

**A
Report
On
Industrial Training
At**

“PYTHON WITH DJANGO”

Submitted in partial fulfillment of the degree of Bachelor of Technology
Rajasthan Technical University



Submitted to:
Mr. Amit Bohra

Submitted by:
Vikash Gupta
(17egjcs178)

DEPARTMENT OF COMPUTER ENGINEERING
GLOBAL INSTITUTE OF TECHNOLOGY, JAIPUR



CERTIFICATE OF COMPLETION

Certificate No: 5304

This is to certify that Vikas Gupta
of GIT, Jaipur
successfully completed Practical Training
on Python + Django
During 25th May 2019 to 6th July 2019
All the best.



 **SUN COMPUTECH**



CHANNEL PARTNER OF



 **Fuji Electric**

 **YASKAWA**



For Certificate Authenticity please contact us at CIC@SeldomIndia.com

Jaipur :
27, Kailash Puri, Near Khandaka Hospital, Tonk Road,
Jaipur, Rajasthan-302018 Phone : +91-9413 240 301


Delhi :
K-108/109, Street #2, Mangal Bazar, Laxmi Nagar, Delhi-110091
Phone : +91-11-22015681, 22425681, 9911335681

 www.seldomindia.com

 info.seldomindia@google.com

 [seldomindia](https://www.facebook.com/seldomindia)

 cic@seldomindia.com

 +91 9413 240 301

 @SeldomIndia

DECLARATION

I hereby declare that the seminar report entitled “**PYTHON WITH DJANGO**” was carried out and written by me under the guidance of **Mr. Sahil Middha**, the founder of Seldom India. This work has not been previously formed the basis for the award of any degree or diploma or certificate nor has been submitted elsewhere for the award of any degree or diploma.

Place: Jaipur

Student Name : Vikash Gupta

Date: 29-09-2019

Roll Number : 17EGJCS178

ACKNOWLEDGMENT

I would like to take this opportunity to show my gratitude towards **Mr. Sahil middha** who helped me in successful completion of my Second Year Practical Training. They have guided, motivated & were source of inspiration for me to carry out the necessary proceedings for the training to be completed successfully.

I am also grateful to the **Mr. Sahil Middha** for his/her guidance and support.

I am thankful to **Mr. Sahil Middha** for his/her kind support and providing me expertise of the domain to develop the project.

I would also like to express my hearts felt appreciation to all of my friends whose direct or indirect suggestions help me to develop this project [and to entire team members for their valuable suggestions.

Lastly, thanks to all faculty members of Computer Engineering department for their moral support and guidance.

Submitted By:

Vikash Gupta

ABSTRACT

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Keywords:

Restful API, HTML, CSS, Django

Implementation Software:

VS Studio Code

Table of contents

S. No.	Title	Page no.
Chapter 1	Introduction	8-12
	1.1 Specification of summer training	8
	1.2 Company specification	8
	1.3 Features	9
	1.4 Live project	12
	1.5 Training details	12
Chapter 2	Technology Specification	13-30
	2.1 Python	13
	2.2 Use of python	14
	2.3 Python future	15
	2.4 Python history	15
	2.5 Python versions	16
	2.6 API	19
	2.7 Need of API	19
	2.8 Type of API	19
	2.9 Advantage of API	21
	2.10 Disadvantage of API	22
	2.11 JSON	22
	2.12 DJANGO	25
	2.13 Django versions	27
	2.14 Django features	29
Chapter 3	Project Description	30-45
	3.1 Objective	30
	3.2 Description	30
	3.3 Technology used in website	30
	3.4 Snapshots	31-43

Chapter 4	Conclusion and limitations of project	46
Chapter 5	Frequently asked questions	47-48
Chapter 6	References	49

List of tables

Table No	Title	Page No.
1	Python Versions	17
2	Django versions	27

List of figures

Figure No	Title	Page No
2.1	Django Flow Diagram	29
3.1	UML diagram	33
3.2	Registration page	34
3.3	Login page	35
3.4	Add product	36
3.5	Dashboard page	37
3.6	Contact page	38
3.7	Log in page for Admin	39
3.8	Admin page	40

3.9	Admin page source code	41
3.10	App.urls	42
3.11	Urls.py	43
3.12	Views.py	44
3.13	Views.py	44
3.14	Views.py	45
3. 15	Models.py	46

Introduction

Company Synopsis

Technology service provider and training organization . The soul mission of founders is to facilitate the education, research and development program; all under one roof. In a very short span of time our team has successfully delivered the impactful service to more than 350 colleges, including the most prestigious institutions of India, such as IIT Mumbai, IIT Delhi and all the NITs. All over the India, with our 6 centers, we are renowned for our own manufacturing unit and unique content. Seldom India is moving ahead with an ideology where practical and theory are equally emphasized. In the vast growing ‘Technical Era’ we are rising with a mission to expand the set boundaries of the ‘techie-brains’ to Explore, Invent and Innovate!

Seldom India is one of the well-known IT organizations in Jaipur. We deliver our services in various sectors like web development, software development, digital marketing, App development, and Professional training and certification. We focus on providing the positive result with our quality work following the tight deadlines either it is development, digital promotion, or training & certification.

We offer job oriented Web designing, Web Development, Android , Digital Marketing course and live project job oriented training for B.Tech and Other Graduates. which is consistent and focused on Practical’s? Our efforts go into providing best industry oriented knowledge focusing on skill enhancement and better practical approach. We pride ourselves in giving full cooperation and best possible placement service for a seeker.

Our Trainers teach you from scratch to advance. Infonic also have its own development center. Actually we will not teach you, we will provide you a guide that how to work in corporate. Our training includes free Interview preparation and personality development classes which will polish both of your interpersonal and technical skills

The candidates who are looking for training and development can search us through industrial training companies in Jaipur, best industrial training companies in Noida, can contact to our Counsellors. Our only one and major focus is to prepare & lift the IT service in its respective position.

By profession we are an IT company, since we provide training also so you can consider us as a Computer Training Institute in Jaipur, Delhi. We have a separate department of training where well educated and industry expert trainers are always available to support the students. The major part of Industrial training is live or practical training, in which students learn how to work in the industry using the advanced technologies.

Feature

Certified Organization

Technology service provider and training organization .The soul mission of founders is to facilitate the education, research and development program; all under one roof.

350+ Colleges

Our team has successfully delivered the impactful service to more than 350 colleges, including the most prestigious institutions of India, such as IIT Mumbai, IIT Delhi and all the NITs.

Work Ideology

SELDOM INDIA is moving ahead with an ideology where practical and theory are equally emphasized.

Career At Seldom India

When you join Seldom I, you don't take up a job , you align yourself to an exciting way of life –a work life that will see you discovering your true mettle.

100% Practical Training

We are the team of designers and developers and we work on live projects , that's why we understand the importance of practical knowledge. Our teaching method is slightly differ from other web designing institutes of Jaipur. We focus on practical's and creativity.

Our professional web designer trainers designed our curriculum in such a way that student can learn things practically. We provide theory sessions as well as practical session to make students ready for job.

During theory session we provide students depth knowledge of technology with important questions regarding interview and seminars and during practical we sharp their coding fluency and make them perfect in coding and creativity.

Infonic focuses more on practical training with sufficient theory background to ensure that a trainee

understands what he is implementing. Practical training is absolutely mandatory and necessary for any trainee who is looking for a stable job. Practical training enables trainee to search and execute

operations which are complicated and requires many basic fundamentals to be used together. Practical training is really boon for a trainee which trains him for doing jobs which comes handy in future in his career.

Our aim is to provide practical training to help students to develop skills and abilities that support professional studies and prepare them for work later on. Another benefit of practical training is that trainee gets to know latest technology and employers' value, benefits from the new technologies and more efficient practices.

Live Project

Live project-based training under the guidance of professional trainers is available. We at infonic provides live project training for B.Tech and MCA students as summer internship. Live Project training gives students practical exposure of industry and it is required to gain practical knowledge

During every course, student have to make a live project based on client's demand, Then We learn every step of Data flow and SDLC that how a project starts and finishes

Involving students in Live project training helps to make them perfect in coding as well as when they work on live project they feel confidence.

