Table of Contents

Nagarik App Clone – E-Government Portal

Chapter 1: Introduction

1.1 Introduction

The Nagarik App Clone is a web-based e-governance portal focused on National ID (NID) and citizenship verification. Built with PHP, HTML, CSS, and JavaScript and intended to run on XAMPP (Apache + MySQL), the system streamlines document submission and verification between citizens and administrators.

1.1.1 Purpose of the Project

- Digitize NID and citizenship verification to reduce manual overhead and delays.
- Provide citizens a clear, trackable submission and status workflow.
- Equip administrators with tools to review, verify/reject, and annotate submissions efficiently.

1.1.2 Project Scope

- User-facing portal for registration, login, and document uploads.
- Admin-facing portal for authentication, triage, verification, and rejection with remarks.
- Persistent storage for users, documents, and admin accounts in MySQL.
- Basic dashboard metrics for submitted/verified/rejected items.

1.1.3 Project Objectives

- Enable secure account creation and authentication for users and admins.
- Support submission of front/back document images with metadata.
- Provide real-time status views (pending, verified, rejected) to users.
- Allow admins to filter by status/type and record decisions with remarks.

Chapter 2: Design and Implementation

2.1 Web Interface Design

- Technology stack: PHP for server-side logic; HTML/CSS/JavaScript for the client; CSS themed with a blue/red gradient palette; JavaScript in js/main.js for client interactions.
- Structure aligns with the provided project tree: separate user and admin entry points, shared assets in css/ and js/.
- Responsive considerations rely on CSS; assets are organized for easy customization.

2.2 Workflow Implementation

- **User portal**: Registration and login lead to a dashboard where users choose NID or citizenship services, upload required images, and track status.
- **Admin portal**: Authenticated admins see pending submissions, open images full-size, and verify or reject with remarks. Status changes reflect on the user dashboard.
- **Status lifecycle**: pending → verified or rejected; rejected items can be resubmitted.

2.3 Database Management

- Backend database: MySQL (phpMyAdmin via XAMPP). Schema is defined in database.sql and mirrored below for quick reference.

```
CREATE TABLE IF NOT EXISTS users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  full_name VARCHAR(100) NOT NULL,
  mobile VARCHAR(15) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE IF NOT EXISTS documents (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  document_type ENUM('nid', 'citizenship') NOT NULL,
  document_number VARCHAR(50) NOT NULL,
  front_image VARCHAR(255) NOT NULL,
  back_image VARCHAR(255),
  status ENUM('pending', 'verified', 'rejected') DEFAULT 'pending',
  remarks TEXT,
  submitted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  verified_at TIMESTAMP NULL,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

