

**BACK-END SYSTEM SOFTWARE DEVELOPMENT OF POCKET PRAISE USING
DJANGO FRAMEWORK**



**Cluster Innovation Centre
University of Delhi**

RAHUL YADAV

May 2015

**Dissertation submitted in partial fulfilment for the degree of
B.Tech (Information Technology and Mathematical Innovations)**

**BACK-END SYSTEM SOFTWARE DEVELOPMENT OF POCKET PRAISE USING
DJANGO FRAMEWORK**

**APPROVED BY SUPERVISING
COMMITTEE:**

Nitin Nain
Co-founder, Pocket Praise

Twisha Dhingra
Assistant Professor,
Cluster Innovation Centre,
University of Delhi

Certificate of Originality

The work embodied in this report entitled “**Back-End System Software development of Pocket Praise using Django Framework**” has been carried out by me at **Impact Ventures**. The manuscript has been subjected to plagiarism test by Turnitin. I declare that the work and language included in this project report is free from any kind of plagiarism.

The work submitted is original and has not been submitted earlier to any institute or university for the award of any degree or diploma.

Rahul Yadav

Acknowledgement

It has been a great honor and privilege working with **Impact Ventures**. I am very grateful to **Mr. Nigel Hembrow** for giving me an opportunity to work with his firm.

The successful completion of this project required efforts from several intellectual minds. I would like to extend my sincere and heartfelt obligation to my external mentor **Mr. Nitin Nain** for providing me with all the support and technical skills to meet the project requirements. His coordination, guidance and willingness to share knowledge helped in my working process and in the successful completion of the project.

I am highly grateful to my internal project mentor **Ms. Twisha Dhingra** whose invaluable guidance and constant faith in me helped me to understand and implement things in the better way.

Rahul Yadav

Abstract

BACK-END SYSTEM SOFTWARE DEVELOPMENT OF POCKET PRAISE USING DJANGO FRAMEWORK

By

Rahul Yadav,

Cluster Innovation Centre, 2015

SUPERVISOR: Nitin Nain

This project introduces the process of developing back-end system software which is used by Impact Venture's new venture called Pocket Praise. This software has mainly three major components: restaurants, customer service individuals and users who visit those restaurants. The users can provide ratings to the customer service individuals based on the service provided to them when they visited the restaurant, via the Pocket Praise mobile app. The data that is generated through the app is handled via this software. To implement this software I have used a tool called Django Framework. This project introduces how I have used Django Framework to build such software and how this software handles user request to retrieve or post data via the mobile app.

Table of Contents

Acknowledgement	i
Abstract.....	ii
Chapter 1: Introduction.....	1
1.1 Pocket Praise: Empowering Customer Service Individuals.....	1
1.2 Pocket Praise: Impact.....	2
1.3 Required Work.....	2
1.4 Limitations	2
1.5 Report Organization.....	2
1.6 Activity and Work Planning	3
Chapter 2: Preliminary.....	4
2.1 Django Framework	4
2.1.1 Model	6
2.1.2 View	6
2.1.3 Template.....	6
2.1.4 Database	6
2.2 Python	7
2.3 OAuth.....	7
2.4 Django REST Framework.....	8
Chapter 3: Project Overview	9
3.1 An overview of the project	9
3.2 Django Framework Development Process.....	9
Chapter 4: Working with Database Tables.....	11
Chapter 5: Using the Software: Registration, APIs and Testing	13
Conclusion	17
References	18

List of Figures

Figure 1: Django Server Initialization	4
Figure 2: Django Generated Files.....	4
Figure 3: Django Project Initialization	5
Figure 4: A Django Request	5
Figure 5: Adding Django OAuth and REST Framework	7
Figure 6: Django with Elastic Beanstalk	10
Figure 7: Favorite Servers Record.....	11
Figure 8: Favorite Servers Table in Database	12
Figure 9: Servers Table in Database	12
Figure 10: Customer Table in Database	12
Figure 11: Django URLs for Pocket Praise	14
Figure 12: JSON data for venue_id =1	15
Figure 13: A GET Request Example	15
Figure 14: Response to the GET Request.....	16

List of Tables

Table 1: Project Timeline	3
---------------------------------	---

Chapter 1: Introduction

This project report introduces the process of developing back-end system software, what it does, how it works and how it can be maintained in future by other developers. This software is used by Impact Venture's new venture called Pocket Praise. This software system is build using Django Framework [1]. Django is written in Python [2] and is an open source web application framework. This back-end system software has three major components which are namely- restaurants, customer service individuals and users who visit those restaurants. In this project report will explain the details of using Django to build this system. The techniques and procedure explained here can be applied to build any such complex software, database-driven websites or an application.

