RESUME BUILDER WEB APP USING AWS

Prepared in the partial fulfillment of the Summer Internship Program on AWS

AT

Under the guidance of

Mrs. Sumana Bethala, APSSDC

Mr. Anil , APSSDC

Submitted by

Akasapu Akhil, 24P35A0501,

Chikoti satish, 24P35A0509,

D lakshmi prasad, 24P35A0515,

Bendalampotti Sivani, 23P31A04D9,

Vinay Nethala, 24A95A0513

(July, 2025)

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all those who have contributed to the successful completion of my summer internship project at **Andhra Pradesh Skill Development Corporation (APSSDC)**. This opportunity has been an enriching and transformative experience for me, and I am truly thankful for the support, guidance, and encouragement I have received along the way.

First and foremost, I extend my sincere regards to Mrs Sumana Bethala and Mr Rama Krishna, my supervisors and mentors, for providing me with valuable insights, constant guidance, and unwavering support throughout the duration of the internship. Their expertise and encouragement have been instrumental in shaping the direction of this project.

I would like to thank the entire team at **Andhra Pradesh Skill Development Corporation (APSSDC)** for fostering a collaborative and innovative environment. The camaraderie, knowledge sharing, and feedback I received from my colleagues significantly contributed to the development and success of this project.

In conclusion, I am honoured to have been a part of this internship program, and I look forward to leveraging the skills and knowledge gained to contribute positively to future endeavours.

Thankyou,

Sincerely,

Akasapu Akhil, 24P35A0501,

Chikoti satish, 24P35A0515,

D lakshmi prasad, 24P35A0515,

Bendalampotti Sivani, 23P31A04D9,

Vinay Nethala, 24A95A0513

ABSTRACT

In the current digital age, the importance of a well-structured and professional resume cannot be overstated. With job markets becoming increasingly competitive, having access to efficient tools that aid in resume creation has become a necessity. This project, **"Resume Builder Web App Using AWS"**, aims to provide users with a user-friendly, secure, and highly available platform to create, manage, and download personalized resumes effortlessly. By leveraging the capabilities of **Amazon Web Services (AWS)**, this web application ensures scalable performance, robust data handling, and minimal downtime.

The core functionality of the application revolves around a dynamic web interface where users can input their personal information, education background, professional experience, skills, and other relevant data. The application then formats this input into professionally styled resume templates that users can preview and download in PDF format. To enhance user experience, the interface is developed using modern web technologies such as **HTML5, CSS3, JavaScript, and React.js**, ensuring responsiveness and interactivity across devices.

On the backend, **AWS services** such as **Amazon S3**, **AWS Lambda**, **Amazon RDS (or DynamoDB)**, **API Gateway**, and **Cognito** are integrated to handle file storage, data processing, database management, authentication, and API interactions. AWS Lambda enables a serverless architecture, allowing automatic scaling and reduced operational costs. User authentication and security are managed via AWS Cognito, which provides secure sign-up, sign-in, and access control features. Amazon S3 is used to store resume templates and user-generated PDF files reliably.

The application is deployed using **AWS Amplify** or **Elastic Beanstalk**, providing seamless CI/CD (Continuous Integration and Deployment) and easy scalability. This cloud-based deployment ensures high availability, low latency, and a global reach, making the application accessible to users around the world at any time.

In conclusion, this Resume Builder Web App not only demonstrates the practical integration of front-end development with robust AWS cloud services but also addresses a real-world need for accessible and professional resume creation tools. It showcases how cloud computing can be harnessed to build scalable, efficient, and user-centric web applications.

In future versions, the platform can be enhanced by integrating AI-powered resume suggestions, job matching features, and multilingual support, making it an even more valuable resource for global users.

ABSTRACT
TABLE OF CONTENTS

INTRODUCTION

In the modern job market, first impressions often begin not in person, but on paper — more specifically, through resumes. A resume is a critical tool in job hunting, serving as a concise representation of an individual's qualifications, experience, skills, and achievements. However, many job seekers struggle with resume formatting, design consistency, and tailoring content to specific job roles. With increasing digitization and the availability of cloud services, there is an emerging demand for automated tools that simplify this process and help users create professional, visually appealing resumes without requiring design expertise.

This project, titled **Resume Builder Web Application using AWS (Amazon Web Services)**, addresses this need by offering a streamlined, interactive platform where users can input their details and generate custom resumes in real-time. The web app is cloud-native and serverless, leveraging a range of AWS services to ensure scalability, performance, reliability, and cost-efficiency. The application allows users to securely store, edit, and download resumes, as well as choose from a set of professionally designed templates that suit different career paths or preferences.

At its core, the Resume Builder App consists of three main components — the frontend (user interface), the backend (business logic and resume generation), and cloud-based services (database, storage, and authentication). The frontend is designed with user experience in mind, utilizing responsive HTML, CSS, and JavaScript, optionally enhanced by frameworks like React.js or Vue.js for more dynamic interactions. This frontend is hosted on **Amazon S3**, taking advantage of S3's static website hosting capabilities. To ensure fast load times across geographic regions, **Amazon CloudFront** is integrated as a CDN (Content Delivery Network), which caches content closer to users

The backend of the application is built using **AWS Lambda**, a serverless compute service that allows developers to run code in response to user events or API calls without provisioning or managing servers. This stateless architecture not only reduces infrastructure overhead but also supports automatic scaling, which is ideal for handling unpredictable workloads. These Lambda functions are exposed through **Amazon API Gateway**, which acts as a secure and scalable API endpoint for frontend-backend communication.

