# Final Lab - Putting It All Together – Version 1.1

## Objective

In this final lab you will be working with everything you have done so far in your JavaScript, Database, and Python course to create a simple application.

You will need to use what you’ve learned so far to:

1. Create an HTML form
2. Use JavaScript to validate that form.
3. Design and implement a database to reflect the requirements of the application.
4. Use Python in order to:
   1. Handle a form submission on the server side.
   2. Save and retrieve data from your database.
5. Host your application on a Raspberry Pi!

## **Application Details**

You are going to be creating an application that allows you to gather information about a student in a course and assign grades to that student for a type of work (Assignments, Midterms, Final Exam). The application should also provide the functionality to be able to retrieve that information and display some results about it in an HTML table.

*To keep this simple we will assume that this student can only be associated with 1 course and 1 work type.*

## **The Form**

Should be in a file called index.html.

|  |  |
| --- | --- |
| Field | Data |
| First Name | Should be a text field that represents a student’s first name. |
| Last Name | Should be a text field that represents a student’s last name. |
| Course | Should be a dropdown/select list that allows you to select 1 course.  You can decide the available courses (CST8260, CST8209, etc.).  *\*\*If you want to take this to the next level: Try populating this select list from data that is retrieved from the database. (not mandatory)* |
| Work Type | Should be a dropdown/select list that allows you to select 1 work type that the grade is for: (Assignments, Midterms, Final Exam).  *\*\*If you want to take this to the next level: Try populating this select list from data that is retrieved from the database. (not mandatory)*  *\*\*If you want to go even further, try handling multiple work types. (not mandatory)* |
| Grades | Should be a text field that allows the user to input comma separated grades. (85, 75, 90, 100)  To keep things simple, we will assume all grades are out of 100. |
| Save | Should be a button that submits the form to your server. |
| Display Results | Should be a link that redirects to another page to show the display data (HTML table). |

## **Form Validation**

Should be in a file called validate.js.

All fields should show a proper validation message if the data was not filled out correctly.

|  |  |
| --- | --- |
| Field | Data |
| First Name | Required field |
| Last Name | Required field |
| Course | Required field |
| Work Type | Required field |
| Grades | Required field |

## **Form Events**

Should be in a file called events.js.

The form should have a submit event handler attached that prevents the form from being submitted until all of the form fields are filled out correctly.

## **HTML Table/Display**

Should be in a file called display.html.

The table should display a list of all the students in the database with the following columns:

1. First Name
2. Last Name
3. Course Name
4. Work Type
5. Grade:
   1. Should be the letter grade equivalent of the combined entered grades.
   2. You will have to parse the comma separated string and calculate the total grade.

There should be a link called “Add another student” that bring you back to your student form.

### **Letter Grades**

|  |  |
| --- | --- |
| Letter | Equivalent |
| A | 80% + |
| B | 70 – 79 % |
| C | 60 – 69 % |
| D | 50 – 59 % |
| F | 0 – 49 % |

## **Database**

The database must be created using SQLite3.

You will have to design and create a database that allows you to properly store the information gathered from the user. This data will also have to be retrieved later on.

### **SQLite3 Setup Instructions**

You will be using SQLite3 in this application so that you can try out a different DBMS. You will probably start to realize by the end of this lab how similar the two are. For this lab you will have to do some research to figure out how to use this DBMS in your application.

The following online tutorial provides a good start to your sqlite3 investigation: <http://zetcode.com/db/sqlite/tool/>

## **Python/Apache**

### **Accessing the Database**

In this application you will be accessing database through Python. You can use this website to help get you started with Python’s SQLite API: <http://zetcode.com/db/sqlitepythontutorial/>.

### **Apache Setup Instructions**

The Raspberry Pi will have to be setup in order to function properly as an Apache server. You will also have to install a few packages in order start working with a database.

Installing some packages:

1. **Update your Pi:**

sudo apt-get update

sudo apt-get upgrade

1. **Install Apache:**

sudo apt-get install apache2 apache2-doc apache2-utils

1. **Install PHP:**

sudo apt-get install libapache2-mod-php5 php5 php-pear php5-xcache

1. **Install MySQL:**

sudo apt-get install mysql-server mysql-client

1. **Install SQLite:**

sudo apt-get install sqlite3

Next you need to configure CGI (the Common Gateway Interface) to allow python scripts to be executed by Apache. This file contains settings for the default site (you will learn in Network Operating Systems how you can create other sites on the same server).

1. **Some Prep to enable Python**

sudo a2dismod mpm\_event

sudo a2enmod mpm\_prefork cgi

1. **Modify an Apache configuration file:**

sudo nano /etc/apache2/sites-enabled/000-default.conf

1. **Add the following right after the line which reads <VirtualHost \*:80\>**

<Directory /var/www/html>

**Options** +ExecCGI

**DirectoryIndex** index.py

</Directory>

**AddHandler** cgi-script .py

1. **Restart Apache**

sudo service apache2 restart

1. **Test your server to check it allows the execution of Python script by creating a file hello.py in the  folder** /var/www/html**:**

sudo nano /var/www/html/hello.py

1. **Add the following to hello.py:**

#!/usr/bin/env python

print "Content-type: text/html\n\n"

print "<h3>Hello Python CST 8279 Raspberry Pi</h3>"

1. **Save the file and make it executable:**

sudo chmod +x /var/www/html/hello.py

1. **Test that your instance of Apache is able to execute python scripts by starting a browser window and typing (<Pi IP address> is the IP address of your Raspberry Pi):**

http://<Pi IP address>/hello.py

1. **You should now see the message:**

"Hello Python CST 8279 Raspberry Pi”

1. **Create a new folder 8279 as a subfolder of /var/www/html:**

cd /var/www/html

sudo mkdir 8279

## **Deliverables**

You will demo your work in class. The remaining lab classes will be used to check your work to make sure that you are on the right track.

## **Marking Scheme Rubric**

This lab has a maximum mark of 9, awarded according to the following rubric.

|  |  |
| --- | --- |
| Criteria | Mark |
| Superior capability. Lab submitted meets or exceeds expected standards | 9 |
| Satisfactory capability, acceptable product/result | 6 |
| Marginal capability, substandard product/result | 3 |
| No capability, unacceptable product/result. Work not submitted | 0 |