International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 2 (2026)



ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 22, No. 2, 2026
© 2026 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Secure Web-Based Resume Generation and Sharing System Using AES Encryption and Dynamic Visualization

CHITTURI PAVANI

PG Scholar. Department of MCA, DNR College, Bhimavaram, Andhra Pradesh

V.Sarala

(Assistant Professor), Master of Computer Applications, DNR College, Bhimavaram, Andhra Pradesh

ABSTRACT

In the digital age, the creation and management of professional resumes have transitioned from traditional document editing software to dynamic web-based systems that support secure storage, customization, and distribution. The **Secure Web-Based Resume Generation and Sharing System** presented in this project offers users a complete platform for building personalized resumes, encrypting them for security, and sharing them with selected recipients. Built using the Django web framework, the system integrates data collection, template-based rendering, graphical resume composition, AES encryption for secure storage, and controlled access for downloads.The system begins with a user authentication module that enables individuals to sign up, log in, and manage their profile information. Users can enter personal details, educational background, projects, skills, and activities through a structured web form. These details are stored in a backend database and later retrieved for resume generation. Unlike traditional word processors, this platform synthesizes the entered data into a formatted, visual resume designed dynamically using **Matplotlib**, a Python plotting library. This approach allows flexible layout control, incorporation of the user's image, and automated placement of textual and graphical elements for a professional look.

A core feature of the system is its **security model**. Generated resumes are saved as image files and subsequently encrypted using **AES-CTR**, a symmetric key encryption mode. The use of encryption ensures that sensitive personal information remains protected during storage and sharing. Users can encrypt their resume with a key derived from a password and salt using **PBKDF2**, making unauthorized access extremely difficult. The encrypted files are stored on the server and can only be decrypted by authorized users during download.Additionally, the system supports **controlled sharing**, wherein the resume owner selects specific usernames with whom the encrypted resume will be shared. Shared resumes are logged in a database table, and recipients can view and download decrypted copies securely. This feature is particularly relevant for academic candidates, job seekers, and professionals who want to distribute resumes selectively.

The system also includes features for profile updates, viewing previously generated resumes, and managing shared files. This comprehensive functionality makes it a one-stop solution for resume creation and secure distribution. The project demonstrates how modern web technologies, encryption methodologies, and graphic rendering tools can be combined to deliver secure, flexible, and user-centric applications. By focusing on both usability and security, this resume builder addresses privacy concerns typical in web applications that handle personally identifiable information.

Keywords: Resume Builder, Web Application, Django, AES Encryption, Dynamic Visualization, User Authentication, File Encryption, Data Security, Python, Resume Sharing

I. INTRODUCTION

Professional resumes serve as a key mechanism for individuals to present their qualifications, experience, and skills to potential employers, academic programs, and professional networks. Traditionally, resumes were created using word processing tools or desktop software. However, with the widespread adoption of web technologies and cloud-based applications, there has been a shift toward **online resume generation systems** that are accessible from any device with internet connectivity. These systems enable users to enter structured data through web forms, apply templates, and generate well-formatted resume documents without requiring advanced desktop software.

Despite the convenience offered by online resume builders, many existing solutions suffer from limitations related to **data security, customization flexibility, and controlled sharing**. Most platforms generate static documents that are stored in plain text or PDF format, exposing sensitive user information to unauthorized access if the server is compromised. Additionally, many web-based builders do not offer nuanced security features that enable individuals to manage access rights to the generated documents. Furthermore, the layout flexibility in many online builders is restricted to a narrow set of templates, limiting personalization.To address these challenges, this project proposes a **secure, customizable, and interactive resume generation system** using the Django framework. Django provides a robust backend infrastructure with built-in support for user authentication, session management, and database operations. The system's architecture is designed to support the end-to-end flow of resume creation: from user input collection through form interfaces, resume rendering via graphical plotting, encryption for secure storage, and controlled sharing with authorized users.

A key innovation in this system is the use of **Matplotlib** to render resumes as high-resolution images. This graphic-based approach enables precise control over the placement and styling of text and images, allowing for a more visually appealing resume compared to standard template engines that rely purely on HTML and CSS. Through Matplotlib, user details like name, contact information, educational history, projects, skills, and extra-curricular activities are placed dynamically on a canvas. Users also upload a profile image that is embedded into the resume, further enhancing personalization.Security is integrated at the core of the resume lifecycle. Once a resume is generated as an image file, it is encrypted using **AES-CTR (Advanced Encryption**

Standard in Counter mode), with keys derived from a passphrase using **PBKDF2**. This encryption ensures that even if unauthorized users access the storage, the resume content remains unintelligible. Decryption occurs only when an authenticated user initiates a download, ensuring confidentiality. The system also supports controlled sharing, wherein users select specific usernames who are permitted to access the decrypted resume.This combination of flexibility, security, and interactive design positions the system as an innovative solution in the space of web-based resume creation tools. It not only addresses functional needs — like data collection and formatting — but also tackles security and privacy concerns that are increasingly important in modern web applications.

