**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. Creating an audit log page is a requirement of HIPAA, which has been completed.

**INSTALLATIONS**

Python:

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

PDFTK:

<https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/pdftk_free-2.02-win-setup.exe>

MySQL:

<http://dev.mysql.com/downloads/mysql/>

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 third party modules:

List of modules for this project are in requirements.txt file in the root directory. Use this to install all requirements (this file includes Djang) with below command line while you are in root of project directory

**pip install -r requirements.txt**

Solving visual c error of My SQL:

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

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

Don’t forget to migrate before running the server for the first time.

**LOGINS**

Django Admin (you will type in this when you run the app for the very first time!):

Username: admin@xenonhealth.com

Password: Xenonhealth

MYSQL:

Host: localhost

Port: 3306

Name: xenonhealth

User: root

Password: Xenonhealth

Login for health fusion (https://login.healthfusion.com/):

User: xenonga

Pass: xenon856$

**IMPORTANT COMMANDS**

To run the project:

python manage.py runserver

To view the project:

<http://127.0.0.1:8000/dashboard/>

To make changes to the database:

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

**Architecture**

**App Logic & Display**

Django Web Framework

(Python 2.7)

**Client**

**Database**

MySQL

**JavaScript**

jQuery.js

jQuery Validation.js

lodash.js

Moment.js

Pikaday.js

**CSS**

BootStrap

Font-Awesome

**Platform**

Windows 8.1

Microsoft Excel 2010

Note that current system is using **PyPiWin32** module to request MS Excel to generate PDF file.

**Work so far**

After you open the BillingApp directory you will notice that the whole thing is divided into multiple apps, namely: accounting, accounts, base, BillingSystem, claims, dashboard, displayContent, infoGatherer, media and report.

Accounting: This takes care of accounting of claims. It includes models like claims, payments, apply and create.

Accounts: This take care of the login system for users.

Base: A base model to be used by other models.

BillingSystem: The main app which contains setting.py

Dashboard: Includes everything associated with the dashboard (main) page.

DisplayContent: This takes care of viewing of patients, going back to claim history of patients etc. Mimics the healthFusion website in functionality.