Training Details

- Training Duration : 45 Days
- Training Period : 25 May 2019 to 6 July 2019
- Daily Contribution Hours : 2 Hours (11:30 A.M to 1:30P.M)
- Training Location : GIT JAIPUR.
- Training Coordinator : Mr. Sahil Middha

Chapter 2

Technology Specification & Language Learned

2.1. Python

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was

a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer **Guido van Rossum** shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council.

2.2. Use of Python

Python is used by hundreds of thousands of programmers and is used in many places. Sometimes

Only Python code is used for a program, but most of the time it's used to do simple jobs while another programming language is used to do more complicated tasks.

Its [standard library](#) is made up of many [functions](#) that come with Python when it is installed. On the [Internet](#) there are many other [libraries](#) available that make it possible for the Python language to do more things. These libraries make it a powerful language; it can do many different things.

Some things that Python is often used for are:

- Web development
- Scientific programming
- Desktop [GUIs](#)
- [Game](#) programming.
- Network programming
- Software Development
- Scientific and Numeric
- Business Applications
- Console based Application
- Audio or Video based Applications
- 3D CAD Applications

2.3. Python Features

- Python is the most popular language due to the fact that it's easier to code and understand it.
- Python is object-oriented programming language and can be used to write functional code too.

- It is a suitable language that bridges the gaps between business and developers.
- Subsequently, it takes less time to bring a Python program to market compared to other languages such as C#/Java.
- Additionally, there are a large number of python machine learning and analytical packages.
- A large number of communities and books are available to support Python developers.
- Nearly all type of applications, ranging from forecasting analytical to UI, can be implemented in Python.

2.4. Python History

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI in Netherland.
- In February 1991, van Rossum published the code (labeled version 0.9.0) to outsources.
- In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
- Python 2.0 added new features like: list comprehensions, garbage collection system.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.

- *ABC programming language* is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.

2.5. Python Version

- Python programming language is being updated regularly with new features and supports. There are lots of updates in python versions, started from 1994 to current release.

A list of python versions with its released date is given below.

Python Version	Released Date
Python 1.0	January 1994
Python 1.5	December 31, 1997
Python 1.6	September 5, 2000
Python 2.0	October 16, 2000
Python 2.1	April 17, 2001
Python 2.2	December 21, 2001
Python 2.3	July 29, 2003

Python 2.4	November 30, 2004
Python 2.5	September 19, 2006
Python 2.6	October 1, 2008
Python 2.7	July 3, 2010
Python 3.0	December 3, 2008
Python 3.1	June 27, 2009
Python 3.2	February 20, 2011
Python 3.3	September 29, 2012
Python 3.4	March 16, 2014
Python 3.5	September 13, 2015
Python 3.6	December 23, 2016
Python 3.7	June 27, 2018

Table: 2.1 Python versions

2.6. Application Programming Interface (API)

API is an abbreviation for Application Programming Interface which is a collection of communication protocols and subroutines used by various programs to communicate between them. A programmer can make use of various API tools to make its program easier and simpler. Also, an API facilitates the programmers with an efficient way to develop their software programs.

Thus, in simpler terms, an API helps two programs or applications to communicate with each other by providing them with necessary tools and functions. It takes the request from the user and sends it to the service provider and then again sends the result generated from the service provider to the desired user.

A developer extensively uses API's in his software to implement various features by using an API call without writing the complex codes for the same. We can create an API for an operating system, database systems, hardware system, for a JavaScript file or similar object-oriented files. Also, an API is similar to a GUI (Graphical User Interface) with one major difference. Unlike GUI's, an API helps the software developers to access the web tools while a GUI helps to make a program easier to understand by the users.

An application program interface (API) is a set of routines, protocols, and tools for building software applications. Basically, an API specifies how software components should interact. Additionally, APIs are used when programming graphical user interface (GUI) components.

2.7. Need of API

Imagine the following scenario: You (as in, your application, or your client, this could be a web browser) wants to access another app's data or functionality. For example, perhaps you want to access all Twitter tweets that mention the *#episodes* hash tag.

You could email Twitter and ask for a spreadsheet of all these tweets. But then you'd have to find a way to import that spreadsheet into your application; and, even if you stored them in a database, as we have been, the data would become outdated *very* quickly. It would be impossible to keep it up to date.

It would be better and simpler for Twitter to provide you a way to query their application to get that data, so that you can view or use it in your own application. It would stay up to date automatically that way.

An API broker access to a different application to provide functionality or access to data, so data can be included in different applications.

2.8. Types of API

There are three basic forms of API: -

WEB APIs:

A Web API also called as Web Services is an extensively used API over the web and can be easily accessed using the HTTP protocols. A Web API is an open source interface and can be used by a large number of clients through their phones, tablets. or PC's.

LOCAL APIs:

In this type of API, the programmers get the local middleware services. TAPI (Telephony Application Programming Interface), .NET are common examples of Local API's.

PROGRAM APIs:

It makes a remote program appears to be local by making use of RPC's (Remote Procedural Calls). SOAP is a well-known example of this type of API.

REST API (Introduction)

Representational State Transfer (REST) is an architectural style that defines a set of constraints to be used for creating web services. REST API is a way of accessing the web services in a simple and flexible way without having any processing.

REST technology is generally preferred to the more robust Simple Object Access Protocol (SOAP) technology because REST uses the less bandwidth, simple and flexible making it more suitable for internet usage. It's used to fetch or give some information from a web service. All communication done via REST API used only HTTP request.

In **HTTP** there are five methods which are commonly used in a REST based Architecture i.e., POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations respectively. There are other methods which are less frequently used like OPTIONS and HEAD.

- **GET:** The HTTP GET method is used to read (or retrieve) a representation of a resource. In the safe path, GET returns a representation in XML or JSON and an HTTP response code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400 (BAD REQUEST).
- **POST:** The POST verb is most-often utilized to create new resources. In particular, it's used to create subordinate resources. That is, subordinate to some other (e.g. parent)
- **NOTE:** POST is neither safe nor idempotent.
- **PUT:** It is used for updating the capabilities. However, PUT can also be used to create a resource in the case where the resource ID is chosen by the client instead of by the server. In other words, if the PUT is to a URI that contains the value of a non-existent resource ID. On successful update, return 200 (or 204 if not returning any

content in the body) from a PUT. If using PUT for create, return HTTP status 201 on successful creation. PUT is not safe operation but it's idempotent.

- **PATCH:** It is used for modify capabilities. The PATCH request only needs to contain the changes to the resource, not the complete resource. This resembles PUT, but the body contains a set of instructions describing how a resource currently residing on the server should be modified to produce a new version. This means that the PATCH body should not just be a modified part of the resource, but in some kind of patch language like JSON Patch or XML Patch. PATCH is neither safe nor idempotent.
- **DELETE:** It is used to delete a resource identified by a URI. On successful deletion, return HTTP status 200 (OK) along with a response body.

2.9. Advantage of API

- **Efficiency:** API produces efficient, quicker and more reliable results than the outputs produced by human beings in an organization.
- **Flexible delivery of services:** API provides fast and flexible delivery of services according to developer's requirements.
- **Integration:** The best feature of API is that it allows movement of data between various sites and thus enhances integrated user experience.
- **Automation:** As API makes use of robotic computers rather than humans, it produces better and automated results.
- **New functionality:** While using API the developers find new tools and functionality for API exchanges.

2.10. Disadvantage of API

- **Cost:** Developing and implementing API is costly at times and requires high maintenance and support from developers.
- **Security issues:** Using API adds another layer of surface which is then prone to attacks, and hence the security risk problem is common in API's.

2.12. JSON

JSON: **J**ava**S**cript **O**bject **N**otation.

JSON is syntax for storing and exchanging data.

JSON is text, written with JavaScript object notation.

