TRIBHUVAN UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

Online Birth Registration System

In partial fulfillment of the requirements for the Bachelor's Degree in Computer Science and Information Technology

Under the Supervision of

Mr. Yubaraj Dahal

Lecturer

Department of Computer Science and Information Technology

Amrit Campus

Thamel, Kathmandu

Submitted by

Ningsang Jabegu (TU Roll No:28198/078)

Amrit Campus

Department of Computer Science and Information Technology

Thamel, Kathmandu

Submitted to

Tribhuvan University

Institute of Science and Technology

Jan 2025

Abstract

The transition from paper-based to digital workflows is a critical objective for modern governance. This project, the "Online Birth Registration System," addresses this need by digitizing and streamlining the processes of birth registration, verification, and certificate generation for a local government office. The system replaces manual ledgers with a secure web application, providing distinct interfaces for citizens, administrators, and verifiers. Citizens can submit new birth registrations and download official A4 certificates. Administrators are equipped with a secure dashboard to review, approve, reject, and manage records. A public-facing verification module allows agencies to confirm certificate authenticity using a registration number or a robust fuzzy-matching algorithm based on name and date of birth. The system is architected with a React and TypeScript frontend and a Node.js (Express) backend. For portability and simplicity, the backend utilizes a CSV file-based storage system with file-locking and backup mechanisms. Key features include a REST API for all CRUD operations, a server-side Levenshtein-distance-based fuzzy search, and client-side PDF generation using html2canvas and jsPDF to produce official A4 certificates from an HTML template. This report details the system's design, implementation, algorithms, and evaluation, demonstrating its efficacy as a lightweight, deployable solution for local government civil registration.

Keywords: *e-Governance, Civil Registration, Vital Statistics (CRVS), Birth Registration, Fuzzy Search, Levenshtein Distance, PDF Generation, React, Node.js, System Design*

Table of Contents

Chapter 1: Introduction

1.1 Background

Civil Registration and Vital Statistics (CRVS) systems are the foundation for establishing legal identity. The registration of births, as the first step in this process, ensures that an individual is recognized by the state, providing access to fundamental rights such as healthcare, education, and social protection (UNICEF, 2021). In many local government offices, particularly in developing regions, this critical process remains manual. Registrars rely on physical ledgers, paper forms, and hand-issued certificates, a practice that is time-consuming, prone to human error, susceptible to data loss from physical damage, and vulnerable to fraud.

The digital transformation of these services, often termed e-governance, offers a powerful solution. A digital system can streamline data entry, enforce validation, provide secure and centralized storage, and automate the issuance of verifiable documents. This project, the "Online Birth Registration System," is situated within this context, aiming to provide a practical, modern, and deployable web application to replace these outdated paper-based workflows.

1.2 Problem Statement

The manual, paper-based birth registration process at local government offices suffers from several significant drawbacks:

1. **Inefficiency**: The process of manually recording, searching, and transcribing data is slow, leading to long wait times for citizens.
2. **Data Insecurity**: Physical ledgers are vulnerable to fire, water damage, or misplacement, resulting in permanent data loss.

3. **Inaccessibility:** Retrieving a record or generating a new certificate copy is a laborious task that requires a physical presence at the office.
4. **Verification Difficulty:** There is no efficient or reliable way for third parties (e.g., schools, employers) to verify the authenticity of a paper certificate, creating opportunities for fraud.
5. **Data Inconsistency:** Manual entry without validation leads to errors, such as misspellings or incorrect dates, which complicate future record-keeping.

This project aims to develop a web-based application that directly addresses these issues by creating a digital, secure, and verifiable system for birth registration.

1.3 Project Aims and Objectives

The primary aim of this project is to design, develop, and implement a secure and user-friendly Online Birth Registration System to digitize and streamline the entire registration lifecycle.

To achieve this aim, the following objectives were set:

- **To replace paper workflows** with a digital interface for citizens to submit birth registration forms.
- **To provide secure administrative tools** for government staff to review, approve, reject, and manage registration records.
- **To develop an automated generator** for an official, A4-printable birth certificate in PDF format upon approval.
- **To implement a public verification module** to confirm certificate authenticity, utilizing both direct search and a fuzzy-matching algorithm to handle typos.

- **To ensure data portability and simplicity** by using a robust CSV-based storage backend with appropriate safeguards.

1.4 Scope and Limitation

1.4.1 Scope

The system is designed as a standalone application for a single local government office. Its scope includes:

- Citizen-facing registration form.
- Admin dashboard for CRUD operations on records.
- Status-based workflow (Pending, Approved, Rejected).
- Client-side PDF generation of a standardized A4 certificate.
- Server-side fuzzy search verification based on name and date of birth.
- Basic user authentication (admin vs. citizen) backed by a CSV file.

1.4.2 Limitations

This project is an academic implementation and has several intentional limitations:

- **Storage:** The use of CSV files is for simplicity and portability. It is not intended for high-concurrency, large-scale (e.g., national) deployments, where a relational database (RDBMS) would be superior.
- **Authentication:** The current authentication is basic. It does not include advanced features like JSON Web Tokens (JWT), session rotation, or multi-factor authentication (MFA).
- **PDF Generation:** The PDF is generated via client-side rasterization (html2canvas), which may not be as crisp as a vector-based, server-side generator.

- **Search:** The fuzzy search scans the entire CSV file (O(n)), which is suitable for thousands of records but would be inefficient for millions.