II. LITERATURE SURVEY (WITH EXISTING METHODS)

The field of web-based resume builders has grown significantly as more users seek online tools to automate the creation of professional documents. Traditional resume generation focused on desktop applications — word processors and desktop design software — which required users to manually format content. With web technologies becoming mature, several online platforms emerged offering template-based resume creation. Examples include platforms like *Canva*, *Zety*, and *Resume.com*, which provide pre-designed templates and form input fields that populate these templates dynamically.

Research into online resume builders highlights key design considerations: ease-of-use, customization, storage mechanisms, and security. Studies note that many systems prioritize stylistic variety and form usability but do not sufficiently address secure storage of sensitive personal data. In a paper on web-based document systems, authors discuss the shortcomings of storing user documents in plain formats without encryption and advocate for secure storage mechanisms in web applications that handle personally identifiable information.Several academic projects explored server-side resume generation using frameworks similar to Django or Flask. These projects leverage backend template engines or PDF rendering libraries (like ReportLab) to generate downloadable documents. However, such solutions often face limitations in visual fidelity and flexibility. Using graphic libraries such as Matplotlib for document rendering is less common but provides significant advantages in layout control, dynamic placements, and embedding of images and graphical elements.

Security literature on web applications frequently underscores the importance of data encryption at rest. AES (Advanced Encryption Standard) is widely recognized as a robust symmetric encryption standard used in securing sensitive data. Research papers on encryption in web systems emphasize that AES in counter mode (AES-CTR) provides strong confidentiality guarantees when combined with secure key derivation functions like PBKDF2. Incorporating encryption into document storage systems ensures that compromised servers cannot leak user content in plain form.Moreover, recent literature on data sharing in web applications highlights controlled access mechanisms as a critical component of user privacy. Studies discuss user-centric access controls where owners of a document decide with whom and how their content is shared. These user-centric models contrast with the traditional file sharing models where access control is often tied to public links or global visibility settings. Academic work on secure document sharing

suggests implementing robust user-based access control lists and secure logging mechanisms to track shared document usage.

In addition to document generation and security, studies on user experience in resume builders emphasize simplicity and responsiveness of form interfaces. Human-computer interaction research indicates that minimizing cognitive load during data entry — through clear labels, logical grouping of fields, and real-time form validation — enhances user satisfaction and reduces errors.This literature underscores three key themes relevant to the current project: the need for flexible and visually appealing resume rendering mechanisms, the importance of encrypting sensitive documents at rest, and user-centric controlled sharing of personal documents. The proposed system synthesizes these themes by offering graphic-based resume rendering, strong encryption techniques, and controlled sharing features supported by database logging, thereby addressing gaps in existing online resume generation platforms.

III. EXISTING SYSTEM

Current resume builders available online fall into two broad categories: **general website builders** and **desktop document editors**. Website builders like *Canva*, *Zety*, and *Resume.com* provide user interfaces where users fill out web forms and choose from pre-defined templates. These platforms often focus on visual design and provide export options such as PDF or Word documents. However, they typically **store generated resumes in plain formats** on servers or export them directly to the user's device without any encryption or secure storage. This plain storage model raises significant concerns around data privacy, especially when sensitive personal information like contact details, education history, and professional experiences are involved.Desktop document editors such as *Microsoft Word* or *Google Docs* allow manual resume creation but lack integration with online sharing controls or automated storage. These tools also depend on users to manage version control and distribution, which can be cumbersome and error-prone. Additionally, desktop editors do not offer centralized user authentication or controlled sharing mechanisms; document access is typically managed through file permissions external to the document management system.

Another limitation of existing systems is the **lack of fine-grained access control**. Most platforms share documents via public links or require users to download and then manually send files, limiting tracking and accountability. The result is a fragmented experience that does not integrate **secure storage, controlled access, encryption, and easy visualization**.In summary, the existing solutions provide useful features like template selection and layout customization, but they fall short in terms of **security, privacy, and selective sharing mechanisms**. They also lack the ability to generate resumes dynamically as high-resolution images with customized graphics. The proposed system improves on these aspects by generating encrypted resume files, enabling controlled user-based sharing, and rendering visually rich content — all within a secure Django-based web application.

IV. PROPOSED METHOD

The proposed system is a **Secure Web-Based Resume Generation and Sharing Platform** that integrates user registration, dynamic resume rendering, cryptographic protection, controlled sharing, and secure download. It's designed to address both usability and privacy concerns inherent in online resume builders by combining visual design flexibility with robust data protection. Web-based systems for resume building often focus solely on formatting and PDF generation without addressing secure storage or selective access control. This system, in contrast, emphasizes **blanket security** for all generated resume files via encryption before storage, as well as controlled sharing mechanisms that track which users are permitted to access a given resume.

