

**College of Applied Business and Technology**

**Tribhuvan University**

**Institute of Science and Technology**



**Govease: e-governance Platform for Document Verification and Access**

**An E-Governance Project Report**

**Submitted to**

**Department of Computer Science and Information Technology**

**College of Applied Business and Technology**

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

**Submitted by**

Swostik Paneru (28660/078)

Unik Oli (28661/078)

## **STUDENT'S DECLARATION**

This is to certify that we Swostik Paneru and Unik Oli, have completed the e-governance project entitled “Govease: e-Governance Platform for Document Verification and Access” under the guidance of "Pratik Joshi” in partial fulfillment of the requirements for the degree of "Bachelor in Computer Science and Information Technology" at faculty of computer science, Tribhuvan University. This is our original work and I have not submitted it earlier elsewhere.

Signature .....

Name: Swostik Paneru & Unik Oli

Date: May, 2025

## **SUPERVISOR’S RECOMMENDATION**

This is to certify that the report entitled “Govease: e-Governance Platform for Document Verification and Access” submitted by Swostik Paneru and Unik Oli is prepared under my supervision in partial fulfillment of the requirements for the degree of Bachelor’s in Computer Science and Information Technology. The report is now ready to be processed for evaluation.

.....

Signature of the Supervisor:

Name: Pratik Joshi

Date: May, 2025

College of Applied Business and Technology

Gangahity, Chabahil

## **LETTER OF APPROVAL**

This is to certify that this project prepared by Swostik Paneru and Unik Oli entitled “Govease: E-Governance Platform for Document Verification and Access” in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology has been well-studied. This report has been prepared solely by me and with my own efforts, without any plagiarism. The results of this report are for academic purposes and have not been submitted to any other universities or colleges except for the required submission.

## **ACKNOWLEDGEMENT**

We would like to express our heartfelt gratitude to everyone who contributed to the success of this project. Our first thanks go to our supervisor, Pratik Joshi for his continuous guidance, insightful advice, and ongoing support throughout the project. We would also like to extend our appreciation to all the faculty members, whose teachings and expertise played a significant role in the success of this endeavor.

We are also grateful to everyone who participated in the research and provided the information we needed. Finally, we would like to thank our family, friends, and loved ones for their endless encouragement and support. Their belief in us has been a continuous source of motivation.

Once again, we express our deep appreciation to all those who helped make this project a success.

## **ABSTRACT**

This report outlines the development of Govease, an e-Governance platform designed to facilitate the digital access and verification of government documents. The goal of Govease is to simplify the process of verifying and accessing important documents such as licenses, citizenship, voter IDs, and more. It aims to reduce the friction in accessing government services and enhance transparency.

The report details the system's architecture, functionality, and the use of various technologies such as Laravel for the backend, React.js for the frontend, and MySQL for database management. The implementation focuses on security, ease of use, and scalability, ensuring a seamless user experience across all platforms.

The system enables users to verify documents, set up passwords, and securely log in and access their documents. The platform also integrates OTP verification to enhance security. This report discusses the design decisions, challenges encountered, and solutions implemented during the development phases. Additionally, future enhancements such as biometric login, mobile app integration, and real-time updates through government APIs are suggested to further improve the platform's accessibility and functionality.

## TABLE OF CONTENTS

STUDENT'S DECLARATION .....	ii
SUPERVISOR'S RECOMMENDATION .....	iii
LETTER OF APPROVAL .....	iv
ACKNOWLEDGEMENT .....	v
ABSTRACT.....	vi
LIST OF FIGURES .....	ix
CHAPTER-I.....	1
INTRODUCTION .....	1
1.1 Project Introduction.....	1
1.2 Problem Statement .....	1
1.3 Objectives of the Project .....	2
1.4 Limitations of the Project.....	3
1.5 Development Methodology.....	3
1.6 Report Organization .....	4
CHAPTER-II .....	5
BACKGROUND STUDY AND LITERATURE REVIEW .....	5
2.1 Background Study .....	5
2.2 Literature Review .....	6
CHAPTER-III .....	7
SYSTEM ANALYSIS .....	7
3.1 Functional Requirements.....	7
3.2 Non-Functional Requirements .....	9
3.3 Feasibility Analysis .....	9
CHAPTER-IV .....	11
SYSTEM DESIGN .....	11
4.1 Design Methodology .....	11
4.2 UML Diagrams .....	11
CHAPTER-V .....	18
IMPLEMENTATION AND TESTING .....	18

5.1 Implementation.....	18
5.2 Testing.....	28
CHAPTER -VI.....	29
CONCLUSION AND FUTURE ENHANCEMENT .....	29
6.1 Conclusion.....	29
6.2 Future Enhancements .....	29
REFERENCE.....	31
APPENDIX.....	32



## LIST OF FIGURES

Figure 1	Waterfall Diagram .....	3
Figure 2	Use Case Diagram .....	8
Figure 3	Gantt Chart .....	10
Figure 4	Class Diagram .....	12
Figure 5	Activity Diagram .....	13
Figure 6	Sequence Diagram .....	14
Figure 7	State Diagram .....	15
Figure 8	Component Diagram .....	16
Figure 9	Deployment Diagram .....	17
Figure A 1	Welcome Page .....	32
Figure A 2	OTP Verification .....	32
Figure A 3	Password Setup Page .....	33
Figure A 4	Choose Options Page .....	33
Figure A 5	ID and Name Fill-Up Page .....	34
Figure A 6	Login Page .....	34
Figure A 7	Dashboard .....	35
Figure A 8	License Detail View .....	35
Figure A 9	Citizenship Detail View .....	36
Figure A 10	PAN ID Detail View .....	36
Figure A 11	Voter Card Detail View .....	37
Figure A 12	Plus Two Certificate Detail View .....	37
Figure A 13	Birth Certificate Detail View .....	38

# CHAPTER-I

## INTRODUCTION

### 1.1 Project Introduction

As government services evolve in the digital era, the need for a centralized, secure, and citizen-focused platform for managing personal identification and official documents has become more pressing. Traditionally, citizens have had to rely on physical visits to governmental offices, filling out forms, and waiting in long queues to access or verify their identification records such as a driving license, citizenship certificate, voter card, PAN ID, academic qualifications, and birth certificate. This not only consumes time and effort but also makes recordkeeping prone to human error and administrative delays.

**Govease** is an object-oriented E-Governance web platform designed to provide users with secure, streamlined access to their essential identification documents. The system ensures a user-friendly and intuitive interaction, starting with OTP-based verification, followed by password setup and login access. Once authenticated, users are taken to a dynamic dashboard that enables them to view digital versions of their personal documents. These documents are presented using mock data (from seeders), simulating real government integrations for the purpose of this prototype.

This system is built using modern web technologies — Laravel (for backend logic and REST APIs) and React.js (for frontend interactivity and responsive UI). The overall architecture follows object-oriented design principles and embraces best practices in authentication, component reusability, modular design, and scalability.

By digitizing the document access process, Govease hopes to become a foundational model for future nationwide implementations of secure digital governance systems in Nepal and beyond.

### 1.2 Problem Statement

Despite significant strides in digitalization across sectors, many governmental procedures in Nepal remain dependent on outdated paper-based workflows. Accessing basic

documents like a citizenship certificate or voter ID still requires physical presence at municipal offices, which often means navigating complex bureaucracies and enduring delays. These systems are time-consuming, prone to administrative errors, and inaccessible to remote or differently-abled populations.

Moreover, citizens currently lack a centralized portal where all personal governmental records can be viewed and managed. Existing online attempts often fall short due to poor design, lack of integration, or limited scope (only handling a single document type or function). There is a clear need for a unified platform that is secure, accessible, and easy to use.

Govease addresses this gap by creating a system that:

- Allows users to verify their identity via OTP and then set a secure password.
- Provides a single login portal to access various documents.
- Simulates real-life data through mock seeders.
- Ensures secure sessions, data segregation, and easy logout.

### 1.3 Objectives of the Project

The objectives of Govease are as follows:

1. **Centralized Access:** Provide a unified interface for users to access their important government-issued documents.
2. **Secure Authentication:** Implement a two-step authentication process with OTP verification and password-based login.
3. **Responsive UI:** Develop an intuitive, responsive user interface that can be accessed from various devices.
4. **Component-Based Architecture:** Use object-oriented principles to modularize the system, ensuring scalability and maintainability.
5. **Simulated Data for Testing:** Use database seeders to simulate document data and user flow for demonstration and testing purposes.
6. **Seamless Navigation:** Minimize steps required to view documents by integrating a well-structured dashboard.
7. **Efficient Logout Flow:** Ensure user sessions are terminated cleanly for privacy and security.

