A PROJECT REPORT ON

# ONLINE RESUME BUILDER

**A Report Submitted to**

**IIDT - Blackbuck Engineers Pvt. Ltd**

Submitted by

**D Harathi**

**Register no:212G1A0571**


# Acknowledgement

I would like to express my heartfelt gratitude to everyone who has supported and contributed to the successful completion of this project, "Online Resume Builder using Django."

First and foremost, I would like to thank Aquib Ajani, my project guide, for their invaluable guidance, continuous support, and encouragement throughout the duration of this project. Their insights and feedback were instrumental in shaping this project to its current form.

I extend my sincere thanks to the faculty and staff of IIDT - Blackbuck Engineers Pvt. Ltd for providing the necessary resources and a conducive environment for the successful completion of this project.

A special thanks to my colleagues and friends for their constant motivation, technical support, and collaboration. Their assistance in various stages of the project has been immensely helpful.

Finally, I am profoundly grateful to my family for their unwavering support, patience, and understanding, which have been my constant source of strength and inspiration throughout this journey.



## Abstract

The Online Resume Builder using Django is a web-based application designed to simplify the process of creating professional resumes. This project leverages the powerful Django framework to provide users with an intuitive, user-friendly platform for designing and generating resumes. The primary objective is to assist users, particularly job seekers, in crafting well-structured and visually appealing resumes that highlight their skills, experiences, and qualifications effectively.

The application features a comprehensive set of tools for creating, editing, and customizing resumes. Users can select from various templates, input personal and professional information, and preview their resumes before finalizing them. The backend of the application is built using Django, ensuring robust performance, scalability, and security. The frontend employs HTML, CSS, and JavaScript to deliver a responsive and interactive user experience.

Key functionalities of the Online Resume Builder include user authentication, template selection, real-time preview, and the ability to download the resume in multiple formats. Additionally, the application provides an administrative interface for managing user data and monitoring system performance.

This project addresses the need for a streamlined and efficient resume-building process, particularly in today's competitive job market. By automating and simplifying resume creation, the Online Resume Builder aims to empower users to present their qualifications in the best possible light, thereby enhancing their chances of securing employment opportunities.


Keywords: Online Resume Builder, Django, Web Application, Resume Templates, User Authentication, Real-time Preview, Downloadable Formats.

# INTRODUCTION

In the contemporary job market, a meticulously crafted resume is indispensable for job seekers aiming to capture the attention of prospective employers. A resume functions as a crucial instrument that showcases an individual's skills, experiences, and qualifications. However, the process of creating a professional resume can be daunting and time-consuming, particularly for those lacking design expertise or familiarity with effective document formatting.

The Online Resume Builder using Django is a sophisticated web-based application designed to mitigate these challenges by providing users with a streamlined, user-centric platform for constructing professional resumes. This project harnesses the capabilities of the Django framework to develop a robust backend, ensuring efficient data management, security, and scalability. The frontend, developed using HTML, CSS, and JavaScript, delivers a responsive and interactive user interface, thereby facilitating an intuitive and accessible resume-building process.

The primary objective of this project is to simplify and automate the resume creation process, enabling users to concentrate on the content of their resumes rather than on formatting and design intricacies. The Online Resume Builder offers an array of pre-designed templates, allowing users to select formats that best align with their needs and preferences. Users can input their personal and professional information, preview their resumes in real-time, and download the final document in various formats, including PDF and Word.

Key features of the Online Resume Builder include user authentication to ensure data privacy and security, a selection of customizable templates, and a real-time preview functionality that permits users to view changes instantaneously. Additionally, the application includes an administrative interface for managing user data and monitoring system performance, thereby ensuring the platform remains efficient and reliable.

This project aspires not only to enhance the user experience by providing a convenient and efficient tool for resume creation but also to address the broader need for accessible and effective job application resources. By streamlining the resume-building process, the Online Resume Builder empowers users to present their qualifications confidently and professionally, thereby enhancing their prospects in the job market.

The subsequent sections of this documentation will provide an exhaustive overview of the system requirements, installation instructions, features and functionalities, architecture, database schema, API documentation, user interface design, testing strategies, deployment procedures, and conclusions. Each section aims to offer comprehensive insights into the development and functionality of the Online Resume Builder using Django.