1.5 Report Structure

This report is organized into ten chapters. **Chapter 1** provides the introduction, problem, objectives, and scope. **Chapter 2** reviews literature on existing CRVS systems, web technologies, and matching algorithms. **Chapter 3** details the system requirements, use cases, and architectural design, including component and sequence diagrams. **Chapter 4** describes the backend and frontend implementation, highlighting key technology choices and code structures. **Chapter 5** provides an in-depth analysis of the fuzzy-matching algorithm (Levenshtein distance) used for verification. **Chapter 6** concludes the report.

Chapter 2: Literature Review

2.1 Overview of Civil Registration and Vital Statistics (CRVS)

Globally, CRVS systems are recognized as a cornerstone of public administration. The United Nations (2014) emphasizes that "universal registration of births... is central to the realization of human rights" (p. 1). A complete CRVS system provides individuals with legal identity and enables governments to produce accurate vital statistics for policy and planning (World Health Organization, 2017). The shift from paper to digital CRVS, or "e-CRVS," is a global trend. Digital systems promise improved data quality, timeliness, and accessibility (Setel et al., 2007). However, successful implementation depends on appropriate technology, stakeholder engagement, and a robust legal framework. This project aligns with this global trend by focusing on the "registration" and "certification" components of a CRVS system at a local level.

2.2 Existing Digital Registration Systems

A review of existing systems reveals a spectrum of solutions. At the high end are comprehensive national systems, such as India's Civil Registration System (crsorgi.gov.in) or the systems in developed nations, which are often deeply integrated with national ID databases, health systems, and statistical offices. These systems are powerful but immensely complex and expensive to build and maintain.

At the other end, various open-source platforms like OpenCRVS have emerged, offering modular, standards-based solutions. While highly capable, they may still be heavyweight for a single, small local office with limited technical resources. This project's chosen architecture (lightweight server, CSV store) is a pragmatic compromise, inspired by the need for a system that is easy to deploy and maintain in a resource-constrained environment, while still providing core digital functionality.

2.3 Technologies for Web Application Development

The project's technology stack aligns with modern web development practices.

- **Backend (Node.js & Express):** Node.js is widely adopted for building scalable and efficient network applications. As described by Teixeira (2020), its event-driven, non-blocking I/O model makes it suitable for data-intensive real-time applications, including REST APIs. Express.js, as a minimal and flexible Node.js framework, provides a robust set of features for web and mobile applications without obscuring the core Node.js features.
- **Frontend (React & TypeScript):** React has become a dominant library for building user interfaces due to its component-based architecture and declarative views (Banker, 2018). The use of TypeScript adds static typing to JavaScript, which, as noted by Bähre (2021), significantly improves code quality, maintainability, and developer productivity in complex applications by catching errors at compile time.

  local level.

2.4 Data-Matching and Verification Techniques

Verifying identity from user-provided data is a classic problem in computer science. Exact string matching fails in the presence of typos, transliteration differences, or name variations (Christen, 2012). This necessitates the use of "fuzzy" or "approximate" string matching.

The **Levenshtein distance** algorithm is a canonical method for this (Levenshtein, 1966). It measures the "distance" between two strings as the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one string into the other. This algorithm is highly relevant for name matching, where errors are often single-character mistakes. Other algorithms like Jaro-Winkler or n-gram similarity exist, but Levenshtein provides a robust and intuitive baseline for this project's scope.

2.5 Secure Certificate Generation

The output of a registration system is the certificate. Digitally, this is often a PDF. Research in secure document generation highlights two main approaches: client-side and server-side. Server-side generation (e.g., using Puppeteer or wkhtmltopdf) can produce high-fidelity, vector-based PDFs but adds server load and complexity.

This project uses a client-side approach (html2canvas, jsPDF). While this converts HTML to a rasterized image within the PDF, it is simple to implement and offloads the generation work to the client's browser. For verification, the use of a unique CERTIFICATE_NO and the fuzzy search fallback provides authenticity, but a future-facing system would also incorporate digital signatures or QR codes linking to a verification URL, as recommended by e-governance security frameworks (OWASP, 2023).

2.5 Research Gap

The literature shows a gap between large-scale, complex national CRVS systems and the needs of small local government offices. There is a need for lightweight, easily deployable, and maintainable systems that still provide core, robust features like digital registration and (critically) reliable verification. This project addresses this gap by creating a system that prioritizes simplicity (CSV storage, simple auth) and portability, while integrating a non-trivial algorithmic solution (fuzzy matching) to solve a key real-world problem (verification with typos).

Chapter 3: Requirements and System Design

3.1 Requirements Elicitation

The requirements for the Online Birth Registration System were gathered by analyzing the existing manual workflow and identifying its key stakeholders. The primary stakeholders are:

- **Citizens/Parents:** Need a simple way to register a birth and receive a certificate.
- **Local Registrar (Admin):** Needs to manage, review, and approve registrations efficiently.
- **External Verifiers:** Need a reliable way to confirm a certificate's authenticity.

The goals of each stakeholder were translated into the functional and non-functional requirements detailed below.

3.2 Functional Requirement

The system must perform the following functions:

- **FR1 (Citizen):** A user shall be able to submit a new birth registration by filling out a web form.
- **FR2 (Citizen):** A user shall be able to download an official A4 birth certificate in PDF format *only* after their registration has been approved.
- **FR3 (Citizen/Verifier):** A user shall be able to access a public verification page.
- **FR4 (Citizen/Verifier):** The system shall allow verification by an exact Certificate Number.
- **FR5 (Citizen/Verifier):** The system shall allow verification by a fuzzy search on Last Name and Date of Birth.
- **FR6 (Admin):** An admin user shall be required to log in to access the administrative dashboard.