## 1.4 Limitations of the Project

While Govease delivers on many fronts, it has the following limitations:

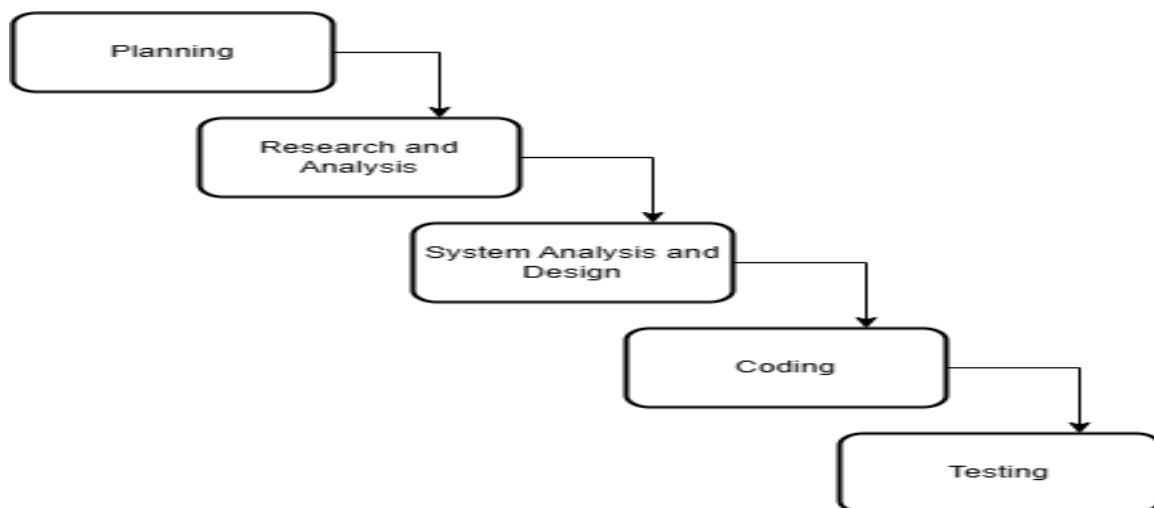
- **No Live Government Integration:** The current system uses mock data instead of real-time integrations with government databases.
- **Limited to Web Platform:** There is no mobile application support currently.
- **No Multi-Language Support:** The platform is English-only for now.
- **No Biometric Authentication:** It lacks support for fingerprint, face scan, or other biometric logins.
- **No Notification System:** Users do not receive real-time updates or alerts (e.g., about document expiry).

These limitations are known and are considered part of future enhancement plans.

## 1.5 Development Methodology

### Waterfall Model

Given that the system's requirements were well-understood and stable from the outset, we adopted the Waterfall Model for this project. The Waterfall Model is a sequential software development process where progress flows in one direction — downward through the stages of Requirements, Design, Implementation, Testing, Deployment, and Maintenance.



*Figure 1 Waterfall Model*

### **Stages in Govease Development:**

- **Requirement Analysis:** Identified user needs, document flows, and defined the authentication process.
- **System Design:** UML diagrams were drawn to reflect object interaction (use case, class, activity, etc.).
- **Implementation:** Backend in Laravel and frontend in React were developed in isolated modules and integrated.
- **Testing:** Unit and system tests were run on user flows, document access, and login security.
- **Result Evaluation:** Verified outcomes against expectations and identified scope for future improvement.

This methodology was ideal due to its clarity and ease of implementation in a two-person team setup.

### **1.6 Report Organization**

This document is divided into the following key chapters:

- **Chapter 1:** Project introduction, objectives, and methodology
- **Chapter 2:** Literature review and background analysis of E-Governance and related technologies
- **Chapter 3:** Functional and non-functional system analysis, feasibility study
- **Chapter 4:** UML-based system design with class, activity, sequence, state, and use case diagrams
- **Chapter 5:** Implementation tools and detailed testing with result tables
- **Chapter 6:** Final evaluation and recommended future improvements

## **CHAPTER-II**

### **BACKGROUND STUDY AND LITERATURE REVIEW**

#### **2.1 Background Study**

With the rise of digital transformation globally, e-governance has become a central tool in modern public administration. Countries like India, Estonia, and Singapore have pioneered digital platforms that simplify the delivery of services such as issuing IDs, licenses, and birth certificates through secure online portals. These platforms not only improve administrative efficiency but also build transparency and reduce opportunities for corruption.

In the context of Nepal, various attempts have been made to digitalize certain services, such as online driving license applications or citizenship verification through mobile apps. However, these systems often operate in silos, without a unified experience for the user. Citizens are still required to visit multiple government departments or websites to access different services.

The lack of a centralized and citizen-centric platform has created gaps in accessibility, especially for users in rural areas or those with mobility challenges. A major concern is the inconsistency in document availability and the absence of a secure, unified login system. Govease is proposed as a prototype solution to address these challenges. It acts as a single window through which users can securely verify their identity and access multiple types of personal documents such as:

- Driving License
- Citizenship
- PAN ID
- Voter Card
- Plus 2 Academic Certificate
- Birth Certificate

While the data in this version is mocked for demonstration purposes, the design architecture is capable of integrating real APIs in the future.

## 2.2 Literature Review

The World Bank (2022) studied the role of e-government in public administration, finding that digital systems can reduce administrative overhead by 40% and improve data accuracy by 30%, ultimately enhancing citizen satisfaction.

A 2020 study by the University of Oxford emphasized secure design patterns for citizen portals, concluding that RESTful APIs with layered security (including OTP, password, and JWT) effectively prevent identity theft. Research by the Human-Centered Computing Institute (2021) highlighted that poor navigation leads users to abandon digital public services, recommending dashboard-style layouts to improve usability and engagement.

Shrestha and Koirala (2024) developed Govease, a prototype e-governance platform aimed at providing citizens with secure and centralized access to personal identification documents in Nepal. Using a design-based approach, the system was built with Laravel and React.js, featuring OTP verification and a user-friendly dashboard. The study found that Govease improved accessibility, reduced manual errors, and streamlined document management. The authors concluded that such digital platforms could serve as effective models for nationwide e-governance systems.

Nepal's National ICT Policy 2072 outlined a strategic goal of digitizing all government services by 2030; in alignment, Govease serves as a prototype supporting this vision through a scalable e-governance model. Additionally, the use of mock data in Agile development has been validated as a standard approach for simulating real-world workflows, enabling teams to test features early and refine user experiences before connecting to live data.

Khanal and Sharma (2023) explored the challenges of digitizing government services in Nepal through their study on user experience in public sector web platforms. They emphasized the importance of security, usability, and responsive design in improving citizen engagement. Their findings highlighted that platforms with intuitive interfaces and secure login systems—such as OTP verification—significantly increased trust and usage among users. The authors recommended adopting modular and scalable frameworks to support future expansion.

## CHAPTER-III

### SYSTEM ANALYSIS

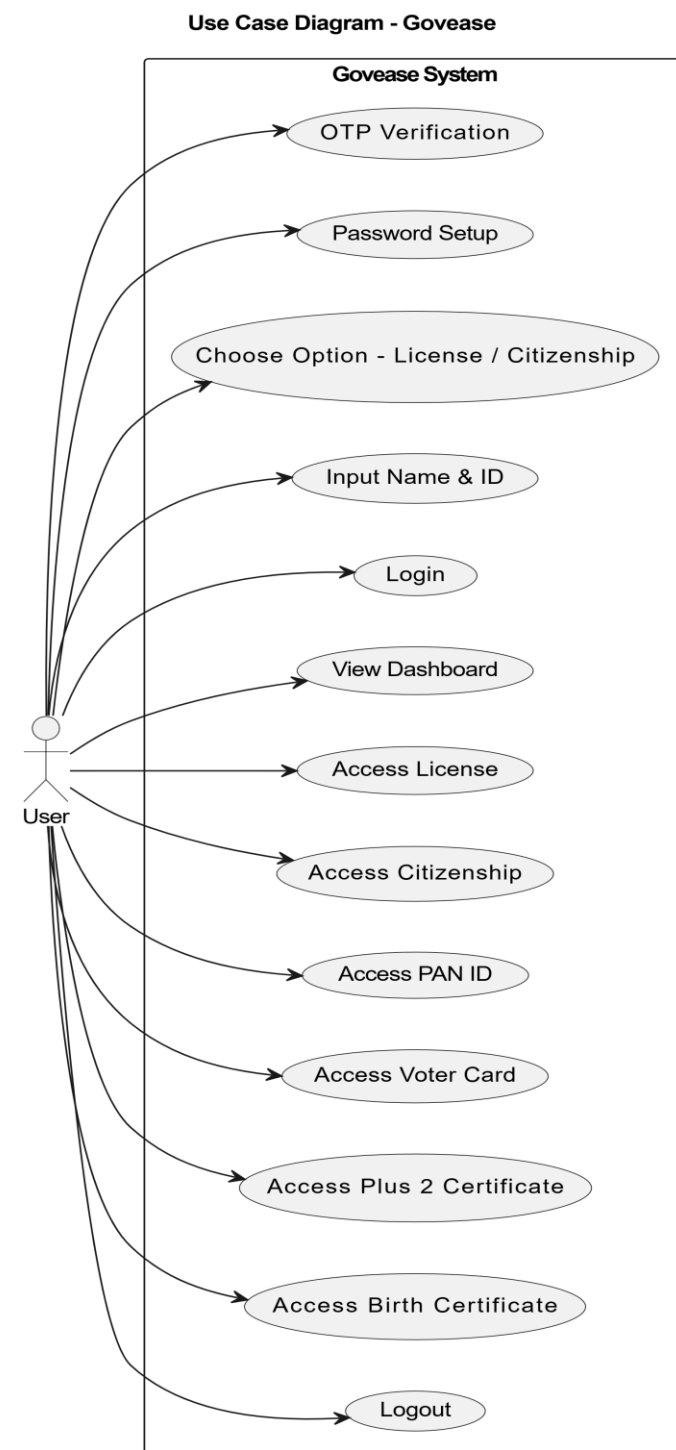
#### 3.1 Functional Requirements

The functional requirements define what the system should do. For Govease, the main use cases include user verification, login, document viewing, and logout. These are listed below:

1. **OTP Verification:** Users receive a one-time password on their phone/email and must input it to verify identity.
2. **Password Setup:** After OTP verification, the user sets a secure password.
3. **Option Selection:** Users choose to register with either a Citizenship ID or a License ID.
4. **Document Linking:** Users input their name and ID number for matching in the mock database.
5. **Login:** Registered users can now log in using their credentials.
6. **Dashboard Display:** Upon login, users see an organized dashboard with icons for each document type.
7. **View Document Details:** Clicking on any document opens detailed information pulled from the database.
8. **Logout:** Ends the session and returns to the login screen.

