



**Spoken Tutorial**  
IIT BOMBAY

# Spoken Tutorial Internship 2023 REPORT

On

**Django Framework**

Submitted by

**Etty Tiwari**

B.E Computer Engineering

Shree L.R. Tiwari College of Engineering

Mumbai University, Thane

Under the guidance of

**Prof. Kannan M. Moudgalya**

Chemical Engineering Department

IIT Bombay

Mentor

**Mr. Sunil Shetye**

**Ms. Ankita Singhal**

December 01, 2023

# **Acknowledgment**

## **Under the mentor Mr. Sunil Shetye & Ms. Ankita Singhal**

I would like to sincerely express my profound gratitude to all those individuals who played an instrumental role in ensuring the successful completion of my Internship at Spoken Tutorial, IIT Bombay. It is with immense appreciation that I take this opportunity to acknowledge their invaluable support and guidance.

First and foremost, I would like to extend my heartfelt thanks to Mr. Sunil Shetye and Ms. Ankita Singhal for their exceptional mentorship throughout my internship. Their patience, expertise, and dedication have been instrumental in shaping my growth and development during this period. I am truly grateful for their unwavering guidance and support.

I am truly honoured to have been selected as an intern at IIT Bombay, and I sincerely appreciate the trust and confidence that the institution has placed in me. I am committed to upholding the principles and values of IIT Bombay and had tried my best to build a positive impact during my time as an intern.

## **Table of contents**

1.	Introduction	01
2.	What is Django	02
	2.1 Goal of Django	02
	2.2 Model-View-Controller (MVC)	02
	2.3 Model-View-Template (MVT)	03
3.	Google Sheet Api Service	04
	3.1 The Google Sheets API	04
	3.2 Gspread	04
4.	API(Application Programming Interface)	05
	Crud Operations	05
	4.1 With Restframework	06
	4.2 Without Restframework	07
5.	CRUD based API Views	08
	5.1 Function based API view	08
	5.2 Class Based APIView	09
6.	What is view set and what is model view set	10
	6.1 ViewSet	10
	6.2 ModelViewSet	10
7.	Django_filters	11
	7.1 SearchFilter	11
	7.2 OrderingFilter	11

8.	Pagination	12
	8.1 Page Number	12
	8.2 Items Per Page	12
	8.3 Previous and Next Buttons	12
9.	Regex	13
10.	ORM	14
	10.1 Objective	14
	10.2 Definition	14
	10.3 Key Concepts	14
	10.4 Usage	14
	10.5 Benefits	14
	10.6 Example	15
	10.7 Outcome	15
11.	AWK	16
	11.1 Pattern Matching	16
	11.2 Data Extraction and Reporting	16
	11.3 Text Transformation	16
	11.4 Built-in Functions	16
	11.5 Input	17
	11.6 Output	17
12.	Validation Error	18
	12.1 Objective	18
	12.2 Approach	18
	12.3 Outcome	18
13.	Conclusion	19

# Chapter 1

## Introduction

In our project, we worked extensively on the backend using Django. We created models and views for students and teachers, tackled data serialization with serializers, and explored various view sets and API views following RESTful principles.

Beyond Django, we seamlessly integrated our app with the Google Console Developer, designing forms for job descriptions. Our journey included delving into Object-Relational Mapping (ORM) for database interactions.

We also deepened our understanding of Django filters and explored AWK for added versatility. In a nutshell, our exploration covered Django development essentials, from models and views to serializers, API views, and external service integration. Our grasp of ORM, Django filters, and AWK highlights our commitment to mastering web development intricacies.

# Chapter 2

## What is Django ?

Django is a high-level web framework written in Python that encourages rapid development and clean, pragmatic design. It follows the Model-View-Controller (MVC) architectural pattern, although in Django, it's often referred to as the Model-View-Template (MVT) pattern.

### 2.1 GOAL of Django:

Django's primary goal is to make it easier for developers to build web applications quickly and efficiently by providing a robust set of tools and conventions. It follows the "Don't Repeat Yourself" (DRY) and "Convention over Configuration" principles, reducing the amount of code you need to write for common use cases.

Django is open source and has a large and active community. It's widely used for building a variety of web applications, from small projects to large, complex systems.

### 2.2 Model-View-Controller (MVC):

**Model:** Represents the data and business logic of the application. It manages the data, logic, and rules of the application.

**View:** Presents the data to the user and handles the user interface. It is responsible for displaying the data received from the controller.

