Microbiome Web Explorer Report

Snehal K. Talati

BNFO 620

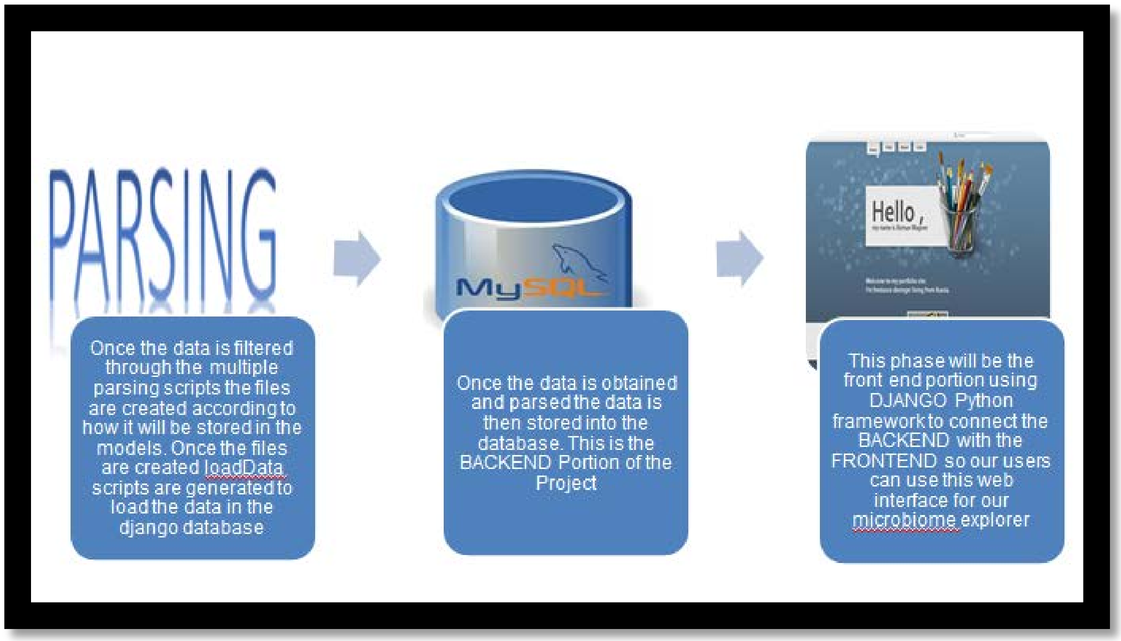
I. Introduction

Scope of the project was to develop a web-based Microbiome Explorer to allow searching and browsing of the analysis done by RDP-Classifier (taxonomy classification software). The biological component was clinically related to the Oral Microbiome, and Chronic Periodontitis Oral Disease. For this project a total of 92 subjects with different levels of chronic periodontitis, 88 of them whom were characterized at both sites. For our purposes in terms of the Web Application we had dropped some samples that were not able to map. Chronic Periodontitis is a common disease of the oral cavity consisting of chronic inflammation of the periodontal tissues caused by the plaque. This application allows a user to obtain analysis and link the Oral microbiome studies with Chronic Periodontitis through the means of operational taxonomic units using the Ribosomal Database Project (RDP) Classifier. The difference in the abundance of OTUs between the depth of the sites (shallow and deep) can be statistically analyzed by this application. This gives the user to obtain information for various OTU's that are present for each patient sample as well as clinically related information pertaining to each patient sample.

A universal primer was designed to amplify the V4-V6 region for oral microbial 16S rRNA sequences. These sequences were then used to obtain the microbial data for parsing information that was useful in analysis, and using RDP to obtain taxonomy data, which was then later incorporated into the web application. The frame work that was used for this application was DJANGO/PYTHON framework. DJANGO is a high level python web framework that encourages rapid development and clean, pragmatic design. The framework consists of object-relational mapper, automatic admin interface, elegant URL design, template system, cache system, and internationalization. All of this support from this framework made it an ideal middle piece framework for the Microbiome Application. PYTHON was the supporting language that was used which is a high-level programming language, its design emphasizes on the readability of the code. The python language offers multi-paradigm features such as object oriented, imperative, functional, and procedural reflective. HTML (Hyper Text Markup Language) was also used to provide the link and features that are present as well as a Cascading Style Sheet, which gives the web application or html page a particular format or style that is followed. JQuery language which is JavaScript is also used to enhance the appearance features on any given web-based application.