```
CREATE TABLE IF NOT EXISTS admins (
   id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(50) NOT NULL UNIQUE,
   password VARCHAR(255) NOT NULL,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO admins (username, password)
VALUES ('admin',
'$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi');
```

2.4 Architecture Overview

- **Presentation**: login.html, register.php, dashboard.php, nid.php, citizenship.php for user flows; admin/index.php and admin/dashboard.php for admin flows.
- **Business logic**: Authentication and submission handling in auth/ (login, register, upload_document, logout).
- **Persistence**: config.php holds database connection; uploads saved under uploads/ (ensure writable permissions).
- **Session management**: PHP sessions secure authenticated endpoints for both users and admins.

2.5 Tools and Libraries

- **PHP**: server-side rendering and request handling.
- **MySQL**: relational storage for users, documents, and admins.
- **HTML/CSS/JavaScript**: front-end structure, styling, and interaction; theme uses primary blue (#1e3c72), secondary blue (#2a5298), red (#dc143c), gradient backgrounds, and card background #d4e4fa.

Chapter 3: Usage and Testing

3.1 Accessing the System

- Deploy the project folder to htdocs (e.g., C:\xampp\htdocs\e goverment).
- Start Apache and MySQL in XAMPP.
- User portal: http://localhost/e%20goverment/login.html
- Admin portal: http://localhost/e%20goverment/admin/

3.2 Setup and Data Preparation

- Create database nagarik_app via phpMyAdmin and import database.sql or run the SQL schema above.
- Default admin credentials: username admin, password admin123 (hashed in seed data).
- Ensure uploads/ directory is writable for storing document images.

3.3 Functional Flows

- **User registration/login**: Create account with mobile and password; authenticate to reach dashboard.
- **Document submission**: Choose NID or citizenship, enter document number, upload front/back images, submit.
- **Status tracking**: Dashboard lists submissions with pending, verified, or rejected; users can view verified documents.
- **Admin verification**: Admin dashboard lists pending items; admins view images, then verify or reject with remarks; rejected items become resubmittable.

3.4 Testing Scenarios

- Registration with unique mobile; duplicate mobile rejection.
- Login success/failure for users and admins.
- Upload validation: required fields, image presence, and size/type constraints (as implemented in upload_document.php).
- Status transitions: pending → verified, pending → rejected, and user visibility of updated status.
- Permission checks: authenticated routes for users/admins; session handling on logout.

Chapter 4: Requirements Specification

4.1 Functional Requirements

- User registration using mobile and password; password hashing and session management.
- User login/logout; persistent sessions for authenticated access.
- Document submission for NID and citizenship: numbers, front/back images, metadata.
- View submission history and status (pending, verified, rejected) with timestamps.
- Admin login; view queues (pending, verified, rejected) and filter by type.
