DRESUME

by

Yash Gupta (2200320140198)

Under the guidance of

Mr. Asheesh Pandey

(Assistant Professor)(Sr. Scale)

Department of Computer Applications

ABES Engineering College

19th Km Stone, NH-09, Ghaziabad (U.P)

July, 2024

DRESUME

by

Yash Gupta (2200320140198)

Submitted to the Department of Computer Applications

in partial fulfillment of the requirements

for the degree of

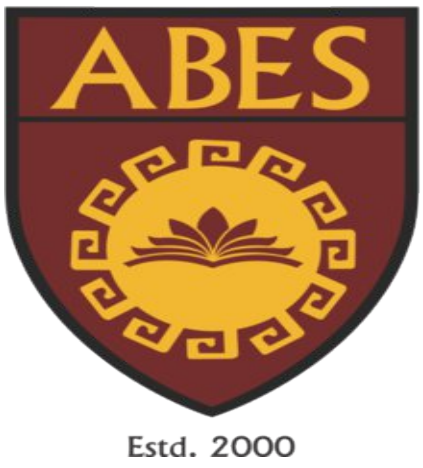
Master of Computer Application

Under the guidance of

Mr. Asheesh Pandey

(Assistant Professor) (Sr.Scale)

Department of Computer Applications



ABES Engineering College

19th Km Stone, NH-09, Ghaziabad (U.P)

July, 2024

DECLARATION

I hereby declare that the work being presented in this report entitled "**dResume**" is an authentic record of my work carried out under the supervision of "**Mr. Asheesh Pandey**"

The matter embodied in this report has not been submitted by me for the award of any other degree.

Date:

Signature of students(s)

Yash Gupta

Dept.: Computer Applications

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of HOD

Prof. (Dr.) Devendra Kumar

HOD-MCA

Dept.:-Computer Applications

Date:

Signature of Supervisor

Mr. Asheesh Pandey

Assistant Professor (Sr. Scale)

CERTIFICATE

This is to certify that Project Report entitled **“dResume”** which is submitted by **Yash Gupta** in partial fulfillment of the requirement for the award of degree **Master of Computer Application** in Department of Computer Applications of **Dr. A.P.J. Abdul Kalam Technical University,** is a record of the candidate’s own work carried out by him/them under my supervision.

The matter embodied in this Major Project Report is original and has not been submitted for the award of any other degree.

The plagiarism percentage evaluated for the content presented is

Supervisor Signature

Mr. Asheesh Pandey

Assistant Professor

(Sr. Scale)

Date:

ACKNOWLEDGEMENT

Introducing the report on the MCA project finished during MCA Final Year fills me with incredible happiness. I owe an exceptional obligation of appreciation to **Mr. Asheesh Pandey (Assistant Professor) Department of Computer Applications, ABESEC, Ghaziabad** for his/her constant support and guidance throughout the course of our work. His /Her sincerity, thoroughness, and perseverance have been a constant source of inspiration for me. It is just his perceptive endeavors that our undertakings have come around.

I also take the opportunity to recognize the contribution of **Prof. (Dr.) Devendra Kumar Head, Department of Computer Applications, ABESEC, Ghaziabad** for his support and assistance during the development of the project.

I also don't want to miss the opportunity to thank the entire department's faculty members for their helpful support and cooperation during the project's development. Last but not least, we want to thank our friends for their help in getting the project done.

Signature of Student(s)

Yash Gupta
2200320140198

ABSTRACT

The current system creates resumes manually either using Word or Google doc, so there are some issues with the current system. The existing system suffered from a number of shortcomings. Since the entire system had to be maintained, the process of storing, maintaining and retrieving information was very tedious and time-consuming. The records were never in a systematic order, there used to be a lot of difficulty in assigning any particular transaction to a particular context. If any information was to be found, it was required to go through various registers, the documents would never exist. something like report generation There would always be wasted time entering records and retrieving records Another problem was that it was very difficult to find errors in entering records Once the records were inserted it was very difficult to update those records.

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

*	Multiplication Symbol
≠	Not Equal
<	Smaller than

LIST OF ABBREVIATIONS

AWS	Amazon Web Service
S3	Simple Storage Service
RDS	Relational Database Service
IAM	Identity and Access Management
EC2	Elastic Compute Cloud

CHAPTER 1
INTRODUCTION

1.1 PROBLEM DEFINITION

A resume is the first meeting between you and a potential employer, more often than ever. So how do you want to be remembered? Wrinkled and messy. Neat and structured. Long and boring. Accurate and interesting. Companies do not have time to interview every applicant who is interested in the job. If they did, there would be no company to work for. They use a process of elimination. That's right - he continues. When a job seeker wants to apply for a job online, they generally need to attach their resume to an email. The online resume builder system provides users with popular resume formats and a better way to showcase their resumes to employers. A job seeker does not need to attach a resume to every email, just provide the URL of your resume and the employer can view the resume online by clicking on the link and also download it.

1.2 OBJECTIVE

The Objectives of Online Resume Builder are -

Simplify the resume creation process: Provide a user-friendly interface to create a professional-looking resume in a short amount of time.

Offer customization options: Allow users to tailor their resume to specific job openings or industries by providing a range of templates, formatting options, and content suggestions.

Improve job search efficiency: Help users create a strong, attention-grabbing resume that increases their chances of getting noticed by potential employers and recruiters.

Enhance job search visibility: Assist users in creating a resume that is optimized for applicant tracking systems (ATS) and search engines, making it more likely to appear in search results.

Save time and effort: Automate the resume creation process, allowing users to focus on other aspects of their job search, such as networking and interviewing.

Improve user experience: Make the resume creation process enjoyable and engaging, with features such as drag-and-drop interfaces, real-time feedback, and easy editing capabilities.

1.3 SCOPE OF PROJECT

Online Resume Builder can be used in accordance with customer requirements. Customers can customize theirs resumes with a selection of topics and details. Services are It's hard to be beaten by competitors because the system is giving customers exactly what they want.

1.4 NEED

The current system creates resumes manually using either Word or Google doc, so there are some issues with the current system. The existing system suffered from a number of shortcomings. Since the entire system had to be maintained, the process of storing, maintaining and retrieving information was very tedious and time-consuming. The records were never in a systematic order, there used to be a lot of difficulty in assigning any particular transaction to a particular context. If any information was to be found, it was required to go through various registers, the documents would never exist. something like generating reports There would always be wasted time entering records and retrieving records Another problem was that it was very difficult to find errors in entering records Once the records were entered it was very difficult to update those records.

CHAPTER 2

LITERATURE REVIEW

A. Manual Creation of Resume

Long Back resumes were created manually by using MS Word and the format at that time was quite different. These are the fields that applicants used to mention in their resumes, in personal information applicants used to write their name, address and phone number. Applicants also used to mention their personal opinion in their resume. That time applicants used to mention their both early and recent education details. Applicants used to create either too short or too long resumes because at that time there were no standards set for ideal length of resumes.

B. Inference Drawn From Conclusion

Some inferences are drawn from literature review and research regarding what should be the length of a resume? What information should be included in a resume? What information should be avoided while writing a resume? Ideally in personal information name, address, phone number, email address should be included and height, weight, religion, birth date, marital status should be avoided. Resumes of one page length are preferred more over resumes of two page length.

CHAPTER 3

FEASIBILITY STUDY

3.1 Purpose

The feasibility study serves as a critical assessment tool to determine the viability and potential success of DRESUME. By evaluating various aspects of the project, including technical, economic, and operational factors, the study aims to provide insights into the feasibility of developing and implementing the platform. The key purposes of the feasibility study are as follows:

Understanding Objectives: The study seeks to gain a comprehensive understanding of DRESUME's objectives, including its intended functionalities, target users, and expected outcomes. By clarifying the project's goals, the feasibility study sets the foundation for subsequent assessments.

Assessing Potential Challenges: Identifying potential challenges and obstacles that may arise during the development and deployment of DRESME is crucial. This includes technical complexities, market competition, regulatory compliance issues, and resource constraints. By anticipating these challenges early on, the feasibility study enables stakeholders to devise strategies to mitigate risks effectively.

Evaluating Expected Outcomes: The feasibility study aims to evaluate the expected outcomes and benefits of DRESUME upon its successful implementation. This involves assessing the potential impact on users, such as improved access to housing options, enhanced user experience, and increased efficiency in the accommodation search process.

3.2 Scope

The feasibility study encompasses a comprehensive assessment of various factors to determine the feasibility of DRESUME. It covers the following key areas:

Technical Feasibility: Technical feasibility is concerned with the availability of hardware and software required for the development of the system, to see compatibility and maturity of the technology proposed to be used and to see the availability of the required technical manpower to develop the system. After the study we came to conclusion that we proceed further with the tools and development environment chosen by us. This was important in our case as we were working on two various phases of the department that will need to be integrated in future to make an extendedsystem.

