
RESUME AUTOMATION USING ARTIFICIAL INTELLIGENCE AND PYTHON WEB APPLICATION

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Degree in Computer Science

By

SANGEETHA S (REG NO: 38290077)

SONIA J (REG NO: 38290083)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

MARCH 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC
(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SONIA.J (Reg. No: 38290083)** and **SANGEETHA.S (Reg. No: 38290077)** who carried out the project entitled **"RESUME ANUTOMATION USING ARTIFICIAL INTELLIGENCE AND PYTHON WEB APPLICATION"** under my / our supervision from November 2020 to March 2021.

Internal Guide

External Guide (if Applicable)

Head of the Department
Dr. L. LAKSHMANAN, M.E., PH.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **SANGEETHA .S (REG NO: 38290077)** hereby declare that the Project Report entitled "**RESUME AUTOMATION USING ARTIFICIAL INTELLIGENCE AND PYTHON WEB APPLICATION**" done by me and my team mate under the guidance of **Dr. G. MEERAGANDHI, B.E., M.E., PH.D.,** is submitted in partial fulfillment of the requirements for the award of Bachelor of degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D.**, Dean, School of Computing and **Dr. L. LAKSHMANAN, M.E., Ph.D.**, Head of the Department, Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr G. MEERAGANDHI, B.E., M.E., PH.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways For the Completion of the project.

ABSTRACT

Django may be a web development framework that assists in building and maintaining quality web applications. Angular CLI is an open-source free framework with a node library package which allow us to develop a dynamic single page application. Resume filter Web Application gives efficient virtual assistance to the Company.

In a company, HR gets many resumes, in order to hire employees based on their company's requirement, it can be done by passing keywords. Ease to access. Django makes it easier to create better web apps quickly and with less code.

Finding suitable candidates for an open role could be a daunting task, especially when there are many applicants. It can impede team progress for getting the right person on the right time. A way of “Resume Automation using Artificial Intelligence and Python Web Application” could really ease the tedious process of fair screening and shortlisting, it would certainly expedite the candidate selection and decision making process.

This system could work with a large number of resumes for first identify the right categories using different keywords, once matching has been done then as per the job description, top candidates could be ranked using resume filters.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	vi
	LIST OF FIGURES	x
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	1
1.1	OVERVIEW OF PROJECT	1
1.2	DJANGO FRAMEWORK	1
1.2.1	DJANGO ARCHITECTURE	2
1.2.2	DJANGO ADVANTAGES	3
1.3	ANGULAR CLI FRAMEWORK	3
1.3.1	MODULES	4
1.3.2	COMPONENTS	4
1.3.3	ROUTING	5
1.3.4	ANGULAR FRAMEWORK	
	ARCHITECTURE	6
2	LITERATURE SURVEY	7
3	AIM AND SCOPE OF PRESENT INVESTIGATION	9

	3.1	AIM	9
	3.2	FUTURE SCOPE	9
4		EXPERIMENTAL OR MATERIAL METHODS	10
	4.1	RESUME FILTER	10
	4.2	BACK-END PROCESS	11
	4.2.1	SETTING.PY	11
	4.2.2	MODELS	17
	4.2.3	VIEWS.PY	18
	4.2.4	URLS.PY	19
	4.3	FRONT-END PROCESS	20
	4.3.1	MODELS	20
	4.3.2	COMPONENTS	20
	4.3.3	SERVICES	21
	4.3.4	ROUTING	22
	4.3.5	“.ts” FILE	22
	4.4	METHODS OR LIBRARIES USED	24
	4.4.1	PYPDF2:	25
	4.4.2	GLOB	25
	4.5	MVT PATTERN	26
	4.6	MYSQL	27
	4.7	ENVIRONMENT	27

	4.7.1	SUBLIME TEXT	27
	4.7.2	VISUAL STUDIO CODE	28
5		RESULTS AND PERFORMANCE ANALYSIS	29
	5.1	SCREEN SHOTS	29
	5.2	HOW TO RUN?	31
	5.2.1	BACK-END COMPILING AND RUNNING PROCESS	31
	5.2.2	FRONT-END COMPILING AND RUNNING PROCESS	32
	5.3	ADVANTAGES OF RESUME FILTER WEB APPLICATION	33
6		SUMMARY AND CONCLUSIONS	
	6.1	SUMMARY	34
	6.2	CONCLUSIONS	34
		REFERENCE	35
		APPENDIX	36
		SOURCE CODE	36

LIST OF FIGURES

FIGURE NO:	FIGURE NAME	PAGE NO:
1.1	Django Logo	1
1.2	Django Architecture Diagram	2
1.3	Angular Logo	4
1.4	Angular Architecture Diagram	6
4.1	Resume filter Logo	10
4.2	Resume Filter Architecture	10
4.3	Installed Apps	12
4.4	Middleware	13
4.5	Templates	14
4.6	Database	15
4.7	Models	17
4.8	Urls.py	19
4.9	MVT Flow Diagram	26
4.10	MySQL Logo	27
4.11	Sublime Text Logo	28
4.12	Visual Studio Code Logo	28
5.1	Registration Page	29
5.2	Login Page	30
5.3	Candidate List	30
5.4	After Running Back-End Process	32
5.5	Compiling Front-End Process	33

LIST OF ABBREVIATIONS

1. **HR** - Human Resources
2. **CSS** - Cascading Style Sheet
3. **SCSS** - Syntactically Awesome Style Sheet
4. **API** - Application programming interface
5. **HTML** - Hypertext Markup Language
6. **DOM** - Document Object Model
7. **URL** - Uniform Resource Locator
8. **PDF** - Portable Document Format
9. **MVT** - Model-View-Template
10. **AI** - Artificial Intelligence

CHAPTER 1

INTRODUCTION

1.1 OVER VIEW OF PROJECT

Django comes with a light-weight web server to facilitate end- to-end application development and testing. Then when it comes to User Interface Angular is one of the top most designing frameworks. We can customize our design with more attractive CSS, SCSS, and Bootstrap from the node library package. Angular with django is the best framework to build a web application.

1.2 DJANGO FRAMEWORK

Django may be a high-level Python web framework that encourages rapid development and clean, pragmatic design. Django makes it easier to create better web apps quickly and with less code. Django has built-in support for Ajax, RSS, Caching and various other frameworks.



Fig1.2: Django Logo

1.2.1 DJANGO ARCHITECTURE

Django follows a Model-View-Controller (MVC) architecture, which is split up into three different parts:

- **MODEL:** Model is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySql).
- **VIEW:** View is the user interface — what you see in your browser when you visit a website. These are represented by HTML/CSS files.
- **CONTROLLER:** Controller is the middleman that connects the view and model together, meaning that it is the one passing data from the model to the view.

Django Architecture

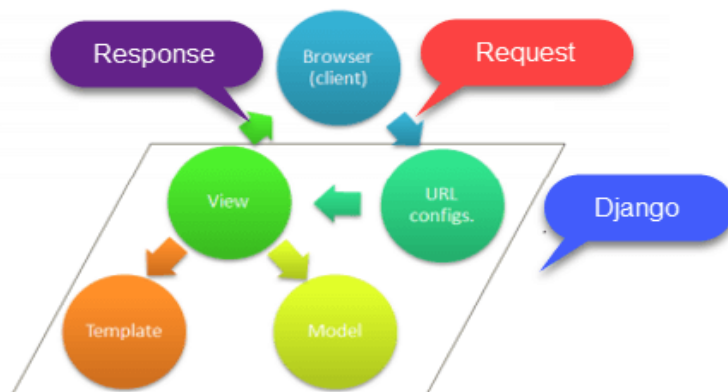


Fig1.2: Django Architecture Diagram

1.2.2 DJANGO ADVANTAGES

- Django supports multilingual websites through its built-in internationalization system.
- It reduces many development time and increases productivity.
- Django provides a nice ready-to-use user interface for administrative activities.
- Django is much simpler and the codes are easy when doing it in Python.
- It had been built to slip the applications from imagination to reality during a blink.
- Django has been in the picture for years and in that tenure, it has emerged as the choice of a lot of businesses for their web applications.
- Django is cross-platform also compatible with the majority of databases making it highly versatile.
- Django because it is powerful enough to create a full-fledged API in only two or three lines of code.
- Django offers the supply of working with its older versions and makes use of its older formats and features.