**Controller:** Manages user input and controls the flow of data between the Model and the View. It processes user actions and updates the Model and View accordingly. In MVC, the Model and View are separate entities, and the Controller acts as an intermediary between them, handling user input and updating the Model and View as needed.

## 2.3 Model-View-Template (MVT) - Django's variant of MVC:

**Model:** Represents the data structure and business logic. It defines how data is stored, retrieved, and processed.

**View:** Handles the presentation logic and interacts with the Model to get the necessary data. It decides what data to display and delegates the rendering to the Template.

**Template:** Defines how the data received from the View should be presented. It focuses on the structure and appearance of the final output.

**Models:** Define your data models using Python classes. These classes define the structure of your database tables and include fields with types such as CharField, IntegerField, DateField, etc.

**Views:** Views handle the logic of your application. They receive requests, process the data as necessary, and return responses. Views can be simple functions or more complex classes.

**Templates:** Templates are used to define the structure of HTML pages, and they allow you to embed Python-like code using Django's template language. Templates help you separate the presentation logic from the business logic.

**URLs:** Django uses a URL conf (URL configuration) to map URLs to views. This is done through regular expressions, making it easy to define URL patterns and direct them to the appropriate view.

**Forms:** Django provides a powerful form handling system that simplifies the process of collecting and validating user input. Forms can be used to handle HTML forms on the client side and manage data on the server side.

**Admin Site:** Django includes a built-in admin interface that allows you to manage your application's data through a web-based interface. It's a quick way to perform CRUD (Create, Read, Update, Delete) operations on your models.

# Chapter 3

## Google sheet API Service:

**Task:** Take data from the JRS google form and then store data in the spoken tutorial database and extract according to need

### Process to achieve it :

To achieve this task I used the google sheet api and gspread.

### 3.1 The Google Sheets API:

It is a RESTful API provided by Google to interact with Google Sheets programmatically. It allows you to read and write data to sheets, create and manage sheets, and perform other operations..

**Authentication:** You need to set up API credentials, obtain an API key or OAuth 2.0 credentials, and authenticate your requests.

**Usage:** You can make HTTP requests to the API endpoint, and the data is usually exchanged in JSON format.

Google Sheet API Documentation:

<https://developers.google.com/sheets/api/guides/concepts>

### 3.2 Gspread:-

gspread is a Python library that simplifies the interaction with Google Sheets. It provides a high-level interface for working with Google Sheets and is built on top of the Google Sheets API.

**Authentication:** Similar to the Google Sheets API, you need to set up credentials for your project. gspread supports both OAuth 2.0 and service account authentication.

**Usage:** With gspread, you can access and manipulate Google Sheets using Python code, making it easier to work with Sheets without dealing directly with HTTP requests.

GitHub Repository: <https://github.com/burnash/gspread>

Gspread Documentation : <https://docs.gspread.org/en/latest/user-guide.html>



# Chapter 4

## Api :

### Crud operations in Django:

- With restframework
- Without restframework

**Task:** create api in django for company registration form

### **Process:**

For creating API we followed crud operation

CRUD operations are fundamental when working with databases and data storage systems, and they are commonly implemented in web applications. One popular framework for building web APIs in Python is Django, and Django REST Framework (DRF) is an extension to Django that makes it easier to build RESTful APIs.

CRUD stands for Create, Read, Update, and Delete. It represents the basic operations that can be performed on data. These operations are fundamental in database management systems and are applicable to various types of data storage, including databases, spreadsheets, and more. Here's a brief explanation of each CRUD operation:

### **Create (C):**

Definition: Creating new records or entries in a database or data storage system.

Example: Inserting a new row of data in a database table or adding a new sheet to a Google Sheets document.

### **Read (R):**

Definition: Retrieving or reading existing data from a database or data storage system.

Example: Querying a database to fetch information from specific rows or reading values from cells in a spreadsheet.

