AMRITA SCHOOL OF ENGINEERING

CHENNAI CAMPUS

Database Management System & Advanced Programming

20CYS204 & 19CSE201

Capstone Project

TOPIC: RFP (RainForest Puppy)

Prevent SQL Injection in Airline Ticket Booking System

TEAM MEMBERS:

- ✓ SRIDHARAN S — CH.EN.U4CYS21080
- ✓ SRISH N — CH.EN.U4CYS21081
- ✓ VIGNESWARAN S — CH.EN.U4CYS21090
- ✓ VISHNU VIGNESH A — CH.EN.U4CYS21093

ACKNOWLEDGEMENT:

Firstly, we would like to thank our Professor, Dr. Chandralekha, for providing us with such a wonderful opportunity to prove ourselves through this project. We would be forever grateful for their mentorship.

We would also like to express our sincere gratitude to the Principal, Dr. Jayakumar for being a constant source of motivation, and providing us with all the necessities.

Table of Content

Abstract:

In today's world, online flight booking is emerging as a very reliable, quick, and effective way to give your businesses a competitive edge. As a result, having a flight booking portal and a flight reservation system to run a successful travel business is becoming increasingly important for long-term success and better conversions for your business.

In the given project, we will be developing a MySQL Database which will help users to find flight details, enquire about flights between two given airports, book tickets, and know the exact price of the tickets to their desired destination.

With the help of online booking, people can book their tickets online through the internet, sitting in the comfort of their homes by a single click of their mouse.

The main objective of the project is management of the Airplane System Database. This is done by creating database of the flights between various airports, user database, booking database and many more.

To access this Airplane Ticket Booking System Project, users have to register by giving all their information such as their name, address sex, age, date of birth, nationality, mobile number, and Email ID. After registering successfully, users will be provided with a login ID and password. The Ticket Management System has applicants and administrators.

Ticket Booking Offices are located in various parts of the state and each office is managed by administrators. Each administrator has a unique identity, name, address, and starting date of work at office in a particular location.

However, such systems are also vulnerable to cyber attacks, especially SQL injection attacks. A SQL injection attack is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details. This is something we need protection against.

Objective:

This documentation illustrates the development and working of an Airplane Booking system, and how to secure it against attacks such as SQL injection. An Airplane Booking System is used as a means for a client to book flight tickets from the comfort of their home. These systems are especially vulnerable to attacks such as SQL injection, and so, require protection against them. We have achieved this by writing a code using Python to prevent such attacks. This would help the Airplane Booking System be deployed nationwide with little risk of data breach due to SQL injection attacks.

Methodology:

This Airplane Booking System makes use of MySQL in the backend. We have also used Python and Flask in the front end and for the program used for protection against SQL injection.

Software:

Bank-end	MYSQL
Front-end	Python/ Flask

Applications:

This Airplane Booking System can be deployed nation-wide. It will help people book their flight tickets from home, and it does so efficiently! For some added protection, we have included a program that would protect transactions and other queries from SQL injection attacks.

People who interact with database:

⇒ Admin
⇒ User

Different actors have different access to the database:

- ✓ Admin can
  - ❖ Check and edit the list of airports in a country.
  - ❖ Can edit and add flights between Airports.
  - ❖ Change and view the schedule for a flight.
  - ❖ Generate chart for a specific flight for a given journey.
- ✓ User can
  - ❖ User can get list of ticket that they booked
  - ❖ User can book multiple ticket.
  - ❖ And they can cancel the ticket they booked.

QUERIES:

#For future flights: Search for future flights based on source city/airport name, destination city/airport name, departure date for one way (departure and return dates for round trip)

---For go trip:

- SELECT * FROM flight WHERE (departure_airport, arrival_airport, date(departure_datetime)) = (%s,%s,%s) AND departure_datetime>CURRENT_TIMESTAMP

---For round trip:

- SELECT * FROM flight WHERE (departure_airport, arrival_airport, date(departure_datetime)) = (%s,%s,%s) AND departure_datetime>CURRENT_TIMESTAMP

#For flight status: see the flights status based on airline name, flight number,

arrival/departure

- SELECT flight_status FROM flight WHERE (airline_name,flight_number,date(departure_datetime))= (%s,%s,%s)

Register:

#Customer register:

- INSERT INTO customer VALUES (%s, %s, md5(%s), %s, %s, %s, %s, %s, %s, %s, %s, %s)

#Staff register:

- INSERT INTO staff VALUES(%s, md5(%s), %s, %s, %s, %s)

Login: 2 types of user login (Customer, and Airline Staff).

