Central African Republic Embassy Website - Complete Documentation

Version: 2.0
Date: January 5, 2026
Repository: https://github.com/groupeouambo/embassy-website

Table of Contents

Executive Summary

The Central African Republic Embassy Website is a comprehensive digital platform designed to facilitate consular services for CAR nationals in the United States, Canada, and Mexico. The platform streamlines visa applications, certificate requests, and diplomatic communications while providing robust administrative tools for embassy staff.

Key Statistics

- **4 Application Types:** Visa, Marriage Certificate, Birth Certificate, Travel Pass
- **2 Language Support:** English and French (bilingual)
- **3 User Roles:** Public Users, Registered Users, Administrator
- **24/7 Service:** Automated chat support and application tracking
- **Mobile-First:** Fully responsive design for all devices

Primary Objectives

1. Simplify consular service applications for users
2. Reduce administrative burden on embassy staff
3. Provide real-time application tracking
4. Enable secure document generation and management
5. Facilitate multilingual communication

System Architecture

High-Level Architecture

```
┌─────────────────────────────────────────────────────────────┐
│                     Frontend (React)                         │
│  ┌──────────┐  ┌──────────┐  ┌──────────┐  ┌──────────┐    │
│  │   Home   │  │   Chat   │  │ Dashboard│  │  Admin   │    │
│  └──────────┘  └──────────┘  └──────────┘  └──────────┘    │
└─────────────────────────────────────────────────────────────┘
                          │ HTTPS/REST API
┌─────────────────────────▼───────────────────────────────────┐
│                  Backend (Node.js/Express)                   │
│  ┌──────────┐  ┌──────────┐  ┌──────────┐  ┌──────────┐    │
│  │   Auth   │  │   APIs   │  │   PDF    │  │  Email   │    │
│  └──────────┘  └──────────┘  └──────────┘  └──────────┘    │
└─────────────────────────────────────────────────────────────┘
                          │
                          │
┌─────────────────────────▼───────────────────────────────────┐
│                   MySQL Database                             │
│  ┌─────────────────────────────────────────────────────┐    │
│  │  Users | Applications | Chat | Password Resets     │    │
│  └─────────────────────────────────────────────────────┘    │
└─────────────────────────────────────────────────────────────┘
```

Technology Stack

Frontend:

- React 18.2.0 with Hooks
- React Router 6.x for navigation
- CSS3 with custom responsive design
- Local storage for session management

Backend:

- Node.js with Express.js framework
- JWT (JSON Web Tokens) for authentication
- bcrypt for password hashing
- PDFKit for document generation
- bwip-js for barcode generation
- SendGrid for email delivery

Database:

- MySQL 8.x with connection pooling
- InnoDB storage engine
- ACID-compliant transactions

Deployment:

- Docker containerization
- Dokploy orchestration
- Traefik reverse proxy with SSL
- Git LFS for large media files

Features Overview

1. Multilingual Chat Widget (NEW)

Description: AI-powered chat assistant supporting English and French languages.

Key Features:

- Language selection at conversation start
- Real-time translation of all responses
- Application tracking integration
- Intelligent routing to services
- Persistent conversation history

User Flow:

1. User clicks chat widget button
2. Selects preferred language (English/French)

3. Provides name and email
4. Engages in conversation
5. Can track applications via barcode

Technical Implementation:

```
// Translation system
const t = (key) => {
  const translations = {
    en: { welcome: 'Hello! Welcome...', ... },
    fr: { welcome: 'Bonjour! Bienvenue...', ... }
  };
  return translations[language]?.[key] || translations.en[key];
};
```

Languages Supported:

- English (Default)
- French (Français)

Chat Capabilities:

- Application submission guidance
- Document requirement information
- Application status tracking
- Fee information
- Appointment scheduling

---

2. Application Tracking System (NEW)

Description: Real-time tracking of all application types using unique barcode identifiers.