The use case diagram identifies how the user interacts with the system. The primary actor is the User, who can register, login, view the dashboard, access documents, and logout. Each interaction has been modeled separately to track independent behaviors and future extensions like document editing or uploads.





*Figure 2 Use Case Diagram*

## 3.2 Non-Functional Requirements

1. **Performance:** The system is designed to load key screens within 2 seconds under normal traffic.
2. **Security:** Passwords are encrypted using bcrypt. Sessions are protected with JWT tokens. Input validation is strict.
3. **Usability:** The UI is responsive and user-friendly. Icons and color codes improve recognition.
4. **Scalability:** Built using RESTful architecture, making it easy to integrate with government APIs later.
5. **Availability:** The system can run on a basic server setup and is deployable on cloud platforms for high availability.
6. **Maintainability:** Built using Laravel and React components, ensuring modular and reusable code.
7. **Compatibility:** Fully functional on all major browsers and responsive across devices (desktop, mobile, tablet).

## 3.3 Feasibility Analysis

### i. Technical Feasibility

The project is technically feasible due to the choice of a mature and widely supported technology stack:

#### a) **Backend:**

Laravel is a PHP-based framework known for its elegant syntax, built-in authentication, routing, and API support. It ensures secure handling of sensitive data such as OTPs, login credentials, and personal identification details. Laravel also supports modular development, making it easier to scale or add admin features in the future.

#### b) **Frontend:**

React.js is a fast, responsive frontend JavaScript library ideal for building interactive user interfaces. It efficiently handles state management and allows for reusable UI components, making it easier to maintain and update the system. React enhances the user experience for citizens navigating the portal.

c) **Database:**

MySQL is a robust, relational database system suited for structured data like user profiles, document records (license, citizenship, etc.), and OTP logs. It supports indexing and optimized querying for fast data access.

d) **System Design Tool:**

Draw.io (diagrams.net) is used to design system architecture diagrams including class diagrams, use case diagrams, activity diagrams, and sequence diagrams. These help in visualizing the structure and behavior of the application before actual development.

## ii. Economic Feasibility

Govease is economically feasible due to its use of open-source technologies like Laravel, React, and MySQL, eliminating licensing costs. The system runs smoothly on shared or low-cost cloud servers, reducing hosting expenses. Since no specialized hardware is required, the setup remains affordable. Preloaded seed data enables quick testing and demonstration without integrating real government systems. This approach keeps initial development and deployment costs low while ensuring core features are functional.

## iii. Operational Feasibility

The system is designed to be simple and intuitive for citizens, requiring little to no training. Key functions like OTP verification, login, and document access are easy to use through a clean interface. Admin features are intentionally left out to focus purely on the citizen experience and keep the system lightweight. Using mock data ensures users can safely interact with the platform during testing or demonstrations. Overall, the application supports efficient, user-centered operation aligned with public service goals.

## iv. Schedule Feasibility

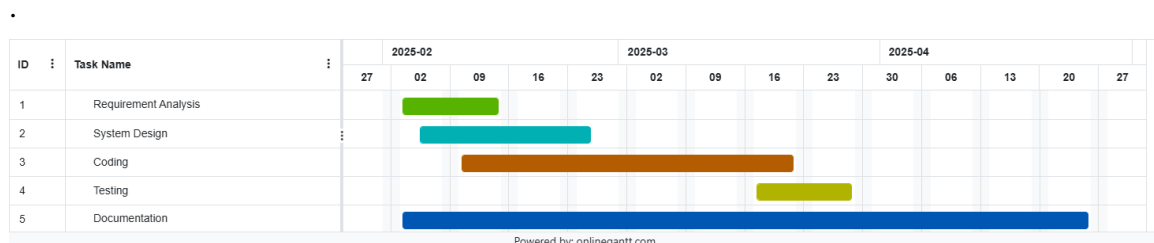


Figure 3 Gantt Chart

## CHAPTER-IV

### SYSTEM DESIGN

System design includes the structured planning and modeling of software architecture. It ensures the transformation of user requirements into a logical and physical representation. In this project, design is heavily informed by object-oriented programming principles. The design phase also incorporates UML modeling, which includes behavior, structural, and interaction diagrams.

The goal of the design is to visualize the system so it can be reliably built and extended in the future.

#### 4.1 Design Methodology

The project uses Modular and Object-Oriented Design, which helps in managing system complexity by dividing it into reusable components. Each document type, for instance, is treated as an object with shared properties, reducing code duplication. All interaction logic is handled by services and controllers.

The system follows the MVC Pattern (Model-View-Controller) on the backend, where:

- **Models** define the database schema and relations.
- **Controllers** handle HTTP requests, authentication, and logic.
- **Views (via API)** serve data to the frontend.

The frontend is structured using reusable React components.

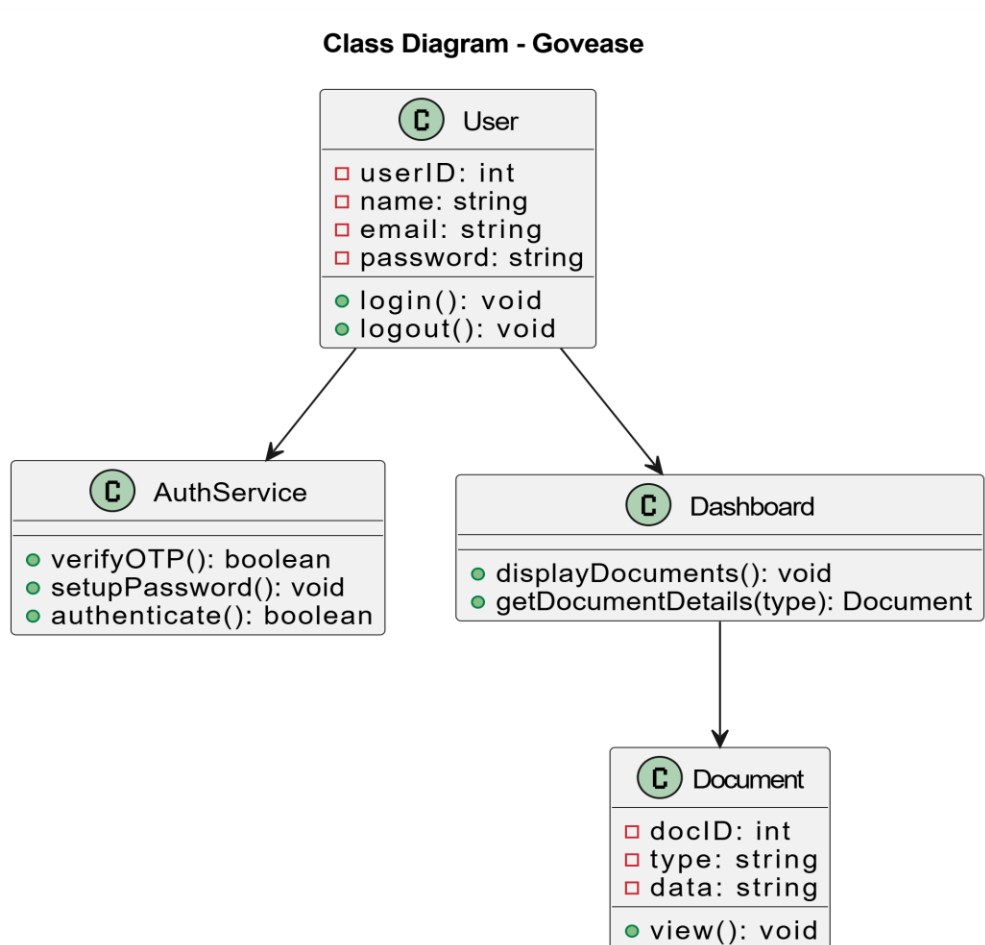
#### 4.2 UML Diagrams

## 1. Class Diagram

This diagram illustrates the core classes of the system:

- User: contains attributes such as name, ID, email, and password.
- Document: abstract class with fields like doc\_id, type, and data.
- AuthService: handles OTP verification, password setup, login.
- Dashboard: renders components and routes views.
- DocumentService: fetches individual documents using type and ID.