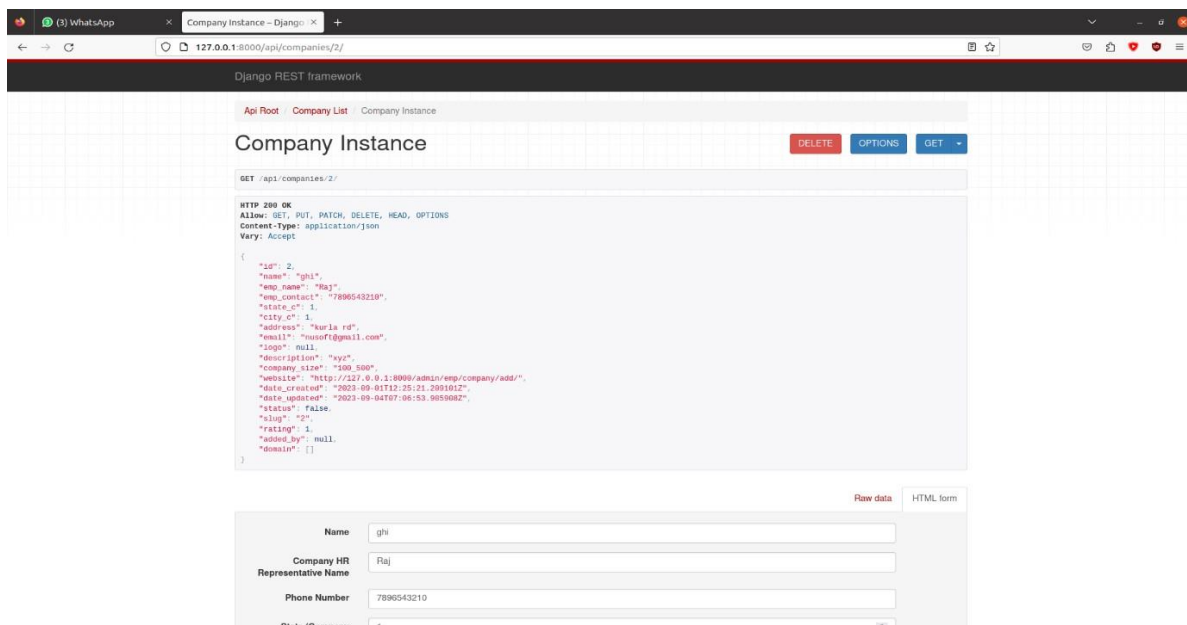
### Update (U):

Definition: Modifying or updating existing data in a database or data storage system.

Example: Updating the content of a specific row or cell with new information.

### Delete (D):

Definition: Removing or deleting existing data from a database or data storage system. Example: Deleting a record from a database table or removing a sheet from a spreadsheet document.



The screenshot shows a web browser window with the URL `127.0.0.1:8000/api/companies/2/`. The page title is "Company Instance" and it features buttons for "DELETE", "OPTIONS", and "GET". The main content area displays the following information:

GET /api/companies/2/

HTTP 200 OK  
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
  "id": 2,
  "name": "ghi",
  "emp_name": "Raj",
  "emp_contact": "7896543210",
  "state_c": 1,
  "city_c": 1,
  "address": "kuria rd",
  "email": "yusof@gmail.com",
  "log": null,
  "description": "xyz",
  "company_size": "100-500",
  "website": "http://127.0.0.1:8000/admin/emp/company/add/",
  "date_created": "2023-09-01T12:05:21.209181Z",
  "date_updated": "2023-09-04T07:06:53.965968Z",
  "status": false,
  "slug": "2",
  "rating": 1,
  "added_by": null,
  "domain": []
}
```

Below the JSON response, there are two tabs: "Raw data" and "HTML form". The "HTML form" tab is active, showing a form with the following fields:

Name	<input type="text" value="ghi"/>
Company HR Representative Name	<input type="text" value="Raj"/>
Phone Number	<input type="text" value="7896543210"/>
State (Company)	<input type="text" value="1"/>

## Serializer

In Django REST Framework, serializers are responsible for converting complex data such as querysets and model instances to native Python datatypes (called serialization) that can then be easily rendered into JSON, XML or other content types which is understandable by Front End. Serializers are also responsible for deserialization which means it allows parsed data to be converted back into complex types, after first validating the incoming data.

A serializer class is very similar to a Django Form and ModelForm class, and includes similar validation flags on the various fields, such as required, max\_length and default. DRF provides a Serializer class which gives you a powerful, generic way to control the output of your responses, as well as a ModelSerializer class which provides a useful shortcut for creating serializers that deal with model instances and querysets.

# Chapter 5

## CRUD based API Views

- i) Function based API view
  - a) Decorator
- ii) Classed based API View
  - a) API View
  - b) Generic API View

### 5.1 Function Based api view

This wrapper provide a few bits of functionality such as making sure you receive Request instances in your view, and adding context to Response objects so that content negotiation can be performed

The wrapper also provide behaviour such as returning 405 Method Not Allowed responses when appropriate, and handling any ParseError exceptions that occur when accessing request.data with malformed input.

By default only GET methods will be accepted. Other methods will respond with "405 Method Not Allowed".

