User guide outline

Spinning up the project

1. From command line Create a new Python 3 virtual environment

```
python3 -m venv ftirdb_env
```

2. Enter the environment by running

```
./ftirdb_env/bin/activate
```

3. Install and upgrade the packaging tools:

```
 pip install --upgrade pip setuptools
```

4. Install the project in editable mode

```
pip install -e ".[testing]"
```

5. Run setup.py

```
Python3 setup.py
```

6. Create the database either from command line of MySQL workbench

```
mysql -u root
create database birddb;
exit
```

6. Initialize and upgrade the database using Alembic:

```
alembic -c development.ini revision --autogenerate -m "init"
```

7. Upgrade to that revision:

```
alembic -c development.ini upgrade head
```

8. Run the project

```
 pserve development.ini
```

This is a user guide to aid anyone who wishes to expand on the BIRDB databank work or recreate it. It includes explanation of the pyramids framework used and incorporated modules, as well as links to useful documentation

There are also comments and doc strings within the code itself to help explain how the code works.

These user guides are worth looking at to gain a comprehensive understanding of how the framework is structured and how it works.

https://docs.pylonsproject.org/projects/pyramid/en/latest/

https://colanderalchemy.readthedocs.io/en/latest/

https://www.sqlalchemy.org/

https://docs.pylonsproject.org/projects/deform/en/master/

- How it is structured - show file structure and explain what each folder contains - explain ideas of how things can be changed if necessary and explain how the system is flexible eg. Could use different template system/different database/ easily change the data structure.

MYSQL

To create the data model the database must already be created in mysql – to do this

Open mySQL cmd line,

Enter password - pass

CREATE DATABASE (name of database);

Connect the database to the application

Line 17 in the development.ini file is very important as this is the connecter to the chosen database.

Database and connector chosen username password database name

```
sqlalchemy.url = mysql+pymysql://root:qkd42cua@127.0.0.1:3306/birdb
```

Using sqlacodegen for converting mysql code to sql alchemy

Pip install sqlacodegen

Type in cmd:

sqlacodegen mysql://root:pass@localhost/newdb --outfile (filename)

Copy and paste file in to 'models' in pyramid

Each table will be one form - you need to 'import' name of each file in to initialise db file, and then follow alembic initialisation steps. If any errors occur, you may have to fix names/fields in the data model.

Editing models in sql alchemy - Example model

```
92
93
94 class state_of_sample(Base):
95     __tablename__ = 'state_of_sample'
96
97     state_of_sample_ID = Column(INTEGER(6), primary_key=True, unique=True,  autoincrement=True,
   info={'colanderalchemy': {'exclude': True}})
98     state = Column(Enum('gas', 'solid', 'dried film', 'liquid',''), nullable=False, info={'colanderalchemy':
   {'exclude': True}})
99     temperature_degrees = Column(INTEGER(11), default=0, info={'colanderalchemy': {'description': 'In degrees
   centigrade'}})
100    pressure_PSI = Column(INTEGER(11), default=0, info={'colanderalchemy': {'validator': Range(min=0, max=1000),
   'description': 'PSI'}})
101    #child
102    sample_ID = Column(Integer, ForeignKey('sample.sample_ID'),  info={'colanderalchemy': {'exclude': True}})
103
```

If you want to exclude a field in the form, then add this code to the the column in sql alchemy

```
info={'colanderalchemy': {'exclude': True}}
```

If you want to add additional information or an explanation of the field eg. units, then add

```
info={'colanderalchemy': {'description': 'atmosphere in psi'}}
```

```
If you want to add additional limitations not specified in the SQL alchemy model
```

```
info={'colanderalchemy': {'validator': Range(min=,max=)}}
```

Make sure names of models are imported in to model __ini__ file, related views, tests, routes and the initialise db script.

```
In routes.py the web addresses are configured
For example

config.add_route('experimentForm', '/experimentForm')

Each web address needs a corresponding view which contains the business
logic and python functions for returning data from the database and a cor-
responding jinja2 template.
```

VIEWS

Example of a view -

Route name - Corresponds to the route which can be found and define in the routes file
Template name - Corresponds to the jinja2 template in the templates file
Return dictionary - Always return all of python variables as a dictionary to allow Jinja 2 template to access them

```
@view_config(route_name='experimentForm',
             renderer='../templates/experimentForm.jinja2')
def experimentForm(request):
    """ Within this function add all required code, depending on the purpose of the web page """
        Dictionary = {}
```

```
return (Dictionary)
```

See commented code for examples of addition data to the database, outputting forms and querying the database.

Example of Jinja2 template -

Print python values - Output dictionary variables by using curly brackets. '| safe' is often used to escape characters in long pieces of code.

Uses a fixed template 'layout.jinja2' and adds to it

Scripts - Add linked to CSS and javascript here

```
{% extends 'layout.jinja2' %}
  <!-- Meta Tags -->
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 <!-- CSS -->
 <link rel="stylesheet" href="C:/ftirdb/ftirdb/static/form.css" type="text/css" />
 <!-- JavaScript -->
 <script type="text/javascript"
       src="C:/ftirdb/ftirdb/static/jquery-2.0.3.min.js"></script>
 <script type="text/javascript"
       src="C:/ftirdb/ftirdb/static/form.js"></script>
</head>
{% block content %}
<div><p>{{ experimentForm | safe}}</p></div>
{% endblock content %}
```