Assignment No. 3

Title: Design GUI in Python (Mini Project)

Problem Statement:

Current service marketplace platforms often present a complex and inconsistent Graphical User Interface (GUI) that creates significant friction during the multi-step booking process, particularly on mobile devices.

Learning Objectives:

• To design a user interface in Python

• To learn simplicity, user centric approach of a GUI in designing

Learning Outcomes:

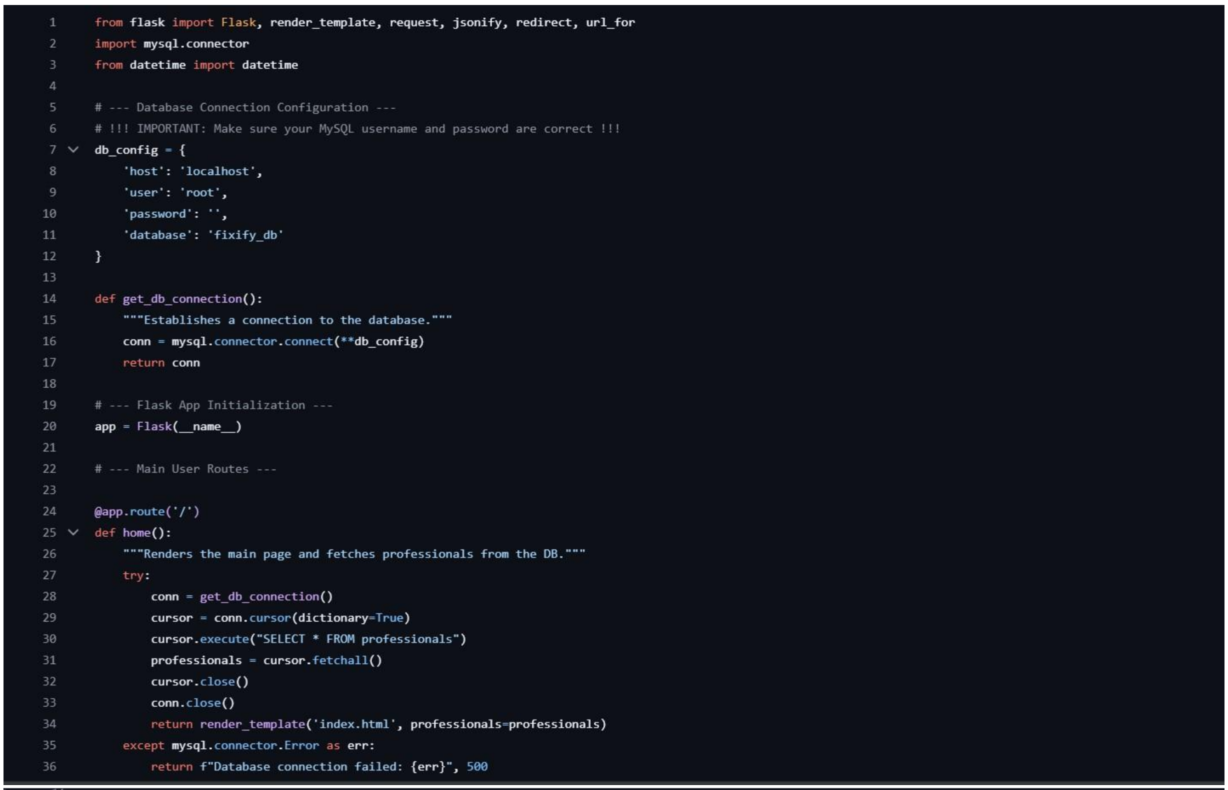
1. Students were able to apply HCI principles and Shneiderman's 8 Golden Rules in interface design.
2. Gained an understanding of how usability, consistency, and feedback enhance user experience.
3. Developed the ability to identify and correct usability issues through HCI guidelines.
4. Learned the importance of user-centered design for creating intuitive and efficient interfaces.
5. Strengthened skills in analyzing and evaluating interfaces using HCI concepts.
6. Experienced how structured design rules can lead to better user satisfaction and system effectiveness.