1.3 ANGULAR CLI FRAMEWORK

Angular may be a platform and framework for building single-page client applications using HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a group of TypeScript libraries that you simply import into your apps.



Fig1.3: Angular Logo

1.3.1 MODULES

Angular Ng Modules differ from and complement JavaScript (ES2015) modules. An Ng Module declares a compilation context for a group of components that's dedicated to an application domain, a workflow, or a closely related set of capabilities. An Ng Module can associate its components with related code, like services, to make functional units.

Every Angular app features a root module, conventionally named App Module, which provides the bootstrap mechanism that launches the appliance. An app typically contains many functional modules.

1.3.2 COMPONENTS

Every Angular application has a minimum of one component, the basis component that connects a component hierarchy with the page document object model (DOM). Each component defines a class that contains application data and logic, and is associated with an HTML template that defines a view to be displayed in a target environment.

The `@Component()` decorator identifies the category immediately below

it as a component, and provides the template and related component-specific metadata.

1.3.3 ROUTING

The Angular Router Ng Module provides a service that lets you define a navigation path among the different application states and view hierarchies in your app. It is modeled on the familiar browser navigation conventions:

- Enter a URL within the address bar and therefore the browser navigates to a Corresponding page.
- Click links on the page and the browser navigates to a new page.
- Click the browser's back and forward buttons and therefore the browser navigates backward and forward through the history of pages you've seen.

The router maps URL-like paths to views rather than pages. When a user performs an action, like clicking a link, that might load a replacement page within the browser, the router intercepts the browser's behavior, and shows or hides view hierarchies.

If the router determines that the present application state requires particular functionality, and therefore the module that defines it hasn't been loaded, the router can lazy-load the module on demand.

The router interprets a link URL according to your app's view navigation rules and data state. You can navigate to new views when the user clicks a button or selects from a drop box, or in response to another stimulus from any source. The router logs activity within the browser's history, therefore the back and forward buttons work also.

To define navigation rules, you associate navigation paths together with your components. A path uses a URL-like syntax that integrates your program

data, in much an equivalent way that template syntax integrates your views together with your program data. You can then apply program logic to settle on which views to point out or to cover, in response to user input and your own access rules.

1.3.4 ANGULAR FRAMEWORK ARCHITECTURE

The metadata for a component class has a template that defines a view. A template combines ordinary HTML with Angular directives and binding which allow Angular to modify the HTML before presenting on the screen.

The metadata for a service class consists of information Angular needs to make it available to components through dependency injection (DI).

For better understanding you can refer the Fig1.4 Angular Architecture Diagram given as follows:

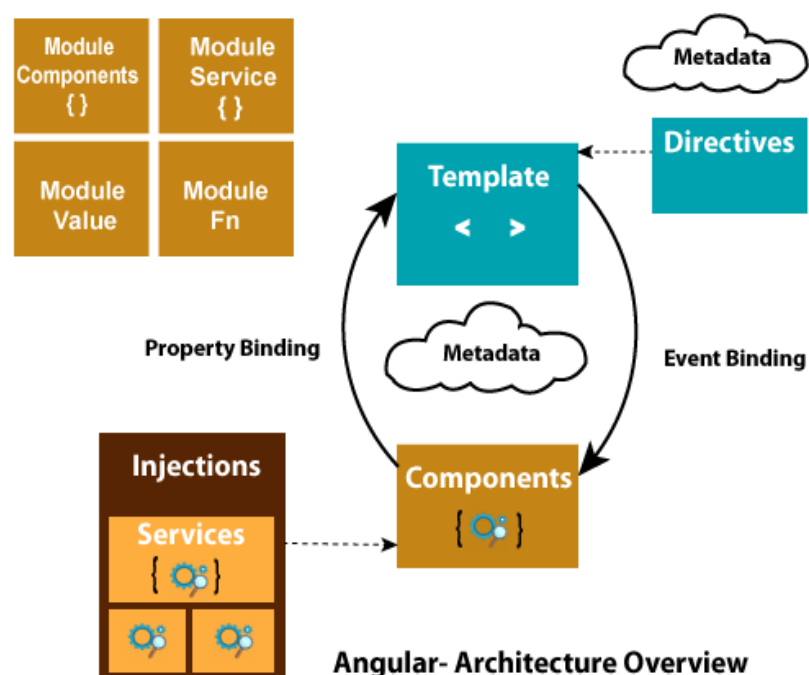


Fig1.4: Angular Architecture Diagram

CHAPTER 2

LITERATURE SURVEY

In the literature, the different techniques are evolved to retrieve information from PDF file and filter the required pdf document from bulk pdf documents. The methodology applied to develop such a system is used either in image processing or in fully or partially implemented PDF extraction library tools. Hui Chao and Jian Fan has developed techniques in [10], that identifies the logical components of a PDF document page into different logical structure regions such as text blocks, images, vector graphics blocks and compound blocks. The PDF document page is separated into three layers of text, image and vector graphic. Each layer become an individual PDF document. Hence the logical structural component is placed in each stratum. For text component's font, size, line spacing and color are identified. For image component the shapes are named. For vector graphics component the object is identified. all separated component are combined and convert to compound component. Similarly Dejean and Meunier in [5] analyzed the logical structure of PDF documents by first recognizing their table contents and re-constructing their logical hierarchy. The Xpdf library is used to extract the different streams contained in the PDF document like text, bitmap and vector images. Sarang Pitale, Tripti Sharma in [13] conducted a survey on different extraction tools for extracting contents like text, images, tables, fonts, etc. from e-books of PDF format for the implementation of digital library. Authors explains the capability and features of some available PDF extraction tools. The tools are used to convert from PDF to text format. Hadjar, K. et. al in [6] have described a novel approach Xed, where they focussed on combining PDF symbol analysis with traditional document image processing techniques. The results of complex layout analysis is used as input to drive a topological sorting algorithm for text pieces. The data are extracted from PDF using Jpedal tool.

The matching between the extracted objects from PDF document and the result of layout analysis is also presented. On the other hand R. Mohemad, et al in [12] described an approach which identify and recognize the layout and structure of PDF document automatically together with the text, paragraph and tabular data. The Jpedal extraction tool identifies all PDF objects and returns the co-ordinates of rectangles in each page of PDF document. To identify tables, the hierarchical agglomerative clustering algorithm is implemented to group the closest tokens by checking the co-ordinate position of objects.

Fang et al [4] proposed a new method for extracting information from PDF files. They use a modified version of PdfBox, to extract and parse the text contained within a PDF file and injects tags into text information to transform it into semi-structured data to aid searching. The focus was on identifying the title, author, address, abstract and keywords of each paper by taking advantage of the text property information from PDFBox. Whereas Hassan, T. In [8] developed a method to obtain text and images from a PDF file by analyzing the layout structure. They used PDFBox library to extract object from PDF file. PDFBox returns data in rectangular box with its coordinate values. They proposed a bottom-up segmentation algorithm to group segments into blocks representing logical elements on the page.

