on your application server will process requests sent by Facebook and produce a bunch of FBML in response. The Facebook server will then process this FBML and properly substitute all `fb:` tags with the information they are meant to display. Experienced web developers would note here that with such a mechanism, `fb:` tags will not work if they are produced on the client-side (e.g. with Javascript) since these never go through Facebook's servers. We will be discussing more on that in the FBJS section.

[11] `http://wiki.developers.facebook.com/index.php/Fb:name`
[12] `http://wiki.developers.facebook.com/index.php/Fb:profile-pic`

Now play around with FBML by editing your canvas page (ie edit `index.php`[13] and make it output FBML).

Aspiration 5:	Include at least 5 different FBML tags in your application. Using a combination of FBML tags, play with some novel ideas and provide us a short explanation on how you managed to pull it off.

Notes: `uid` is one important attribute. It appears on almost everything that requires user data. As such, the PHP library has convenience variable `$user` to represent this number, accessed by calling `$facebook->user`. As noted, this is also the number string you see assigned to `id` in your profile pic link url. e.g.

`http://www.facebook.com/album.php?profile=1&id=`**`630308427`**

Canvas v Profile

So we've had some fun with the canvas page, the heart of your application's operations. The profile page, however, is what gets the most views, and is the interface through which users interact with each other on Facebook. In the early days of the platform Facebook introduced profile boxes which allow applications to publish contents onto the user's profile page. Profile boxes are limited in their capabilities though - no callback to your server is made. The box is rendered using cached content instead. Since then Facebook has introduced more ways for applications to create a pressence in the profile, and we would certainly want to take advantage of those. To that end, let us introduce ourselves to application tabs and the publisher.

Profile boxes will soon be deprecated, so you should not plan to rely on them.

Application tabs[14] are tabs on the profile page that show application content. If an application supports application tabs, a user may choose to add a tab to his/her profile that features content from that application. Adding application tab support to your application is easy - you simply specify a Tab URL in your application settings (under Profiles $>$ Profile Tab $>$ Tab URL), and then create that page on your server. Like your canvas, you can use all the interesting things you have learnt so far (like FBML) to construct your application tab. There are a number of key differences though. Here are some notable ones: it cannot autoplay flash or autoload Javascript, cannot contain iframes, and cannot show advertisements. You should refer to the wiki page (see footnote) for full details.

The profile publisher [15] is the little form at the top of your profile page and home page that let's you write wall posts, update your status, and share external links and videos. But that is not all! Facebook allows your application to register a custom, application-specific publisher that will allow your user to publish content generated by

[13] When playing with FBML in PHP, sometimes you will be dealing with multi-line FBML. Utilize *heredoc* for this: `http://www.php.net/types.string#language.types.string.syntax.heredoc`

[14] `http://wiki.developers.facebook.com/index.php/Tabbed_Profile`

[15] `http://wiki.developers.facebook.com/index.php/Publisher`

your application in the same way they publish wall posts, youtube videos and all those other things! You can surely imagine by now the kind of visibility a properly designed custom publisher can give your application. Implementing the custom publisher is a little involved though. You would need to fill in the fields in Profiles $>$ Profile Publisher with appropriate callback URLs and refer to the wiki page (see footnote) for details on what those callback pages should do.

Aspiration 6:	Give some thought to how you could integrate your application with the profile page. In addition to your mandatory canvas page, provide support for either a profile application tab or a custom profile publisher. You are welcome to do both if you think this is good.

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 many more advanced concepts that are taught in both the SQL workshop and the web performance workshop.

We have to keep something for the workshops, otherwise nobody will attend them...

A database application may store several *databases*. Hence, while each application will usually use it's 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. While many 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 attend the workshop!!

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). All these are explained in greater detail in MySQL workshop.

After this section, and after sitting through our awesome MySQL workshop, you should be ready to produce a schema for your growing application. Do consider how efficient your schema will be, specifically how complex will it be to access the most commonly accessed data. Think about number of queries and number of tables accessed to complete a single user query. Your schema should be graphical, and indicates 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 7: 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 commmand `mysql_query(query string, db variable)`[16].

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 8:	Tell us some user queries (at least 3) in your app that needs database access. Explain how you complete these queries. Provide the actual SQL queries you use.

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

Facebook Query Language (FQL)

Now that you have used a bit of SQL, we will introduce you to FQL, a SQL-like language that you can use to query Facebook's data. It is very similar to SQL and is pleasantly simple and elegant.

