UNIVERSITY OF CALGARY
FACULTY OF SCIENCE
Department of Physics and Astronomy

Physics Teaching Laboratory Website Manual

Version: 0.7

Contents

1.1 Lab Information Hub Website

1.1.1 Introduction

The purposes of the pjl website is to be the central information hub for the educational physics labs. It is a base of knowledge from which the department can work collaboratively on building the future of education physics labs.

Guiding Principles

- Documents must be accompanied by code required to generate the document.
- Documents posted in PDF format.
- Only documents that have been deployed are to be posted.
- **NOT FINISHED**

NEED CONTENT

1.1.2 Goals

- Central
- Accessible
- Secure
- Transferable

NEED MORE CONTENT

1.1.3 How Instructors Can Use the Hub

NEED MORE CONTENT

1.1.4 How Technicians Can Use the Hub

NEED MORE CONTENT

1.2 Definitions

Development Space: A folder (/dev) inside the the Testing Version root directory that contains all files that are edited when adding content to the repository or the inventory. It also contains tools used for adding content. **LOOK INTO IF I CAN MOVE ALL OTHER FILES THAT CAN BE EDITED WHEN ADDING CONTENT HERE**.

Inventory: Complete list of all equipment in the educational labs. This information is organized in a xml file (/data/equipmentDB.xml), and references photos and manuals stored in an equipment folder (/staffresources/equipment).

Live Version: The most current version of the Physics Educational Laboratory Information Hub available to the public. This version lives on the apache web-server.

Repository: Complete collection of all lab document files (/data/repository), and a xml file (/data/labDB.xml) that contains organizational information about the files in the repository.

Root Folder: A folder that contains all file related to the Physics Educational Laboratory Hub (pjl-web). All references to a folder or file are made relative to the root folder.

Testing Version: A version of the Physics Educational Laboratory Information Hub that needs to have recent edits tested. This version lives on the development server.

Web-Server: A centrally hosted server that is running apache2.

1.3 Acronyms

CLI: Command Line Interface

GUI: Graphical User Interface

2 Process for Editing the Website

2.1 Introduction

- Live Version - Public facing version website at pjl.ucalgary.ca. This is what the public is meant to see. This server can be accessed while on the admin vpn by ssh into 10.41.90.58.
- Test Version - Public facing version website at pjldev.ucalgary.ca - Meant for testing out new code without disrupting the live site. This server can be accessed while on the admin vpn by ssh into 10.41.90.80.
- Development Version - Local version of the website code. Where all changes are made. Can be view by starting a local web server (explained later). All files live on the pjl file server. To edit the website file mount root of website code to /usr/local/master on any pjl linux computer.

2.2 Overview of Editing Process

Follow these steps for making all updates to website. The purpose of each step has been explained in Sections **??** - **??**.

1. Sync Testing Version with Live Version. See Section **??**.
2. Make changes to the website. See Section **??**.
3. Test the changes. See Section **??**.
4. Sync Live Version with Testing Version. See Section **??**.

2.3 Sync Testing Version with Live Version

The equipment data base can be modified using the live version of the website, therefor it is important to check to make sure that any changes made to the equipment database by use of the live website have been applied to the development space before more changes are made.

The script "liveUpdate.py" (Listing **??**) will compare the files "data/equipmentDB.xml" on the live server with the one in the development space. If the live version in newer it will replace the version in the development space with the one on the live server.

To sync run...

```
sudo ./liveUpdate.py
```

For more information on how the script works run...

```
sudo ./liveUpdate.py −−help
```

2.4 Make Changes to the Website

All changes to the website should be made on the development space on slug (/usr/local/master/pjl-web). The only exception to this rule is that the equipment database equipmentDB.xml can be modified live by using the inventory website in edit mode, as mentioned in section **??**. It is to only make changes in one place at a time. Do not make changes using the live website if changes are being made to the development space.

For specifics on how to make changes to the repository see Chapter **??**.

For specifics on how to make changes to the equipment inventory see Chapter **??**.

For specifics on how to make changes to the schedules see Section **??**.

For specifics on how to make changes to the safety documents see **NEED THIS SECTION**

For specifics on how to make changes to the standard procedure documents see **NEED THIS SECTION**

2.5 Testing the Changes

Inside the folder pjl-web/date are the most current version of the repository xml file (labDB.xml) and the equipment xml file (equipmentDB.xml). Inside the same folder are the past nine version of each xml file. For example there exists files labDB-0.xml to labDB-8.xml where labDB-0.xml is the newest past version and labDB-8.xml is the oldest.