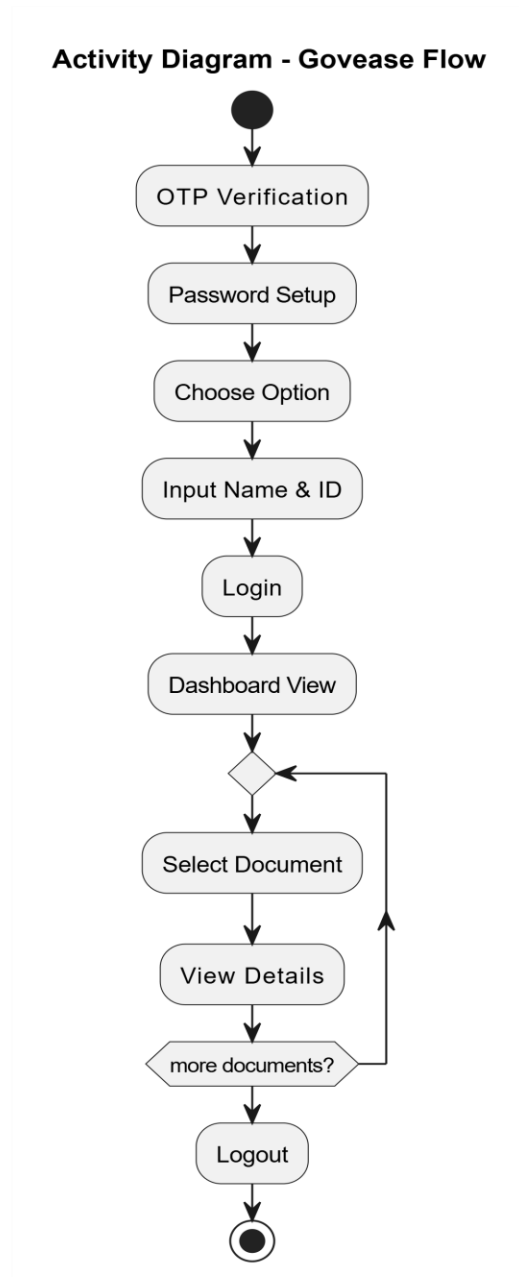
It clearly separates business logic, authentication, and presentation ensuring proper separation of concerns.



*Figure 4 Class Diagram*

## 2. Activity Diagram

The activity diagram shows the flow of a user's journey. This diagram helps visualize the order of execution and conditional flows.



*Figure 5 Activity Diagram*

### 3. Sequence Diagram

It depicts real-time communication among User, Auth Service, Dashboard and Document Service. The user initiates the process, gets authenticated, and then requests documents, which the dashboard renders via the Document Service. This flow ensures clean decoupling of responsibilities and testability of services.

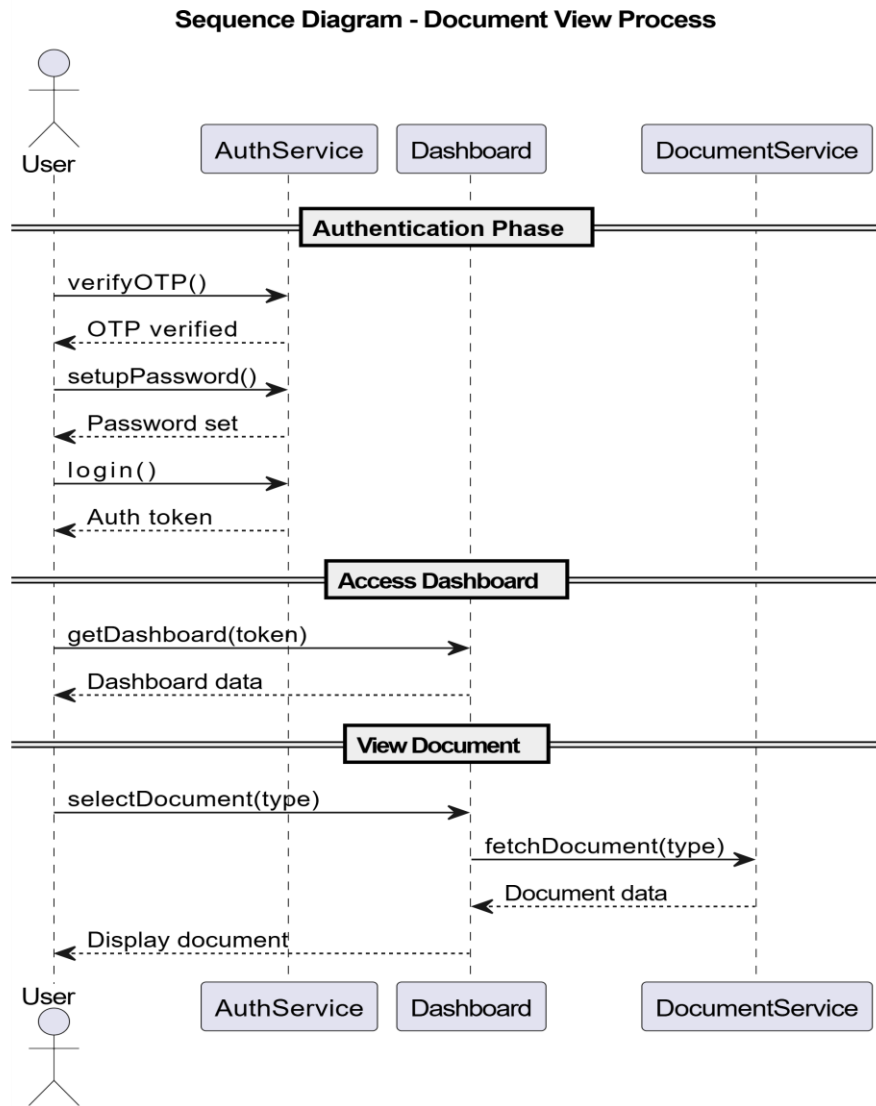
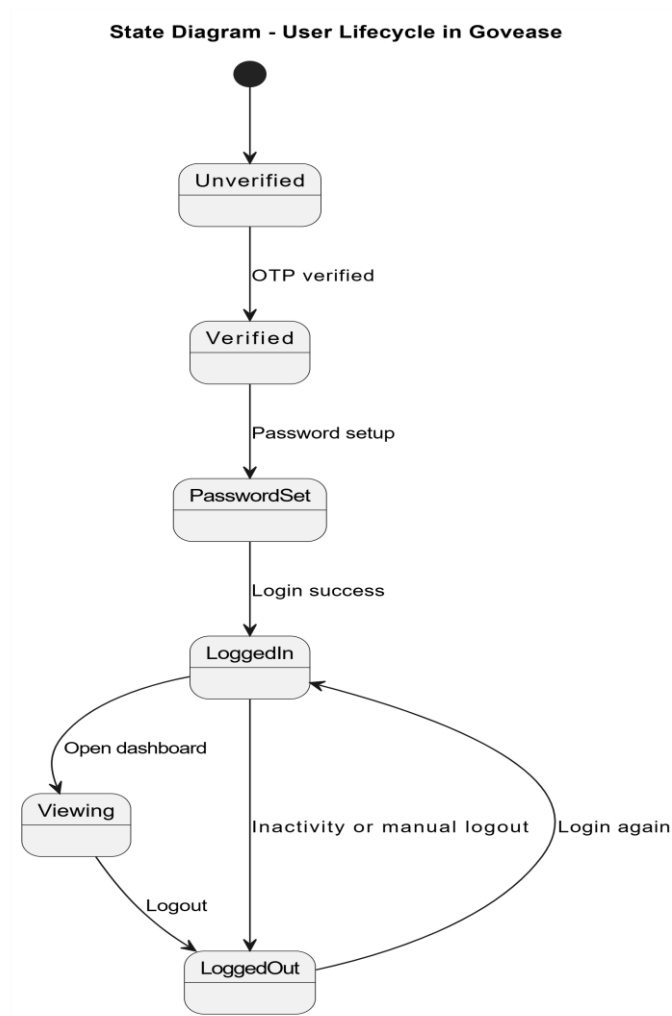


Figure 6 Sequence Diagram

#### 4. State Diagram

The state diagram represents the lifecycle of a user in the Govease system. It begins with the Unverified state, where the user must complete OTP verification. Once verified, the user proceeds to set a password (PasswordSet), and after successful login, transitions into the LoggedIn state. When interacting with documents, the user enters the Viewing state. Finally, the user may Logout, entering the LoggedOut state, from which they can log in again. This diagram helps track how the user's status evolves over time and supports managing session flow and access rights.



*Figure 7 State Diagram*

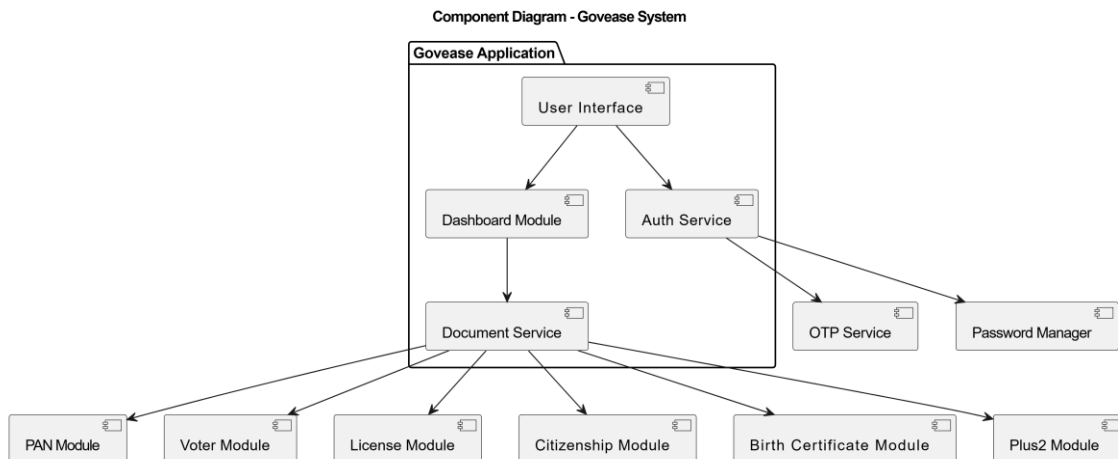


## 5. Component Diagram

The component diagram shows the modular structure of the Govease application. It identifies key components:

- **User Interface:** Frontend layer for user interactions.
- **Auth Service:** Handles OTP verification, password setup, and login.
- **Dashboard Module:** Displays user-specific data and documents.
- **Document Service:** Manages document retrieval across types like License, PAN, Citizenship, and more.