The Facebook API and FQL are different ways to access the same underlying data about your users. FQL provides a classier way to retrieve those underlying data. For example, to get the profile pictures of a person's friends using rest-API, you would probably call `friends.get` followed by `user.getInfo`. Using FQL, it is simply

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

In short, it's very similar to SQL, and behaves as you'd expect SQL to behave.

Using FQL, we can minimize the amount of back and forth calls between the client application and Facebook server (it also reduces the amount of data transferred). We specify exactly the information we want and Facebook's servers will process the request. (And yes, it is likely that they can do this faster than us. Leave Facebook to do what it's good at: churning social network data.) Using the API, we may have to make a few calls and discard some irrelevant data (as demonstrated above, we needed to make 2 API calls, while using FQL, 1 call is sufficient). With FQL, you can perform complex queries and obtain results which you would have a hard time producing with just the API.

You can think of FQL as a declarative interface to your users' data (you specify **what** you want) while the traditional API presents an imperative interface (you specify **how** to get what you want).

Mini FQL Tutorial

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`, `friend`, and `cookies`. The complete list of table is available here:

`http://wiki.developers.facebook.com/index.php/FQL_Tables`

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`, `birthday` and so on.

To send an FQL query, you need to use the API call `fql.query`. In PHP, it will be as follows:

```
$query = "SELECT ...";
$result = $facebook->api_client->fql_query($query);
```

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)[17].

You may use the `WHERE...IN...` form instead to perform pseudojoins.

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

Now fire up the Facebook Test Console and play with some queries of your own. You can also use the test console to try, debug, and refine your queries. Now that you have FQL in your arsenal, you can start doing many cool and complicated user data manipulation in your applications.

> **Aspiration 9:** Show us some of your most interesting FQL queries and explain what they are used for (2-3 examples).

You may recall that many applications have a page that shows either: (a) all friends with the application installed; or (b) an invitation page with all friends (or a sensible subset) that do not have the application installed. Add one of the two (or both!) to your app. You may use FQL if you deem it necessary.

> **Aspiration 10:** A page showing either (a) all friends who have your application installed; or (b) all friends who do not have your application installed.

Handling post-authorize and post-remove

Remember the post-authorize and post-remove URLs in the application settings page? They are used so that you can set up the application for new users and perform some clean-up when the user removes the application.

The difference between Post-authorize Redirect and Post-authorize Callback is as follows. Post-authorize Redirect is the page (in your app) where the user will be redirected to after authorizing (installing) your app. A sensible Post-authorize Redirect page will be one that shows a welcome message, for example. On the other hand, Post-authorize Callback specifies the URL that will be invoked (or `pinged`) in the background when the user authorizes. The URL specified here needn't produce any display data, and usually performs new user set up work in the database and any where else necessary.

To enable post-authorize, simply enter a URL for Post-Authorize Redirect URL in the application settings and create that page. Alternatively, you can use an existing page, possibly passing a URL parameter (e.g. `index.php?action=new`) to indicate that the app is freshly installed.

Post-remove is similar to Post-authorize Callback. Basically Facebook will automatically invoke the URL specified when users remove your app. Again, this should do back-end cleanup work, no display data is needed. For example, you may want to remove the user and his data from your database.

That means you don't get to say good-bye to your user through this URL.

[17] Take a look at http://wiki.developers.facebook.com/index.php/Sample_FQL_Queries for more valid, interesting examples of FQL query.

> **Aspiration 11:** Handle your application installation with the post-authorize URL. If relevant, handle the post-remove as well.

Facebook Javascript (FBJS)

And here we come to a really fun part of this assignment. This subsection and the next two subsections will deal exclusively with Facebook-style Javascript, dubbed FBJS, along with some animation and AJAX.

FBJS uses the same syntax as normal Javascript, so people with Javascript background will likely find FBJS very familiar. For the rest, we have already prepared workshop materials on Javascript. You can read it ahead of the workshop to prepare yourself in advance.

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. It supports functions as first-class objects.

As your application runs on the Facebook platform, Facebook executes your Javascript code on a sandbox. It basically parses your Javascript and performs variable and property renaming (by prepending them with a tag specific to your application). Facebook also strips away potentially dangerous code. Finally, it provides a "client" library (a la Prototype and jQuery) that provides a high-level API for more advanced functionalities. We are going to highlight several of key things to note here. For a good starting document, you should at least read the FBJS wiki page[18].

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 IE6 and Firefox 3.

> **Aspiration 12:** Curiosity will not kill you - and in this case it might earn you credits: tell us, in your own words, either (a) how does the event model work in FBJS (hint: FBJS == Javascript); or (b) the wiki page mentions that FBJS does not support `useCapture` in its event model, what is that?

