Temple Dharshan Booking App

A PROJECT REPORT

Submitted by

SURYA S (2303811724321115)

in partial fulfillment of requirements for the award of the course

CGB1221-DATABASE MANAGEMENT SYSTEMS

in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE-2025

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"Temple Dharshan Booking App"** is the bonafide work of **SURYA S (2303811724321115)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE

Dr.T.AVUDAIAPPAN, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

ASSOCIATE PROFESSOR

Department of Artificial Intelligence

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

SIGNATURE

Mrs.S. GEETHA, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of Artificial Intelligence

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on ...04.06.2025.....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

I declare that the project report on **"Temple Dharshan Booking App"** is the result of original work done by me and best of my knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1221 – DATABASE MANAGEMENT SYSTEMS.**

.

Signature

SURYA S

Place: Samayapuram

Date: 04.06.2025

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. T. AVUDAIAPPAN, M.E.,Ph.D.,** Head of the department, **ARTIFICIAL INTELLIGENCE** for providing his encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mrs.S.GEETHA, M.E.,** Department of **ARTIFICIAL INTELLIGENCE,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

INSTITUTE

Vision:

- To serve the society by offering top-notch technical education on par with global standards.

Mission:

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all – round personalities respecting moral and ethical values.

DEPARTMENT

Vision:

- To become a renowned hub for AIDS technologies to producing highly talented globally recognizable technocrats to meet industrial needs and societal expectation.

Mission

- To impart advanced education in AI and Data Science, built upon a foundation in Computer Science and Engineering.
- To foster Experiential learning equips students with engineering skills to tackle real-world problems.
- To promote collaborative innovation in AI, Data Science , and related research and development with industries.
- To provide an enjoyable environment for pursuing excellence while upholding strong personal and professional values and ethics.

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

- **PEO1:** Compete on a global scale for a professional career in Artificial Intelligence and Data Science**.**
- **PEO2:** Provide industry-specific solutions for the society with effective communication and ethics..
- **PEO3:** Enhance their professional skills through research and lifelong learning initiatives.

PROGRAM SPECIFIC OUTCOMES (PSO)

- **PSO1:** Capable of finding the important factors in large datasets, simplify the data, and improve predictive model accuracy.
- **PSO2:** Capable of analyzing and providing a solution to a given real-world problem by designing an effective program.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The Temple Darshan Booking App is a comprehensive digital solution designed to simplify and modernize the process of scheduling temple visits and darshan appointments for devotees across India. In response to the growing number of temple visitors and the inefficiencies of traditional queuing systems, this app offers a user-friendly interface that enables devotees to register, log in securely, choose from a wide list of temples, book preferred dates and times, and view their booking history in a structured and accessible format. Developed using Python and SQLite for the backend and Gradio for the frontend interface, the application ensures a lightweight, cost-effective, and scalable architecture, suitable for deployment even in cloud-based environments like Google Colab. Key features include personalized user profiles displaying full name, unique user ID, and booking records; a simulated payment module for future integration of secure transaction gateways; and a curated temple list with location-wise filtering. The app promotes safer and more organized darshan experiences by reducing on-site crowding and providing a pre-planned, contactless booking system. It also supports future enhancements such as multilingual access, QR-based check-ins, real-time temple schedule updates, and dedicated dashboards for temple administrators. The project stands as a promising step towards digitizing religious services, improving operational efficiency, and enhancing devotee satisfaction. By bridging traditional practices with modern technology, the Temple Darshan Booking App sets a foundation for accessible, efficient, and spiritually fulfilling temple visits, demonstrating its potential for real-world impact and further development. The application can also be extended to include live temple queue status, feedback collection from users, and integration with government tourism platforms for broader outreach. With rising demand for digital pilgrim services, the app offers an adaptable framework that can be customized to meet the specific requirements of temples across regions and cultures, ensuring inclusivity and convenience for users of all age groups and background.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD DATABASES FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Temple Darshan Booking App is a digital platform designed to simplify and manage online bookings for temple visits. It allows users to register, log in, choose temples, book darshan slots, and view their booking history. Built with Python, SQLite, and Gradio, the app offers a lightweight and scalable solution. It improves user convenience, reduces on-site crowding, and provides features like simulated payments, personalized profiles, and temple listings. The app supports future upgrades such as QR check-ins, multilingual support, and admin dashboards, making it a modern, adaptable tool for efficient temple visit management.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Objective

The main objective of this project is to develop a web-based temple booking application that allows devotees to schedule darshan visits in advance, manage their bookings, and avoid long queues and overcrowding at temple sites.

1.2 Overview

Temples often face challenges related to crowd control, especially during festivals and weekends. Our application offers a digital solution for pre-booking darshan slots, allowing for smoother temple operations and enhanced visitor experiences. The system is simple, lightweight, and suitable for deployment on platforms like Colab or local servers.

1.3 SQL and Database Concepts

SQLite is used as the backend database to store user credentials, temple listings, and booking history. SQL queries are used for data retrieval, insertion, and validation. The use of primary and foreign keys ensures data integrity and efficient relationship mapping.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed system is structured as a modular web application, with each component handling a specific function to ensure seamless operation and user experience. The first step involves designing a clean and intuitive user interface (UI) using **Gradio**, which allows easy interaction between users and the application through a browser-based layout. This includes screens for registration, login, temple selection, and booking confirmation.