Security and user management are integral parts of any modern web application. For this purpose, **Amazon Cognito** is used to handle user authentication and authorization. Cognito enables features like user registration, login, password recovery, and social login integration (e.g., Google, Facebook) while adhering to best security practices like encryption and token-based authentication.

One of the most important features of the app is the **resume generation module**. Once users input their information, the backend formats this data into a selected template using libraries like Puppeteer, pdfkit, or jsPDF, converting it into a professional-grade PDF file. These files can be made available for direct download or stored temporarily in an S3 bucket for user access. The formatting process ensures that resumes maintain consistency across different devices and platforms.

The decision to use AWS as the underlying infrastructure offers several advantages. AWS provides high availability and disaster recovery options by default, ensuring that the application remains operational even under stress. The pay-as-you-go model is cost-effective, especially for startups or developers working on limited budgets, as it avoids the upfront costs associated with traditional server hosting. Additionally, AWS offers monitoring, logging, and alerting tools such as **Amazon CloudWatch**, which help developers keep track of application performance and user activity.

Furthermore, by following a **microservices-based, loosely coupled architecture**, the app is easier to maintain, test, and expand. New features, such as AI-powered resume suggestions or integration with job boards, can be added without disrupting the existing system.
In summary, the **Resume Builder Web App using AWS** exemplifies the modern approach to web application development by combining serverless architecture, user-focused design, and the robustness of cloud computing. It simplifies the resume creation process for users while maintaining high standards of performance, scalability, and security. Beyond its functional purpose, this project also serves as an educational showcase of how various AWS services can be orchestrated to build a complete, real-world cloud application.

2 METHODOLOGY

The development of the Resume Builder Web Application using AWS followed a structured, iterative, and cloud-native methodology. This approach ensured the efficient fulfillment of project requirements while maintaining scalability, performance, and a positive user experience. The development lifecycle was divided into distinct phases, including requirements gathering, design, implementation, database setup, testing, deployment, and iterative refinement. Each phase played a vital role in the realization of the final product. The following sections outline these phases in detail.

2.1. Requirements Gathering

The project was initiated with an extensive requirement analysis to identify the core functionalities and expectations of the Resume Builder Web App. Discussions were held with end users, academic mentors, and peers to gain insights into essential features such as user data entry, template selection, PDF generation, and user authentication. The requirements gathering process also involved analyzing similar existing platforms to identify gaps and opportunities for innovation.

2.2. Design

Based on the requirements, a comprehensive system design was developed, focusing on modularity, scalability, and user experience. The architecture of the application was planned to include a serverless backend, cloud-hosted frontend, and secure database storage. The frontend design prioritized an intuitive interface, responsive layout, and minimalistic design principles to support seamless use across desktops, tablets, and smartphones. Diagrams and flowcharts were created to illustrate the interactions between the frontend, backend APIs, authentication layer, and database.

2.3. Implementation

The implementation phase involved transforming the design into a working application using cloud-native technologies and modern web development practices. The frontend was developed using HTML, CSS, and JavaScript, ensuring compatibility and responsiveness. Amazon S3 was used for static hosting of frontend files. Backend logic was implemented using AWS Lambda functions written in Python, triggered via Amazon API Gateway. These functions handled user inputs, resume data processing, and PDF generation. Integration with libraries like pdfkit enabled dynamic document creation. Authentication functionality was optionally implemented using Amazon Cognito to manage user sessions securely.

2.4. Database Design

To manage and store user data efficiently, a NoSQL database schema was designed using Amazon DynamoDB. The schema included tables for user profiles, resume content, and metadata related to template selection and file storage. In cases requiring relational data or complex querying, Amazon RDS with MySQL was considered as an alternative. The database design focused on scalability, quick access times, and seamless integration with AWS Lambda.

2.5. Testing

Testing was conducted at various stages to ensure reliability, performance, and functionality. This included:

Unit Testing: Individual Lambda functions were tested for correct behavior.

Integration Testing: Verified seamless interaction between frontend, backend, and the database.

User Acceptance Testing (UAT): Real users interacted with the system to validate functionality, layout, and usability. Feedback was gathered to identify any user experience issues or bugs, which were addressed promptly.

Special attention was given to testing PDF generation under different inputs and ensuring consistent template rendering.

2.6. Deployment

The deployment phase involved configuring and launching the application on AWS. Static frontend content was deployed on Amazon S3 and distributed via Amazon CloudFront to ensure fast access globally. Backend functions were deployed using AWS Lambda, with API Gateway acting as the interface. User data was stored in DynamoDB, and temporary resume files were stored in Amazon S3. Security best practices were followed, including setting IAM roles, enabling HTTPS, and applying cross-origin resource sharing (CORS) policies. Deployment automation tools like AWS SAM (Serverless Application Model) were considered to simplify Lambda deployments.

2.7. Iterative Refinement

An iterative development model was followed throughout the lifecycle of the project. Regular feedback loops with mentors, peers, and potential users guided incremental improvements. Each iteration focused on adding new features, enhancing performance, or addressing bugs. Continuous testing, monitoring, and optimization helped ensure the system's reliability and alignment with user expectations.

.

