COMMITTEE MEMBERS

Brad Frere (Chair) bradfrere@lostdogdev.com	Lost Dog Developments Inc.
Corey Johnson	Sphere Gravity
Matt Waggoner	EVGO
Richard Fullbrook	
Travis Ralston	

****Please see additional info below****

APPENDIX DATABASE

A database is an efficient way to collect, organize, and retrieve data. By sending queries to a database server such as MySQL, we can manipulate the data within the database, as well as obtain data within it according to the patterns and specifications that fit what we want to retrieve.
A typical database is set up in the following way:

Tables:
"Tables" are used to encapsulate related pieces of data. For example, if I was storing information about students in a school, I would create a table of students within the database. This table could hold information such as:

- Names
- ID numbers
- Grade levels / Homerooms

By organizing logically related data into different tables, we make it easier to understand our database, and also easier to work with the data inside of it.

Rows/Columns:
Having a students table in our database is a good start; but how does one store data *within* this table? This is where rows and columns (like a spreadsheet in Excel or Numbers) come into play.
In each table, the programmer specifies columns which store different pieces of data (the three types of data above would be separate columns in the Students table). Then when we want to actually add a student to the Students table, we add a new row, and fill each column in that row with the appropriate data.

Querying the Database:
At this point, we have a table called "students" and within that table, three columns named "name," "id," and "grade". In order to programmatically manipulate and utilize this data, we need to send queries to the database to instruct it regarding what to change or provide for us.

Select:

The "SELECT" keyword is used to gather information *from* the database. Selecting content from the database follows the format:

```
SELECT [list of column names] FROM [table name] [optional: any other keywords];
```

For example, if we want to collect a list of names of all students in the school, we can use the following query:

```
SELECT name FROM students;
```

Perhaps we would like to retrieve a more specific list of students – all of the students in grade 11. SELECT can be combined with various other keywords such as WHERE, to form more specific queries. To select all students in grade 11, a query as follows would do the trick:

```
SELECT name FROM students WHERE grade=11;
```

The MySQL manual provides an extensive list of keywords that can be used to form different types of SELECT queries.

Insert:

The "INSERT" keyword is used to put data into the database. This keyword is almost always followed by the "INTO" keyword which specifies which database table the data should be inserted into.
To add a new student into our database, we follow the format (square brackets are to show placeholders):

```
INSERT INTO [table name] ([list of column names]) ([values]);
```

So to insert a grade 12 student named Bob with the ID number 1272958 into the students table, the following query would be used:

```
INSERT INTO students (id,name,grade) (1272958,"Bob",12);
```

Update:

When a student advances to the next grade, it would be very inefficient to have to remove all of those student's records and re-enter them with a new grade level attached to their record. For this reason, database servers provide the "UPDATE" keyword which can be used to alter the contents of rows stored in the table.

The UPDATE format is as follows:

```
UPDATE [table name] SET [expression] WHERE [conditions];
```

So to perform the grade incrementing operation mentioned above one first needs to specify which student (or students) needs to have their grade incremented. If we want to change the grade level of a single student, we need a reference that applies to them uniquely. Their ID number is perfect for this! We will use this ID number in the "conditions" place above to single them out. (We'll use the ID number of the student we just inserted:
Then we need to set their grade level to one higher than its current value. So the following query will get the job done:

```
UPDATE students SET grade=grade+1 WHERE id=1272958;
```

If, instead, it's the end of the year and (assuming everyone passed) we want to change *all* of the students into the next grade level, our query needs to be more broad. Grade 12 students shouldn't be turned into grade 13 students (it doesn't make sense in Alberta!) so we want any student whose grade level is *less* than grade 12 to be incremented. MySQL's arithmetic operators are good for this:

```
UPDATE students SET grade=grade+1 WHERE grade<12;
```

There are *many* more common database keywords and functions which can be very helpful in data management for your website. A complete list, including extensive documentation, is available on the MySQL website. This is only a small example.
Tying this database logic into the logic of your program or website makes it possible to create user systems, online stores, analytic systems, etc. Learning the basics of database operation is highly recommended.

A good introduction to PHP and MySQL programming can be found here:

http://www.w3schools.com/php/php_mysql_intro.asp

APPENDIX SERVER SIDE SCRIPT

Server-side logic allows dynamic generation of pages, communication with other machines and software (including a database server) and many other functionalities that facilitate the creation of a dynamic website.

The include Statement:
For a server-side script to execute, it needs to be invoked. This happens in one of two ways:

1. Either the user navigates to the php file directly:

   ```
   http://example.com/test.php
   ```

   Where the script contained in test.php would be executed.

2. Alternately, a script can be invoked through another script using the "include" statement. To "include" another PHP file inside of a script (thereby executing its contents) the following line can be added to your script:

   ```
   include "test2.php";
   ```

   Let's have a look at the effect of such an include statement. Imagine a test.php as follows:

   ```
   <?php
   echo "test2.php outputs the following: ";
   include "test2.php";
   ?>
   ```

   Then test2.php contains:

```
<?php
echo “These are the contents of test2.php!”;
?>
```

The output of running test.php is:

```
test2.php outputs the following: These are the contents of test2.php!
```

This statement is useful for re-using code without writing it multiple times, for creating templates, or for separating your code into manageable files.

Transferring Data:
A critical part of making your website dynamic is the ability to accept input from users. Two standard methods of sending data to a web server are through “GET” and “POST” transfers. PHP provides access to these data through two superglobal (accessible everywhere in your program) arrays called **$_GET** and **$_POST**.

GET:
Data sent through the URL (or through other methods by external programs) is captured in the GET array. The following is a comparison between a URL with GET info, and the corresponding **$_GET** array created by PHP:

```
http://example.com/test3.php?title=CoolPage
```

There are many server side scripting languages, PHP being one of them. A good place to start learning PHP Server Side scripting is here:

http://www.w3schools.com/php/
APPENDIX CLIENT SIDE SCRIPT

Client-Side scripting (most commonly, JavaScript) allows for a dynamic and interactive user experience on your website. Using JavaScript, the elements on the page can be adjusted, created, or removed without having to refresh or navigate to a new page. All modern browsers are also able to send data to and from servers using JavaScript.

The DOM:
JavaScript’s Document Object Model (DOM) provides the interface through which we can access elements and input on web pages. Various built-in functions allow us to target, manipulate, and create/remove certain elements from the “document tree” that comprises the page.

Embedding JavaScript:
JavaScript can either live inside of the HTML page you create, or can reside in an external file which is *referenced* in HTML files where it is used.

Inline:
The *inline* method is done simply using the <script> tag which is usually placed between the <head> and </head> tags.: