Configuration Settings (config.py)

1. Executive Summary

Centralizes all application configuration variables to ensure security and maintainability. Separates code from configuration data.

2. Code Logic & Functionality

1. **Class-Based Config**: Uses a `Config` class to namespace settings.
2. **Environment Variables**: Fetches sensitive keys (SECRET_KEY, DB_URI) from the OS environment for security.
3. **Mail Settings**: Defines SMTP server details for email notifications.

3. Key Concepts & Definitions

- **Environment Variable**: Dynamic named values that can affect the way running processes will behave on a computer.
- **Secret Key**: A cryptographic key used to sign session cookies.

4. Location Details

Path: `config.py` **Type**: .PY File

5. Source Code Preview (Snippet)

Running typical software analysis on this file:

```
"""
Configuration settings for School Activity Booking System
Production-ready with environment variable support
"""
import os
from datetime import timedelta

class Config:
    """Base configuration"""
    # Security
    SECRET_KEY = os.environ.get('SECRET_KEY') or 'dev-secret-key-change

    # Database
    SQLALCHEMY_DATABASE_URI = os.environ.get('DATABASE_URL') or 'sqlite
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    SQLALCHEMY_ECHO = False  # Set to True for SQL debugging

    # Email configuration
    MAIL_SERVER = os.environ.get('MAIL_SERVER', 'smtp.gmail.com')
    MAIL_PORT = int(os.environ.get('MAIL_PORT', 587))
```

```
    MAIL_USE_TLS = os.environ.get('MAIL_USE_TLS', 'true').lower() == 't
    MAIL_USERNAME = os.environ.get('MAIL_USERNAME', 'greenwoodinternati
    MAIL_PASSWORD = os.environ.get('MAIL_PASSWORD', 'muesmgjpulyscdmv')
    MAIL_DEFAULT_SENDER = os.environ.get('MAIL_DEFAULT_SENDER', 'greenw

    # Payment Configuration (Stripe)
    STRIPE_PUBLIC_KEY = os.environ.get('STRIPE_PUBLIC_KEY', 'pk_test_51
    STRIPE_SECRET_KEY = os.environ.get('STRIPE_SECRET_KEY', 'sk_test_51

    # Session Configuration
    SESSION_COOKIE_SECURE = False  # Set to True in production with HTT
    SESSION_COOKIE_HTTPONLY = True
    SESSION_COOKIE_SAMESITE = 'Lax'
    PERMANENT_SESSION_LIFETIME = timedelta(minutes=30)  # 30-minute ses
    SESSION_REFRESH_EACH_REQUEST = True

    # Application Settings
    MAX_CONTENT_LENGTH = 16 * 1024 * 1024  # 16MB max file upload
    UPLOAD_FOLDER = os.path.join(os.path.dirname(os.path.abspath(__file

    # Pagination
    ITEMS_PER_PAGE = 20

class DevelopmentConfig(Config):
    """Development configuration"""
    DEBUG = True
    TESTING = False
    SQLALCHEMY_ECHO = False  # Enable to see SQL queries

class ProductionConfig(Config):
    """Production configuration - SECURE DEFAULTS"""
    DEBUG = False
    TESTING = False
    SESSION_COOKIE_SECURE = True  # Require HTTPS

    # Override with environment variables (REQUIRED in production)
    @classmethod
    def init_app(cls, app):
        # Ensure critical settings are from environment in production
        if not os.environ.get('SECRET_KEY'):
            raise ValueError("SECRET_KEY environment variable must be s
        if not os.environ.get('MAIL_PASSWORD'):
            print("WARNING: MAIL_PASSWORD not set, emails may fail!")

class TestingConfig(Config):
    """Testing configuration"""
    TESTING = True
    SQLALCHEMY_DATABASE_URI = 'sqlite:///:memory:'
    WTF_CSRF_ENABLED = False

config = {
    'development': DevelopmentConfig,
    'production': ProductionConfig,
```

```
    'testing': TestingConfig,
    'default': DevelopmentConfig
}
```