Burcu Yildiz, et al in [3] described an approach for extracting table from PDF file. They use PdftoHtml tool to extract text from PDF file. The authors used a document-based approach to extract table. The tool returns text chunks with its property information such as top, left, width, height and font. The algorithms are reduced to merge text, position the table, insert cells and content of PDF file. Further a prototype is generated to evaluate the performance of algorithm. This approach does not handle any language specific features. Whereas X.Y. Lin, et al in [14] proposed a method by combining rule based and learning based methods to identify mathematical expressions in PDF document. This work is divided into different steps in the first preprocessing step the different types of objects are matched with mathematical expression elements. In the second step text lines are extracted, in the third step feature analysis is done to extract character and

layout features and then the rule based and support vector methods are applied to identify formula areas.

CHAPTER 3

AIM AND SCOPE OF PRESENT INVESTIGATION

3.1 AIM

Building AI based software with the help of MVT pattern which is good for stable performance, Python is the base language with the help of Django framework for the backend process and the user interface is build with the help of angular framework which is also stable while scaling the process.

3.2 FUTURE SCOPE

Future scope of this project includes improving the accuracy of the search results of the resumes, Also it can be filtered by various condition. Improving the accessibility will improve the standard of the project accessible for various type of user like blind people by adding voice search option.

Once we get enough of data from the current feature with the help of AI we could improve the searching option to more efficient.

CHAPTER 4

EXPERIMENTAL OR MATERIALS AND METHODS

4.1 RESUME FILTER

Resume filter Web Application works as a virtual assistance for every HR of a Company. The combination of the Django framework and Angular framework makes the application versatile and flexible. The application can handle large data at a time without any delay with the influence of Django. Angular allows the user to experience the multiple outcomes of the User Interface. Data has been stored in MySQL DB.



Fig4.1: Resume filter Logo

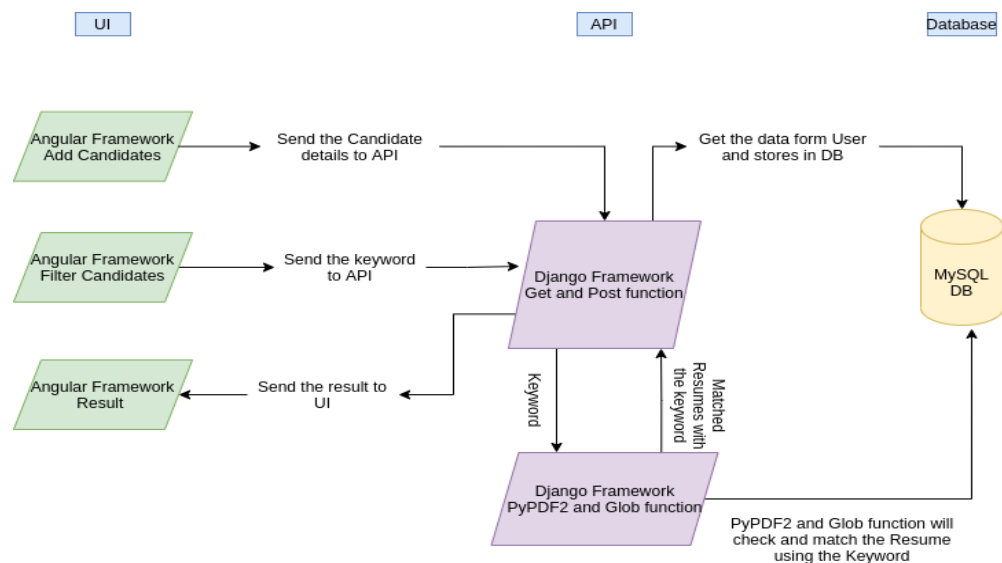


Fig4.2: Resume Filter Architecture

4.2 BACK-END PROCESS

We need to follow some certain architecture to develop project in Django. We see one-by-one.

4.2.1 SETTINGS.PY

settings.py is a core file in Django projects. It holds all the configuration values that your web app must work; database settings, logging configuration, where to seek out static files, API keys if you're employed with external APIs, and a bunch of other stuff.

Example code

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',
```

```

# Django REST framework
'rest_framework',

# Customers application
'filterResume.apps.FilterresumeConfig',

# CORS
'corsheaders',

]

```

```

32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     # Django REST framework
41     'rest_framework',
42     # Customers application
43     'filterResume.apps.FilterresumeConfig',
44     # CORS
45     'corsheaders',
46 ]
47

```

Fig4.3: Installed apps

```

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',

    # CORS

```

```

'corsheaders.middleware.CorsMiddleware',
'django.middleware.common.CommonMiddleware',
]

```

```

CORS_ORIGIN_ALLOW_ALL = False

```

```

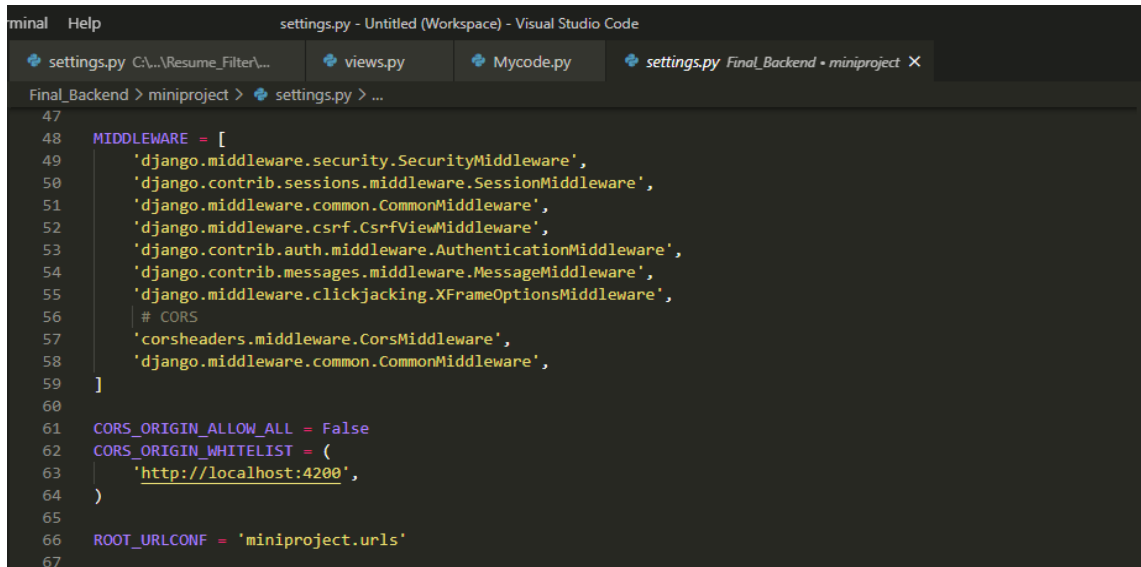
CORS_ORIGIN_WHITELIST = (
    'http://localhost:4200',
)

```

```

ROOT_URLCONF = 'miniproject.urls'

```



```

47
48 MIDDLEWARE = [
49     'django.middleware.security.SecurityMiddleware',
50     'django.contrib.sessions.middleware.SessionMiddleware',
51     'django.middleware.common.CommonMiddleware',
52     'django.middleware.csrf.CsrfViewMiddleware',
53     'django.contrib.auth.middleware.AuthenticationMiddleware',
54     'django.contrib.messages.middleware.MessageMiddleware',
55     'django.middleware.clickjacking.XFrameOptionsMiddleware',
56     # CORS
57     'corsheaders.middleware.CorsMiddleware',
58     'django.middleware.common.CommonMiddleware',
59 ]
60
61 CORS_ORIGIN_ALLOW_ALL = False
62 CORS_ORIGIN_WHITELIST = (
63     'http://localhost:4200',
64 )
65
66 ROOT_URLCONF = 'miniproject.urls'
67

```

Fig4.4: Middleware