- **FR7 (Admin):** An admin shall be able to list, search, and filter all registration records.
- **FR8 (Admin):** An admin shall be able to view the full details of a single registration.
- **FR9 (Admin):** An admin shall be able to approve a pending registration.
- **FR10 (Admin):** An admin shall be able to reject a pending registration and must provide a REJECT_REASON.
- **FR11 (System):** The system shall persist all registration data in a CSV file.
- **FR12 (System):** The system shall provide REST API endpoints for all CRUD and search functionalities.

3.3 Non-Functional Requirement

The system must adhere to the following quality attributes:

- **NFR1 (Security):** Admin passwords must be hashed (using bcrypt) before being stored. Admin-only routes must be protected from public access.
- **NFR2 (Portability):** The data store (CSV) must be human-readable and easily transferable to other systems or for manual backup.
- **NFR3 (Usability):** The citizen-facing forms and admin dashboard must be intuitive, responsive, and easy to navigate.
- **NFR4 (Robustness):** The system must safely handle concurrent writes to the CSV file (via file locking) and create backups during writes.
- **NFR5 (Accuracy):** The fuzzy search algorithm must have a high recall rate for common typographical errors.
- **NFR6 (Compliance):** The generated A4 certificate must visually match the official government-specified layout, including watermarks and coat of arms.

3.3 Use Case Analysis

The use cases describe the interactions between the actors (Citizen, Admin, Verifier) and the system.

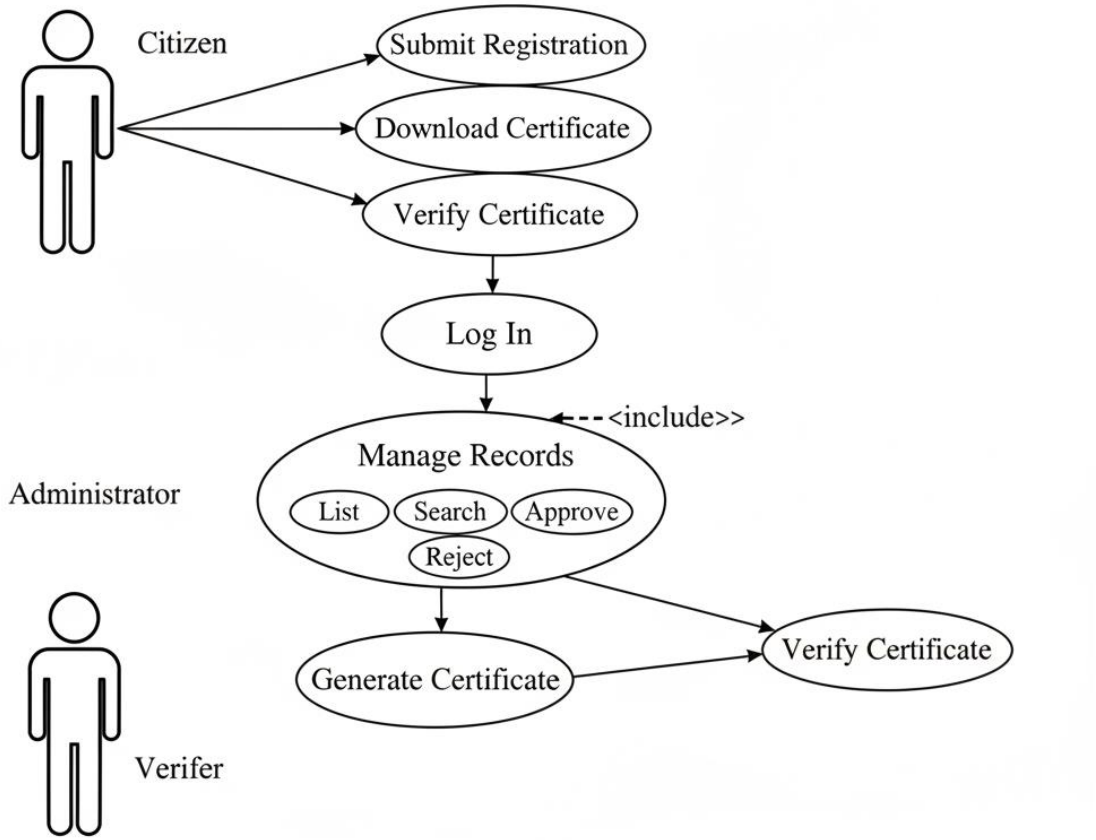

Figure 3.1: Use Case Diagram for the Online Birth Registration System

3.5 System Architecture

The system follows a modern client-server architecture, with a clear separation of concerns between the frontend (presentation layer) and the backend (logic and data layer).