- Admin decisioning: verify or reject with mandatory remarks.
- File storage for uploaded images with path references in DB.

4.2 Non-Functional Requirements

- Usability: Clear forms, helpful validation, responsive layout.
- Reliability: Durable storage in MySQL; error handling and input validation.
- Security: Password hashing, session hardening, input sanitization, basic rate limiting (future).
- Performance: Support concurrent uploads and admin reviews with minimal latency.
- Maintainability: Simple, modular PHP files and consistent directory structure.
- Portability: XAMPP-based local deployment; Apache + MySQL compatible hosting.

4.3 Constraints and Assumptions

- Running on XAMPP (Apache, PHP, MySQL) for local development and demos.
- Minimal external dependencies; standard PHP extensions assumed.
- Single-tenant database schema; multi-tenant support out of scope.

Chapter 7: Use Cases and User Stories

7.1 Primary Use Cases

- UC-01 Register Account: Citizen creates an account with mobile and password.
- UC-02 Authenticate: Citizen/admin logs in and obtains a session.
- UC-03 Submit Documents: Citizen uploads NID or citizenship images with numbers.
- UC-04 Track Status: Citizen views current status and remarks.
- UC-05 Verify/Reject: Admin opens submission, inspects images, and makes a decision with remarks.
- UC-06 Resubmit: Citizen resubmits after rejection with corrected images or data.

7.2 User Stories

- As a citizen, I want to upload clear front/back images so my verification is swift.
- As an admin, I want to filter pending NID vs citizenship to process efficiently.
- As a citizen, I want to see why I was rejected so I can fix it.
- As an admin, I need to view large images without downloading files manually.

Chapter 8: System Architecture

8.1 High-Level Architecture

- Presentation layer: HTML pages rendered by PHP; CSS for styling; JS for form enhancements.
- Application layer: PHP endpoints for auth, upload, and status changes.
- Data layer: MySQL database; file system storage for images under uploads/.

8.2 Request Flow

1. Browser requests PHP page (e.g., login, dashboard, upload form).
2. PHP processes form submissions, validates fields, updates DB records.
3. On success, server returns a new view or JSON status (where applicable).
4. Images are stored on disk; file paths saved in the documents table.

8.3 Session and Access Control

- PHP sessions identify authenticated users and admins.
- Protected routes check session variables and redirect to login if missing.
- Admin endpoints are isolated under admin/ with separate checks.

Chapter 9: Detailed Design

9.1 Modules

- Authentication (auth/): login.php, register.php, logout.php handle session lifecycle.
- Submission (auth/upload_document.php): Validates inputs, stores files, inserts document row.
- User Views: login.html, register.php, dashboard.php, nid.php, citizenship.php.
- Admin Views: admin/index.php (login), admin/dashboard.php (queue and actions), admin/logout.php.

9.2 Validation Rules

- Required fields: mobile, password for auth; document type/number and images for submissions.
- File types: limit to images (e.g., JPG/PNG); enforce size caps where configured.