```

TEMPLATES = [

```

```

{

```

```

    'BACKEND': 'django.template.backends.django.DjangoTemplates',

```

```

    'DIRS': [],

```

```

    'APP_DIRS': True,

```

```

'OPTIONS': {
    'context_processors': [
        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
    ],
},
}
]

```

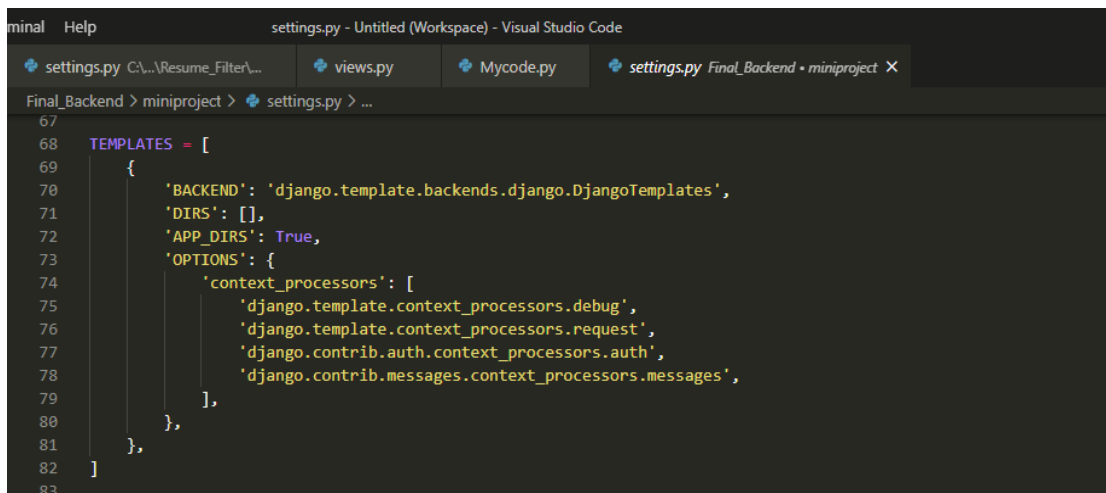


Fig4.5: Templates

WSGI_APPLICATION = 'miniproject.wsgi.application'

Database

<https://docs.djangoproject.com/en/3.1/ref/settings/#databases>

DATABASES = {

 'default': {

```

    'ENGINE': 'django.db.backends.mysql',
    'NAME': 'Resume',
    'USER': 'root',
    'PASSWORD': '1234567',
    'HOST': '127.0.0.1',
    'PORT': '3306',
}
}
# Password validation
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators

```



```

89
90 DATABASES = {
91     'default': {
92         'ENGINE': 'django.db.backends.mysql',
93         'NAME': 'resume',
94         'USER': 'root',
95         'PASSWORD': 'S@ngeetha1234',
96         'HOST': '127.0.0.1',
97         'PORT': '3306',
98     }
99 }
100
101
102 # Password validation
103 # https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators

```

Fig4.6: Database

```

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
        'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {

```

```
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

Internationalization

<https://docs.djangoproject.com/en/3.1/topics/i18n/>

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

Static files (CSS, JavaScript, Images)

<https://docs.djangoproject.com/en/3.1/howto/static-files/>

STATIC_URL = '/static/'

4.2.2 MODELS

Model class of Django contains the global declaration of variables to create the respective table in the MySQL Database.

Example code

from django.db import models

Create your models here.

class RegisterDetails(models.Model):

userName = models.CharField(max_length=70, blank=False, default="")

emailId = models.CharField(max_length=50, blank=False, default=1)

password = models.CharField(max_length=20, blank=False, default="")

class CandidateDetails(models.Model):

candidateName = models.CharField(max_length=70, blank=False, default="")

mobileNumber = models.CharField(max_length=70, blank=False, default="")

email = models.CharField(max_length=20, blank=False, default="")

resume = models.CharField(max_length=2000, blank=False, default="")

```

1  from django.db import models
2
3  # Create your models here.
4  class RegisterDetails(models.Model):
5      userName = models.CharField(max_length=70, blank=False, default='')
6      emailId = models.CharField(max_length=50, blank=False, default='')
7      password = models.CharField(max_length=20, blank=False, default='')
8
9  class CandidateDetails(models.Model):
10
11      candidateName = models.CharField(max_length=70, blank=False, default='')
12      mobileNumber = models.CharField(max_length=70, blank=False, default='')
13      email = models.CharField(max_length=20, blank=False, default='')
14      resume = models.CharField(max_length=2000, blank=False, default='')
15      match = models.BooleanField(default=False)
16      percentage = models.CharField(max_length=70, blank=False, default='')
17

```

Fig4.7: Models

4.2.3 VIEWS.PY

A view is a callable which takes a request and returns a response. This can be more than just a function, and Django provides an example of some classes which can be used as views. These allow you to structure your views and reuse code by harnessing inheritance.

Example code

```

from django.shortcuts import render
from django.http import HttpResponse
from django.http.response import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from rest_framework.parsers import JSONParser
from rest_framework import status

```

```

from filterResume.models import RegisterDetails, CandidateDetails
from filterResume.serializers import AdminSerializer, CandidateSerializer

```

```

@csrf_exempt
def register(request):
    # if request.method == 'GET':
    #     filterResume = RegisterDetails.objects.all()
    #     filterResume_serializer = FilterResumeSerializer(filterResume,
many=True)
    #     return JsonResponse(filterResume_serializer.data, safe=False)

```

```

#         # In order to serialize objects, we must set 'safe=False'

if request.method == 'POST':

    admin_data = JSONParser().parse(request)
    admin_serializer = AdminSerializer(data=admin_data)
    if admin_serializer.is_valid():
        admin_serializer.save()
        return JsonResponse(admin_serializer.data,
status=status.HTTP_201_CREATED)
    return JsonResponse(admin_serializer.errors,
status=status.HTTP_400_BAD_REQUEST)

```

4.2.4 URLS.PY

When a user makes a request for a page on your web app, **Django** controller takes over to look for the corresponding view via the **url.py** file, and then return the HTML response or a 404 not found error, if not found. In **url.py**, the most important thing is the "urlpatterns" tuple.

Example code

```

from django.conf.urls import url

from filterResume import views

urlpatterns = [

    url(r'^register/$', views.register),

    url(r'^candidateregister/$', views.candidateregister),

    url(r'^filter/$', views.filter),

].

```

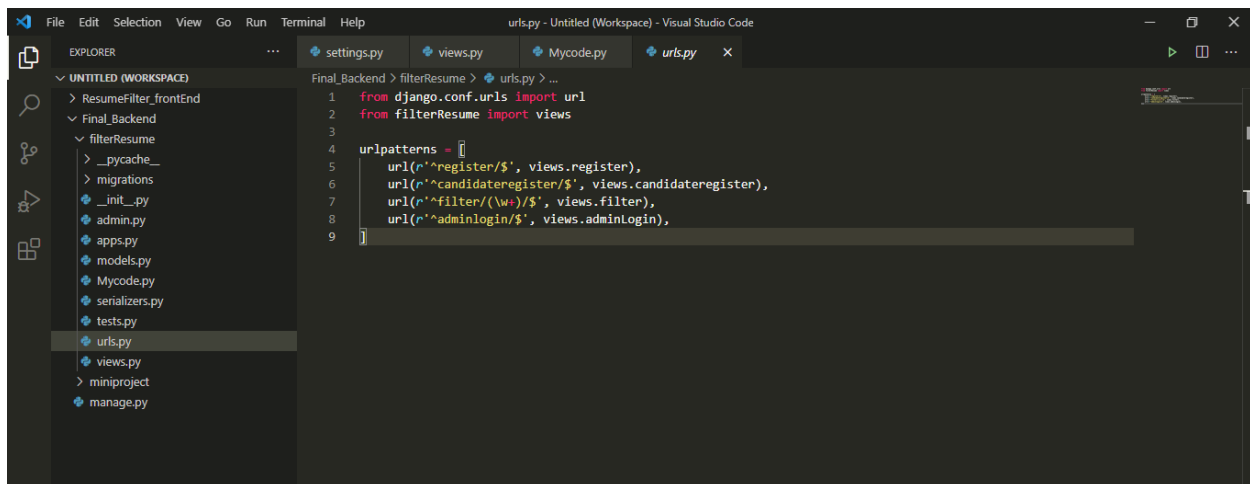


Fig4.8: URLS.PY

4.3 FRONT-END PROCESS

We need to follow some certain architecture to develop project in Angular framework. We see one-by-one.