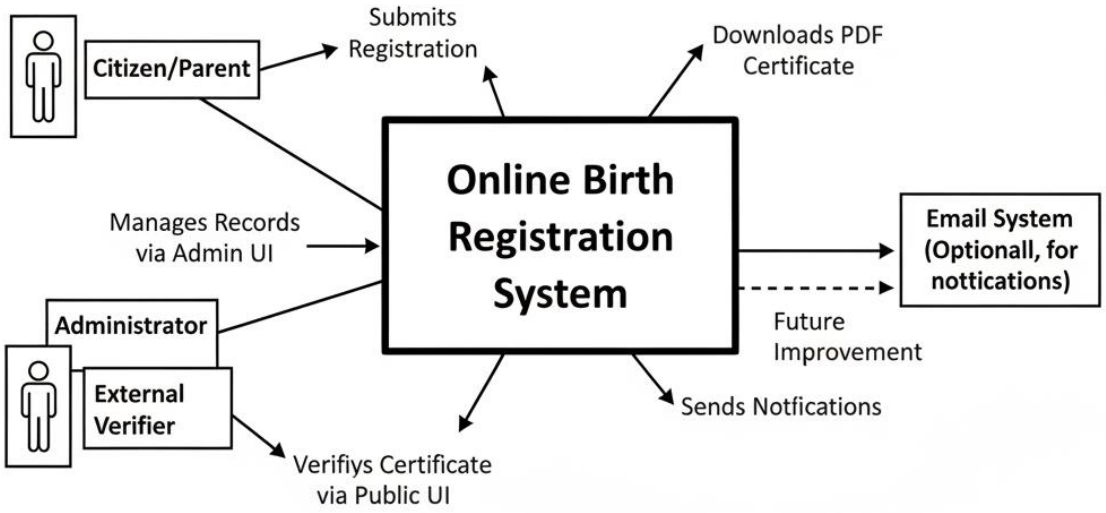

Figure 3.2: Use Case Diagram for the Online Birth Registration System

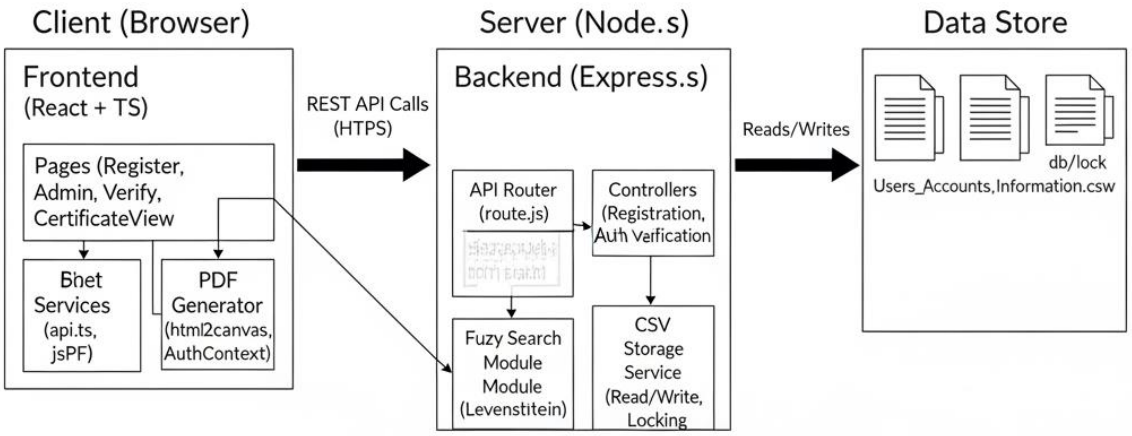

Figure 3.3: High-Level Component Diagram

3.6 Sequence Diagram

Sequence diagrams illustrate the flow of interactions for key use cases.