```
(@api view())
```

```
@api view(['GET', 'POST', 'PUT', 'DELETE'])
```

```
def function name(request):
```

Similarly we applied Function based API View for the student, job, domain and discipline.

### 5.2 Class Based APIView

REST framework provides an APIView class, which subclasses Django's View class. APIView classes are different from regular View classes in the following ways:

Requests passed to the handler methods will be REST framework's Request instances, not Django's HttpRequest instances.

Handler methods may return REST framework's Response, instead of Django's HttpResponse. The view will manage content negotiation and setting the correct renderer on the response.

Any APIException exceptions will be caught and mediated into appropriate responses.

Incoming requests will be authenticated and appropriate permission and/or throttle checks will be run before dispatching the request to the handler method

Similarly we applied Class based API View for the student, job, domain and discipline.

# Chapter 6

## What is view set and what is model view set

### **i. ViewSet:**

A ViewSet in DRF is an abstraction that combines the logic for handling HTTP methods with the data retrieval and manipulation. It is similar to Django's class-based views but is tailored for API development. A ViewSet can handle various actions such as listing, creating, retrieving, updating, and deleting instances of a model.

### **ii. ModelViewSet:**

A ModelViewSet is a specific type of ViewSet provided by DRF that is designed to work with Django models. It automatically generates CRUD (Create, Read, Update, Delete) operations based on the model and is a convenient way to create a fully functional API for a Django model.

# Chapter 7

## **Django\_filters :**

Django Filters is a powerful application for filtering the results of a Django QuerySet based on user input. It provides a convenient way to allow users to dynamically filter data in a Django application, typically in the context of a web interface. Django Filters works well with Django models and integrates seamlessly with Django views and templates.

In Django REST Framework (DRF), the SearchFilter and OrderingFilter are two commonly used filters that help developers implement search functionality and control the ordering of query results in API views.

### **7.1 SearchFilter**

The SearchFilter is used for searching and filtering query results based on a search term provided by the user. It performs a case-insensitive search on the specified fields of the model.

### **7.2 OrderingFilter**

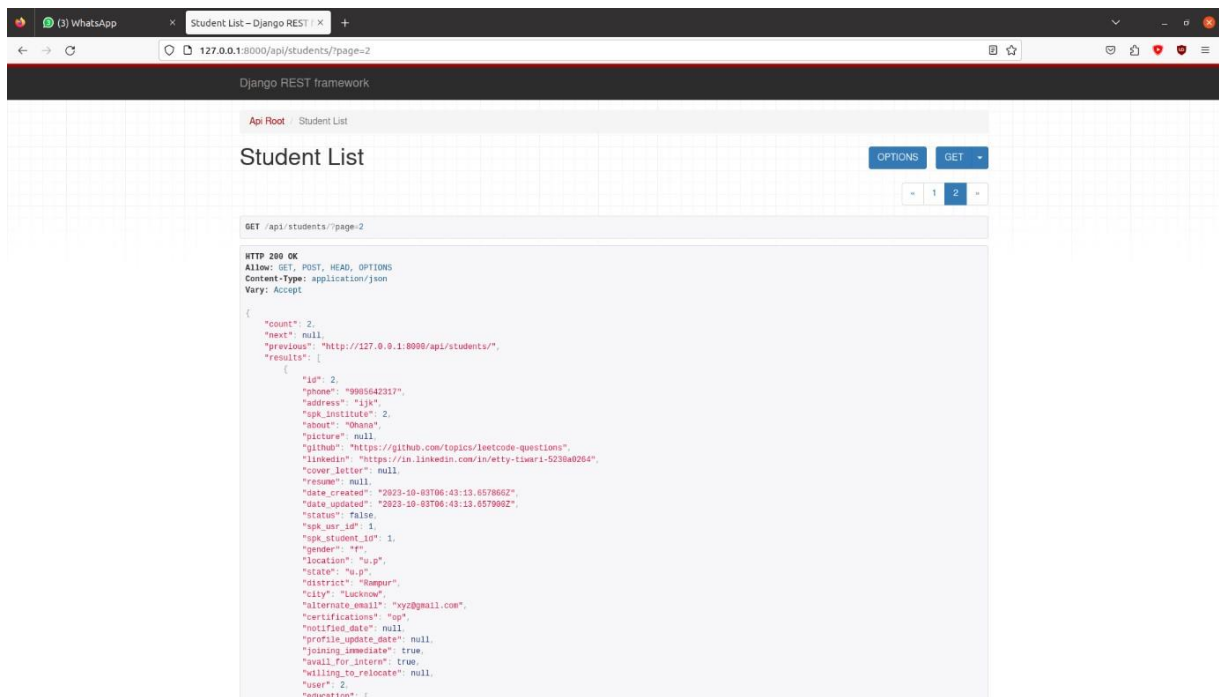
The OrderingFilter is used to control the ordering of query results based on one or more fields. It allows users to specify the order in which the results should be returned.

Similarly we applied Django filters for the company, student, job, domain and discipline.

# Chapter 8

## Pagination:

In Django, pagination is a mechanism that allows you to divide a large set of query results into smaller, more manageable subsets or pages. This is particularly useful when you have a large dataset to display, such as a list of blog posts or search results, and you want to display only a limited number of items on each page.