4.3.1 MODEL

The model class of Angular contains the global declaration of variables. We can import the model class in any typescript component to access the variables.

Example code

```
export class FilterCandidateDataList
{
candidateName:string;
```

```
mobileNumber:string;  
email:string;  
resume:string;  
match:string;  
percentage:string;
```

4.3.2 COMPONENTS

Major a part of the event with Angular 7 is completed within the components. Components are basically classes that interact with the .html file of the component, which gets displayed on the browser.

Component = One .html file + One .css or .scss file + One .ts file

1. candidate-list.component.css.
2. candidate-list.component.html.
3. candidate-list.component.spec.ts.
4. candidate-list.component.ts.

4.3.3 SERVICES

Angular services are singleton objects that get instantiated just one occasion during the lifetime of an application.

They contain methods that maintain data throughout the lifetime of an application, i.e. data does not get refreshed and is available all the time. The main objective of a service is to arrange and share business logic, models, or data and functions with different components of an Angular application.

Example code

```
@Injectable()  
export class UserService {
```

```

private baseUrl1= "http://127.0.0.1:8000";
constructor(private http: HttpClient) { }
register(user: User) {
  console.log(user);
  return this.http.post(this.baseUrl1+"/register/", user);
}
passKeywords(Keywords:String):Observable<any>{
  console.log("from service",Keywords);
  return
  this.http.get<CandidateDataList[]>(`${this.baseUrl1}/filter/${Keywords}/`);
}

```

4.3.4 ROUTING

In a single-page app, you modify what the user sees by showing or hiding portions of the display that correspond to particular components, instead of going bent the server to urge a replacement page. As users perform application tasks, they have to maneuver between the various views that you've got defined. To handle the navigation from one view to the next, you use the Angular Router. The Router enables navigation by interpreting a browser URL as an instruction to vary the view.

Example code

```

export const AdminLayoutRoutes: Routes = [
  { path:'admin-signup', component: AdminSignupComponent },
  { path:'admin-signin', component: AdminSigninComponent },
  { path:'candidate-list', component: CandidateListComponent },
];

```

4.3.5 “.ts” FILE

.ts file of a component is the controller of that respective component. All the functions have been coded in the .ts file.

Example code

```
export class CandidateListComponent implements OnInit {
  constructor(private userData:UserService, private userService:
  UserService,private router: Router,) { }
  keywordForm=new FormGroup({
  filterKeyword : new FormControl()
  });
  loading = false;
  submitted = false;
  candidateList: any[];
  keyValue:any;
  Key_value:any;
  filterCandidate: any[]
  dataSource:any;
  filterDataSource:any;
  displayedColumns = ["candidateName","mobileNumber", "email",
  "resume","percentage"];
  ngOnInit() {
    this.userData.getCandidateList().subscribe(data => {
    this.candidateList = data;
    console.log("candidateList");
    console.log(this.candidateList);
    this.dataSource = new
    MatTableDataSource<CandidateDataList>(this.candidateList);
```

```

});
}
onSubmit() {
  this.keyValue = this.keywordForm.controls[
  console.log("from KeyValue", this.keyValue);
  // stop here if form is invalid
  if (this.keywordForm.invalid) {
    console.log("Error");
    return;
  }
  this.loading = true;
  this.Key_value = this.keyValue.split(
  this.userService.passKeywords(this.Key_value)
  .subscribe(
    data => {
      this.submitted = true;
      this.filterCandidate = data;
      console.log("candidateListAfterGet")
      console.log(this.filterCandidate);
      this.filterDataSource = new
      this.loading = false;
      this.router.navigate(['/candidate
    },
    error => {
      console.log(error);
      this.loading = false;

```

```
});  
}  
}
```

4.4 METHODS OR LIBRARIES USED:

In this resume filter process, the main method we use to filter the resume is pypdf2 and glob libraries.

4.4.1 PYPDF2

The pypdf2 package is a pure-python pdf library that you can use for splitting, merging, cropping, and transforming pages in your pdfs. according to the pypdf2 website, you can also use pypdf2 to add data, viewing options, and passwords to the pdfs, too. finally, you can use pypdf2 to extract text and metadata from your pdfs.

pypdf2 is actually a fork of the original pypdf, which was written by mathieu fenniak and released in 2005. however, the original pypdf's last release was in 2014. a company called phaseit, inc. spoke with mathieu and ended up sponsoring pypdf2 as a fork of pypdf.

at the time of writing, the pypdf2 package hasn't had a release since 2016. however, it is still a solid and useful package that is worth your time to learn.

Installation command for pypdf2 is **python -m pip install pypdf2**

4.4.2 GLOB

The glob module finds all the path names matching a specified pattern according to the rules used by the Unix shell, although results are returned in arbitrary order. No tilde expansion is done, but `*`, `?`, and character ranges expressed with `[]` will be correctly matched. This is done by using the `os.scandir()` and `fnmatch.fnmatch()` functions in concert, and not by actually invoking a subshell. Note that unlike `fnmatch.fnmatch()`, glob treats filenames beginning with a dot (`.`) as special cases.

(For tilde and shell variable expansion, use `os.path.expanduser()` and `os.path.expandvars()`).

4.5 MVT Pattern

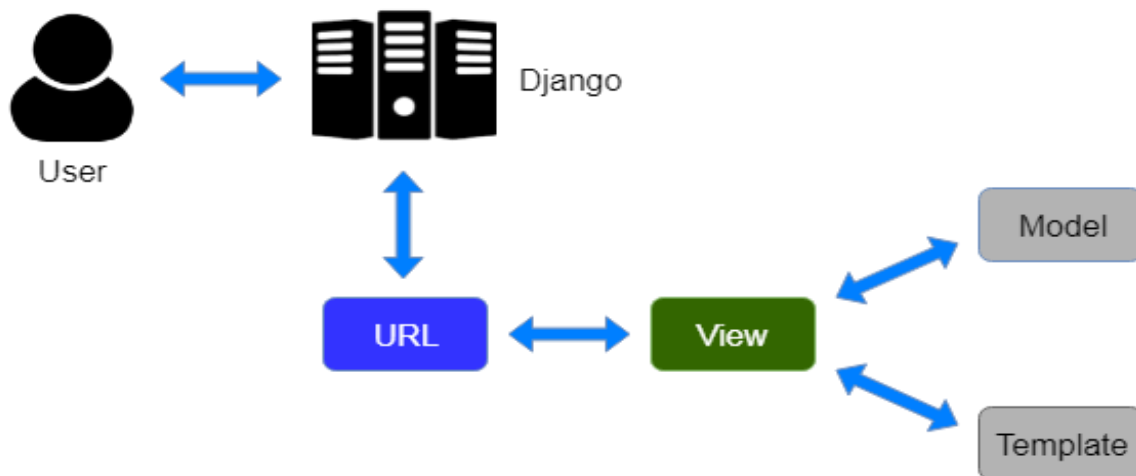


Fig4.9: MVT Flow Diagram

The MVT (Model View Template) is a software design pattern. It is a collection of three important components Model View and Template.