#Airline Staff login:

- SELECT * FROM staff WHERE username = %s and userpassword = md5(%s)

#Customer login:

- SELECT * FROM customer WHERE email = %s and customer_password = md5(%s)'

Customer use cases:

View My flights:

#View past flights:

- SELECT flight.airline_name, flight.flight_number, flight.departure_datetime,flight.departure_airport,flight.arrival_datetime, flight.arrival_airport, purchase.sold_price FROM customer, purchase, ticket NATURAL JOIN flight WHERE customer.email = purchase.email AND purchase.ticket_id = ticket.ticket_id AND flight.departure_datetime <=
  CURRENT_TIMESTAMP AND customer.email = %s

#View future flights: The default should be showing for the future flights.

- SELECT flight.airline_name, flight.flight_number, flight.departure_datetime,flight.departure_airport,flight.arrival_datetime, flight.arrival_airport, purchase.sold_price FROM customer, purchase, 'ticket NATURAL JOIN flight WHERE customer.email =purchase.email

AND purchase.ticket_id = ticket.ticket_id AND flight.departure_datetime >= CURRENT_TIMESTAMP AND customer.email = %s

Search for flights:

#Search for go trip:

- SELECT * FROM flight WHERE (departure_airport, arrival_airport, date(departure_datetime)) =(%s,%s,%s) AND departure_datetime>CURRENT_TIMESTAMP

#Search for round trip:

- SELECT * FROM flight WHERE (departure_airport, arrival_airport, date(departure_datetime)) =(%s,%s,%s) AND departure_datetime>CURRENT_TIMESTAMP

Purchase tickets:

#Customer choose the tickets by showing the capacity, occupancy, ticket id and base

price of the ticket

- SELECT COUNT(ticket_id) AS capacity FROM ticket WHERE (airline_name,flight_number,departure_datetime) = (%s,%s,%s)
- SELECT COUNT(ticket_id) AS occupancy FROM ticket NATURAL JOIN purchase WHERE (airline_name,flight_number,departure_datetime)= (%s,%s,%s)
- SELECT ticket_id FROM ticket NATURAL LEFT OUTER JOIN purchase WHERE (airline_name,flight_number,departure_datetime) = (%s,%s,%s) AND email IS null
- SELECT base_price FROM flight WHERE (airline_name,flight_number,departure_datetime) =(%s,%s,%s)

#Customer buy the tickets

- INSERT INTO purchase VALUES (%s,%s,%s,%s,%s,%s,%s,CURRENT_TIMESTAMP)

Give Ratings and Comment on previous flights:

- SELECT * FROM customerrate WHERE email = %s

#delete the comment

- DELETE FROM customerrate WHERE (email,airline_name,flight_number,departure_datetime)
- = (%s,%s,%s,%s)

#add the comment

- INSERT INTO customerrate VALUES (%s,%s,%s,%s,%s,%s)

Track My Spending:

#Track the total money spent for the past year

- SELECT SUM(sold_price) AS yearSpending, CURRENT_DATE AS enddate,date(CURRENT_DATE-10000) AS startdate FROM purchase WHERE email=%s AND purchase_datetime >= date(CURRENT_DATE-10000) GROUP BY email

#Track the total money spent for the past 6 months

- SELECT CURRENT_DATE, year(purchase_datetime) AS year, month(purchase_datetime) AS month, SUM(sold_price) AS spending FROM purchase WHERE email=%s GROUP BY year(purchase_datetime),month(purchase_datetime) HAVING (year-year(CURRENT_DATE))*12+month>= month(CURRENT_DATE)-5

#search spending for a particular range:

SELECT year(purchase_datetime) AS year, month(purchase_datetime) AS month,

SUM(sold_price) AS spending FROM purchase WHERE date(purchase_datetime) >= %s AND

date(purchase_datetime) <= %s AND email = %s GROUP BY

year(purchase_datetime),month(purchase_datetime)

Airline Staff use cases:

View flights:

#Showing all the future flights operated by the airline he/she works for the next 30 days:

SELECT * FROM flight WHERE departure_datetime <= CURRENT_TIMESTAMP + INTERVAL 30

day AND departure_datetime >= CURRENT_TIMESTAMP

#Search based on range of dates, source/destination airports/city.