Next, the **SQLite database schema** is created to store essential data such as user details, temple information, and booking history. Tables are structured to maintain relationships and ensure data integrity, enabling efficient retrieval and updates.

The **backend logic** is developed in Python to handle core functionalities like authenticating user credentials during login, securely registering new users, managing temple listings, processing bookings, and storing the relevant data in the database.

Finally, a **payment simulation feature** is added to confirm bookings and represent the possibility of integrating real-world payment gateways in future versions. This completes the booking flow and ensures a complete, end-to-end user journey from login to confirmation.

This modular design ensures that each part of the system can be maintained and enhanced independently, supporting scalability and future development.

2.2 Block Diagram

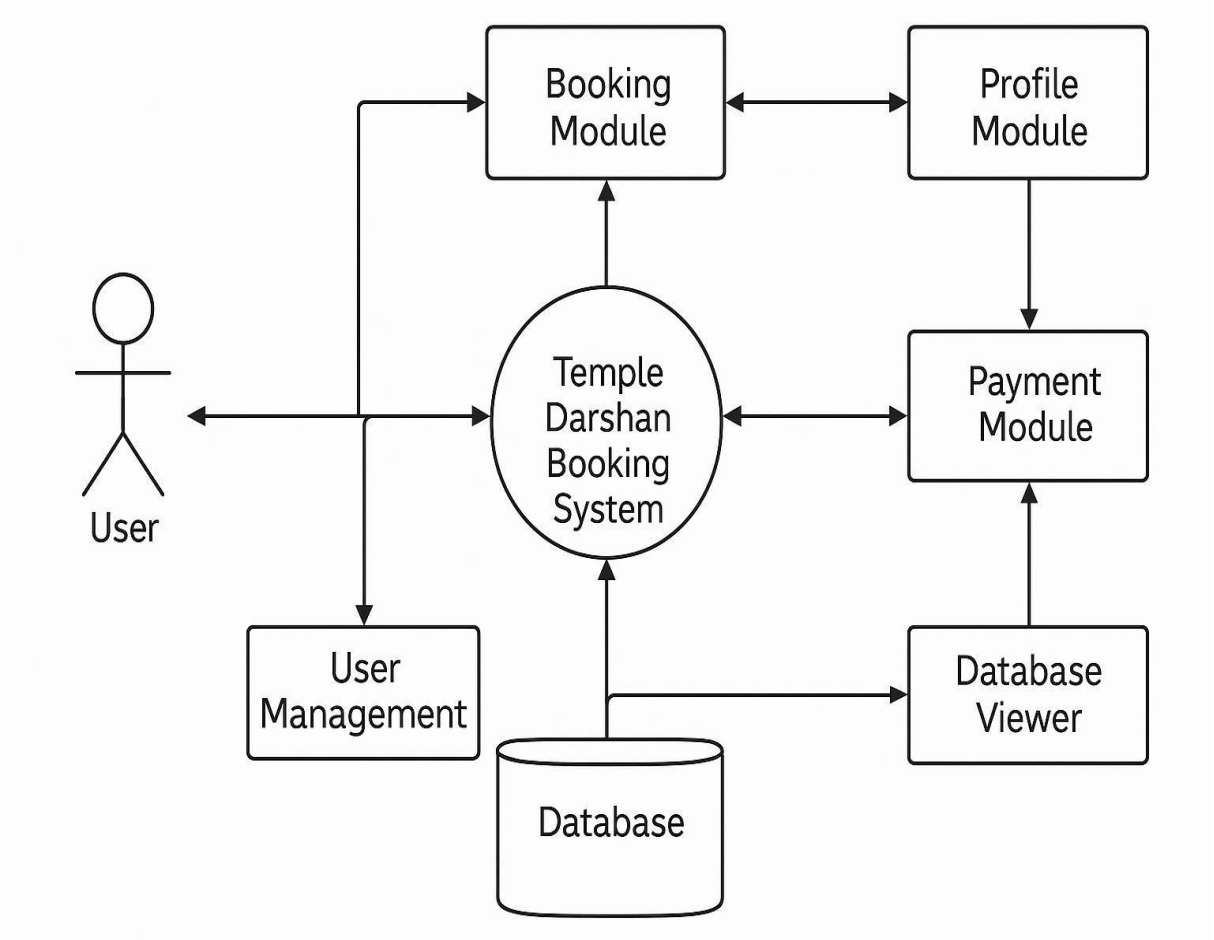

FIGURE 2.1.: Block diagram for temple dharshan app

CHAPTER 3

MODULE DESCRIPTION

3.1 Login and Registration Module

This module plays a vital role in ensuring authenticated and secure access to the application. The **Login and Registration** module is the entry point for all users. It includes a **registration page** where new users can create an account by entering their full name, email address, and password. Upon submission, the system checks whether the email already exists in the database to prevent duplicate registrations. Passwords are securely stored and validated using hashing mechanisms to enhance data security.

In the **login page**, existing users can access their account using their registered email and password. The system checks the provided credentials against the database and grants access if the data is valid. Proper validation is in place to handle common issues such as incorrect passwords, unregistered emails, and blank fields. Successful login redirects the user to the profile dashboard, while failures provide user-friendly error messages. This module ensures that only legitimate users can access booking features, maintaining the integrity and safety of the app. Furthermore, session management can be implemented to maintain active user state, improving overall usability.