1.1 Pocket Praise: Empowering Customer Service Individuals

The concept on which Pocket Praise focuses upon is “Global Platform for Customer Service”. The mission of Pocket Praise is to elevate great customer service. Every day we receive services, sometimes they are good and sometimes bad. We notice bad service and we also notice great service. Sometimes we say thank you or perhaps we tip. We share with our friends the stories of really bad service and also really delightful service. Overall, we all like to be served.

The current scenario is that the online world focuses on either the venue or on the employer. The venues are able to receive online reviews which helps boosting their profiles. Great favorable reviews are great for their business.

But the transparency that LinkedIn [3] brought to organizations in the white color industries has not yet happened in the service industry. Pocket Praise enables the customer to review the customer service individuals instead of the venue.

Pocket Praise (app) is a simple & easy-to-use mobile application for customers to rate and review the service experience provided to them by customer service individuals. Customers can share their reviews on social media. Service individuals can improve their careers via Pocket Praise.

1.2 Pocket Praise: Impact

This idea to rate service individual is fundamentally different and novel in its own way. For the first time through Pocket Praise customers can connect to, follow and communicate with the individuals they appreciate and rely on. The service staff can get feedback to help them improve, build a strong online profile which could prove helpful for developing their career. They can also receive career opportunities and get access to training opportunities. Feedback forms, online reviews, and monetary tipping have been around forever. Now, customers will be able to do this online and on mobile get satisfaction from saying positive words, and knowing they can help improve someone's life.

1.3 Required Work

Pocket Praise has two parts. One is the front-end which is a mobile app which is to be used by everybody. The second part is the back-end system software to aid the mobile app which I implemented during my period as an intern. This back-end system software is built using Django web application framework. Django was basically developed for the news oriented site of the world company in Kansas [4]. It simplifies and accelerates the development process of complex web and mobile applications like a photo sharing website [5]. Django's excellently designed framework is constantly under development and at present, it includes four major parts: Models, APIs, Settings and URLs. Our software consists of three major components which are Servers, Venues and Customers.

1.4 Limitations

Complex back-end software such as the one I made here generally takes some time to test and validate. Due to the development time constraint the software might have some bugs. Django is extremely fast, secure, comes with a lot of add-on apps and highly scalable which makes it easier to remove bugs and improve the created software.

1.5 Report Organization

Chapter 1 introduces the concept of Pocket Praise, describes the scope of work and limitations. In next Chapter, which is Chapter 2, I provide the preliminaries of the report, basically an introduction to Django framework and its components. The chapter also introduces the software tools used in the project like OAuth [6] and REST Framework [7].

Chapter 3 is an overview of the whole project and includes the details of the development process of the software. Chapter 4 provide the details of each database component in Django linked to this software and how to work with Django database to create tables for users, servers and venues. Chapter 5 includes the details of user registrations, the application programming interfaces and finally the testing of this software.

1.6 Activity and Work Planning

The following table shows the timeline I followed during the period of my internship:

Timeline	[17-3-2015 to 20-5-2015]							
Task List	1	2	3	4	5	6	7	8
Django Installation and basic understanding								
Learning Python and Django								
APIs								
Prototype v0								
Prototype v1								
Testing								
User Testing and Reports								