## Advantages:

- User-Friendly Interface
- Real-Time Preview
- Multi-Format Export Options
- Secure User Authentication
- Efficient Data Management
- Automated Formatting
- Scalability

## Disadvantages:

- Dependence on Internet Connectivity
- Data Privacy Concerns
- Limited Support for Non-Standard Resumes
- User Authentication Complexity
- Limited Support for Non-Standard Resumes
- Ongoing Maintenance Requirements

## Software and Hardware Requirements

## Software Requirements

1. **Operating System**
    - Windows 10 or later
    - macOS 10.14 (Mojave) or later
    - Linux (any modern distribution)
2. **Web Browser**
    - Google Chrome (latest version)
    - Mozilla Firefox (latest version)
    - Microsoft Edge (latest version)
    - Safari (latest version)
3. **Development Environment**
    - Python 3.8 or later
    - Django 3.2 or later
    - Virtualenv for managing the project environment
4. **Database**
    - PostgreSQL 12 or later
    - SQLite (for development and testing purposes)
5. **Libraries and Dependencies**
    - Django REST framework for API development
    - Gunicorn for WSGI HTTP Server
    - psycopg2 for PostgreSQL integration

- Other Python packages as listed in the requirements.txt file

6. **Version Control System**
   - Git (latest version)
7. **Integrated Development Environment (IDE)**
   - Visual Studio Code
   - PyCharm
   - Sublime Text
   - Any preferred code editor
8. **Other Tools**
   - Postman for API testing
   - Docker (optional, for containerization)
   - Nginx (for production deployment)

## Hardware Requirements

1. **Development Machine**
   - **Processor:** Intel Core i5 or equivalent AMD processor (minimum)
   - **RAM:** 8 GB (minimum), 16 GB or more recommended
   - **Storage:** 256 GB SSD (minimum), 512 GB or more recommended
   - **Display:** 13-inch display with 1080p resolution (minimum), larger screen and higher resolution recommended for better productivity
2. **Server (for Deployment)**
   - **Processor:** Multi-core server-grade processor (Intel Xeon or equivalent)
   - **RAM:** 8 GB (minimum), 16 GB or more recommended
   - **Storage:** 100 GB SSD (minimum), scalable storage depending on the number of users and data volume
   - **Network:** Reliable internet connection with sufficient bandwidth to handle user traffic
   - **Operating System:** Ubuntu Server 20.04 LTS or other server-grade Linux distributions
3. **Additional Hardware**
   - **Backup Storage:** External hard drive or cloud storage service for regular backups
   - **Uninterruptible Power Supply (UPS):** To protect against power outages and ensure continuous operation of the development and production environment

## Motivation:

- **Empowering Users**: Providing an easy-to-use tool that helps users create professional resumes without needing design or technical skills.
- **Showcasing Skills**: Demonstrating your proficiency in Django, web development, and project management.
- **Filling a Gap**: Offering a tailored solution for those who find existing resume builders either too complex or lacking specific features.
- **Enhancing User Experience**: Creating a seamless and intuitive interface for users to build and manage their resumes.
- **Learning and Growth**: Gaining hands-on experience with Django and web development practices to further your career and project skills.

## Literature Review:

- **Existing Resume Builders**:
  - **Tools and Features**: Analyze popular resume builders like Canva, Resume.io, or LinkedIn's resume builder. Note their features, user interface, and strengths.
  - **User Experience**: Look into studies or reviews about what users appreciate in these tools and any common pain points.

- **Web Development Technologies**:
  - **Django Framework**: Review Django's capabilities in web development, including its ORM, templating engine, and form handling.
  - **Frontend Technologies**: Explore HTML, CSS, and JavaScript frameworks that can enhance the frontend experience.

- **Design Principles**:
  - **UI/UX Design**: Research best practices in designing user-friendly interfaces and ensuring a smooth user experience.
  - **Responsive Design**: Understand the importance of creating a responsive design that works well on various devices.