3.2 Profile and Booking History Module

Once logged in, users are taken to their **profile dashboard**, which provides a personalized view of their account. The module displays key user details such as their **full name**, a system-generated **unique user ID** (formatted as `username_tem`), and their **complete darshan booking history**.

The booking history section fetches data from the SQLite database, listing all previously booked temples, the associated dates, and booking statuses. This module is dynamically updated each time a new booking is confirmed. It gives users a clear view of their engagement with the app and helps in avoiding duplicate bookings or date conflicts.

Moreover, the profile interface is designed for clarity, with well-organized cards or panels for user info and booking history. There is also scope for future enhancements like editing the

user profile, deleting booking records, or exporting booking receipts as PDFs. This module ensures transparency for users and supports continuity in temple visits.

3.3 Temple Booking Module

The **Temple Booking Module** is the core feature of the application. It provides users with a rich and organized list of over 100 temples across different districts of India. Each temple is displayed with its name and location, and the user can select a preferred one using dropdowns or clickable lists.

Once a temple is selected, the user can choose the desired date for darshan using a date picker. The interface ensures that users cannot pick past dates or unavailable slots, providing basic validation to improve user experience. After date selection, the user proceeds to the booking confirmation.

Upon confirmation, the booking is saved in the database, associated with the user's profile. Internally, the system generates a **booking ID** and stores details like user ID, temple ID, and date. This seamless experience simplifies the traditional darshan booking process, making it digital, quick, and user-friendly. The design also leaves room for future features such as time slots, darshan types (VIP/Regular), or group bookings.

3.4 Payment Simulation Module

The **Payment Simulation Module** is designed to mimic the behavior of a real-world payment gateway. Although it does not involve any actual financial transactions, it simulates the process through a user-friendly interface. After confirming the booking, users are directed to the payment screen, where they can select mock payment options like UPI, Debit/Credit Card, or Net Banking.

Once a payment option is selected and the "Pay Now" button is clicked, a simulated success message appears, marking the booking as "Paid" in the database. The payment status is updated in real time and is reflected in both the user's booking history and the admin view (if enabled).

This simulation prepares the app for **future integration with real payment gateways** such as Razorpay, Paytm, or Stripe. The backend is already structured to handle payment statuses, which makes the transition to a real system easier. It also supports the future possibility of integrating **donation options**, providing a complete spiritual and philanthropic booking experience.

3.5 Admin View (Only for administrators)

The **Admin View** module is an optional but powerful backend feature reserved for system administrators. It provides an overview of the application's usage, including all registered users, temple booking records, and payment status logs. Admins can log in through a separate access panel and manage core aspects of the system.

Through this module, administrators can perform essential database management tasks such as viewing bookings by date, temple, or user, exporting booking records, or filtering unpaid bookings. It also provides valuable insights into user behavior, most-visited temples, and booking patterns.

Additionally, this module can be extended with **analytical tools**, including charts or statistics, showing temple-wise bookings, monthly trends, or peak booking hours. Admins may also add, remove, or update temple entries, keeping the system current and scalable.

summary

Once authenticated, users access the **Profile and Booking History Module**, where they can view personal information like their full name, a unique user ID, and a complete list of their past bookings, all dynamically fetched from the database.

The **Temple Booking Module** allows users to browse and select from over 100 temples across different districts. After selecting a temple and a preferred darshan date, the booking is saved to the database and linked to the user's account. The **Payment Simulation Module** provides a basic interface to confirm bookings, simulating payment options like UPI or cards, and updates booking statuses accordingly.

Finally, the optional **Admin View Module** enables system administrators to monitor and manage user bookings, view trends, and maintain the temple database. Together, these modules offer a user-friendly, functional, and scalable system for digital darshan scheduling.

CHAPTER 4

CONCLUSION

The Temple Darshan Booking System successfully demonstrates the potential of leveraging modern web technologies to enhance the management and experience of religious activities. By providing a platform that digitizes temple darshan bookings, the system brings convenience to devotees while also streamlining temple administration. This approach significantly reduces physical crowding, eliminates the need for long queues, and ensures a more structured and time-efficient darshan experience. The system supports core functionalities such as user registration, login authentication, temple listing, booking scheduling, and basic profile management. Its modular architecture and use of tools like Gradio and SQLite ensure ease of development and scalability.

Additionally, the centralized storage of user and booking data enables better record-keeping and data analysis, which can help in predicting peak times and planning crowd control measures effectively. The simplified user interface ensures accessibility for users with varying technical backgrounds, making it suitable for use in rural as well as urban areas. Furthermore, the platform encourages digital literacy among users by introducing them to online services in the context of faith and tradition. The system can also be adapted to support festival bookings, special poojas, and temple tours, making it a versatile solution for temple management in the digital age. Overall, the project proves that even traditional religious services can greatly benefit from digital transformation, ultimately leading to more satisfactory experiences for both devotees and temple authorities.

Future Scope

- Integrating real-time payment gateways.
- Adding multi-language support.
- Creating a mobile app version using Ionic or Flutter.
- Enabling SMS/Email booking confirmations.
- Admin dashboards for temple authorities with booking insights and visitor management.