Economic Feasibility: Economic feasibility involves estimating the costs associated with developing, deploying, and maintaining the DRESUME platform. This includes expenses related to software development, infrastructure setup, marketing, staffing, and ongoing operational costs. The study also considers potential revenue streams, such as subscription fees, advertisements, or transaction commissions, to determine the platform's financial viability. By comparing the projected costs and benefits, stakeholders can assess the economic feasibility of DRESUME and make informed decisions regarding investment.

Operational Feasibility: Operational feasibility focuses on evaluating whether DRESUME can be successfully operated and maintained in the long term. It considers factors such as user acceptance, usability, support requirements, regulatory compliance, and scalability. The study assesses the platform's ability to meet user needs effectively, handle increasing user loads, and adapt to evolving market conditions. By addressing operational considerations upfront, stakeholders can ensure that DRESUME remains sustainable and competitive in the long run.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 FUNCTIONAL REQUIREMENTS

User Registration and Authentication:

- Users should be able to create accounts with a valid Username and password.
- Secure authentication mechanisms should be implemented to protect user accounts.

Portfolio:

- Users can add and view their reviews.

Blog:

- Users can create and edit their blog posts.

Contact:

- Users can contact the person by filling out the contact form.

Key Skills:

- User can showcase their key skills and edit them.

Coding Skills:

- User can show and edit their coding skills.

4.2 NON – FUNCTIONAL REQUIREMENTS

Performance:

- A website should serve a specified number of concurrent users without degrading performance.

Security:

- User data must be stored securely.

Usability:

- Websites should have an intuitive and responsive design for a positive user experience.

4.3 Hardware Requirements

The hardware requirements specify the minimum hardware specifications that the server hosting the DRESUME platform should meet to ensure optimal performance. These requirements include:

Minimum 8GB RAM to ensure sufficient memory for handling user requests and database operations.

500GB Hard Disk to accommodate the storage of property listings, user data, and other platform-related files.

Intel Core i5 Processor or equivalent to provide adequate processing power for handling concurrent user requests and maintaining system responsiveness.

4.4 Software Requirements

The software requirements outline the necessary software components and technologies that the DRESUME platform relies on for its development and operation. These requirements include:

Operating System: DRESUME should be compatible with popular operating systems such as Windows, macOS, and Linux to ensure broad accessibility for users.

Backend: The backend of the platform should be built using Node.js and Express.js, which provide a robust and scalable framework for developing server-side applications.

Frontend: The frontend of DRESUME should be developed using React.js, a powerful JavaScript library for building dynamic user interfaces.

Database: MongoDB, a NoSQL database, should be used to store property listings, user profiles, and other platform-related data due to its flexibility and scalability.

Other Tools: Redux should be used for state management, while JWT (JSON Web Tokens) should be employed for user authentication to ensure secure and reliable user authentication and authorization mechanisms.

4.5 USE CASES

User Registration and Authentication:

- Users should be able to create accounts with a valid Username and password.
- Secure authentication mechanisms should be implemented to protect user accounts.

Portfolio:

- Users can add and show their Reviews..

Blog:

- Users can create and edit their blog posts.

Contact:

- Users can contact the person by filling out the contact form.

Key Skills:

- User can showcase their key skills and edit them.

Coding Skills:

- User can show and edit their coding skills.

Performance:

- A website should serve a specified number of concurrent users without degrading performance.

Security:

- User Details must be stored Safely.

Usability:

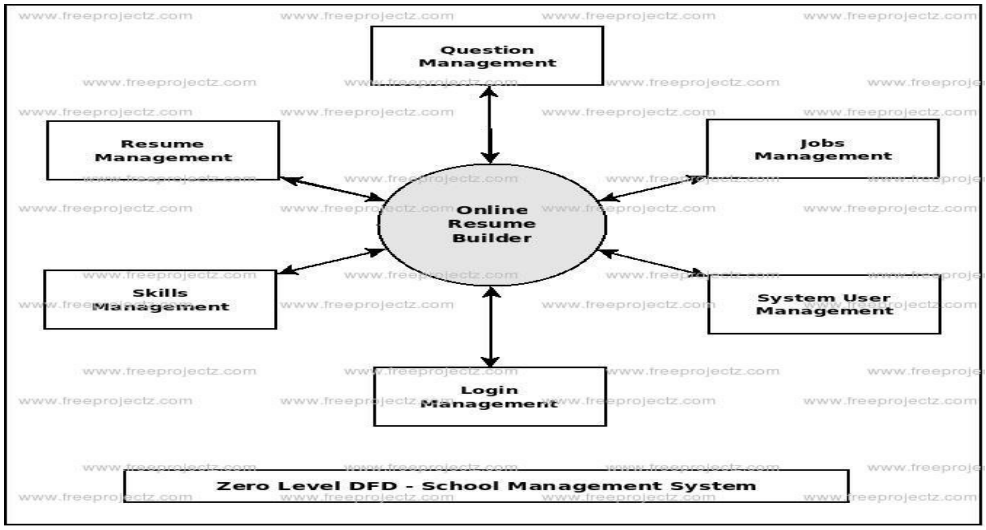
- The website should have an intuitive and responsive design for a positive user experience.