Table 1: Project Timeline

Chapter 2: Preliminary

2.1 Django Framework

Django is a high level open source framework written in Python which helps to develop web applications. Generally, to write a web application/software, we need a Web Server Gateway Interface, Routing, SQL interface and templates to make dynamic web pages [9]. Django is basically a collection of all these things we need and a lot more features which could be included in a Django project to develop our app in a faster and efficient manner. The operating system I have used throughout the project is Ubuntu. To interact with Django and Python, commands are called in terminal. These commands create Django projects and applications inside the project. Django is then instructed to initiate its development server using these commands.



```
rahul@rahul-HP-ProBook-4431s:~/internship_project/pocketpraise$ python manage.py
runserver
Performing system checks...

System check identified no issues (0 silenced).
May 18, 2015 - 04:40:33
Django version 1.8b1, using settings 'pocketpraise.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Figure 1: Django Server Initialization

When we create a Django project or an application, several files are automatically generated. Most of these files already have a range of functionalities.

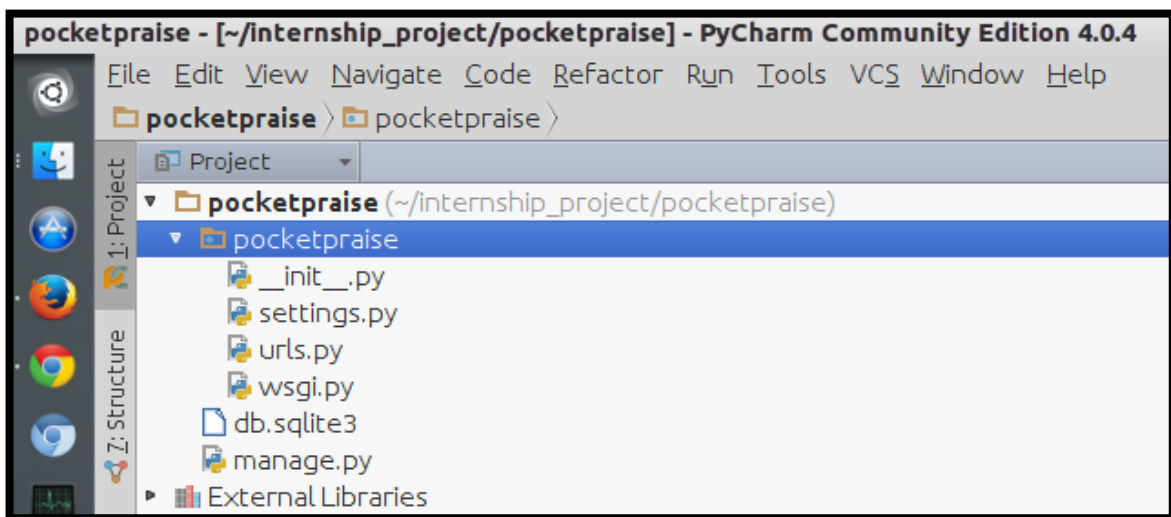


Figure 2: Django Generated Files

```
rahul@rahul-HP-ProBook-4431s: ~/internship_project/pocketpraise
rahul@rahul-HP-ProBook-4431s:~$ mkdir internship_project
rahul@rahul-HP-ProBook-4431s:~$ cd internship_project/
rahul@rahul-HP-ProBook-4431s:~/internship_project$ django-admin startproject poc
ketpraise
rahul@rahul-HP-ProBook-4431s:~/internship_project$ cd pocketpraise/
rahul@rahul-HP-ProBook-4431s:~/internship_project/pocketpraise$ python manage.py
migrate
Operations to perform:
  Synchronize unmigrated apps: staticfiles, messages
  Apply all migrations: admin, contenttypes, auth, sessions
Synchronizing apps without migrations:
  Creating tables...
    Running deferred SQL...
  Installing custom SQL...
Running migrations:
  Rendering model states... DONE
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying sessions.0001_initial... OK
```

Figure 3: Django Project Initialization

The settings.py file is one of the most important file in a Django project. This file is global for the project and for all added applications. It contains information regarding path storage of files, included applications, host/domain names and middleware. The Object-relational mapping is an effective tool provided by Django for database management. It supports most of the databases such as MySQL, Oracle, PostgreSQL and SQLite. I started the project with Django v1.8 which is till date the latest version of the framework. To transfer over to other newer version the administrator should comply with the changelog to avoid any errors in the software. An example of Django request made by a user is shown in Figure 4.

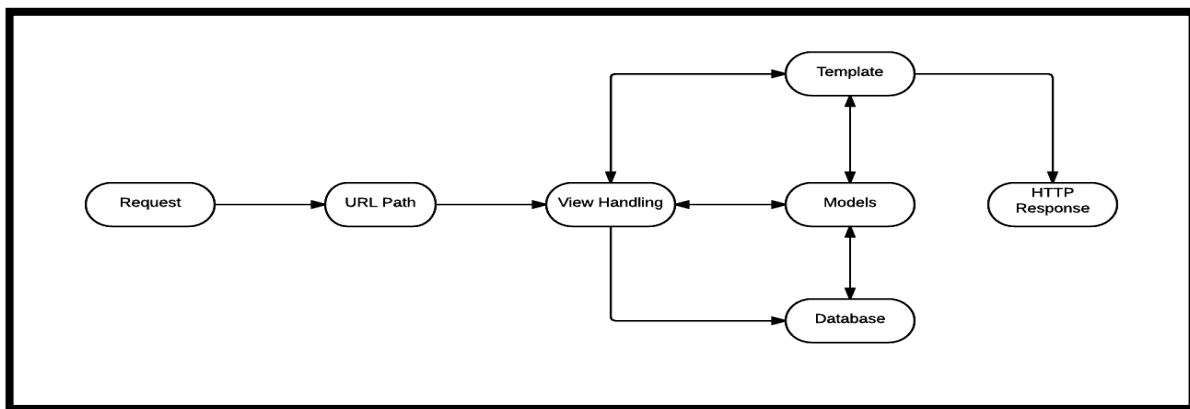


Figure 4: A Django Request

2.1.1 Model

A model [1] in Django is a single, definitive data source containing the essential fields and behaviors of the data. In general, one model is one table in the database. Every single attribute in the model represents a field of a table in the database. Django provides a set of automatically generated database application programming interfaces (APIs) for user convenience.

2.1.2 View

A view [1] is a short form of view function. It is a file containing Python function which takes web requests and returns web responses. A response can be an image or HTML content or XML documents or a 404 error, and so on. The desired output is generated via the written arbitrary logic contained in view. Linking the view function with a particular URL requires to use a structure called URLconf which maps URLs to view functions. Django lets users to choose how to create URLs without any limitations.

2.1.3 Template

Django's template [1] is basically a simple text file which can generate a text based format like HTML and XML automatically. Thus, the template contains variables and tags. Variables are replaced by the result when the template is evaluated. The logic of the template is controlled by tags. Filters can be used to modify variables. As an example, an uppercase filter can convert the variable from lowercase into uppercase.

2.1.4 Database

The default database in Django is SQLite. It is also already included in Python. We only need one command to set up database in Django:

