University Enrollment System

Full Stack Project Operational Workflow

Date: January 16, 2026

System Type: Flask (Python) Full Stack Web Application

1. System Overview

This application is a comprehensive university management portal designed to streamline the student enrollment process. It features a secure Role-Based Access Control (RBAC) system distinguishing between Administrators and Students.

Core Capabilities:

- **Secure Authentication:** Hashed passwords, Session management, Login decorators.
- **Admin Dashboard:** Course CRUD, Student Management, Analytics.
- **Student Portal:** Profile management, Course Browsing, Self-Enrollment.
- **Payments:** Real-time UPI integration with QR generation.
- **Notifications:** Automated Email confirmations with PDF attachments.

2. User Workflows

A. Authentication Module

entry point for all users.

- **Registration:** New users sign up as 'Student'. System auto-generates an Enrollment Number (e.g., UNIV2026001).
- **Login:** Users input Email/Password. System authenticates hash and checks Role.
- **Routing:** Admins redirect to `/admin/dashboard`. Students redirect to `/student/dashboard`.

B. Admin Workflow

Authorized personnel manage the academic data.

- **Dashboard View:** See quick stats (Total Students, Courses, Recent Enrollments).
- **Manage Courses:** Create new courses (Set Name, Code, Fee, Seats). Edit existing details.
- **Manage Students:** View all registered students. Delete unauthorized accounts.
- **System Oversight:** Monitor database integrity and enrollment logs.

C. Student Journey (End-to-End)

The primary lifecycle of a student user.

Phase	Action	System Process
1. Profile	Update Info	Uploads Profile Pic, sets Phone/Address.
2. Discovery	View Courses	Browse available courses. Filter by name.
3. Action	Click 'Enroll'	Checks Seat Availability. Creates 'Pending' Enrollment.
4. Payment	Scan QR Code	Generates UPI Link. User pays via PhonePe/GPay.
5. Confirm	Click 'Paid'	Updates Status -> 'Enrolled'. Decrements Seats.
6. Receipt	Receive Email	System mails 'Enrollment Confirmed' + Rules.pdf.

Key Technical Highlight: Payment Integration

The payment system uses a hybrid approach for maximum reliability:

1. **Dynamic Generation:** Calculates Fee from DB.
2. **Deep Linking:** `upi://pay?...` allows one-tap mobile payments.
3. **Dual Rendering:** Uses local `qrcode` library, falls back to `qrserver.com` API if local fails.

3. System Architecture

A. Database Schema (ERD Description)

Model	Fields	Relationships
User	id, name, email, password_hash, role	One-to-One with StudentDetails
Course	id, name, code, fee, seats, credits	One-to-Many with Enrollments
Enrollment	id, student_id, course_id, status	Link Table (Many-to-Many logic)

B. Application Structure

```
Pro_MCA/
■■■ app.py              # Application Factory & Entry Point
■■■ models.py           # Database Models (SQLAlchemy)
■■■ config.py           # Configuration (Secret Keys, DB URI)
■■■ utils.py            # Helpers (PDF, Email, QR Code)
■■■ routes/             # Blueprints (Modular Routing)
■   ■■■ auth_routes.py    # Login/Register
■   ■■■ admin_routes.py   # Admin Ops
■   ■■■ student_routes.py # Student Ops
■■■ templates/          # HTML Frontend (Jinja2)
    ■■■ auth/           # Login Screens
    ■■■ admin/          # Dashboard & Forms
    ■■■ student/        # Portal & Payment
```

Generated by AI Assistant.