CHAPTER 5

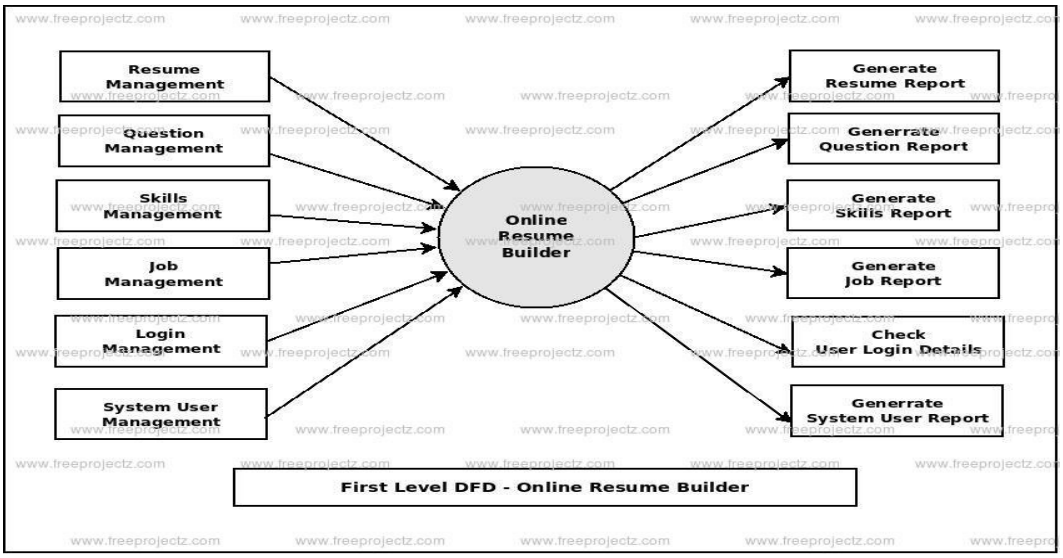
SYSTEM DESIGN

5.2 DATA FLOW DIAGRAM(DFD)

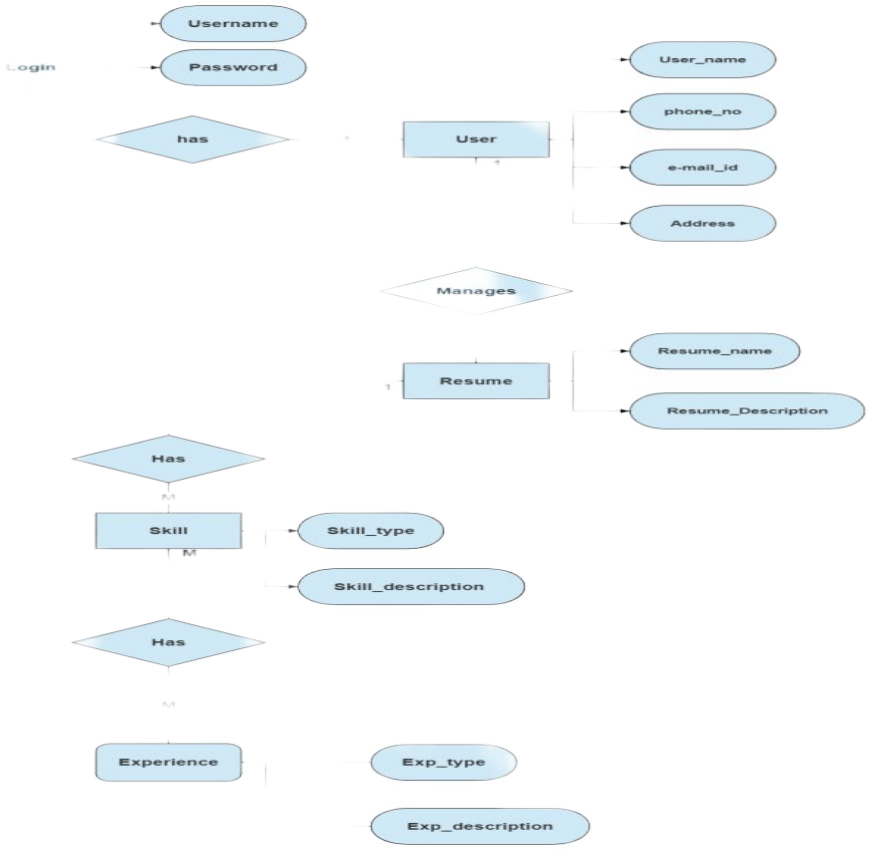
Zero Level DFD



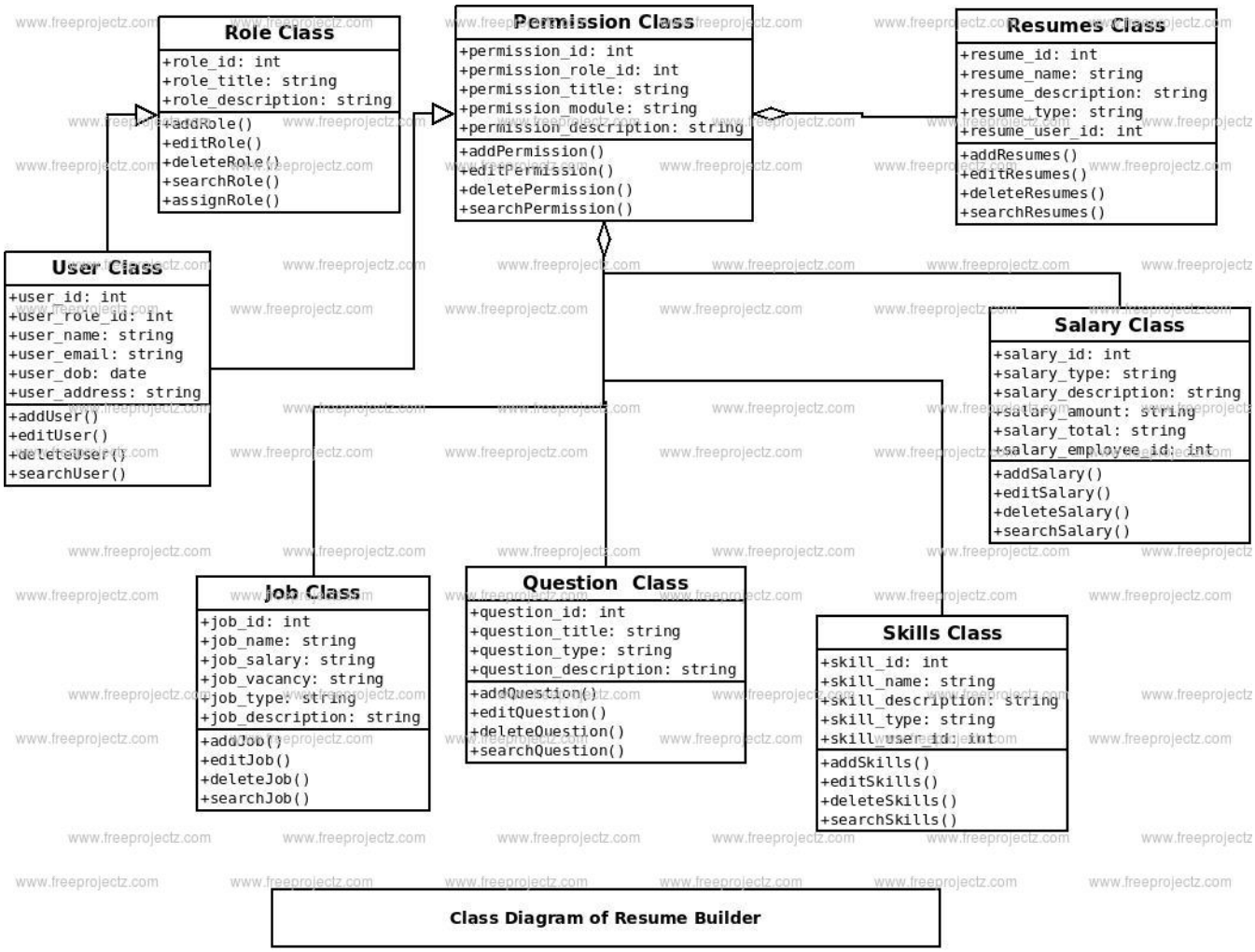
First Level DFD



5.3 ER-DIAGRAM

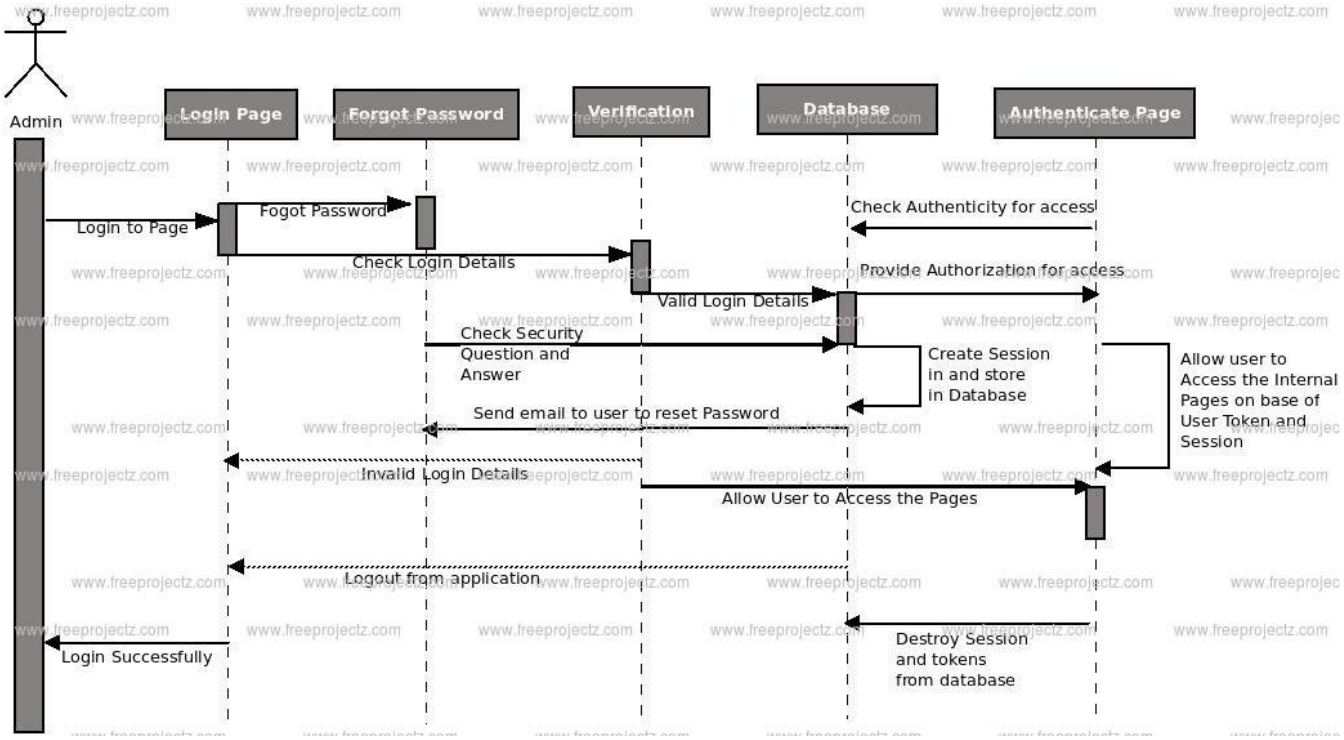


5.4 CLASS DIAGRAM

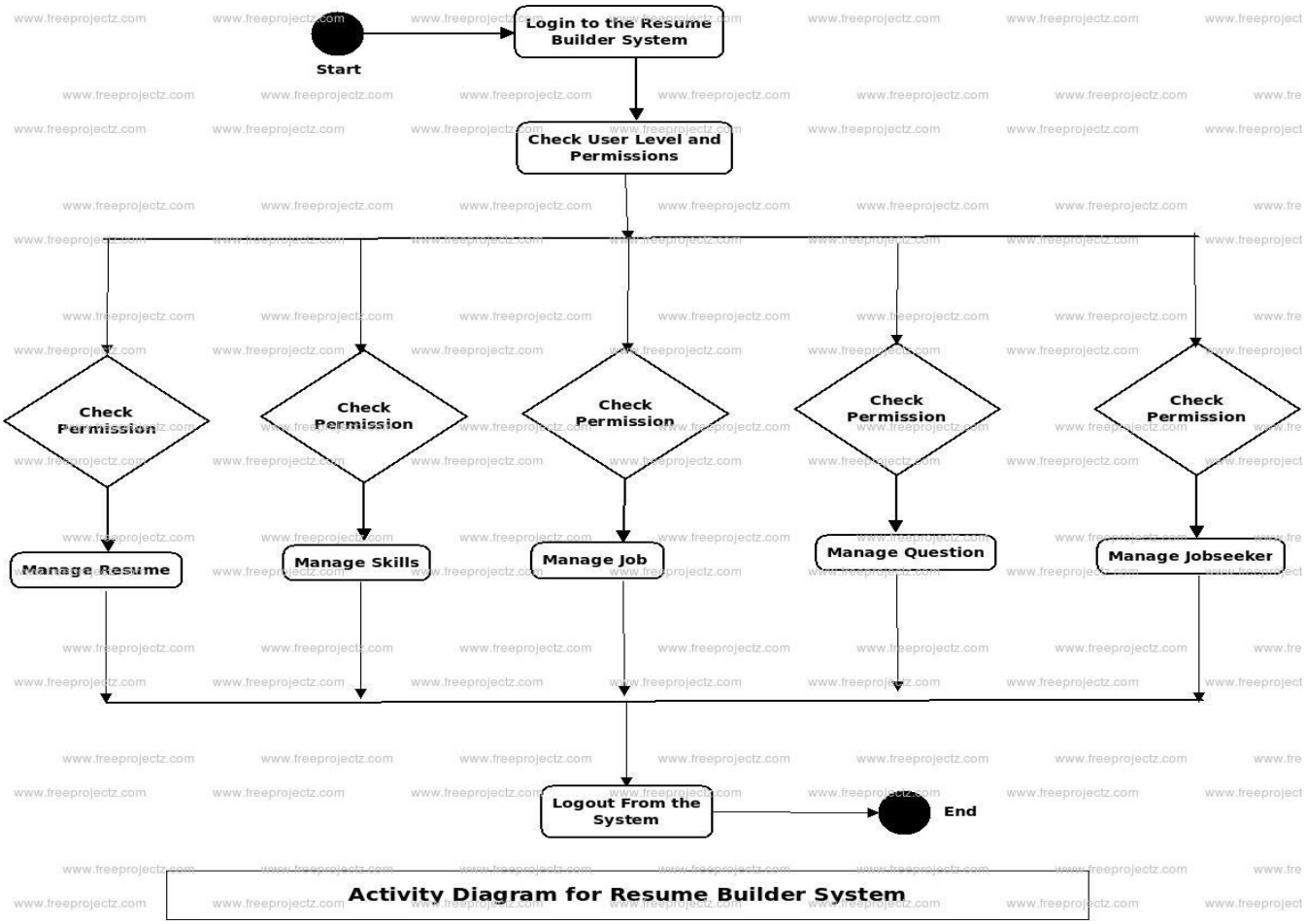


Class Diagram of Resume Builder

5.5 SEQUENCE DIAGRAM



5.6 ACTIVITY DIAGRAM