MODEL: Model is going to act as the interface of your data. It is responsible for maintaining data. It is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySQL, Postgres).

VIEW: The View is the user interface — what you see in your browser when you render a website. It is represented by HTML/CSS/Javascript and Jinja files.

TEMPLATE: A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

4.6 MYSQL

MySQL has stand-alone clients that allow users to interact directly with a MySQL database using SQL, but more often MySQL is used with other programs to implement applications that need relational database capability. MySQL is a component of the LAMP web application software stack, which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. MySQL is used by many database-driven web applications, including Drupal, Joomla, phpBB, and WordPress



Fig4.10: MySQL Logo

4.7 ENVIRONMENT

Most of the framework will run through Visual Studio Code so we use VS Code to run the Front-end Angular framework and for Back-end Sublime Text is used as a run-time environment.

4.7.1 SUBLIME TEXT

Sublime Text is a shareware cross-platform source code editor with a Python application programming interface. It natively supports many programming languages and markup languages, and functions can be added by users with plugins, typically community-built and maintained under free-software licenses.



Fig4.11: Sublime Text Logo

4.7.2 VISUAL STUDIO CODE

Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and mac OS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

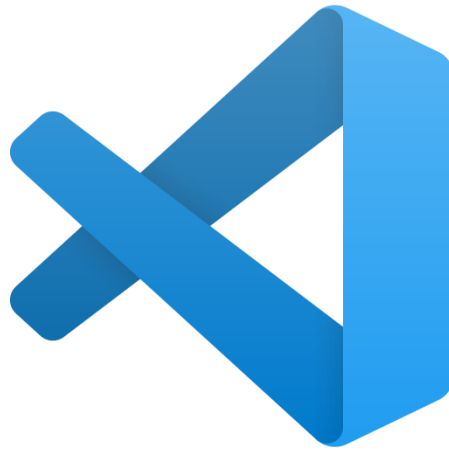


Fig4.12: Visual Studio Code Logo

CHAPTER 5

RESULTS AND PERFORMANCE ANALYSIS

5.1 SCREEN SHOTS

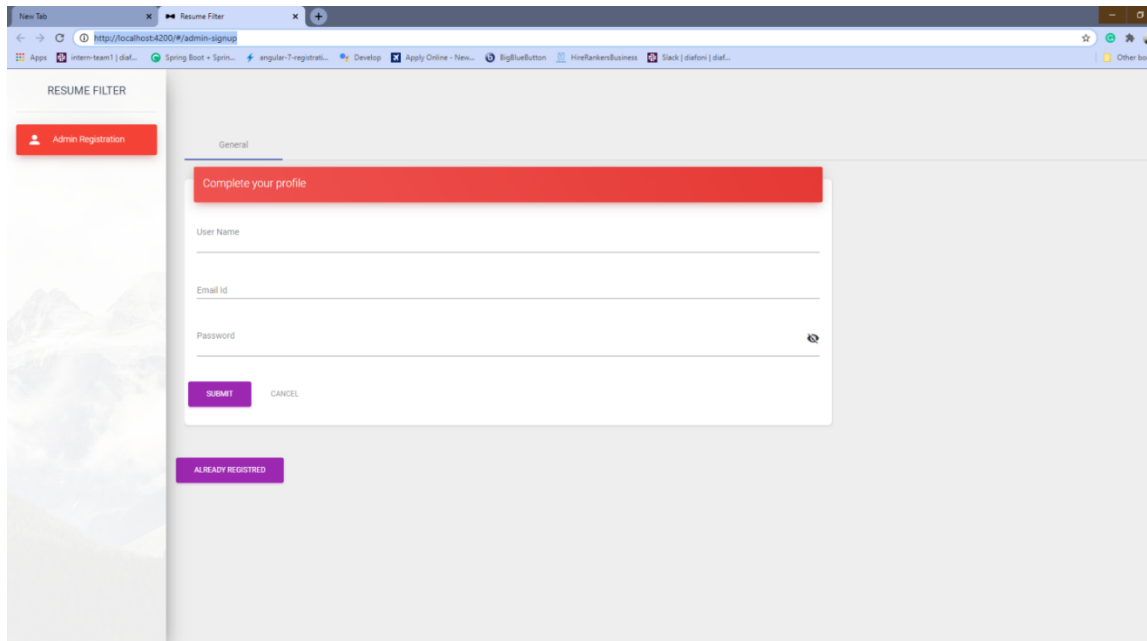


Fig5.1: Registration Page

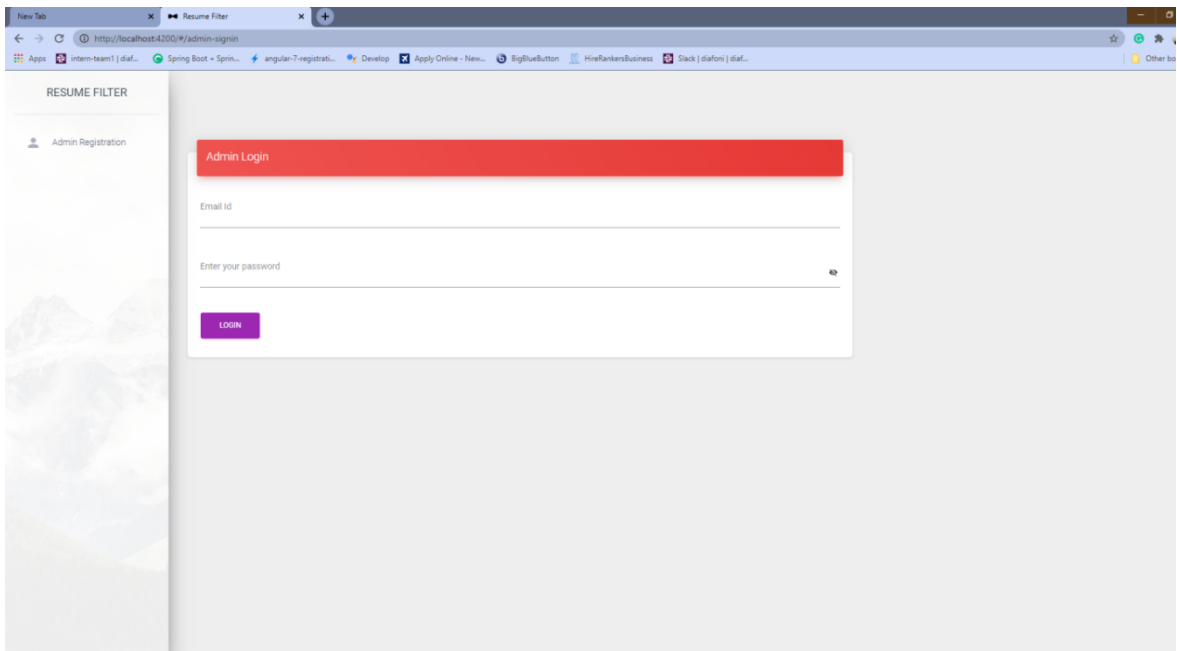


Fig5.2: Login Page