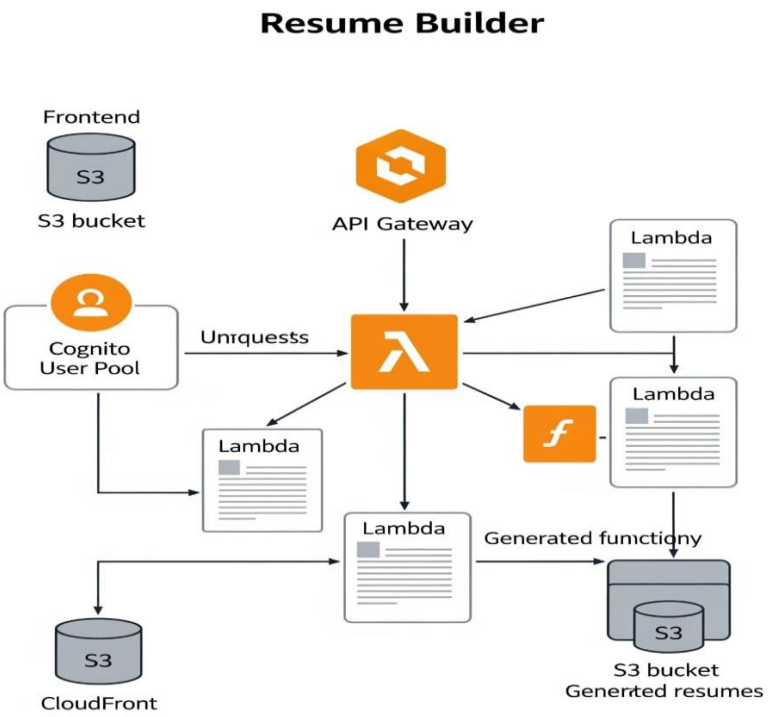
3. SYSTEM DESIGN / ARCHITECTURE

The **Resume Builder Web Application** is designed as a cloud-based, serverless web platform that enables users to create, customize, and download professional resumes. The system is architected using a **three-tier model**, comprising the **presentation layer**, **application logic layer**, and **data storage layer**. AWS services are utilized extensively to ensure the application is **scalable, secure, and cost-effective**, supporting a seamless and responsive user experience.

3.1Presentation Layer

The presentation layer is the user-facing component of the application. It is responsible for displaying the user interface, capturing user input, and rendering the final output in an interactive and accessible format. This layer is designed using **HTML5**, **CSS3**, and **JavaScript**, with optional use of frontend frameworks such as **React.js** for enhanced interactivity. The interface is hosted on **Amazon S3** as a static website and distributed globally through **Amazon CloudFront** for optimized performance.

Figure 3.1: Flowchart of Presentation Layer



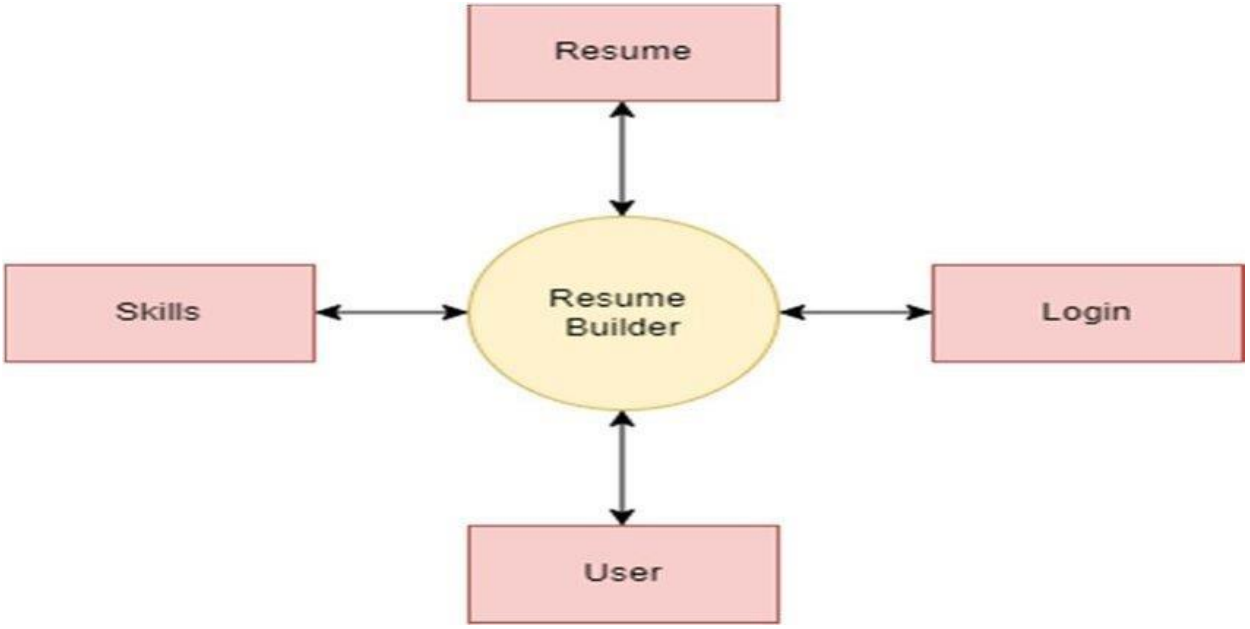
Key Features of the Presentation Layer:

- **Resume Input Form:**
  A clean, form-based interface allows users to enter personal information, educational background, work experience, skills, and other relevant data.
- **Template Selection:**
  Users can choose from a set of predefined, visually appealing templates that determine the structure and design of the resume.
- **Live Preview & Editing:**
  Real-time rendering of the resume allows users to preview their changes before generating the final PDF document.
- **Responsive Design:**
  The interface is optimized for all device types (desktop, tablet, mobile) using responsive web design principles.
- **Download Option:**
  A clearly visible button allows users to generate and download their resume in PDF format upon completion.