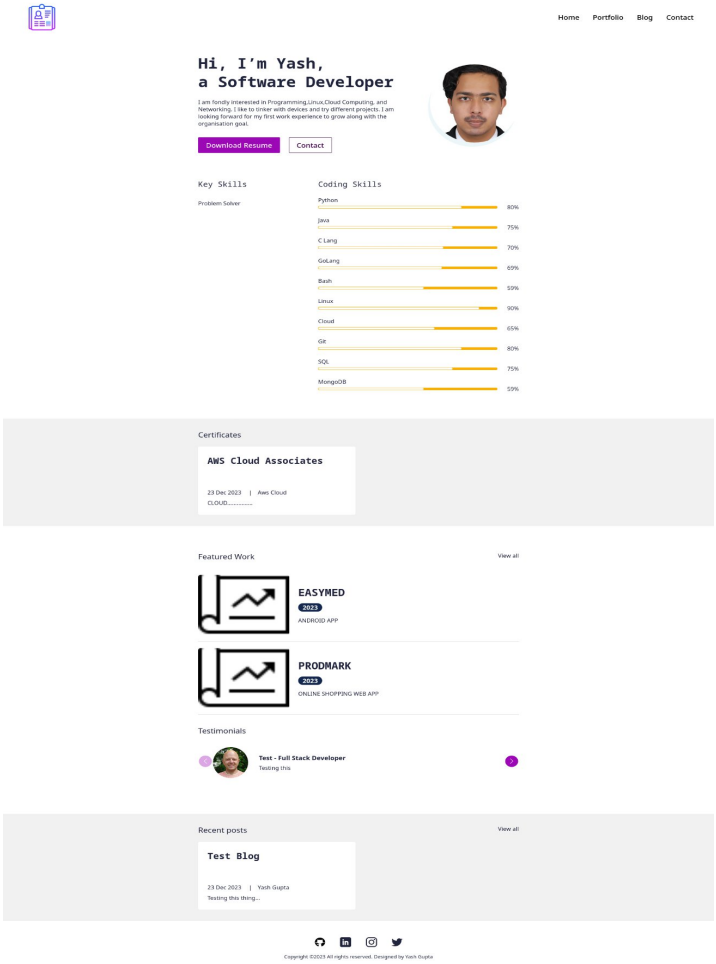


Activity Diagram for Resume Builder System

CHAPTER 6

GUI

Modules Screenshots



Home Portfolio Blog Contact

See my recent
projects below


Home Portfolio Blog Contact

See my recent
blogs below




Home Portfolio Blog Contact

Test Blog

Yash Gupta 23 Dec 2023

Some text to fill this up.....

Testing again.....

bye.....

Home Portfolio Blog Contact

Contact us below

Name

*Full name..