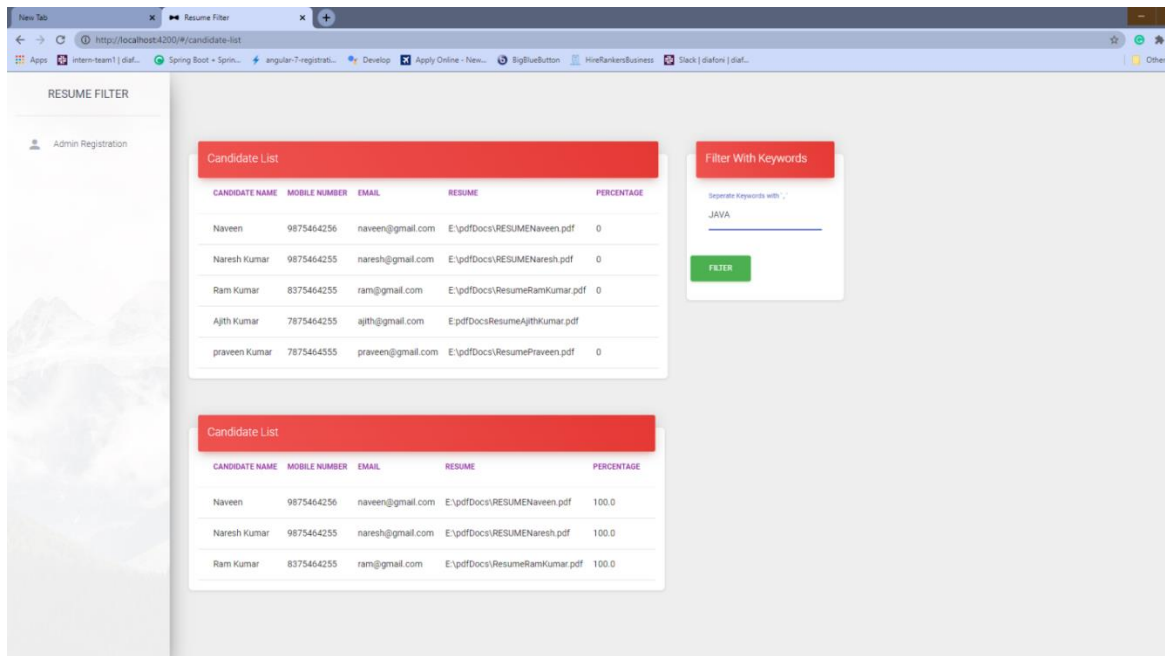


Fig5.3: Candidate List

5.2 HOW TO RUN?

5.2.1 BACK-END COMPILING AND RUNNING PROCESSES

Step1: Django REST framework works on top of Django and helps us to build RESTful Web Services flexibly. To install this package, run command: **pip install djangorestframework**.

Step2: Create Django project named **miniproject** with command: **django-admin start project mini project**.

Step3: Connect the database in “**settings.py**” with your MySQL username and password.

Step4: Run following commands to create new Django App named **filterResume**:
python manage.py startapp filterResume.

Step5: After creating model, Run initial migration for data model.

Run command: **python manage.py makemigrations filterResume.**

Step6: Run the following Python script to apply the generated migration:
python manage.py migrate filterResume.

Step7: Run **Django** server with command: **python manage.py runserver.**

After the Server started check your APIs in Postman before integrate in Front-end.

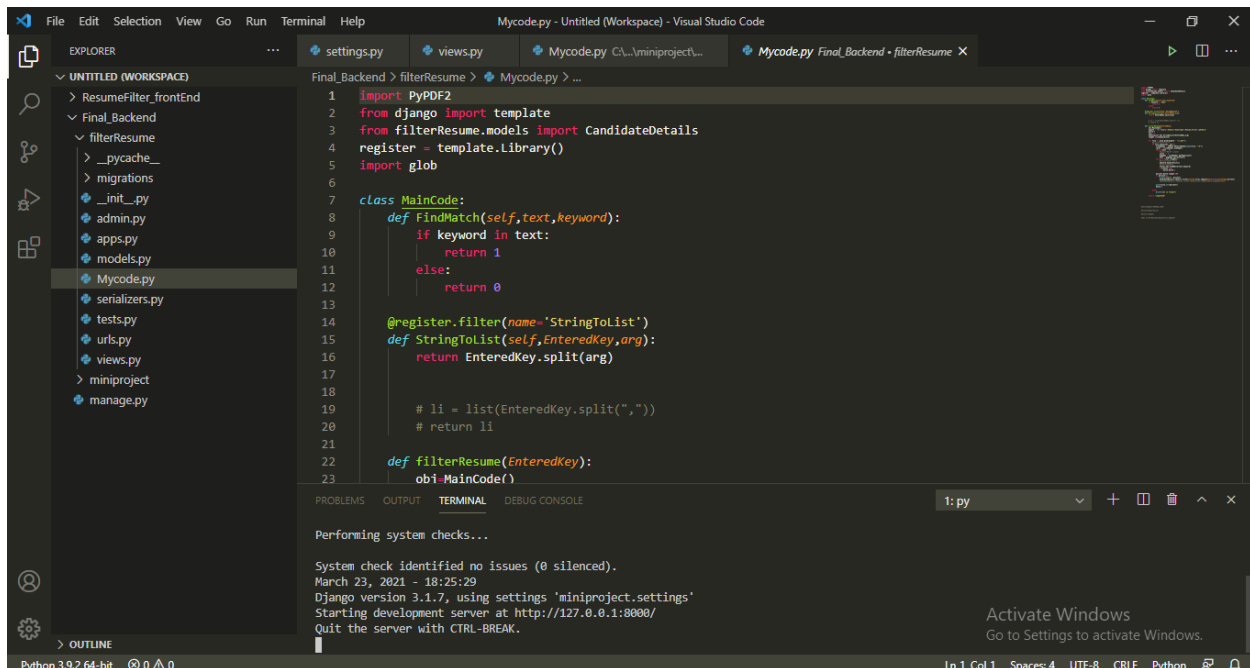


Fig5.4: After Running Back-End Process

5.2.2 FRONT-END COMPILING AND RUNNING PROCESSES

Step1: Since Angular is an open-source free framework we can download variety of designs from websites for free. After download we need extract into our local and customize as we wish.

Step2: Import the extracted project into our **Visual Studio Code**. (**File → Add Folder to Workspace → Import from the local**)

Step3: Click “**Terminal → New terminal**”.

Step4: Change the baseUrl port w.r.t Back-end server port.

Step5: In terminal, first we need to install Node Package Manager(npm) using the command “**npm install**”.

Step6: Then run the project using the command “**ng serve**”.

Step7: Check the output in the browser with the help of URL, for eg., “<http://localhost:4001>”.