- Remark required for rejection decisions.

9.3 Error Handling

- User-friendly error banners on form pages.
- Server-side try/catch around DB operations; log technical errors.
- Graceful fallback when file upload fails (do not partially persist metadata).

Chapter 10: Database Design and Optimization

10.1 Tables and Relationships

- users 1..N documents via documents.user_id with cascading deletes.
- admins standalone for admin authentication.

10.2 Indexes and Keys

- Unique index on users.mobile prevents duplicate accounts.
- Consider compound indexes on documents(document_type, status, submitted_at) for admin filters.

10.3 Data Integrity

- ENUMs constrain document_type and status to known values.
- Foreign key ensures orphaned documents are not left in DB.

10.4 Storage and Backups

- Images stored in uploads/ with sanitized filenames.
- Periodic DB dumps via mysqldump; file-level backups for uploads/.

Chapter 11: Security, Privacy, and Compliance

11.1 Authentication and Sessions

- Hash passwords using password_hash() (bcrypt) and verify with password_verify().
- Regenerate session IDs on login; set secure cookie flags when using HTTPS.

11.2 Input and File Handling

- Sanitize inputs using parameterized queries (PDO prepared statements recommended).
- Validate MIME type and extension; optionally re-encode images server-side.
- Enforce file size limits; reject executable content.

11.3 Authorization and Least Privilege

- Separate user and admin routes; deny-by-default policy.
- Restrict direct access to uploads/ via .htaccess or signed URLs pattern if needed.

11.4 Privacy and Compliance

- Store only necessary data (data minimization).
- Provide clear remarks on rejections without exposing sensitive internal notes.
- Align with local data protection laws; add consent/ToS notices.

Chapter 12: API Design (Planned)

12.1 Goals

- Provide programmatic access for mobile clients and integrations.
- Maintain parity with web features: auth, submit, status, admin actions.

12.2 Endpoint Sketches

- POST /api/auth/login → token
- POST /api/documents → create submission
- GET /api/documents → list for user
- POST /api/admin/documents/{id}/verify → verify
- POST /api/admin/documents/{id}/reject → reject with remarks

---

Chapter 13: UI/UX Design Guidelines

13.1 Design Principles

- Clarity: concise labels, inline help, and obvious actions.
- Consistency: shared styles in css/style.css; reusable form patterns.
- Accessibility: color contrast, larger touch targets, keyboard navigability.

13.2 Visual Language

- Colors: primary blue (#1e3c72), secondary blue (#2a5298), red (#dc143c), card background #d4e4fa.
- States: Use neutral greys for pending, green for verified, red for rejected.

13.3 Responsive Behavior

- Mobile-first layout; stack form fields; adaptive image previews.

Chapter 14: Implementation Details

14.1 Directory Structure

- admin/: Admin auth and dashboard.
- auth/: User auth and document upload handlers.
- css/, js/: Assets for styling and interactions.
- Root PHP: User-facing pages (login, register, dashboard, nid, citizenship).
- config.php: Database connection and bootstrap.

14.2 Configuration

