LUMEN ERP

Educational Resource Planning System

Documentation Date: February 20, 2026

Version: 1.0

Status: Production Ready

A comprehensive Flask-based ERP system designed for educational institutions to manage students, faculty, courses, attendance, fees, and institutional announcements.

TABLE OF CONTENTS

1. Project Overview
2. Technology Stack
3. System Architecture
4. Database Schema
5. User Roles & Permissions
6. Features & Modules
7. API Endpoints & Routes
8. Getting Started
9. Project Structure
10. Key Components

1. PROJECT OVERVIEW

Project Name: Lumen ERP

A complete Educational Resource Planning (ERP) system built with Flask, designed to streamline institutional operations. Lumen ERP provides comprehensive tools for managing student information, faculty profiles, attendance tracking, fee management, course allocation, leave management, and institutional communications through broadcast notifications.

Key Objectives:

- Centralize student and faculty information management
- Track attendance with detailed analytics and insights
- Manage course allocations and class assignments
- Handle fee collection and financial records
- Manage leave requests and approvals
- Institutional communication via broadcasts
- Role-based access control for security
- Real-time data updates and notifications

2. TECHNOLOGY STACK

Category	Technology	Version
Backend Framework	Flask	3.1.2
Python Version	Python	3.10+
ORM	SQLAlchemy	2.0.46
Database Integration	Flask-SQLAlchemy	3.1.1
Authentication	Flask-Login	Latest
Database	SQLite	3.x
Frontend	HTML5/CSS3/JavaScript	Vanilla JS
Templating	Jinja2	Latest
WSGI Server	Werkzeug	Included
Design Pattern	Neumorphic (Soft-UI)	Custom CSS

3. SYSTEM ARCHITECTURE

Multi-Tier Architecture:

Layer	Components	Responsibility
Presentation	HTML Templates CSS Styling JavaScript	User Interface & User Experience
Application	Flask Routes Blueprints Business Logic	Request Handling & Processing
Data Access	SQLAlchemy ORM Model Classes	Database Operations
Database	SQLite DB Tables & Relationships	Data Storage & Retrieval

4. DATABASE SCHEMA

Main Tables (13 Total):

User

Field	Type	Description
id	Integer	Primary Key
username	String	Unique username
email	String	User email address
password	String	Hashed password
role	String	Admin, HOD, Faculty, Student

StudentDetails

Field	Type	Description
id	Integer	Primary Key
user_id	Integer	Foreign Key to User
roll_number	String	Student roll number
department	String	Department/Branch
semester	Integer	Current semester

Broadcast

Field	Type	Description
id	Integer	Primary Key
title	String	Broadcast title
content	Text	Broadcast message
scope	String	institution or department
created_by_id	Integer	FK to User
is_pinned	Boolean	Pin status

5. USER ROLES & PERMISSIONS

Role	Description	Key Permissions
Admin	System Administrator	Full access, broadcast institution-wide, manage all users
HOD	Head of Department	Department management, dept broadcasts, class allocation
Faculty	Teaching Staff	View roster, mark attendance, create notes, view grades
Student	Regular Student	View attendance, fees, notes, leave requests, broadcasts

6. FEATURES & MODULES

• Authentication & Authorization

■ Secure login with password hashing

■ Role-based access control (RBAC)

■ Session management with Flask-Login

• Student Management

■ Student profile creation and updates

■ Roll number and department tracking

■ Semester progression management

• Attendance Tracking

■ Daily attendance marking

■ Subject-wise attendance breakdown

■ Attendance analytics with 4 periods (daily, weekly, monthly, semester)

■ Individual percentage calculations

■ Visual progress indicators

• Fee Management

■ Fee structure creation

■ Payment tracking

■ Fee reminders and notifications

• Leave Management

■ Leave request submission

■ Approval workflow

■ Leave balance tracking

■ History and reports

• Class & Course Allocation

■ Course to class allocation

■ Faculty assignment to classes

■ Semester-based allocation

■ Request-based allocation system

• Broadcast Messaging

■ Institution-wide announcements (Admin)

■ Department-specific broadcasts (HOD)

■ Auto-refresh every 60 seconds

■ Pin important messages

■ Delete and edit functionality

• Event Management

■ Event creation and scheduling

■ Event visibility settings

■ Notification system

• Certificate Management

■ Certificate generation

■ Digital certificate storage

■ Verification system

• Responsive Design

■ Mobile-friendly interface

■ Tablet optimization

■ Desktop experience

■ Touch-friendly buttons (44px minimum)