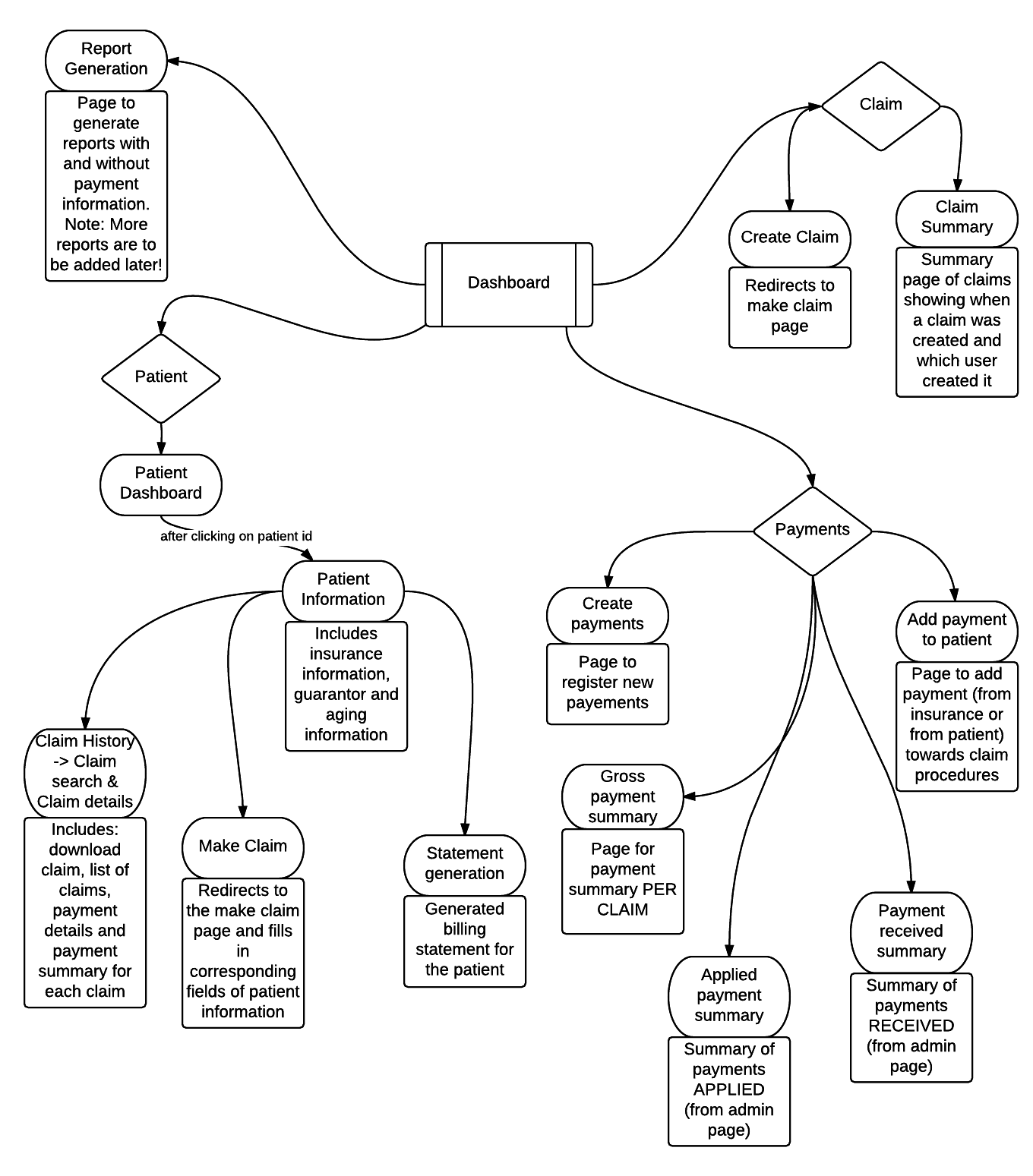
infoGatherer: Contains models for the database like personal information and diagnostic code. It also contains views.py for claim generation.

Media: Stores all generated claim and statements. These can viewed or downloaded from the app, in the browser.

Report: Contains stuff relating to report generation, like transaction reports.

**Flow of the app:**

The app starts with the dashboard, which is connected to all other components in the app. There are links in the navbar too, which is not covered here. There are links on the dashboard for guidance to every part of the app (directly or indirectly).



**Audit log (info/auditlog):**

Maintaining an audit log is a HIPAA requirement, which has been completed. This log maintained entries of old and new values for any change made to the database via the admin page. It also has the ability to see who made the changes and when were the changes made. The whole thing is split into three parts for EACH table: modified (keeps track of changes made to existing entry), created (keeps track of new entries) and deleted (keeps track of deleted entries).

**Make Claim (info/postad):**

This form is used to generate claims. The generated claim opens in a new tab as a pdf. This claim generation also triggers things like saving the claim in the server and saving the path to the saved claim in the database, creating claim object, creating Procedure objects, creating Charge objects. These saved claims can be retrieved from the claim history page of a patient.

In claim form, patient is a person who receives services. Insured is a person who hold the right to reimburse expense from insurance company. Both patient and Insured are “**Personal\_Information**” model. Insurance company information will be captured by “**Payer**” model.

One person can hold many insurances. “**Insurance\_Information**” model will capture relation between a person and a company.

Referring Provider is a doctor who refers patient to another doctor who actually perform services. This is captured by “**ReferringProvider**” model.

For “**Provider**” model, there are three types of provider on the claim, differentiated by field “**role**”.

