



**ADBMS**

**Project Proposal:**  
**Smart Resume Builder with AI**  
**Integration**

NAME: FATIMA SHAHEEN

CLASS: CS 4B

SUBMITTED TO : DR BEENISH

## Abstract

The Smart Resume Builder is a dynamic, web-based application developed as part of an Advanced Database Management project. Its primary purpose is to provide users with a convenient and efficient way to create professional resumes with AI-powered skill suggestions.

This project emphasizes the importance of integrating a well-structured backend database system, ensuring secure and versioned data management using advanced database concepts such as triggers, stored procedures, prepared statements, and indexing. The system facilitates the creation of personalized resumes with real-time preview, multiple templates, and PDF export functionality.

## **Table of Contents**

1. Introduction
2. System Overview
3. Features
  - 3.1 Frontend Features
  - 3.2 Database-Oriented Features
4. Technologies Used
5. Database Architecture
  - 5.1 Tables and Relationships
  - 5.2 Stored Procedures
  - 5.3 Triggers
  - 5.4 Prepared Statements
6. Backend Logic
7. Advantages and Disadvantages
8. Applications
9. Conclusion

## 1. Introduction

The Smart Resume Builder is designed to simplify and automate the process of resume creation.

It caters to users who may not be familiar with resume formatting or content structuring by offering a guided, form-based input method.

The system is developed using modern web technologies for the frontend and PHP with MySQL on the backend.

As a part of the Advanced Database course, the project also serves as a practical demonstration of implementing core database management principles in a real-world application.

## 2. System Overview

The system is divided into several interactive components, each handling specific parts of the resume creation process:

- Personal Information: Collects name, contact, job title, location, and summary.
- Experience Section: Users can add multiple job entries with achievements.
- Education Section: Add academic history with institutions, degrees, and dates.
- Skills Section: Manual and AI-suggested skills based on job title.
- Projects Section: Add detailed project history with start/end dates, URLs, and descriptions.
- Live Preview: Shows a real-time updated resume based on user input.
- Template Selector: Allows users to choose from multiple professional layouts.
- PDF Export: Converts the preview into a downloadable PDF document.
- Save: Stores the resume into a database, maintaining version history.

## 3. Features

### 3.1 Frontend Features

The frontend of the **Smart Resume Builder** is designed to be intuitive, responsive, and feature-rich, ensuring that users can easily build, customize, preview, and save their resumes in real time. The system is divided into several interactive components, each handling a specific part of the resume-building process. These components collectively contribute to a seamless and engaging user experience.

## Responsive Tab-Based Interface

The interface is organized into multiple tabs representing different sections of the resume such as:

- **Personal Info**
- **Experience**
- **Education**
- **Skills**
- **Projects**
- **Preview**
- **Templates**

Each tab contains form elements or configuration options relevant to that part of the resume. The design adapts to different screen sizes, ensuring a mobile-friendly and user-centric experience.

## Dynamic Form Handling

Users can add multiple entries to sections like:

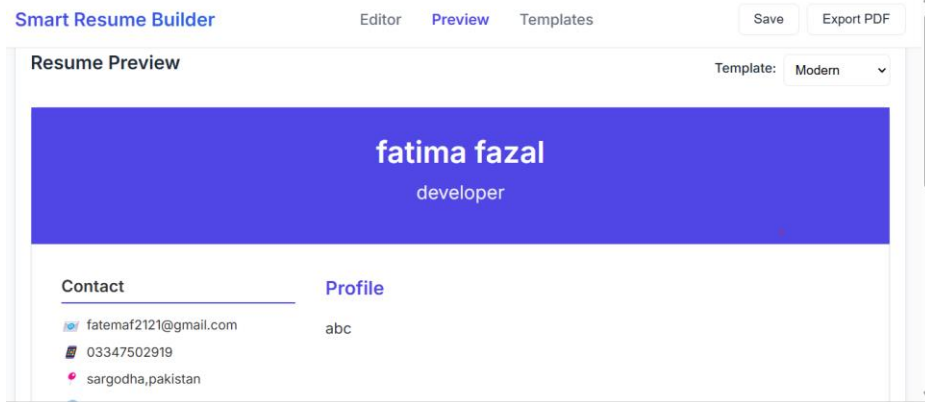
- **Experience**
- **Education**
- **Projects**

Each entry is handled as a card component that can be expanded, collapsed, edited, or deleted. JavaScript dynamically appends these cards without requiring a page reload. This modular structure enhances usability and reflects how real resumes are structured.