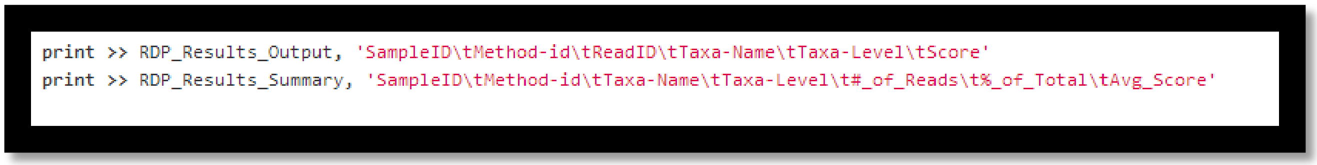
II. Work Flow:

Figure 1. Diagram which represents the Work Flow of Project


PARSING
Once the data is filtered through the multiple parsing scripts the files are created according to how it will be stored in the models. Once the files are created loadData scripts are generated to load the data in the django database
MySQL
Once the data is obtained and parsed the data is then stored into the database. This is the BACKEND Portion of the Project
Hello,
This phase will be the front end portion using DJANGO Python framework to connect the BACKEND with the FRONTEND so our users can use this web interface for our microbiome explorer

The work flow present in the diagram is illustrated in Figure 1 to give the reader of this report a visual guide to follow. The first phase of this work flow is the parsing of the data. We were given many datasets to work with. The first dataset was a directory of patient samples in FASTA format that contained about 84 samples of which one was deleted as it we could not map that particular sample. Once we had the set of fasta files we used these files and ran it through RDP Classifier to create two new files which were tab delimited files profile summary and read assignment files. Figure 2 below shows how these files were structured.

Figure 2: Structure of the RDP Profile Summary and RDP Read Assignment Files

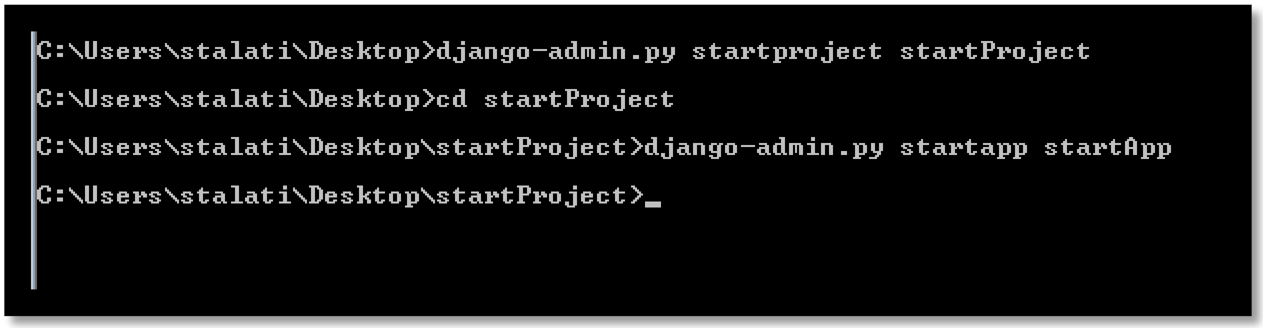


```
print >> RDP_Results_Output, 'SampleID\tMethod-id\tReadID\tTaxa-Name\tTaxa-Level\tScore'
print >> RDP_Results_Summary, 'SampleID\tMethod-id\tTaxa-Name\tTaxa-Level\t#_of_Reads\t%_of_Total\tAvg_Score'
```

The read assignment file was dropped later on due to the time constraints and size of the file to load in the web application. The rest of the files that were created were created from an oral clinical file that was an excel file given to us. One file contained the details for each patient and all the attributes that were associated with that particular patient. The second file gave us details on what date the samples were ran. The barcodes for these samples were taken from fasta file names that were given in the directory. These barcodes were added as well as the run date to the attributes file. A pipeline was created to do all the parsing to create a sample attributes master table, sample file, taxonomy file, classification method file, project information file, and profile summary file. All of these files were created so they are tab delimited. The reason for the files to be tab separated it so all of this information will be used to store into the database models.