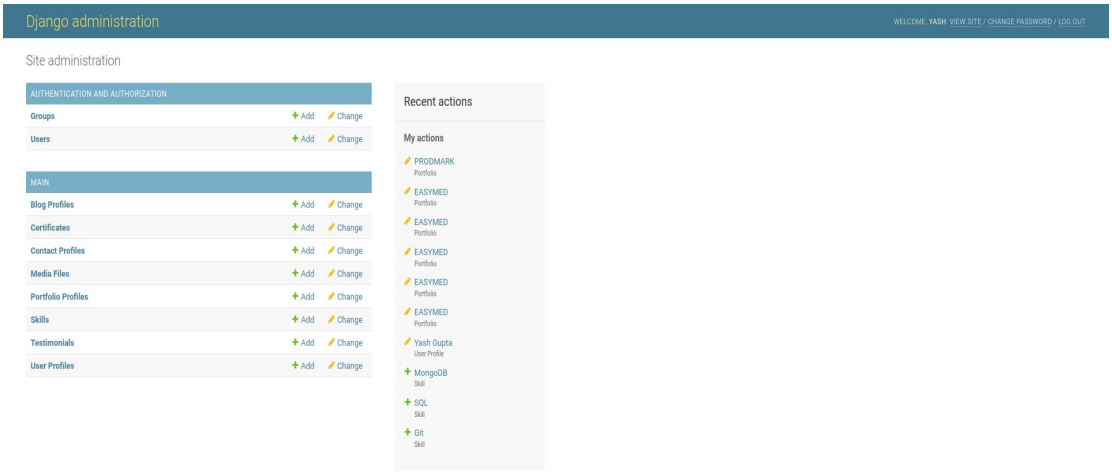
Email

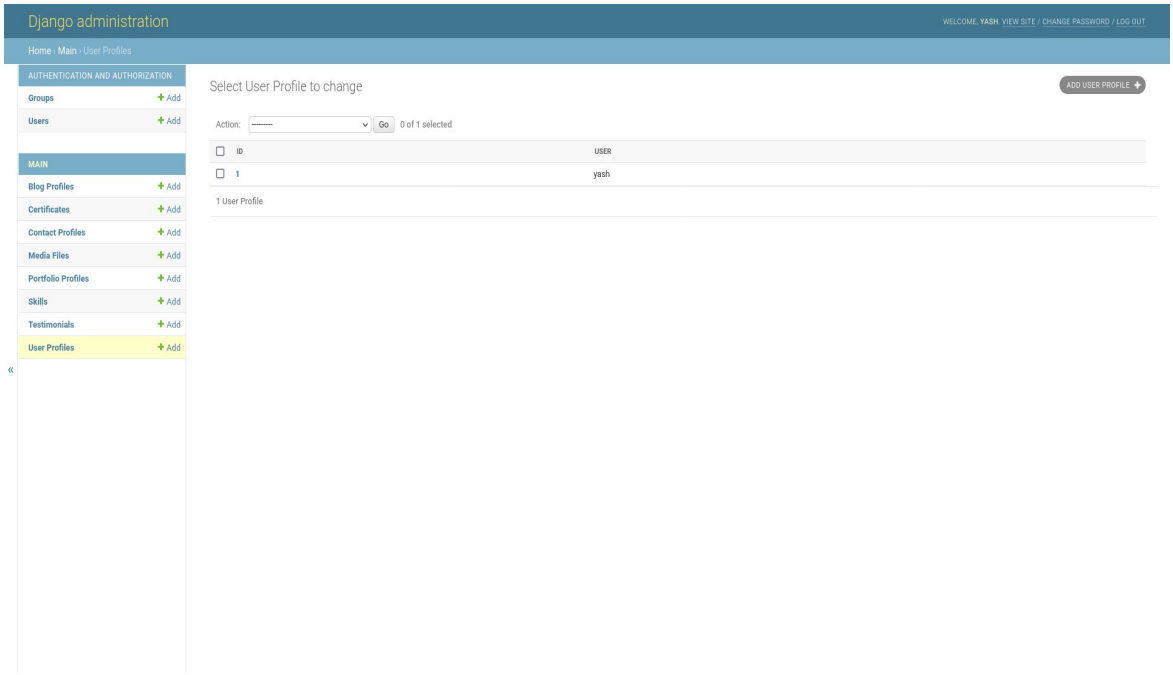
*Email..

Message

*Message..

Submit


Django administration
WELCOME, YASH VIEW SITE / CHANGE PASSWORD / LOG OUT
Site administration
AUTHENTICATION AND AUTHORIZATION
Groups + Add Change
Users + Add Change
MAIN
Blog Profiles + Add Change
Certificates + Add Change
Contact Profiles + Add Change
Media Files + Add Change
Portfolio Profiles + Add Change
Skills + Add Change
Testimonials + Add Change
User Profiles + Add Change
Recent actions
My actions
PRODMARK Portfolio
EASYMED Portfolio
EASYMED Portfolio
EASYMED Portfolio
EASYMED Portfolio
EASYMED Portfolio
Yash Gupta User Profile
MongoDB Skill
SQL Skill
Git Skill


Django administration
WELCOME, YASH VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Main › User Profiles
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
MAIN
Blog Profiles + Add
Certificates + Add
Contact Profiles + Add
Media Files + Add
Portfolio Profiles + Add
Skills + Add
Testimonials + Add
User Profiles + Add
Select User Profile to change
ADD USER PROFILE +
Action: ---------- Go 0 of 1 selected
ID
USER
1
yash
1 User Profile

CODING

HTML

Base.html

```
{% load static %}
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>{% block title %}{%endblock %}</title>
    <meta name="author" content="Did Coding Limited & James Granger Design">
    <link rel="canonical" href="{{request.path}}"/>
    <link rel="home" href="{% url 'main:home' %}"/>
    <meta name="description" content="{% block description %}{% endblock %}">
    <meta name="keywords" content="{% block keywords %}{% endblock %}">
    <link rel="shortcut icon" type="image/x-icon" href="{% static 'images/icon.jpg' %}">
    <link rel="apple-touch-icon" type="image/jpg" href="{% static 'images/icon.jpg' %}">
    <!-- ================================
    Start CSS
    ================================ -->
    <link href="{% static 'css/bootstrap.min.css' %}" rel="stylesheet">
    <link rel="stylesheet" href="https://unpkg.com/swiper@7.0.5/swiper-bundle.min.css">
    <link href="{% static 'css/style.css' %}" rel="stylesheet">
    {% block extend_header%}{%endblock%}
```

```
    <!-- ================================
    End CSS
    ================================ -->
  </head>
  <body>
    {% include 'main/partials/messages.html' %}
    {% include 'main/partials/nav.html' %}
    {% block content %}
    {% endblock %}
    {% include 'main/partials/footer.html' %}
    <!-- ================================
    Start Scripts
    ================================ -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script src="{% static 'js/bootstrap.bundle.min.js' %}"></script>
    <script src="https://unpkg.com/swiper@7.0.5/swiper-bundle.min.js"></script>
    <script src="{% static 'js/script.js' %}"></script>
    {% block extend_footer %}{%endblock%}
    <!-- ================================
    End Scripts
    ================================ -->
  </body>
</html>
```

Blog.html

```
{% extends 'main/base.html' %}
{% load static %}
```

```
<!-- ================================
Start SEO blocks
================================ -->
{% block title %}{% endblock %}
{% block decription %}{% endblock %}
{% block keywords %}{% endblock %}
<!-- ================================
END SEO blocks
================================ -->

<!-- ================================
Start CSS blocks
================================ -->
{% block extend_header %}{% endblock %}
<!-- ================================
END CSS blocks
================================ -->

<!-- ================================
Start script blocks
================================ -->
{% block extend_footer %}{% endblock %}
<!-- ================================
END script blocks
================================ -->

<!-- ================================
Start Content
```

```
================================== -->
{% block content %}
<section>
  <div class="innerPageBannerCol">
    <div class="container">
      <div class="row g-4 g-md-3  align-items-center">
        <div class="col-md-6">
          <div class="bannerContent">
            <h1 class="xlTitle pb-md-3">See my recent blogs below</h1>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>

<section>
  <div class="lightBg">
    <div class="container">
      <div class="portfolioContentMain">
        <div class="row g-3 g-md-4 g-lg-5 portfolioRow">
          {% for obj in object_list %}
          <div class="col-md-6 pColMain">
            <div class="pCol">
              <a href="{% url 'main:blog' slug=obj.slug %}" >
              <img src="{{obj.image.url}}" alt="..." class="pImg">
              </a>
            </div>
```

```
          </div>
          {% endfor %}


        </div>
      </div>
    </div>
  </div>
</section>
<!-- ================================
End Content
================================= -->
{%endblock%}
```

Portfolio.html

```
{% extends 'main/base.html' %}
{% load static %}

<!-- ================================
Start SEO blocks
================================= -->
{% block title %}{% endblock %}
{% block decription %}{% endblock %}
{% block keywords %}{% endblock %}
<!-- ================================
END SEO blocks
================================= -->

<!-- ================================
```

Start CSS blocks

================================== -->

{% block extend_header %}{% endblock %}

<!-- ==================================

END CSS blocks

================================== -->

<!-- ==================================

Start script blocks

================================== -->

{% block extend_footer %}{% endblock %}

<!-- ==================================

END script blocks

================================== -->

<!-- ==================================

Start Content

================================== -->

{% block content %}

<section>

            <h1 class="xlTitle pb-md-3">See my recent projects below</h1>

```
    </div>
   </div>
  </div>
</section>


<section>
  <div class="lightBg">
   <div class="container">
    <div class="portfolioContentMain">
     <div class="row g-3 g-md-4 g-lg-5 portfolioRow">
      {% for obj in object_list %}
       <div class="col-md-6 pColMain">
        <div class="pCol">
         <a href="{% url 'main:portfolio' slug=obj.slug %}" >
         <img src="{{obj.image.url}}" alt="..." class="pImg" >
         </a>
        </div>
       </div>
      {% endfor %}

     </div>
    </div>
   </div>
  </div>
</section>
<!-- ================================
End Content
================================ -->
```

{% endblock%}

Base.html

```
{% load static %}
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>{% block title %}{%endblock %}</title>
    <meta name="author" content="Did Coding Limited & James Granger Design">
    <link rel="canonical" href="{{request.path}}"/>
    <link rel="home" href="{% url 'main:home' %}"/>
    <meta name="description" content="{% block description %}{% endblock %}">
    <meta name="keywords" content="{% block keywords %}{% endblock %}">

    <link rel="shortcut icon" type="image/x-icon" href="{% static 'images/icon.jpg' %}">
    <link rel="apple-touch-icon" type="image/jpg" href="{% static 'images/icon.jpg' %}">

    <!-- ================================
    Start CSS
    ================================ -->
    <link href="{% static 'css/bootstrap.min.css' %}" rel="stylesheet">
    <link rel="stylesheet" href="https://unpkg.com/swiper@7.0.5/swiper-bundle.min.css">
    <link href="{% static 'css/style.css' %}" rel="stylesheet">
    {% block extend_header%}{%endblock%}
```

```
    <!-- ================================
    End CSS
    ================================ -->


  </head>
  <body>
    {% include 'main/partials/messages.html' %}
    {% include 'main/partials/nav.html' %}

    {% block content %}
    {% endblock %}

    {% include 'main/partials/footer.html' %}

    <!-- ================================
    Start Scripts
    ================================ -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script src="{% static 'js/bootstrap.bundle.min.js' %}"></script>
    <script src="https://unpkg.com/swiper@7.0.5/swiper-bundle.min.js"></script>
    <script src="{% static 'js/script.js' %}"></script>
    {% block extend_footer %}{%endblock%}
    <!-- ================================
    End Scripts
    ================================ -->
  </body>
</html>
```

