**ABOUT**

Xenon Health is currently using Health Fusion for its billing and claim purposes.

The system is easy to use but is proving to be costly. Having an in-house billing system will prove cost effective in the longer run.

Building such a system:

Any system that deals with patient information needs to comply with HIPAA privacy rules.

These rules specify the way the patient information should be protected by specifying the guidelines that need to be followed while designing such a system.

The guidelines state what needs to be done but do not provide any help on how to do them.

Things to think about:

* Connecting with the clearing houses
* Setting up the claims forms

**WORK SO FAR**

Trying to replicate the already existing health fusion system.

**models.py [Database Models] :**

**Personal\_Information**: To store patient information – personal information and account information.

**Guarantor\_Information**: Guarantor’s information.

\*If self, no need to fill up these details.

**Payer**: Insurance Providers

**Insurance\_Information**: Connecting the Patient, Payer and Insurance Information

**Locations**: ??

[CLAIMS]

**Provider**: Billing/ Rendering Service Provider

**Diagnosis\_Codes**: ICD 10 quotes

(Initially imported from an excel through a script)

**Procedure\_Codes**: CPT Codes

(Initially imported from an excel through a script)

**Claims**: To store claim details

\*Incomplete

**forms.py**

PatientForm: A form that models after the Personal\_Information model to allow user to fill in patient information

GuarantorForm: A form that models after the Guarantor\_Information model to allow user to fill in guarantor information

InsuranceForm: A form that models after the Insurance\_Information model to allow user to fill in insurance information

ClaimsForm: \*Incomplete

**views.py:**

**Secure Login:**

HIPAA compliance requires a login mechanism to ensure that the system and data is protected from unauthorized access.

user\_login and user\_logout define basic login and authentication system to allow user to log in and out of the portal.

\*Incomplete work on the password reset module

**Audit Logs:**

Another criteria to ensure data security and to hold every individual who accesses this data accountable for their actions, an audit log is setup. This audit log ensures logging of any change that happens to the data along with the name of the person who made that change and the date and time of change.

The admin\_log can be used to view this logging information.

**get\_patient\_info:**

This view is used to:

1. Display an empty form to allow entering of new information
2. Save the data from the form once you submit it
3. Display all patients and their chart numbers (if who=all)
4. Display information about a specific patient (if who=patient’s id)

**get\_insurance\_info:**

This view is used to:

1. Display an empty form to allow entering of new information
2. Save the data from the form once you submit it

**get\_guarantor\_info:**

This view is used to:

1. Display an empty form to allow entering of new information
2. Save the data from the form once you submit it

**\*Similar views should be defined to handle claims**

**admin.py**

The billing system is split into two parts:

One is the front end that is accessible to the whole of billing team (through valid logins) that will allow them to add patient, guarantor, insurance and claim information.

The other is a backend – admin panel, that will allow specific users to add and monitor providers, payers and code information.

**urls.py:** This file holds the url to view mapping. That is which url will generate a call to which view.

**IMPORTANT COMMANDS**

To run the project:

python manage.py runserver 8080

To make changes to the database:

1. python manage.py makemigrations
2. python manage.py migrate

**LOGINS**

Django Admin:

Username: admin

Password: Xenonhealth

MYSQL:

Host: localhost

Port: 3306

Name: xenonhealth

User: root

Password: Xenonhealth

**INSTALLATIONS**

Python:

<https://www.python.org/ftp/python/2.7.11/python-2.7.11.msi>

Test Setup:

1. Upon completion of installation, open a command prompt and type python. If you see the Python prompt, installation was successful. If not you will have to set your Windows installation’s **PATH** environment variable by adding the line “C:\python27;C:\python27\scripts;” to the already existing path variable.
2. Setup **PYTHONPATH** if not already set to “C:\Python27\Lib\site-packages\”

Installing Setuptools and PIP

Django: [Version: 1.8.4]

pip install django

MySQL:

pip install MySQL-python

Solving visual c error:

Goto: http://www.codegood.com/archives/129

And run: MySQL-python-1.2.3.win32-py2.7.exe

Other Packages:

Audit Log:

pip install django-audit-log

Read Excel data into our models:

Pip install pyexcel

Pip install pyexcel-xls

Pip install pyexcel-xlsx

Countries: Drop down of all the countries in the forms

Pip install django-countries

Languages:

Pip install django-languages

SSN & Phone Number Formatting:

Pip install django-localflavor

**REFERENCES**

Django:

http://www.tangowithdjango.com/book17/

Audit Logs:

http://django-audit-log.readthedocs.org/en/latest/index.html

Excel Data Access and Manipulation:

<http://pyexcel.readthedocs.org/en/latest/>

HIPAA Documentations:

[HIPAA Security Series #4 - Technical Safeguards - techsafeguards.pdf](http://www.hhs.gov/ocr/privacy/hipaa/administrative/securityrule/techsafeguards.pdf)

[HIPAA Security Series #2 - Administrative Safeguards - adminsafeguards.pdf](http://www.hhs.gov/ocr/privacy/hipaa/administrative/securityrule/adminsafeguards.pdf)