Once the first command is used the very first command that is used to start DJANGO after installation is django-admin.py <projectName> followed by django-admin.py startApp <appName>. Once these steps are create all the files are created to start making the Django application. Once the file is created these are the commands need to be run:

Figure 3: Screenshot to start Django

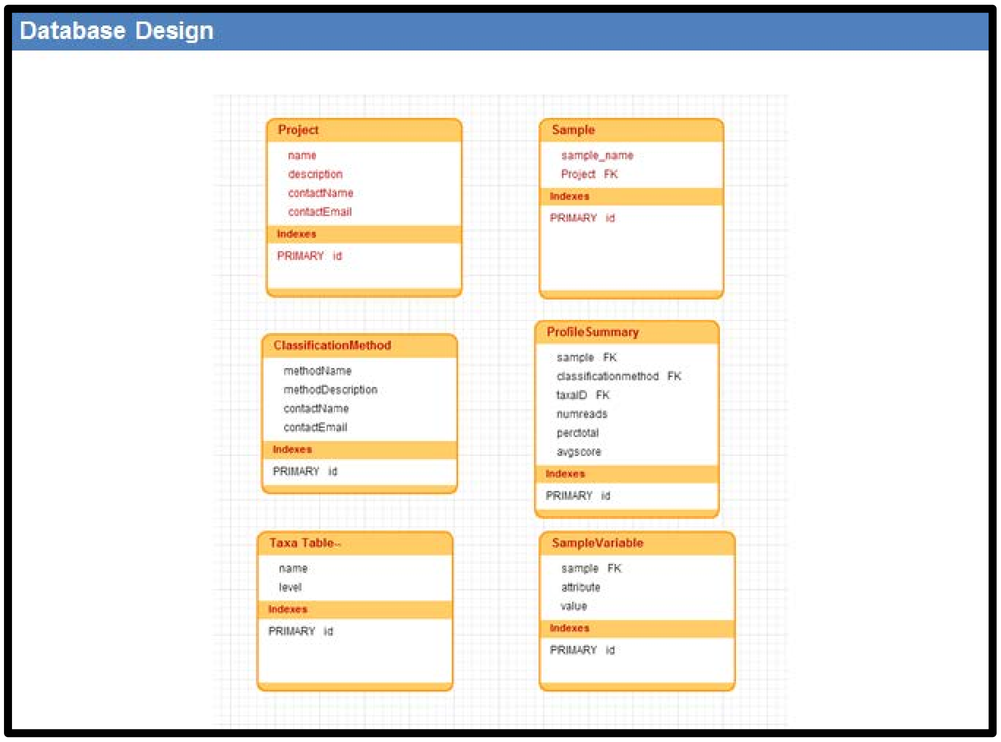


```
C:\Users\stalati\Desktop>django-admin.py startproject startProject

C:\Users\stalati\Desktop>cd startProject

C:\Users\stalati\Desktop\startProject>django-admin.py startapp startApp

C:\Users\stalati\Desktop\startProject>_
```

After successful creation of the folders the django framework creates files that have to be manipulated in several places.

The second phase is where we create the models in django that correlate with the information that we have in our files that we created in the first phase. The way the models are defined in order to search properly in our web application is using the auto key that is generated by Django.