Nav.html

```
{% load static %}

<!-- ================================

Start Navigation Bar

================================ -->

<header>

   <div class="headerCol">

   <div class="container-fluid">

      <div class="row align-items-center">

      <div class="col-auto">

         <div class="logoCol"><a href="{% url 'main:home' %}"><img src="{% static
'images/icons8-resume-100.png' %}" alt="..."></a></div>

      </div>

      <div class="col">

         <div class="d-md-none">

         <button class="navToggle">

            <span class="navToggle__text">Toggle Menu</span>

         </button>

         </div>

         <div class="navCollapseCol">

         <div class="navCol">

            <ul>

            <li><a href="{%url 'main:home'%}">Home</a></li>

            <li><a href="{%url 'main:portfolios' %}">Portfolio</a></li>

            <li><a href="{% url 'main:blogs' %}">Blog</a></li>

            <li><a href="{%url 'main:contact'%}">Contact</a></li>

            </ul>

         </div>
```

```
            </div>
        </div>
        </div>
    </div>
    </div>
</header>
<!-- ================================
End Navigation Bar
================================ -->
```

Footer.html

```
{% load static %}
<!-- ================================
Start Footer
================================ -->
<footer>
    <div class="footerCol">
    <div class="container">
        <ul class="socialCol">
        <li><a href="https://github.com/techyyash/"><img src="{% static 'images/github.svg' %}" alt="..."></a></li>
        <li><a href="https://www.linkedin.com/in/yash-gupta-03573a119/"><img src="{% static 'images/linkedin.svg' %}" alt="..."></a></li>
        <li><a href="https://instagram.com"><img src="{% static 'images/insta.svg' %}" alt="..."></a></li>
        <li><a href="https://twitter.com"><img src="{% static 'images/twitter.svg' %}" alt="..."></a></li>
```

```
        </ul>

        <div class="copyrightCol">

        <p>Copyright ©{% now 'Y' %} All rights reserved. Designed by <a
href="https://github.com/techyyash/" target="_blank">Yash Gupta</a></p>

        </div>

    </div>

    </div>

</footer>

<!-- ================================

End Footer

================================ -->
```

0001_initial.py

```
import ckeditor.fields

from django.conf import settings

from django.db import migrations, models

import django.db.models.deletion



class Migration(migrations.Migration):


    initial = True


    dependencies = [

        migrations.swappable_dependency(settings.AUTH_USER_MODEL),

    ]
```

```
    operations = [
        migrations.CreateModel(
            name='Certificate',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('date', models.DateTimeField(blank=True, null=True)),
                ('name', models.CharField(blank=True, max_length=50, null=True)),
                ('title', models.CharField(blank=True, max_length=200, null=True)),
                ('description', models.CharField(blank=True, max_length=500, null=True)),
            ],
            options={
                'verbose_name': 'Certificate',
                'verbose_name_plural': 'Certificates',
            },
        ),
        migrations.CreateModel(
            name='ContactProfile',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('timestamp', models.DateTimeField(auto_now_add=True)),
                ('name', models.CharField(max_length=100, verbose_name='Name')),
                ('email', models.EmailField(max_length=254, verbose_name='Email')),
                ('message', models.TextField(verbose_name='Message')),
            ],
            options={
                'verbose_name': 'Contact Profile',
                'verbose_name_plural': 'Contact Profiles',
```

```
            'ordering': ['timestamp'],
        },
    ),
    migrations.CreateModel(
        name='Media',
        fields=[
            ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
            ('image', models.ImageField(blank=True, null=True, upload_to='media')),
            ('url', models.URLField(blank=True, null=True)),
            ('name', models.CharField(blank=True, max_length=200, null=True)),
            ('is_image', models.BooleanField(default=True)),
        ],
        options={
            'verbose_name': 'Media',
            'verbose_name_plural': 'Media Files',
            'ordering': ['name'],
        },
    ),
    migrations.CreateModel(
        name='Skill',
        fields=[
            ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
            ('name', models.CharField(blank=True, max_length=20, null=True)),
            ('score', models.IntegerField(blank=True, default=80, null=True)),
        ],
        options={
            'verbose_name': 'Skill',
```

```
            'verbose_name_plural': 'Skills',
        },
    ),
    migrations.CreateModel(
        name='TagProfile',
        fields=[
            ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
            ('name', models.CharField(blank=True, max_length=20, null=True)),
        ],
        options={
            'verbose_name': 'Tag',
            'verbose_name_plural': 'Tags',
        },
    ),
    migrations.CreateModel(
        name='Testimonial',
        fields=[
            ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
            ('thumbnail', models.ImageField(blank=True, null=True, upload_to='')),
            ('name', models.CharField(blank=True, max_length=200, null=True)),
            ('role', models.CharField(blank=True, max_length=200, null=True)),
            ('quote', models.CharField(blank=True, max_length=500, null=True)),
        ],
        options={
            'verbose_name': 'Testimonial',
            'verbose_name_plural': 'Testimonials',
            'ordering': ['name'],
```

```
            },
        ),
        migrations.CreateModel(
            name='TypeProfile',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('name', models.CharField(blank=True, max_length=20, null=True)),
            ],
            options={
                'verbose_name': 'Type Profiles',
                'verbose_name_plural': 'Type Profiles',
            },
        ),
        migrations.CreateModel(
            name='UserProfile',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('title', models.CharField(blank=True, max_length=200, null=True)),
                ('bio', models.TextField(blank=True, null=True)),
                ('cv', models.FileField(blank=True, null=True, upload_to='cv')),
                ('skills', models.ManyToManyField(blank=True, to='main.Skill')),
                ('user', models.OneToOneField(on_delete=django.db.models.deletion.CASCADE,
to=settings.AUTH_USER_MODEL)),
            ],
            options={
                'verbose_name': 'User Profile',
                'verbose_name_plural': 'User Profiles',
```

```
            },
        ),
        migrations.CreateModel(
            name='Portfolio',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('name', models.CharField(blank=True, max_length=200, null=True)),
                ('description', models.CharField(blank=True, max_length=500, null=True)),
                ('body', ckeditor.fields.RichTextField(blank=True, null=True)),
                ('image', models.ImageField(blank=True, null=True, upload_to='portfolio')),
                ('slug', models.SlugField(blank=True, null=True)),
                ('portfolio_tags', models.ManyToManyField(blank=True, to='main.TagProfile')),
                ('portfolio_types', models.ManyToManyField(blank=True, to='main.TypeProfile')),
            ],
            options={
                'verbose_name': 'Portfolio',
                'verbose_name_plural': 'Portfolio Profiles',
                'ordering': ['name'],
            },
        ),
        migrations.CreateModel(
            name='Blog',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('timestamp', models.DateTimeField(auto_now_add=True)),
                ('author', models.CharField(blank=True, max_length=200, null=True)),
                ('name', models.CharField(blank=True, max_length=200, null=True)),
```

```
            ('description', models.CharField(blank=True, max_length=500, null=True)),
            ('body', ckeditor.fields.RichTextField(blank=True, null=True)),
            ('slug', models.SlugField(blank=True, null=True)),
            ('image', models.ImageField(blank=True, null=True, upload_to='blog')),
            ('blog_tags', models.ManyToManyField(blank=True, to='main.TagProfile')),
            ('blog_types', models.ManyToManyField(blank=True, to='main.TypeProfile')),
        ],
        options={
            'verbose_name': 'Blog',
            'verbose_name_plural': 'Blog Profiles',
            'ordering': ['timestamp'],
        },
    ),
]
```

Python

Urls.py

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from django.views.static import serve
from django.conf.urls import url

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('main.urls', namespace="main")),
```