One last thing before we enumerate the differences between FBJS and normal JavaScript: 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 it's 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

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.

[18] http://wiki.developers.facebook.com/index.php/FBJS

DOM tree are in turn reflected in the actual displayed HTML page, so the DOM ends up being a useful abstract model through which you could program dynamic web content in a systematic and elegant manner.

So, now we are ready! Some important differences between FBJS and regular Javascript are:

What, you expected us to say 'All important differences'? We need to train you to do your own resource hunting you know..

- DOM objects manipulation is now done through a setter and getter. Instead of `element.id`, you use `element.getId()` or `element.setId('some-id')`. Complete list is in the wiki page.
- Beware of script execution order. None of your script will be executed until the first user interaction. So if you rely on Javascript to render your page correctly, prepare to be really disappointed.
- Manipulating styles are done using `element.setStyle` method. You should only need to remember one form of it, the form that accepts a Javascript object containing mappings from CSS properties to its values. Remember, you have to camel-cased the CSS properties as per normal Javascript. e.g.

  ```
  element.setStyle({color: '#efefef', minWidth: '400px'});
  ```

Btw, one of these properties are not supported by IE6.

- `innerHTML` is not supported. Instead, use node creation and manipulations methods, such as `document.createElement`, `element.appendChild`, `element.insertBefore`, `element.setTextValue`, etc.
- Well, you may include FBML. The catch is that since FBML needs to be parsed and filled in with actual data, Facebook requires you to declare them in a `fb:js-string`[19] FBML element. You can then use `element.setInnerFBML` method to use this in the Javascript.

> **Aspiration 13:** Show us a clever usage of `fb:js-string` in your application. Alternatively, show us an interesting DOM manipulation through FBJS that occurs in your application.

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, employing Google Analytics as a measure of self-importance, and polishing your apps in terms of user experience.

[19] http://wiki.developers.facebook.com/index.php/Fb:js-string

Feeds

Feeds are a good way to advertise your application. They remind users that your application exists and may intrigue non-users. They are unfortunately a double-edged sword. Use them correctly and they may boost your application's popularity. Poorly designed feeds, however, may cause more harm than good. Remember those applications that spam you with notifications of stuff you don't even care about? Yes, avoid that. Remember, a feed should make sense and add real value to the user experience or you may displease your users and their friends.

There are two ways to publish feed stories. The traditional way is to make an API call that directly publishes a story to the user's feed[20]. Facebook has had a harrowing history of various great successes and vicious backlashes with their constantly changing feeds mechanisms, so today direct publishing of feeds is pretty tightly controlled - the user needs to grant your application explicit permission to do direct publishes [21]. The intention is that your application should preferably not use this traditional and messy method of feed publishing but should use the new 'Feed forms' mechanism wherever applicable.

Feed forms[22] are more interactive than traditional direct publishing. Basically your application would construct a feed story and then prompt the user with a message asking if they would like to publish the feed story. The feed will then be published if the user says yes. This way you give the user direct control over what gets published and what doesn't, making it more friendly to the user and also allowing you to skip requiring an additional explicit permission. In line with the interactive nature of this feature, feed forms are implemented in FBJS, via the call Facebook.streamPublish[23]. You can do a lot with this call, such as including graphics and even video's in the feed story, and you are also given the choice of publishing not only to your user's and friends' walls but also to the walls of Facebook events and groups.

Aspiration 14:	We want some feeds! **BUT** remember to put thought into this. Nuisance feeds will not only earn you no credit but may incur penalization!

Facebook and Privacy

While we're on the topic of feed publishing let's take a moment to also talk through the idea of 'privacy'. As the Internet grows and the services that live on it grow more and more sophisticated and personal, privacy concerns have been steadily growing in importance. Especially in a social network like Facebook, which holds in its storage servers a wealth of its users' personal information. In fact, Facebook's history is riddled with scandals and outrages relating to Facebook's failure to properly respect

[20] See http://wiki.developers.facebook.com/index.php/Stream.publish

[21] See publish_stream in http://wiki.developers.facebook.com/index.php/Extended_permissions/Stream_permissions

[22] See http://wiki.developers.facebook.com/index.php/Feed_Forms

[23] http://wiki.developers.facebook.com/index.php/Facebook.streamPublish

their users' privacy. For example, Facebook at one point automatically published feed stories about it whenever their users made online item purchases, and this caused a massive user backlash because users think that such information should remain private.