Tracking Methods:

1. Chat widget modal
2. Direct URL access
3. User dashboard
4. Email notifications

Barcode Format:

- Type: Code 128 barcode
- Content: Tracking number (e.g., TRK-2026-00123)
- Fallback: APP-{id} format

Tracked Information:

- Application type (Visa, Marriage, Birth, Travel)
- Applicant name
- Current status
- Submission date
- Last update timestamp

Status Types:

- Pending Review
- Under Processing
- Approved
- Rejected
- Document Required

API Endpoint:

```
GET /api/track/:trackingNumber

Response:
{
  "id": 123,
  "type": "Visa",
  "typeDetail": "Tourist Visa",
  "applicantName": "John Doe",
  "status": "Under Processing",
  "createdAt": "2026-01-01T10:00:00Z",
  "updatedAt": "2026-01-03T14:30:00Z"
}
```

---

3. Password Recovery System (NEW)

Description: Secure email-based password reset functionality with time-limited tokens.

Recovery Flow:

Step 1: Request Reset

- User navigates to /forgot-password
- Enters registered email address
- Receives reset link via email

Step 2: Email Delivery

- SendGrid API sends email
- Link format: `/reset-password?token={JWT}`
- Token expires in 1 hour

Step 3: Password Reset

- User clicks link from email
- Token validated server-side
- Enters new password (min 6 characters)
- Password confirmation required

Step 4: Completion

- Password updated in database (bcrypt hashed)
- Token deleted from database
- Auto-redirect to signin page

Security Features:

- JWT token with 1-hour expiration
- One-time use tokens
- bcrypt password hashing (10 rounds)
- Token stored with user_id index
- HTTPS-only transmission

Database Table:

```
CREATE TABLE password_resets (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  user_id INT UNSIGNED NOT NULL,
  token TEXT NOT NULL,
  expires_at DATETIME NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE KEY unique_user_id (user_id)
);
```

---

4. Enhanced PDF Generation with Barcodes (UPDATED)

Description: All application PDFs now include scannable barcodes for tracking.

PDF Components:

1. Embassy letterhead
2. Application details

3. Barcode with tracking number
4. QR code (optional)
5. Official signature block

Barcode Specifications:

- Symbology: Code 128
- Scale: 3x
- Height: 10mm
- Includes human-readable text
- Positioned at bottom of document

Implementation:

```
const barcodeText = application.tracking_number || `APP-${application.id}`;
const barcode = await bwipjs.toBuffer({
  bcid: 'code128',
  text: barcodeText,
  scale: 3,
  height: 10,
  includetext: true
});

doc.image(barcode, 50, 700, { width: 200 });
```

PDF Types Generated:

1. Visa Application Confirmation
2. Marriage Certificate
3. Birth Certificate
4. Travel Pass (Emergency)

---

5. Admin Restriction (UPDATED)

Description: Strict access control limiting admin privileges to a single authorized email.

Admin Email: admin@usrcaembassy.org

Access Control Logic:

```
const isAdmin = user.username.toLowerCase() === 'admin@usrcaembassy.org';
```

Admin Capabilities:

- View all applications
- Update application status
- Manage users
- Access chat conversations
- View analytics
- Site configuration

Security Measures:

- Email-based authentication
- Case-insensitive comparison
- JWT token with admin flag
- Protected routes with middleware
- Session timeout after 24 hours

---

6. User Dashboard

Description: Personalized dashboard for registered users to manage applications.

Dashboard Sections:

Statistics Cards:

- Total Applications
- Pending Applications
- Approved Applications
- Rejected Applications

Recent Applications:

- Application type
- Tracking number (copyable)
- Status badge
- Submission date
- Quick actions

Available Actions:

- View application details
- Download PDF
- Track application
- Submit new application
- Update profile

Dashboard Features:

- Real-time status updates
- One-click tracking number copy
- Color-coded status badges
- Responsive card layout
- Quick access buttons

---

7. Application Types

7.1 Visa Applications

Visa Types:

1. Tourist Visa
2. Business Visa
3. Student Visa
4. Work Visa
5. Transit Visa

Required Information:

- Personal details (name, DOB, nationality)
- Passport information
- Contact information
- Purpose of visit
- Duration of stay
- Accommodation details
- Financial information

Required Documents:

- Passport copy (valid 6+ months)
- Passport photo (recent)
- Travel itinerary
- Hotel reservations
- Financial statements
- Invitation letter (if applicable)

Processing:

1. Online form submission
2. Document upload
3. Fee payment
4. Embassy review
5. Approval/rejection
6. Visa issuance

Processing Time: 5-10 business days

Fees:

- Tourist Visa: $50
- Business Visa: $100
- Student Visa: $75
- Work Visa: $150

7.2 Marriage Certificate

Purpose: Official marriage certificate issuance for CAR nationals.