## Real-Time Resume Preview

The **Preview** tab shows a live version of the resume that updates instantly as the user types in the editor. This is achieved through JavaScript event listeners and DOM manipulation that mirror the input fields into the preview layout.

It helps users visualize how their resume will appear to recruiters, reducing the need for back-and-forth editing.

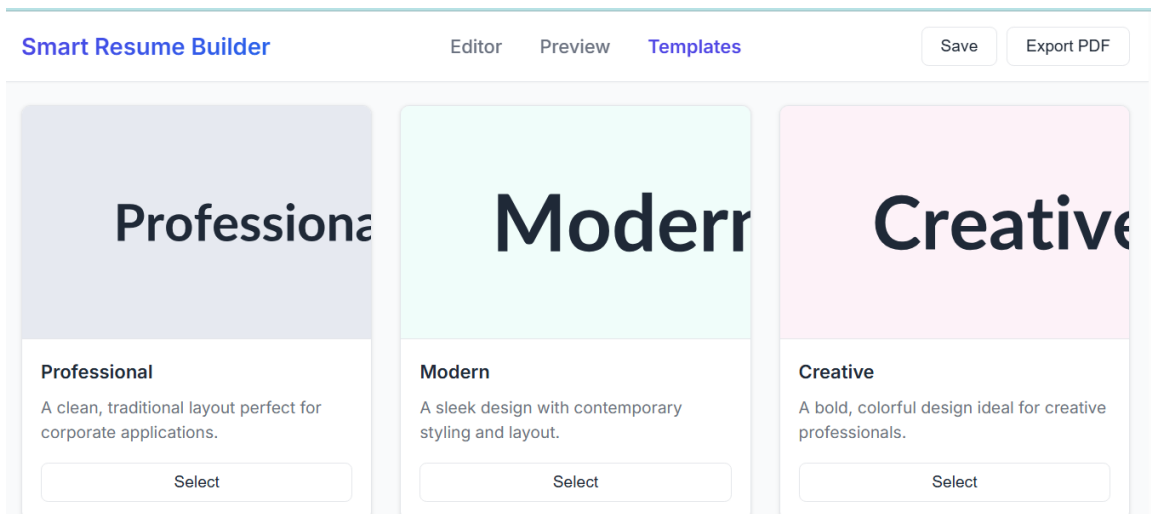


## Template Selection System

The **Templates** tab allows users to choose from four layout styles:

- Simple
- Modern
- Professional
- Creative

Each template modifies fonts, color schemes, and layout spacing. The system applies the selected style instantly to the preview using dynamic class switching, allowing users to experiment with visual appeal before finalizing their resume.

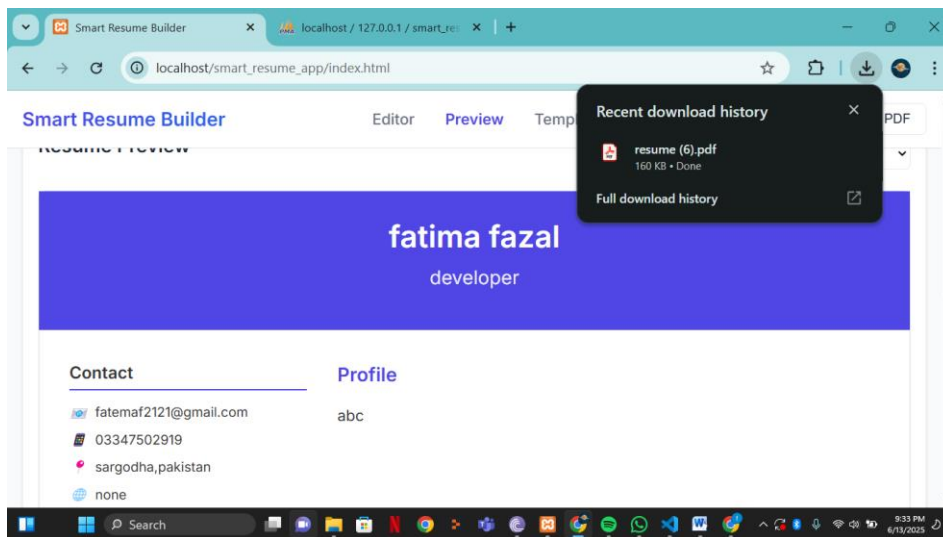


## PDF Export Functionality

The system integrates:

- **html2canvas**: Captures the preview area as an image
- **jsPDF**: Embeds the image into a PDF file

The export function ensures proper scaling and formatting, creating a high-quality PDF ready for download or printing. It even handles multi-page resumes using calculated dimensions and content height detection.