```
DATABASE_ENGINE = 'sqlite3'
```

It is also possible to include multiple databases if needed. External databases need username and password. The user must migrate them using terminal accordingly. The models.py file is used for creating database tables and fields. Tables and fields can be added manually using either terminal or admin interface or using database editors such as SQLite Manager. Any changes to the database must be followed by updating the database tables using following commands:

```
$ python manage.py makemigrations
```

```
$ python manage.py syncdb
```

2.2 Python

Python [2] is an open source object oriented programming language used to build the Django Framework. Python is a dynamic scripting language which is quiet similar to Perl and Ruby [2]. It has a garbage collector for automatic memory management and python also support dynamic semantics. Python has a support for modules and packages which also encourages program modularity and reusability of code. A lot of code editors are available for writing quicker and neat codes in Python. I have used an editor by JetBrains called PyCharm for its intelligent coding assistance and language aware code completion.

2.3 OAuth

OAuth [6, 10] is an open standard for authorization. It allows notifying a resource provider say LinkedIn that the resource owner say me, grants permission to a third-party say Pocket Praise access to their information for example, your connections. OAuth allows for access at different levels i.e., read-only or read-write allowing an application like Pocket Praise to automatically synchronize your LinkedIn contacts. OAuth allows to access granularity, i.e., allowing LinkedIn user to choose what to share and what not to with other apps using LinkedIn's APIs. It also let users to manage access from the resource provider's application which means there is a provision for revoking access at any time. The Django OAuth Toolkit package provides OAuth support and works with Python.

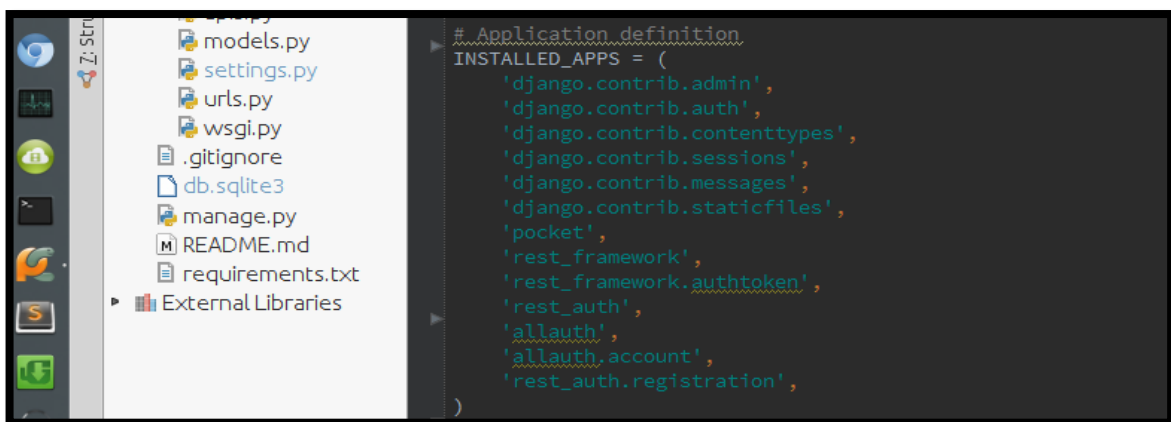


Figure 5: Adding Django OAuth and REST Framework

2.4 Django REST Framework

REST stands for Representational State Transfer. It is an architectural style for software which consists of guidelines and best practices for creating scalable web applications. Django REST framework [7] is basically a powerful and sophisticated toolkit that makes it easier to build Web APIs. It offers web browse-able version of the API we create and there is an option of returning raw JSON.

Chapter 3: Project Overview

3.1 An overview of the project

The desired result of the back-end system software is an integrated Amazon Web Services based application. It includes the following major components:

- Restaurant

A restaurant is basically a venue where say, food is served for example say, QD's. Now, websites like Zomato have features to rate that restaurant based on food. Venues such as QD's are able to receive online reviews through Zomato which helps boost their profiles.

- Server

A server say Shyam is a customer service individual who for example serves at QD's. Now, a server went unnoticed even though his service is integral to both the restaurant and his career. He might get a tip for his performance and services, but that doesn't help him professionally.

- Users

Users are the customers. They are the ones who visit the restaurant. Let's say Vasu went QD's. She liked the food there and was also impressed by the service that Shayam provided. To reward the restaurant she writes a beautiful online review about QD's but she could not do any such thing for Shyam.

Until now, the focus has been on the employers and not on the service individuals. Pocket Praise offers the customers to rate and review the service provided to them, which service individuals can use to improve themselves and build a strong online profile.

3.2 Django Framework Development Process

Building such complicated system needs proper Database, APIs, Functions to interact between them and User Interface. Sufficient functionalities are provided in Django framework to implement these components. Django has model corresponding to database, for user interface we have templates and the mobile application, OAuth for verifications and REST framework to deal with the API part. The Model component in Django helps maintaining tables in the database and its template component helps to write dynamic html

using a combination of both html syntax and Django syntax. APIs are handled via the Django REST framework. The functions in between are managed by a view component in Django which reads the request from user interface analyze the request as to whether it's a get or post request and makes corresponding changes in the database. To keep the software up and running for the mobile application, Pocket Praise uses a cloud service by Amazon Web Services called Elastic Beanstalk. It is maintained by my mentor Mr. Nitin Nain.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': ' ', # os.environ['RDS_DB_NAME'],
        'USER': 'master', # os.environ['RDS_USERNAME'],
        'PASSWORD': '*****', # os.environ['RDS_PASSWORD'],
        'HOST': 'aagrtzmjfoeu12.cggapxxx9cv4.ap-southeast-1.rds.amazonaws.com', # os.environ['RDS_HOSTNAME'],
        'PORT': '5432', # os.environ['RDS_PORT'],
    }
}
```