Requirements:

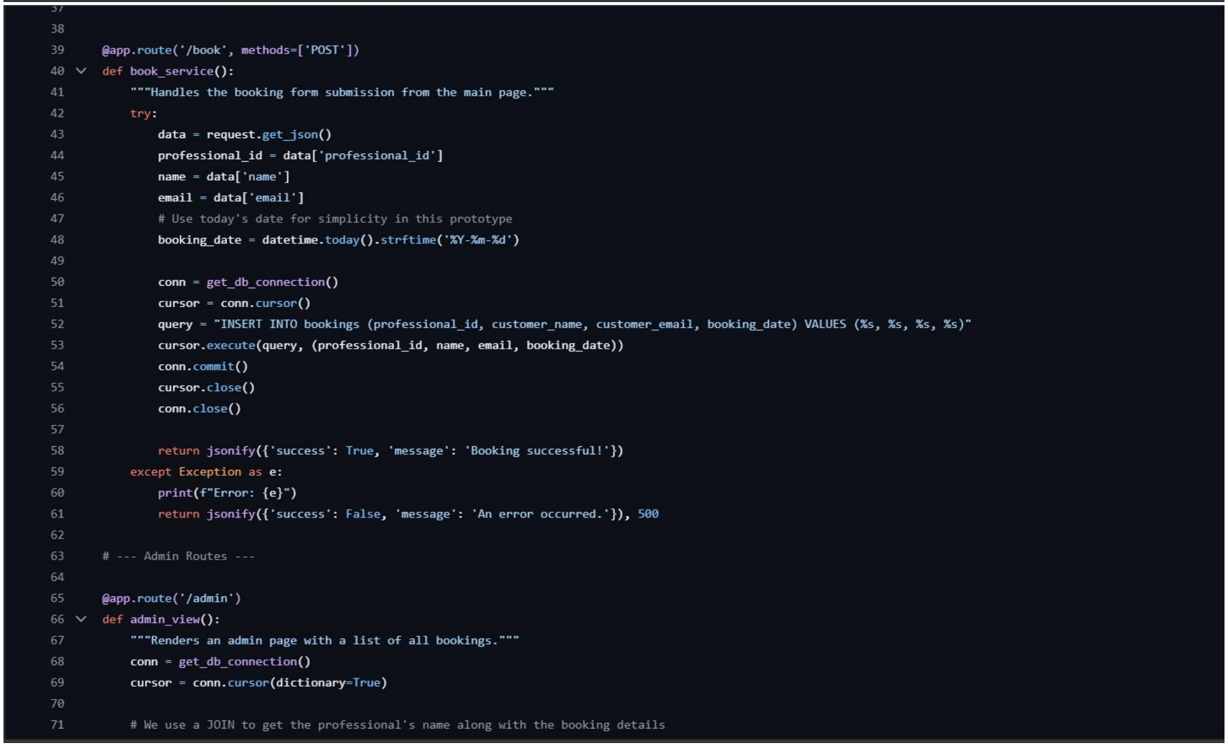
Hardware: Intel Core i3, RAM- Minimum 4 GB, 200 MB free space.

Software: Windows/Linux , Python 3.8, VS Code / PyCharm, Tkinter.