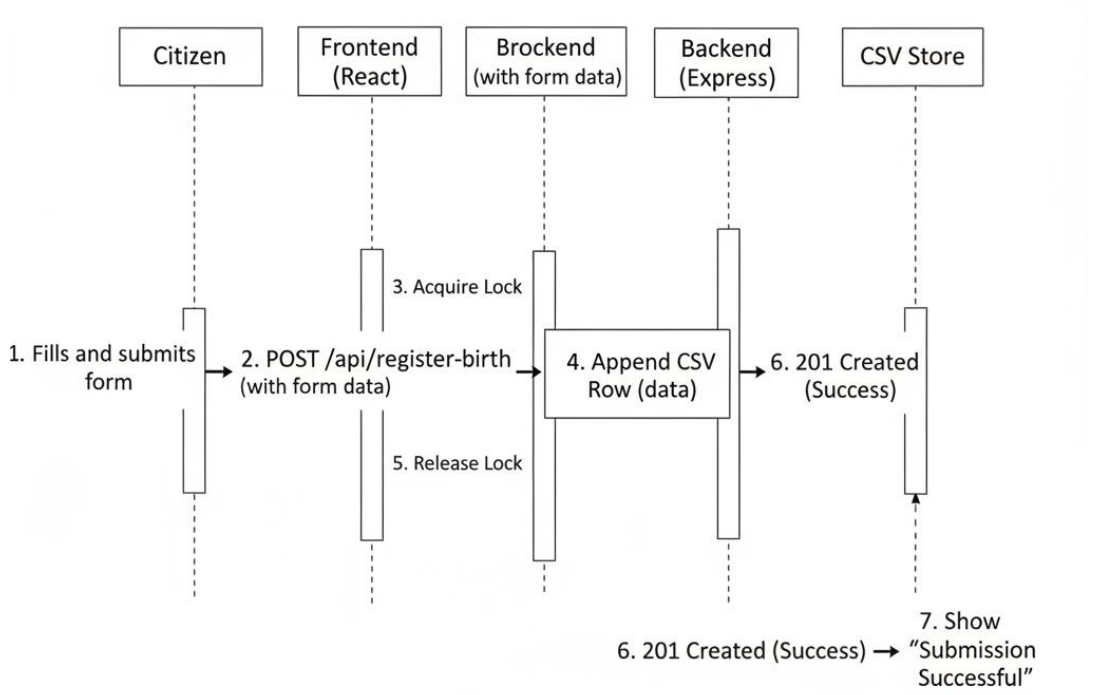

Figure 3.4: Sequence Diagram for New Birth Registration

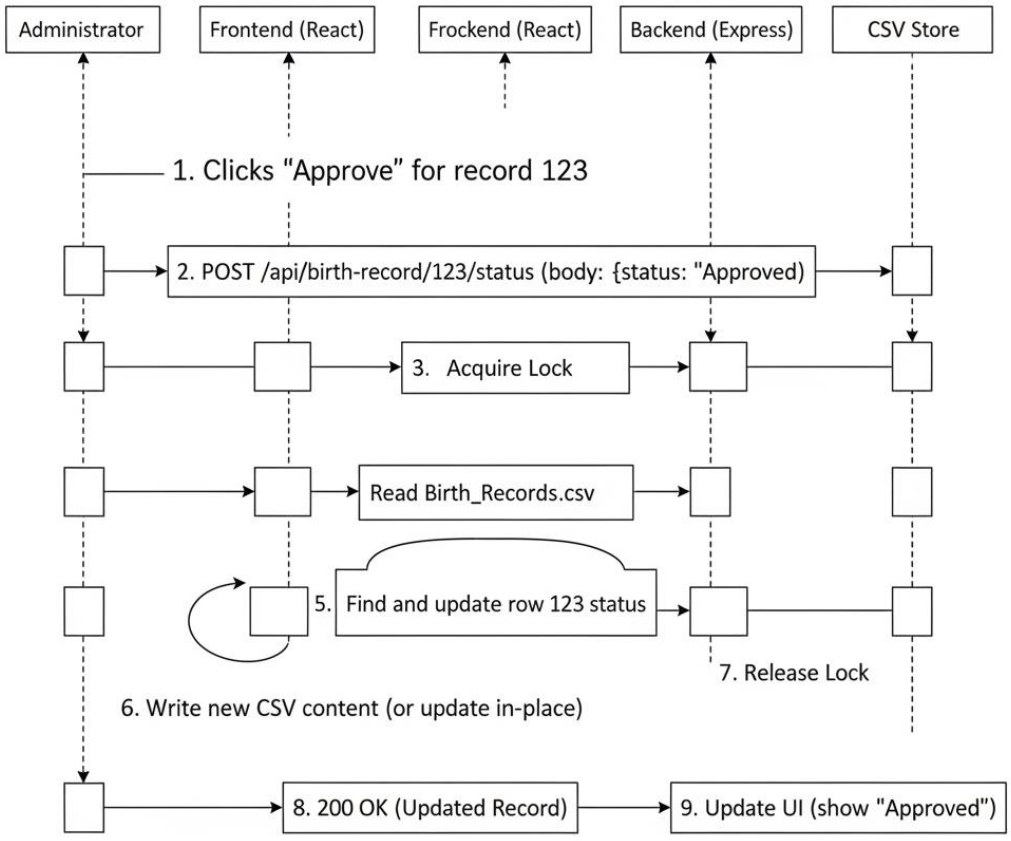

Figure 3.5: Sequence Diagram for Admin Approval Flow

Figure 3.6: Sequence Diagram for Verification (Fuzzy Search Fallback)

3.7 Data Design

The system's data is managed in two primary CSV files.

Table 3.1: Data Model for Birth_Records.csv

Header	Type	Description
ID	String	Unique identifier (e.g., UUID or timestamp)
CERTIFICATE_NO	String	Official, human-readable certificate number
CHILD_FIRST_NAME	String	Child's given name
CHILD_LAST_NAME	String	Child's surname
GENDER	String	e.g., "Male", "Female", "Other"
DATE_OF_BIRTH	String	Date of birth in AD (ISO 8601: YYYY-MM-DD)
NEPALI_DOB	String	Date of birth in BS (Vikram Samvat)
PLACE_OF_BIRTH	String	e.g., Hospital name or address
PROVINCE	String	Administrative province
FATHER_FIRST_NAME	String	Father's given name
FATHER_LAST_NAME	String	Father's surname
MOTHER_FIRST_NAME	String	Mother's given name
MOTHER_LAST_NAME	String	Mother's surname
PERMANENT_ADDRESS	String	Full permanent address
CONTACT_NUMBER	String	Phone number of informant
REGISTERED_BY	String	ID of the informant/citizen account
REGISTERED_AT	Timestamp	Date and time of submission
STATUS	String	"Pending", "Approved", "Rejected"

REJECT_REASON	String	Text field for admin, if STATUS is "Rejected"

Table 3.2: Normalized API Record Shape (JSON)

Key	Type	Description
id	String	Maps from ID
certificateNo	String	Maps from CERTIFICATE_NO
childFullName	String	Computed: (FIRST, MIDDLE, LAST)
dobAD	String	Maps from DATE_OF_BIRTH
dobBS	String	Maps from NEPALI_DOB
fatherName	String	Computed: (FIRST, MIDDLE, LAST)
motherName	String	Computed: (FIRST, MIDDLE, LAST)
status	String	Maps from STATUS
rejectReason	String	Maps from REJECT_REASON

Chapter 4: Implementation

The system was implemented using the MERN stack, with modifications—substituting MongoDB with a CSV file system for the data layer. The backend provides a RESTful API, and the frontend is a responsive single-page application (SPA).

4.1 Backend Implementation

The backend is built with Node.js and the Express.js framework, prioritizing simplicity and robustness for the CSV storage mechanism.