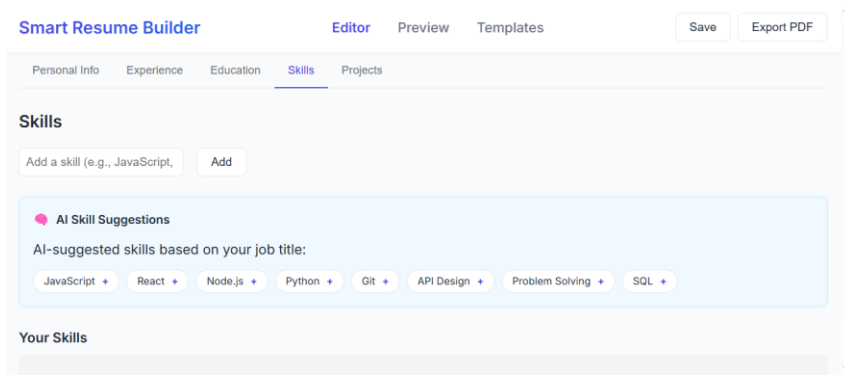


## AI-Supported Skills Entry

In the **Skills** tab, users can either:

- receive **AI-based suggestions** based on the entered **Job Title** in the Personal Info section

This feature intelligently reduces guesswork for users who aren't sure which skills to list, especially students or entry-level applicants.



## Save Functionality with Versioning

Once the resume is complete, users can click “**Save**”, which sends the data to the backend using JavaScript’s `fetch()` method. The PHP backend (`save_resume.php`) processes the request by:

- Saving all resume sections into MySQL tables
- Calling a stored procedure to automatically create a **new version number**
- Ensuring old versions are retained for future access

This feature acts like **version control for resumes**, allowing users to maintain historical edits without data loss.

## Interactive Components

- **Personal Information** – Collects contact details, name, job title, etc.
- **Experience** – Add or edit job roles and achievements.
- **Education** – Enter academic history with degree and duration.
- **Skills** – Input manual or AI-suggested skills with proficiency.
- **Projects** – Record academic/technical project details.
- **Preview** – Shows real-time resume view as user types.
- **Templates** – Choose from 4 professional design layouts.
- **PDF Export** – Export the resume into a printable PDF file.
- **Save to Database** – Stores resume with automatic version control.



## Frontend Interface:

**Smart Resume Builder**

EditorPreviewTemplates

SaveExport PDF

Personal InfoExperienceEducationSkillsProjects

### Personal Information

First Name

fatima

Last Name

fazal

Email

fatemaf2121@gmail.com

Phone

03347502919

Job Title

developer

### 3.2 Database-Oriented Features

- User data persistence using normalized relational tables.
- Resume versioning using a dedicated table with automatic version incrementing via trigger.
- Stored procedures for structured insert logic (e.g., `InsertResumeVersion``).
- Prepared statements used in PHP to prevent SQL injection.
- Indexes on primary and frequently queried keys for performance (e.g., email, resume\_id).

## 4. Technologies Used

- HTML5 & CSS3 for responsive structure and styling.
- JavaScript for interactivity, form handling, and client-side logic.
- PHP for server-side scripting and database communication.
- MySQL for structured, relational data storage.
- XAMPP as a local web server environment (includes Apache and MySQL).
- phpMyAdmin for database management.
- jsPDF & html2canvas for generating PDFs from HTML content.

## 5. Database Architecture

This section explains the structure and advanced features of the backend MySQL database. It includes schema organization, table relationships, use of triggers, stored procedures, indexing strategies, and data integrity constraints.

The system ensures that each resume is saved with historical tracking, allowing users to manage and revert to previous versions. All resume-related data is linked via `resume_id` and stored securely using optimized queries.