Bonus:	Facebook recently implemented new privacy settings, and every existing user has been prompted to review and confirm the settings for their account. Notably, Facebook has, apparently deliberately, defaulted all privacy fields to 'recommended settings' instead of 'keep old settings' in the confirm request. For some extra credit (and hopefully some happy thinking on your part), give us a writeup on what this means in terms of user privacy, and why do you think Facebook chose to act as they did. (1%)

Google Analytics

Web developers are obsessed with the usage statistics of their applications. As you are developing a Facebook application, you should be too! We are going to use Google Analytics for this assignment. Google Analytics offers a fairly detailed analysis of your users, including breakdown by countries, by screen resolution, by landing pages, etc. It also plots really pretty graphs depicting the usage statistics of your applications.

Such logs can be useful for many things! Knowing which pages are least accessed allows you to weigh the importance of these pages and decide whether you want to make them more visible to the users. Knowing the landing pages and most highly accessed pages for your application lets you optimize these pages for faster download time and rendering. Knowing the breakdown of 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 in term of users' browsers are useful in determining whether you need to spend more time testing new features and changes in those browsers.

To add Google Analytics, start by exploring the Google Analytics website[24] and creating an account. Also take a look at the FBML tag `fb:google-analytics`[25].

Aspiration 15:	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.

[24] http://www.google.com/analytics/
[25] http://wiki.developers.facebook.com/index.php/Fb:google-analytics

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 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 is the user interface 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. installing the application, posting photos in the Photos application, or deleting wall posts in Wall application).

And forces you!

User interactions that you expect to be most 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 submenu 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 had carefully considered the design of your application, this section should provide free marks for your team and serves as a reminder that user experience plays an important role in how successful your application will be. Conversely, a widely-used application may see a decrease in usage after receiving a poorly-conceived UI make-over.

Aspiration 16:	Describes 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 have decided to discard, if any.

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

Facebook Insights

This section is optional. However, you are strongly recommended to give your views on Facebook Insights (see aspiration).

Facebook Insights can complement Google Analytics as it provides information that Analytics alone can never provide. As the Facebook servers are sitting in the middle of the users and your server, it can mine data that is not otherwise accessible.

To access Insights, go to the Developer application and click on the "More" link for your application (under headings "My Applications"), when the popup menu appears, click on "Statistics".

The statistics page itself involves five sections:

- The "Usage" tab provides details on the your application usage statistics, such as the number of active users, number of API calls your application made, and the application performance (HTTP request time and FBML rendering time).
- The "HTTP Request" tab provides statistics on the number of successful HTTP requests (200) and unsuccessful ones (404).
- The "Allocation" tab may be of importance to you. Facebook is able to measure interactions between users and notifications, e-mails, and requests your application sent. Depending on it, Facebook may allow you to have more or less privileges (e.g. if the users click on user-to-user notification often, it may indicate the usefulness of the notification and your application may be allowed to send more of such notifications).
- The "User Response" tab provides data that Facebook uses to determine application privileges (see the previous tab). It is also important for you as the developer as it provides insights on how useful are the viral aspects of your applications (such as notifications, requests, emails).
- The "Feature" tab provides data on user interaction statistics oriented around specific Facebook Platform features. For example, you can find out the number of Canvas Page views, or number of notifications sent, among others.

Aspiration 17: We (the teaching staff) are evaluating the usefulness of Facebook Insights and you can help us! At the end of the semester, we would be really grateful if you would tell us how you had used Insights throughout the semester and in which ways has it benefited you and your application. *(Optional)*

FBJS Animation (Optional)

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

My warnings at the section on Feeds apply here too. Animating your pages may spice up your applications, possibly make it easier and funner to use, and may even earn you cookies (and coolness points!). But! Too much of it (or even small amounts

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...

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.

Facebook provides a simple yet effective Javascript animation library. We will provide a general overview of this library here. The library consists of a single public class. You can create an object of that type by calling the `Animation` function, which accepts the element to be animated as its argument. You use this object to "record" a desired animation sequence, and when you want to play it, you can call the object's `go` method.

Facebook borrows a really nice pattern from Prototype that allows you to chain the method calls. Basically, all 'recording' methods return the object itself at the end, so you could chain the recording and playing like this:

```
Animation(document.getElementById(...))
    .to('background', '#efefef').from('background', '#abababb')
    .to('color', '#ab5691')
    .go();
```

Neat! We recorded the animation we want and simply call `go` at the end! To do this, you will need to be familiar with the methods provided by the `Animation` object. Here are some basic ones:

- `to` animates the property given as its first parameter from its original value to the given value (second parameter). You may animate background color, text color, width, and height, among others.
- `from` is used if you want to specify the initial property value (instead of animating from its original value). A use-case for this will be to create flashing effect.
- `by` method simply change the value of the property immediately (without intermediate values that cause the animation effect).
- `duration` (specified in millisecond) is used to override the default 1 second animation period.
- `checkpoint` method provides a checkpoint in the middle of the animation sequence you specified. It is a way to combine two (or more) animation as one logical step. Any animation specified before a call to this method will be animated first, and then followed by those specified after.