4.1.1 Technology Stack

- **Runtime:** Node.js (v18.x)
- **Framework:** Express.js (v4.x)
- **Key Libraries:** csv-parser, csv-writer (for data handling), bcrypt (for password hashing), fast-levenshtein (for fuzzy search).

4.1.2 API Endpoint Design

A RESTful API structure was implemented to handle all data operations. This decouples the backend logic from the frontend presentation.

Table 4.1: Key API Endpoints

Method	Endpoint	Description
POST	/api/login	Authenticates an admin user.
POST	/api/register-birth	Submits a new birth registration.
GET	/api/birth-records	Lists all records (for admin).
GET	/api/birth-record/:id	Fetches a single record by its ID or CERTIFICATE_NO.

POST	/api/birth-record/:id/status	Updates a record's status (Approved/Rejected).
GET	/api/birth-record/search	Performs fuzzy search. (Query: ?name=...&dob=...)

4.1.3 Data Storage and persistence

The core of the backend's data layer is its interaction with CSV files. To mitigate the risks of file-based storage, a "safe write" mechanism was implemented.

- **File Locking:** Before any write operation (new registration, status update), the system creates a .lock file. If this file already exists, the incoming request is queued or rejected to prevent a race condition. This ensures that only one process can write to the CSV at a time, maintaining data integrity.
- **Backups:** Before overwriting the main Birth_Records.csv (e.g., during a status update), a backup copy is created (e.g., Birth_Records.csv.bak).
- **CSV Quoting:** All data written to the CSV files uses robust quoting to handle fields that may contain commas, newlines, or quotation marks (e.g., an address or a REJECT_REASON).

4.1.4 User Authentication

Authentication is handled at the API level. Admin users are stored in Users_Accounts_Information.csv.

1. When an admin attempts to log in via POST /api/login, the system reads the user CSV.
2. It finds the user by email and compares the provided password with the stored hash using bcrypt.compare().

3. If successful, a token (or in this simple implementation, a session/cookie) is returned, which the frontend must include in headers for protected routes.
4. A server-side middleware checks for this valid token on all routes prefixed with /api/admin/.

4.2 Frontend Implementation

The frontend is a modern React application built with TypeScript and Vite, designed to be fast, responsive, and maintainable.

4.2.1 Technology Stack

- **Runtime:** React (v18.x)
- **Framework:** TypeScript
- **Bundler**: Vite
- **Styling**: Utility-first CSS (Tailwind-like)
- **Key Libraries:** react-router-dom (routing), axios (or fetch wrapper for API calls), html2canvas, jspdf (for PDF generation).

4.2.2 Component Structure

The application is organized into logical directories:

- src/pages/: Contains top-level route components (e.g., Register.tsx, AdminDashboard.tsx, CertificateView.tsx).
- src/components/: Contains reusable UI elements (e.g., Button.tsx, Dialog.tsx, ProtectedRoute.tsx).
- src/lib/: Contains utility functions, including the centralized api.ts wrapper for fetch.

- src/contexts/: Contains React Context providers, such as AuthContext.tsx for managing the admin's login state.

. 4.2.3 State Management

For most of the application, local component state (useState) and prop-drilling are sufficient. For cross-cutting concerns like user authentication, React Context is used. The AuthContext provides the current user object and login/logout functions to all components wrapped within it. A ProtectedRoute.tsx component wraps all admin pages, checking the AuthContext and redirecting to the login page if the user is not authenticated.:

4.2.4 Certificate Generation and Rendering

A critical frontend feature is the generation of the A4 certificate.

1. **CertificateView.tsx:** This component is a React component styled to look *exactly* like the official A4 paper certificate. It uses precise width, height, font-size, and position properties, with embedded assets like the coat of arms. It receives the record data as props.
2. **PDF Download:** The download functionality is handled by html2canvas and jsPDF.

Table 4.1: Screenshot of the Rendered A4 Certificate Template (in-browser)

Schedule–12
(Related with Rule 7)

Government of Nepal
Ministry of Federal Affairs and Local Development

Office of Local Registrar
Province 1, Panchthar District, Phidim Municipality

जन्म दर्ता प्रमाणपत्र

(Birth Registration Certificate)

Registration No.: **BC-2025-70139** Date of Registration: **2025-11-06T11:38:55.361Z**

This is to certify, as per the birth register maintained at this office and the information provided by **the informant** in the information form of schedule 2, that Mr/Ms. **Nuna Jabegu**, son/daughter of **Mr. Ningsang Jabegu** and **Mrs. Poonam Tumbamphe (Jabegu)**, a resident of Ward No. **10**, **Phidim**, was born on **2025-10-27** (AD: 2025-10-27) at **home**.

पूरा नाम / Full Name : Nuna Jabegu	**जन्म मिति / Date of Birth :** 2025-10-27
लिङ्ग / Sex : female	**स्थायी ठेगाना / Permanent Address :** Yangnam - 5, Panchthar
जन्मस्थान / Birth Place : home	
बाबुको विवरण / Father's Details **पूरा नाम / Full Name :** Ningsang Jabegu **NIN / Citizenship No. :** 123-456-78-9	**आमाको विवरण / Mother's Details** **पूरा नाम / Full Name :** Poonam Tumbamphe **NIN / Citizenship No. :** 321-654-87-6
लिङ्ग / Sex : female	**जन्मस्थान / Birth Place :** home
सूचकको विवरण / Informant's Details **पूरा नाम / Full Name :** Ningsang Jabegu **Relationship::** Father **NIN / Citizenship No.:** 123-456-78-9	

Date, District and Citizenship No. if Citizenship Certificate is Issued to :

A. Father : 123-456-78-9

B. Mother : 321-654-87-6

Local registrar's :

Signature :

Name and surname: Dr. Binod Kumar Bhattarai

Date: 2025-11-06T11:38:55.361Z

This certificate is auto-generated by the Online Birth Registration System.