If changes have been made to the document repository run...

```
sudo ./repWheel.py
```

This script will remove the oldest version, labDB-8.xml, shuffle the rest of the version down one, and then replace the current version with the development version (pjl-web/dev/labDB.xml).

Similarly if changes have been made to the inventory run...

```
sudo ./eqWheel.py
```

Now start a local web-server to check the Development Version, buy running the following code while in the pjl-web folder

```
python3 −m http.server
```

The Testing Version can be view by opening a web browser and going to the URL ”localhost:8000”. Confirm that changes were made as expected.

The links for the website can be manually tested by running.

```
./linkCheck.py
```

Check into any result that does not return “STATUS: 200”

2.6 Sync Live Version with Testing Version

Once the changes to the development space have been made and tested the changes can be pushed to the web-server.

Update the live version by running...

To sync run...

```
sudo ./liveUpdate.py
```

For more information on how the script works run...

```
sudo ./liveUpdate.py --help
```

3 Repository Content

3.1 Introduction

"Experiment Documents" are a collection of documents used by students in the educational labs, as well as all supporting documents. **ADD SOME EXAMPLES**

3.1.1 Criteria for adding document

Documents can only be added to the repository if they meet the following criteria.

1. The files include the pdf given to students to be used in their course work.
2. All files need to generate the pdf are included.

3.1.2 Directory structure

At top most level is a folder called "repository" that contains all experiment related documents.

At the second level all the files are organized by lab experiment. Each experiment has a folder that is labeled with a naming scheme where the first four characters are the unique identifier number, followed by the name of the lab. The lab name should be descriptive of the experiment itself. In this folder is also a folder called "Support_Docs" that contains any documents useful for the experiment, but not actually used to generate the student document.

At the third level files are organized into versions. Each folder follows a naming scheme where the the first four characters are the unique lab identifier number, followed by "PHYS" followed by the Course Number followed by a two character semester identifier, followed by the year. Each folder contains all the file used to generate the pdf given to students in the course, semester, and year as identified in the folder label.

Directory structure sample.

```
/repository
└─0072-Nuclear-Decay
   └─0001-PHYS123FA2017
```

```
│
│      ├── lab.tex
│      ├── photo.jpg
│      └── student.pdf
└── Support_Docs
```

3.2 Editing Experiment Documents

3.2.1 Introduction

All changes should be made to the fill with the word “FULL” in the title. This is one document should contain everything needed to compile the student version and the TA version of an experiment. Different version of a experiment are compiled using the script pjldoc.py **REFERENCE TO HOW TO USE THIS SCRIPT**

3.2.2 Repository xml template

```
<Labs>
    <Lab labId=”0001”>
        <Name />
        <Disciplines>
            <Discipline />
            ...
        </Disciplines>
        <Topics>
            <Topic />
            ...
        </Topics>
        <Versions>
            <Version>
                <Path />
                <Semester />
                <Year />
                <Course />
                <Directory />
            </Version>
            ...
        </Versions>
        <Equipment>
            <Item id=”0001”>
                <Name />
                <Amount />
            </Item>
            ...
        <Equipment />
        <Type />
        <SupportDocs>
            <Doc>
                <Name />
                <Path />
            </Doc>
            ...
        </SupportDocs>
        <Software>
            <Name />
            ...
        </Software>
    </Lab>
    ...
</Labs>
```

3.3 Preparing Experiment Document Content

3.3.1 Introduction

- All tex in one file (lab and companion guide)
- standard preamble file
- pjldoc script for compiling documents
- documents prepared with a root directory of ”under-construction”
- document editing timeline

3.3.2 Preparing the Documents

The following instructions where made specifically for physics 325 in winter 2019. Adjust the names for course and semester.

1. Create folder for course named in the form similary to **PHYS325WI2019**.
2. Inside course folder place a sub folder for each lab named in the form **0078-PHYS325WI2019**.
3. Inside each lab folder there should be...
   - tex file which include student version and companion guide. Name in the form **NAME-FULL-WI2019.tex All edits are made to this file**
   - Any documents referenced in the tex file which are need for compiling
   - Folder called **Support_Docs** that contain any important documents that are not needed to compile pdfs, such as sample data.
   - File called **standard-preamble.tex**
4. Inside main course folder make a text file called **physics325-lab-order**. Inside this folder list the id numbers of each experiment in the order it should appear in the manual. Be careful not to leave a black line at the end of the file.