As always the wiki page[26] on animation provides more insights on more advanced usages of the `Animation` object. Keep two things in mind while reading: calls to `Animation` returns an object; calls to methods of the object returns the modified object.

[26] http://wiki.developers.facebook.com/index.php/FBJS/Animation

Aspiration 18:	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 v. 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 request is sent off and a response received simultaneously 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 call 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 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 sans reload. This is what makes asynchronous calls 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 Facebook

While Facebook's Javascript sandboxing might prevent developers from using many ordinary AJAX methods, it makes up for this by making available its own AJAX object that is both competent and intuitive to use[27]. The basic flow goes something like this: First, create a new `Ajax` object. Set the response type to something sensible by setting the `responseType` property. Then, set the `ondone` property to a function to execute when the AJAX request completes. Finally, fire off the AJAX request by calling the `post` method. Something like this:

```
function make_ajax_request() {
    var ajax = new Ajax();
    ajax.responseType = Ajax.FBML;
    ajax.ondone = function (data) { alert("Ajax request completed!"); };
    ajax.post("http://myserver.com/targetscript/");
    // ...and we just fired off the AJAX call!
}
```

Of course, more advanced functionalities are also available if you require them (like setting POST parameters). Study the API reference and the example page on the wiki to see what more you can do with Facebook's provided capabilities. It should cater to most, if not all, of your AJAX needs.

> **Aspiration 19:** Describe any cool utilization of AJAX in your application. *(Optional)*

Facebook Connect-ing

Facebook Connect is a recently introduced API that allows you to access Facebook's user data from an external locations, like external web sites and mobile / IPhone applications. You will be able to ask users to log in to their Facebook account, and thereafter be able to access their Facebook-related information. The framework also permits you to use many features available to you when developing in-Facebook applications in external sites, like FBML.

While this may have no direct tie-in with your applications on Facebook, depending on the nature of your application, it may provide you with an avenue to implement interesting and useful supporting functionality for your applications in the supported platforms. It may even be the other way round - if you have an existing service or application that resides outside of Facebook, and you're using a Facebook application to support that service, this may help you create a tighter integration and possibly provide additional features.

[27] Here's the API reference: http://wiki.developers.facebook.com/index.php/FBJS#AJAX and here's a bunch of helpful examples of AJAX usage: http://wiki.developers.facebook.com/index.php/FBJS/Examples/Ajax

It may not always make sense of course, but the purpose of this section is to make you aware that Facebook Connect exists, and help you think about what might you be able to do with it, and about possibly expanding some functionality beyond the Facebook platform. Surprise us! Your own creativity and innovation is the limit. We should warn you to keep your project scope manageable though. A small but well-executed project would easily trump a greatly ambitious but poorly executed / incomplete one.

Aspiration 20:	Use Facebook Connect to provide external support for your application in a cool and creative way, then tell us about it! *(Optional)*

Grading Scheme

The grading of the assignment is divided into two components: satisfying the aspirations (70%) and "coolness" factor (30%). There are 16 compulsory aspirations (not including Aspiration 0). Two of the aspirations are not graded (Aspirations 1 and 3); the remaining aspirations are worth 5% each for a total of 70%.

This assignment has 2 key deliverables, a mid-assignment milestone and the final application itself. You are expected to have completed the first 10 aspirations by the mid-assignment milestone.

Mid-assignment Milestone (due 22 Jan 2010):

- A page or two of short write-up on the application idea and execution plans. Restriction: no longer than 2 A4 sides, 12pt. font, Georgia or equivalent.
- Application URL. Your application must already be online and mostly working. Not everything has to work perfectly, but all the requirements up to Aspiration 10 must have been satisfied.
- A SQL schema (for Aspiration 7). While your SQL schema may change on the second week, we will grade the SQL scheme that you submitted here. Be sure to plan well.

Application Submission (due 29 Jan 2010):

- Completion of all compulsory aspirations (up to Aspiration 16).
- A page or two of short 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 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 skill 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. 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 skill).
3. Proof of working application: either publish your application publicly and provide a link to your canvas page (or somewhere where we can actually log in or add 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, by upload your one-(or two-)page pitch for your applicaiton 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` and `src` directory should reside.

As a reminder, there is a total of **20 aspirations** (excluding Aspiration 0) in this assignment. 3 of these milestones are optional.

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!