1. Billing – is a company who perform billing procedure. Normally, this is Xenon Health self.
2. Location – is a location of facility where services has been conducted.
3. Rendering – is a doctor who perform the service for patients.

Diagnosis code is the code to describe what kind of injury patient has, captured by “**dx**” model

Procedure is a line to describe services the doctor has done for that claim. There can be up to 6 procedures per claim. Each line has

Date of service – the date that service has been done. From date to To date will always be the same day for the current system.

Place of service – This will be the code determined by location provider

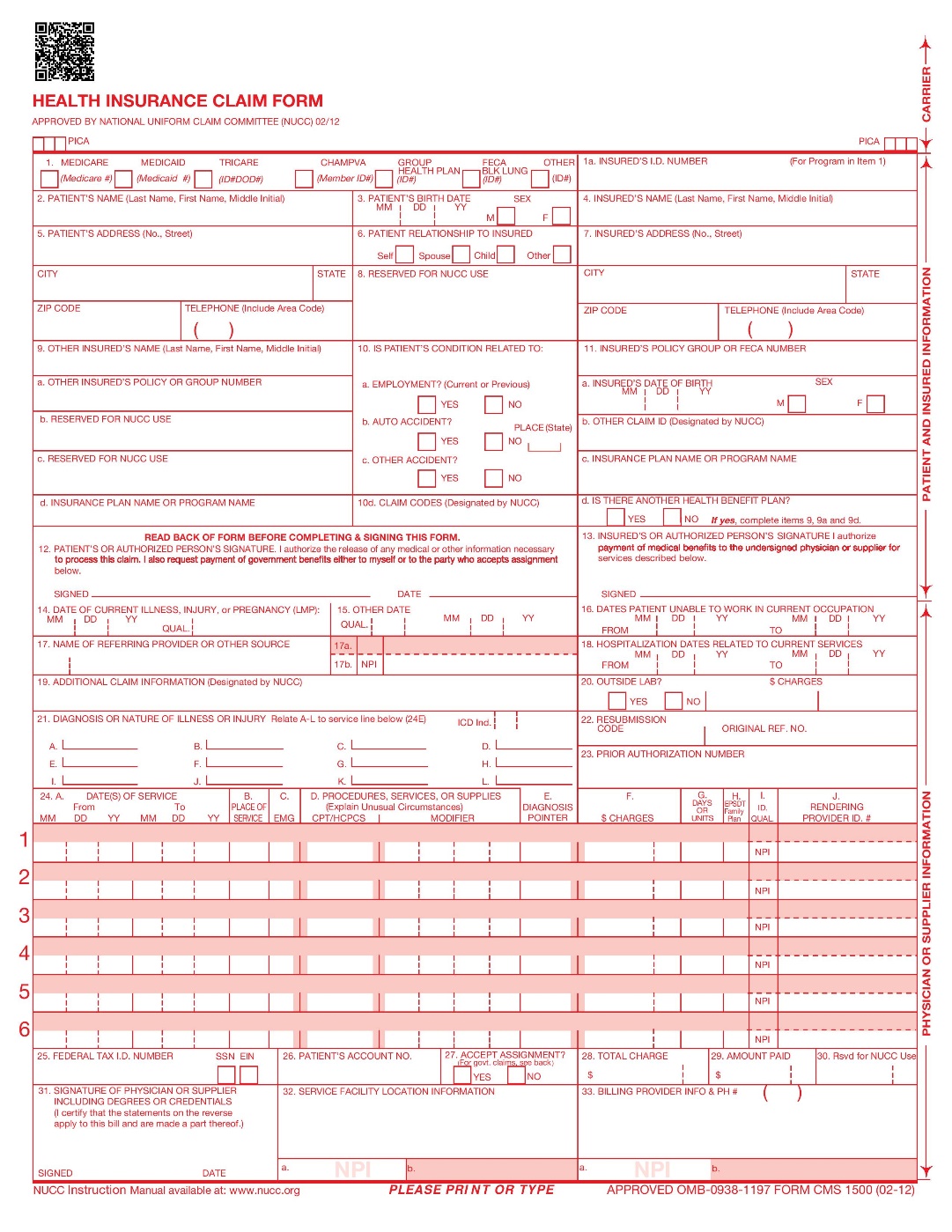
CPT is the code of treatments. This is captured by “**CPT**” model

Modifier – is indicated by CPT

Diagnosis Pointer – is to indicate what kind of injury for that treatment. The code must be presented in Diagnosis section.