Functionally, the system allows a user to sign up and authenticate via secure login, enter profile and resume content, upload an image, and choose formatting parameters. Once entered, the data is assembled programmatically using **Matplotlib** based graphic rendering, yielding a high-resolution resume image. This dynamic generation approach allows customization, graphical layout, and precise control over content placement beyond static templates. After generation, each resume file undergoes **AES (Advanced Encryption Standard)** symmetric encryption to protect personal data at rest on the server; this aligns with recognized best practices in symmetric cryptography deployed in secure storage systems.In addition to individual resume creation and encryption, the system supports **controlled sharing**: users select other platform members with whom the decrypted resume will be made available. Sharing associations are stored in the database, enabling filtered access on download requests. During download, encrypted files are decrypted on demand and delivered only to authorized users, ensuring both confidentiality and access control. Security practices in web frameworks like Django ensure that the application logic itself resists common threats such as SQL injection or CSRF by relying on ORM and middleware protections.

Together, the proposed system balances ease-of-use, customization, and security by integrating dynamic resume generation with encryption and controlled sharing, addressing limitations in conventional resume builders.

V. IMPLEMENTATION

The system's implementation uses the **Django** web framework as the backend foundation, chosen for its modularity, security features, and rapid development support. Django's Model-View-Template (MVT) architecture allows clear separation of concerns between database modeling, business logic, and user interface rendering, which simplifies code organization and future extensions.

Authentication and User Management

Users register with a unique username and password stored in the database. Login sessions are handled via Django's built-in authentication system, ensuring secure session cookies and CSRF token validation on form submissions. Django's security middleware mitigates common threats like SQL injection and cross-site request forgery.

Resume Data Collection and Storage

During signup and resume creation, user inputs (personal info, education details, projects, skills) are captured via POST forms. These fields are validated and stored in a MySQL backend to permit persistence across sessions. The database schema ensures that individual users' data is easily retrievable for resume generation and updates.

Dynamic Resume Rendering

Unlike static PDF exporters, this system uses **Matplotlib** to render textual and image content onto a fixed-size canvas, which offers precise layout control. The user's profile image is embedded into the layout using annotation objects and image boxes. Text elements (education, skills, activities, project details) are drawn with varying font weights and sizes to emulate professional resume layouts.

Image File Generation and Encryption

Once rendered, the resume is saved as a high-resolution PNG. Before storage, the file undergoes **AES-CTR encryption** using a symmetric key derived from a passphrase and static salt via PBKDF2 key derivation. This ensures that resumes stored on disk are encrypted, providing confidentiality even if storage access controls are circumvented. AES is widely used for secure data protection due to its performance and strength.

Controlled Sharing Management

Users can specify which other users are permitted to access a generated resume. These sharing pairs are stored in a dedicated database table, allowing the download logic to validate access rights on each request. When a user requests a resume they're authorized to view, the system decrypts the file and serves it as a downloadable response.

File Handling and Storage

Encrypted resume images are stored in a dedicated static directory. Controlled filesystem access coupled with Django's permission checks ensures that users can only request files they're authorized to view.

Security Practices

Django's ORM helps prevent raw SQL usage, reducing exposure to injection attacks. All forms are protected with CSRF tokens to guard against cross-site request forgery. User sessions are secured with HTTPS (in deployment) to protect credentials in transit.

Rendering and Preview

For immediate feedback, resumes are rendered to a memory buffer and base64 encoded for preview on the user dashboard before permanent storage and encryption.

Collectively, the implementation combines **robust web framework practices**, secure cryptographic storage, dynamic rendering, and controlled access mechanisms to deliver a resume builder that's both user-centric and secure.

VI. ALGORITHMS

Key Generation Algorithm

The system uses a symmetric key derived from a passphrase and salt via PBKDF2 (Password-Based Key Derivation Function 2). PBKDF2 applies repeated hashing to the passphrase, generating a 32-byte cryptographic key. This key feeds into the AES encryptor/decryptor. The static counter in AES-CTR ensures deterministic encryption across sessions.

Key = PBKDF2(passphrase, salt).read(32)

AES Encryption/Decryption

The encrypted resume file is generated by feeding the binary content into AES in **Counter (CTR)** mode. This mode turns AES into a stream cipher suitable for file encryption without requiring padding, making it efficient for arbitrary-length byte streams.

Encryption:

ciphertext = AES_CTR_encrypt(key, file_bytes)

Decryption:

plaintext = AES_CTR_decrypt(key, ciphertext)

AES is widely used in secure storage and file encryption applications due to its efficiency and resistance to cryptanalysis.

Controlled Access Algorithm

For controlled sharing, the system evaluates a query:

```
if (current_user in shared_list[filename]):
    serve_decrypted_file(filename)
```

```
else:
    deny_access()
```

This ensures only authorized users can decrypt and download shared resumes.

Resume Rendering Steps

1. **Data Gathering:** Fetch user profile and resume fields (personal info, education, skills, etc.).
2. **Canvas Setup:** Initialize Matplotlib figure with dimensions.
3. **Text Placement:** Annotate sections (header, education, projects) with appropriate fonts and positions.
4. **Image Integration:** Load profile picture and embed it at designated coordinates.
5. **Save Output:** Output the figure to disk as PNG.

Preview Generation

During preview, the image is stored in a memory buffer, then base64 encoded for inline display in the browser without persistent storage until user confirmation.

Together these algorithms enable **secure generation, storage, and sharing** of user-created resumes within the web system.