Required Information:

- Spouse 1 details (full name, DOB, nationality)
- Spouse 2 details (full name, DOB, nationality)
- Marriage date
- Marriage location
- Witness information

Required Documents:

- Both spouses' passports
- Marriage license
- Passport photos (both spouses)
- Witness signatures

Processing Time: 3-5 business days

Fee: $75

7.3 Birth Certificate

Purpose: Birth certificate for children of CAR nationals.

Required Information:

- Child details (name, DOB, birthplace)
- Father details
- Mother details
- Hospital information

Required Documents:

- Hospital birth record
- Parents' passports
- Parents' marriage certificate
- Child's passport photo

Processing Time: 3-5 business days

Fee: $50

7.4 Travel Pass (Emergency)

Purpose: Emergency travel document when passport is unavailable.

Valid For:

- Lost/stolen passport situations
- Expired passport emergencies
- One-way travel to CAR

Required Information:

- Personal details
- Current location
- Emergency contact
- Reason for emergency document

Processing Time: 24-48 hours (expedited)

Fee: $200

---

8. Chat System

Architecture:

```
User → Chat Widget → Backend API → Database
  ↓
User Types Message
  ↓
POST /api/chat
  ↓
Message Stored
  ↓
Bot/Admin Response
  ↓
GET /api/chat/conversations/:id/messages
```

Features:

- Real-time messaging
- Conversation threading
- Admin reply system
- Conversation status (open/closed)
- Message history
- Typing indicators
- Read receipts

Message Types:

- User message
- Bot response
- Admin response
- System notification

Conversation Lifecycle:

1. User initiates chat
2. Provides contact info
3. Messages exchanged
4. Admin can join
5. Conversation closed

---

9. Mobile Responsiveness

Breakpoints:

- Desktop: > 1024px
- Tablet: 768px - 1023px
- Mobile (Large): 481px - 767px
- Mobile (Small): ≤ 480px

Mobile Optimizations:

Chat Widget:

- Full-screen mode on small devices
- Larger touch targets (44x44px minimum)
- 16px input font (prevents iOS zoom)
- Simplified header layout
- Bottom-anchored send button

Forms:

- Single-column layout
- Larger input fields
- Simplified validation messages
- Touch-friendly buttons
- Auto-complete disabled where needed

Dashboard:

- Stacked card layout
- Full-width action buttons
- Collapsible sections
- Optimized table scrolling
- Hamburger menu navigation

CSS Media Queries:

```
@media (max-width: 768px) {
  .chat-widget-window {
    width: calc(100vw - 24px);
    height: calc(100vh - 100px);
  }
}

@media (max-width: 480px) {
  .chat-widget-window {
    width: 100vw;
    height: 100vh;
    border-radius: 0;
  }
}
```

Database Schema

Users Table (login)

```
CREATE TABLE login (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(100) NOT NULL,
  lastname VARCHAR(100) NOT NULL,
  username VARCHAR(255) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  INDEX idx_username (username)
);
```

Visa Applications

```
CREATE TABLE visa_applications (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  tracking_number VARCHAR(50) UNIQUE,
  user_name VARCHAR(255) NOT NULL,
  first_name VARCHAR(100),
  last_name VARCHAR(100),
  date_of_birth DATE,
  nationality VARCHAR(100),
  passport_number VARCHAR(50),
  visa_type VARCHAR(50),
  purpose_of_visit TEXT,
  duration_of_stay INT,
  status VARCHAR(50) DEFAULT 'Pending Review',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

Marriage Applications