3.2 Application Logic Layer

The application logic layer contains the backend functionality that handles data processing, template rendering, and file generation. This layer is **serverless** and is built using **AWS Lambda** functions written in **Python**, triggered via **Amazon API Gateway**. It acts as the intermediary between the frontend and the data storage layer.

Figure 1.2: Flowchart of application layer



Key Components of the Application Logic Layer:

The Application Logic Layer is the central layer responsible for processing user data, applying business logic, rendering templates, and generating output files (PDF resumes). It connects the frontend with the backend infrastructure and handles most of the dynamic operations of the application. Below are the key components involved:

1. Amazon API Gateway

- Role: Acts as the entry point for all frontend requests.
- Function: Routes HTTP requests from the frontend to the appropriate backend Lambda function.
- Features: Provides request throttling, monitoring, and security via API keys or Cognito integration.

2. AWS Lambda

- Role: Core execution environment for backend logic.
- Function: Executes backend code (Python/Node.js) in response to events triggered by API Gateway.
- Serverless Advantage: No need to manage servers; automatically scales with traffic.
- Key Tasks Performed:
    - Accepting and validating user data.
    - Formatting data for template rendering.
    - Triggering PDF generation.
    - Interfacing with database/storage layers.

3. Template Renderer

- Role: Dynamically merges user data with predefined HTML templates.
- Function: Populates placeholders (like name, skills, education) in HTML/CSS resume templates with actual data provided by the user.
- Technologies: May use Jinja2 (Python), Handlebars (Node.js), or string templating logic.

4. PDF Generation Module

- Role: Converts rendered HTML resume into a downloadable PDF.
- Technologies Used:
    - pdfkit (Python) or WeasyPrint
    - Puppeteer (Node.js) – for headless Chromium rendering
- Output: Generates a high-quality, printable resume PDF.

5. Amazon Cognito (Optional but Recommended)

- Role: Provides authentication and user management.
- Function: Handles user sign-up, login, and session management.
- Benefit: Ensures only authorized users can access and manage their resumes.

6. Communication with Data Layer

- Data Access: Lambda functions interact with the data storage layer (e.g., DynamoDB, S3).
- Function: Fetching, updating, or saving user profile data, resume content, or generated PDFs.
- Security: IAM roles restrict Lambda functions to access only required resources.

7. Response Handler

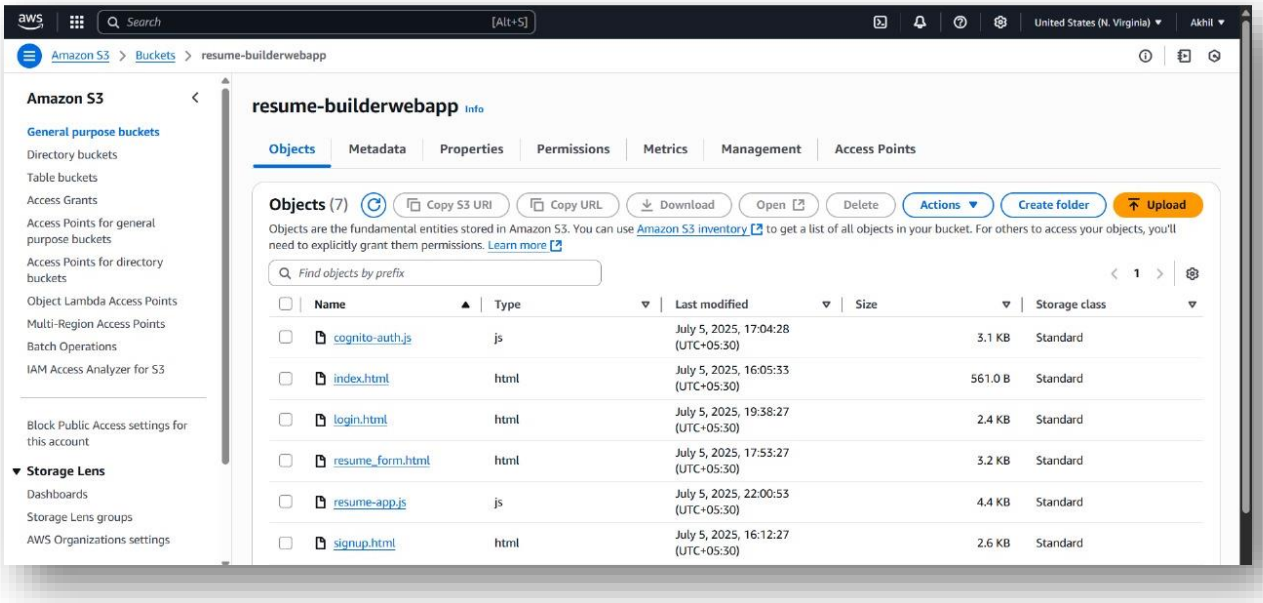
- Role: Packages and sends responses back to the frontend.
- Function: Sends status messages, preview links, download URLs, or error notifications to the client via API Gateway.

