This document describes the steps executed to complete the required task.

Required components:

Python:

Already Installed on Ubuntu machine.

Virtualenv:

Installed it via **pip** and created a virtual environment named **django-env.**

Django:

Activated virtual environment by

source ~/DjangoProject/django-env/bin/activate

Installed django v1.9 using pip in virtual environment.

Django-Project:

Django is used to create web apps. When a user clicks on a URL of Django app, It searches that URL in urls.py (in project directory). Once a URL is matched, it looks for associated view definition to be loaded in *views.py* (of a particualr app) and renders the response.

As per instructions of Django Tutorial, I did following steps.

1. Created a django project by running following command
   django-admin startproject mysite

This created a directory structure under *DjangoProject/* as *mysite/ .*

mysite/ included

1. File *manage.py* to run the project
2. Directory *mysite* which contained the project settings.

mysite/mysite/ contained

1. urls.py -> contains all the URLs used for this project. Syntax :

```
urlpatterns = [
url("URL to be called", "view function to be load", name="name of view"),...

            ]
```

2. settings.py -> database connection, registered apps etc
3. wsgi.py -> settings of server

Django App:

Once project is setup, we need to create an App to implement the required functionalities.

I created an app using

```
$ python manage.py startapp todos
```

It created a directory structure under mysite/mysite/. It had

1. *models.py* which contained database model TodoTask definitions for ***todos*** app
2. *app.py,* settings related to ***todos*** app
3. *tests.py,* unit tests regarding this app would go here
4. *urls.py,* contained URLs of ***todos*** app. We have following URLs defined
    a. GET /todos/list_all to list all the TODOs
    b. POST /todos/add_todo to create a TODO task
    c. POST /todos/resolve to resolve a TODO task

    This file is used as a link in project's *urls.py* using include keyword as

    ```
    url(r'^todos/', include(todos.urls')),
    ```
5. *views.py,* All the views definitions to be rendered when a specific URL is hit would go here. View can return data as string or can render some html file with specific data provided etc. Data provided for a specific html file must be passed in dict format and they keys of that dict will render the data at places as specified in that ***html*** file. These html files can go under *templates/* to have all templates at one place.
6. *admin.py*, Django provides an admin interface to communicate with database via interface. Through this interface, we can view all records of some particular table, delete some records, filter records as per requirement etc. To enable this interface we need to create admin user which I created as per instructions. We need to register a particular database model to view this in that interface. So, we need to register our models in this file as

    ```
    admin.site.register(TodoTask)
    ```

    After this Django admin interface included our model which had URL /admin/todos/todotask/ to view all tasks
7. Added *forms.py* to render model fields as input fields to add a task

Register App: Once app is setup, we need to add this in section *INSTALLED_APPS* under *settings.py.*

Database setup: Needed to create database models as per definitions in *models.py.* The steps executed to achieve this are

1. *python manage.py makemigrations* which created migration file to create databse models in specifed database*7*
2. *python manage.py migrate* to create required tables in database

Views.
We are using SQLite (default) database here.

Admin Interface: **Users** can be viewed/added and deleted from
/admin/auth/user/
Login Path: */admin/login/?next=/admin/*
Bootstrap: Bootstrap's latest version 3.3.6 used to make the view look nice
Jquery: *AJAX* calls added to

1. *create* a task
   When user clicks on **Create Task** button, we validate that the fields are not empty and make HTTP POST call to save this task in database
2. mark a task as *resolved*
   User can see a column **Resolved** with checkbox as input filed. When user clicks on checkbox, AJAX request is sent to update the is_checked field of that task.

References:

1. https://docs.djangoproject.com/en/1.9/intro
2. https://docs.djangoproject.com/en/1.9/ref/forms/widgets/
3. http://getbootstrap.com/javascript/#modals
4. http://www.w3schools.com/bootstrap/bootstrap_ref_js_modal.asp