```
CREATE TABLE marriage_applications (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  tracking_number VARCHAR(50) UNIQUE,
  user_name VARCHAR(255) NOT NULL,
  spouse1_first_name VARCHAR(100),
  spouse1_last_name VARCHAR(100),
  spouse2_first_name VARCHAR(100),
  spouse2_last_name VARCHAR(100),
  marriage_date DATE,
  marriage_location VARCHAR(255),
  status VARCHAR(50) DEFAULT 'Pending Review',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

Birth Certificate Applications

```
CREATE TABLE birth_certificate_applications (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  tracking_number VARCHAR(50) UNIQUE,
  user_name VARCHAR(255) NOT NULL,
  child_first_name VARCHAR(100),
  child_last_name VARCHAR(100),
  child_date_of_birth DATE,
  child_place_of_birth VARCHAR(255),
  father_name VARCHAR(255),
  mother_name VARCHAR(255),
  status VARCHAR(50) DEFAULT 'Pending Review',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

Travel Pass Applications

```
CREATE TABLE travel_pass_applications (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  tracking_number VARCHAR(50) UNIQUE,
  user_name VARCHAR(255) NOT NULL,
  first_name VARCHAR(100),
  last_name VARCHAR(100),
  date_of_birth DATE,
  reason_for_emergency TEXT,
  current_location VARCHAR(255),
  status VARCHAR(50) DEFAULT 'Pending Review',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

Chat Conversations

```
CREATE TABLE chat_conversations (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  user_name VARCHAR(255),
  user_email VARCHAR(255),
  language VARCHAR(10) DEFAULT 'en',
  status VARCHAR(20) DEFAULT 'open',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

Chat Messages

```
CREATE TABLE chat_messages (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  conversation_id INT UNSIGNED NOT NULL,
  message TEXT NOT NULL,
  sender_type ENUM('user', 'bot', 'admin') NOT NULL,
  sender_name VARCHAR(255),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (conversation_id) REFERENCES chat_conversations(id) ON DELETE CASCADE
);
```

Password Resets

```
CREATE TABLE password_resets (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  user_id INT UNSIGNED NOT NULL,
  token TEXT NOT NULL,
  expires_at DATETIME NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE KEY unique_user_id (user_id),
  INDEX idx_token (token(255)),
  INDEX idx_expires (expires_at)
);
```

API Documentation

Authentication Endpoints

POST /api/register

Description: Register a new user account

Request Body:

```
{
  "firstname": "John",
  "lastname": "Doe",
  "username": "john.doe@email.com",
  "password": "securePassword123"
}
```

Response:

```
{
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIs...",
  "user": {
    "id": 1,
    "fullName": "John Doe",
    "username": "john.doe@email.com",
    "isAdmin": false
  }
}
```

POST /api/login

Description: Authenticate user and receive JWT token

Request Body:

```
{
  "username": "john.doe@email.com",
  "password": "securePassword123"
}
```

Response:

```
{
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIs...",
  "user": {
    "id": 1,
    "fullName": "John Doe",
    "username": "john.doe@email.com",
    "isAdmin": false
  }
}
```

Application Endpoints

POST /api/visa-application

Description: Submit a visa application

Headers:

```
Authorization: Bearer {token}
```

Request Body:

```
{
  "firstName": "John",
  "lastName": "Doe",
  "dateOfBirth": "1990-01-01",
  "nationality": "Central African Republic",
  "passportNumber": "CAR123456",
  "visaType": "Tourist Visa",
  "purposeOfVisit": "Tourism",
  "durationOfStay": 14
}
```

Response:

```
{
  "success": true,
  "applicationId": 123,
  "trackingNumber": "TRK-2026-00123",
  "message": "Visa application submitted successfully"
}
```

GET /api/applications/:username

Description: Get all applications for a user

Headers:

```
Authorization: Bearer {token}
```

Response:

```
{
  "visa": [...],
  "marriage": [...],
  "birth": [...],
  "travelPass": [...]
}
```

GET /api/track/:trackingNumber

Description: Track application status by tracking number

Response:

```
{
  "id": 123,
  "type": "Visa",
  "typeDetail": "Tourist Visa",
  "applicantName": "John Doe",
  "status": "Under Processing",
  "createdAt": "2026-01-01T10:00:00Z",
  "updatedAt": "2026-01-03T14:30:00Z"
}
```

PDF Generation Endpoints

GET /api/visa-pdf/:id

Description: Generate PDF for visa application

Headers:

```
Authorization: Bearer {token}
```

Response: PDF file download

Password Recovery Endpoints

POST /api/password-reset/request

Description: Request password reset email

Request Body:

```
{
  "email": "john.doe@email.com"
}
```

Response:

```
{
  "success": true,
  "message": "If an account exists with this email, a password reset link has been sent."
}
```

POST /api/password-reset/reset

Description: Reset password with token

Request Body:

```
{
  "token": "eyJhbGciOiJIUzI1NiIs...",
  "newPassword": "newSecurePassword123"
}
```

Response:

```
{
  "success": true,
  "message": "Password has been reset successfully"
}
```

Chat Endpoints

POST /api/chat

Description: Start chat or send message

Request Body:

```
{
  "conversationId": 1,
  "userName": "John Doe",
  "userEmail": "john.doe@email.com",
  "message": "I need help with my visa application",
  "language": "en"
}
```

Response:

```
{
  "success": true,
  "conversationId": 1,
  "message": {
    "id": 42,
    "message": "I'd be happy to help...",
    "sender_type": "bot",
    "created_at": "2026-01-05T10:00:00Z"
  }
}
```

GET /api/chat/conversations

Description: Get all chat conversations (Admin only)

Headers:

```
Authorization: Bearer {adminToken}
```

Response:

```
[
  {
    "id": 1,
    "user_name": "John Doe",
    "user_email": "john.doe@email.com",
    "language": "en",
    "status": "open",
    "created_at": "2026-01-05T09:00:00Z",
    "message_count": 5
  }
]
```

Admin Endpoints

GET /api/admin/applications

Description: Get all applications (Admin only)

Headers:

```
Authorization: Bearer {adminToken}
```

Response:

```
{
  "visa": [...],
  "marriage": [...],
  "birth": [...],
  "travelPass": [...]
}
```

PUT /api/admin/applications/:type/:id

Description: Update application status (Admin only)

Headers:

```
Authorization: Bearer {adminToken}
```

Request Body:

```
{
  "status": "Approved"
}
```

Response:

```
{
  "success": true,
  "message": "Application status updated successfully"
}
```

Deployment Guide

Prerequisites

1. **Server Requirements:**
    - Ubuntu 20.04+ or similar Linux distribution
    - Docker and Docker Compose installed
    - Minimum 2GB RAM, 20GB storage
    - Public IP address or domain
2. **External Services:**
    - MySQL 8.x database
    - SendGrid account (API key)
    - GitHub account with LFS support
    - Domain name with SSL certificate

Environment Variables

Create `.env` file in backend directory:

```
# Server Configuration
PORT=5000
NODE_ENV=production

# Database Configuration
DB_HOST=localhost
DB_PORT=3306
DB_USER=embassy_user
DB_PASSWORD=your_secure_password
DB_NAME=embassy_db

# JWT Configuration
JWT_SECRET=your_jwt_secret_key_min_32_characters

# SendGrid Configuration
SENDGRID_API_KEY=SG.xxxxxxxxxxxxxxxxxxxxx
CONTACT_EMAIL=noreply@usrcaembassy.org
CONTACT_FROM=noreply@usrcaembassy.org

# Frontend URL
FRONTEND_URL=https://usrcaembassy.org

# CORS Origins
CORS_ORIGINS=https://usrcaembassy.org,https://www.usrcaembassy.org
```

Docker Deployment

1. **Build Frontend:**

```
cd /path/to/project
npm install
npm run build
```

2. **Build Backend Docker Image:**

```
cd embassy-backend
docker build -t embassy-backend .
```

3. **Run Backend Container:**

```
docker run -d \
  --name embassy-backend \
  -p 5000:5000 \
  --env-file .env \
  embassy-backend
```

4. **Serve Frontend with Nginx:**

