AI Code Intelligence Report

Generated On: 2026-02-05 14:34:38.726453

Prediction: Average
Score: 65/100
Grade: B

Extracted Metrics:

filename: tmpn1dy22ap.py
language: Python
lines_of_code: 178
num_functions: 17
num_comments: 11
comment_ratio: 0.05789473684210526
avg_line_length: 22.861313868613138
max_line_length: 84
indentation_consistency: 0.1111111111111111
nesting_depth: 0
cyclomatic_complexity: 1.6470588235294117
num_imports: 7
num_loops: 0
num_conditionals: 10
num_exceptions: 0
has_docstring: 0
avg_tokens_per_line: 3.752808988764045
keyword_density: 0.48314606741573035
blank_lines_ratio: 0.3102189781021898
avg_identifier_quality: 0.8823529411764706

AI Suggestions:

■ Add docstrings for better documentation.

Submitted Code:

```
from flask import Flask, request, render_template, redirect, url_for, session, flash
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash
from functools import wraps
```

```
import random
import datetime

app = Flask(__name__)

# ---------------- CONFIG ----------------
import os
basedir = os.path.abspath(os.path.dirname(__file__))

app.config['SQLALCHEMY_DATABASE_URI'] = \
'sqlite:///' + os.path.join(basedir, 'plant_nursery.db')

app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.secret_key = "plantnursery_secret_key"

db = SQLAlchemy(app)

# ---------------- MODELS ----------------

class User(db.Model):
id = db.Column(db.Integer, primary_key=True)
username = db.Column(db.String(255), unique=True, nullable=False)
password = db.Column(db.String(255), nullable=False)


class Order(db.Model):
id = db.Column(db.Integer, primary_key=True)
username = db.Column(db.String(255))
transaction_id = db.Column(db.String(255))
amount = db.Column(db.Float)
date = db.Column(db.String(255))


class Booking(db.Model):
id = db.Column(db.Integer, primary_key=True)
username = db.Column(db.String(255))
occasion = db.Column(db.String(255))
booking_date = db.Column(db.String(50))
delivery_address = db.Column(db.String(255))
message = db.Column(db.Text)
amount = db.Column(db.Float)
payment_status = db.Column(db.String(50), default="Pending")


OCCASION_PACKAGES = {
"Birthday Special": 50,
"Wedding Special": 200,
"Valentine Combo": 99,
"Graduation Combo": 45,
"Housewarming Kit": 120
}
```

```
# ---------- LOGIN REQUIRED ----------

def login_required(f):
@wraps(f)
def decorated(*args, **kwargs):
if "user" not in session:
return redirect(url_for("login_page"))
return f(*args, **kwargs)
return decorated


# ---------- PAGE ROUTES ----------

@app.route('/')
def login_page():
return render_template("login.html")


@app.route('/register')
def register_page():
return render_template("register.html")


@app.route('/home')
@login_required
def home():
return render_template("first_page.html")


@app.route('/shop')
@login_required
def shop():
return render_template("shop.html")


@app.route('/about')
@login_required
def about():
return render_template("About.html")


@app.route('/contact')
@login_required
def contact():
return render_template("contact.html")


@app.route('/payment')
@login_required
def payment():
amount = request.args.get("amount", 0)
return render_template("payment.html", amount=amount)
```

```
@app.route('/prebooking')
@login_required
def prebooking():
return render_template("prebooking.html", packages=OCCASION_PACKAGES)


@app.route('/mybookings')
@login_required
def mybookings():
bookings = Booking.query.filter_by(
username=session["user"]
).all()
return render_template("mybookings.html", bookings=bookings)


@app.route('/logout')
def logout():
session.clear()
return redirect(url_for("login_page"))


# ---------- AUTH ----------

@app.route('/register', methods=['POST'])
def register_user():
username = request.form.get('username')
password = request.form.get('password')
confirm = request.form.get('confirm_password')

if not username or not password:
flash("Please fill all fields!")
return redirect(url_for("register_page"))

if password != confirm:
flash("Passwords do not match!")
return redirect(url_for("register_page"))

if User.query.filter_by(username=username).first():
flash("User already exists!")
return redirect(url_for("register_page"))

hashed = generate_password_hash(password)
new_user = User(username=username, password=hashed)
db.session.add(new_user)
db.session.commit()

flash("Registration successful! Please login.")
return redirect(url_for("login_page"))
```

```
@app.route('/login', methods=['POST'])
def login_user():
username = request.form.get('username')
password = request.form.get('password')

user = User.query.filter_by(username=username).first()

if user and check_password_hash(user.password, password):
session["user"] = username
return redirect(url_for("home"))

flash("Invalid Username or Password!")
return redirect(url_for("login_page"))



# ---------- PAYMENT ----------

@app.route('/process_payment', methods=['POST'])
@login_required
def process_payment():

amount = request.form.get("amount")

if not amount:
return redirect(url_for("shop"))

txn = "TXN" + str(random.randint(100000, 999999))
date = datetime.datetime.now().strftime("%d-%m-%Y %H:%M:%S")

# Save order
new_order = Order(
username=session["user"],
transaction_id=txn,
amount=float(amount),
date=date
)

db.session.add(new_order)

# ■ VERY IMPORTANT PART
# Update latest booking to Paid
latest_booking = Booking.query.filter_by(
username=session["user"],
payment_status="Pending"
).order_by(Booking.id.desc()).first()

if latest_booking:
latest_booking.payment_status = "Paid"
```

```
db.session.commit()

return redirect(url_for("receipt", txn=txn))


# ---------- BOOKING ----------

@app.route('/book', methods=['POST'])
@login_required
def book():
occasion = request.form.get("occasion")
booking_date = request.form.get("booking_date")
address = request.form.get("delivery_address")
message = request.form.get("message")

amount = OCCASION_PACKAGES.get(occasion)

new_booking = Booking(
username=session["user"],
occasion=occasion,
booking_date=booking_date,
delivery_address=address,
message=message,
amount=amount
)

db.session.add(new_booking)
db.session.commit()

return redirect(url_for("payment", amount=amount))

@app.route('/receipt')
@login_required
def receipt():

txn = request.args.get("txn")

if not txn:
return "No Payment Record Found"

order = Order.query.filter_by(
transaction_id=txn,
username=session["user"]
).first()

if not order:
return "No Payment Record Found"

return render_template("receipt.html", order=order)
```

```
# ---------- MAIN ----------

if __name__ == '__main__':
with app.app_context():
db.create_all()

app.run(debug=True)
```