Figure 4 Database Design



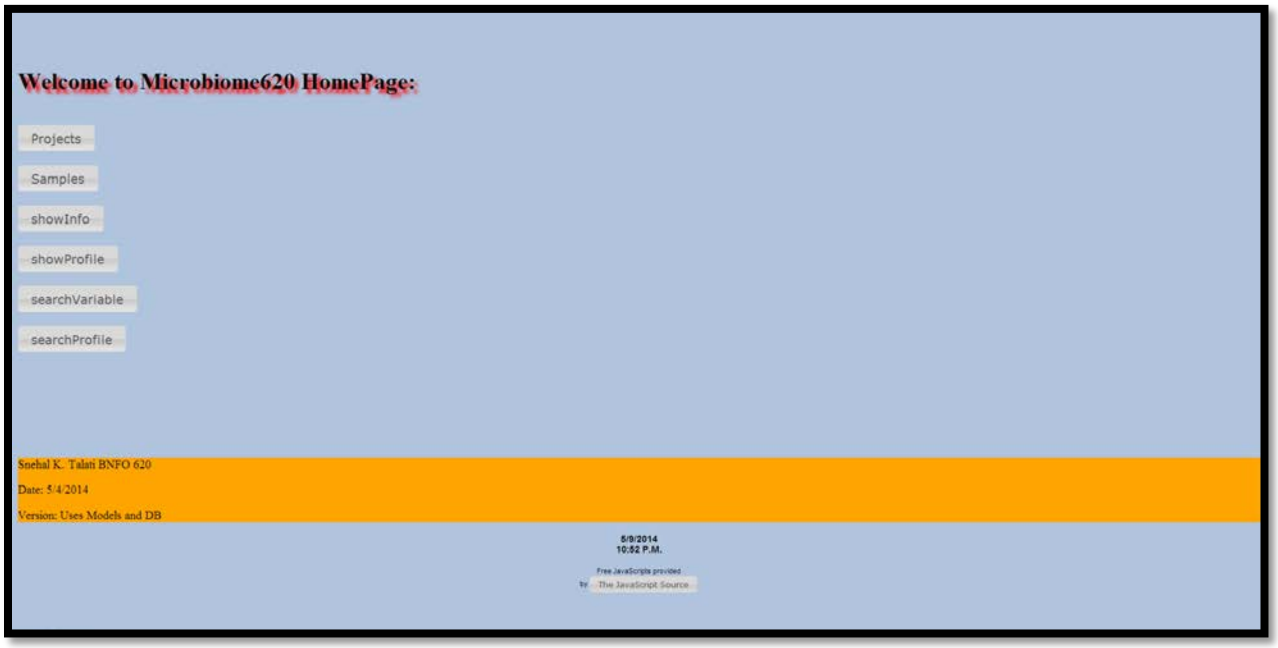
Once the models are created loading scripts were created to load the data into the database from the files from the first phase that we created. In this process to view the actual data and verifying to make sure the data is in the database admin.py is modified to view that the data is correctly inputted. This wraps up pretty much the second phase once the data has been loaded.

The last phase is the actual phase that the web application is connected with the various parts that exist within the app and the project as we created with the commands from figure 3. The way I approached this was to start off by adding the urls to urls.py and making sure a page is created for every link that we will have. However, before I do that I make sure I added the app in settings.py making sure when I run the program it executes properly. After the urls.py is updated then I create a template folder within the project directory and make the .html template pages for every link that I have in my urls.py.

Figure 5 URLS.py

```
urlpatterns = patterns('',
    # Examples:
    # url(r'^$', 'MicrobiomeDB.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),

    url(r'^admin/', include(admin.site.urls)),
    url(r'^$','MicrobiomeSnehalApp.views.home'),
    url(r'^projects','MicrobiomeSnehalApp.views.projects', name='projects'),
    url(r'^showInfo','MicrobiomeSnehalApp.views.showInfo', name='showInfo'),
    url(r'^showProfile', 'MicrobiomeSnehalApp.views.showProfile', name='showProfile'),
    url(r'^sample', 'MicrobiomeSnehalApp.views.samples', name = 'sample'),
    url(r'^searchVariable', 'MicrobiomeSnehalApp.views.variablesearch', name = 'variablesearch'),
    url(r'^searchProfile', 'MicrobiomeSnehalApp.views.profileSearch', name = 'searchProfile'),
    #url(r'^showProfile','MicrobiomeSnehalApp.views.showProfile', name='showProfile'),
)
```

Once the urls.py, settings.py, forms.py, and the template pages are created in the template folder the next step is to change the views.py where the actual view is created. The views is what has how it will be displayed the user.



Welcome to Microbiome620 HomePage:

Snehal K. Talati BNFO 620

Date: 5/4/2014

Version: Uses Models and DB

5/9/2014
10:52 P.M.

Free JavaScripts provided
by The JavaScript Source

Figure 6 Homepage for Microbiome 620

Figure 6 shows the homepage of how it looks the time at the bottom of the page is JavaScript which I tried to learn and implement. This is the next step one can take is to use jquery plugins like HighCharts to allow the user to actually visualize the data.

This concludes this report and the web-application gives basic functionality to go through the oral microbiome data. Along the way there were many skills that were obtained such as using a version control called GitHub which allowed us to collaborate with our team members from our group. Obtaining this skill to use version control software like GitHub has been very valuable and was a great way to keep track of the changes that are made in this framework. Mainly because DJANGO requires you to change many things in multiple places to connect the back end with the front end. Overall I would like to give a special thanks to our teaching assistant Megan Martinez and professor Nihar Sheth for teaching us very marketable skills especially the introduction to GitHub. Also another special thanks to my team members Hardikh, Bryan, Archana, and Rahul.