### 5.1 Tables and Relationships

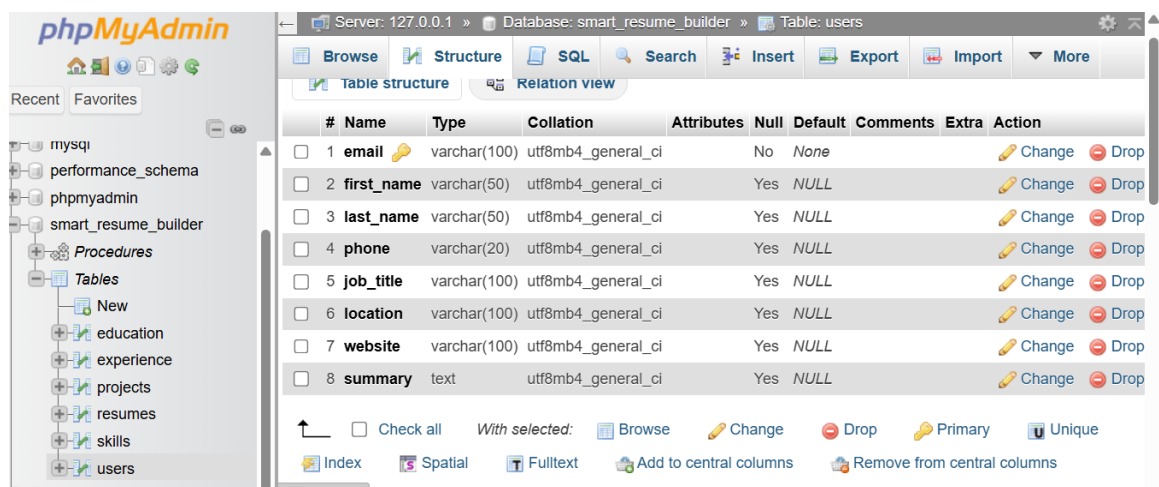
The database follows a normalized relational structure consisting of:

- **users:** Contains user details (email, name, phone, etc.). Email acts as the primary key.
- **resumes:** Stores resume metadata such as version number, timestamp, and associated user email.
- **experience, education, skills, projects:** Hold the content of each resume section, all linked to the `resumes` table via `resume_id`.

This separation ensures **modularity**, **data integrity**, and avoids redundancy.

### Database Interface :

#### USERS:



## RESUMES:

The screenshot shows the phpMyAdmin interface. On the left, the database structure tree is visible, with 'smart\_resume\_builder' selected. Under 'Tables', the 'resumes' table is highlighted. The main panel displays the 'Table structure' for 'resumes'. The table has four columns: 'id' (int(11), primary key, auto-increment), 'email' (varchar(100), unique), 'version' (int(11)), and 'created\_at' (timestamp). The interface includes various tools like 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', and 'Import'. At the bottom, there is a console window showing the URL: 'localhost/phpmyadmin/index.php?route=/table/structure...'.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(11)			No	None		AUTO_INCREMENT
2	email	varchar(100)	utf8mb4_general_ci		Yes	NULL		Unique
3	version	int(11)			Yes	NULL		
4	created_at	timestamp			No	current_timestamp()		

## OTHER TABLES:

The first screenshot shows the 'education' table structure. It has nine columns: 'id' (int(11), primary key, auto-increment), 'resume\_id' (int(11), foreign key), 'school' (varchar(100)), 'degree' (varchar(100)), 'field\_of\_study' (varchar(100)), 'start\_date' (date), 'end\_date' (date), 'location' (varchar(100)), and 'description' (text). The second screenshot shows the 'experience' table structure. It has eight columns: 'id' (int(11), primary key, auto-increment), 'resume\_id' (int(11), foreign key), 'title' (varchar(100)), 'company' (varchar(100)), 'start\_date' (date), 'end\_date' (date), 'location' (varchar(100)), and 'achievements' (text).

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(11)			No	None		AUTO_INCREMENT
2	resume_id	int(11)			Yes	NULL		Foreign key to resumes.id
3	school	varchar(100)	utf8mb4_general_ci		Yes	NULL		
4	degree	varchar(100)	utf8mb4_general_ci		Yes	NULL		
5	field_of_study	varchar(100)	utf8mb4_general_ci		Yes	NULL		
6	start_date	date			Yes	NULL		
7	end_date	date			Yes	NULL		
8	location	varchar(100)	utf8mb4_general_ci		Yes	NULL		
9	description	text	utf8mb4_general_ci		Yes	NULL		

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(11)			No	None		AUTO_INCREMENT
2	resume_id	int(11)			Yes	NULL		Foreign key to resumes.id
3	title	varchar(100)	utf8mb4_general_ci		Yes	NULL		
4	company	varchar(100)	utf8mb4_general_ci		Yes	NULL		
5	start_date	date			Yes	NULL		
6	end_date	date			Yes	NULL		
7	location	varchar(100)	utf8mb4_general_ci		Yes	NULL		
8	achievements	text	utf8mb4_general_ci		Yes	NULL		