Figure 6: Django with Elastic Beanstalk

Chapter 4: Working with Database Tables

To start building the back-end software, we need to create a database. All the tables in the database for this software include information for restaurant, servers or customer service individual and users or customers and so forth. These database tables are created initially when the back-end system software is deployed. A little information has to be inserted into the database at the beginning, such as venue information and server information. However, most of the information about these things will be inserted or updated in the database dynamically, for example, choosing a favorite server. Every time I want to create a new favorite server, I will basically insert a tuple into the favorite server table to make the database consistent with the real world.

To have a favorite server table in the database, I first need to select which database system I am going to use. As I have already discussed in section 2.1.4, Django supports almost all popular databases. The one I used for this back-end system software is SQLite3 when testing the software locally. Migration to PostgreSQL is required when moving to AWS. Writing just one line of code sets up the database for the user:

```
DATABASE_ENGINE = 'sqlite3'
```

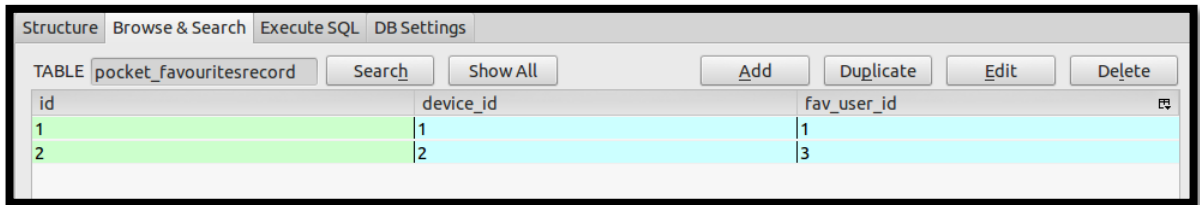
Now, I create the favorite user table in the database. Django framework uses a class known as model to represent the database table schema. Creating a table we need writing a new class derived from model class. For example:

```
class FavouritesRecord(models.Model):
    id = models.IntegerField(primary_key=True),
    device_id = models.CharField(max_length=255)
    fav_user_id = models.CharField(max_length=255)
    favoured_on = models.DateField
```