- **Resume Building Best Practices**:
  - **Resume Formats**: Study different resume formats and their effectiveness. Look into what makes a resume stand out to employers.
  - **Content Structuring**: Review guidelines on structuring resume content for clarity and impact.

- **Security and Privacy**:

- **Data Protection**: Investigate best practices for securing user data, especially when handling personal and sensitive information.
- **Compliance**: Ensure understanding of relevant data protection regulations (e.g., GDPR) and how they apply to your project.

- **Case Studies and Previous Work**:

  - **Similar Projects**: Analyze other projects or research papers related to resume builders or similar applications built with Django.
  - **Challenges and Solutions**: Learn from the challenges faced and solutions implemented in similar projects.

## Proposed system

## 1.System Overview

- **Purpose**: To provide an intuitive web-based platform that allows users to create, customize, and manage professional resumes.
- **Scope**: Focus on user-friendly design, customizable resume templates, and secure handling of user data.

## 2. Features

**User Management**:

- **Registration and Authentication**: Secure user registration, login, and password recovery.
- **Profile Management**: Users can update personal details and preferences.

**Resume Building**:

- **Templates**: A selection of customizable resume templates.
- **Editing Tools**: Form-based interface to input and modify resume content (e.g., education, work experience, skills).
- **Preview and Export**: Live preview of the resume and options to export in various formats (e.g., PDF).

**Customization**:

- **Theme Options**: Allow users to select different themes or styles for their resumes.
- **Content Sections**: Users can add, remove, or rearrange sections as needed.

**Data Handling**:

- **Saving and Loading**: Save user resumes securely and enable retrieval for future edits.
- **Privacy and Security**: Implement measures to protect user data and ensure privacy.

## 3. Architecture

**Frontend**:

- **Technologies**: HTML, CSS, JavaScript for creating a responsive and interactive user interface.
- **Frameworks/Libraries**: Potential use of frontend frameworks (e.g., React or Vue.js) if needed.

**Backend**:

- **Framework**: Django for handling server-side logic, user authentication, and data management.
- **Database**: Use Django's ORM with a relational database (e.g., PostgreSQL or SQLite) to store user data and resume information.

**APIs**:

- **RESTful API**: Develop a RESTful API for resume operations if integrating with external systems or for a more modular approach.

## 4. User Interface

**Dashboard**:

- **Overview**: User-friendly dashboard where users can manage their resumes.
- **Navigation**: Clear navigation to access different features and sections.

**Resume Editor**:

- **Editing Experience**: WYSIWYG (What You See Is What You Get) editor or form-based input for ease of use.
- **Customization Options**: Intuitive tools for modifying templates and content.

## 5. Security and Privacy

**Authentication**:

- **Secure Login**: Use Django's authentication system for secure user login.
- **Session Management**: Implement secure session handling.

**Data Protection**:

- **Encryption**: Encrypt sensitive data both in transit and at rest.
- **Compliance**: Adhere to data protection regulations relevant to user data.

## 6. Testing and Deployment

**Testing**:

- **Unit Testing**: Implement unit tests for backend logic and functionality.
- **User Testing**: Conduct usability testing to gather feedback and improve the user experience.

**Deployment**:

- **Hosting**: Choose a reliable hosting provider for deploying the Django application.
- **CI/CD**: Set up continuous integration and deployment pipelines for streamlined updates.

## 7. Future Enhancements

- **Additional Templates**: Regularly update and add new resume templates.
- **Advanced Features**: Explore adding features like resume analytics or integration with job application systems.

## Keywords and Definitions:

### 1. Django

- **Definition**: A high-level Python web framework that encourages rapid development and clean, pragmatic design. It provides tools and libraries for building web applications efficiently.

### 2. Template

- **Definition**: A pre-designed layout or structure used as a starting point for creating resumes. Templates can be customized to fit the user's needs and preferences.

### 3. RESTful API

- **Definition**: A style of web service that allows communication between different systems using HTTP requests. RESTful APIs use standard HTTP methods and are stateless.

### 4. User Authentication

- **Definition**: The process of verifying a user's identity before granting access to the system. This typically involves login credentials like username and password.