CHAPTER 5

APPENDIX A SOURCE CODE

```
import gradio as gr
import sqlite3
import datetime
import pandas as pd

# ------------------ SQLite Setup ------------------
conn = sqlite3.connect("temple_darisan.db", check_same_thread=False)
cursor = conn.cursor()

 Drop tables if they already exist (for dev/testing)
cursor.execute("DROP TABLE IF EXISTS users")
cursor.execute("DROP TABLE IF EXISTS bookings")

# Create users table
cursor.execute("""
CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    email TEXT UNIQUE,
    password TEXT,
    full_name TEXT
)
""")

# Create bookings table
cursor.execute("""
CREATE TABLE IF NOT EXISTS bookings (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
    user_id INTEGER,
    temple_name TEXT,
    darisan_date TEXT,
    time_slot TEXT,
    payment_status TEXT,
    FOREIGN KEY(user_id) REFERENCES users(id)
)
""")
conn.commit()

# Track current user session info
current_user_id = {"id": None}
current_user_name = {"name": ""}
current_user_booking_id = {"id": None}  # Store booking id for cancellation

# ----------------- Backend Functions -----------------

def register_user(name, email, password):
    if not name or not email or not password:
        return "❗ All fields are required."
    try:
        cursor.execute("INSERT INTO users (full_name, email, password)
VALUES (?, ?, ?)",
                       (name, email, password))
        conn.commit()
        return f"✅Registered successfully! Please log in using: {email}"
    except sqlite3.IntegrityError:
        return "+ Email already registered. Please log in instead."
    except Exception as e:
```

```
        return f"+ Error: {str(e)}"


def login_user(email, password):
    cursor.execute("SELECT * FROM users WHERE email=? AND
password=?", (email, password))
    user = cursor.fetchone()
    if user:
        current_user_id["id"] = user[0]
        current_user_name["name"] = user[3]
        current_user_booking_id["id"] = None  # Reset booking id on new login
        return f" Welcome, {user[3]}!", user[3]
    else:
        return "+ Invalid email or password.", ""


def book_darisan(temple, date, slot):
    if not current_user_id["id"]:
        return " Please log in first."
    # Optional: validate date format and date not in past
    try:
        booking_date = datetime.datetime.strptime(date, "%Y-%m-%d").date()
        if booking_date < datetime.date.today():
            return "+ Darisan date cannot be in the past."
    except ValueError:
        return "+ Date format must be YYYY-MM-DD."


    cursor.execute(
        "INSERT INTO bookings (user_id, temple_name, darisan_date, time_slot,
payment_status) VALUES (?, ?, ?, ?, ?)",
        (current_user_id["id"], temple, date, slot, "Pending"))
```

```
    conn.commit()
    return f"✅ Booking for {temple} on {date} at {slot} successful! Proceed to Payment tab."

def make_payment():
    if not current_user_id["id"]:
        return "❌ Login first."
    cursor.execute("SELECT * FROM bookings WHERE user_id=? AND payment_status='Pending'", (current_user_id["id"],))
    booking = cursor.fetchone()
    if booking:
        cursor.execute("UPDATE bookings SET payment_status='Paid' WHERE id=?", (booking[0],))
        conn.commit()
        return "✅Payment successful!"
    else:
        return "+ No pending bookings found."

def get_paid_booking(user_id):
    cursor.execute("""
        SELECT b.id, u.full_name, b.temple_name, b.darisan_date, b.time_slot
        FROM bookings b JOIN users u ON b.user_id = u.id
        WHERE b.user_id=? AND b.payment_status='Paid'
        ORDER BY b.darisan_date DESC LIMIT 1
    """, (user_id,))
    booking = cursor.fetchone()
    if booking:
        current_user_booking_id["id"] = booking[0]
        return (
            "✅ Booking found below.",
```

```
            booking[1], booking[2], booking[3] + " at " + booking[4],
            gr.update(visible=True)
        )
    else:
        current_user_booking_id["id"] = None
        return "+ No paid booking found.", "", "", "", gr.update(visible=False)

def cancel_booking(user_id):
    booking_id = current_user_booking_id.get("id")
    if not booking_id:
        return "  No booking to cancel."
    cursor.execute("DELETE FROM bookings WHERE id=? AND user_id=?", (booking_id, user_id))
    conn.commit()
    current_user_booking_id["id"] = None  # Reset booking id after cancellation
    return "  Booking cancelled. 80% refund has been initiated."

def view_db():
    users_df = pd.read_sql_query("SELECT id, full_name, email FROM users", conn)
    bookings_df = pd.read_sql_query("""
        SELECT b.id, u.full_name, b.temple_name, b.darisan_date, b.time_slot, b.payment_status
        FROM bookings b JOIN users u ON b.user_id = u.id
    """, conn)
    return users_df, bookings_df

# ----------------- Gradio UI -----------------

with gr.Blocks() as app:
```

```
gr.Markdown("## 🛕 Temple Darisan Booking Software", elem_id="title", elem_classes="text-3xl font-bold text-center")

with gr.Tab("➡️ Register"):
    gr.Markdown("### Create Your Account")
    reg_name = gr.Textbox(label="Full Name")
    reg_email = gr.Textbox(label="Email")
    reg_pass = gr.Textbox(label="Password", type="password")
    reg_btn = gr.Button("Register")
    reg_msg = gr.Markdown()
    reg_btn.click(fn=register_user, inputs=[reg_name, reg_email, reg_pass], outputs=reg_msg)

with gr.Tab("🔐 Login"):
    gr.Markdown("### Login to Your Account")
    email = gr.Textbox(label="Email")
    password = gr.Textbox(label="Password", type="password")
    login_btn = gr.Button("Login")
    login_msg = gr.Markdown()
    full_name = gr.Textbox(label="Full Name", interactive=False)
    login_btn.click(fn=login_user, inputs=[email, password], outputs=[login_msg, full_name])

with gr.Tab("👤 Profile"):
    gr.Markdown("### Your Profile Info")
    profile_name = gr.Textbox(label="Full Name", value=lambda: current_user_name["name"], interactive=True)
    profile_id = gr.Textbox(label="User ID", value=lambda: str(current_user_id["id"]), interactive=True)

with gr.Tab("🛕 Book Darisan"):
```

```
gr.Markdown("### Select Your Temple and Time Slot")
temple = gr.Dropdown([
    "Tirupati, Chittoor",
    "Palani, Dindigul",
    "Rameswaram, Ramanathapuram",
    "Meenakshi Temple, Madurai",
    "Kedarnath Temple,Rudraprayag",
    "Badrinath Temple,Chamoli",
    "Somnath Temple,Gir Somnath",
    "Dwarkadhish Temple,Devbhumi Dwarka",
    "Akshardham Temple,New Delhi",
    "Mahakaleshwar Temple,Ujjain",
    "Omkareshwar Temple,Khandwa",
    "Kashi Vishwanath Temple,Varanasi",
    "Vaishno Devi Temple,Reasi",
    "Shirdi Sai Baba Temple,Ahmednagar",
    "Siddhivinayak Temple,Mumbai",
    "Mahalaxmi Temple,Kolhapur",
    "Trimbakeshwar Temple,Nashik",
    "Bhimashankar Temple,Pune",
    "Grishneshwar Temple,Aurangabad",
    "Aundha Nagnath Temple,Hingoli",
    "Parli Vaijnath Temple,Beed",
    "Ram Mandir,Ayodhya",
    "Lingaraja Temple,Bhubaneswar",
    "Jagannath Temple,Puri",
    "Konark Sun Temple,Puri",
    "Kamakhya Temple,Guwahati",
    "Kalighat Kali Temple,Kolkata",
```