**Pagination** typically involves the following components:

**8.1 Page Number:** This indicates which page of results you are currently viewing. Users can navigate between pages using page numbers.

**8.2 Items Per Page:** This defines how many items should be displayed on each page. For example, you might choose to display 10 blog posts per page.

**8.3 Previous and Next Buttons:** These buttons allow users to navigate to the previous or next page.



# Chapter 9

## Regex

**Task:** Extracting Minimum and maximum salary from the company google form

**Process:** We used regex to extract text separately

Regex, stands for Regular Expression, is a powerful tool for pattern matching and text manipulation. It allows you to define a pattern (a sequence of characters) and search for or manipulate text that matches that pattern. Regular expressions are widely used in programming, text editors, and various other applications where you need to work with text data efficiently.

# Chapter 10

## ORM:

Object-Relational Mapping (ORM) part:

### 10.1 Objective:

Achieve communication between a Django web application and a relational database without using SQL queries directly.

### 10.2 Definition:

ORM is a technique that maps database entities to Python objects and vice versa. In Django, it allows you to interact with the database using Python code and classes.

### 10.3 Key Concepts:

Models: Python classes representing database tables.

Fields: Attributes of models representing database columns.

QuerySet: Abstraction for database queries, allowing interaction with the database using Python.

### 10.4 Usage:

Models: Define models by creating Python classes that subclass `django.db.models.Model`.

Fields: Define attributes in models to represent database columns, specifying data types.

QuerySet: Use it to perform CRUD operations on the database using Python code rather than SQL queries.

### 10.5 Benefits:

Abstraction: Developers work with Python objects, abstracting away the underlying database structure.

Portability: Easily switch between different database backends without changing code.

**10.6 Example:**

Define a model with fields (e.g., CharField, IntegerField) to represent a table.

Use the model to create, read, update, and delete records in the database without writing SQL queries.

**10.7 Outcome:**

Developers can interact with the database using Python code, making it more convenient and abstracting away the complexities of SQL queries.

# Chapter 11

## AWK

**Task:** Make an output file(using AWK) which has 2 columns: Line number & to email so basically in the mail log file it contain many data but we just want the emails and the line number

AWK is a powerful and versatile programming language and command-line utility for pattern scanning and text processing. It is mainly used for processing and analyzing text files, particularly in the context of Unix and Unix-like operating systems.

**11.1 Pattern Matching:** awk is designed for processing text files line by line, applying patterns to identify and process specific lines or fields.

**11.2 Data Extraction and Reporting:** It is commonly used for extracting specific columns or fields from structured text data and generating reports based on patterns or conditions.

**11.3 Text Transformation:** awk allows you to perform various text transformations, such as changing the format of data, replacing text, or reordering fields.

**Variables and Control Structures:** awk includes variables, loops, and conditional statements, making it a more expressive language compared to simple text processing tools.

**11.4 Built-in Functions:** awk provides a variety of built-in functions for string manipulation, arithmetic operations, and other common task.

\

# 11.5 INPUT:

```

mailout
~/Desktop/mail

Loading mailout from ~/Desktop/mail

1 maillog:Oct 1 03:35:14 koyna postfix/qmgr[1953]: 4ABCF8454024: from=<no-reply@spoken-tutorial.org>, size=2630, rcpt=1 (queue active)
2 maillog:Oct 1 03:35:14 koyna postfix/qmgr[1953]: E80BF8446C35: from=<no-reply@spoken-tutorial.org>, size=2623, rcpt=1 (queue active)
3 maillog:Oct 1 03:35:14 koyna postfix/qmgr[1953]: 8BCEBF810C8C7: from=<no-reply@spoken-tutorial.org>, size=1663, rcpt=1 (queue active)
4 maillog:Oct 1 03:35:14 koyna postfix/smtp[30979]: 4ABCF8454024: host gmail-smtp-in.l.google.com[2080:1450:4010:c0f:1b] said: 452-4.2.2 The email account that you tried to reach is over quota. Please direct 452-4.2.2 the recipient to 452 4.2.2 https://support.google.com/mail/?p=OverQuotaTemp v10-20020a2e9f44000000b02bf23d2340s11445391jk.606 - gsntp (In reply to RCPT TO command)
5 maillog:Oct 1 03:35:14 koyna postfix/smtp[30981]: 8BCEBF810C8C7: host gmail-smtp-in.l.google.com[64.233.161.27] said: 452-4.2.2 The email account that you tried to reach is over quota. Please direct 452-4.2.2 the recipient to 452 4.2.2 https://support.google.com/mail/?p=OverQuotaTemp x29-20020a2c256d000000b0e503382af2b5s10926079lfm.603 - gsntp (In reply to RCPT TO command)
6 --
7 maillog:Oct 1 03:45:15 koyna postfix/qmgr[1953]: 30735F8446C36: from=<no-reply@spoken-tutorial.org>, size=2799, rcpt=1 (queue active)
8 maillog:Oct 1 03:45:15 koyna postfix/qmgr[1953]: E80BF8446C35: from=<no-reply@spoken-tutorial.org>, size=2624, rcpt=1 (queue active)
9 maillog:Oct 1 03:45:15 koyna postfix/qmgr[1953]: E7186F8446C33: from=<no-reply@spoken-tutorial.org>, size=2645, rcpt=1 (queue active)
10 maillog:Oct 1 03:45:15 koyna postfix/qmgr[1953]: 530B5F845EC23: from=<no-reply@spoken-tutorial.org>, size=2606, rcpt=1 (queue active)
11 maillog:Oct 1 03:45:15 koyna postfix/smtp[30986]: 30735F8446C36: host gmail-smtp-in.l.google.com[64.233.161.27] said: 452-4.2.2 The email account that you tried to reach is over quota. Please direct 452-4.2.2 the recipient to 452 4.2.2 https://support.google.com/mail/?p=OverQuotaTemp q1-20020a2c2510100000000084fbc3cc516s10287541fb.200 - gsntp (In reply to RCPT TO command)
12 maillog:Oct 1 03:45:15 koyna postfix/smtp[30988]: 530B5F845EC23: host gmail-smtp-in.l.google.com[64.233.161.27] said: 452-4.2.2 The email account that you tried to reach is over quota. Please direct 452-4.2.2 the recipient to 452 4.2.2 https://support.google.com/mail/?p=OverQuotaTemp x12-20020a2e9c8c000000b02bfc2023e34s11152577lji.649 - gsntp (In reply to RCPT TO command)
13 --
14 maillog:Oct 1 03:50:15 koyna postfix/qmgr[1953]: E82ABF8446881: from=<no-reply@spoken-tutorial.org>, size=2564, rcpt=1 (queue active)
15 maillog:Oct 1 03:50:15 koyna postfix/qmgr[1953]: 2237EF8444A3C: from=<no-reply@spoken-tutorial.org>, size=1598, rcpt=1 (queue active)
16 maillog:Oct 1 03:50:15 koyna postfix/smtp[31025]: 2237EF8444A3C: host gmail-smtp-in.l.google.com[2080:1450:4010:c08:1b] said: 452-4.2.2 The email account that you tried to reach is over quota. Please direct 452-4.2.2 the recipient to 452 4.2.2 https://support.google.com/mail/?p=OverQuotaTemp n4-20020a2e90440000000000c03384085s1106318011jg.54 - gsntp (In reply to RCPT TO command)
17 maillog:Oct 1 03:50:16 koyna postfix/smtp[31025]: 2237EF8444A3C: to=aherakarnikt@gmail.com, relay=gmail-smtp-in.l.google.com[64.233.161.26]:25, delay=63494, delays=8494/0.01/9.31/0.04, dsn=4.2.2, status=deferred (host gmail-smtp-in.l.google.com[64.233.161.26] said: 452-4.2.2 The email account that you tried to reach is over quota. Please direct 452-4.2.2 the recipient to 452 4.2.2 https://support.google.com/mail/?p=OverQuotaTemp e02-20020a056512480200b004ff95c293afs199113551fb.143 - gsntp (In reply to RCPT TO command))
