Application Entry Point (app.py)

1. Executive Summary

This is the core controller of the Flask application. It initializes the application context, configures database extensions, and defines the primary route handlers for the web interface.

2. Code Logic & Functionality

1. **Initialization**: Creates the Flask app instance and loads configuration from `config.py`.
2. **Database Setup**: Initializes SQLAlchemy and defines the database models (Parent, Child, Booking).
3. **Route Definition**: Maps URL endpoints (like `/login`, `/dashboard`) to Python functions.
4. **Request Handling**: Processes incoming GET/POST requests, handles form validation, and renders HTML templates.

3. Key Concepts & Definitions

- **Flask**: A micro web framework for Python.
- **Route**: A URL pattern mapped to a specific function.
- **Decorator**: The `@app.route` syntax used to modify functions.
- **Context**: The state of the application during a request.

4. Location Details

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

5. Source Code Preview (Snippet)

Running typical software analysis on this file:

```
"""
School Activity Booking System - Flask Application
Main application entry point
"""
from flask import Flask, render_template, request, redirect, url_for, s
# Trigger reload
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_ha
from datetime import datetime, timedelta
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
import os
from itsdangerous import URLSafeTimedSerializer, BadSignature, Signatur
```

```
from reportlab.lib.pagesizes import letter
from reportlab.lib import colors
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Pa
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib.units import inch
from io import BytesIO
from flask import send_file
from flask_mail import Mail, Message
from flask_wtf.csrf import CSRFProtect
from functools import wraps
from config import config

# Enhanced PDF Invoice Generator
from enhanced_invoice import get_enhanced_invoice_pdf


# Initialize extensions
db = SQLAlchemy()
mail = Mail()
csrf = CSRFProtect()

def create_app(config_name='default'):
    app = Flask(__name__)
    app.config.from_object(config[config_name])

    db.init_app(app)
    mail.init_app(app)
    csrf.init_app(app)

    # Register context processors
    @app.context_processor
    def inject_now():
        return {'now': datetime.utcnow()}



    return app

app = create_app()

# ==================== Database Models ====================

class Parent(db.Model):
    """Parent/Guardian model"""
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)
    full_name = db.Column(db.String(120), nullable=False)
    phone = db.Column(db.String(20))
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    bookings = db.relationship('Booking', backref='parent', lazy=True,
    children = db.relationship('Child', backref='parent', lazy=True, ca
```

```
    waitlists = db.relationship('Waitlist', backref='parent', lazy=True

    def set_password(self, password):
        self.password = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password, password)

class Admin(db.Model):
    """Admin model"""
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    def set_password(self, password):
        self.password = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password, password)

class Tutor(db.Model):
    """Tutor model"""
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)
    full_name = db.Column(db.String(120), nullable=False)
    specialization = db.Column(db.String(200))
    qualification = db.Column(db.String(300))
    bio = db.Column(db.Text)
    linkedin_url = db.Column(db.String(200))
    teaching_philosophy = db.Column(db.Text)
    years_experience = db.Column(db.Integer)
    education = db.Column(db.Text)
    certifications = db.Column(db.Text)
    status = db.Column(db.String(20), default='approved')  # pending, a
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    def set_password(self, password):
        self.password = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password, password)


class Child(db.Model):
    """Child model"""
    id = db.Column(db.Integer, primary_key=True)
    parent_id = db.Column(db.Integer, db.ForeignKey('parent.id'), nulla
    name = db.Column(db.String(120), nullable=False)
    age = db.Column(db.Integer)
    grade = db.Column(db.String(20))
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
```

```
    bookings = db.relationship('Booking', backref='child', lazy=True, c
    attendance_records = db.relationship('Attendance', backref='child',

class Activity(db.Model):
    """Available activities model"""
    id = db.Column(db.Integer, primary_key=True)
    tutor_id = db.Column(db.Integer, db.ForeignKey('tutor.id'), nullabl
    name = db.Column(db.String(120), nullable=False)
    description = db.Column(db.Text)
    price = db.Column(db.Float, nullable=False)
    max_capacity = db.Column(db.Integer, default=20)
    day_of_week = db.Column(db.String(20), nullable=False)
    start_time = db.Column(db.String(10), nullable=False)
    end_time = db.Column(db.String(10), nullable=False)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    # Relationships
    tutor = db.relationship('Tutor', backref='activities', lazy=True)
    bookings = db.relationship('Booking', backref='activity', lazy=True
    waitlists = db.relationship('Waitlist', backref='activity', lazy=Tr

class Booking(db.Model):
    """Booking model"""
    id = db.Column(db.Integer, primary_key=True)
    parent_id = db.Column(db.Integer, db.ForeignKey('parent.id'), nulla
    child_id = db.Column(db.Integer, db.ForeignKey('child.id'), nullabl
    activity_id = db.Column(db.Integer, db.ForeignKey('activity.id'), n
    booking_date = db.Column(db.Date, nullable=False)

... [Code Truncated for Documentation Readability - See Source File for
```