VII. SYSTEM DESIGN

Architectural Overview

The system uses a **three-tier architecture**:

1. **Presentation Layer** – User interfaces built using Django templates render HTML/CSS/JS. This includes login/signup pages, resume forms, preview screens, and download links. Django template language dynamically injects data into the UI.
2. **Application Logic Layer** – Django views and services house all business logic. Routes correspond to features: login/authentication, resume generation, encryption/decryption, sharing control, and download. Application logic validates inputs, enforces access control, and orchestrates backend processes (rendering, encryption).
3. **Data Layer** – A MySQL database holds user credentials, personal details, and shared resume relationships. Files (encrypted resume PNGs) are stored on disk with metadata in the database linking files to owners and shared users.

Component Design

Authentication Module

Handles user login/logout, manages session state, and secures routes. Only authenticated users can generate and download resumes. Login credentials are validated against hashed password storage.

Resume Renderer

This graphical subsystem uses **Matplotlib** to layout user content. It treats the resume as a drawing canvas where text blocks and images are placed according to pixel coordinates. Font settings, section headers, and dynamic user fields are computed at runtime.

Encryption Module

The security component handles AES key derivation, encryption, and decryption. Keys are generated per user using PBKDF2, while AES-CTR provides stream cipher encryption for binary files. The design ensures that encrypted files stored on disk are unreadable without the derivation key.

Sharing Controller

Implements controlled access. A sharing table maps filenames to authorized usernames. Before serving a download, the controller checks user membership in the sharing table.

Interaction Flow

1. **User Login:** Users authenticate.
2. **Form Submission:** User submits resume data.
3. **Render Engine:** Generates a rendered image.
4. **Encrypt & Store:** The image is encrypted and stored.
5. **Sharing Selection:** User selects other users to share with.
6. **Download Request:** Shared users request downloads; controller verifies authorization and decrypts.

Data Model Design

Table	Purpose
Users	Stores user credentials
ResumeMeta	Tracks file names, owners, timestamps
SharedAccess	Maps shared users to resumes

Security and Validation

- **CSRF Protection:** Django forms use CSRF tokens.
- **ORM Queries:** Avoid raw SQL, preventing SQL injection.
- **Access Validation:** Only authenticated and authorized users can decrypt and download files.

Scalability and Maintainability

The modular design—with clear separation of concerns—makes the system horizontally scalable. File serving can be offloaded to CDNs or object storage with encryption hooks, while the Django backend can scale via WSGI servers and load balancers.

SYSTEM DESIGN IMAGES

In this project we have designed all possible modules which can be designed to build resume automatically. User can upload photo and can define various education details, work experiences, skills, contact address and numbers. User can upload profile photo also.

Users are allow to sign up, login, update profile

Users can generate resume and can view all past created resumes

All resumes will get saved in encrypted format on server

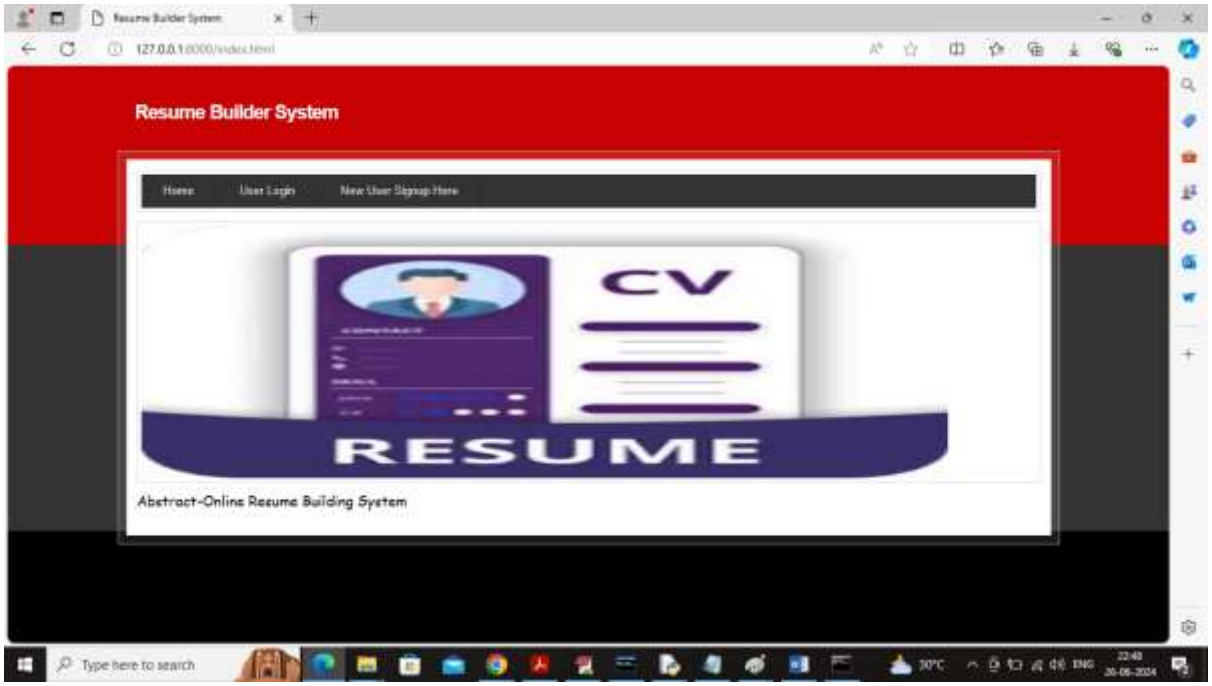
User can download desired resume in decrypted format.