1.1. Data Storage Layer

The **Data Storage Layer** is responsible for storing, retrieving, and managing all persistent data within the Resume Builder Web Application. This includes user profile information, resume content (such as skills, education, experience), and the final generated PDF files. The system uses a combination of AWS-managed storage services to ensure high availability, scalability, and security.

Key Objectives of the Data Storage Layer:

- Efficient storage and retrieval of resume data.
- Secure storage of user-generated files (e.g., resume PDFs).
- Scalable handling of concurrent users and requests.
- Seamless integration with application logic through AWS services.



Key Aspects of Data Storage

The Data Storage Layer plays a crucial role in the functionality and reliability of the Resume Builder Web Application. It ensures the secure, structured, and scalable management of all user and system-generated data. Below are the key aspects that define the data storage strategy of the project:

1. Scalability

- The system uses Amazon DynamoDB and Amazon S3, both of which are serverless and automatically scale with usage.
- Able to handle varying amounts of data from multiple users concurrently without performance degradation.

2. Data Organization

- DynamoDB stores structured data (user profiles, education, experience, etc.) in JSON-like documents, making it easy to retrieve and update specific entries.
- Amazon S3 stores binary and media data, such as resume PDFs and optional user-uploaded assets (e.g., profile images).

3. Security and Data Protection

- All data is encrypted at rest and in transit using AWS KMS (Key Management Service).
- Access to data is controlled using fine-grained IAM (Identity and Access Management) policies, ensuring only authorized components can read/write.
- Pre-signed URLs are used to allow secure temporary access to private files stored in S3.

4. Reliability and Availability

- Amazon S3 provides 99.999999999% (11 nines) durability and high availability by default, ensuring resume files are never lost.
- DynamoDB ensures low-latency read/write operations with optional multi-region replication for added reliability**.**

5. **Performance Efficiency**

- DynamoDB's provisioned throughput and auto-scaling ensure consistent performance under varying load.
- Indexed attributes allow fast queries, even for large datasets.
- S3 supports fast retrieval and caching via Amazon CloudFront for publicly available resumes or templates.

6. Data Versioning and Backup

- S3 versioning can be enabled to preserve older versions of resumes.
- Point-in-time recovery (PITR) in DynamoDB allows rollback to any previous state within the last 35 days (if enabled).

7. Integration with Backend Services

- AWS Lambda interacts directly with DynamoDB and S3 to store and fetch data in real time.
- The data layer is tightly integrated into the serverless application architecture, allowing dynamic resume generation without managing infrastructure.

8. Cost Effectiveness

- Pay-as-you-go pricing for both DynamoDB and S3 makes the system cost-efficient, especially for startups or student projects.
- Storage costs are optimized by storing only generated PDFs and relevant metadata—no unnecessary logs or intermediate files.

9. Data Model Flexibility

- NoSQL nature of DynamoDB allows flexibility in storing variable-length data such as projects, certifications, or skills.
- Easy to update or add new fields without schema migration.

10. Support for Real-Time Access

- Resume data is accessible instantly for real-time preview, editing, or download.
- Minimal read/write latency ensures a responsive user experience.

2. Authentication & Authorization Layer

Authentication and authorization are critical components of the Resume Builder Web App, ensuring that users can securely access and manage their personal data and resumes. This layer provides secure access control, user identity management, and protection of user-specific resources. The application leverages AWS-native services to implement robust, scalable, and secure authentication and authorization.

2.1 Purpose of the Layer

- **Authentication** verifies the identity of users (e.g., login with email and password).
- **Authorization** ensures that users can access only their own data (resumes, profiles, etc.) and are prevented from accessing or modifying data belonging to others.

2.3 Technologies Used

Amazon Cognito

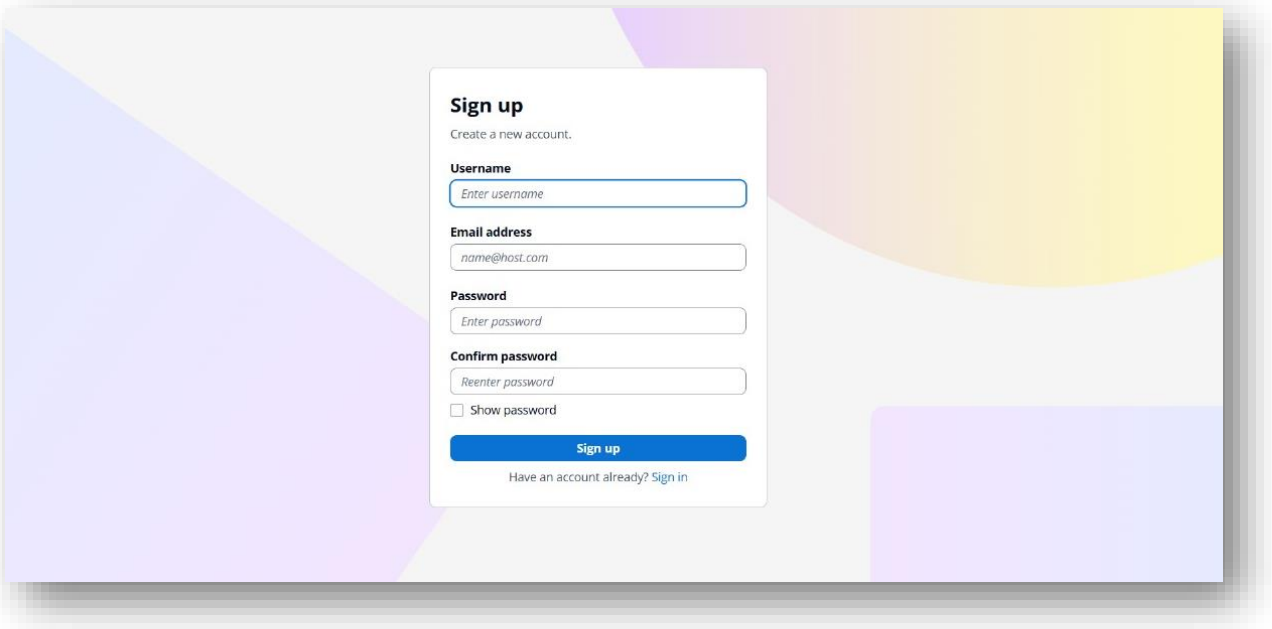
- Functionality:
  - Manages user sign-up, sign-in, and user pool creation.
  - Issues JWT (JSON Web Tokens) to users upon successful authentication.
  - Handles password policies, email verification, and MFA (Multi-Factor Authentication) if enabled.
- Advantages:
  - Fully managed service; scales automatically.
  - Seamlessly integrates with API Gateway and Lambda for protected endpoints.
  - Eliminates the need to build a custom authentication backend.