```
    url(r'^media/(?P<path>.*)$', serve,
        {'document_root': settings.MEDIA_ROOT}),
    url(r'^static/(?P<path>.*)$', serve,
        {'document_root': settings.STATIC_ROOT}),
]
if settings.DEBUG:
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

admin.py

```
from django.contrib import admin
from . models import (
    UserProfile,
    ContactProfile,
    Testimonial,
    Media,
    Portfolio,
    Blog,
    Certificate,
    Skill
    )


@admin.register(UserProfile)
class UserProfileAdmin(admin.ModelAdmin):
        list_display = ('id', 'user')


@admin.register(ContactProfile)
```

```
class ContactAdmin(admin.ModelAdmin):
        list_display = ('id', 'timestamp', 'name',)


@admin.register(Testimonial)
class TestimonialAdmin(admin.ModelAdmin):
    list_display = ('id','name','is_active')


@admin.register(Media)
class MediaAdmin(admin.ModelAdmin):
    list_display = ('id', 'name')


@admin.register(Portfolio)
class PortfolioAdmin(admin.ModelAdmin):
    list_display = ('id','name','is_active')
    readonly_fields = ('slug',)


@admin.register(Blog)
class BlogAdmin(admin.ModelAdmin):
    list_display = ('id','name','is_active')
    readonly_fields = ('slug',)


@admin.register(Certificate)
class CertificateAdmin(admin.ModelAdmin):
    list_display = ('id','name')


@admin.register(Skill)
class SkillAdmin(admin.ModelAdmin):
    list_display = ('id','name','score')
```

views.py

```
from django.shortcuts import render
from django.contrib import messages
from .models import (
        UserProfile,
        Blog,
        Portfolio,
        Testimonial,
        Certificate
    )

from django.views import generic


from . forms import ContactForm


class IndexView(generic.TemplateView):
    template_name = "main/index.html"

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        testimonials = Testimonial.objects.filter(is_active=True)
        certificates = Certificate.objects.filter(is_active=True)
        blogs = Blog.objects.filter(is_active=True)
        portfolio = Portfolio.objects.filter(is_active=True)

        context["testimonials"] = testimonials
        context["certificates"] = certificates
        context["blogs"] = blogs
        context["portfolio"] = portfolio
        return context


class ContactView(generic.FormView):
    template_name = "main/contact.html"
    form_class = ContactForm
    success_url = "/"

    def form_valid(self, form):
        form.save()
        messages.success(self.request, 'Thank you. We will be in touch soon.')
        return super().form_valid(form)
```

```
class PortfolioView(generic.ListView):
        model = Portfolio
        template_name = "main/portfolio.html"
        paginate_by = 10

        def get_queryset(self):
                return super().get_queryset().filter(is_active=True)


class PortfolioDetailView(generic.DetailView):
        model = Portfolio
        template_name = "main/portfolio-detail.html"

class BlogView(generic.ListView):
        model = Blog
        template_name = "main/blog.html"
        paginate_by = 10

        def get_queryset(self):
                return super().get_queryset().filter(is_active=True)


class BlogDetailView(generic.DetailView):
                                                model = Blog
        template_name = "main/blog-detail.html"
```

models.py

```
from django.db import models
from django.contrib.auth.models import User
from django.template.defaultfilters import slugify
from ckeditor.fields import RichTextField


class Skill(models.Model):
    class Meta:
        verbose_name_plural = 'Skills'
        verbose_name = 'Skill'

    name = models.CharField(max_length=20, blank=True, null=True)
    score = models.IntegerField(default=80, blank=True, null=True)
    image = models.FileField(blank=True, null=True, upload_to="skills")
    is_key_skill = models.BooleanField(default=False)

    def __str__(self):
        return self.name

class UserProfile(models.Model):
```

```
    class Meta:
        verbose_name_plural = 'User Profiles'
        verbose_name = 'User Profile'

    user = models.OneToOneField(User, on_delete=models.CASCADE)
    avatar = models.ImageField(blank=True, null=True, upload_to="avatar")
    title = models.CharField(max_length=200, blank=True, null=True)
    bio = models.TextField(blank=True, null=True)
    skills = models.ManyToManyField(Skill, blank=True)
    cv = models.FileField(blank=True, null=True, upload_to="cv")

    def __str__(self):
        return f'{self.user.first_name} {self.user.last_name}'


class ContactProfile(models.Model):

    class Meta:
        verbose_name_plural = 'Contact Profiles'
        verbose_name = 'Contact Profile'
        ordering = ["timestamp"]
    timestamp = models.DateTimeField(auto_now_add=True)
    name = models.CharField(verbose_name="Name",max_length=100)
    email = models.EmailField(verbose_name="Email")
    message = models.TextField(verbose_name="Message")

    def __str__(self):
        return f'{self.name}'



class Testimonial(models.Model):

    class Meta:
        verbose_name_plural = 'Testimonials'
        verbose_name = 'Testimonial'
        ordering = ["name"]

    thumbnail = models.ImageField(blank=True, null=True, upload_to="testimonials")
    name = models.CharField(max_length=200, blank=True, null=True)
    role = models.CharField(max_length=200, blank=True, null=True)
    quote = models.CharField(max_length=500, blank=True, null=True)
    is_active = models.BooleanField(default=True)

    def __str__(self):
        return self.name


class Media(models.Model):

    class Meta:
```

```
        verbose_name_plural = 'Media Files'
        verbose_name = 'Media'
        ordering = ["name"]

    image = models.ImageField(blank=True, null=True, upload_to="media")
    url = models.URLField(blank=True, null=True)
    name = models.CharField(max_length=200, blank=True, null=True)
    is_image = models.BooleanField(default=True)

    def save(self, *args, **kwargs):
        if self.url:
            self.is_image = False
        super(Media, self).save(*args, **kwargs)
    def __str__(self):
        return self.name

class Portfolio(models.Model):

    class Meta:
        verbose_name_plural = 'Portfolio Profiles'
        verbose_name = 'Portfolio'
        ordering = ["name"]
    date = models.DateTimeField(blank=True, null=True)
    name = models.CharField(max_length=200, blank=True, null=True)
    description = models.CharField(max_length=500, blank=True, null=True)
    body = RichTextField(blank=True, null=True)
    image = models.ImageField(blank=True, null=True, upload_to="portfolio")
    slug = models.SlugField(null=True, blank=True)
    is_active = models.BooleanField(default=True)

    def save(self, *args, **kwargs):
        if not self.id:
            self.slug = slugify(self.name)
        super(Portfolio, self).save(*args, **kwargs)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return f"/portfolio/{self.slug}"


class Blog(models.Model):

    class Meta:
        verbose_name_plural = 'Blog Profiles'
        verbose_name = 'Blog'
        ordering = ["timestamp"]

    timestamp = models.DateTimeField(auto_now_add=True)
    author = models.CharField(max_length=200, blank=True, null=True)
```

```
    name = models.CharField(max_length=200, blank=True, null=True)
    description = models.CharField(max_length=500, blank=True, null=True)
    body = RichTextField(blank=True, null=True)
    slug = models.SlugField(null=True, blank=True)
    image = models.ImageField(blank=True, null=True, upload_to="blog")
    is_active = models.BooleanField(default=True)

    def save(self, *args, **kwargs):
        if not self.id:
            self.slug = slugify(self.name)
        super(Blog, self).save(*args, **kwargs)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return f"/blog/{self.slug}"


class Certificate(models.Model):

    class Meta:
        verbose_name_plural = 'Certificates'
        verbose_name = 'Certificate'

    date = models.DateTimeField(blank=True, null=True)
    name = models.CharField(max_length=50, blank=True, null=True)
    title = models.CharField(max_length=200, blank=True, null=True)
    description = models.CharField(max_length=500, blank=True, null=True)
    is_active = models.BooleanField(default=True)

    def __str__(self):
        return self.name
```

CHAPTER 7

TESTING
(TEST PLAN/CASES/RESULT)

7.1 Test Plan

A comprehensive test plan is crucial to ensure the reliability and functionality of DRESUME. The test plan outlines the testing strategy, objectives, scope, and resources required for testing. It includes various types of testing such as unit testing, integration testing, system testing, and user acceptance testing (UAT). The objectives of the test plan are to verify that all functionalities work as intended, identify and fix any bugs or issues, ensure compatibility across different devices and browsers, and validate the overall user experience. The scope of testing covers all features and functionalities of DRESUME, including user authentication, property listings, search and filtering, messaging system, and user dashboard. The resources required for testing include testing tools, devices, and personnel responsible for conducting and documenting the tests.

7.2 Test Cases

Test cases are detailed descriptions of specific scenarios and inputs to be tested, along with expected outcomes and actual results. Each test case covers a particular aspect or functionality of DRESUME and includes steps to execute the test, input data, expected results, and actual results observed during testing. For example, a test case for user authentication would include steps to register a new user, input valid credentials for login, and verify that the user is successfully logged in. Test cases are designed to validate the correctness and robustness of DRESUME and ensure that it meets the specified requirements and user expectations.