"Dakshineswar Temple,Kolkata",

"Belur Math,Howrah",

"Annamalaiyar Temple,Tiruvannamalai",

"Sabarimala Temple,Pathanamthitta",

"Guruvayur Temple,Thrissur",

"Padmanabhaswamy Temple,Thiruvananthapuram",

"Chamundeshwari Temple,Mysuru",

"Hoysaleswara Temple,Hassan",

"Chennakesava Temple,Hassan",

"Virupaksha Temple,Ballari",

"Murudeshwar Temple,Uttara Kannada",

"Udupi Sri Krishna Temple,Udupi",

"Kukke Subramanya Temple,Dakshina Kannada",

"Mookambika Temple,Udupi",

"Sringeri Sharada Peetham,Chikkamagaluru",

"Dharmasthala Temple,Dakshina Kannada",

"Banashankari Temple,Bagalkot",

"Shani Shingnapur,Ahmednagar",

"Tulja Bhavani Temple,Osmanabad",

"Jejuri Khandoba Temple,Pune",

"Bhuleshwar Temple,Pune",

"Babulnath Temple,Mumbai",

"Rajarajeshwara Temple,Kannur",

"Narasimha Swamy Temple,Nandyal",

"Yadagirigutta Temple,Yadadri Bhuvanagiri",

"Keesaragutta Temple,Medchal",

"Basara Saraswathi Temple,Nirmal",

"Chilkur Balaji Temple,Ranga Reddy",

"Kanaka Durga Temple,Vijayawada",

"Simhachalam Temple,Visakhapatnam",

"Annavaram Temple,Kakinada",

"Srikalahasti Temple,Tirupati",

"Vontimitta Temple,Kadapa",

"Kotappakonda Temple,Guntur",

"Mantralayam Temple,Kurnool",

"Alampur Jogulamba Temple,Jogulamba Gadwal",

"Bhadrakali Temple,Warangal",

"Medaram Temple,Mulugu",

"Mallikarjuna Temple,Srisailam",

"Thanumalayan Temple,Kanyakumari",

"Sri Ranganathaswamy Temple,Tiruchirappalli",

"Nataraja Temple,Chidambaram",

"Brihadeeswarar Temple,Thanjavur",

"Swamimalai Murugan Temple,Thanjavur",

"Thiruthani Murugan Temple,Tiruvallur",

"Thiruvarur Temple,Tiruvarur",

"Tenkasi Temple,Tenkasi",

"Sarangapani Temple,Kumbakonam",

"Vaitheeswaran Temple,Ramanathapuram",

"Kanchi Kamakshi Temple,Kanchipuram",

"Kapaleeshwarar Temple,Chennai",

"Vadapalani Murugan Temple,Chennai",

"Santhome Basilica,Chennai",

"St. Thomas Mount,Chennai",

"Velankanni Church,Nagapattinam",

"Our Lady of Good Health Church,Velankanni",

"Little Flower Church,Trivandrum",

"St. George Forane Church,Angamaly"

```
        ], label="Select Temple")
        darisan_date = gr.Textbox(label="Darisan Date (YYYY-MM-DD)")
        time_slot = gr.Radio(choices=["Morning", "Afternoon", "Evening"],
label="Select Time Slot")
        book_btn = gr.Button("Book Darisan")
        book_msg = gr.Markdown()
        book_btn.click(fn=book_darisan, inputs=[temple, darisan_date, time_slot],
outputs=book_msg)

    with gr.Tab("💳 Payment"):
        gr.Markdown("### Make Payment for Your Booking")
        pay_btn = gr.Button("Pay Now")
        pay_msg = gr.Markdown()
        pay_btn.click(fn=make_payment, inputs=[], outputs=pay_msg)

    with gr.Tab("❌ Cancel Booking"):
        gr.Markdown("### Cancel Your Paid Booking and Get 80% Refund")
        view_msg = gr.Markdown()
        name = gr.Textbox(label="Name", interactive=True)
        temple = gr.Textbox(label="Temple", interactive=True)
        slot_time = gr.Textbox(label="Date & Time", interactive=True)
        cancel_btn = gr.Button("Cancel Booking", visible=True)
        cancel_status = gr.Markdown()


        # Show current paid booking
        view_btn = gr.Button("View My Paid Booking")
        view_btn.click(fn=get_paid_booking, inputs=[gr.State(lambda:
current_user_id["id"])],
                       outputs=[view_msg, name, temple, slot_time, cancel_btn])
```

```
        cancel_btn.click(fn=cancel_booking, inputs=[gr.State(lambda:
current_user_id["id"])], outputs=cancel_status)


    # Move the Database Info tab inside the gr.Blocks context
    with gr.Tab("🔒 Database Info"):
        gr.Markdown("### Admin Access Required to View Database Info")

        # Input field and submit button for access code
        admin_code = gr.Textbox(label="Enter Admin Access Code",
type="password")
        submit_code_btn = gr.Button("Submit Code")
        access_status = gr.Markdown()

        # Secure components (initially hidden)
        users_tbl = gr.Dataframe(headers=["ID", "Name", "Email"], visible=False)
        bookings_tbl = gr.Dataframe(headers=["Booking ID", "User Name",
"Temple", "Date", "Time Slot", "Payment Status"], visible=False)
        db_refresh = gr.Button("Refresh Database", visible=False)

        # Access code check function
        def check_admin_access(code):
            if code == "2303811714821044":  # <- Change this to your desired secure
code
                return ("✅Access granted.",
                        gr.update(visible=True),
                        gr.update(visible=True),
                        gr.update(visible=True))
            else:
                return ("❌ Invalid code. Access denied.",
```

```
                gr.update(visible=False),
                gr.update(visible=False),
                gr.update(visible=False))
        # Button click connects the function
        submit_code_btn.click(
            fn=check_admin_access,
            inputs=[admin_code],
            outputs=[access_status, users_tbl, bookings_tbl, db_refresh]
        )

        # Refresh click (only works after access)
        db_refresh.click(fn=view_db, inputs=[], outputs=[users_tbl, bookings_tbl])


if __name__ == "__main__":
    app.launch()
```