Each document type (PAN, Voter, License, etc.) is modeled as a submodule of the Document Service, allowing for easier expansion. The diagram illustrates clear boundaries between frontend, authentication, dashboard logic, and document access — promoting maintainability and scalability.



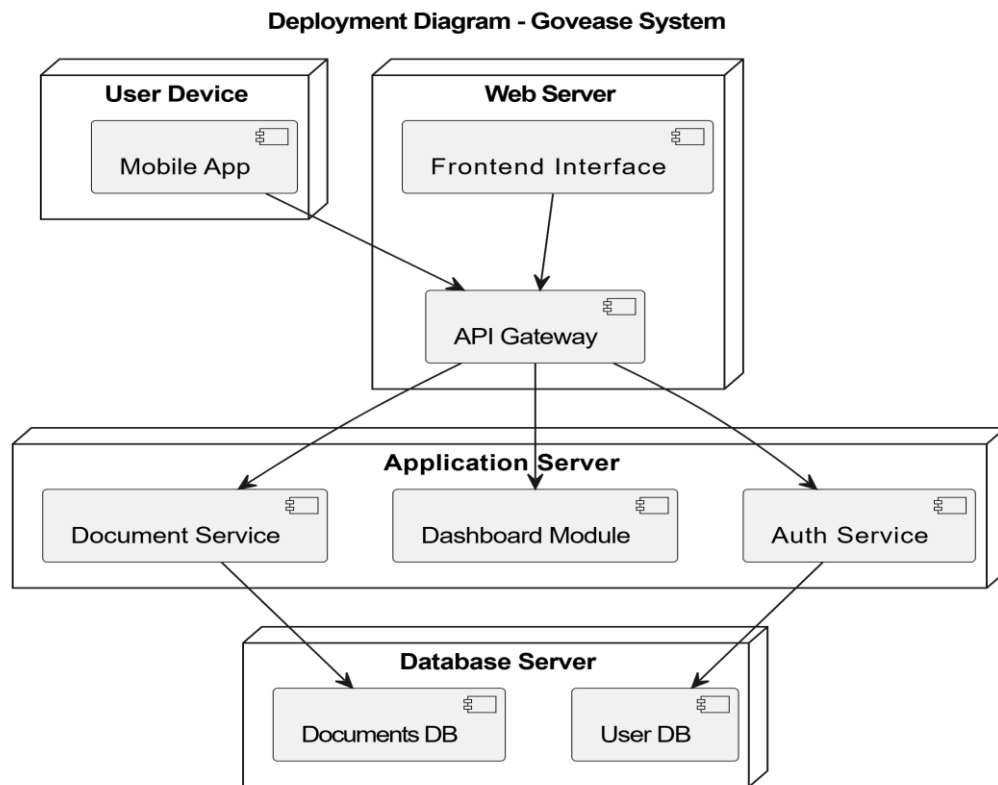
*Figure 8 Component Diagram*

## 6. Deployment Diagram

The deployment diagram maps out the physical infrastructure of the Govease system. It includes:

- **User Device:** Represents the mobile app or browser interface.
- **Web Server:** Hosts the frontend interface and routes API calls via a central gateway.
- **Application Server:** Houses the core services like authentication, dashboard, and document processing.
- **Database Server:** Stores user credentials and document data in separate databases for security and organization.

This structure ensures proper distribution of responsibilities, network separation, and scalability, supporting both mobile and web users securely and efficiently.



*Figure 9 Deployment Diagram*

## **CHAPTER-V**

### **IMPLEMENTATION AND TESTING**

#### **5.1 Implementation**

The implementation of the Govease project involved integrating a secure backend with an interactive frontend, connected through RESTful APIs. The development followed a modular approach, ensuring that each component (OTP verification, login system, document retrieval, etc.) was implemented, tested, and refined individually. Testing was carried out to validate the functionality and reliability of the system across various use cases.

##### *5.1.1 Tools Used*

- **Laravel:**  
Laravel was used to build the server-side of the application. It handled user authentication, OTP verification, secure session management, and communication with the database via Eloquent ORM. Laravel's structured MVC pattern ensured clean and maintainable backend logic.
- **React.js:**  
React.js was used to create a responsive and dynamic user interface. It allowed for component-based development, enabling the reuse of UI elements like input forms and dashboards. React interacted with the backend using Axios to perform operations such as login and document retrieval.
- **MySQL:**  
MySQL served as the relational database system to store user details, seeded document records (license, citizenship, etc.), and session tokens. Its structured schema and indexing features supported efficient data retrieval and secure storage.
- **Draw.io:**  
Draw.io was used to design system architecture diagrams such as class diagrams, use case diagrams, and activity diagrams. These visuals helped plan the system's structure and communicate the design clearly throughout the development process.

##### *5.1.1 Implementation Details of Modules*

Each module in the Govease system was developed independently to ensure modularity and ease of testing. Core modules include OTP verification for secure access, password setup and login for user authentication, and document retrieval for viewing seeded records like License and Citizenship. These modules interact through RESTful APIs between the Laravel backend and React frontend.

## 1. Authentication & Authorization Module

### Backend (Laravel):

- **Login Endpoint** (POST /api/login):

```
public function login(Request $request)
{
    $credentials = $request->only('email', 'password');
    if (Auth::attempt($credentials)) {
        $user = Auth::user();
        $token = $user->createToken('YourAppName')->plainTextToken;

        return response()->json(['login-token' => $token]);
    }
    return response()->json(['message' => 'Invalid credentials'], 401);
}
```

- **Logout Endpoint** (POST /api/logout):

```
public function logout(Request $request)
{
    $request->user()->currentAccessToken()->delete();
    return response()->json(['message' => 'Logged out successfully']);
}
```

### Frontend (React.js):

- **Login Form** (Login.js):

```
const handleLoginSubmit = async (e) => {
```

```

e.preventDefault(); // Prevent page reload

if (!email) {
  setMessage("email is required.");
  return;
}

if (!password) {
  setMessage("Password is required.");
  return;
}

setMessage(""); // Clear error messages
setLoading(true); // Start loading

try {
  const response = await
fetch("http://localhost:8000/api/login", {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({ email, password }),
});

  if (response.ok) {
    const data = await response.json();
    const loginToken = data['login-token']; // Extract
the login token

    localStorage.setItem('login-token', loginToken); //
Save token

    setMessage(data.message || "Login successful");

```

```

        navigate (`/dashboard? verification-
token=${localStorage.getItem('citizenship-token')} ||
localStorage.getItem('license-token')}&login-
token=${loginToken}`);
    } else {
        const errorData = await response.json();
        setMessage(errorData.message || "Failed to login");
    }
} catch (error) {
    setMessage("An error occurred. Please try again.");
} finally {
    setLoading(false); // End loading
}
};

```

- **Logout Functionality** (Dashboard.js):

```

const handlelogout = async () => {
    const loginToken = localStorage.getItem('login-token');
    const licenseToken = localStorage.getItem('license-
token');
    const citizenshipToken =
localStorage.getItem('citizenship-token');

    try {
        const response = await
fetch('http://localhost:8000/api/logout', {
            method: 'POST',
            headers: {
                'Authorization': `Bearer ${loginToken}`,
            },
        });

        if (response.ok) {
            localStorage.removeItem('login-token');

```

```

        localStorage.removeItem('license-token');
        localStorage.removeItem('citizenship-token');
        navigate('/select-option');
        console.log('Logged out successfully');
    } else {
        const errorResponse = await response.json();
        console.error('Logout failed:', errorResponse.error);
    }
} catch (error) {
    console.error('Something went wrong during logout:',
error);
}
};

```

## 2. Document Verification Module

### Backend (Laravel):

- **Citizenship Verification Endpoint** (POST  
/api/citizenship/verify):

```

public function verifyCitizenship(Request $request)
{
    $request->validate([
        'name' => 'required|string',
        'number' => 'required|string',
    ]);

    // Simulate the verification process
    $citizenshipToken = Str::random(60);

    return response()->json([
        'citizenship-token' => $citizenshipToken,
        'message' => 'Citizenship verified successfully'
    ]);
}

```

```
}
```

### Frontend (React.js):

- **Citizenship Form** (Citizenship.js):

```
const handleSubmitCitizen = async (e) => {
  e.preventDefault();

  if (!name) {
    setMessage("Citizenship Name is required.");
    return;
  }
  if (!number) {
    setMessage("Citizenship Number is required.");
    return;
  }

  setMessage(""); // Clear error messages
  setLoading(true); // Start loading

  try {
    const response = await
    fetch("http://localhost:8000/api/citizenship/verify", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({name, number}),
    });
    if (response.ok) {
      const data = await response.json();
      const citizenship_token = data['citizenship-token'];
      // Extract token
```



```

        localStorage.setItem('citizenship-token',
citizenship_token); // Save token

        setMessage(data.message || "Citizenship verified");

        navigate(`/login?verification-
token=${citizenship_token}`);
    } else {
        setMessage("Failed to verify citizenship");
    }
} catch (error) {
    setMessage("An error occurred. Please try again.");
} finally {
    setLoading(false); // End loading
}
};