18 --
19 maillog:Oct 1 04:05:15 koyna postfix/qmgr[1953]: 4E45BF8445413: from=<no-reply@spoken-tutorial.org>, size=1580, rcpt=1 (queue active)
20 maillog:Oct 1 04:05:15 koyna postfix/qmgr[1953]: 90BF3F810C533: from=<no-reply@spoken-tutorial.org>, size=1596, rcpt=1 (queue active)
21 maillog:Oct 1 04:05:15 koyna postfix/qmgr[1953]: E81FF8445400: from=<no-reply@spoken-tutorial.org>, size=2569, rcpt=1 (queue active)
22 maillog:Oct 1 04:05:15 koyna postfix/qmgr[1953]: 8255EF8444699: from=<no-reply@spoken-tutorial.org>, size=1607, rcpt=1 (queue active)
23 maillog:Oct 1 04:05:15 koyna postfix/qmgr[1953]: 25E7F8445400: from=<no-reply@spoken-tutorial.org>, size=2557, rcpt=1 (queue active)
24 maillog:Oct 1 04:05:15 koyna postfix/smtp[31166]: connect to betand.spoken-tutorial.in[95.216.73.143]:25: No route to host
25 maillog:Oct 1 04:05:15 koyna postfix/smtp[31166]: 4DF9CF8422008: to=<noreply@betand.spoken-tutorial.in>, relay=none, delay=10112, delays=10112/0.02/0.11/0, dsn=4.4.1, status=deferred (connect to betand.spoken-tutorial.in[95.216.73.143]:25: No route to host)
26 --
27 maillog:Oct 1 04:10:15 koyna postfix/qmgr[1953]: 72020F8445416: from=<no-reply@spoken-tutorial.org>, size=1596, rcpt=1 (queue active)
28 maillog:Oct 1 04:10:15 koyna postfix/qmgr[1953]: 66F3DF8445808: from=<no-reply@spoken-tutorial.org>, size=1523, rcpt=1 (queue active)
29 maillog:Oct 1 04:10:15 koyna postfix/smtp[31177]: 66F3DF8445808: host gmail-smtp-in.l.google.com[2080:1450:4010:c08:1b] said: 452-4.2.2 The email account that you tried to reach is over quota. Please direct 452-4.2.2 the recipient to 452 4.2.2 https://support.google.com/mail/?p=OverQuotaTemp t2-20020a2e9550e00000b02bfbf672d371s108384081jh.594 - gsntp (In reply to RCPT TO command)
30 maillog:Oct 1 04:10:15 koyna postfix/smtp[31176]: 72020F8445416: host gmail-smtp-in.l.google.com[64.233.165.27] said: 452-4.2.2 The email account that you tried to reach is over quota. Please direct 452-4.2.2 the recipient to 452 4.2.2 https://support.google.com/mail/?p=OverQuotaTemp z6-20020a2e7e0600000000b02bfbf255eedbs11504107ljc.40 - gsntp (In reply to RCPT TO command)
31 --
32 maillog:Oct 1 04:16:46 koyna postfix/ptcleanup[31210]: EC0B1F845E432: uid=48 from=<no-reply@spoken-tutorial.org>
33 maillog:Oct 1 04:16:46 koyna postfix/cleanup[31220]: EC0B1F845E432: message-id=<6518a550ca7154.4593706@onlinetest.spoken-tutorial.org>
34 maillog:Oct 1 04:16:46 koyna opendkim[1077]: EC0B1F845E432: DKIM-Signature field added (s=koyna, d=spoken-tutorial.org)
35 maillog:Oct 1 04:16:47 koyna postfix/qmgr[1953]: 25E7F8445400: from=<no-reply@spoken-tutorial.org>, size=2557, rcpt=1 (queue active)
36 maillog:Oct 1 04:16:47 koyna postfix/ptcleanup[31210]: 15131F845F10E: uid=48 from=<no-reply@spoken-tutorial.org>
37 maillog:Oct 1 04:16:47 koyna postfix/cleanup[31220]: 15131F845F10E: message-id=<6518a550efbd26.50392529@onlinetest.spoken-tutorial.org>
38 maillog:Oct 1 04:16:47 koyna opendkim[1077]: 15131F845F10E: DKIM-Signature field added (s=koyna, d=spoken-tutorial.org)
39 maillog:Oct 1 04:16:47 koyna postfix/qmgr[1953]: 15131F845F10E: from=<no-reply@spoken-tutorial.org>, size=2329, rcpt=1 (queue active)
40 maillog:Oct 1 04:16:47 koyna postfix/ptcleanup[31210]: 31C0BF845F10E: uid=48 from=<no-reply@spoken-tutorial.org>
41 maillog:Oct 1 04:16:47 koyna postfix/cleanup[31220]: 31C0BF845F10E: message-id=<6518a550f30032.4433720@onlinetest.spoken-tutorial.org>

```

# 11.6 OUTPUT:

```

output_file.txt
~/Desktop/mail

1 |/ aherakarnikt@gmail.com
2 | noreply@betand.spoken-tutorial.in
3 | b2lec1@pace.ac.in
4 | ash1e2407@gmail.com
5 | soulavinas7@gmail.com
6 | aherakarnikt@gmail.com
7 | noreply@betand.spoken-tutorial.in
8 | deveshahajans@gmail.com
9 | ash1e2407@gmail.com
10 | noreply@pace.ac.in
11 | soulavinas7@gmail.com
12 | aherakarnikt@gmail.com
13 | ash1e2407@gmail.com
14 | soulavinas7@gmail.com
15 | aherakarnikt@gmail.com
16 | ash1e2407@gmail.com
17 | soulavinas7@gmail.com
18 | aherakarnikt@gmail.com
19 | soulavinas7@gmail.com
20 | ash1e2407@gmail.com
21 | noreply@pace.ac.in
22 | noreply@pace.ac.in
23 | noreply@pace.ac.in
24 | noreply@betand.spoken-tutorial.in
25 | deveshahajans@gmail.com
26 | ash1e2407@gmail.com
27 | soulavinas7@gmail.com
28 | ash1e2407@gmail.com
29 | ash1e2407@gmail.com
30 | ash1e2407@gmail.com
31 | ash1e2407@gmail.com
32 | ash1e2407@gmail.com
33 | aherakarnikt@gmail.com
34 | ash1e2407@gmail.com
35 | ash1e2407@gmail.com
36 | ash1e2407@gmail.com
37 | ash1e2407@gmail.com
38 | ash1e2407@gmail.com
39 | ash1e2407@gmail.com
40 | ash1e2407@gmail.com
41 | aherakarnikt@gmail.com
42 | noreply@betand.spoken-tutorial.in
43 | ash1e2407@gmail.com
44 | ash1e2407@gmail.com
45 | ash1e2407@gmail.com
46 | ash1e2407@gmail.com
47 | ash1e2407@gmail.com
48 | ash1e2407@gmail.com
49 | ash1e2407@gmail.com
50 | ash1e2407@gmail.com
51 | ash1e2407@gmail.com
52 | ash1e2407@gmail.com
53 | ash1e2407@gmail.com
54 | ash1e2407@gmail.com

```

# Chapter 12

## Validation Error:

### 12.1 Objective

Ensure that when a user registers a company, whether using the full form or short form, there are no duplicates in the database.

### 12.2 Approach

Check if the entered company name (full form or short form) already exists in the database. If it does, raise a validation error to prevent duplicate entries.

### 12.3 Outcome

Users cannot register a company with a name that is too similar to an existing one, avoiding duplicates in the database.

# Chapter 13

## Conclusion

This internship has been a transformative journey that has empowered me with invaluable insights, skills, and experiences that will resonate throughout my personal and professional life. The opportunities to apply theoretical knowledge in a practical setting, to contribute to meaningful projects, and to work alongside exceptional individuals have been truly life-changing.

I extend my heartfelt appreciation to everyone involved in making this internship a remarkable and influential chapter of my academic and professional trajectory. Your unwavering support, belief in my abilities, and the wealth of opportunities and knowledge I have gained through this internship at Spoken Tutorial Team, IIT Bombay, will forever hold a special place in my heart.

## Reference

- **Github Link :**  
<https://github.com/Spoken-tutorial/Employer-Recommendation-System.git>
- **Django Documentation:**  
<https://docs.djangoproject.com/en/4.2/>
- **Youtube Link:**  
[https://youtube.com/playlist?list=PLbGui\\_ZyuhijTKyrlu-0g5GcP9nUp\\_HlN&si=\\_a3YyNcoG88rciL9](https://youtube.com/playlist?list=PLbGui_ZyuhijTKyrlu-0g5GcP9nUp_HlN&si=_a3YyNcoG88rciL9)
- **REST Framework Link:**  
<https://blog.logrocket.com/django-rest-framework-create-api/>
- **AWK Link:**  
<https://www.gnu.org/s/gawk/manual/gawk.html>