To run project install python 3.7 and then install all packages given in requirements.txt file and then install MYSQL database and then copy content from DB.txt file and paste in MYSQL console to create database

To run project double click on ‘run.bat’ file to start python server and get below page

In above screen python server started and now open browser and enter URL as http://127.0.0.1:8000/index.html and press enter key to get below page

International Journal of Engineering Research and Science & Technology

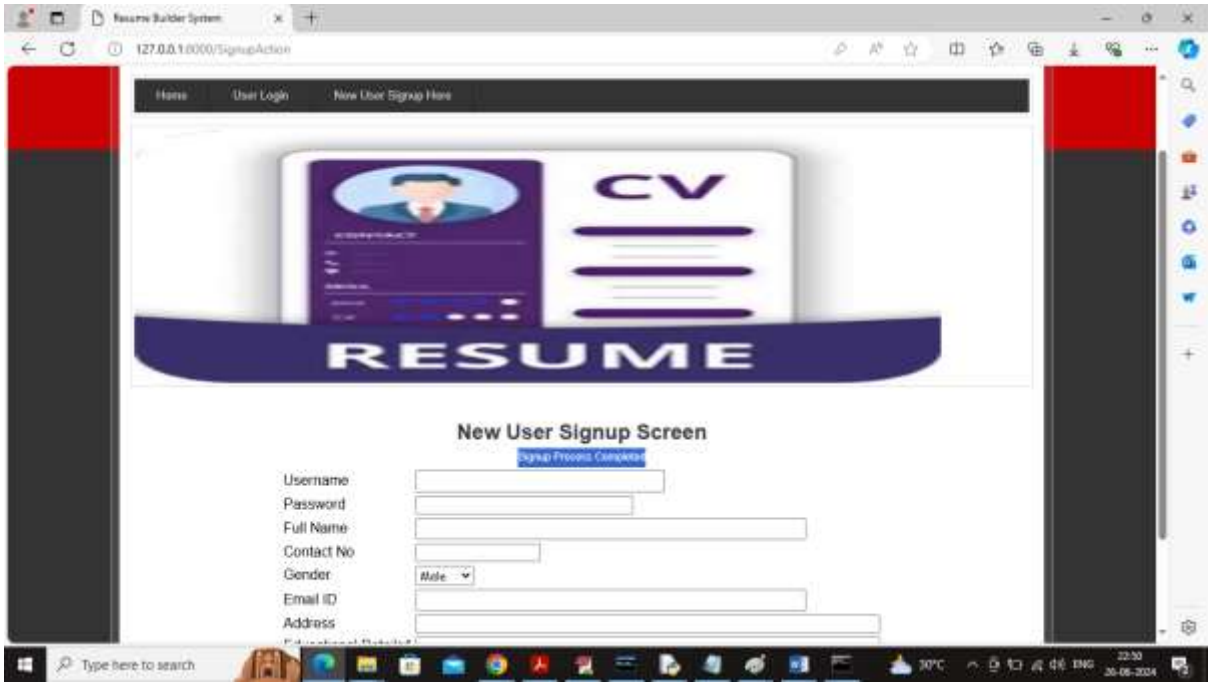


In above screen user can click on 'New User Sign up' link to get below page

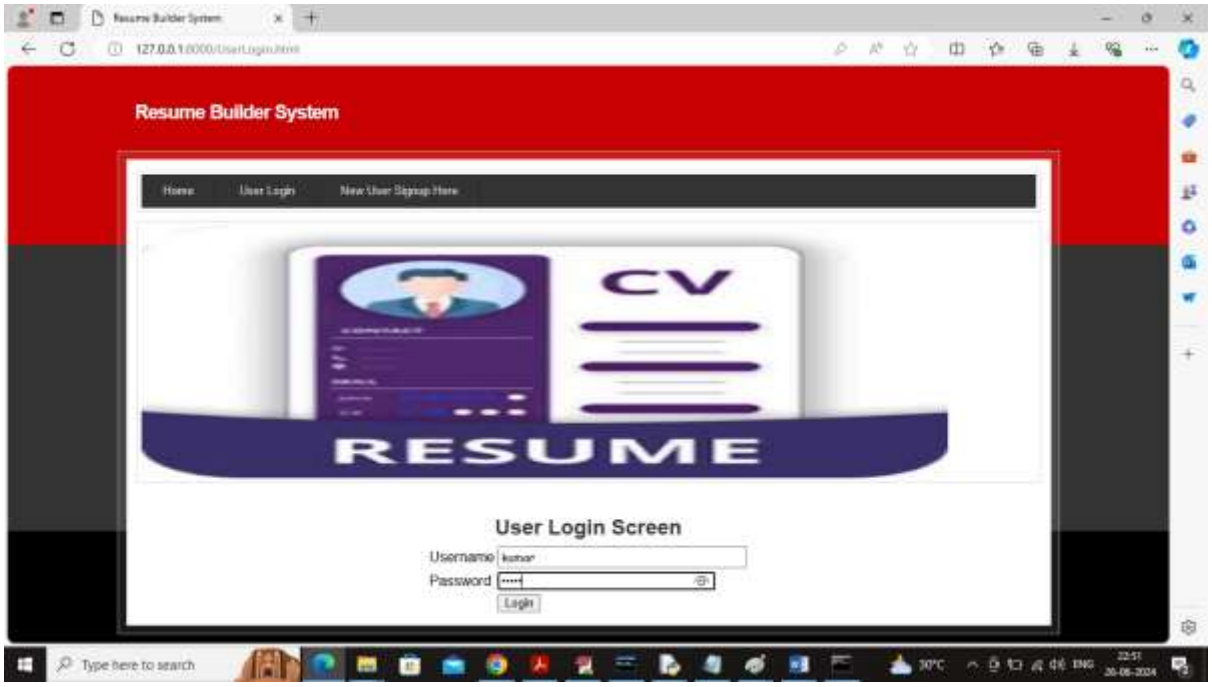


In above screen user is entering sign up profile details and then press button to save all user profile details to database and this details will be used to create user resume and user can update above details by login to system