Figure 7: Favorite Servers Record

This part of the code creates a table called `pocket_favoritesrecord` which stores the information of favorite servers. This table has two columns: device id which corresponds to the customer of the users group who visited the restaurant and favorite user id that corresponds to the customer service individual. Device id and favorite user id are foreign keys that refer to two other tables- users and servers respectively. The structure of these

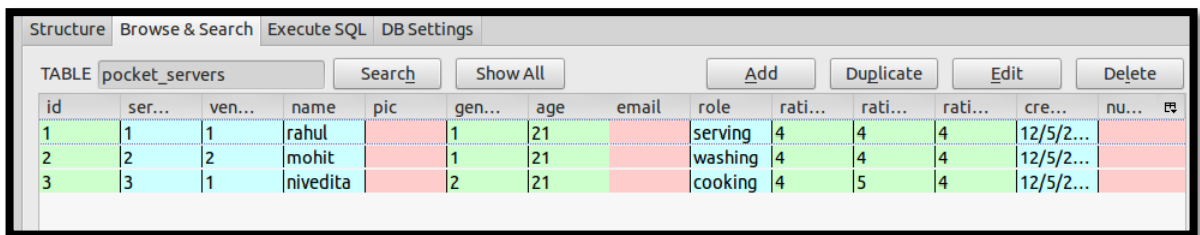
two tables is very similar with the favorite record table. This is how the actual pocket_favoritesrecord table in the database looks like:



id	device_id	fav_user_id
1	1	1
2	2	3

Figure 8: Favorite Servers Table in Database

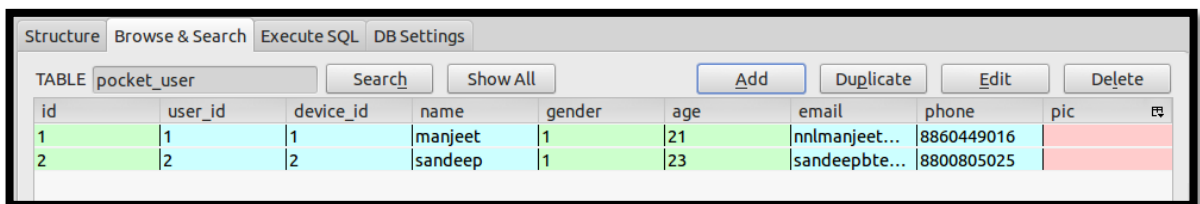
Server component also has its own table, which used to create this favorites record table. It is a table which has already been introduced above, and server table is the one which has the record of all the servers at all the venues.



id	ser...	ven...	name	pic	gen...	age	email	role	rati...	rati...	rati...	cre...	nu...
1	1	1	rahul		1	21		serving	4	4	4	12/5/2...	
2	2	2	mohit		1	21		washing	4	4	4	12/5/2...	
3	3	1	nivedita		2	21		cooking	4	5	4	12/5/2...	

Figure 9: Servers Table in Database

User/ Customer table, venues table and servers table are defined in the core data component and provides information which is used all over the back-end system software. If a waiter no longer works at a restaurant we can update the corresponding tables to match. A total of twenty such tables were created for the system software.



id	user_id	device_id	name	gender	age	email	phone	pic
1	1	1	manjeet	1	21	nnlmanjeet...	8860449016	
2	2	2	sandeep	1	23	sandeepbte...	8800805025	

Figure 10: Customer Table in Database

Chapter 5: Using the Software: Registration, APIs and Testing

The project involves the signing in of users and customer service individuals for the mobile app to work for either of them. This is done via the android app. At first, the user must register to Pocket Praise with sufficient personal information. At present, the android app supports signing in through LinkedIn, Google+ and Facebook. Django's allauth add-on consists of several attributes including

- Username
- Password
- Email
- First Name
- Last Name
- Forgot Password
- Verification of email

These verifications happens at the mobile application end. Considering these attributes to be accurate during signing up, I have implemented these in models.py file.