Exchanging Data

When exchanging data between a browser and a server, the data can only be text. JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server. We can also convert any JSON received from the server into JavaScript objects. This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

Sending Data

If you have data stored in a JavaScript object, you can convert the object into JSON, and send it to a server:

```
var myObj = {name: "John", age: 31, city: "New York"};
var meson = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

Receiving Data

If you receive data in JSON format, you can convert it into a JavaScript object:

```
var myJSON      = '{"name":"John",      "age":31,      "city":"New      York"}';  
var myObj      = JSON.parse(myJSON);  
document.getElementById("demo").innerHTML = myObj.name;
```

What is JSON?

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight data-interchange format
- JSON is "self-describing" and easy to understand
- JSON is language independent

JSON uses JavaScript syntax, but the JSON format is text only. Text can be read and used as a data format by any programming language.

Why use JSON?

Since the JSON format is text only, it can easily be sent to and from a server, and used as a data format by any programming language.

JavaScript has a built-in function to convert a string, written in JSON format, into native JavaScript objects:

JSON.parse ()

So, if you receive data from a server, in JSON format, you can use it like any other JavaScript object.

JSON Syntax Rules

JSON syntax is derived from JavaScript object notation syntax:

- Data is separated by commas
- Curly braces hold objects
- Square Data is in name/value pairs
- brackets hold arrays

JSON - Evaluates to JavaScript Objects

The JSON format is almost identical to JavaScript objects.

In JSON, *keys* must be strings, written with double quotes:

```
{ "name":"John" }
```

JSON is like XML Because

- Both JSON and XML are "self describing" (human readable)
- Both JSON and XML are hierarchical (values within values)
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an XMLHttpRequest

JSON is unlike XML Because

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write

Data types and syntax

JSON's basic data types are:

- **Number:** a signed decimal number that may contain a fractional part and may use exponential E notation, but cannot include non-numbers such as NaN. The format makes no distinction between integer and floating-point. JavaScript uses a double-precision floating-point format for all its numeric values, but other languages implementing JSON may encode numbers differently.
- **String:** a sequence of zero or more Unicode characters. Strings are delimited with double-quotation marks and support a backslash escaping syntax.
- **Boolean:** either of the values true or false
- **Array:** an ordered list of zero or more values, each of which may be of any type. Arrays use square bracket notation with comma-separated elements.
- **Object:** an unordered collection of name–value pairs where the names (also called keys) are strings. Since objects are intended to represent associative arrays, it is recommended, though not required, that each key is unique within an object. Objects are delimited with curly brackets and use commas to separate each pair, while within each pair the colon ':' character separates the key or name from its value.
- **null:** An empty value, using the word null

Whitespace is allowed and ignored around or between syntactic elements (values and punctuation, but not within a string value). Four specific characters are considered whitespace for this purpose: space, horizontal tab, line feed, and carriage return. In particular, the byte order mark must not be generated by a conforming implementation (though it may be accepted when parsing JSON). JSON does not provide syntax for comments.

Early versions of JSON (such as specified by RFC 4627) required that a valid JSON text must consist of only an object or an array type, which could contain other types within them.

2.12. Django

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support. Django is a web application framework written in Python programming language. It is based on MVT (Model View Template) design pattern. The Django is very demanding due to its rapid development feature. It takes less time to build application after collecting client requirement.