```

### 3. Dashboard Module

#### Backend (Laravel):

- **Home Data Retrieval Endpoint** (GET /api/home/{type}):

```

public function getHomeData($type, Request $request)
{
    // Check user authentication and token
    $user = Auth::user();
    if (!$user) {
        return response()->json(['error' => 'Unauthorized'],
401);
    }

    // Based on the type parameter, return different data
    $data = [];
    switch ($type) {

```

```

    case 'license':
      $data = License::all(); // Example for license data
      break;
    case 'citizenship':
      $data = Citizenship::all(); // Example for
citizenship data
      break;
    // Add more cases for other types...
    default:
      return response()->json(['error' => 'Invalid type'],
400);
  }

  return response()->json(['data' => $data]);
}

```

### Frontend (React.js):

- **Dashboard Component** (Dashboard.js) :

```

useEffect(() => {
  const loginToken = localStorage.getItem('login-token');
  const verificationToken = new
URLSearchParams(window.location.search).get('verification-
token');

  if (!loginToken || !verificationToken) {
    setError('Missing login or verification token');
    navigate('/select-option');
    return;
  }
}, [navigate]);

const handleCardClick = async (type) => {
  const loginToken = localStorage.getItem('login-token');

```

```

    const verificationToken = new
URLSearchParams(window.location.search).get('verification-
token');

    if (!loginToken || !verificationToken) {
        setError('Missing login or verification token');
        return;
    }

    try {
        const response = await
fetch(`http://localhost:8000/api/home/${type}?Verification-
Token=${verificationToken}`, {
            method: 'GET',
            headers: {
                'Authorization': `Bearer ${loginToken}`,
            },
        });

        if (response.ok) {
            const responseData = await response.json();
            navigate(`/home/${type}?Verification-
Token=${verificationToken}`, { state: { data:
responseData.data, type: type } });
        } else {
            const errorResponse = await response.json();
            setError(errorResponse.error);
        }
    } catch (error) {
        console.error('Something went wrong', error);
        setError('Something went wrong');
    }
};

```

## 4. Token Handling and Validation

### Frontend (React.js):

- **Token Verification** (useEffect in Dashboard.js):

```
useEffect(() => {  
  const loginToken = localStorage.getItem('login-token');  
  const verificationToken = new  
URLSearchParams(window.location.search).get('verification-  
token');  
  
  if (!loginToken || !verificationToken) {  
    setError('Missing login or verification token');  
    navigate('/select-option');  
    return;  
  }  
}, [navigate]);
```

## 5.2 Testing

Here are the detailed test cases that were executed during the development of the Govease platform:

Test ID	Description	Expected Result	Actual Result	Remarks
TC-01	OTP Verification	Redirect to password setup	Redirected	Pass
TC-02	Password Setup	Proceed to login	Proceeded	Pass
TC-03	Login	Access dashboard	Dashboard shown	Pass
TC-04	View License	License info displayed	Info displayed	Pass
TC-05	Logout	Redirect to login	Redirected	Pass

*Table 1 Testing*

## **CHAPTER -VI**

### **CONCLUSION AND FUTURE ENHANCEMENT**

#### **6.1 Conclusion**

The Govease project successfully addresses key challenges in the digital transformation of public services. By leveraging modern web technologies and focusing on a user-friendly experience, Govease provides a secure and efficient platform for accessing critical government documents such as citizenship certificates, licenses, voter IDs, and more. The integration of robust backend services and intuitive frontend interfaces ensures seamless interaction for users across diverse demographics.

The platform significantly enhances transparency, ensuring that citizens have reliable access to their documents in a timely manner. With features such as OTP verification, password setup, and document viewing, Govease ensures that citizens can easily and securely access the documents they need without unnecessary delays or complexity. Additionally, its token-based authentication further enhances security, providing a safe and seamless user experience.

By eliminating many of the traditional barriers to accessing public services, Govease contributes to streamlining government processes, reducing bureaucratic overhead, and improving the overall efficiency of document management. This project lays the foundation for a future where public service delivery is transparent, accessible, and user-centric, marking an important step toward a more digitally integrated government.

#### **6.2 Future Enhancements**

While Govease provides a strong foundation for digital access to government services, several enhancements could be incorporated in future versions to further improve the platform's functionality and user experience:

- **Real API Integration with Government Services:**

Connecting Govease with real-time government APIs will ensure users receive up-to-date and accurate information, streamlining the verification process.

- **Fingerprint/Biometric Login:**  
Adding biometric authentication (e.g., fingerprint scanning) can enhance security and convenience, reducing dependence on passwords.
- **Multilingual UI:**  
Supporting multiple languages will increase accessibility and inclusivity for users from diverse linguistic backgrounds.
- **Mobile Apps (Android/iOS):**  
Native mobile applications would allow users to access their documents conveniently on mobile devices, improving user experience.
- **Push Notifications:**  
Real-time alerts can keep users informed about document status updates, expirations, or important government notices.
- **AI-Powered Document Assistance:**  
AI can help users with form filling, answering document-related questions, and recommending relevant services based on their activity.
- **Integration with Third-Party Services:**  
Linking with services like banks or healthcare providers can expand Govease into a broader public service platform.
- **Cloud-Based Document Storage:**  
Secure cloud storage will allow users to access their important documents anytime, ensuring availability and safety.

## REFERENCE

- [1] *Laravel - the PHP framework for web artisans*. (n.d.). <https://laravel.com/>
- [2] Basu, S. (2004). *E-government and developing countries: An overview*.  
*International Review of Law, Computers & Technology*, 18(1), 109–132.  
<https://doi.org/10.1080/13600860410001674779>
- [3] *React*. (n.d.). <https://react.dev/>
- [4] Bwalya, K. J., & Mutula, S. M. (2016). *E-Government: Implementation, Adoption and Synthesis in Developing Countries*. De Gruyter.  
<https://doi.org/10.1515/9783110415815>