- Store DB credentials in config.php; consider environment variables for production.
- Ensure uploads/ is writable by the web server user.

14.3 Logging and Monitoring (Future)

- PHP error logs for backend issues.
- Access logs to monitor usage and spot anomalies.

Chapter 15: Testing Strategy and Results

15.1 Test Types

- Unit-level validation for input sanitization and helper functions.
- Integration tests for auth, upload, and status transitions.
- Manual UI testing for flows: register → submit → status; admin verify/reject.

15.2 Test Cases

- Duplicate mobile registration should fail with clear message.
- Missing images or invalid types should block submission.
- Admin rejection must require remarks and reflect on user dashboard.

15.3 Results Summary

- Core happy paths functional; edge cases documented for future hardening.

Chapter 16: Performance and Scalability

16.1 Bottlenecks

- Image upload and processing; database filtering under heavy queues.

16.2 Optimizations

- Add pagination for admin lists; introduce indexes for common filters.
- Defer image resizing to background jobs (future).

16.3 Scaling Path

- Move static/image delivery to a CDN; store on object storage (S3-compatible).
- Use connection pooling and query caching where applicable.

Chapter 17: Deployment and Operations

17.1 Environments

- Local: XAMPP stack for development and demos.
- Production: Apache/Nginx + PHP-FPM + MySQL or MariaDB.

17.2 Deployment Steps

- Copy codebase; set permissions; import database.sql.
- Configure config.php; create admin seed if needed.
- Verify uploads directory and test sample flows.

17.3 Observability

- Enable access and error logs; rotate regularly.
- Future: Health endpoints and uptime checks.

Chapter 18: Maintenance and Support

18.1 Routine Tasks

- Apply security updates; rotate admin credentials; prune stale uploads if policy allows.

18.2 Data Management

- Backup schedules for DB and uploads; test restores quarterly.

18.3 Incident Response

- Document runbooks for upload failures and DB outages; communicate user-facing incidents.

Chapter 19: Accessibility and Localization

19.1 Accessibility

- WCAG-inspired checks: contrast, focus outlines, alt text for images, ARIA roles where needed.

19.2 Localization

- Prepare for i18n by avoiding hard-coded strings; centralize labels.
- Right-to-left (RTL) adjustments in CSS if applicable.

Chapter 20: Legal, Ethical, and Governance

20.1 Legal

- Comply with national data protection regulations; define data retention policies.
- Terms of Service and Privacy Policy pages (future additions).

20.2 Ethical

- Minimize data collection; provide transparency on verification criteria.
- Consider appeal workflows for contested rejections.

20.3 Governance

- Separation of duties; peer review for admin decisions in sensitive cases.

Chapter 21: Risk Management and Mitigation

21.1 Risks

- Data breach, credential stuffing, storage overrun, and biased decision-making.

21.2 Mitigations

- Strong password policies, rate limiting/captcha, secure storage, regular audits.
- Admin training and dual-control for high-risk verifications.

Chapter 22: Project Management and Timeline