Django is a free and open source web application framework written in Python. A framework is nothing more than a collection of modules that make development easier. They are grouped together, and allow you to create applications or websites from an existing source, instead of from scratch. This is how websites - even simple ones designed by a single person - can still include advanced functionality like authentication support, management and admin panels, contact forms, comment boxes, file upload support, and more. In other words, if you were creating a website from scratch you would need to develop these components yourself. By using a framework instead, these components are already built, you just need to configure them properly to match your site.

The official project site describes Django as "a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. "Django offers a big collection of modules which you can use in your own projects. Primarily, frameworks exist to save developers a lot of wasted time and headaches and Django is no different. You might also be interested in learning that Django was created with front-end developers in mind. "Django's template language is designed to feel comfortable and easy-to-learn to those used to working with HTML, like designers and front-end developers. But it is also flexible and highly extensible, allowing developers to augment the Template

language as needed."

If you're going to be working with Python, especially for web applications or web design, you'll want to remember the Django framework. It will certainly come in handy.

This framework uses a famous tag line: The web framework for perfectionists with deadlines.

By using Django, we can build web applications in very less time. Django is designed in such a manner that it handles much of configure things automatically, so we can focus on application development only.

Django Database Migrations

Migration is a way of applying changes that we have made to a model, into the database schema. Django creates a migration file inside the **migration** folder for each model to create the table schema, and each table is mapped to the model of which migration is created.

Django provides the various commands that are used to perform migration related tasks. After creating a model, we can use these commands.

Make migrations: It is used to create a migration file that contains code for the tabled schema of a model.

Migrate: It creates table according to the schema defined in the migration file.

Saltgrade: It is used to show a raw SQL query of the applied migration.

Show migrations: It lists out all the migrations and their status.

2.13. VERSIONS

Release Series	Latest Release	End of mainstream support ¹	End of extended support ²
2.2 LTS	2.2.3	December 2019	April 2022
2.1	2.1.10	April 1, 2019	December 2019
2.0	2.0.13	August 1, 2018	April 1, 2019
1.11 LTS	1.11.22	December 2, 2017	April 2020
1.10	1.10.8	April 4, 2017	December 2, 2017
1.9	1.9.13	August 1, 2016	April 4, 2017
1.8 LTS	1.8.19	December 1, 2015	April 1, 2018
1.7	1.7.11	April 1, 2015	December 1, 2015
1.6	1.6.11	September 2, 2014	April 1, 2015
1.5	1.5.12	November 6, 2013	September 2, 2014