Charge – is the amount that patient is charged

Rendering Provider ID – ID of doctor who perform the service. This will be the same for all lines in current system.



Patient

Insured

Insurance (Payer)

Diagnosis

Referring Provider

Procedure (CPT)

Rendering Provider

Location Provider

Billing Provider

**Payment:**

When Xenon Health file a claim to an insurance company, the company doesn’t give us money in exact amount for just that claim. They rather pay a big pile in certain period of time. Xenon Health will manage and distribute the money for unpaid claims ourselves. “**Payment**” model will capture just amount of money, payer (insurance company), and some other information. There will be no information about claim in this model since it is not directly related.

For each claim record, we have “**Procedure**” model to capture each line because it is required for generating report to show balance for each procedure. Charge on every procedure is supposed to be paid by payer. However, it is possible and common that payer will not pay total amount but rather pay some and leave the rest to patient. This is why we have “**Charge**” model to keep track of remaining balance needed to be paid by insurance and patient. Charge for insurance is normally linked to procedure as one to one. But there can be several charges for patient for a procedure. With the current relation between procedure and charge, we are able to keep track of balance for each kind and show it on statement report.

Note that there are two types of payment which are payment from insurance and from patient as well. Only payment from insurance can be used to cover charges of insurance. The same goes to patient.

“**Apply**” model will capture amount of money from payment to charge. Note that it is not necessary that charge appearing on that claim needs to be paid in exact amount. For example, the company is charge $100. The insurance may say they will pay only $70 and patient is not supposed to pay anything as well. This is where we use “adjustment” field to modify balance. In this case, apply record will has $70 in amount and $30 in adjustment which makes balance zero.

**Admin Page (/admin/):**

The billing system is split into two parts:

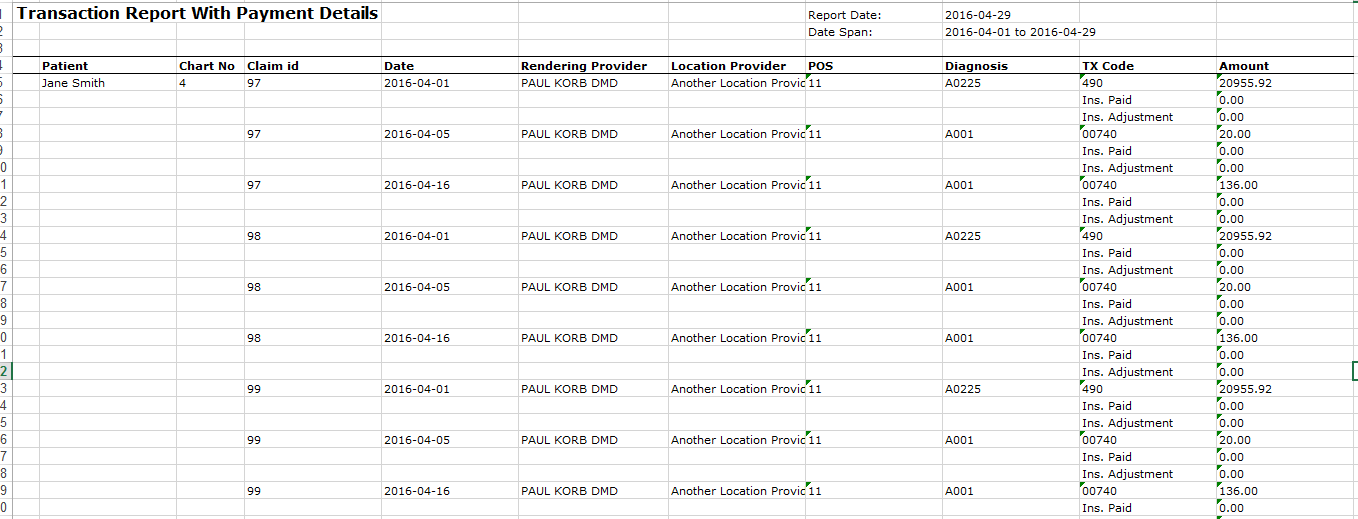
One is the front end that is accessible to the whole of billing team (through valid logins) that will allow them to add patient, guarantor, insurance and claim information.