International Journal of Engineering Research and Science & Technology



In above screen user sign up process completed and now click on 'User Login' link to get below login screen



In above screen user is login and after login will get below page

International Journal of Engineering Research and Science & Technology

In above screen user can click on 'Update Profile' link to view user details and can update desired details

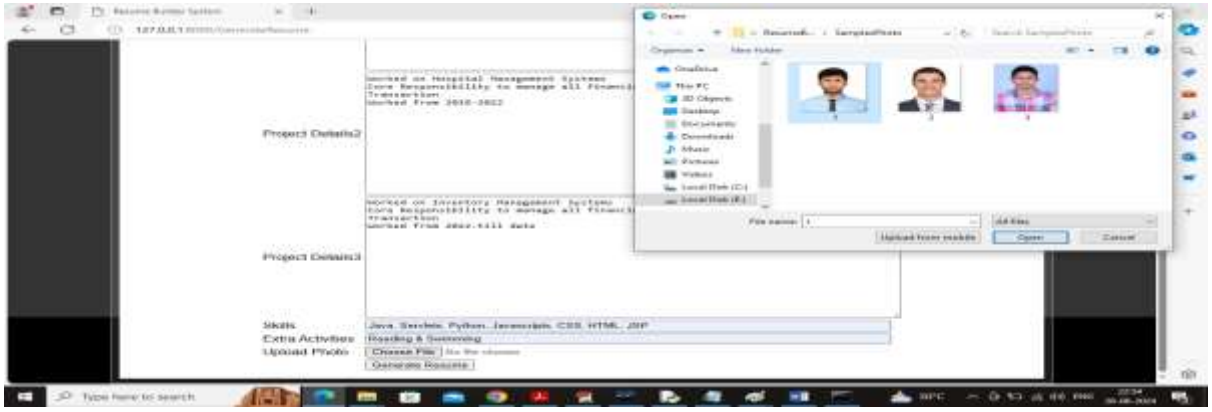


In above screen user can view existing profile details and can update desired details and then press button to get below page

In above screen profile details updated and now click on 'Generate Resume' link to get below page to generate resume