Chapter 5: Alogrithms and Data Processing

A significant challenge in any real-world registration system is data verification. Users may make typographical errors, use different spellings (transliteration), or omit middle names. This system addresses this challenge by implementing a server-side fuzzy search algorithm.

5.1 The Need for Fuzzy Matching in Verification

If a verifier searches for a certificate, an exact database match is too brittle. A query for "Ningsang Jabegu" would fail if the record is "Ningsang Jabegoo" or "Ningsa Jabegu." Similarly, an exact date match for "1999-10-27" would fail if the query was "1999-10-28."

The system's verification endpoint (GET /api/birth-record/search) is designed to overcome this. It uses a combined scoring heuristic based on:

1. **Name Similarity:** How "close" the query name is to the name in the record.
2. **Date of Birth Match:** Whether the query DOB matches the record's DOB exactly.

5.2 Levenshtein Distance Algorithm

To quantify "name similarity," the Levenshtein distance algorithm was chosen.

5.2.1 Defination

The Levenshtein distance between two strings (e.g., a and b) is the minimum number of single-character edits (insertions, deletions, or substitutions) required to change a into b (Levenshtein, 1966).

For example, the distance between "JABEGU" and "JABEGOO" is 1 (one substitution: U -> O). The distance between "NINGSANG" and "NINGSA" is 2 (two deletions: N, G).

5.2.2 Implementation

The algorithm is implemented using **dynamic programming**, filling a 2-D matrix (table) where cell *(i, j)* stores the distance between the first *i* characters of string *a* and the first *j* characters of string *b*.

Recurrence relation

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j)+1 & \text{(deletion)} \\ \operatorname{lev}_{a,b}(i,j-1)+1 & \text{(insertion)} \\ \operatorname{lev}_{a,b}(i-1,j-1)+\text{cost} & \text{(substitution)} \end{cases} & \text{otherwise.} \end{cases}$$

Where cost is 0 if *a[i] = b[j]* and 1 otherwise.

5.2.3

5.3 Normalized Similarity Score

The raw distance (e.g., 2) is not informative by itself. A distance of 2 between “CAT” and “DOGS” is large, but the same distance between “Ningsang” and “Nings” is small.

Hence, the distance is normalized into a score between 0 and 1:

$$\text{Similarity} = 1 - \frac{\text{Levenshtein Distance}}{\max(\text{len}(a), \text{len}(b))}$$

.

Chapter 6: Conclusion and Future Recommendations

6.1 Conclusion

This project, the "Online Birth Registration System," has demonstrated a successful and practical approach to digitizing a critical local government function. By leveraging a modern web stack (React, Node.js) and integrating a robust algorithmic solution (Levenshtein fuzzy search), the system effectively addresses the core problems of inefficiency, data loss, and fraud associated with manual, paper-based registration.

The system provides a clear and separate workflow for its key stakeholders: a simple registration and download process for citizens, a secure and efficient management dashboard for administrators, and a reliable verification portal for external agencies. The architectural choice to use a CSV backend, while a limitation in terms of scale, proved to be an effective strategy for a self-contained, portable, and easily understood academic project.

The research and development of the fuzzy search verification module (Chapter 5) is a key contribution, moving beyond simple data storage to provide an intelligent data-matching feature that solves a common real-world problem..

6.2 Future Recommendations

While the current implementation has clear limitations, particularly regarding security and scalability, it serves as a powerful proof-of-concept. The "Future Work" (Chapter 9) outlines a clear roadmap for hardening this prototype into a production-grade

system by migrating to a relational database, implementing JWT-based authentication, and enhancing the certificate generation process.

In conclusion, this project fulfills all the requirements of a seventh-semester project by delivering a complete, functional, and well-documented web application that solves a complex, real-world problem

References

Bähre, S. (2021). *TypeScript in practice: A guide to scalable and maintainable JavaScript applications*. O'Reilly Media.

Banker, S. (2018). *React and React Native: A complete guide to web and mobile development*. Apress.

Christen, P. (2012). *Data matching: Concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady, 10*(8), 707–710.

OWASP. (2023). *Top 10 web application security risks*. Open Web Application Security Project. https://owasp.org/www-project-top-ten/

Setel, P. W., Macfarlane, S. B., Szreter, S., Mikkelsen, L., Jha, P., & Stout, S. (2007). A scandal of invisibility: Making everyone count by counting everyone. *The Lancet, 370*(9598), 1569–1577. https://doi.org/10.1016/S0140-6736(07)61307-5

Teixeira, M. (2020). *Professional Node.js: A guide to building scalable, production-ready applications*. John Wiley & Sons.

United Nations. (2014). *Principles and recommendations for a vital statistics system (Revision 3)*. Department of Economic and Social Affairs, Statistics Division. https://unstats.un.org/unsd/demographic/standmeth/principles/M19Rev3en.pdf

World Health Organization (WHO). (2017). *Civil registration and vital statistics (CRVS): Strengthening systems*. https://www.who.int/health-topics/civil-registration-and-vital-statistics

Appendices

Appendix A: Key API Endpoint Reference

Method	Endpoint	Description
POST	/api/login	Authenticates an admin user.
POST	/api/register-birth	Submits a new birth registration.
GET	/api/birth-records	Lists all records (for admin).
GET	/api/birth-record/:id	Fetches a single record by its ID or CERTIFICATE_NO.
POST	/api/birth-record/:id/status	Updates a record's status (Approved/Rejected).
GET	/api/birth-record/search	Performs fuzzy search. (Query: ?name=...&dob=...)