22.1 Phases

- Phase 1: Prototype (auth, upload, admin decisions).
- Phase 2: UX polish, validations, and indexes.
- Phase 3: Notifications, API, and observability.

22.2 Timeline

- Weeks 1–2: Core features and DB schema.
- Weeks 3–4: Admin flows, testing, hardening.
- Weeks 5–6: API draft, documentation, and deployment runbook.

Chapter 23: Cost and Feasibility Analysis

23.1 Costs

- Hosting (LAMP stack), storage (uploads), and maintenance personnel.

23.2 Feasibility

- Technically feasible with commodity hosting; incremental features planned to control scope.

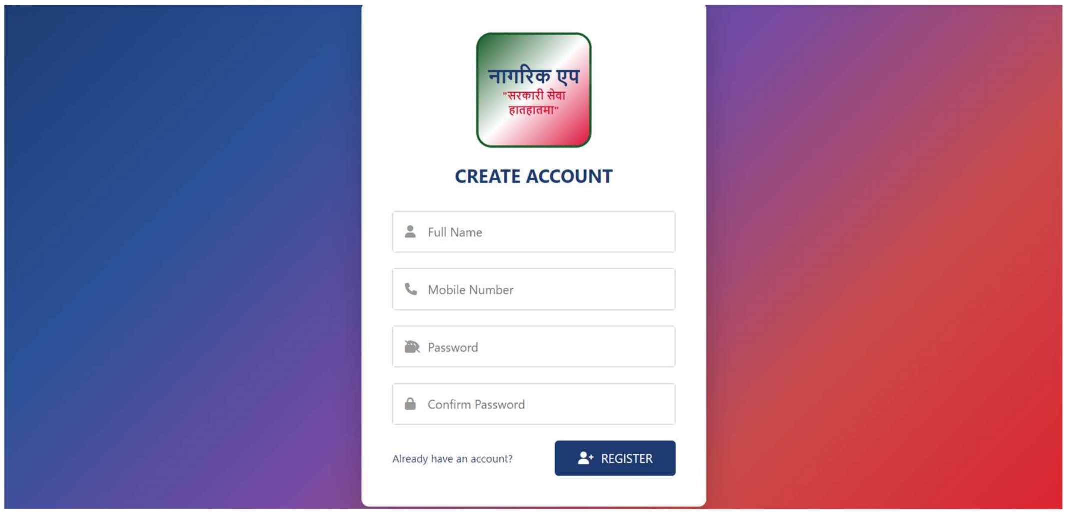
Chapter 24: Expanded Future Work

24.1 Enhancements

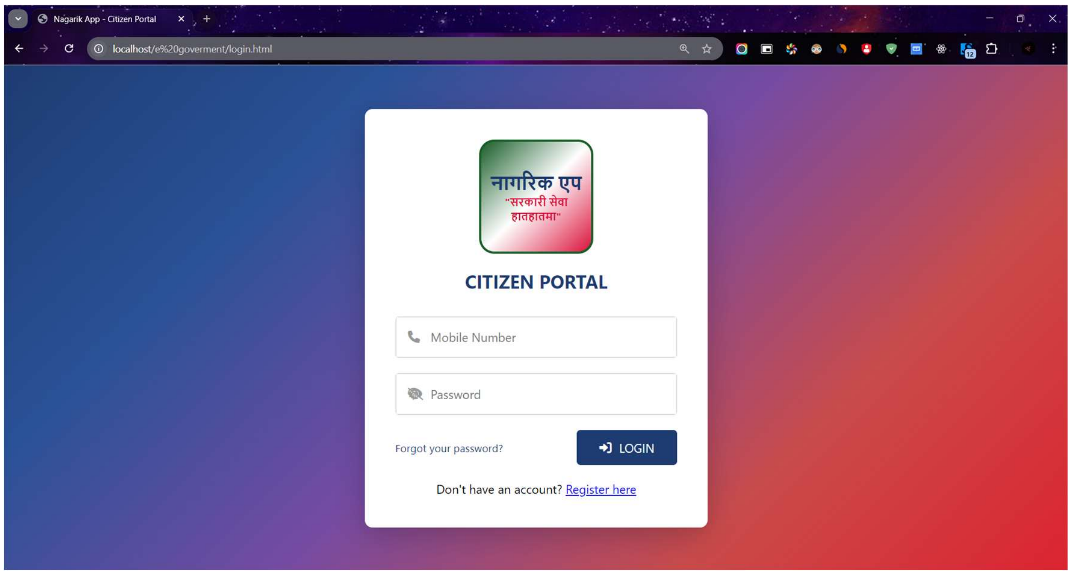
- SMS/email alerts, QR verification, audit logs, RBAC, bulk admin actions.
- Background jobs for image processing and virus scanning.
- Mobile app and public APIs with OAuth2/OIDC.

Chapter 25: Web Design Interface

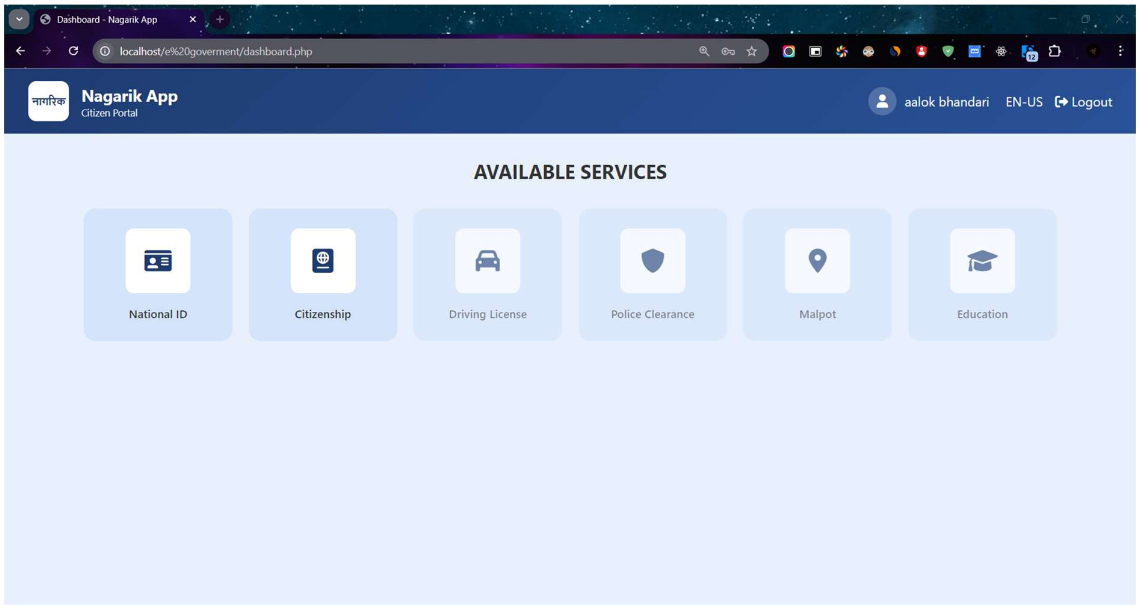
25.1 Sign-up Interface



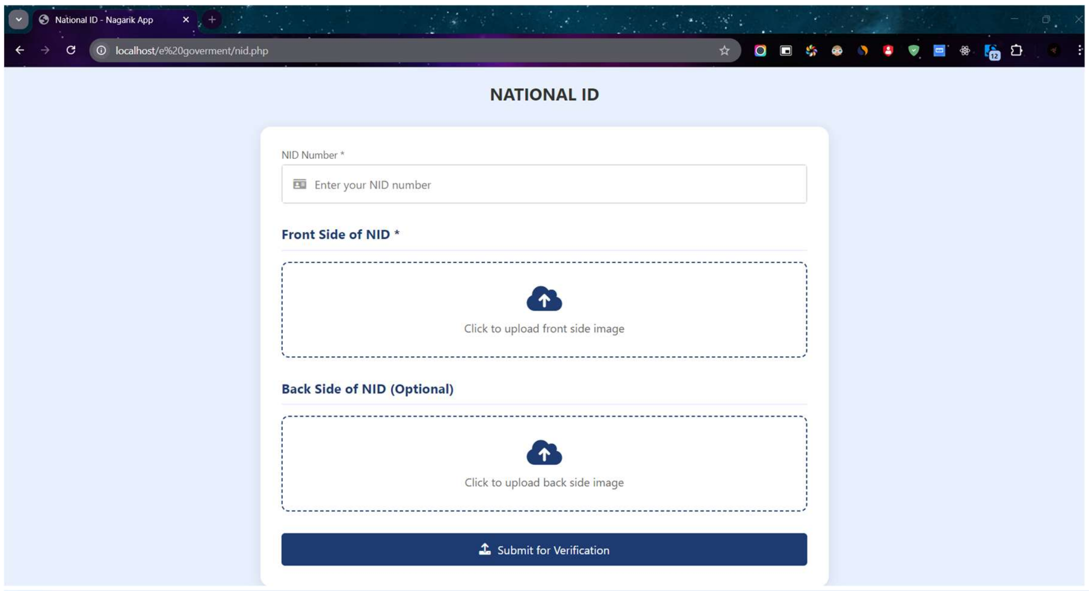
25.2 Login Interface


Nagarik App - Citizen Portal
localhost/e%20goverment/login.html
नागरिक एप
"सरकारी सेवा हातहातमा"
CITIZEN PORTAL
Mobile Number
Password
Forgot your password?
LOGIN
Don't have an account? Register here

25.3 Dashboard (Available Services)



25.4 Document Registration



National ID - Nagarik App

localhost/e%20goverment/nid.php

NATIONAL ID

NID Number *

Enter your NID number

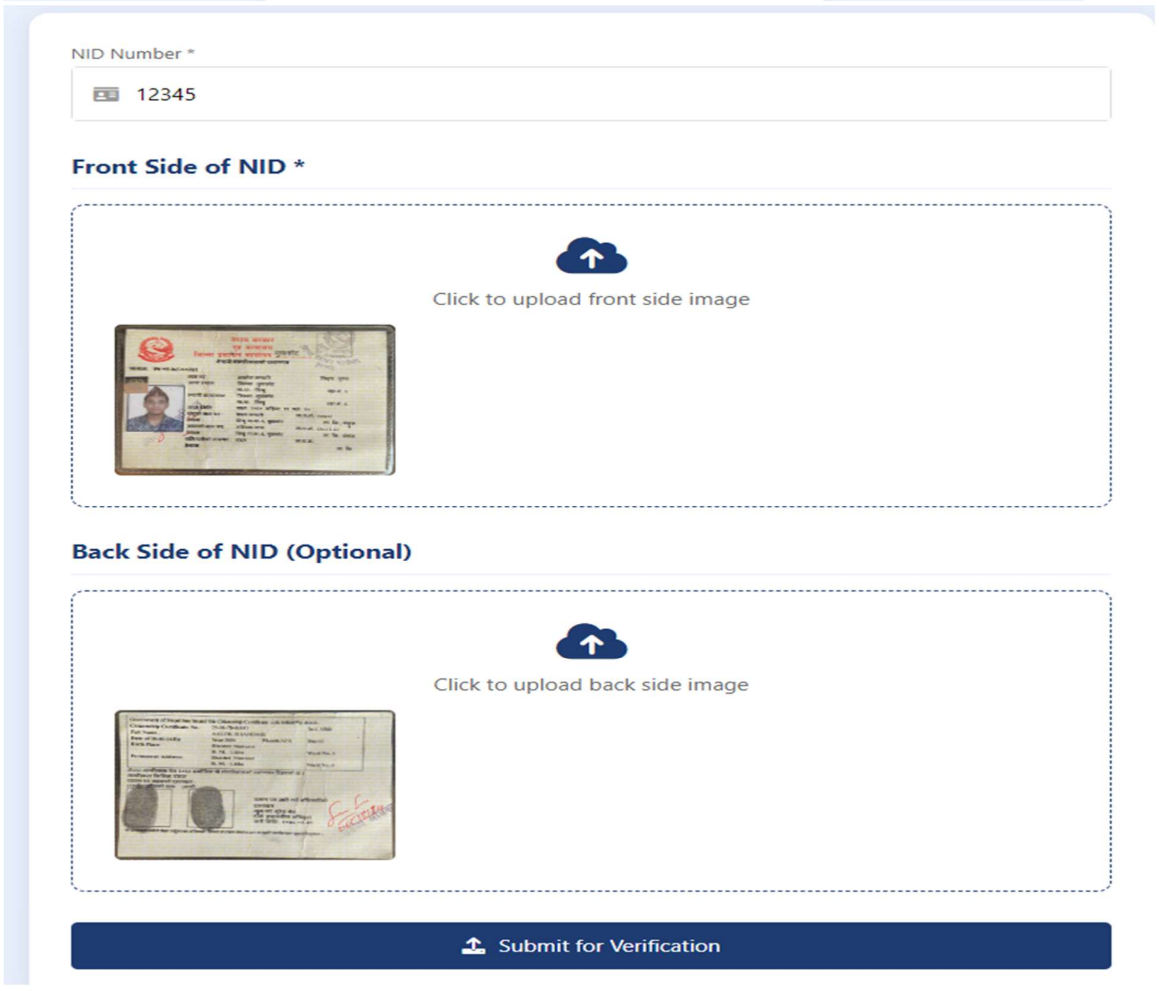
Front Side of NID *

Click to upload front side image

Back Side of NID (Optional)

Click to upload back side image

Submit for Verification



NID Number *

12345

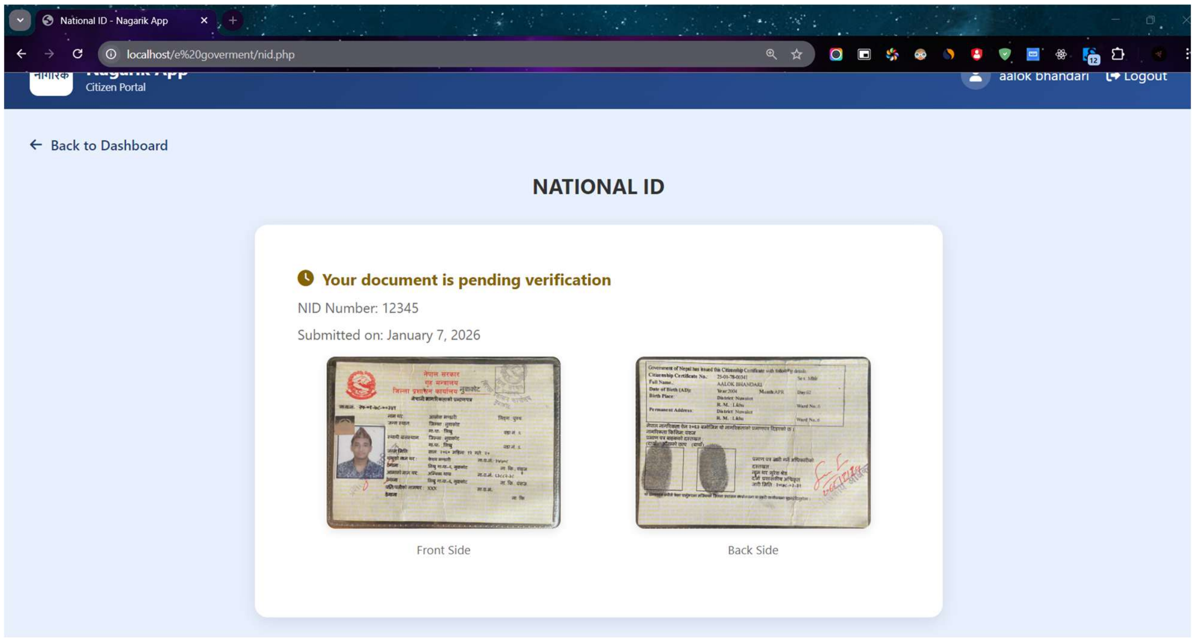
Front Side of NID *

Click to upload front side image

Back Side of NID (Optional)

Click to upload back side image

Submit for Verification


National ID - Nagarik App

localhost/e%20goverment/nid.php

Citizen Portal

aalok bhandari Logout

Back to Dashboard

NATIONAL ID

Your document is pending verification

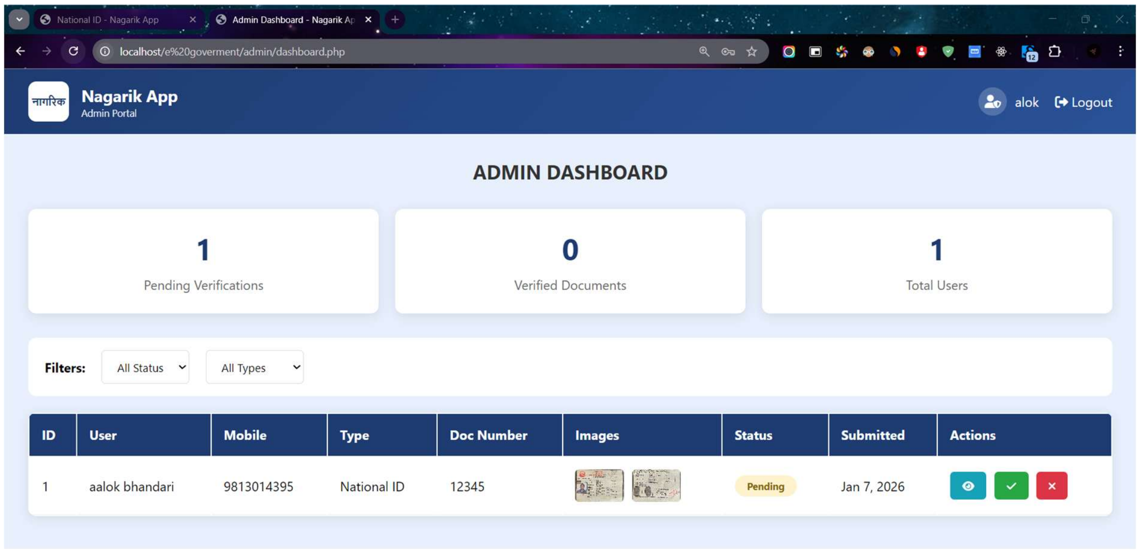
NID Number: 12345

Submitted on: January 7, 2026

Front Side

Back Side

25.5 Admin Interface


National ID - Nagarik App

Admin Dashboard - Nagarik App

localhost/e%20goverment/admin/dashboard.php

नागरिक **Nagarik App**

Admin Portal

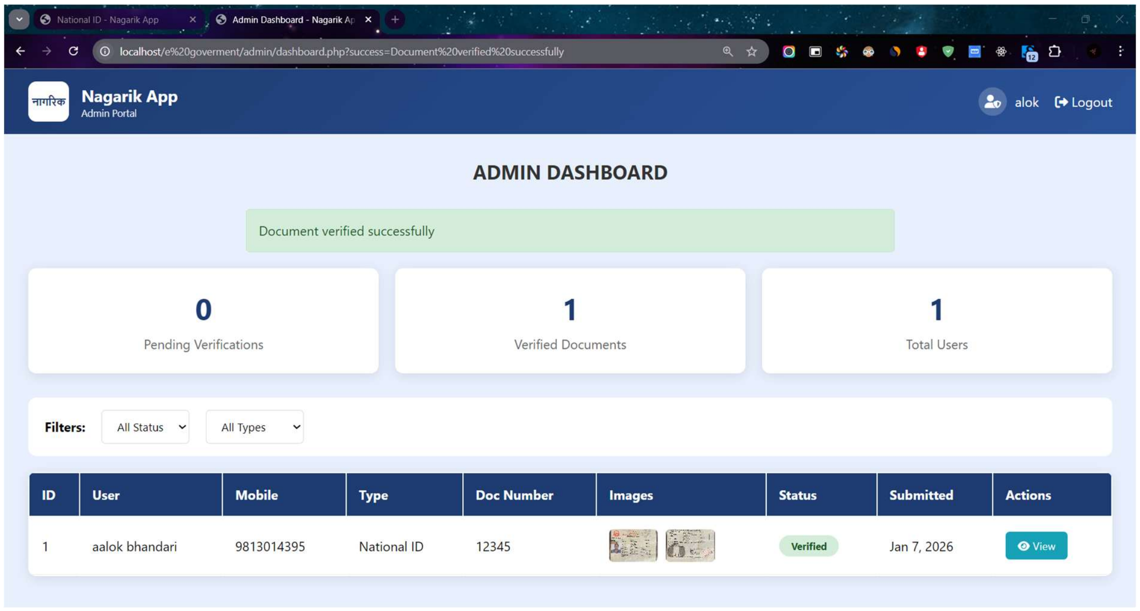
alok Logout

ADMIN DASHBOARD

1	0	1
Pending Verifications	Verified Documents	Total Users

Filters: All Status All Types

ID	User	Mobile	Type	Doc Number	Images	Status	Submitted	Actions
1	aalok bhandari	9813014395	National ID	12345		Pending	Jan 7, 2026	



National ID - Nagarik App
Admin Dashboard - Nagarik Ap
localhost/e%20goverment/admin/dashboard.php?success=Document%20verified%20successfully
नागरिक
Nagarik App
Admin Portal
alok
Logout

ADMIN DASHBOARD

Document verified successfully

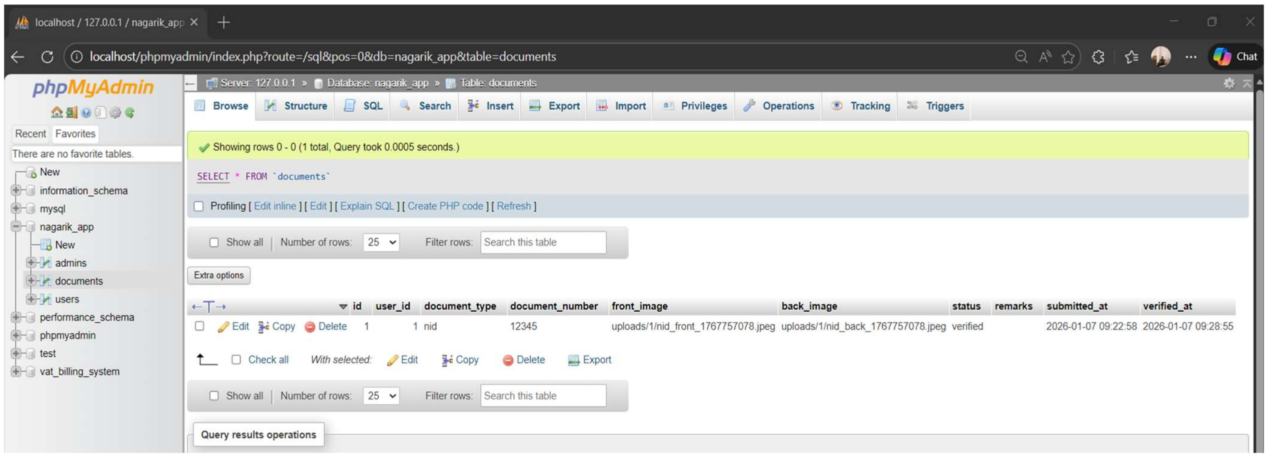
0	1	1
Pending Verifications	Verified Documents	Total Users

Filters: All Status · All Types

ID	User	Mobile	Type	Doc Number	Images	Status	Submitted	Actions
1	aalok bhandari	9813014395	National ID	12345		Verified	Jan 7, 2026	View

25.6 Database

Document Database



Showing rows 0 - 0 (1 total, Query took 0.0005 seconds.)