3.3.3 Compiling Manuals and PDFs

Any set of documents that have been prepared as laid out in section **??** can be compiled in different configurations with the script **pjldoc.py**.

Help Section for pjldoc.py

A full list of flags and uses of the pjldocs.py script can be found by running

```
pjldoc.py --help
```

Generating Student PDFs

To compile all of the student PDFs

```
pjldoc.py PHY325WI2019 −s −c −i 0
```

To compile individual PDF of the second lab listed in physics325-lab-order

```
pjldoc.py PHY325WI2019 −s −c −i 2
```

Generating Companion Guides for TA

To compile all of the Companion Guide PDFs

```
pjldoc.py PHY325WI2019 −t −c −i 0
```

3.4 Adding new version of an existing lab

Note that a version can only be added once, so make sure that everything outlined below is as desired before proceeding to step **??**

1. Make a folder that will contain all file relating to the experiment to add. (ex "0078-PHYS325WI2019" is a suggested name for a folder that would be used to add lab 0078, used in physics 325, for the Winter 2019 semester.)
2. Inside the lab folder make a folder called "Support_Docs". This folder name is not optional because it will be reference in the scripts used for document and website maintenance.
3. In the main folder place. **??**
   - Main tex file. (ex, Rutherford-Scattering-FULL-WI2019.tex)
   - Student tex file. (ex, Rutherford-Scattering-ST-WI2019.tex)
   - TA guide if it exists. (ex, Rutherford-Scattering-CG-WI2019.tex)
   - PDF of student version of lab. (ex. Rutherford-Scattering-ST-WI2019.pdf)
   - Any file needed to compile the student version of the pdf. (ex, setup-photo.jpg or standard-preamble.tex)
   - Any file that is particular this version of an experiment. (ex template-WI2019.xlsx)
   - Place any general documents into the folder called "Support_Docs". (ex, Interesting-Paper.pdf)
4. Run the command.

   ```
   sudo ./repositoryEdit −−add
   ```

   Example *I*/**O** from script when adding experiment version.

   Enter lab ID number:

   0087

   Adding version to "Rutherford Scattering"

```
/0078-PHYS325WI2019
  └─ Rutherford-Scattering-FULL-WI2019.tex
  └─ Rutherford-Scattering-ST-WI2019.tex
  └─ Rutherford-Scattering-CG-WI2019.tex
  └─ Rutherford-Scattering-ST-WI2019.pdf
  └─ setup-photo.jpg
  └─ standard-preamble.tex
  └─ template-WI2019.xlsx
  └─ Support_Docs
       └─ Interesting-Paper.pdf
```

Figure 3.1. Directory structure and sample contents.

```
Enter absolute path for directory containing lab:
/home/pgimby/labs/under-construction/PHYS325WI2019/0078-PHYS325WI2019
Enter the name of the student version pdf:
Rutherford-Scattering-ST-WI2019.pdf
Enter course number:
325
Enter semester:
Winter
Year
2019
Would you like to edit the equipment list for this lab? y/N
n
Would you like to edit the software list for this lab? y/N
n
Would you like to edit the disciplines list for this lab? y/N
n
Would you like to edit the topics list for this lab? y/N
n
Is this information correct? N/y:
y
```

Note that the lists of equipment, software, disciplines, and topics can be edited here if desired. For more information on the see **NEED REFERENCE**

5. Sync live version. See section **??**

3.5 Adding a new lab to the repository

Before beginning ensure that all equipment used in the new experiment are in the lab inventory, and have equipment ID number.

1. Create a folder for the new lab (example “new-lab-folder”), and place all files for generating student pdf, and the student pdf in new-lab-folder

2. Inside new-lab-folder make a directory called “Support_Docs”, and put all documents relevant to lab, but not needed for generation of pdf into it. This might include research papers, sample data, Excel spreadsheets, etc.

   ```
   sudo ./repositoryEdit −n
   ```

   The command can also be run in test mode by executing...

   ```
   sudo ./repositoryEdit −n −t
   ```

   The script will now take you through several steps to gather the information needed to properly add this new lab to the repository. There are several safeties built into the code, but there will be a request to review the input information and confirm that it is correct. Please take time at this point to carefully review metadata entered.

   The disciplines, topics, and software entries must align with the master list contained in /data/validDiciplines.txt, /data/validTopics.txt, and /data/validSoftware.txt. New items must be added to these master lists before they can be added to a lab.