1.4 LTS	1.4.22	February 26, 2013	October 1, 2015
1.3	1.3.7	March 23, 2012	February 26, 2013

Table: 2.2 Django versions

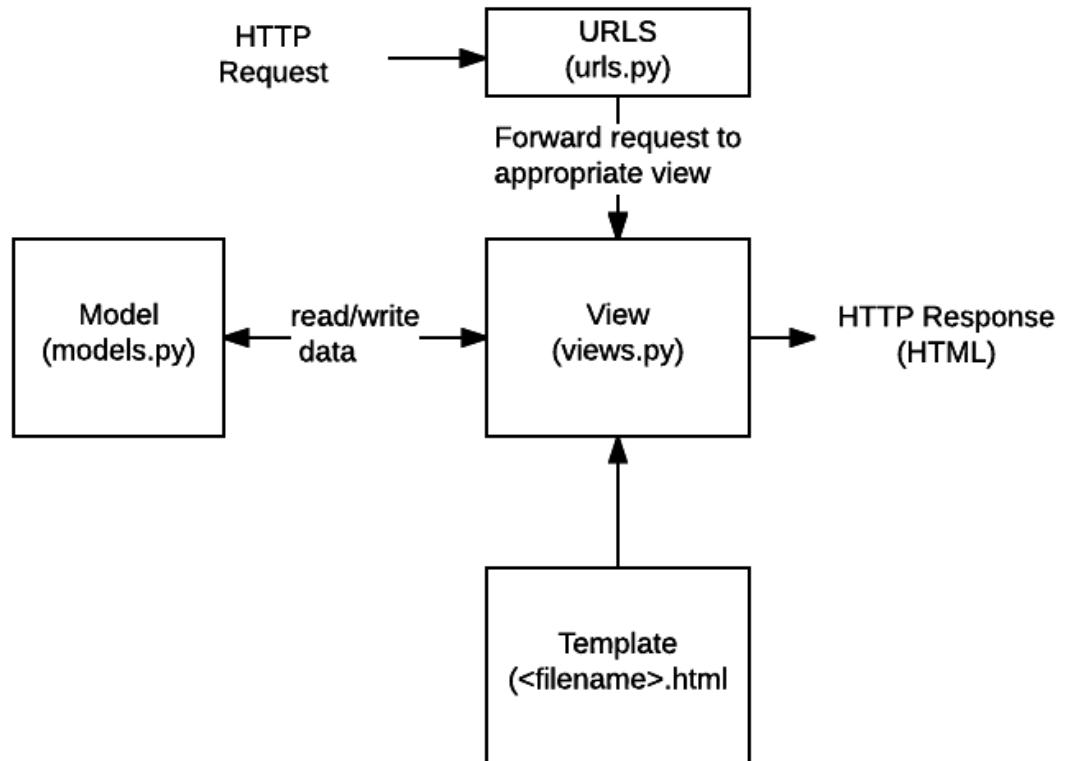


Figure: 2.1 Flow diagram of Django frame work

2.14. Features

Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box".

Versatile

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format.

Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically.

Scalable

Django uses a component-based architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers.

Portable

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavors of Linux, Windows, and Mac OS X. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

Maintainable

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code.

PROJECT DESCRIPTION

3.1. OBJECTIVE

Online shopping is the process of buying goods and services from seller who sell on the internet and people can purchase just about anything from companies that provide their products online.

3.2. Description

The Web Portal open up with Home Page after the which is redirected to login screen which provides a user to get access to the use of Portal. There is also a provision for registration. Here you can Add and Delete item as your requirement.

3.3. Technology used in the web site

- VS Studio Code is using an IDE that Provide Many Extensions. In this Python, java HTML, CSS, Java Script and other language code can be written. Vs Studio provide Terminal Also for Compile the code.
- The received data was parsed using JSON Array object in java and afterwards populated according to the use of the up and user requirements.

Django is a Web Framework that a used makes a dynamic website. In this Models, Views, URLs, Setting Files are used. IT current version is 2.2

UML diagram of online shopping system

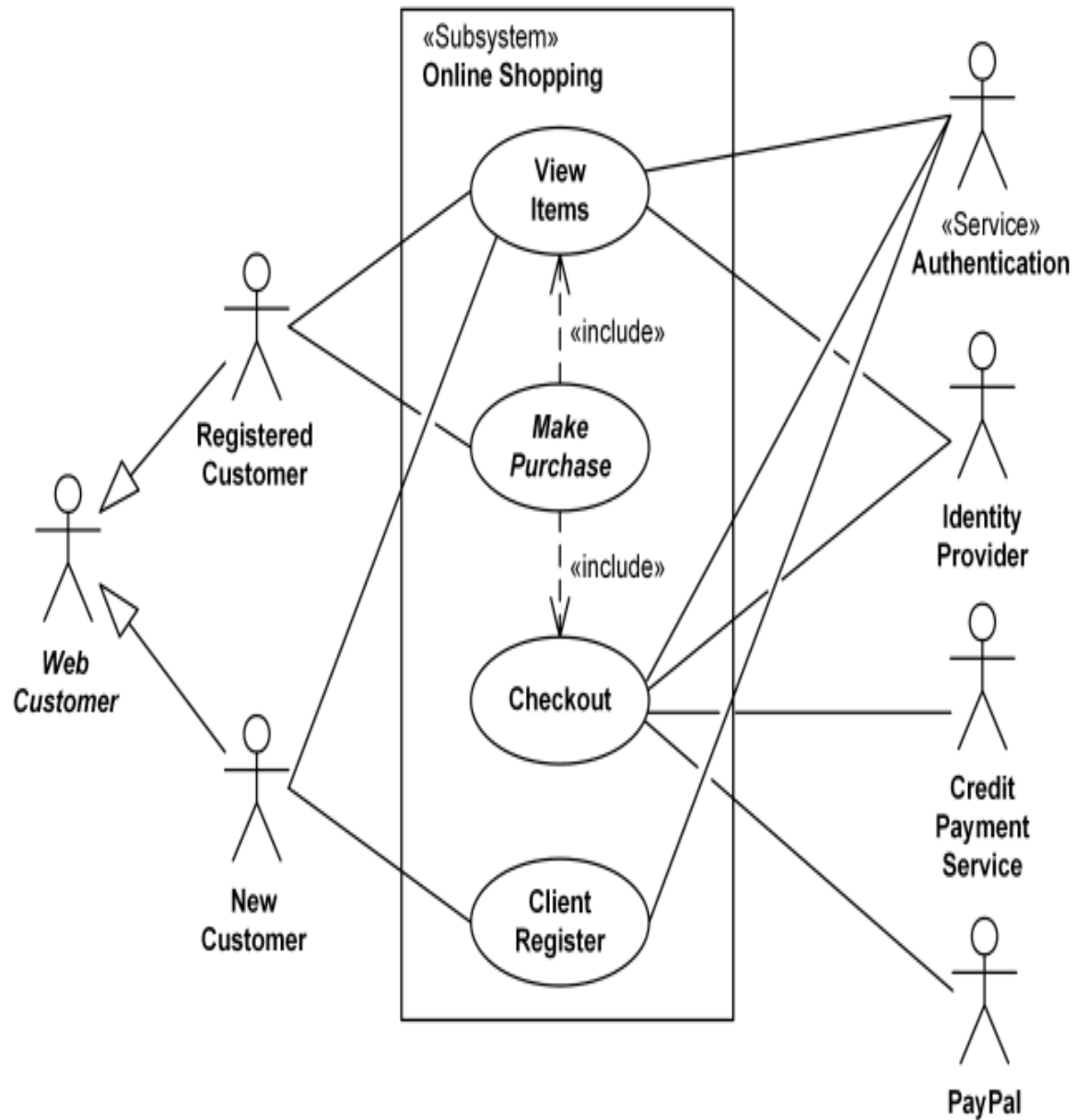


Fig: 3.1 UML diagram

3.4. SNAPSHOT

Registration Form

Name
Enter Your Name

Email
Enter Your Email

Password
Enter Your Password

Phone
Enter Your Phone Number

Shop_Name
Enter Your Shop Name

Product_Type
Enter Your Product Type

Submit

Figure: 3.2 Registration page

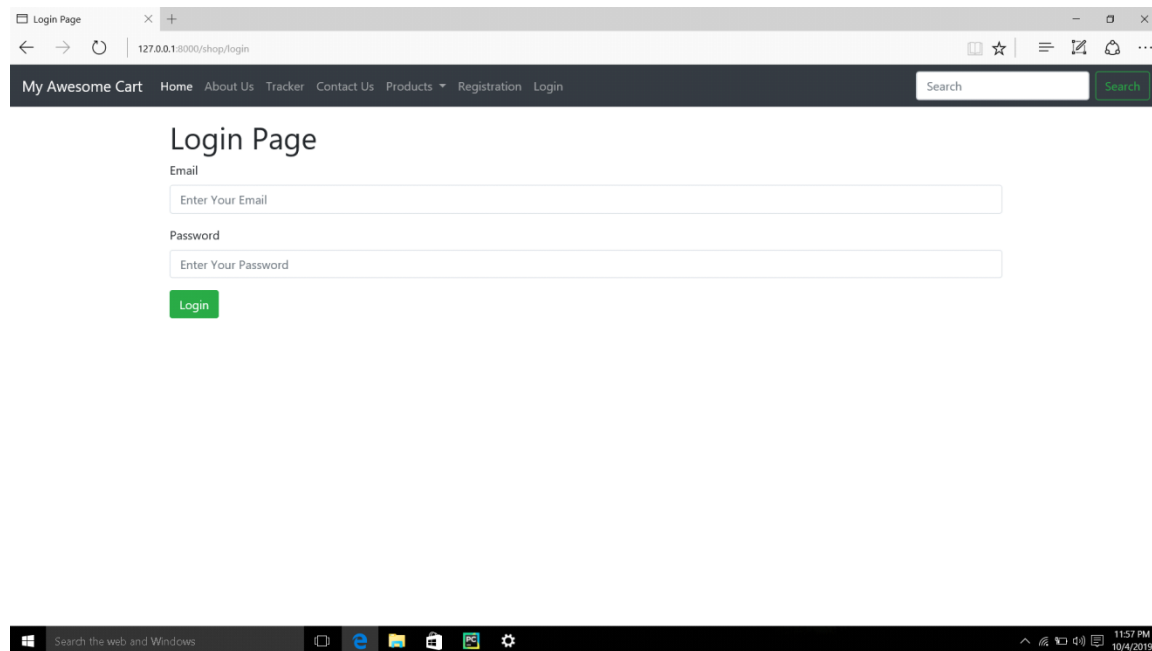


Figure 3.3 Login Page

The screenshot displays the 'Add product' form in the Django administration interface. The form includes the following fields and controls:

- Product name:** A text input field.
- Category:** A text input field.
- Subcat:** A text input field.
- Price:** A numeric input field with a value of 0 and a spinner control.
- Desc:** A text input field.
- Prod date:** A date input field set to 'Today' with a calendar icon. A note below states: 'Note: You are 5.5 hours ahead of server time.'
- Image:** A file upload field with a 'Browse...' button and the text 'No file selected.'

At the bottom right of the form, there are three buttons: 'Save and add another', 'Save and continue editing', and 'SAVE'.