Appendix B: Sample CSV Data Structure

Users_Accounts_Information.csv

ID,NAME,EMAIL,PASSWORD_hash,SALT,PHONE,ADDRESS,ROLE,**SECRET_CODE**

administrator-1756730772017-1,Ningsang Jabegu (Limbu),ningsanglimbu5203@gmail.com,$2b$10$kNXdqUvAwfan1o8JOHNuS.poTGXivKMLS9t7UR5Fp3A64f0Msn17.,$2b$10$kNXdqUvAwfan1o8JOHNuS.,9806060663,Phidim 10 - Panchthar,Administrator, **999A**

citizen-1756737101300-9,Poonam Limbu,citizen@brc.gov.np,$2b$10$5ZIMifhWwIsIBweDyAtoX.4rLWT/NAqvKKT2wng7BIQIR0R5zmHta,$2b$10$5ZIMifhWwIsIBweDyAtoX.,9745477122,Phidim 1 Panchthar - Nepal,Citizen,

guest-1757666350174-8,Parash Khadka,guest@brc.gov.np,$2b$10$0yZ4KONXHWExAcHC7JWGO.eouo.I2YwPm6g3/W1n./pjqVycevsoi,$2b$10$0yZ4KONXHWExAcHC7JWGO.,9876543210,Phidim 1 Panchthar- Nepal,Guest,

Users_Accounts_Information.csv

ID,CERTIFICATE_NO,CHILD_FIRST_NAME,CHILD_MIDDLE_NAME,CHILD_LAST_NAME,GENDER,DATE_OF_BIRTH,NEPALI_DOB,**PLACE_OF_BIRTH,**PROVINCE,DISTRICT,MUNICIPALITY,WARD,FATHER_FIRST_NAME,FATHER_MIDDLE_NAME,FATHER_LAST_NAME,FATHER_CITIZENSHIP_NO,MOTHER_FIRST_NAME,**MOTHER_MIDDLE_NAME,**MOTHER_LAST_NAME,MOTHER_CITIZENSHIP_NO,PERMANENT_ADDRESS,CONTACT_NUMBER,REMARKS,REGISTERED_BY,REGISTERED_AT,STATUS,REJECT_REASON

BR-1762429135361-245,BC-2025-70139,Nuna,,Jabegu,female,2025-10-27,2082-07-11,**home,**Province 1,Panchthar,Phidim,10,Ningsang,,Jabegu,123-456-78-9,Poonam,**,**Tumbamphe,321-654-87-6,"Yangnam - 5, Panchthar",9806060663,Nothing to say.,administrator-1756730772017-1,2025-11-06T11:38:55.361Z,approved,

BR-1762448253121-225,BC-2025-83179,Yukfung,Hang,Jabegu,male,2025-10-25,2082-07-09,**healthPost,**Province 1,Panchthar,Phidim,10,Ningsang,,Jabegu,123-456-78-9,Poonam,**,**Tumbamphe,321-654-87-6,Phidim 10 Yangnam,9806060663,He is my son.,citizen-1756737101300-9,2025-11-06T16:57:33.123Z,rejected,Please upload the valid citizenship number of the father.

Appendix C: Sample Generated A4 Certificate

Schedule–12
(Related with Rule 7)

Government of Nepal
Ministry of Federal Affairs and Local Development

Office of Local Registrar
Province 1, Panchthar District, Phidim Municipality

जन्म दर्ता प्रमाणपत्र

(Birth Registration Certificate)

Registration No.: **BC-2025-70139** Date of Registration: **2025-11-06T11:38:55.361Z**

This is to certify, as per the birth register maintained at this office and the information provided by **the informant** in the information form of schedule 2, that Mr/Ms. **Nuna Jabegu**, son/daughter of **Mr. Ningsang Jabegu** and **Mrs. Poonam Tumbamphe (Jabegu)**, a resident of Ward No. **10**, **Phidim**, was born on **2025-10-27** (AD: 2025-10-27) at **home**.

पूरा नाम / Full Name : Nuna Jabegu	**जन्म मिति / Date of Birth :** 2025-10-27
लिङ्ग / Sex : female	**स्थायी ठेगाना / Permanent Address :** Yangnam - 5, Panchthar
जन्मस्थान / Birth Place : home	
बाबुको विवरण / Father's Details **पूरा नाम / Full Name :** Ningsang Jabegu **NIN / Citizenship No. :** 123-456-78-9	**आमाको विवरण / Mother's Details** **पूरा नाम / Full Name :** Poonam Tumbamphe **NIN / Citizenship No. :** 321-654-87-6
लिङ्ग / Sex : female	**जन्मस्थान / Birth Place :** home
सूचकको विवरण / Informant's Details **पूरा नाम / Full Name :** Ningsang Jabegu **Relationship::** Father **NIN / Citizenship No.:** 123-456-78-9	

Date, District and Citizenship No. if Citizenship Certificate is Issued to :

A. Father : 123-456-78-9

B. Mother : 321-654-87-6

Local registrar's :

Signature :

Name and surname: Dr. Binod Kumar Bhattarai

Date: 2025-11-06T11:38:55.361Z

This certificate is auto-generated by the Online Birth Registration System.