7. API ENDPOINTS & ROUTES

Authentication Routes (/auth:

• GET /login - Display login form

• POST /login - Process login credentials

• GET /register - Display registration form

• POST /register - Create new user account

• GET /logout - Logout user session

Main Routes (/

• GET /dashboard - User dashboard

• GET /attendance - View attendance records

• GET /attendance/analysis - Attendance analytics

• GET /broadcasts - View broadcasts

• GET /api/broadcasts/refresh - Auto-refresh JSON (60s interval)

Admin Routes (/admin)

• GET /admin - Admin panel main page

• GET /admin/broadcasts - View broadcasts

• POST /admin/broadcasts - Create broadcast

• POST /admin/broadcasts//pin - Toggle pin status

• POST /admin/broadcasts//delete - Delete broadcast

HOD Routes (/hod)

• GET /hod - HOD panel main page

• GET /hod/broadcasts - View dept broadcasts

• POST /hod/broadcasts - Create dept broadcast

• POST /hod/broadcasts//pin - Toggle pin status

• POST /hod/broadcasts//delete - Delete broadcast

8. GETTING STARTED

Prerequisites:

• Python 3.10 or higher

• pip package manager

• Git (optional)

Installation Steps:

1. Clone the repository from GitHub

2. Create a virtual environment: python -m venv venv

3. Activate the virtual environment: venv\Scripts\activate (Windows)

4. Install dependencies: pip install -r config/requirements.txt

5. Initialize database: python scripts/init_db.py

6. Run the application: python app.py

Access the Application:

• URL: http://127.0.0.1:5000

• Default Admin: admin / password (from database)

• Create additional users through admin panel

9. PROJECT STRUCTURE

ERP_system/ ■ ■■■ **config/** # Configuration directory ■ ■■■ config.py # Flask configuration ■ ■■■ extensions.py # SQLAlchemy & extensions ■ ■■■ requirements.txt # Python dependencies ■ ■■■ .gitignore # Git ignore file ■ ■■■ **docs/** # Documentation & assets ■ ■■■ dashboard.png # Dashboard screenshot ■ ■■■ login.png # Login screenshot ■ ■■■ PROJECT_DOCUMENTATION.pdf # This document ■ ■■■ **instance/** # Instance-specific files ■ ■■■ college.db # SQLite database ■ ■■■ **routes/** # Application routes/blueprints ■ ■■■ __init__.py # Blueprint registration ■ ■■■ auth.py # Authentication routes ■ ■■■ main.py # Main/core routes ■ ■■■ admin.py # Admin-specific routes ■ ■■■ hod.py # HOD-specific routes ■ ■■■ **scripts/** # Utility scripts ■ ■■■ init_db.py # Database initialization ■ ■■■ check_db.py # Database verification ■ ■■■ verify_features.py # Feature testing ■ ■■■ reset_db_except_admin.py # Reset non-admin data ■ ■■■ **static/** # Static assets ■ ■■■ css/ ■ ■ ■■■ style.css # Main stylesheet ■ ■■■ js/ ■ ■ ■■■ background3d.js # 3D background animation ■ ■■■ uploads/ # User uploads ■ ■■■ **templates/** # HTML templates (Jinja2) ■ ■■■ base.html # Base template with navigation ■ ■■■ login.html # Login page ■ ■■■ dashboard.html # User dashboard ■ ■■■ admin_panel.html # Admin dashboard ■ ■■■ hod_panel.html # HOD dashboard ■ ■■■ attendance.html # Attendance page ■ ■■■ attendance_analysis.html # Analytics page ■ ■■■ broadcasts.html # Broadcast feed ■ ■■■ admin_broadcasts.html # Admin broadcast creator ■ ■■■ hod_broadcasts.html # HOD broadcast creator ■ ■■■ manage_users.html # User management ■ ■■■ edit_user.html # User editor ■ ■■■ fees.html # Fee management ■ ■■■ leaves.html # Leave requests ■ ■■■ notes.html # Course notes ■ ■■■ certificates.html # Certificates ■ ■■■ allot_classes.html # Class allocation ■ ■■■ time_slots.html # Time slot management ■ ■■■ view_attendance.html # Attendance view ■ ■■■ calendar.html # Event calendar ■ ■■■ **app.py** # Flask application entry point ■■■ **models.py** # SQLAlchemy ORM models (13+) ■■■ **utils.py** # Utility functions ■ ■■■ **README.md** # Project README

10. KEY COMPONENTS

app.py

Flask application factory. Initializes the app, registers blueprints, sets up the database, and configures login management.