APPENDIX B SCREENSHOTS

Register module

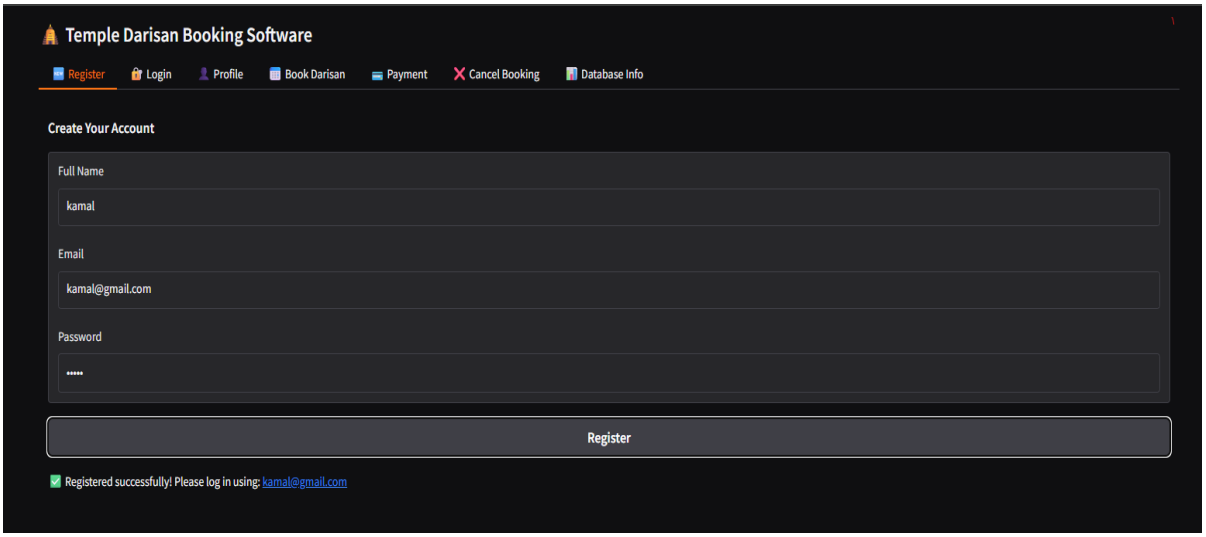

Fig B.1 register page

Login module

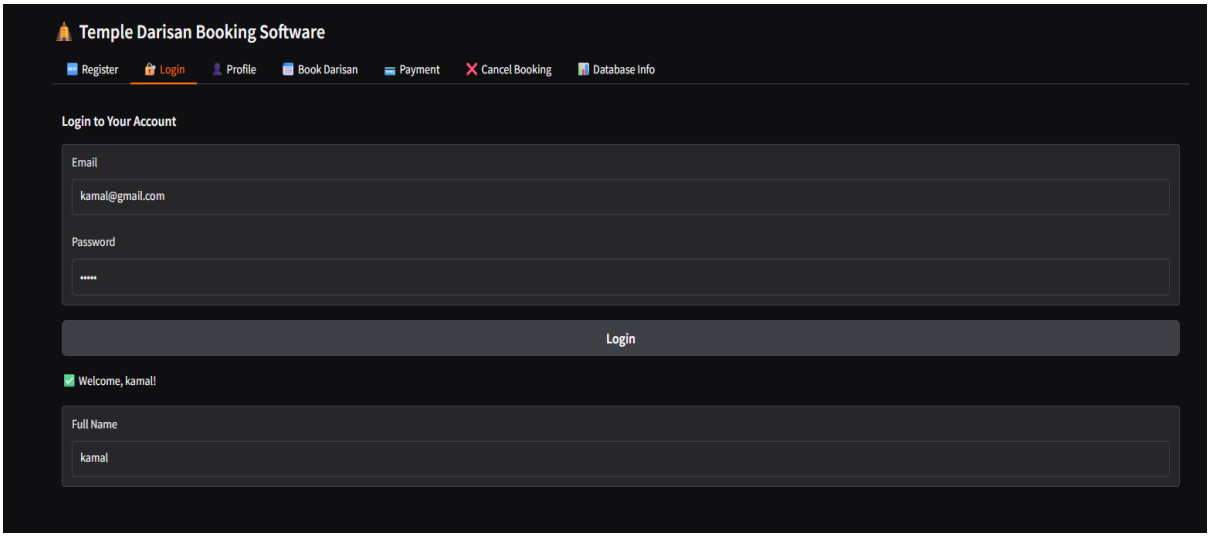

Fig B.2 Login page

Booking Module

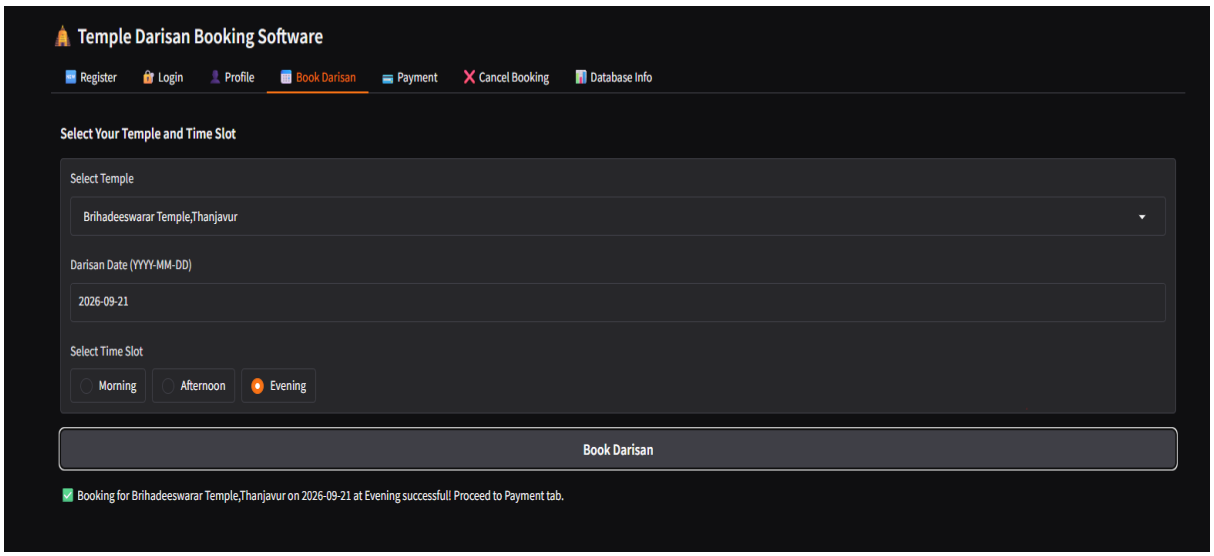

Fig B.3 Booking page

Payment Module

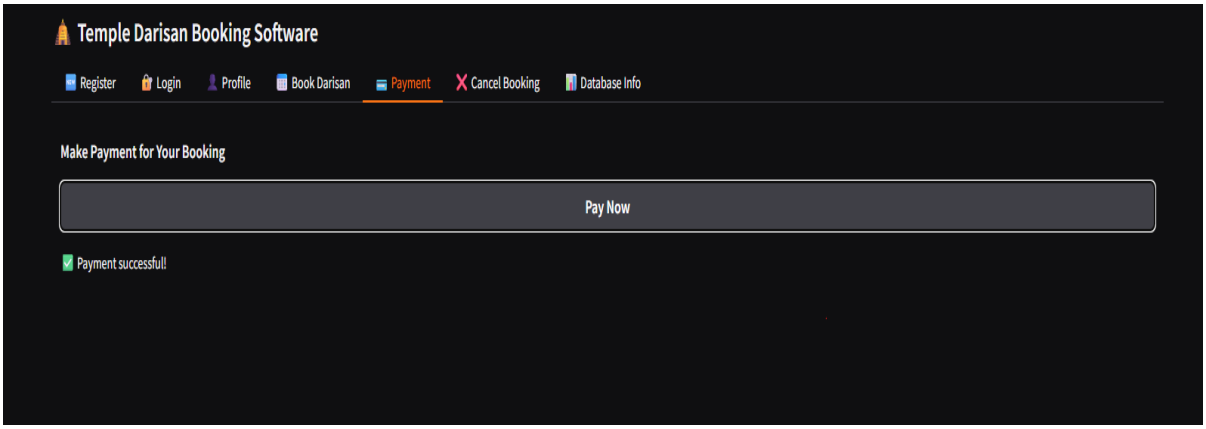

Fig B.4 Payment page

Security Module

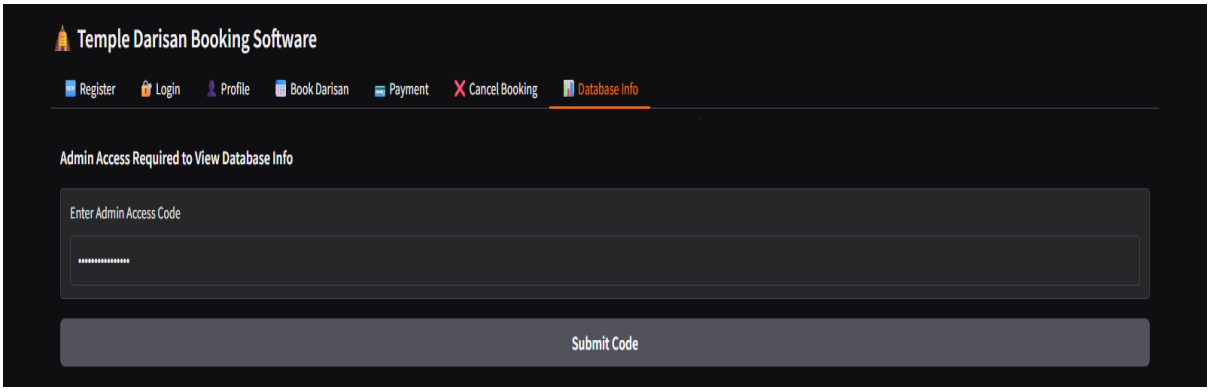

Fig B.5 Verification page

Admin Module

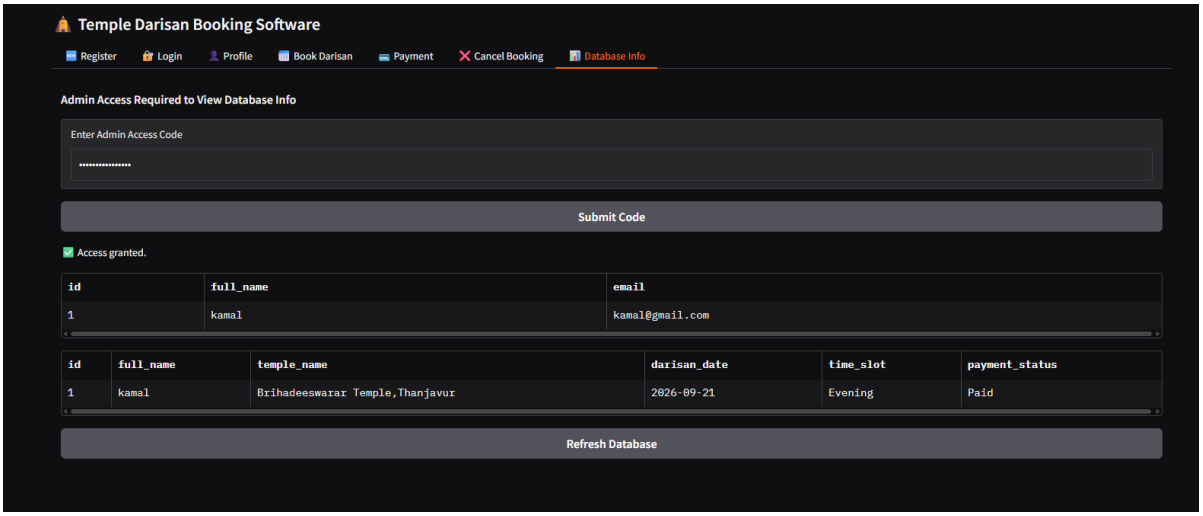

Fig B.6 Database page

REFERENCES

1. **TTD Online Booking (tirupatibalaji.ap.gov.in)**
   - Official site for Tirupati temple darshan, accommodation, and donation services.
   - Features: user login, booking calendar, ticket availability, and payment.
2. **Vaishno Devi Shrine Board (maavaishnodevi.org)**
   - Provides darshan, helicopter booking, and puja services.
   - Includes a well-structured user interface and detailed booking module.
3. **ISKCON Temple Services**
   - Many ISKCON temples offer online darshan and event bookings.
   - Simple UI, direct booking flow.

4. **"Build a Temple Booking System in Python Flask"** – Medium Article
[Search on Medium.com or Dev.to for similar guides]

5.SQLite + Flask Integration Tutorial
Good for understanding how to link user actions to data storage efficiently.

6.Gradio UI for Python Apps
Check https://www.gradio.app for building simple yet interactive Python UIs (since you use it).