   - Name, **Name as to be Seen on Website** *Use Standard Title Capitalize Convention.*
   - Type, **Type** *Must be either Lab or Labatorial.*
   - Disciplines, **Discipline1, Discipline2** *Disciplines must comma separated be taken from the approved list* **Need location of this list.**
   - Topics, **Topic1, Topic2** *Topics must comma separated be taken from the approved list* **Need location of this list.**
   - Semester, **Semester** *Winter, Spring, Summer, or Fall*
   - Year, **Year** *: Four digits.*
   - Course, **Course Number** *Three digit number corresponding to the course the experiment was used in.*
   - Equipment, **equipID-(Amount)-[alternate equipID], equipID-(Amount)** *equipID is four digit code of equipment in inventory, Amount is how many are needed, alternate ID is the four digit code of equipment in inventory that can be used if the primary unit is not available. IDs amounts and alternate IDs separated by “-”, and items in equipment list separated by “,”*
   - Software, **Software1, Software2** *Name of all software needed. Must be software from the list of supported software* **Need location of this list**
   - PDF, **PDF exact Name** *This needs to be the exact name of the student pdf*

4 Inventory Content

4.1 Inventory structure

Each item in the inventory should have a unique identifier number, and a unique name. An item can be either a stand alone item, or a kit. If the item is a kit, it will need to include a list of the items in the kit. If only part of the kit is needed for an experiment the repository xml can reference those items by creating a equipment tag in the labDB.xml that has the id number for the kit, but the name for the individual item(s). Each item has a place for any number of manuals, and one picture.

All changes outlined in sections **??** and **??** are made to the Development Version via the CLI. Once all changes have been made, and are satisfactory, the changes must be pulled to the Live Version.

Changes made by the GUI, as outlined in Section **??**, can be made to either the Development Version or to the Live Version.

WARNNING! To ensure changes are not lost, only make changes to the Live Version or the Development Version. In order to switch methods the Live Version and the Development Version must be synced.

4.2 Adding New Equipment

From the /dev/python-tools folder, the command,

```
./equipmentEdit.py −n
```

A prompt will appear that will ask for information regarding the new item. Enter all the information available. It is ok leave some fields blank just as long as there is a name. Once all the available information is added, the user will be asked to confirm the information entered. The new item entry will then go through a validation process to ensure that the name is unique. If everything checks out the new item will be added to the /dev version of the equipmentDB.xml file.

Note: To add Greek letters enter them as (ex. {Omega} or {mu}).

4.3 Adding Photos of Equipment

When taking photos note that they will require editing before they are added to the database. The final version of photo will be square, so keep this in mind when taking the photographs. Note that all images must be in jpg formate.

1. Place images in /usr/local/master/labs/rawphotos
2. Rename all images using the scheme [idnum]img.jpg where [idnum] is the id number of the piece of equipment photographed.
3. Run the conversion script

   ```
   ./convertImg.py
   ```

4. Enter angle to rotate photos. Photos from some cameras will look like they are properly orientated until they are posted on the website, at which time they will look like they are sideways. It is recommended that when using a new camera that the first image to add is used as a test case to determine if the images need to be rotated by the conversion script.

   Edited version will now appear in /usr/local/master/rawimages/output

5. Visually check all photographs in the output folder to confirm that they are still acceptable after they have been converted.
6. Move all photographs ready to be added to the database to /staffresources/equipment/equipimg
7. From /dev/python-tools run the script

   ```
   ./equipmentEdit.py −i
   ```

   to update the images in the equipmentDB.xml.

8. Check local version of website, and once the photos are acceptable remove all photos from /usr/local/-master/rawimages so that they are not added with the next batch of photos.
9. Update live version.

4.4 Adding Equipment Manuals

All manuals for equipment in inventory are contained in the folder /staffresources/equipment/equipman. Each piece of equipment can have as many manuals or other related documents as needed but they need to meet the following conditions.

- They must be in PDF format.
- They must be uniquely named.
- The name of must start with the equipment items four digit id number.

For example, equipment item 0001 “Fluke Multimeter” can have manuals called 0001man.pdf, 0001diagram.pdf, 0001man-2.pdf.