Figure: 3.4 Add products

Figure 3.1 is the registration page for seller and after the reg. he can log in and log out fig 3.2.

This is the login page fig: 3.2, if the user wants to access the site then he has to login first. The email-id should be authenticated and the password should be at least of 8 letters which includes one capital letter, one small letter one digit and one special symbol.

If a user does not want to login again and again if he is accessing the on the regular basis then for this, he can store his username and password through remember me.

Figure 3.3 shows that here seller add any product category wise and seller also set all the detail of product and cost of product.

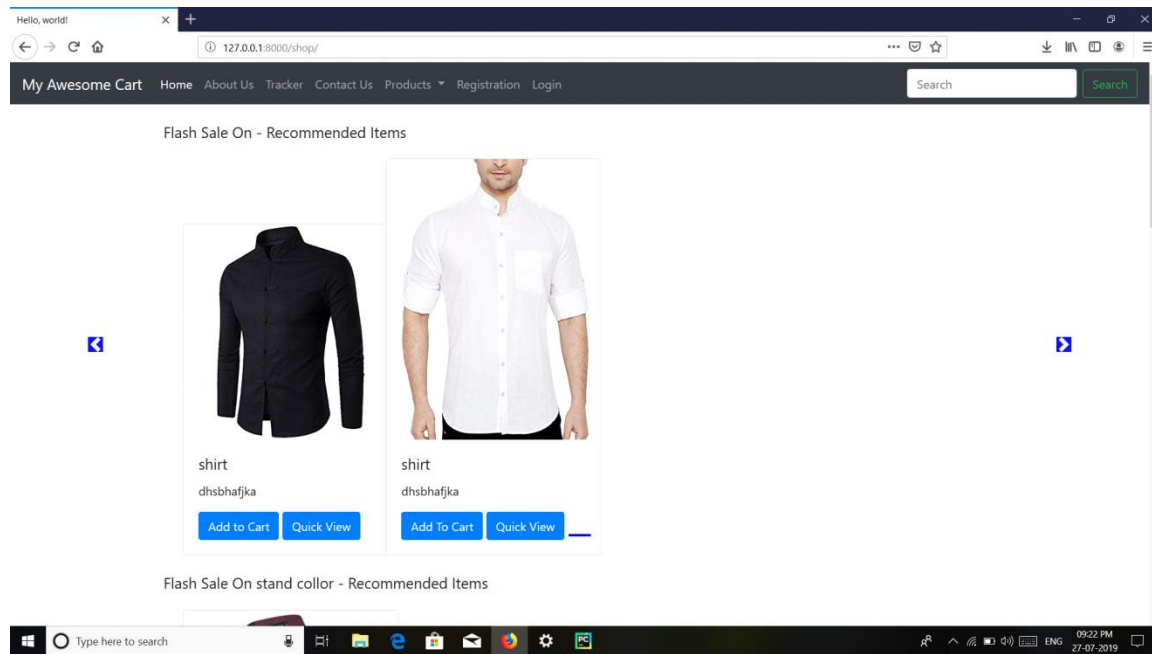


Figure: 3.5 Dashboard Page

The screenshot shows a web browser window with a single tab titled 'Contact'. The address bar displays '127.0.0.1:8000/shop/contact/'. The website's navigation bar includes links for 'Home', 'About Us', 'Tracker', 'Contact Us', 'Products', 'Registration', and 'Login', along with a search bar. The main content area is titled 'Contact Us' and contains a form with the following fields: 'Name' (with placeholder 'Enter Your Name'), 'Email' (with placeholder 'Enter Your Email'), 'Phone' (with placeholder 'Enter Your Phone Number'), and a larger text area for 'How May We Help You?'. A green 'Submit' button is located at the bottom of the form. The Windows taskbar at the bottom shows the search bar and several application icons, with the system clock indicating 09:28 PM on 27-07-2019.

Figure: 3.6 contact page

This is the state page fig: 3.4. This is the Home page, it is view part for consumer where consumer bought the product and check out detail of product. Fig: 3.5 contact no. of seller. Consumer tracks the product status and finds the detail of product.

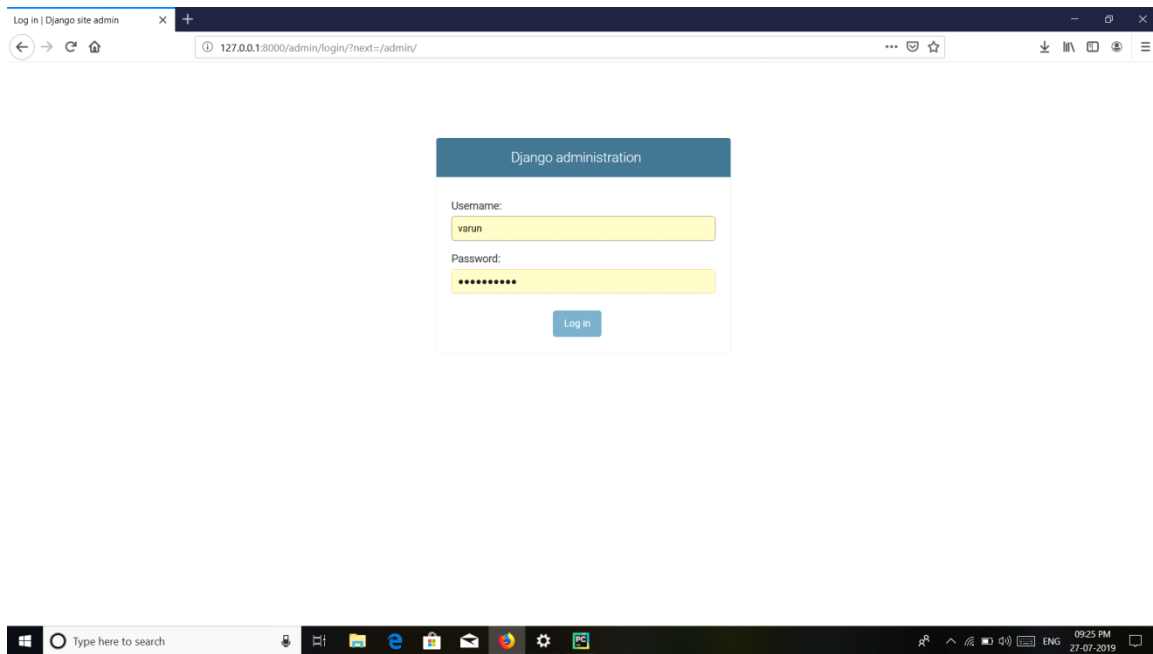


Figure: 3.7 login page for admin

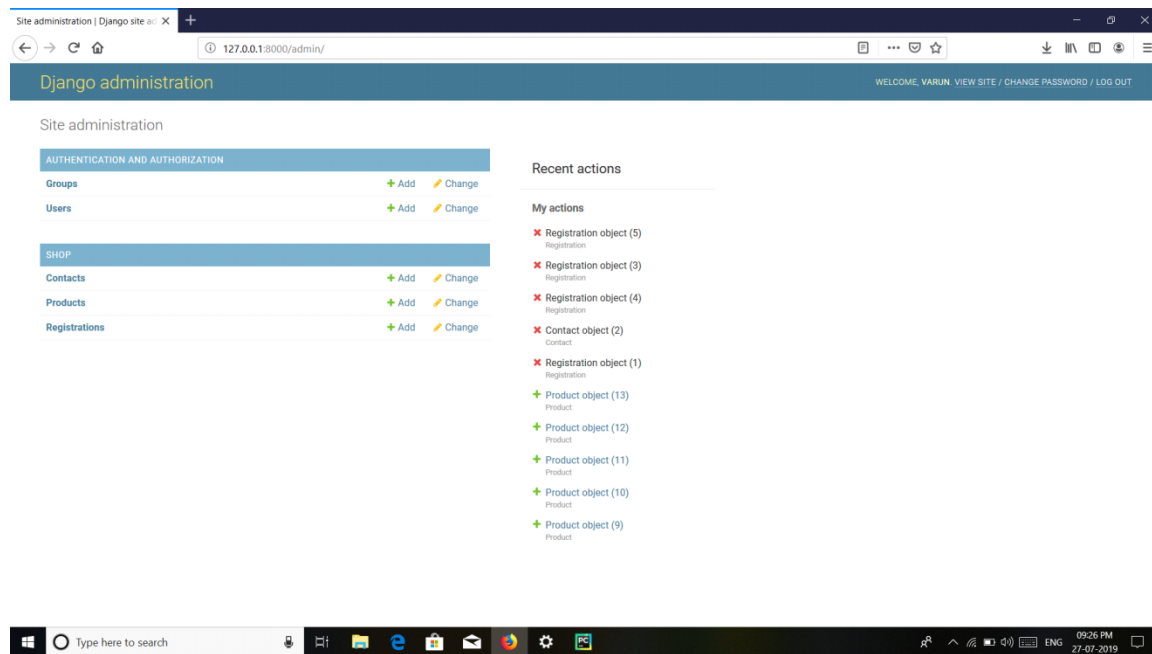


Figure: 3.8 Admin page

In fig: 3.7 the admin has many hooks for customization, but beware of trying to use those hooks exclusively. If you need to provide a more process-centric interface that abstracts away the implementation details of database tables and fields, then it's probably time to write your own views.

Source code of my project

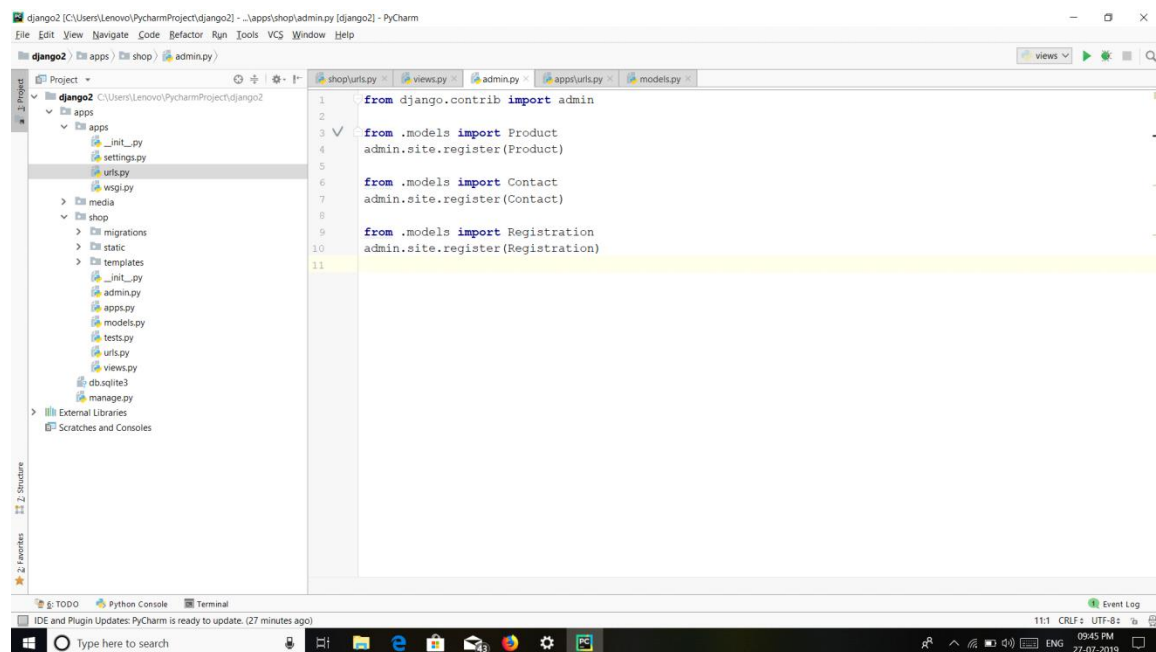


Figure: 3.9 Admin page

Figure: 3.8 show that admin can access the contact, reg. page and add product page.

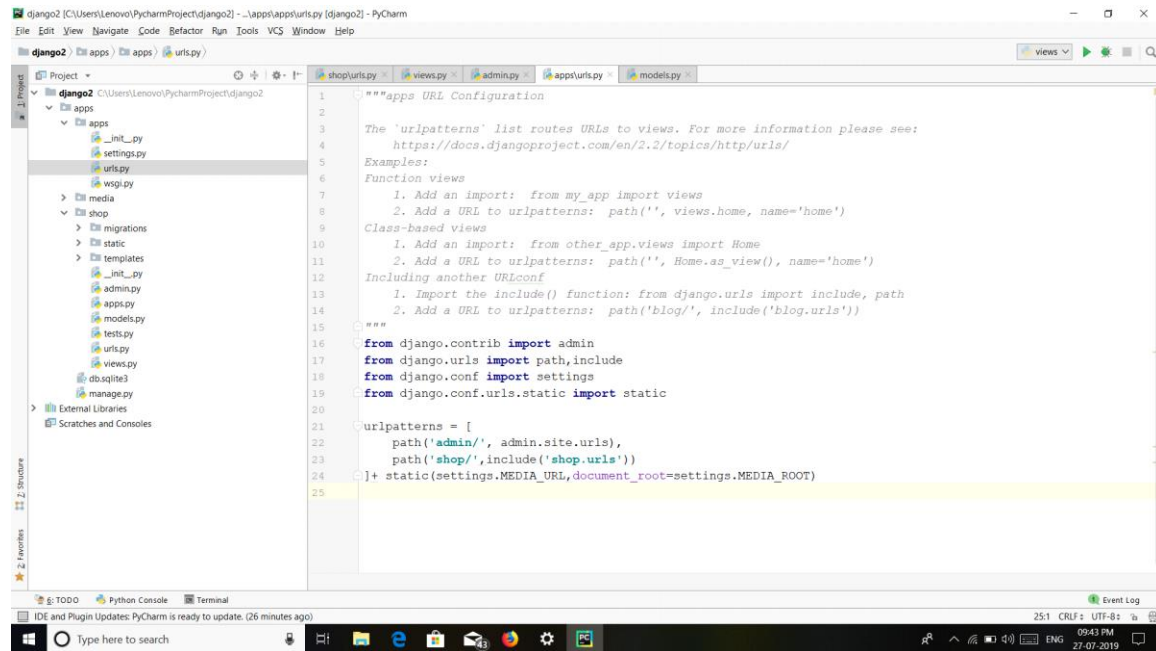


Figure: 3.10 App.urls

In Figure: 3.9 shows after running the source code for open admin page type admin in URL and for home page type shop.

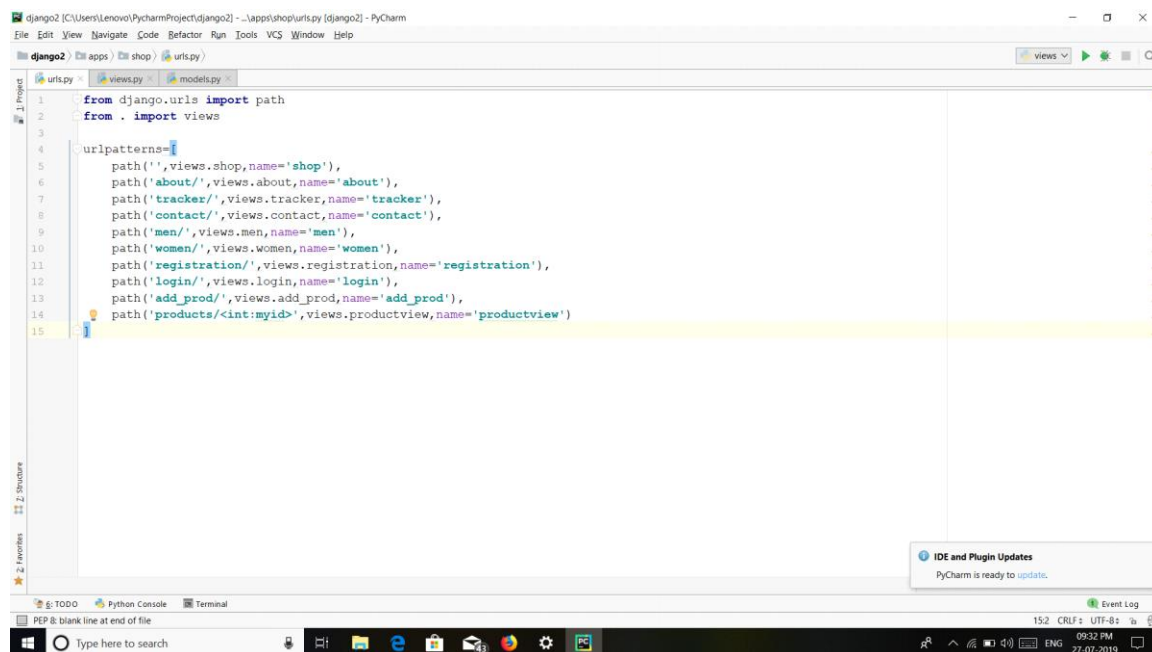
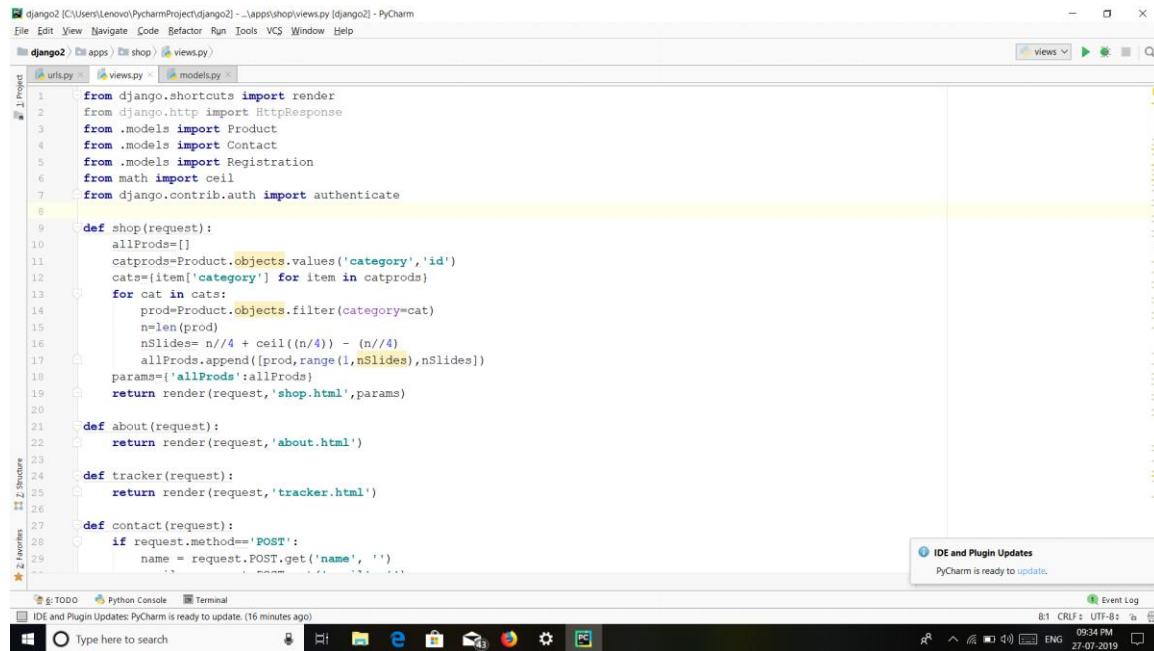


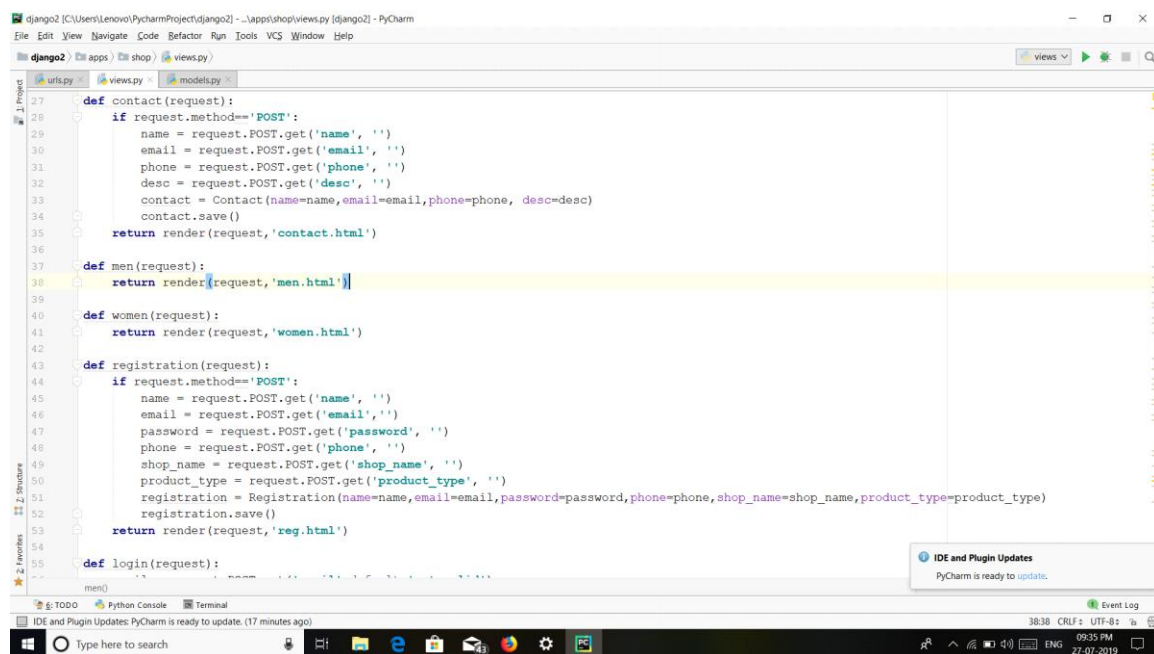
Figure: 3.11 urls.py

In figure: 3.10 shows Title bar of site where entry e.g. Home, contact us, about, tracker, category etc.