### 5. User Authorization


- **Definition**: The process of determining what an authenticated user is allowed to do within the system. It involves managing permissions and access levels.

## 6. ORM (Object-Relational Mapping)

- **Definition**: A programming technique used to convert data between incompatible type systems in object-oriented programming languages. Django's ORM allows interaction with the database using Python objects.

## 7. WYSIWYG (What You See Is What You Get)

- **Definition**: A type of editor that allows users to see how their content will look as they are editing it, providing a more intuitive editing experience.

## 8. Responsive Design

- **Definition**: An approach to web design that ensures a website looks good and functions well on various devices and screen sizes, including desktops, tablets, and smartphones.

## 9. Encryption

- **Definition**: The process of converting information or data into a secure format that is unreadable without a decryption key, used to protect sensitive information.

## 10. CI/CD (Continuous Integration/Continuous Deployment)

- **Definition**: A set of practices and tools that automate the process of integrating code changes and deploying them to production environments, improving the development workflow.

## 11. Database Schema

- **Definition**: The structure of a database, including tables, fields, relationships, and constraints, which defines how data is organized and accessed.

## 12. CRUD Operations

- **Definition**: The basic operations performed on data: Create, Read, Update, and Delete. These operations are fundamental to interacting with databases and APIs.

## 13. Usability Testing

- **Definition**: A technique used to evaluate a product by testing it with real users. The goal is to identify usability issues and improve the user experience.

### 14. Session Management

- **Definition**: The process of managing user sessions, including handling logins, timeouts, and user interactions during a session to maintain state and security.

### 15. Data Protection Regulations

- **Definition**: Legal requirements and standards for safeguarding personal and sensitive data, such as GDPR (General Data Protection Regulation) in the EU.

## Implementation

### 1. Project Setup

**Environment**:

- **Install Django**: Set up your development environment by installing Django and other necessary packages.
- **Create a Django Project**: Initialize a new Django project using `django-admin startproject project_name`.
- **Create an App**: Set up a Django app within the project using `python manage.py startapp app_name`.

### 2. Database Design

**Models**:

- **User Model**: Utilize Django's built-in User model or extend it if needed.
- **Resume Model**: Design a model to represent resumes, including fields like user, template, and content sections.
- **Content Sections**: Create models for different resume sections (e.g., education, experience) if needed.

**Migrations**:

- **Define Migrations**: Create and apply migrations to set up your database schema with `python manage.py makemigrations` and `python manage.py migrate`.

### 3. User Authentication

**Registration and Login**:

- **Forms**: Create forms for user registration, login, and password recovery using Django's form handling.
- **Views**: Implement views to handle user authentication processes.

- **Templates**: Design templates for login, registration, and password management pages.

## 4. Resume Management

**Templates**:

- **Template Selection**: Implement a feature to select and apply different resume templates.
- **Customization**: Allow users to customize their chosen templates.

**Editing**:

- **Form-Based Editing**: Build forms to input and edit resume content, such as work experience and education.
- **Preview**: Create a preview feature to show how the resume will look before saving.

**Saving and Loading**:

- **Save**: Implement functionality to save user resumes in the database.
- **Load**: Allow users to retrieve and edit previously saved resumes.

## 5. Frontend Development

**User Interface**:

- **Design**: Develop a responsive and user-friendly interface using HTML, CSS, and JavaScript.
- **Frontend Frameworks**: Consider using frameworks like Bootstrap or Tailwind CSS to enhance design.

**Integration**:

- **Connect Backend**: Ensure that frontend forms and views are properly connected to the Django backend for data handling.

## 6. Export and Sharing

**Export Formats**:

- **PDF Export**: Implement functionality to export resumes in PDF format using libraries like WeasyPrint or wkhtmltopdf.
- **Other Formats**: Optionally, support other formats like DOCX or plain text.

## 7. Security and Privacy

**Data Protection**:


- **Encryption**: Use HTTPS for secure data transmission and encrypt sensitive data.
- **Authentication**: Ensure secure user authentication and session management.

**Compliance**:

- **Regulations**: Adhere to data protection regulations (e.g., GDPR) to safeguard user data.

