PROJECT REPORT

SECURE JOB APPLICATION FORM USING DJANGO:

1. Introduction

The objective of this project is to develop a **secure job application form** using Django that allows applicants to submit their **name, email, phone number, resume (PDF only), and a cover letter**. The system ensures **data validation, CSRF protection, and administrative functionality** for managing applications.

2. Features

- **Applicants can:**
  - Submit name, email, phone number, resume (PDF only), and cover letter.
  - Receive confirmation upon successful submission.
- **Form Validation:**
  - Resume must be in **PDF format**.
  - Phone number must be **exactly 10 digits**.
- **Admin Panel:**
  - View all job applications.
  - Shortlist or reject candidates.
- **Security Features:**
  - CSRF protection enabled.
  - Secure file uploads.
  - Validation to prevent malicious inputs.

3. System Requirements

3.1 Software Requirements:

- Python (3.x)
- Django (latest stable version)
- SQLite / PostgreSQL (Database)
- HTML, CSS (Frontend)

3.2 Hardware Requirements:

- Processor: Intel Core i3 or higher
- RAM: 4GB or more
- Storage: Minimum 500MB free space

4. Project Implementation

4.1 Django Project Setup

1. Install Django:

   ```
   Pip install Django
   ```

2. Create a Django project and app:
3. `django-admin startproject jobapplication`
4. `cd jobapplication`
   `django-admin startapp applications`

5. Add `applications` to `INSTALLED_APPS` in `settings.py`.
6. Configure `MEDIA_URL` and `MEDIA_ROOT` to handle file uploads.

4.2 Database Model (models.py)

```
from django.db import models

class JobApplication(models.Model):
    STATUS_CHOICES = [
        ('Pending', 'Pending'),
        ('Shortlisted', 'Shortlisted'),
        ('Rejected', 'Rejected'),
    ]
    name = models.CharField(max_length=100)
    email = models.EmailField()
    phone = models.CharField(max_length=10)
    resume = models.FileField(upload_to='resumes/')
    cover_letter = models.TextField()
    status = models.CharField(max_length=20, choices=STATUS_CHOICES,
default='Pending')

    def __str__(self):
        return f"{self.name} - {self.status}"
```

4.3 Form Validation (forms.py)

```
from django import forms
from .models import JobApplication

class JobApplicationForm(forms.ModelForm):
    class Meta:
        model = JobApplication
        fields = ['name', 'email', 'phone', 'resume', 'cover_letter']

    def clean_phone(self):
        phone = self.cleaned_data.get('phone')
        if not phone.isdigit() or len(phone) != 10:
            raise forms.ValidationError("Phone number must contain exactly
10 digits.")
        return phone

    def clean_resume(self):
        resume = self.cleaned_data.get('resume')
```

```
        if resume and not resume.name.endswith('.pdf'):
            raise forms.ValidationError("Only PDF files are allowed.")
        return resume
```

4.4 View to Handle Form Submission (views.py)

```
from django.shortcuts import render, redirect
from .forms import JobApplicationForm
from django.contrib import messages

def job_application_view(request):
    if request.method == "POST":
        form = JobApplicationForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            messages.success(request, "Your application has been submitted
successfully!")
            return redirect('job_application')
    else:
        form = JobApplicationForm()
    return render(request, 'job_app/job_application_form.html', {'form':
form})
```

4.5 Admin Panel Configuration (admin.py)

```
from django.contrib import admin
from .models import JobApplication

@admin.register(JobApplication)
class JobApplicationAdmin(admin.ModelAdmin):
    list_display = ('name', 'email', 'phone', 'status')
    list_filter = ('status',)
    search_fields = ('name', 'email')
    actions = ['mark_shortlisted', 'mark_rejected']

    def mark_shortlisted(self, request, queryset):
        queryset.update(status='Shortlisted')
    mark_shortlisted.short_description = "Mark selected applications as
Shortlisted"

    def mark_rejected(self, request, queryset):
        queryset.update(status='Rejected')
    mark_rejected.short_description = "Mark selected applications as
Rejected"
```

4.6 URL Routing (urls.py)

```
from django.urls import path
from .views import job_application_view

urlpatterns = [
    path('apply/', job_application_view, name='job_application'),
]
```

4.7 Template (job_application_form.html)

```
<!DOCTYPE html>
<html lang="en">
```

```html
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Job Application</title>
</head>
<body>
    <h2>Job Application Form</h2>
    {% if messages %}
        <ul>
            {% for message in messages %}
                <li>{{ message }}</li>
            {% endfor %}
        </ul>
    {% endif %}
    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

5. Testing and Debugging

- **Unit Testing:**
    - Validate phone number input.
    - Validate resume file format.
- **Security Testing:**
    - CSRF protection is enabled.
    - Secure file upload handling.
- **Admin Functionality Testing:**
    - Ensure applications can be shortlisted or rejected.

6. Conclusion

This project successfully implements a **secure job application system** using Django, ensuring **data validation, security, and administrative functionalities**. The system can be extended with additional features like **email notifications** and **multiple job listings** in the future.

7. Future Enhancements

- Add email notifications for applicants.
- Allow applicants to check their application status.
- Integrate external job APIs for dynamic job postings.