Once all manuals to add have been added to the folder of manuals run the script

```
./equipmentEdit.py −m
```

4.5 Deleting Old Equipment

NEED TO DEAL WITH MANUALS AND PHOTO OF DELETED EQUIPMENT

To remove a piece of old equipment run, from the python-tools folder, the command,

```
./equipmentEdit.py −d [idnum]
```

If the piece of equipment is currently listed as part of the equipment list for a current lab the script will prompt you to make sure that you know this. Ideally the equipment list in the lab repository should be updated first before removing the equipment. This will help to keep the lab equipment lists and the equipment database synced.

4.6 Updating Equipment with Website GUI - Not Functional

Need to create a way to sync changes made via the GUI to the development code

The details of an item in the inventory can be updated on the items inventory information screen while running the inventory page in edit mode. In order to access this feature the user must but in the PJL_Admins group.

4.6.1 Editing the Live Version - Not Functional

Editing the Live Version can be helpful when making changes to existing inventory items. Once new items need to be added the GUI method will no longer work. This tool is best used for making a few small changes on the fly, like adjusting the number of an item needing repair, or adding a new location, or a model number.

Changes cannot be made with this method and with the CLI tools simultaneously. If changes are made with this method then the Development Space will need to be updated before using the CLI or the method outlined in section **??**. Failure to do so will result in all changes being lost.

To use the GUI to make changes the live site simply access the inventory though pjl.ucalgary.ca and click on "edit mode".

4.6.2 Editing the Development Space - Not Functional

Making changes directly to the Development Version is useful when doing a full inventory. This is done by starting a local webserver that will display the Development Version.

Navigate to the pjl-web folder in the command line and run the command

```
python3 −m http.server
```

Open up a web browser and go to localhost:8000 **This function will make changes to the live website. I WILL NEED TO BUILD IN FUNCTION TO THE CLI THAT WILL ENSURE THAT CHANGES MADE TO THE LIVE SITE ARE UPDATED IN THE DEVELOPMENT CODE BEFORE THE DEVELOPMENT CODE CAN BE EDITED**

5 Student Resources

6 Miscellaneous Content

6.1 Schedules

1. Create spreadsheet of schedules for the semester. (ex, schedule-WI2019.xlsx)
2. Create a PDF of the experiment schedule. (ex, schedule-WI2019.pdf)
3. Create a PDF of the lab room schedule. (ex, room-WI2019.pdf)
4. Place spreadsheet and PDFs in to folder /pjl-web/data/schedules
5. Copy /pjl-web/data/schedule-WI2019.pdf to /pjl-web/data/schedule-current.pdf
6. Copy /pjl-web/data/rooms-WI2019.pdf to /pjl-web/data/rooms-current.pdf
7. Add the previous schedule (ex, schedule-FA2018.pdf) to schedule archive

   ```
   nano /pjl-web/data/schedules/schedule-index.html
   ```

   Paste the code...

   ```
   <a href="/data/schedules/schedule-FA2018.pdf" target="_blank">
     <div class="schedule-download" >
       <i class="far fa-calendar" style="font-size: 20px;" aria-hidden=true></i>
       <div>
         <p>Experiment Schedule - Fall 2018</p> <!-- description -->
       </div>
     </div>
   </a>
   ```

   ...directly before the similar code that exist for two semesters previous (ex, schedule-SU2018.pdf).
8. Add the previous schedule (ex, schedule-FA2018.pdf) to schedule archive

   ```
   nano /pjl-web/data/schedules/room-index.html
   ```

   Paste the code...

   ```
   <a href="/data/schedules/rooms-FA2018.pdf" target="_blank">
     <div class="schedule-download" >
       <i class="far fa-calendar" style="font-size: 20px;" aria-hidden=true></i>
       <div>
         <p>Rooms Schedule - Fall 2018</p> <!-- description -->
       </div>
     </div>
   </a>
   ```

   ...directly before the similar code that exist for the two semesters ago.

6.2 Safety

6.2.1 Orientation

- Located in /pjl-web/data/safety/lab-safety-manual/
- Edit latest tex file and call it Orientation-WI2019.tex (adjusted for current semester and year)
- Overwrite file called Orientation.tex with updated version.
- Compile Orientation.tex
- **THERE IS A SYMLINK IN PARENT DIRECTORY. WHICH FILE DOES PJLDOCS LOOK FOR?**

6.3 Procedural Documents

NEED CONTENT

6.4 Templates

NEED CONTENT

7 Future Ideas

- change equipment amounts to be stored by location so that they view by storage unit when printed by CLI, but make it so the website displays the total amount.

8 Code