## 8. Testing

**Unit Tests**:

- **Backend Testing**: Write unit tests for Django models, views, and forms.
- **Frontend Testing**: Test the frontend components and interactions.

**User Testing**:

- **Usability Testing**: Conduct usability testing to gather feedback and refine the user experience.

## 9. Deployment

**Hosting**:

- **Choose a Platform**: Deploy the Django application on a hosting service like Heroku, AWS, or DigitalOcean.
- **Configuration**: Configure the production environment settings, including database connections and static files.

**CI/CD**:

- **Set Up Pipelines**: Implement continuous integration and deployment pipelines for automated testing and deployment.

## 10. Documentation and Maintenance

**Documentation**:

- **User Guides**: Provide documentation for users on how to use the resume builder.
- **Developer Documentation**: Document the codebase and deployment procedures for future maintenance.

**Maintenance**:

- **Updates**: Regularly update the application with new features and security patches.
- **Bug Fixes**: Address and fix any issues reported by users.

## Challenges Faced:

- Design and User Experience
- Template Customization
- Data Security and Privacy
- Handling File Uploads and Exports
- Performance and Scalability
- Integration with Frontend
- Testing and Debugging

## System testing:

System testing for the Online Resume Builder application involves a comprehensive approach to ensure that all components function correctly and meet the specified requirements. The primary objectives of system testing are to verify the application's functionality, validate its usability, evaluate its performance, and ensure security and compatibility.

**Unit testing** is the first step in the process, focusing on individual components such as models, views, and forms. Using Django's built-in testing framework, each component is tested to confirm that it behaves as expected. The results of these tests provide a detailed view of the correctness of each unit within the application.

**Integration testing** follows, with the goal of verifying that different parts of the system work together seamlessly. This includes testing interactions between various components, such as models and views, as well as ensuring that APIs function correctly. Integration tests help identify any issues arising from component interactions.

**Performance testing** evaluates how the application performs under different conditions, including load testing and response time measurements. Tools such as Apache JMeter or Locust are used to simulate various user loads and measure the application's performance to ensure it can handle expected traffic and usage patterns.

**Acceptance testing** validates that the application meets all business requirements and is ready for deployment. This phase includes User Acceptance Testing (UAT) with stakeholders to confirm that the application fulfills its intended purpose and receives final approval before release.

### Test Results

The system testing of the Online Resume Builder application yielded valuable insights into the application's performance, functionality, and overall quality. Each testing phase produced specific results that collectively confirm the application's readiness for deployment.