Table structure		Relation view						
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(11)		No	None			AUTO_INCREMENT
2	resume_id	int(11)		Yes	NULL			
3	name	varchar(100)	utf8mb4_general_ci	Yes	NULL			
4	start_date	date		Yes	NULL			
5	end_date	date		Yes	NULL			
6	url	varchar(200)	utf8mb4_general_ci	Yes	NULL			
7	description	text	utf8mb4_general_ci	Yes	NULL			

## 5.2 Stored Procedures

Stored procedures encapsulate multi-step operations. This project uses:

- **InsertResumeVersion(email):**  
Called whenever a user saves their resume. It:
  - Retrieves the latest version number
  - Increments it
  - Creates a new entry in the `resumes` table
  - Returns the new `resume_id` for linking experience, education, skills, and projects

Stored procedures help ensure **atomic operations** and reduce redundancy in PHP code.

Name	Type	Returns
<input type="checkbox"/> InsertResumeVersion	PROCEDURE	 Edit  Execute  Export  Drop

## 5.3 Triggers

A **MySQL Trigger** is used for **versioning**:

- **AFTER UPDATE ON resumes:**  
Before any resume data is changed, the current state is automatically backed up. This ensures **version history is preserved** without needing frontend logic.

This allows users to always retain previous versions of their resume and improves **auditability**.

```

70 -- TRIGGER FOR VERSIONING
71 DELIMITER $$
72 CREATE TRIGGER before_resume_insert
73 BEFORE INSERT ON resumes
74 FOR EACH ROW
75 BEGIN
76     DECLARE max_version INT DEFAULT 0;
77     SELECT IFNULL(MAX(version), 0) INTO max_version FROM resumes WHERE email =
NEW.email;
78     SET NEW.version = max_version + 1;
79 END $$
80 DELIMITER ;

```

## 5.4 Resume Versioning Logic

Resume versioning is a **core backend feature**. Every time a user clicks save:

1. The `save_resume.php` script calls the stored procedure `InsertResumeVersion(email)`
2. A new version number is created and inserted into the `resumes` table
3. The returned `resume_id` is used to insert related experience, education, skills, and projects
4. A database trigger ensures a backup of the previous state before any update

This ensures:

- No data is overwritten or lost
- Users can track and revert changes
- Scalability and history retention

## 5.5 Prepared Statements

To secure data operations in `save_resume.php`, all queries use **prepared statements**, for example:

```

// Insert or update user info
$stmt = $conn->prepare("INSERT INTO users (email, first_name, last_name, phone, job_title, location, website, summary)
VALUES (?, ?, ?, ?, ?, ?, ?, ?)
ON DUPLICATE KEY UPDATE
    first_name = VALUES(first_name),
    last_name = VALUES(last_name),
    phone = VALUES(phone),
    job_title = VALUES(job_title),
    location = VALUES(location),
    website = VALUES(website),
    summary = VALUES(summary)");
$stmt->bind_param("sssssss", $email, $p['firstName'], $p['lastName'], $p['phone'],
    $p['jobTitle'], $p['location'], $p['website'], $p['summary']);
$stmt->execute();

```

This provides:

- **SQL injection protection**
- **Type-safe input binding**
- **Better performance for repeated queries**

## 6. Backend Logic

The file `save\_resume.php` receives JSON-formatted data sent via JavaScript fetch requests. It decodes this data, validates its presence, and inserts records into corresponding tables using secure prepared statements.

It also calls a stored procedure to track resume versioning and uses a helper function to parse date formats.

## 7. Advantages and Disadvantages

Advantages:

- Enables easy and fast resume creation.
- Real-time preview and export without third-party tools.
- Clean separation of concerns between UI and database logic.
- Secure and efficient backend with advanced database features.

Disadvantages:

- Requires XAMPP/local server environment.
- Currently supports only local data storage (no user authentication).
- PDF layout depends slightly on browser rendering.

## 8. Applications

- Useful for students to create their first professional resumes.
- Recruiters can assist candidates in quick resume generation.
- Freelancers and job seekers benefit from resume version tracking.

## 9. Conclusion

This project demonstrates an effective blend of frontend and backend technologies. By focusing on real-time user interaction and advanced database techniques, the Smart Resume Builder becomes a powerful yet simple tool that reflects professional software development and database design principles.