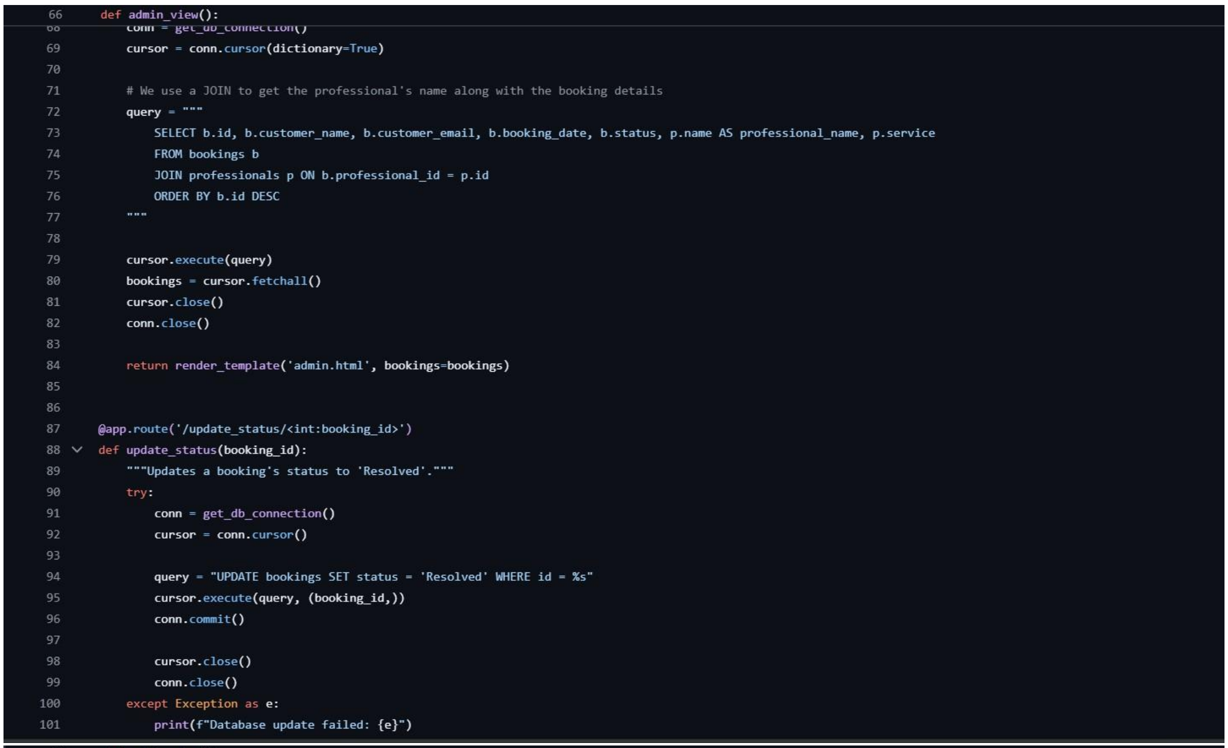
Implementation Steps:



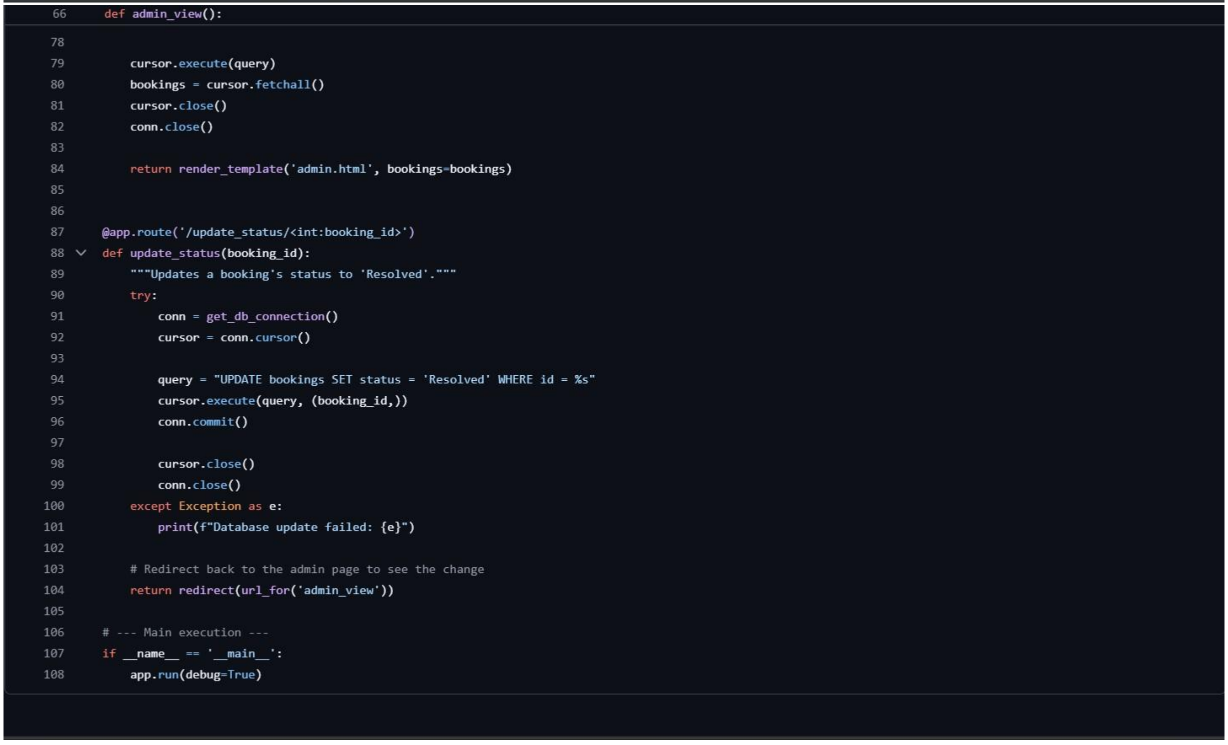
```python
from flask import Flask, render_template, request, jsonify, redirect, url_for
import mysql.connector
from datetime import datetime

# --- Database Connection Configuration ---
# !!! IMPORTANT: Make sure your MySQL username and password are correct !!!
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '',
    'database': 'fixify_db'
}

def get_db_connection():
    """Establishes a connection to the database."""
    conn = mysql.connector.connect(**db_config)
    return conn

# --- Flask App Initialization ---
app = Flask(__name__)

# --- Main User Routes ---

@app.route('/')
def home():
    """Renders the main page and fetches professionals from the DB."""
    try:
        conn = get_db_connection()
        cursor = conn.cursor(dictionary=True)
        cursor.execute("SELECT * FROM professionals")
        professionals = cursor.fetchall()
        cursor.close()
        conn.close()
        return render_template('index.html', professionals=professionals)
    except mysql.connector.Error as err:
        return f"Database connection failed: {err}", 500
```



```python


@app.route('/book', methods=['POST'])
def book_service():
    """Handles the booking form submission from the main page."""
    try:
        data = request.get_json()
        professional_id = data['professional_id']
        name = data['name']
        email = data['email']
        # Use today's date for simplicity in this prototype
        booking_date = datetime.today().strftime('%Y-%m-%d')

        conn = get_db_connection()
        cursor = conn.cursor()
        query = "INSERT INTO bookings (professional_id, customer_name, customer_email, booking_date) VALUES (%s, %s, %s, %s)"
        cursor.execute(query, (professional_id, name, email, booking_date))
        conn.commit()
        cursor.close()
        conn.close()

        return jsonify({'success': True, 'message': 'Booking successful!'})
    except Exception as e:
        print(f"Error: {e}")
        return jsonify({'success': False, 'message': 'An error occurred.'}), 500

# --- Admin Routes ---

@app.route('/admin')
def admin_view():
    """Renders an admin page with a list of all bookings."""
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # We use a JOIN to get the professional's name along with the booking details
```



```python
def admin_view():
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # We use a JOIN to get the professional's name along with the booking details
    query = """
        SELECT b.id, b.customer_name, b.customer_email, b.booking_date, b.status, p.name AS professional_name, p.service
        FROM bookings b
        JOIN professionals p ON b.professional_id = p.id
        ORDER BY b.id DESC
    """

    cursor.execute(query)
    bookings = cursor.fetchall()
    cursor.close()
    conn.close()

    return render_template('admin.html', bookings=bookings)


@app.route('/update_status/<int:booking_id>')
def update_status(booking_id):
    """Updates a booking's status to 'Resolved'."""
    try:
        conn = get_db_connection()
        cursor = conn.cursor()

        query = "UPDATE bookings SET status = 'Resolved' WHERE id = %s"
        cursor.execute(query, (booking_id,))
        conn.commit()

        cursor.close()
        conn.close()
    except Exception as e:
        print(f"Database update failed: {e}")
```



```python
def admin_view():

    cursor.execute(query)
    bookings = cursor.fetchall()
    cursor.close()
    conn.close()

    return render_template('admin.html', bookings=bookings)


@app.route('/update_status/<int:booking_id>')
def update_status(booking_id):
    """Updates a booking's status to 'Resolved'."""
    try:
        conn = get_db_connection()
        cursor = conn.cursor()

        query = "UPDATE bookings SET status = 'Resolved' WHERE id = %s"
        cursor.execute(query, (booking_id,))
        conn.commit()

        cursor.close()
        conn.close()
    except Exception as e:
        print(f"Database update failed: {e}")
```

```python

    # Redirect back to the admin page to see the change
    return redirect(url_for('admin_view'))

# --- Main execution ---
if __name__ == '__main__':
    app.run(debug=True)
```

Conclusion: Through this mini project, students were able to apply HCI concepts and Shneiderman’s 8 Golden Rules, which improved usability, efficiency, and user satisfaction, while highlighting the importance of designing intuitive, user-friendly, and effective interfaces.