```
docker run -d \
  --name embassy-frontend \
  -p 80:80 -p 443:443 \
  -v $(pwd)/build:/usr/share/nginx/html \
  -v $(pwd)/nginx.conf:/etc/nginx/nginx.conf \
  nginx:alpine
```

Dokploy Deployment

1. **Connect Repository:**
   - Link GitHub repository: `groupeouambo/embassy-website`
   - Enable Git LFS support
   - Set branch: `main`

2. **Configure Build:**
   - Build provider: Nixpacks
   - Build command: `npm run build`

    - Start command: `npm run start`

3. **Set Environment Variables:**

    - Add all variables from `.env` file
    - Enable automatic deployments on push

4. **Configure Domain:**

    - Add custom domain: `usrcaembassy.org`
    - Enable SSL with Traefik
    - Configure DNS A record

5. **Deploy:**

    - Trigger manual deployment
    - Monitor build logs
    - Verify application health

Database Setup

```
# Access MySQL
mysql -u root -p

# Create database
CREATE DATABASE embassy_db CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

# Create user
CREATE USER 'embassy_user'@'localhost' IDENTIFIED BY 'your_secure_password';

# Grant privileges
GRANT ALL PRIVILEGES ON embassy_db.* TO 'embassy_user'@'localhost';
FLUSH PRIVILEGES;

# Import schema (if needed)
mysql -u embassy_user -p embassy_db < schema.sql
```

SSL Configuration (Traefik)

```
# traefik.yml
entryPoints:
  web:
    address: ":80"
    http:
      redirections:
        entryPoint:
          to: websecure
          scheme: https
  websecure:
    address: ":443"

certificatesResolvers:
  letsencrypt:
    acme:
      email: admin@usrcaembassy.org
      storage: acme.json
      httpChallenge:
        entryPoint: web
```

Post-Deployment Checklist

- [ ] Database connection successful
- [ ] SendGrid email delivery working
- [ ] JWT authentication functional
- [ ] PDF generation working
- [ ] File uploads operational
- [ ] Chat widget responsive
- [ ] Application tracking functional
- [ ] Password recovery working
- [ ] Admin dashboard accessible
- [ ] SSL certificate valid

- [ ] Mobile responsiveness verified
- [ ] All API endpoints tested

Security Considerations

Authentication & Authorization

Password Security:

- bcrypt hashing with 10 salt rounds
- Minimum 6 character requirement
- Password confirmation on reset
- Secure token transmission (HTTPS only)

JWT Tokens:

- 256-bit secret key
- 24-hour expiration
- Payload includes: user ID, username, admin status
- HttpOnly cookies (recommended for production)

Admin Access Control:

- Single authorized email: admin@usrcaembassy.org
- Exact match required (case-insensitive)
- Protected routes with middleware
- Token must include `isAdmin: true`

Data Protection

Database Security:

- Parameterized queries (SQL injection prevention)
- Connection pooling with limits
- Encrypted connections (SSL/TLS)
- Regular backups
- User data anonymization on deletion

File Upload Security:

- File type validation
- Size limits enforced
- Virus scanning (recommended)
- Secure storage paths
- Access control

CORS Configuration:

```
const corsOptions = {
  origin: process.env.CORS_ORIGINS?.split(',') || ['http://localhost:3000'],
  credentials: true,
  optionsSuccessStatus: 200
};
```

API Security

Rate Limiting:

- Implement rate limiting on sensitive endpoints
- Prevent brute force attacks
- DDoS protection

Input Validation:

- Express-validator for all inputs
- Sanitization of user data
- Length restrictions
- Type checking

Error Handling:

- Generic error messages to users

- Detailed logging server-side
- No stack traces in production
- Secure error codes

Email Security

SendGrid Integration:

- API key stored in environment variables
- Domain authentication (SPF, DKIM)
- From address verification
- Link validation in emails

Password Reset:

- Time-limited tokens (1 hour)
- One-time use tokens
- Token deletion after use
- Secure link generation

Technical Stack

Frontend Technologies

Core:

- React 18.2.0
- React Router DOM 6.x
- JavaScript ES6+

Styling:

- CSS3 with custom properties
- Flexbox and Grid layouts
- Media queries for responsiveness
- CSS animations

State Management:

- React Hooks (useState, useEffect, useCallback)
- Local storage for persistence
- Context API for global state

Backend Technologies

Core:

- Node.js 18.x LTS
- Express.js 4.x
- MySQL2 (Promise-based)

Authentication:

- jsonwebtoken 9.x
- bcryptjs 2.x

Document Generation:

- PDFKit
- bwip-js (barcodes)

Email:

- @sendgrid/mail

Validation:

- express-validator

Development Tools

Version Control:

- Git
- Git LFS for large files
- GitHub for hosting

Code Quality:

- ESLint
- Prettier (recommended)

Testing:

- Jest (recommended)
- React Testing Library (recommended)

Build Tools:

- Create React App
- Webpack (via CRA)
- Babel (via CRA)

Maintenance & Support

Regular Maintenance Tasks

Daily:

- Monitor application logs
- Check email delivery status
- Review error reports

Weekly:

- Database backup verification
- Security update checks
- Performance monitoring
- User feedback review

Monthly:

- Dependency updates
- Security audit
- Database optimization
- Backup testing

Quarterly:

- Comprehensive security review
- User experience audit
- Performance optimization
- Feature planning

Monitoring Recommendations

Application Monitoring:

- Server uptime monitoring
- Response time tracking
- Error rate monitoring
- Database performance

User Analytics:

- Application submission rates
- Chat engagement metrics
- User registration trends
- Page view statistics

Backup Strategy

Database Backups:

- Daily automated backups

- 30-day retention period
- Encrypted backup storage
- Regular restore testing

File Backups:

- Uploaded documents backup
- PDF archive backup
- Configuration files backup

Support Contacts

Technical Support:

- Email: tech@usrcaembassy.org
- Response time: 24-48 hours

Administrative Support:

- Email: admin@usrcaembassy.org
- Phone: +1-XXX-XXX-XXXX

Appendix

A. Glossary

CAR: Central African Republic

JWT: JSON Web Token - A compact token format for secure information transmission

LFS: Large File Storage - Git extension for versioning large files

PDF: Portable Document Format

SSL/TLS: Secure Sockets Layer / Transport Layer Security

CRUD: Create, Read, Update, Delete operations

API: Application Programming Interface

CORS: Cross-Origin Resource Sharing

B. Common Issues & Solutions

Issue: Users cannot login
Solution: Verify database connection, check JWT secret, ensure password hash format

Issue: Emails not being sent
Solution: Verify SendGrid API key, check from address verification, review SendGrid logs

Issue: PDF generation fails
Solution: Check PDFKit installation, verify file permissions, review error logs

Issue: Chat widget not loading
Solution: Check CORS settings, verify API endpoints, inspect browser console

C. Change Log

Version 2.0 (January 2026):

- Added multilingual chat widget (English/French)
- Implemented application tracking system
- Added password recovery functionality
- Enhanced PDF generation with barcodes
- Restricted admin access to single email
- Improved mobile responsiveness
- Added Git LFS for video files

Version 1.0 (Previous):

- Initial release
- Basic application submission

- User authentication
- Admin dashboard
- PDF generation
- Chat system

D. Future Enhancements

Planned Features:

- Online payment integration
- Document verification system
- Appointment scheduling
- SMS notifications
- Multi-factor authentication
- Advanced analytics dashboard
- Mobile application (iOS/Android)
- Video consultation integration

Conclusion

The Central African Republic Embassy Website represents a comprehensive digital transformation of consular services. With robust features including multilingual support, real-time tracking, secure authentication, and mobile-optimized design, the platform provides efficient service delivery to CAR nationals across North America.

The system architecture ensures scalability, security, and maintainability while offering an intuitive user experience for both applicants and administrative staff.

For technical support or questions about this documentation, please contact the development team at tech@usrcaembassy.org.

Document Version: 1.0
Last Updated: January 5, 2026
Prepared By: Development Team
Review Status: Final