AWS IAM (Identity and Access Management)

- Functionality:
  - Enforces secure communication between Lambda, DynamoDB, S3, and other services.
  - IAM roles and policies limit Lambda access to only authorized data.

- Example: A Lambda function used to fetch user data from DynamoDB will have a **policy attached that allows it to access only specific tables.**

Authentication & Authorization Layer picture:



3. **Monitoring and Logging**

Monitoring and logging are essential for maintaining the performance, security, and reliability of the Resume Builder Web App. They enable real-time visibility into system operations, help in debugging issues, and ensure that the application meets expected service levels. AWS provides a comprehensive suite of tools to handle monitoring, logging, and alerts, which have been integrated into this project.

3.1 **Objectives of Monitoring and Logging**

- Detect system errors and exceptions in real time.
- Track user activity for auditing and usage analysis.
- Monitor resource utilization and API performance.
- Maintain a record of changes, failures, and user interactions.
- Ensure system uptime and scalability through proactive alerts.

3.2 Monitoring Tools and Services Used

Amazon CloudWatch

- Purpose: Centralized monitoring service for all AWS resources and applications.
- Functionality:
  - Collects logs and metrics from AWS Lambda, API Gateway, and other services.
  - Automatically generates dashboards for visualizing system health.
  - Enables alarms and notifications for anomalies.
- Use Cases in the Project:
  - Monitoring Lambda function execution times and error rates.
  - Tracking API Gateway request/response metrics.
  - Setting alarms for high error counts or slow response times.

Amazon CloudWatch Alarms

- Purpose: Sends alerts when defined thresholds are breached.
- Usage:
  - Alerting when Lambda execution errors exceed a set limit.
  - Notifying admins if API response times spike or resources are throttled. **Amazon CloudTrail**
- Purpose: Logs all API activity across AWS services for security and audit purposes.
- Functionality:
  - Tracks who performed what action and when.
  - Identifies unauthorized access attempts or suspicious behavior.
- Use Case in Project:
  - Auditing access to sensitive services like DynamoDB and S3.
  - Keeping a traceable log of admin and user actions.

3.3 Logging Mechanisms

AWS Lambda Logging with CloudWatch

- Each Lambda function is configured to output logs using console.log() (Node.js) or print() (Python).

- These logs include:
    - Start and end of execution.
    - Input parameters (for debugging).
    - Errors and exceptions encountered.
    - PDF generation status messages.

API Gateway Access Logging

- Logs each request made to the API.
- Captures details such as:
    - Request origin (IP address, headers).
    - HTTP status codes (200, 400, 500).
    - Latency metrics.

Custom Application Logs

- Backend logic may include:
    - User resume creation timestamps.
    - Resume template selection logs.
    - File upload and download success/failure logs.

5.4 Security and Data Compliance

- Logs are encrypted using AWS-managed keys.
- Logs are retained in CloudWatch Logs groups with defined retention periods.
- Access to logs is restricted through IAM policies to prevent unauthorized viewing or editing.

6. CI/CD Layers (Continuous Integration & Deployment)

The Resume Builder Web App uses CI/CD practices to automate the process of building, testing, and deploying code. This ensures faster updates, fewer errors, and consistent deployment across environments.

Benefits

- Automated and fast deployments
- Fewer manual errors
- Easy rollback and version control
- Faster delivery of new features

4. Implementation

The implementation phase of the Resume Builder Web Application involved translating the system design into a functional and deployable product. The project was developed using a modular and serverless approach, leveraging Amazon Web Services (AWS) for scalability, reliability, and cost-efficiency. This section outlines how each component of the system was built and integrated.

4.1 Frontend Implementation

The frontend of the application serves as the user interface, allowing users to enter their resume details and preview the final output.