```
1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect
3 from .models import Product
4 from .models import Contact
5 from .models import Registration
6 from math import ceil
7 from django.contrib.auth import authenticate
8
9 def shop(request):
10     allProds=[]
11     catprods=Product.objects.values('category','id')
12     cats=(item['category'] for item in catprods)
13     for cat in cats:
14         prod=Product.objects.filter(category=cat)
15         n=len(prod)
16         nSlides= n//4 + ceil((n/4)) - (n//4)
17         allProds.append([prod,range(1,nSlides),nSlides])
18     params={'allProds':allProds}
19     return render(request,'shop.html',params)
20
21 def about(request):
22     return render(request,'about.html')
23
24 def tracker(request):
25     return render(request,'tracker.html')
26
27 def contact(request):
28     if request.method=='POST':
29         name = request.POST.get('name', '')
```

Figure: 3.12 views.py (1)



```
27 def contact(request):
28     if request.method=='POST':
29         name = request.POST.get('name', '')
30         email = request.POST.get('email', '')
31         phone = request.POST.get('phone', '')
32         desc = request.POST.get('desc', '')
33         contact = Contact(name=name,email=email,phone=phone, desc=desc)
34         contact.save()
35     return render(request,'contact.html')
36
37 def men(request):
38     return render(request,'men.html')
39
40 def women(request):
41     return render(request,'women.html')
42
43 def registration(request):
44     if request.method=='POST':
45         name = request.POST.get('name', '')
46         email = request.POST.get('email', '')
47         password = request.POST.get('password', '')
48         phone = request.POST.get('phone', '')
49         shop_name = request.POST.get('shop_name', '')
50         product_type = request.POST.get('product_type', '')
51         registration = Registration(name=name,email=email,password=password,phone=phone,shop_name=shop_name,product_type=product_type)
52         registration.save()
53     return render(request,'reg.html')
54
55 def login(request):
56     # ...
```

Figure: 3.13 views.py (2)

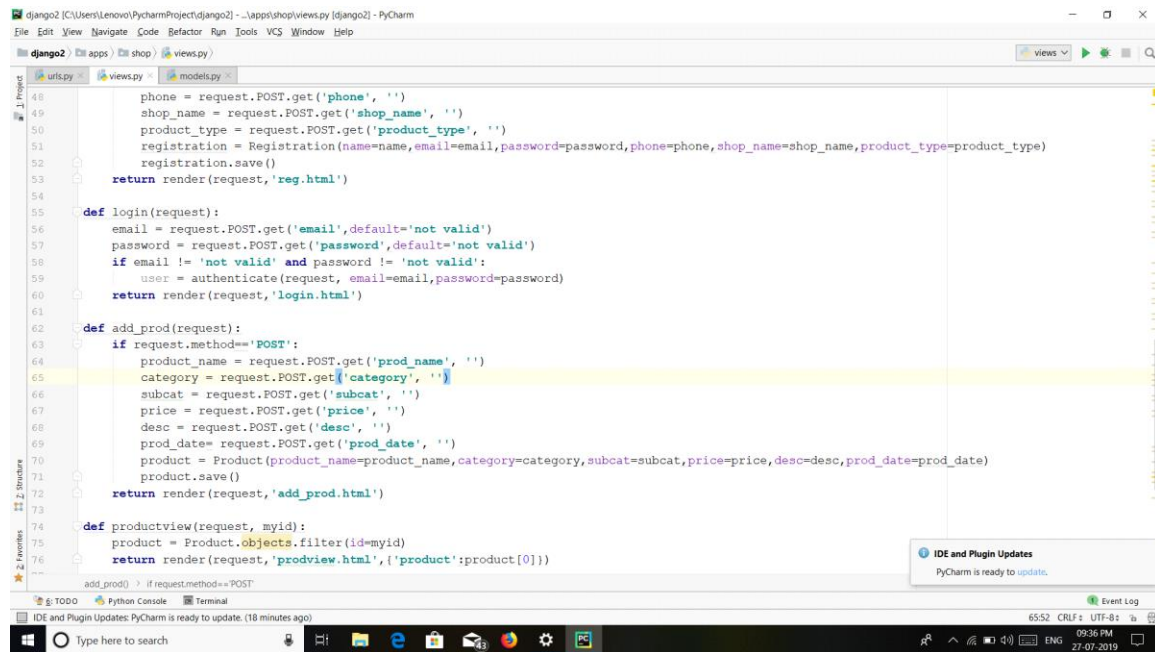


Figure: 3.14 views.py (3)

In fig :(3.11, 3.12, 3.13) code of title bar entry. e.g. in contact us what entry fill it is written in views.py and all the other title bar header entry code written here.

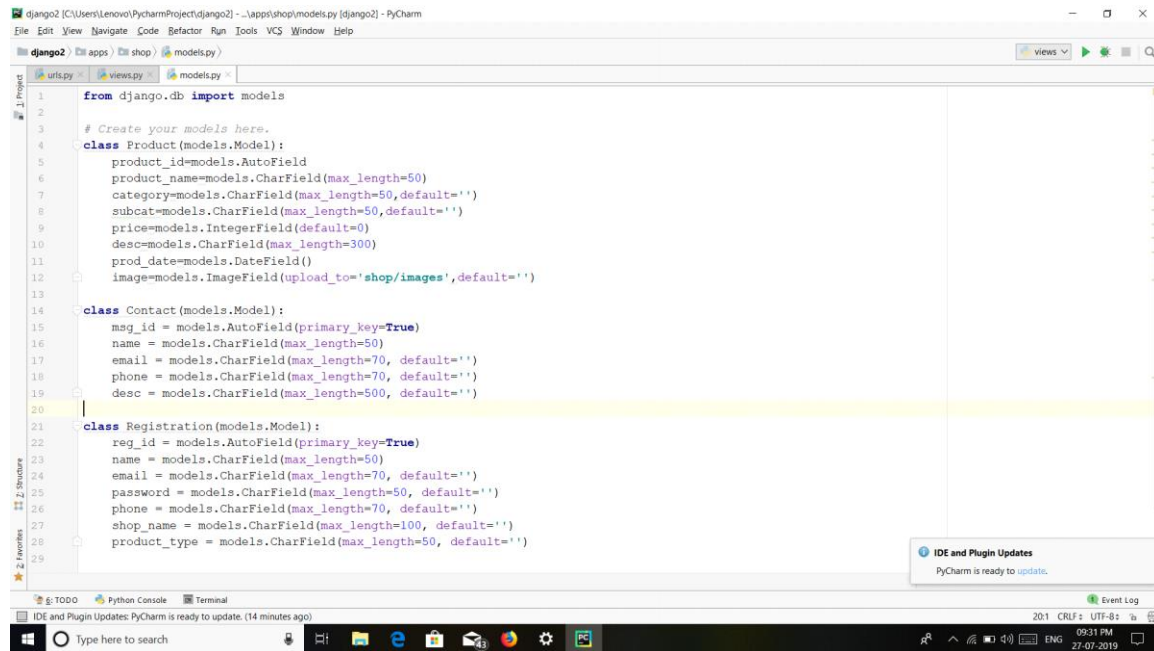


Figure: 3.15 models.py

Source code models.py (fig: 3.14) create a Database table that maintain the correct entry from view part.

CONCLUSION AND LIMITATION OF PROJECT

Conclusion:

The development of the project was a great experience of playing with logics and server data with efficiency. The errors handled were with of great use which loaded the developing skill with affluently and held with the completion of application with great success.

There are some limitations in this project and those limitations are:

- Internet is required to access the Web Portal.
- The user has to login to access the Web Portal

PROJECT AND FREQUENTLY ASKED QUESTIONS

Future Scope of Project:

- Today everything is done through internet, people are getting used to the internet and they don't even have so much of times to remember the Task. By using this Web portal, they can always remember the task.
- We can use this portal as a history of last days task that we done.
- Never worry about the things again
- To do list helps millions of people feel more in control of their lives.

Frequently Asked Questions:

Q1- Why has you chosen rest ape as your backend instead of SQLite or mongo-db or some other?

Ans- Restful API is a real-time engine with backward connectivity. I.e. you might build an app where clients subscribe to events on specific data and server actively informs clients about changes. Data layer is hosted for you. It's a nice kick start solution. Including authentication management.

Q2- What is REST API?

Ans- RSS (Rich Site Summary; originally RDF Site Summary; often called Really Simple Syndication) uses a family of standard web feed formats to publish frequently updated information: blog entries, news headlines, audio, video.

Q3- Can the application is accessed when the Internet connectivity is not there?

Ans- Obviously no, the news is fetched from the server of the respective newspapers directly and news can't be downloaded from the server until the Internet connectivity is there.

Q4 Are there any validations in password also?

Ans- Yes, there is a strict validation for password i.e. minimum 8 characters should be there, first character should be an alphabet and it should be in capital letters. At least one special symbol and one digit should be used in the password.

Q5- How the data is managed in the SQLite base?

Ans- The data is stored in a tree structure, not in a tabular form. There is a main parent in the structure and is followed by the children to maintain the hierarchy.

REFERENCES

- <https://developer.android.com/index.html>
- <https://dogriffiths.github.io/HeadFirstAndroid/>
- <http://www.tutorialspoint.com/android/>