models.py

SQLAlchemy ORM models defining 13+ database tables including User, Student, Faculty, Attendance, Broadcast, Fee, Leave, Event, Certificate, TimeSlot, ClassAllotment, and related models.

routes/main.py

Core application routes for dashboard, attendance tracking, attendance analysis, and broadcast reading with 60-second auto-refresh API.

routes/admin.py

Admin-specific routes for admin panel, institution-wide broadcast management, and system administration features.

routes/hod.py

Head of Department routes for department management, department-specific broadcasts, and HOD panel functionality.

routes/auth.py

Authentication routes for login, registration, and logout with secure password hashing and session management.

config/config.py

Flask configuration settings including database URI, secret keys, and environment variables.

config/extensions.py

SQLAlchemy database instance, Flask-Login manager, and Flask-Migrate configuration.

static/css/style.css

Neumorphic (soft-UI) design stylesheet with responsive breakpoints (991px tablets, 768px mobile, 480px small phones). Includes hamburger menu, animations, and touch-friendly buttons.

static/js/background3d.js

3D background animation using Three.js or Canvas API for enhanced visual appeal.

RESPONSIVE DESIGN DETAILS

Breakpoints:

Device Type	Max Width	Features
Desktop	1200px+	Full navigation, all features visible, multi-column layouts
Tablet	991px-1199px	Collapsible hamburger menu, single columns, optimized spacing
Mobile	480px-990px	Full hamburger menu, touch-friendly buttons (44px), reduced padding
Small Phone	<480px	Ultra-compact layouts, readable text, minimal padding

Mobile Optimization Features:

- Hamburger navigation menu that slides open/closed with animation
- Touch-friendly buttons with minimum height of 44 pixels
- Responsive typography that scales from 3rem (desktop) to 1.25rem (mobile)
- 16px minimum font size on inputs to prevent iOS auto-zoom
- Flexible grid layouts that adapt from 4-column to single column
- Word-break support for long content on narrow screens
- Reduced padding and margins on mobile devices
- Full-width buttons and form fields on small screens
- Optimized card layouts for mobile viewing

AUTO-REFRESH & REAL-TIME UPDATES

The broadcast system implements a 60-second auto-refresh mechanism to provide near real-time updates without requiring manual page refresh.

Technical Implementation:

- **API Endpoint:** GET /api/broadcasts/refresh returns JSON with current broadcasts
- **Interval:** setInterval() runs every 60 seconds
- **Update Method:** Fetch API calls endpoint and updates DOM with new broadcasts
- **Animation:** New broadcasts slide into view with smooth transitions
- **Security:** XSS protection with escape() function for broadcast content
- **Performance:** Efficient JSON response with only necessary data

SECURITY FEATURES

- **Password Hashing:** All passwords are hashed using werkzeug.security before storage
- **Session Management:** Flask-Login manages user sessions with secure cookies
- **CSRF Protection:** Forms include CSRF tokens to prevent cross-site attacks
- **Role-Based Access Control:** Routes check user role before granting access
- **SQL Injection Prevention:** SQLAlchemy ORM prevents SQL injection attacks
- **XSS Protection:** User content is escaped to prevent script injection
- **Input Validation:** All user inputs are validated before processing

DEVELOPMENT & MAINTENANCE

Database Management Scripts:

init_db.py: Initializes the SQLite database with all tables. Run once during setup.

check_db.py: Verifies database integrity and lists all tables with column information.

verify_features.py: Tests all major features and functionality for compatibility.

reset_db_except_admin.py: Clears all data except admin user for testing/demo purposes.

Color Palette:

Usage	Hex Color	RGB	Purpose
Primary	#2a4d69	(42, 77, 105)	Headers, navigation, primary buttons
Accent	#63ace5	(99, 172, 229)	Links, highlights, secondary elements
Text Secondary	#4b86b4	(75, 134, 180)	Subheadings, secondary text
Success	#2ecc71	(46, 204, 113)	Success messages, positive actions
Error	#e74c3c	(231, 76, 60)	Error messages, destructive actions

FUTURE ENHANCEMENTS

- Dark mode toggle for accessibility
- Mobile app (React Native) for iOS and Android
- Advanced analytics dashboard with charts and graphs
- Email notifications for broadcasts and important updates
- Grade management system with GPA calculation
- Online exam and assessment system
- Parent portal for student progress tracking
- SMS notifications for critical updates
- API documentation (Swagger/OpenAPI)
- Multi-language support (i18n)
- Single Sign-On (SSO) integration
- Cloud deployment on AWS/Azure/GCP

END OF DOCUMENTATION