- **Technologies Used:**
  - HTML5, CSS3, JavaScript (optionally React.js for dynamic rendering)
- **Key Features:**
  - Responsive form for inputting user data such as name, education, experience, and skills.
  - Real-time preview of the resume layout and design.
  - Template selection dropdown to choose preferred resume style.
- **Deployment:**
  - Hosted on **Amazon S3** as a static website.
  - **Amazon CloudFront** used as a Content Delivery Network (CDN) to improve performance and accessibility.

4.2 Backend Implementation

The backend handles data processing, resume generation, and interaction with AWS services.

- **Technologies Used:**
  - **AWS Lambda** for serverless compute (Python or Node.js)
  - **Amazon API Gateway** for routing HTTP requests from frontend to backend
- **Key Functionalities:**
  - Accept user input data via API.
  - Populate HTML/CSS resume templates with user data.
  - Convert the rendered HTML into a downloadable PDF using libraries like pdfkit or Puppeteer.
- **Security:** API endpoints are secured with **Amazon Cognito** to allow access only to authenticated users.

4.3 Database Integration

A NoSQL database structure was implemented to store user-specific data securely.

- **Technology Used: Amazon DynamoDB**
- **Data Stored:**
    - User profiles (name, email, phone)
    - Resume content (education, skills, projects, work experience)
    - Template preferences
- **Benefits:**
    - Schema-less flexibility for variable resume content
    - Fast read/write performance
    - Scales automatically with user load

4.4 File Storage

All generated resume files are stored securely and made available for download.

- **Service Used: Amazon S3**
- **Functionality:**
    - Store the final PDF resume.
    - Provide pre-signed URLs for secure, time-limited download links.
- **Access Control:** Files are private by default and accessible only through user-specific tokens or permissions.

4.5 User Authentication

To protect user data and allow personalization, an authentication system was implemented.

- **Service Used: Amazon Cognito**
- **Features:**
    - Secure sign-up and login process
    - JWT token-based authorization for API access
    - Optional multi-factor authentication (MFA) for enhanced security

4.6 Deployment and CI/CD

The deployment process was fully automated using CI/CD pipelines.

- **Tools and Services:**
    - **GitHub** for source code version control
    - **AWS CodePipeline** and **CodeBuild** for continuous integration and deployment
    - **S3 and Lambda deployment** using build scripts and templates
- **Benefits:**

- Code is automatically built and deployed on every update.
- Faster rollout of features and fixes.
- No manual intervention needed for deployment.

4.7 Monitoring and Logging

To ensure system reliability, monitoring and logging tools were integrated:

- **Amazon CloudWatch** for tracking Lambda and API Gateway metrics
- **CloudTrail** for security and user activity audits
- **Logs:** Include user activity logs, system errors, and API performance data.