The communication between the Django database and mobile application is aided via the APIs created in the apis.py file. An example of getting a record of favorite servers is given as under:

```
def get_user_favourites(x, device_id):
    fav_list =
    FavouritesRecord.objects.filter(device_id=device_id)

    if not fav_list:
        return JsonResponse({})
    else:
        members = list()
        count = 0

        for server in fav_list:
            server_data =
            Servers.objects.filter(server_id=server.fav_user_id)
            if not server_data:
                continue
            else:
                for s in server_data:
                    avg = (s.rating1 + s.rating2 + s.rating3)
```

/ 3

```
members.append({  
    "id": s.server_id,  
    "name": s.name,  
    "venueId": s.venue_id,  
    "pic": s.pic,  
    "mf": s.gender,  
    "age": s.age,  
    "role": s.role,  
  
    "rating": {"avg": avg, "param1":  
s.rating1, "param2": s.rating2, "param3": s.rating3},  
    "favouritedOn": timezone.now()  
})  
  
count += 1  
  
return JsonResponse({'numReviews': count, 'people':  
members})
```

This API provides the list of all the servers that any particular user has favorited. If the user does not have a favorite server the API returns an empty JSON.

The defined URLs in the file `urls.py`, we have the following urls available for APIs to post or receive the data:

```
1. ^api/setdeviceid/(?P<device_id>.+)/$  
2. ^api/venue/details/(?P<device_id>.+)/(?P<venue_id>.+)/$  
3. ^api/server/rate/add/(?P<device_id>.+)/$  
4. ^api/server/rate/(?P<device_id>.+)/$  
5. ^api/favourite/set/(?P<device_id>.+)/(?P<favourited_users_id>.+)/$  
6. ^api/favourite/(?P<device_id>.+)/$  
7. ^api/history/(?P<device_id>.+)/$  
8. ^api/user/profile/update/(?P<device_id>.+)/$  
9. ^api/user/profile/(?P<device_id>.+)/(?P<user_id>.+)/$  
10. ^password/reset/$ [name='rest_password_reset']  
11. ^password/reset/confirm/$ [name='rest_password_reset_confirm']  
12. ^login/$ [name='rest_login']  
13. ^logout/$ [name='rest_logout']  
14. ^user/$ [name='rest_user_details']  
15. ^password/change/$ [name='rest_password_change']  
16. ^signup/$  
17. ^api/demo/venue/details/(?P<device_id>.+)/(?P<venue_id>.+)/$  
18. ^api/demo/user/favourites/(?P<device_id>.+)/$  
19. ^api/demo/user/history/(?P<device_id>.+)/$  
20. ^api/demo/user/profile/(?P<device_id>.+)/(?P<user_id>.+)/$
```

Figure 11: Django URLs for Pocket Praise

The details associated with the venue (id=1) can be retrieved through API corresponding to URL number 2 in Figure 11. The obtained data is in JSON format as seen in Figure 12.

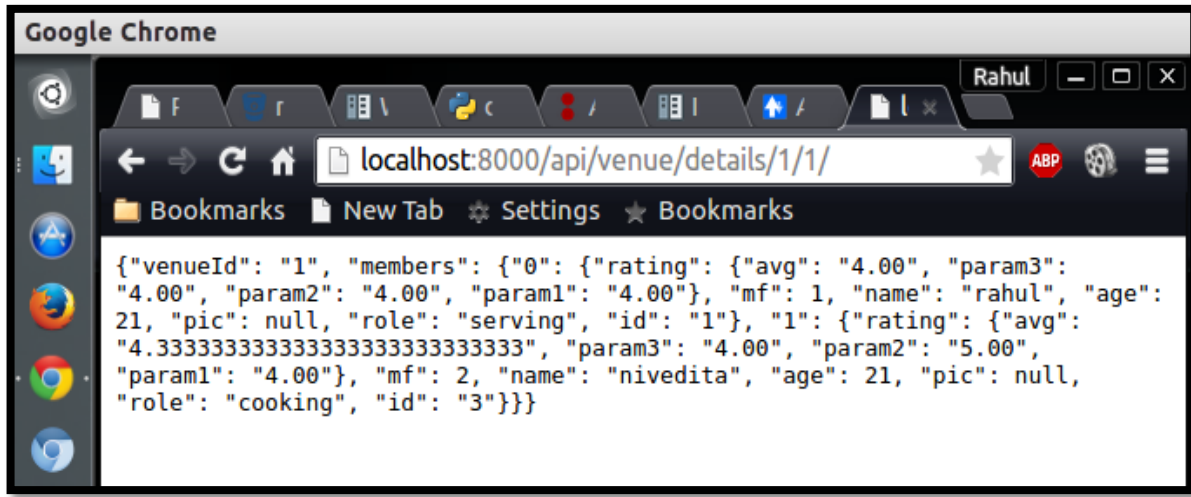


Figure 12: JSON data for venue_id = 1

For testing the software we used an extension available for Google Chrome called Advanced Rest Client [11]. This extension or program helps the web developers to create and test customized HTTP requests. The user activities in the Pocket Praise mobile app creates similar kind of requests that either makes changes to the database i.e., a POST Request or just receives information from the database i.e., a GET Request (see Figure 13).