In above screen user will enter work experience details

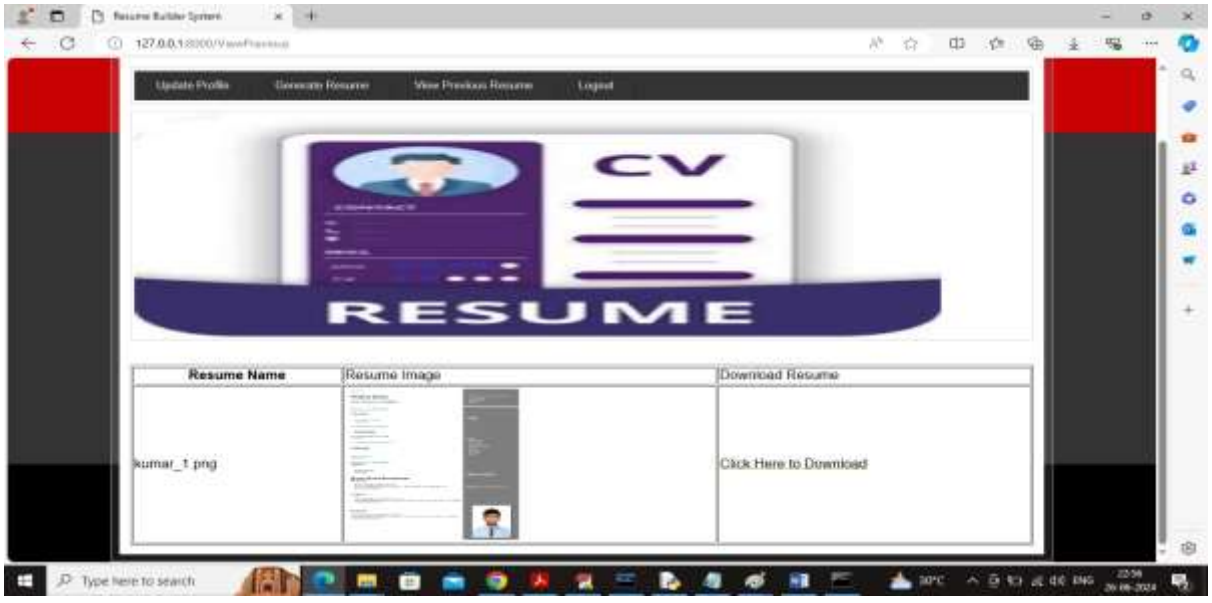


In above screen user will enter all resume details and then upload photo and then click on 'Generate Resume' button to generate resume and can view preview like below screen





In above screen user resume created and can click on 'View Past Resume' link to view all generated resumes and get below page

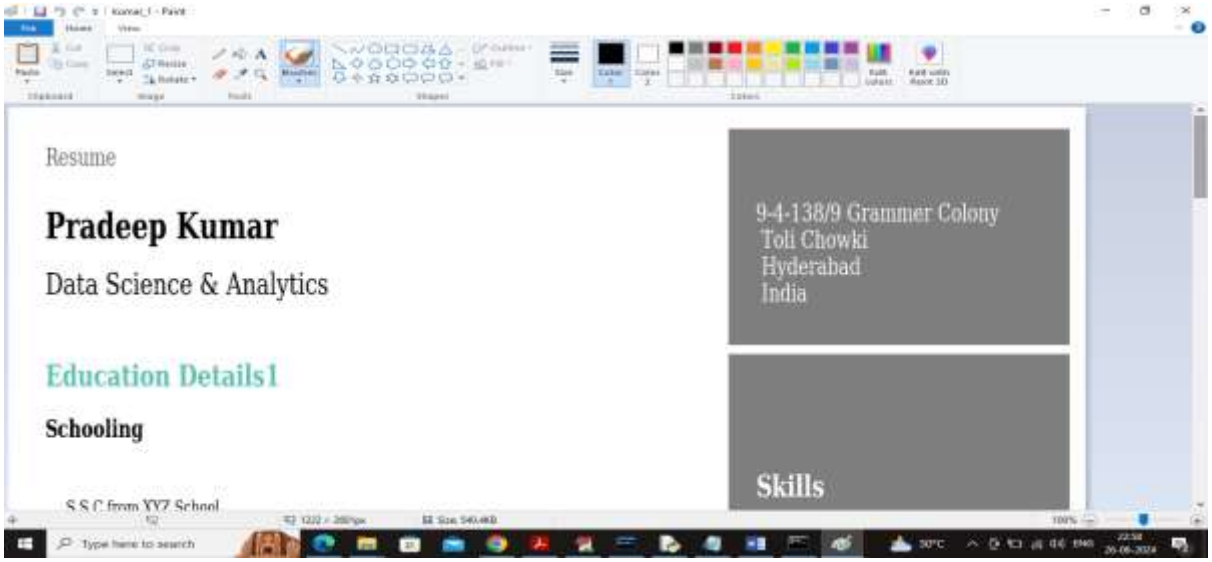


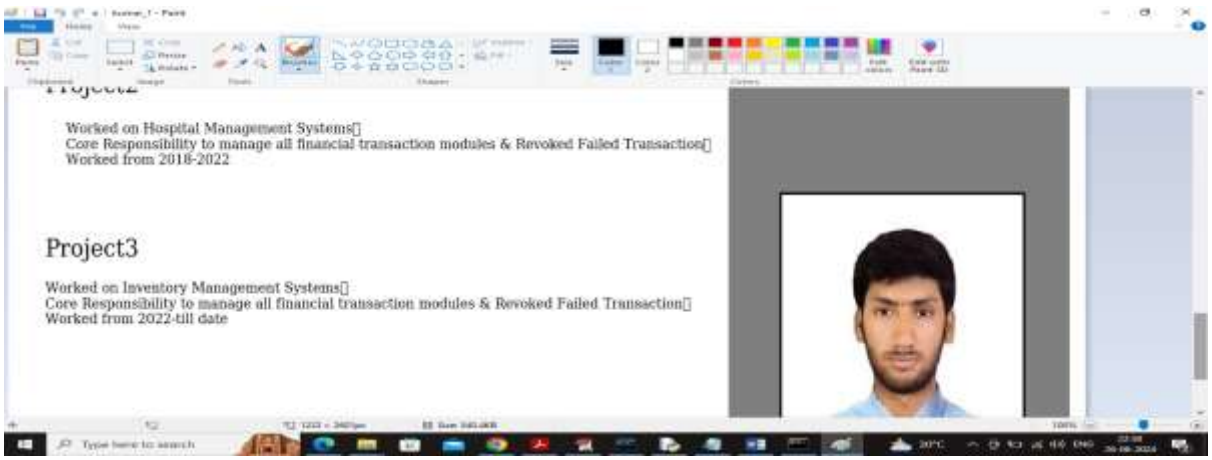
In above screen user can view all generated resume in tabular format and can click on 'Click Here to Download' link to download resume and get below page