## Coding:

### home.html

```
{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <meta name="description" content="The Curriculum Vitae of Sakshi-Aafiya-Hasti Bloggs.">

   <title>Sakshi-Aafiya-Hasti - Curriculum Vitae</title>

   <link rel="stylesheet" href="{% static 'css/style.css' %}">

   <link href="http://fonts.googleapis.com/css?family=Rokkitt:400,700|Lato:400,300"
rel="stylesheet">

   <!--[if lt IE 9]>

   <script src="//html5shiv.googlecode.com/svn/trunk/html5.js"></script>

   <![endif]-->

</head>

<body id="top">

   <div id="cv" class="instaFade">

      <div class="mainDetails">

         <div id="headshot" class="quickFade">

            <img src="{% static 'images/headshot.jpg' %}" alt="Alan Smith">

         </div>


         <div id="name">
```

```
            <h1 class="quickFade delayTwo">{{ name }}</h1>
        <h2 class="quickFade delayThree">SOFTWARE DEVELOPER</h2>
    </div>

    <div id="contactDetails" class="quickFade delayFour">
        <ul>
            <li>Email: {{ email }}</li>
            <li>Mobile: {{ mobile }}</li>
        </ul>
    </div>
    <div class="clear"></div>
</div>

<div id="mainArea" class="quickFade delayFive">
    <section>
        <article>
            <div class="sectionTitle">
                <h1>Personal Profile</h1>
            </div>
            <div class="sectionContent">
                <p>Address: {{ address }}</p>
            </div>
        </article>
        <div class="clear"></div>
    </section>
```


```
<section>
    <div class="sectionTitle">
        <h1>Work Experience</h1>
    </div>
    <div class="sectionContent">
        <article>
            <h2>{{ experience_1_title }}</h2>
            <p class="subDetails">{{ experience_1_dur }}</p>
            <p>{{ experience_1_desc }}</p>
        </article>
        <article>
            <h2>{{ experience_2_title }}</h2>
            <p class="subDetails">{{ experience_2_dur }}</p>
            <p>{{ experience_2_desc }}</p>
        </article>
    </div>
    <div class="clear"></div>
</section>

<section>
    <div class="sectionTitle">
        <h1>Key Skills</h1>
    </div>
    <div class="sectionContent">
```

```
            <ul class="keySkills">
                <li>{{ skills_1 }}</li>
                <li>{{ skills_2 }}</li>
                <li>{{ skills_3 }}</li>
                <li>{{ skills_4 }}</li>
            </ul>
        </div>
        <div class="clear"></div>
    </section>

    <section>
        <div class="sectionTitle">
            <h1>Education</h1>
        </div>
        <div class="sectionContent">
            <article>
                <h2>{{ education_1 }}</h2>
                <p class="subDetails">{{ education_1_dur }}</p>
                <p>Score: {{ education1_score }}</p>
            </article>
            <article>
                <h2>{{ education_2 }}</h2>
                <p class="subDetails">{{ education_2_dur }}</p>
                <p>Score: {{ education2_score }}</p>
            </article>
```

```
                </div>
                <div class="clear"></div>
            </section>
        </div>
    </div>

    <script type="text/javascript">
        var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "http://www.");
        document.write(unescape("%3Cscript src='" + gaJsHost + "google-analytics.com/ga.js' type='text/javascript'%3E%3C/script%3E"));
    </script>
    <script type="text/javascript">
        var pageTracker = _gat._getTracker("UA-3753241-1");
        pageTracker._initData();
        pageTracker._trackPageview();
    </script>
</body>
</html>
```