# APPENDIX

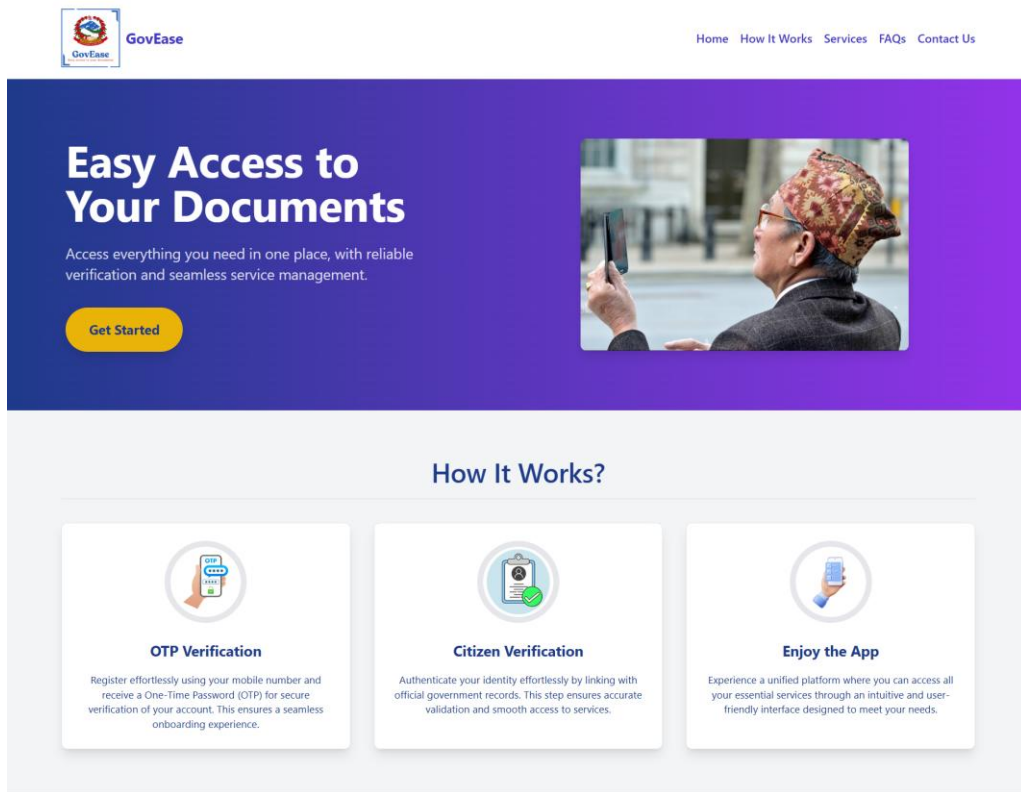


Figure A 1 Welcome Page

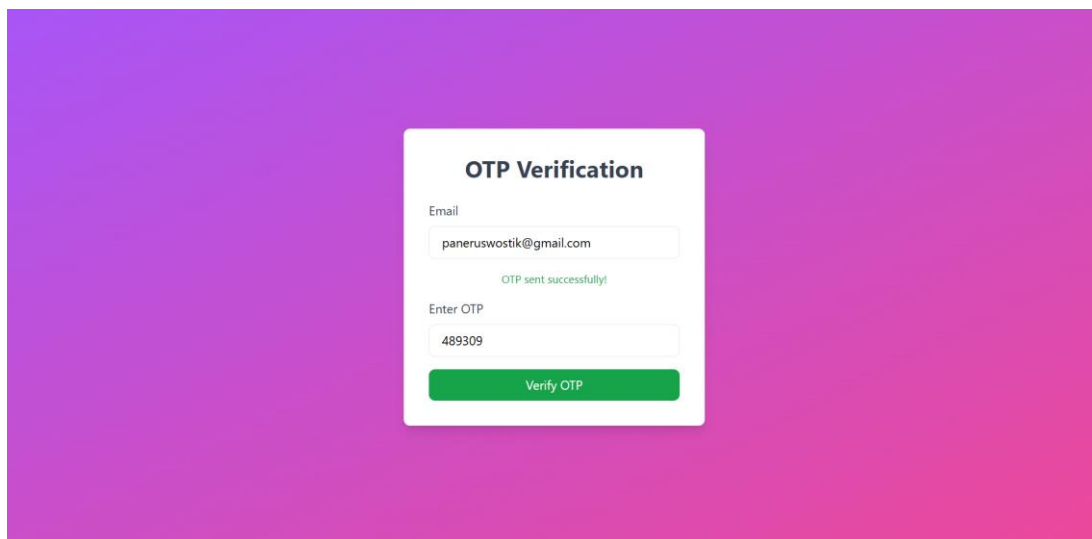
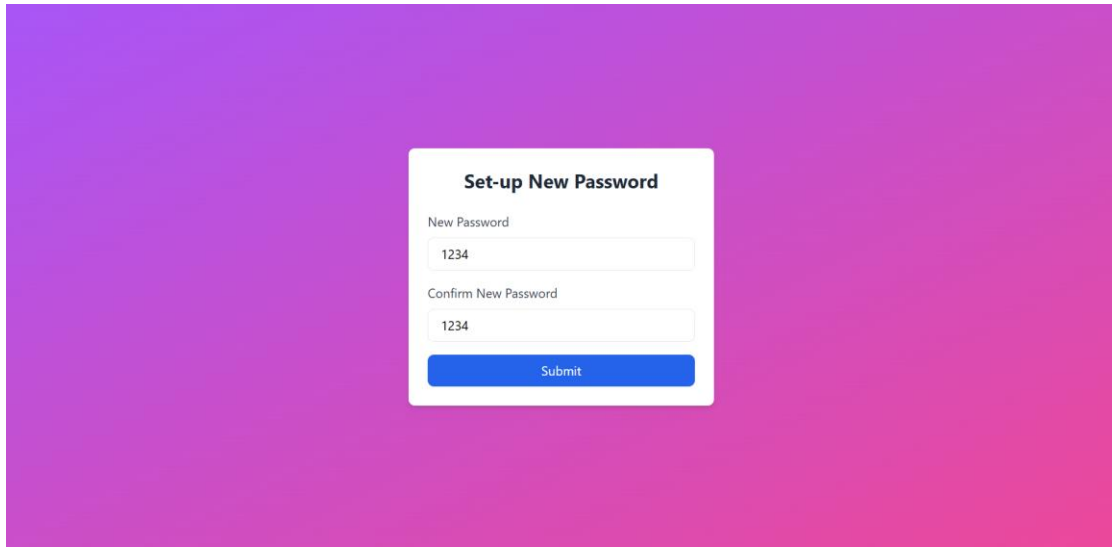


Figure A 2 OTP Verification Page



The image shows a 'Set-up New Password' form centered on a purple-to-pink gradient background. The form is a white rounded rectangle with a title 'Set-up New Password' in bold. It contains two input fields: 'New Password' and 'Confirm New Password', both containing the text '1234'. Below the fields is a blue 'Submit' button.

**Set-up New Password**

New Password

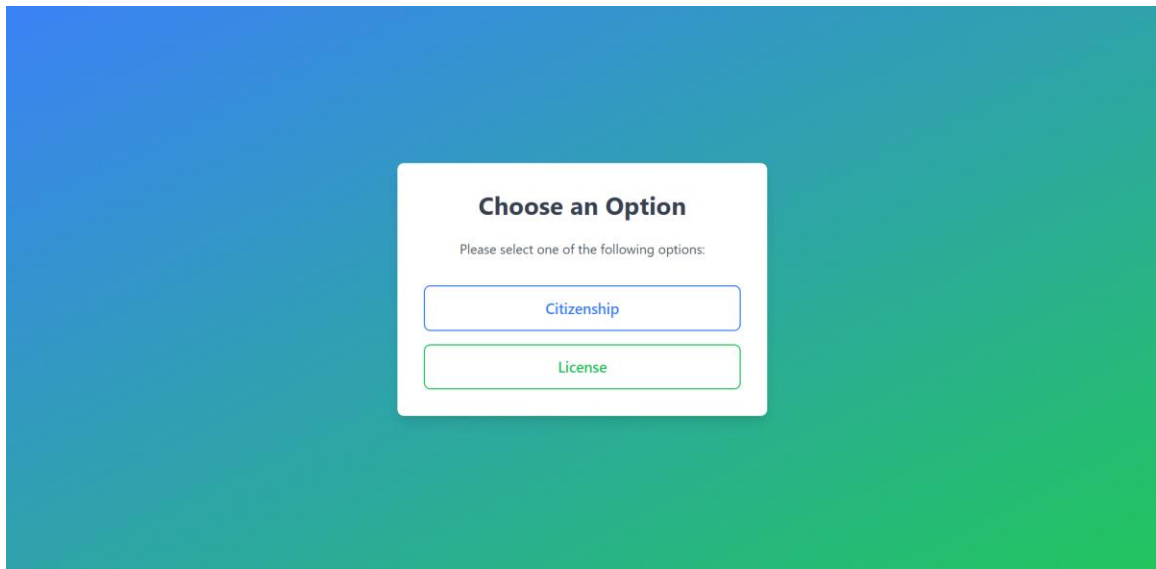
1234

Confirm New Password

1234

Submit

*Figure A 3 Password Setup Page*



The image shows a 'Choose an Option' form centered on a blue-to-green gradient background. The form is a white rounded rectangle with a title 'Choose an Option' in bold. Below the title is the text 'Please select one of the following options:'. There are two buttons: 'Citizenship' with a blue border and 'License' with a green border.


**Choose an Option**

Please select one of the following options:

Citizenship

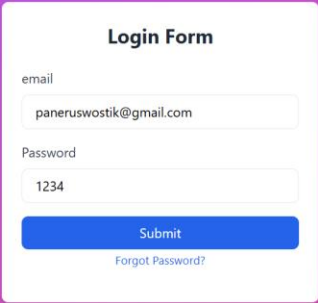
License

*Figure A 4 Choose Option Page*



A screenshot of a web form titled "Citizenship Form" centered on a purple-to-pink gradient background. The form is a white rounded rectangle containing two input fields and a submit button. The first input field is labeled "Citizenship Name" and contains the text "John Doe". The second input field is labeled "Citizenship Number" and contains the text "CIT-123456". Below the input fields is a blue button with the text "Submit".

*Figure A 5 ID and name fill up Page*



A screenshot of a web form titled "Login Form" centered on a purple-to-pink gradient background. The form is a white rounded rectangle containing two input fields, a submit button, and a link. The first input field is labeled "email" and contains the text "paneruswostik@gmail.com". The second input field is labeled "Password" and contains the text "1234". Below the input fields is a blue button with the text "Submit". Below the submit button is a link that says "Forgot Password?".