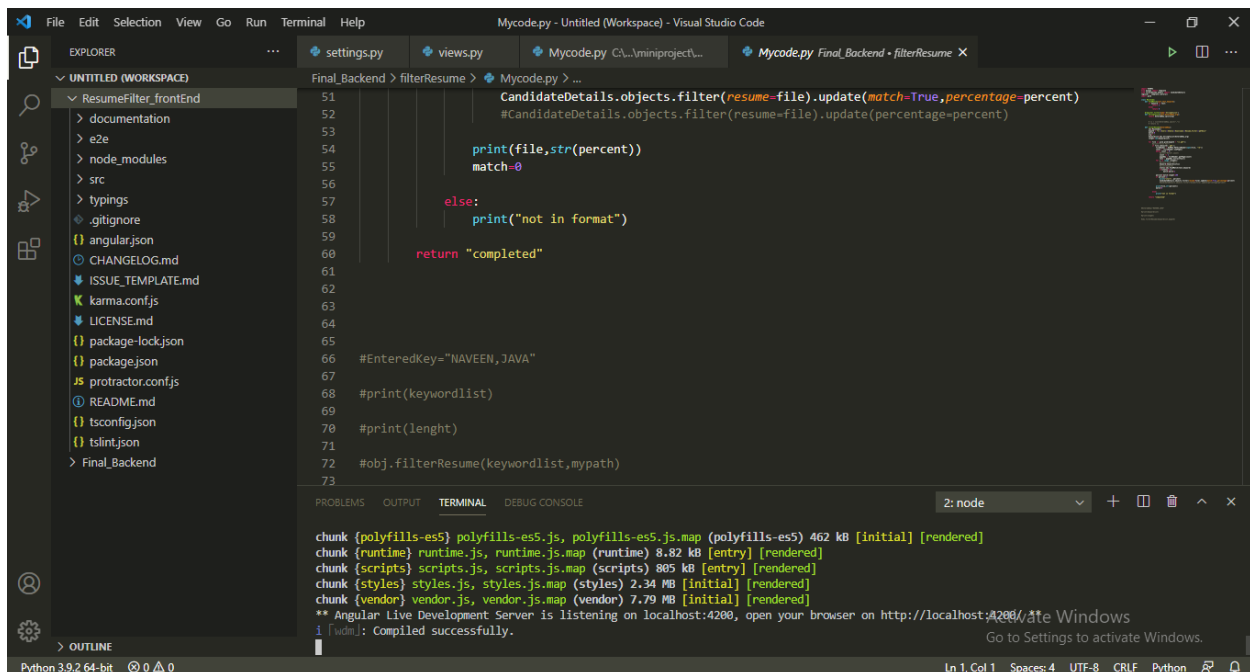


Fig5.5: Compiling Front-End Process

5.3 ADVANTAGES OF RESUME FILTER WEB APPLICATION

1. Platform independent.
2. 24/7 accessible services.
3. Ease to use.

4. Customizable.
5. Can handle large data.
6. More productive and speedy.
7. Cost-effective.
8. Flexible.

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 SUMMARY

The Resume Filter Web Application is made flexible and versatile using the Django framework and Angular framework. The application can handle large data at a time without any delay with the influence of Django. Angular allows the user to experience the multiple outcomes of the User Interface. Data has been stored in MySQL DB. Here the main method we use to filter the resume is pypdf2 and glob libraries. The pypdf2 package is a pure-python pdf library that you can use for splitting, merging, cropping, and transforming pages in your pdfs. The glob module finds all the path names matching a specified pattern according to the rules used by the Unix shell, although results are returned in arbitrary order. No tilde expansion is done, but *,?, and character ranges expressed with [] will be correctly matched. This complete process is done by using the MVT(Model View Template) pattern which is a software design pattern. It is a collection of three important components Model View and Template. Django acts as the main server in which the filter and match process is going to happen with the given pdf files. This can be achieved by giving the path where all the resumes are stored and by the help of pypdf2 and glob libraries. Angular acts as a user interface where the user or admin can register the candidate details and can get the required resumes by typing the keyword.

6.2 CONCLUSION

Resume Filter Virtual Assistant can provide tremendous benefits by improving the scalability, manageability, and performance of data and time. In this paper, the data handling techniques in Django and Angular CLI were reviewed. Furthermore, this is essential to guarantee the quality of the project and this to yield more accurate approximate results. Also, we highlighted the critical technical challenges of data handling to support and enhance the project in the future.

REFERENCE

- [1] D. Velmurugan, M. S. Sonam, S. Umamaheswari, S. Parthasarathy, K .R. Arun “A Smart Reader for Visually Impaired People Using Raspberry PI” Phil. Trans. Roy. in IJESC, ISSN 2321 3361K
- [2] W. Jia¹, L. Sun, Z. Zhong, X. Mo, G. Ma, Q. Huo. “A Robust Approach to Detecting Text from Images of Whiteboards and Handwritten Notes”, 2017, ICDAR, ISSN 2379-2140/17
- [3] S. Sonth, J. S. Kallimani. “OCR Based Facilitator for the Visually Challenged”, 2017, 2017 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques ICEECOT, ISSN 978-1-5386-2361-9/17
- [4] M. Young, “The Technical Writer’s Handbook. Mill Valley”, GHTC2016, INSPEC ACCESSION Number: 16673853. 1
- [5] C. Patel, A. Patel, D. Patel, “Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study. International Journal of Computer Applications” (0975 – 8887) Volume 55– No.10, October 2012
- [6] R. Smith, “An Overview of the Tesseract OCR Engine”, Google.com.
- [7] C. Webcam, "Logitech C525 HD Webcam, Foldable with 720p Video & Autofocus", Logitech.com, 2018. [Online]. Available: <https://www.logitech.com/en-us/product/hd-webcam-c525>. [Accessed: 02- Oct- 2018].
- [8] S. Musale and V. Ghiye, "Smart reader for visually impaired", 2018 2nd International Conference on Inventive Systems and Control (ICISC), 2018.

- [9] A. Manikandan, S. Choudhury and S. Majumder, "Text reader for visually impaired people: Any reader", 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPSCI), 2017.
- [10] S. Sabab and M. Ashmafee, "Blind Reader: An intelligent assistant for blind", 2016 19th International Conference on Computer and Information Technology (ICCIT), 2016.
- [11] [S. Sonth and J. Kallimani, "OCR based facilitator for the visually challenged", 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECOT), 2017.
- [12] A. Jain and J. Sharma, "Classification and interpretation of characters in multi-application OCR system", 2014 International Conference on Data Mining and Intelligent Computing (ICDMIC), 2014.
- [13] P. Thakare, K. Shubham, P. Ankit, R. Ajinkya and S. Om, "Interactive Reader and Recogniser System", 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018.
- [14] P. Manwatkar and S. Yadav, "Text recognition from images", 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015.
- [15] S. Karmakar, U. Zhu, "Visualizing Text Readability", 2011 2010 6th International Conference on Advanced Information Management and Service (IMS), 2010
- [16] J. Sodnik, G. Jakus and S. Tomažič, "Enhanced Synthesized Text Reader for Visually Impaired Users", 2010 Third International Conference on Advances in Computer-Human Interactions, 2010.
- [17] P. Manwatkar and K. Singh, "A technical review on text recognition from images", 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO), 2015.

APPENDIX

SOURCE CODE

```
import PyPDF2
```

```
from django import template

from filterResume.models import CandidateDetails

register = template.Library()

import glob
```

```
class MainCode:
```

```
    def FindMatch(self,text,keyword):
```

```
        if keyword in text:
```

```
            return 1
```

```
        else:
```

```
            return 0
```

```
@register.filter(name='StringToList')
```

```
def StringToList(self,EnteredKey,arg):
```

```
    return EnteredKey.split(arg)
```

```
def filterResume(EnteredKey):
```

```
    obj=MainCode()
```

```
    mypath = "C:\\Users\\Sheela\\Downloads\\Resume_Filter\\pdfDocs"
```

```
    match=0
```

```
    arg=','
```

```

keywordlist=obj.StringToList(EnteredKey,arg)

lenght=len(keywordlist)


for file in glob.glob(mypath + "\*.pdf"):

    #print("first For")

    if file.endswith('.pdf'):

        fileReader = PyPDF2.PdfFileReader(open(file, "rb"))

        count = (fileReader.numPages)

        while count > 0:

            #print("While",count)

            count -= 1

            pageObj = fileReader.getPage(count)

            text = pageObj.extractText()

        for i in range (lenght):

            #print("inner for")

            keyword=keywordlist[i]

            #print(keyword)

            status=obj.FindMatch(text,keyword)

            if(status==1):

                match=match+1


percent=match/lenght*100

```

```
if percent>1:
```

```
    print("start", percent)
```

```
CandidateDetails.objects.filter(resume=file).update(match=True,percentage=percent)
```

```
#CandidateDetails.objects.filter(resume=file).update(percentage=percent)
```

```
    print(file,str(percent))
```

```
    match=0
```

```
else:
```

```
    print("not in format")
```

```
return "completed"
```