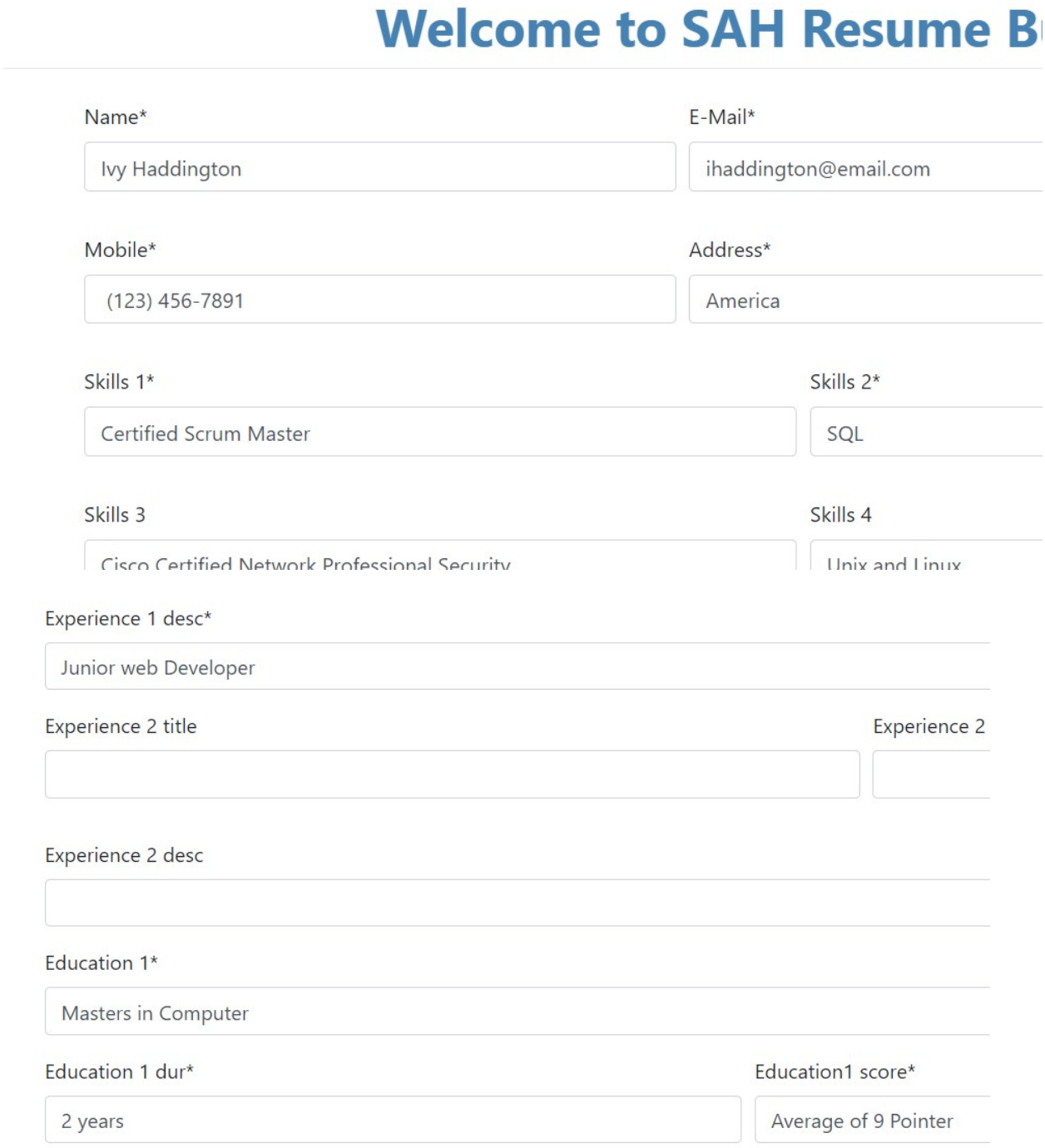
**Info.html**

```
{% load crispy_forms_tags %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
        <meta name="description" content="The Curriculum Vitae of Sakshi-Aafiya-Hasti Bloggs.">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Sakshi-Aafiya-Hasti Resume Builder</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
        integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
        crossorigin="anonymous">
</head>
<body style="padding:20px;">
    <h1 class="mt-2 text-center font-weight-bolder" style="color:steelblue;">Welcome to SAH Resume Builder!!!</h1>
    <hr class="mt-0 mb-4">

    <!-- Form Rendering -->
    <form method="POST">
        {% csrf_token %}
        {% crispy form form.helper %}
        <button type="submit" class="btn btn-primary">Submit</button>
    </form>
</body>
</html>
```

**Results:**


Welcome to SAH Resume B
Name*
Ivy Haddington
E-Mail*
ihaddington@email.com
Mobile*
(123) 456-7891
Address*
America
Skills 1*
Certified Scrum Master
Skills 2*
SQL
Skills 3
Cisco Certified Network Professional Security
Skills 4
Unix and Linux
Experience 1 desc*
Junior web Developer
Experience 2 title
Experience 2
Experience 2 desc
Education 1*
Masters in Computer
Education 1 dur*
2 years
Education1 score*
Average of 9 Pointer

# Ivy Haddington

## SOFTWARE DEVELOPER

*Personal Profile*

Address: America

*Work Experience*

Software Engineer

*Feb '15 - Current*

Junior web Developer

*Key Skills*

Certified Scrum Master
SQL
Cisco Certified Network Profess
Security

## Conclusion:

The Online Resume Builder project, named **OnlineResumeBuilder**, has successfully met its objectives by creating a comprehensive platform for users to build and manage their resumes. Utilizing Django as the backend framework, the project provided a robust and scalable solution. The development process focused on creating a user-friendly interface that guides users through the resume-building process seamlessly. By leveraging Django's powerful features, the project efficiently handled backend operations, ensuring data integrity and security. Additionally, the implementation of efficient CRUD (Create, Read, Update, Delete) functionalities facilitated the effective management of user data and resume entries. The project was structured to allow for future enhancements and easy maintenance, ensuring its long-term usability. This project has not only provided a valuable tool for users to create professional resumes but also enhanced the developer's skills in web development, backend integration, and project management. The final submission, **OnlineResumeBuilder.** reflects the effort and dedication put into delivering a high-quality product.