International Journal of Engineering Research and Science & Technology

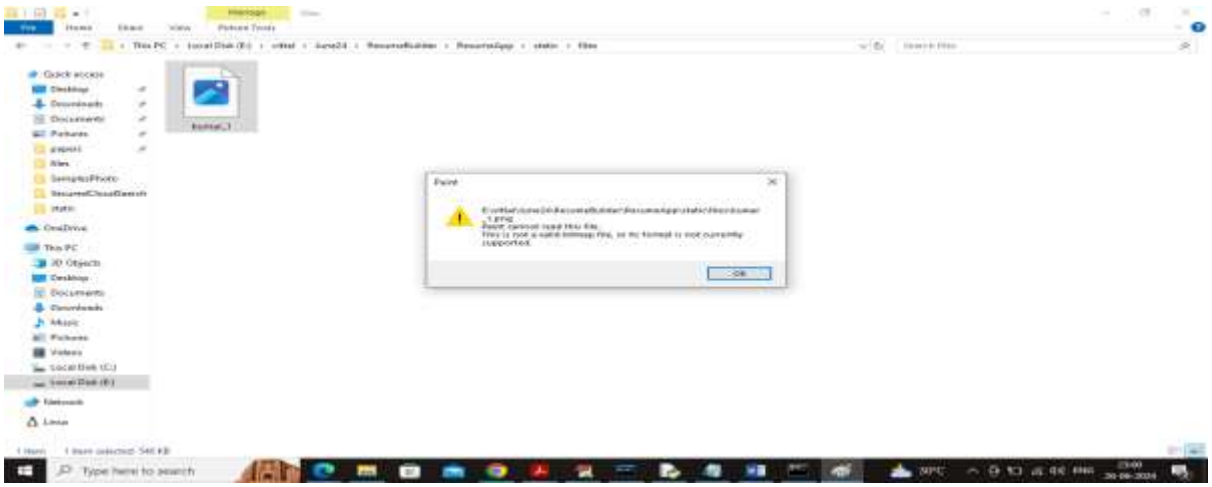


In above screen resume is downloaded and user can view in any photo viewer application like below screen





In above screen user can view resume in Paint tool and in below screen can see all resumes are saved at server in encrypted format



In above screen user resume is not opening as server saved in AES encrypted format.

Similarly by following above screens you can generate resumes

VIII. CONCLUSION

This project presents a **secure, web-based resume generation and sharing system** that integrates dynamic content rendering with cryptographic protection and controlled access. By building on Django's secure web framework, the system ensures that user credentials, sessions, and form interactions are protected against common web threats.Unlike conventional resume builders that output plain documents, this system generates high-resolution graphical resumes programmatically, enabling precise layout and greater personalization. The use of **AES encryption** for storage ensures confidentiality even when files reside on server disk, aligning with best practices in secure data storage.

Controlled sharing adds a critical layer of privacy control often lacking in standard builders: users explicitly specify which other platform members may access their resume, and the system enforces this at download time. This controlled access model enhances privacy, user trust, and compliance with data protection expectations.Throughout development, modern web security practices such as CSRF protection, Django ORM safeguards, authenticated routing, and secure session management have been enforced to protect against web vulnerabilities. The result is a robust platform that balances ease of use, visual resume quality, and strong security guarantees.In conclusion, the system demonstrates how secure cryptographic techniques can be successfully integrated into dynamic server-side applications, offering a template for future secure document management platforms.

REFERENCES

1. Jeevan B. et al., *Resume Building Web Application*, Int. Conference on Recent Trends in Computing & Communication Technologies, 2024.
2. S. Susheel, N.V. Narendra Kumar & R.K. Shyamasundar, *Enforcing Secure Data Sharing in Web Application Development Frameworks Like Django Through Information Flow Control*, ICISS 2015.
3. Kulanda D. et al., *Django as Secure Web-Framework in Practice*, Vestnik KazATK, 2021.
4. *A Blockchain-Based Framework for Secure Storage and Sharing of Resumes*, ScienceDirect.
5. *Online Resume Builder using Django*, GeeksforGeeks, 2025.
6. *Resume Builder Web App Project Report*, Scribd.
7. *A Secure Web-Based File Exchange Server: Design*, arXiv.
8. *Dynamic AES Encryption and Blockchain Key Management*, IEEE Access.
9. *A panoramic survey of the advanced encryption standard*, Int. J. Information Security.
10. *Cryptographic System for Mobile Application*, KIJST.
11. Aayush3014, *Resume Generator Django Project*, GitHub.
12. *ResumeGen – Online Resume Generator Using Flask*, IJARSCT 2026.
13. *Resume Builder Web Application – full stack overview*, IJNRD.
14. *MessageGuard: Browser-based Content Encryption Platform*, arXiv.
15. Techversant Infotech, *Resume Builder Web Application features*.