The other is a backend – admin panel that will allow specific users to add, change and deleted existing records in all displayed models.

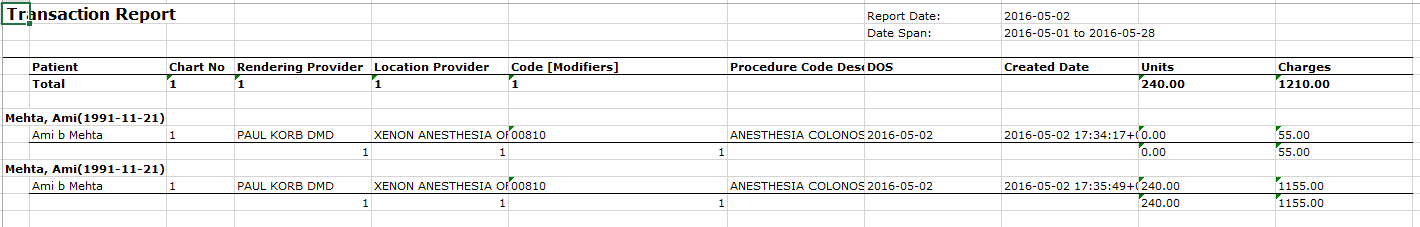
**Report Generation:**

The reports that are talked about in the flowchart are:

* Report generation with payment
  + This report includes a summary of payments made to and from patients on per person basis. This generated report should look like the one shown below.