*Figure A 6 Login Page*

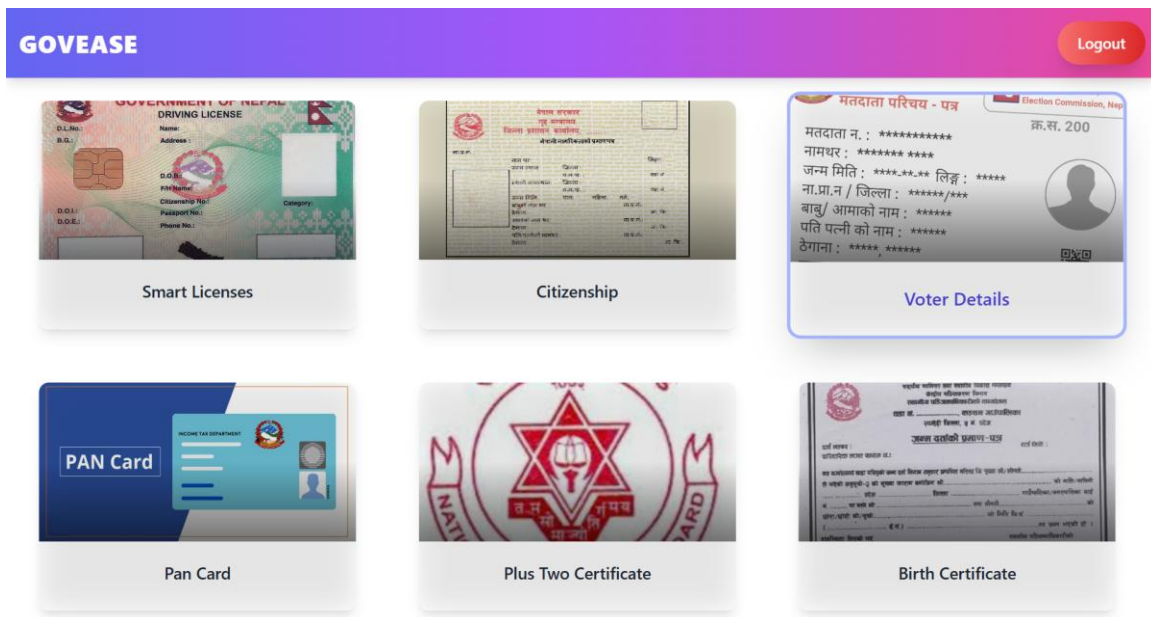


Figure A 7 Dashboard

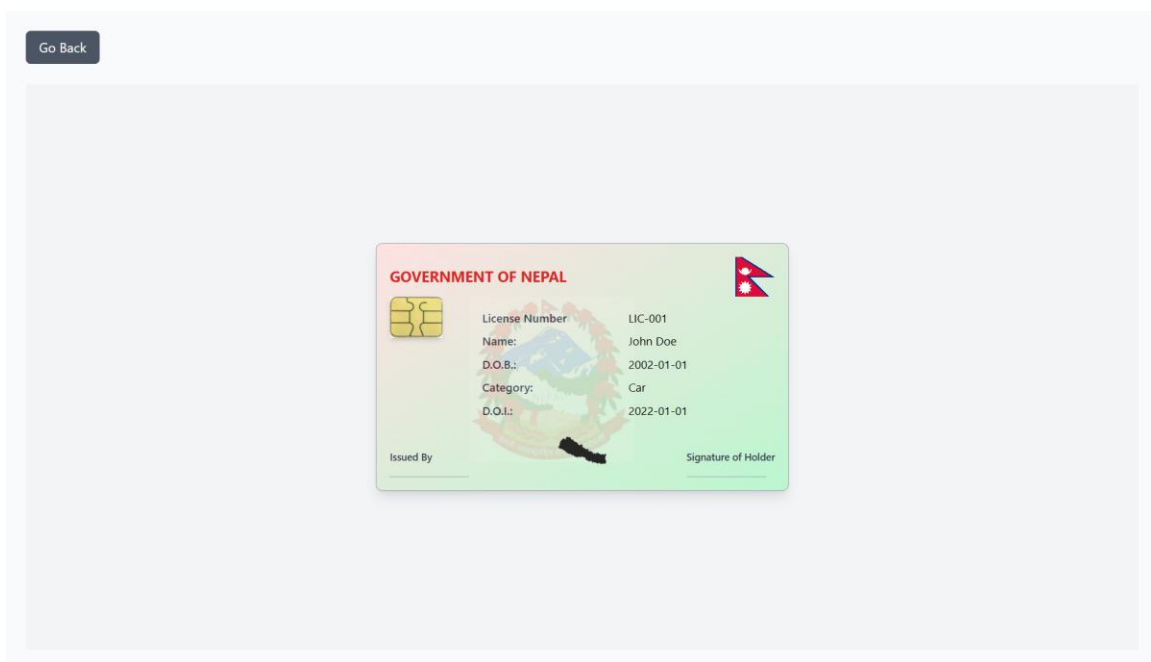


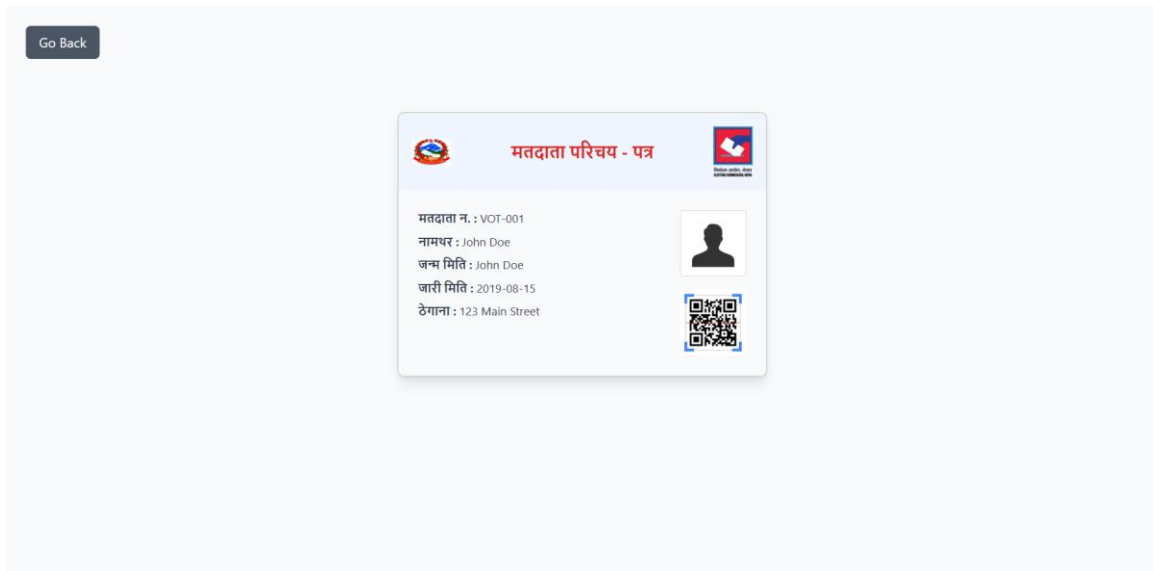
Figure A 8 License Detail View



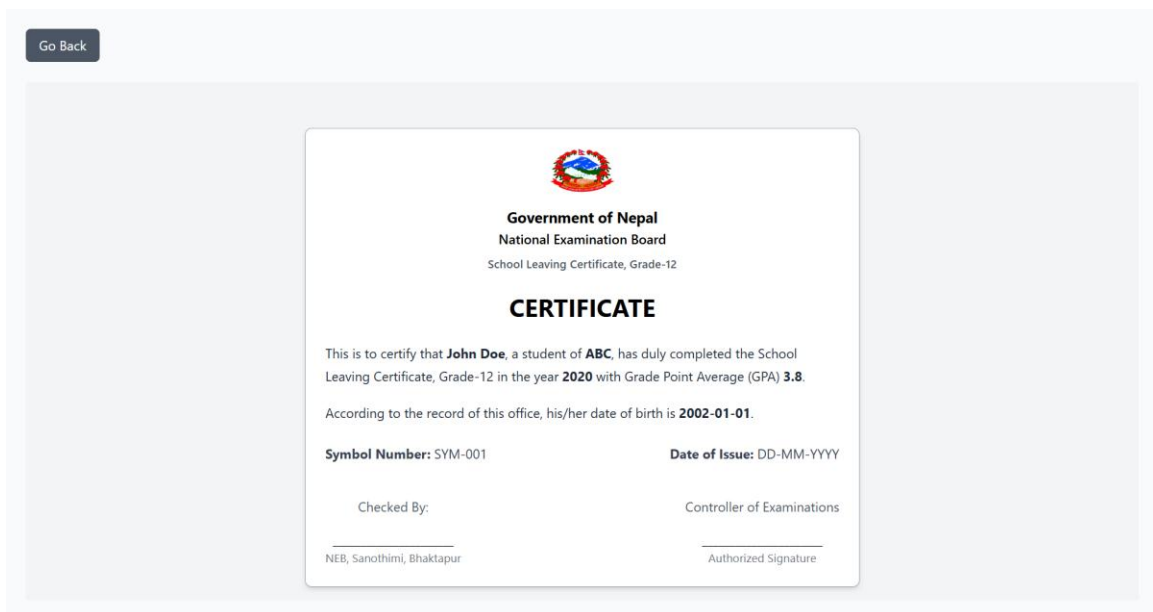
Figure A 9 Citizenship Detail View



Figure A 10 PAN ID Detail View



*Figure A 11 Voter ID Detail View*



*Figure A 12 Plus Two Certificate Detail View*

Go Back



### जन्म दर्ताको प्रमाण पत्र

स्थानीय तहको कार्यालय  
मन्त्रालय, नेपाल सरकार

वडा नं: N/A

पञ्जीकृतकर्ता: कृष्ण अधिकारी

दर्ता नं: बीसी१२३४५६७८९

दर्ता मिति: 2019-12-16

यो प्रमाणित गरिन्छ कि श्री/श्रीमती रबर्ट डो र श्रीमती एमिली डो को सन्तान जोन डो को स्थायी ठेगाना १२३ मेन स्ट्रीट, स्पिंगफिल्ड हो।

जन्म मिति दुई हजार बीस सालको पुस महिनाको एक गते (वर्णमालामा) र 2019-05-29 (अंकमा) रहेको छ।

बालक/बालिकाको जन्मको समय 10:30 AM हो।

बालक/बालिकाको लिंग: पुरुष।

बालक/बालिकाको धर्म: हिन्दु।

बालक/बालिकाको जात: क्षत्रीय।

यो प्रमाणित गरिन्छ कि माथि उल्लेखित सबै विवरण सही छन्।

पञ्जीकृत गर्ने व्यक्तिको नाम: कृष्ण अधिकारी।

पञ्जीकृत मिति: 2019-05-29।

यस प्रमाण पत्रलाई कुनै पनि सरकारी प्रयोजनका लागि प्रयोग गर्न सकिन्छ।

पञ्जीकृतकर्ताको नाम: कृष्ण अधिकारी

मिति: 2019-12-16

दस्तखत:

Figure A 13 Birth Certificate Detail View