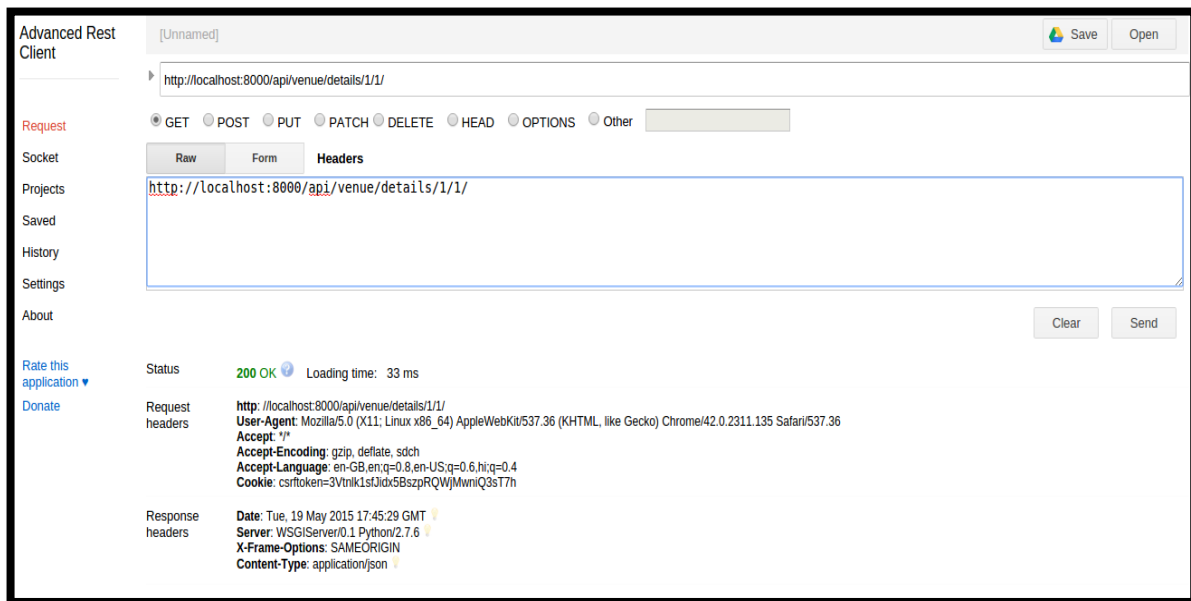


Figure 13: A GET Request Example

The response of the GET Request made to extension is shown in Figure 14. It can be checked that the response is same as we obtained in Figure 12. Similarly, we can write JSON data to the database using post request and test that the software is working.



Figure 14: Response to the GET Request

Conclusion

The Django framework gives us a simple and reliable way to create the back-end system software of Pocket Praise. It provides powerful functionalities and concise syntax to help programmers deal with the database, the web requests and the logic put into it.

The experience of developing the APIs component in the system also helped me learning more of web programming with Django. Though, I didn't actually get to work with AWS, I saw how it works and how it keeps the system online 24x7. Within the Django framework, I have successfully accomplished the requirements of the system. Once this back-end software clears the testing phase, it can be used to serve the required purpose. The next phase of this software could be to implement Hadoop to make it scalable. Machine Learning algorithms can be implemented to find duplicate users and also the fake ones.

Although the software have been tested by various means, any future problems could be fixed easily. New features can be added to the software as Django further develops.

References

- [1] The Django Book Homepage, <http://www.djangobook.com/en/2.0/index.html>, February 2015.
- [2] Python Documentation, <https://www.python.org/doc/>, February 2015.
- [3] LinkedIn Homepage, <https://www.linkedin.com/>, February 2015.
- [4] Django Framework Homepage, <https://www.djangoproject.com/>, February 2015.
- [5] Instagram Engineering Webpage, <http://instagram-engineering.tumblr.com/post/13649370142/what-powers-instagram-hundreds-of-instances>, February 2015
- [6] OAuth Homepage, <http://oauth.net/>, March 2015
- [7] Django Rest Framework Webpage, <http://www.django-rest-framework.org/>, March 2015
- [8] Stack Overflow Homepage, <http://stackoverflow.com/>, February 2015.
- [9] Effective Django Homepage, <http://www.effectivedjango.com/index.html>, February 2015.
- [10] Authomatic Homepage, <http://peterhudec.github.io/authomatic/index.html>, March 2015.
- [11] Advanced Rest Client for Chrome, <https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddfnphfgcellkdfbfbjeloo>, April 2015