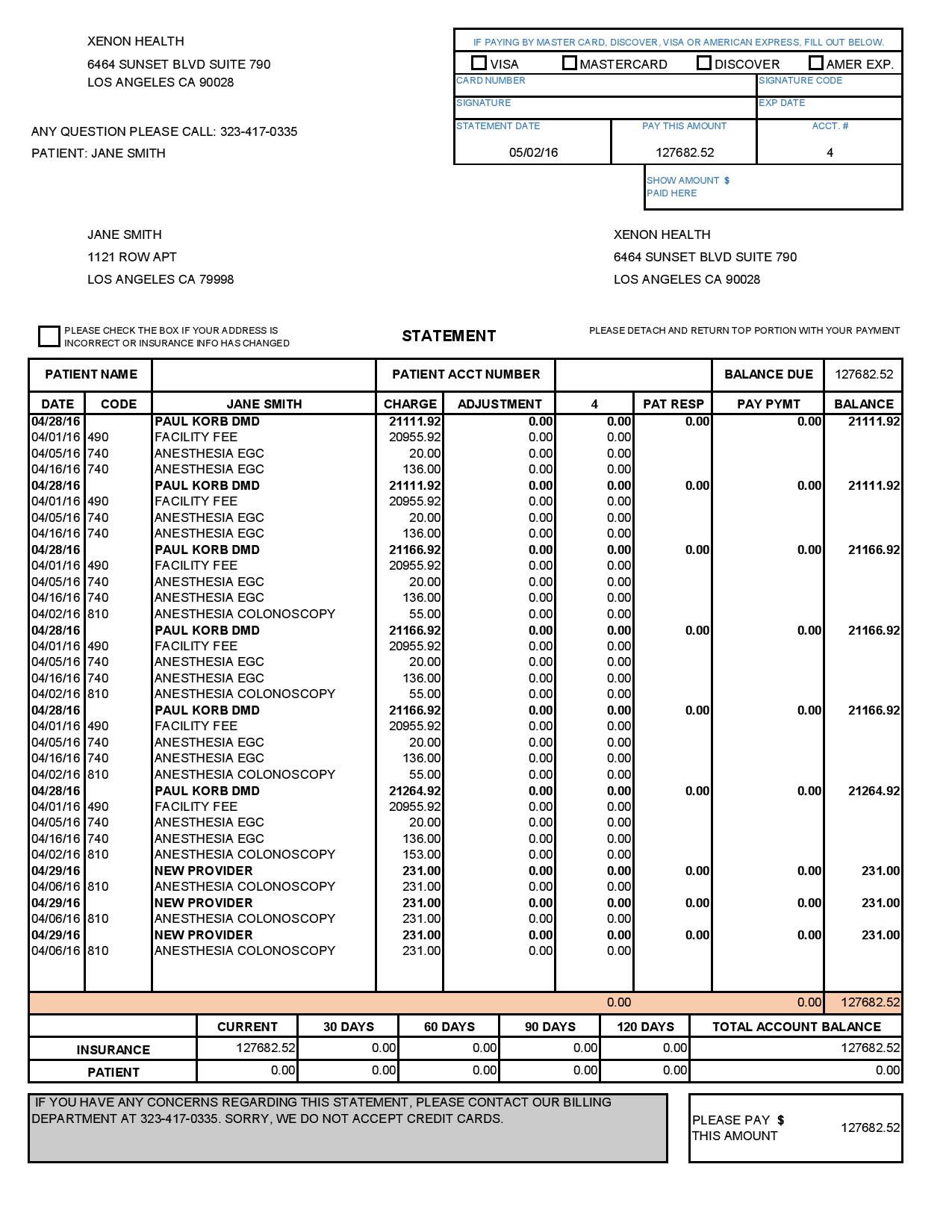
* + The TX code can include:
    - CPT code
    - Ins. Paid (the amount paid by the insurance company)
    - Ins. Adjustment: Amount adjusted for the insurance company
    - Co-pay: If any co-pa has to paid by the patient it will show up here
    - Deductible: If any deductible has to be paid by the patient it will be shown here
    - Other: IF any other extra amount has to be paid by the patient it will be shown here
    - Pat. Balance: The balance amount due for a patient
    - Note: Co-pay, deductible, other and pat. Balance will only be shown when the patient has been charged for it. (as you can see, the above screen shot doesn’t show pat information)
  + The end of report also includes a summary about charges, payments and adjustments.
* Report generation without payment
  + This report is almost is the same report with the payments; in this report we don’t process payment details, and we concentrate more on the procedure codes and units associated with a claim transaction.
  + As you can see in the screenshot this report is arranged per claim and whereas the report with payments is arranged per person.



* + This report includes a short summary after every claim and there is no cumulative summary in the end of the file.

**Statement generation:**

This part of reports is to generate statements for patients. These statements look like the one shown below. In short, it is a bill for the patient showing all amounts that are due.



Statement generation can be accessed from the dashboard and from individual patient records (patient dashboard -> patient id).

The key features of statement generation are:

* Date, CPT code, description of service, charges, adjustments (if any), ins paymnt, pat responsible, pay pymnt and balance.
* Each report includes a billing provider, rendering provider, patient information, balance range, personalized message, and of course, billing amounts.
  + All these can actually be filtered if they are searched from the statement report generation page (the link on dashboard).
* The statements reports are always saved from in the statement history list (found from the link on the dashboard).

**REFERENCES**

Django:

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

http://www.djangobook.com/en/2.0/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)