7.3 Test Results

After executing the test cases, the results of testing are documented and summarized. This includes a summary of the testing outcomes, highlighting any issues or defects encountered during testing, such as bugs, errors, or inconsistencies. The test results also include details of any successful tests and 59confirmation that the functionalities work as intended. Additionally, any deviations from the expected results are noted, along with the steps taken to address them.

7.4 Issues and Resolutions

This section provides a detailed discussion of significant issues encountered during testing and the steps taken to resolve them. It includes descriptions of the issues identified, their root causes, and the actions taken to fix them. Additionally, it may include discussions on any challenges faced during testing, such as compatibility issues, performance issues, or usability issues, and the strategies employed to overcome them

CHAPTER 8

CONCLUSION

8.2 PROJECT LIMITATION

Despite the successful development of DRESUME, there may be certain limitations encountered during the project. These limitations could include technical constraints, such as limitations of the chosen technologies or frameworks, resource limitations, such as budget or time constraints, or scope changes that may have affected the project deliverables. It is essential to acknowledge these limitations to provide a realistic assessment of the project's achievements and to identify areas for improvement in future projects.

8.2 FUTURE SCOPE

Online Resume Builder can be used in accordance with customer requirements. Customers can customize theirs resumes with a selection of topics and details. Services are It's hard to be beaten by competitors because the system is giving customers exactly what they want.

CHAPTER 9

REFERENCES

- Wikipedia.org
- www.w3schools.com
- Software Engineering – R.S.Pressma
- JavaScript By McGraw hill Publication
- Youtube.com

Dupli Checker

PLAGIARISM SCAN REPORT

9%
Plagiarised

91%
Unique

Date	2024-02-27
Words	985
Characters	6675

Content Checked For Plagiarism

The current system creates resumes manually either using Word or Google doc, so there are some issues with the current system. The existing system suffered from a number of shortcomings. Since the entire system had to be maintained, the process of storing, maintaining and retrieving information was very tedious and time-consuming. The records were never in a systematic order, there used to be a lot of difficulty in assigning any particular transaction to a particular context. If any information was to be found, it was required to go through various registers, the documents would never exist. something like report generation There would always be wasted time entering records and retrieving records Another problem was that it was very difficult to find errors in entering records Once the records were inserted it was very difficult to update those records

A resume is the first meeting between you and a potential employer, more often than ever.
So how do you want to be remembered? Wrinkled and messy. Neat and structured. Long and boring. Accurate and interesting. Companies do not have time to interview every applicant who is interested in the job. If they did, there would be no company to work for. They use a process of elimination. That's right - he continues. When a job seeker wants to apply for a job online, they generally need to attach their resume to an email. The online resume builder system provides users with popular resume formats and a better way to showcase their resumes to employers. A job seeker does not need to attach a resume to every email, just provide the URL of your resume and the employer can view the resume online by clicking on the link and also download it.

The purpose of this online resume builder is to help users create a professional resume for themselves. Candidates don't need to invest extra time in planning and creating a polished resume. They can immediately enter their information in the pop-up box and a resume with a nice layout will be generated for them.

The current system creates resumes manually using either Word or Google doc, so there are some issues with the current system. The existing system suffered from a number of shortcomings. Since the entire system had to be maintained, the process of storing, maintaining and retrieving information was very tedious and time-consuming. The records were never in a systematic order, there used to be a lot of difficulty in assigning any particular transaction to a particular context. If any information was to be found, it was required to go through various registers, the documents would never exist. something like generating reports There would always be wasted time entering records and retrieving records Another problem was that it was very difficult to find errors in entering records Once the records were entered it was very difficult to update those records.

The purpose of this online resume builder is to help users create a professional resume for themselves. Candidates don't need to invest extra time in planning and creating a polished resume. They can immediately enter their information in the pop-up box and a resume with a nice layout will be generated for them.

User registration and verification:
· Users should be able to create accounts with a valid username and password.

· **Secure authentication mechanisms should be implemented to protect user accounts.**

Portfolio:

· Users can add and view their reviews..

blog:

· Users can create and edit their blog posts.

Contact:

· Users can contact the person by filling out the contact form.

Key Skills:

· User can showcase their key skills and edit them.

Coding Skills:

· User can show and edit their coding skills.

Performance:

· A website should serve a specified number of concurrent users without degrading performance.

Safety:

· User data must be stored securely.

Applicability:

· Websites should have an intuitive and responsive design for a positive user experience.

User registration and verification:

· Users should be able to create accounts with a valid username and password.

· **Secure authentication mechanisms should be implemented to protect user accounts.**

Portfolio:

· Users can add and view their reviews..

blog:

· Users can create and edit their blog posts.

Contact:

· Users can contact the person by filling out the contact form.

Key Skills:

· User can showcase their key skills and edit them.

Coding Skills:

· User can show and edit their coding skills.

Non-functional requirements:

Performance:

· A website should serve a specified number of concurrent users without degrading performance.

Safety:

· User data must be stored securely.

Applicability:

· Websites should have an intuitive and responsive design for a positive user experience.

Advantages of Online Resume Builder:

· Save time

· Resume templates help you save time by allowing you to focus on customizing the details.

· Display achievements

· Resume builders can help you display your accomplishments, skills, and certifications at a glance.

· Consistent information

Digital Resume provides consistent and accurate information based on the data it is programmed with, reducing the likelihood of human error in relaying details.

Scope of the Digital Resume project:

· **Secure authentication mechanisms should be implemented to protect user accounts.**

Portfolio:

· Users can add and view their reviews..

blog:

· Users can create and edit their blog posts.

Contact:

· Users can contact the person by filling out the contact form.

Key Skills:

· User can showcase their key skills and edit them.

Coding Skills:

· User can show and edit their coding skills.

Performance:

· A website should serve a specified number of concurrent users without degrading performance.

Safety:

· User data must be stored securely.

Applicability:

· Websites should have an intuitive and responsive design for a positive user experience.

User registration and verification:

· Users should be able to create accounts with a valid username and password.

· **Secure authentication mechanisms should be implemented to protect user accounts.**

Portfolio:

· Users can add and view their reviews..

blog:

· Users can create and edit their blog posts.

Contact:

· Users can contact the person by filling out the contact form.

Key Skills:

· User can showcase their key skills and edit them.

Coding Skills:

· User can show and edit their coding skills.

Non-functional requirements:

Performance:

· A website should serve a specified number of concurrent users without degrading performance.

Safety:

· User data must be stored securely.

Applicability:

· Websites should have an intuitive and responsive design for a positive user experience.

Advantages of Online Resume Builder:

· Save time

· Resume templates help you save time by allowing you to focus on customizing the details.

· Display achievements

· Resume builders can help you display your accomplishments, skills, and certifications at a glance.

· Consistent information

Digital Resume provides consistent and accurate information based on the data it is programmed with, reducing the likelihood of human error in relaying details.

Scope of the Digital Resume project:

Online Resume Builder can be used in accordance with customer requirements. Customers can customize theirs resumes with a selection of topics and details. Services are It's hard to be beaten by competitors because the system is giving customers exactly what they want.

In conclusion, online resume builder is one of the most fantastic systems for people who are either fresher in their domain or if they don't have enough ideas about resume or don't have enough time to create a good resume resume. **or patterns, then this platform is a very productive place for them.** It saves a lot of time and is cost effective. As technology continues to evolve, this project is a testament to the potential of innovative solutions in simplifying the college selection process, ultimately benefiting users.

Matched Source

Similarity 9%

Title:The One Question That Should Guide Your Daily Life

Apr 9, 2021 — So, how do you want to be remembered for your time on earth? That's the first and foremost question that should guide our everyday life ...

https://www.omaritani.com/blog/how-do-you-want-to-be-remembered

---

Similarity 5%

Title:Web Application Penetration Testing Services

Implement Secure Authentication: Robust and secure authentication mechanisms should be implemented to protect user accounts and sensitive information. This ...

https://bluegoatcyber.com/services/penetration-testing/web-application-penetration-testing/

Similarity 2%

Title:docshare.tips › online-resume-builder-report_5747dOnline Resume Builder-Report - DocShare.tips

a) Creating resumes online. b) Customizing the look and details. c) Keeping track of the customers and their resumes. 1.2 Scope Online Resume Builder can be used in accordance with the requirements of the customers. Customers can customize their resumes with their choice of themes & details.

https://docshare.tips/online-resume-builder-report_5747d540b6d87f83a78b4671.html/

Similarity 2%

Title:resume builder.pptx - SlideShare

Dec 21, 2022 · Conclusion • The online resume builder is one of the most fantastic systems for the people who are either recently graduated students in their domain or if they don't have enough idea about the resume or don't have enough time to create the resume of good designs 🔎or patterns, then this platform is a very productive place for them ✔ .

https://www.slideshare.net/learnEnglish51/resume-builderpptx