```
SELECT * FROM `documents`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

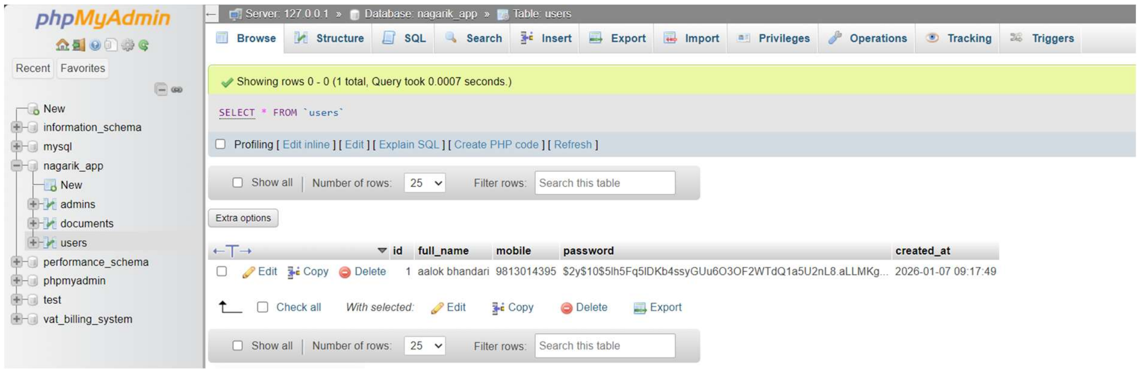
	id	user_id	document_type	document_number	front_image	back_image	status	remarks	submitted_at	verified_at
Edit Copy Delete	1	1	nid	12345	uploads/1/nid_front_1767757078.jpeg	uploads/1/nid_back_1767757078.jpeg	verified		2026-01-07 09:22:58	2026-01-07 09:28:55

Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table

Query results operations

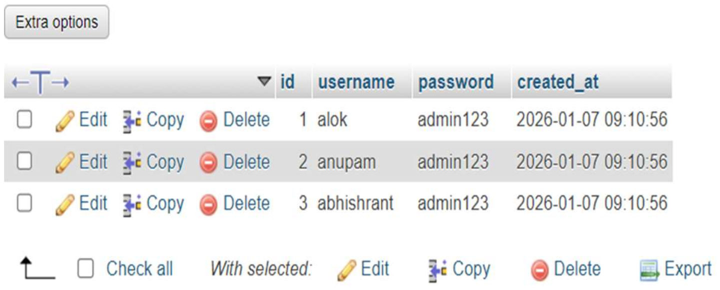
User Database


Server: 127.0.0.1 » Database: nagarik_app » Table: users

Showing rows 0 - 0 (1 total, Query took 0.0007 seconds.)

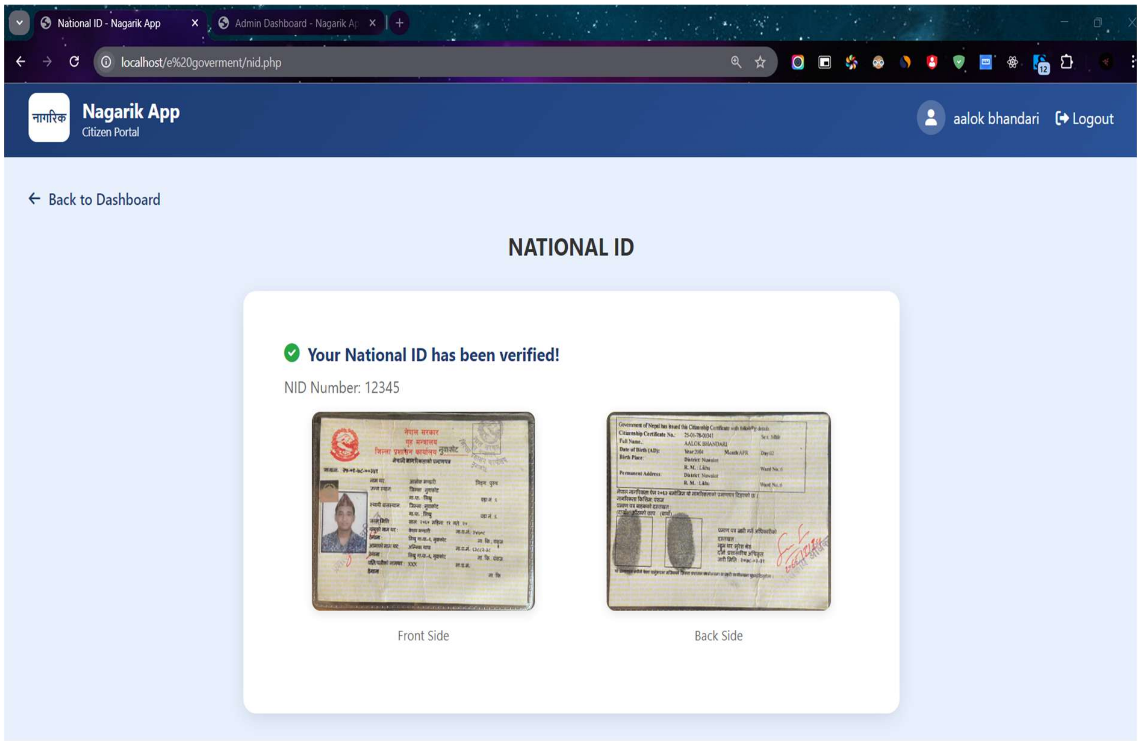
```
SELECT * FROM `users`
```

id	full_name	mobile	password	created_at
1	aalok bhandari	9813014395	$2y$10$5lh5Fq5lDKb4ssyGUu6O3OF2WTdQ1a5U2nL8.aLLMKg...	2026-01-07 09:17:49



id	username	password	created_at
1	alok	admin123	2026-01-07 09:10:56
2	anupam	admin123	2026-01-07 09:10:56
3	abhishrant	admin123	2026-01-07 09:10:56

Finally the document is verified by the admin and is successfully registered in the app.


NATIONAL ID

Your National ID has been verified!

NID Number: 12345

Chapter 26: Conclusion

26.1 Achievements

- Delivered a minimal e-governance workflow for NID/citizenship verification with separate user and admin experiences.
- Implemented secure authentication (hashed passwords) and session-based access control.
- Enabled end-to-end document submission, review, and status tracking with remarks.

26.2 Limitations

- No built-in rate limiting or captcha for abuse prevention.
- Image validation/security hardening (MIME checks, size limits) depends on deployment configuration.
- Lacks multilingual UI and accessibility-focused adjustments.

26.3 Future Enhancements

- Add email/SMS notifications on status changes.
- Implement audit trails and richer admin analytics.
- Introduce role-based access beyond single admin role.
- Provide responsive UI refinements and stronger client-side validation.
- Add APIs for mobile clients and integrate captcha/rate limiting.

Chapter 27: References

- OWASP Cheat Sheets: Authentication, Session Management, File Upload Security.
- Government digital service design manuals and accessibility guidelines.
- Database normalization and indexing best practices.

Appendices

Appendix A: Screenshots

- Home/Login page
- User dashboard
- NID submission form
- Citizenship submission form
- Admin dashboard (pending/verified/rejected views)

Appendix B: References

- Project root README for setup and feature overview.
- Database schema in database.sql for migrations.

Appendix C: Glossary and Acronyms

- NID: National ID.
- RBAC: Role-Based Access Control.
- PII: Personally Identifiable Information.
- ToS: Terms of Service.

Appendix D: API Endpoint Sketches

- Auth: /api/auth/login, /api/auth/logout.
- Citizen: /api/documents (POST, GET), /api/documents/{id} (GET).
- Admin: /api/admin/documents (GET filters), /verify, /reject.

Appendix E: Test Cases Matrix

- TC-01: Register with new mobile → success.
- TC-02: Register with duplicate mobile → error shown.
- TC-03: Upload invalid file type → blocked.
- TC-04: Admin reject without remarks → blocked.
- TC-05: Verify updates user dashboard → status visible.

Appendix F: ER Diagram Description

- Entities: User, Document, Admin.
- Relationships: User 1..N Document.
- Attributes: Document has type, number, images, status, remarks, timestamps.

Appendix G: UI Screens Wireframe Notes

- Login/Register: Minimal fields, clear errors.
- Dashboard: Card list by status with filters.
- Upload: Step-by-step with previews and constraints.
- Admin Dashboard: Queue with quick actions and detail modal.