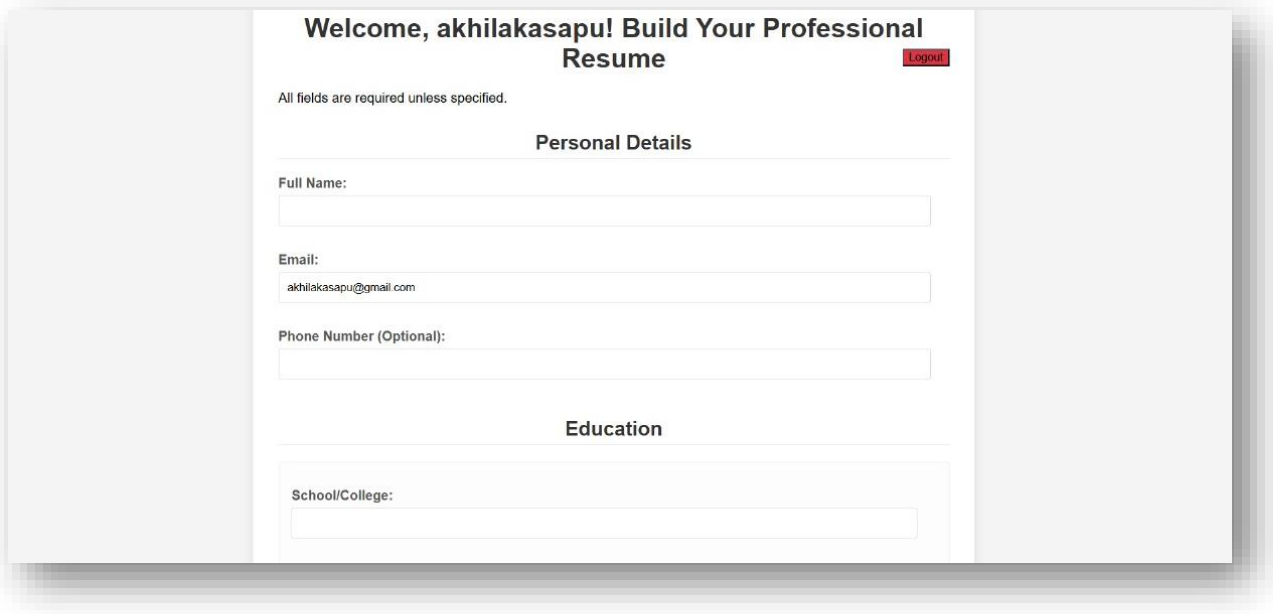
5. Results

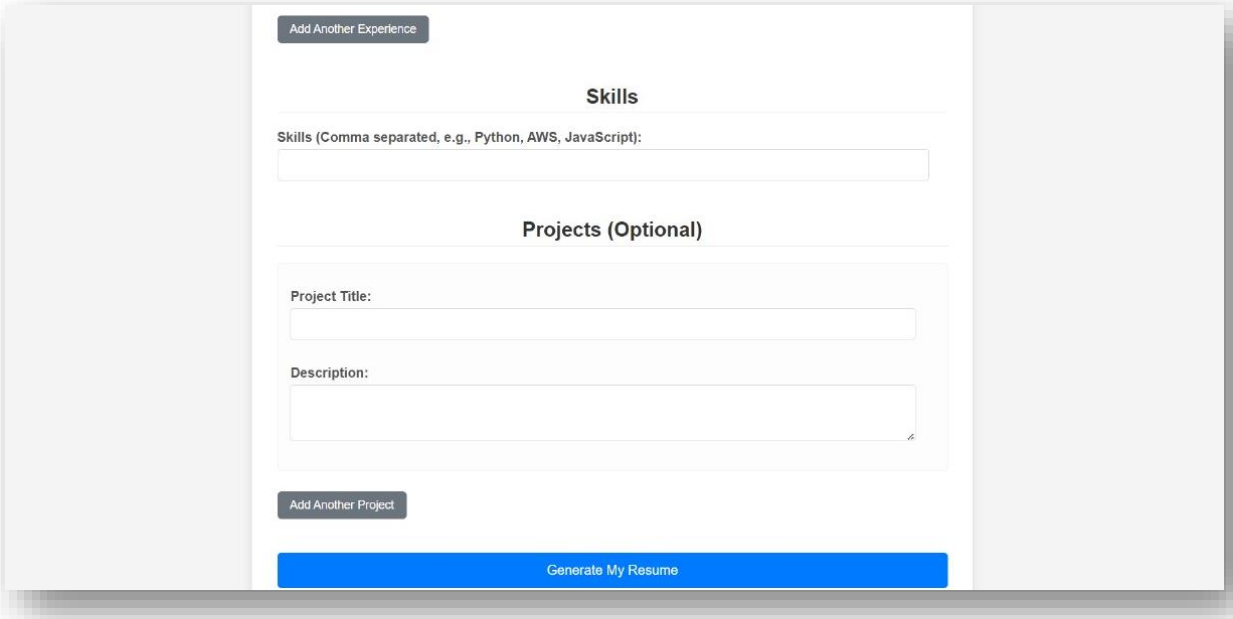
The Resume Builder Web App was successfully developed and deployed using AWS services. The system met all functional and performance goals, providing a secure, responsive, and user-friendly platform for generating professional resumes.

🟩 Key Achievements:

- **Resume Creation:** Users can input data and generate resumes in PDF format.
- **Dynamic Templates:** User data is merged with HTML templates in real-time.
- **PDF Generation:** High-quality resumes are generated using serverless backend logic.
- **Secure Login:** Amazon Cognito enables secure user authentication and data access.
- **Data Storage:** DynamoDB stores resume content; S3 stores downloadable PDFs.
- **Deployment:** The app is fully hosted and deployed using AWS Lambda, S3, and API Gateway.

Outputs:




Add Another Experience
Skills
Skills (Comma separated, e.g., Python, AWS, JavaScript):
Projects (Optional)
Project Title:
Description:
Add Another Project
Generate My Resume


Akhil Akasapu
Email: akhilakasapu@gmail.com | Phone: 8886716698
Summary
N/A
Skills
• java
Education
btech from aditya
Graduation Year: 2027
Experience
tl at amazon

6 Conclusion

The Resume Builder Web Application is a cloud-native project designed to help users create, manage, and download professional resumes easily. Through the use of various Amazon Web Services (AWS) components, the project successfully achieves its objectives of providing a secure, scalable, and user-friendly platform. This application showcases the practical implementation of a serverless architecture that ensures high performance and reliability with minimal operational overhead.

The project was developed using a modular approach. The frontend, built with HTML, CSS, and JavaScript, offers a simple and responsive interface for users to input their resume details. The backend, powered by AWS Lambda and API Gateway, processes this information, integrates it into pre-designed HTML templates, and generates a downloadable PDF using dynamic rendering logic. Resume files are securely stored in Amazon S3, while user data is maintained in Amazon DynamoDB, enabling fast access and retrieval. Authentication is handled by Amazon Cognito, ensuring that users can only access their own data and files.

One of the key highlights of the project is the use of serverless services, which eliminates the need to manage physical servers or infrastructure. AWS automatically handles scaling, availability, and resource optimization, making the system highly efficient and cost-effective. Additionally, CI/CD (Continuous Integration and Deployment) was implemented using tools like GitHub and AWS CodePipeline, enabling seamless and automated code deployment upon each update.

The implementation of real-time monitoring and logging through Amazon CloudWatch and CloudTrail further enhances the application's reliability and maintainability. These tools provide valuable insights into system performance, user activity, and potential errors, allowing for proactive improvements and issue resolution.

Overall, the Resume Builder Web App not only delivers on its core functionalities but also demonstrates the power of cloud computing in building modern web applications. It meets both technical and user-centered goals, offering a professional and secure way for users to build resumes anytime, anywhere. The project provides a solid foundation for future enhancements such as adding more design templates, multilingual support, integration with job portals, or AI-based resume suggestions.

This project serves as a practical example of how cloud services can simplify development while offering robust and scalable solutions for real-world problems.