SELECT * FROM flight WHERE date(departure_datetime) >= %s AND date(departure_datetime)

<= %s AND (departure_airport,arrival_airport) = (%s,%s)

#Search only based on range of dates

SELECT * FROM flight WHERE date(departure_datetime) >= %s AND date(departure_datetime)

<= %s

#Search only based on source/destination airports/city

SELECT * FROM flight WHERE (departure_airport,arrival_airport) = (%s,%s)

Create new flights:

#check if departure airport exist

SELECT * FROM airport WHERE airport_code = %s

#check if arrival airport exist

SELECT * FROM airport WHERE airport_code = %s

#check if airplane exist

SELECT * FROM airplane WHERE (airline_name,airplane_id) = (%s,%s)

#delete repeat information

DELETE FROM flight WHERE (airline_name,flight_number,departure_datetime) = (%s,%s,%s)

#Insert flight

INSERT INTO flight VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)

#create ticket

INSERT INTO ticket VALUES (%s,%s,%s,%s)

Change Status of flights:

UPDATE flight SET flight_status = %s WHERE

(airline_name,flight_number,departure_datetime) = (%s,%s,%s)

Add airplane in the system:

#see all the airplanes owned by the airline he/she works for

SELECT * FROM airplane WHERE airline_name = %s

#Add airplane

INSERT INTO airplane VALUES (%s,%s,%s)

Add new airport in the system:

#see all the airports

SELECT * FROM airport

#add the airport

INSERT INTO airport VALUES (%s,%s,%s)

View flight ratings:

#See all the comments and ratings of that flight given by the customers

SELECT * FROM flight WHERE date(departure_datetime) >= %s AND date(departure_datetime)

<= %s AND (departure_airport,arrival_airport) = (%s,%s)

#See average ratings of that flight given by the customers

SELECT AVG(rate) AS avg_rating FROM customerrate WHERE

(airline_name,flight_number,departure_datetime) = (%s,%s,%s)

View frequent customers:

SELECT customer.email AS email, customer_name, COUNT(purchase.ticket_id) AS

ticketsBought, SUM(sold_price) AS totalSpending FROM customer,purchase,ticket WHERE

purchase.email = customer.email AND purchase_datetime >= CURRENT_TIMESTAMP -

INTERVAL 1 year AND purchase.ticket_id=ticket.ticket_id AND ticket.airline_name = %s

GROUP BY customer_name ORDER BY ticketsBought DESC LIMIT 10

#Seach for particular customer

SELECT DISTINCT airline_name, flight_number, departure_datetime, purchase.ticket_id

FROM purchase, ticket NATURAL JOIN flight WHERE purchase.ticket_id=ticket.ticket_id AND

purchase.email = %s ORDER BY departure_datetime DESC

View reports and View Earned Revenue:

#For amounts of ticket and the total revenue in the last month:

SELECT COUNT(ticket.ticket_id) AS ticketSold, SUM(purchase.sold_price) AS totalRevenue

FROM ticket,purchase WHERE ticket.ticket_id = purchase.ticket_id AND ticket.airline_name =

%s AND purchase.purchase_datetime >= CURRENT_TIMESTAMP - INTERVAL 1 month

#For amounts of ticket and the total revenue in the last year:

SELECT COUNT(ticket.ticket_id) AS ticketSold, SUM(purchase.sold_price) AS totalRevenue

FROM ticket,purchase WHERE ticket.ticket_id = purchase.ticket_id AND ticket.airline_name =

%s AND purchase.purchase_datetime >= CURRENT_TIMESTAMP - INTERVAL 1 year

#Search for the amounts of ticket and the total revenue in a time range:

SELECT year(purchase.purchase_datetime) AS theYear, month(purchase.purchase_datetime)

AS theMonth, COUNT(ticket.ticket_id) AS ticketSold, SUM(purchase.sold_price) AS

totalRevenue FROM ticket,purchase WHERE ticket.ticket_id = purchase.ticket_id AND

ticket.airline_name = %s AND date(purchase.purchase_datetime) >= %s and

date(purchase.purchase_datetime) <= %s GROUP BY year(purchase.purchase_datetime),

month(purchase.purchase_datetime)

View Top destinations:

#Find the top 3 most popular destinations for last 3 months:

SELECT airport.airport_code, airport.airport_name, airport.airport_city,

COUNT(ticket.ticket_id) AS ticket_num FROM purchase NATURAL JOIN ticket NATURAL JOIN

flight, airport WHERE flight.arrival_airport = airport.airport_code AND flight.airline_name =

%s AND purchase.purchase_datetime >= CURRENT_TIMESTAMP - INTERVAL 3 month GROUP

BY airport.airport_code ORDER BY ticket_num DESC LIMIT 3

#Find the top 3 most popular destinations for last year:

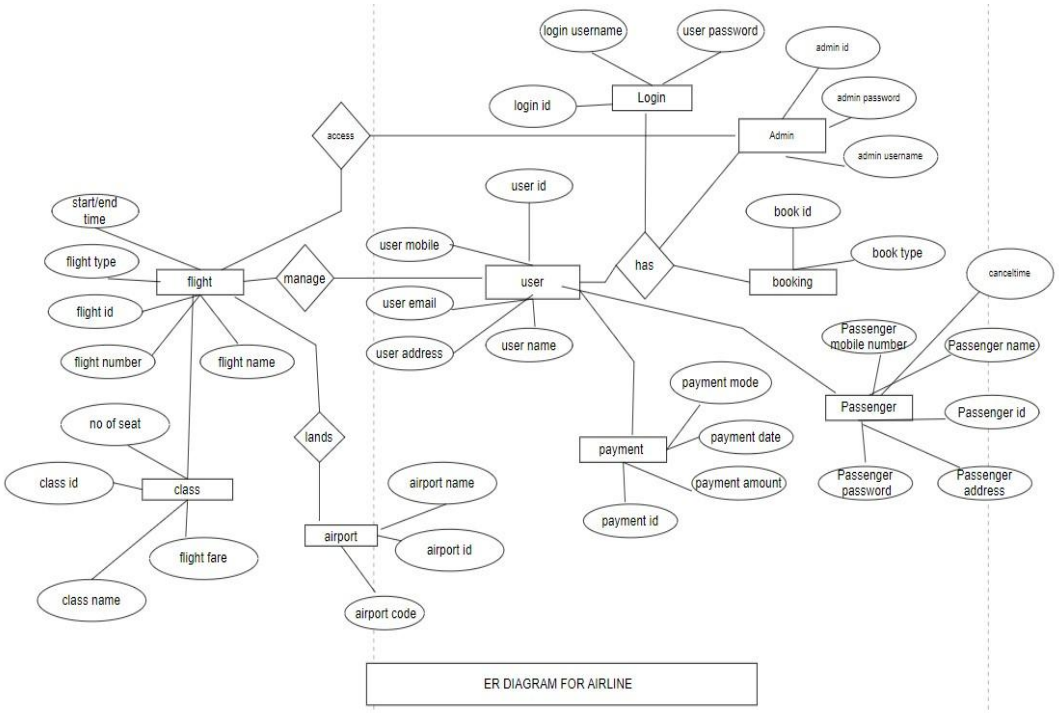
SELECT airport.airport_code, airport.airport_name, airport.airport_city,

COUNT(ticket.ticket_id) AS ticket_num FROM purchase NATURAL JOIN ticket NATURAL JOIN

flight, airport WHERE flight.arrival_airport = airport.airport_code AND flight.airline_name = %s AND purchase.purchase_datetime >= CURRENT_TIMESTAMP - INTERVAL 1 yearGROUP BY

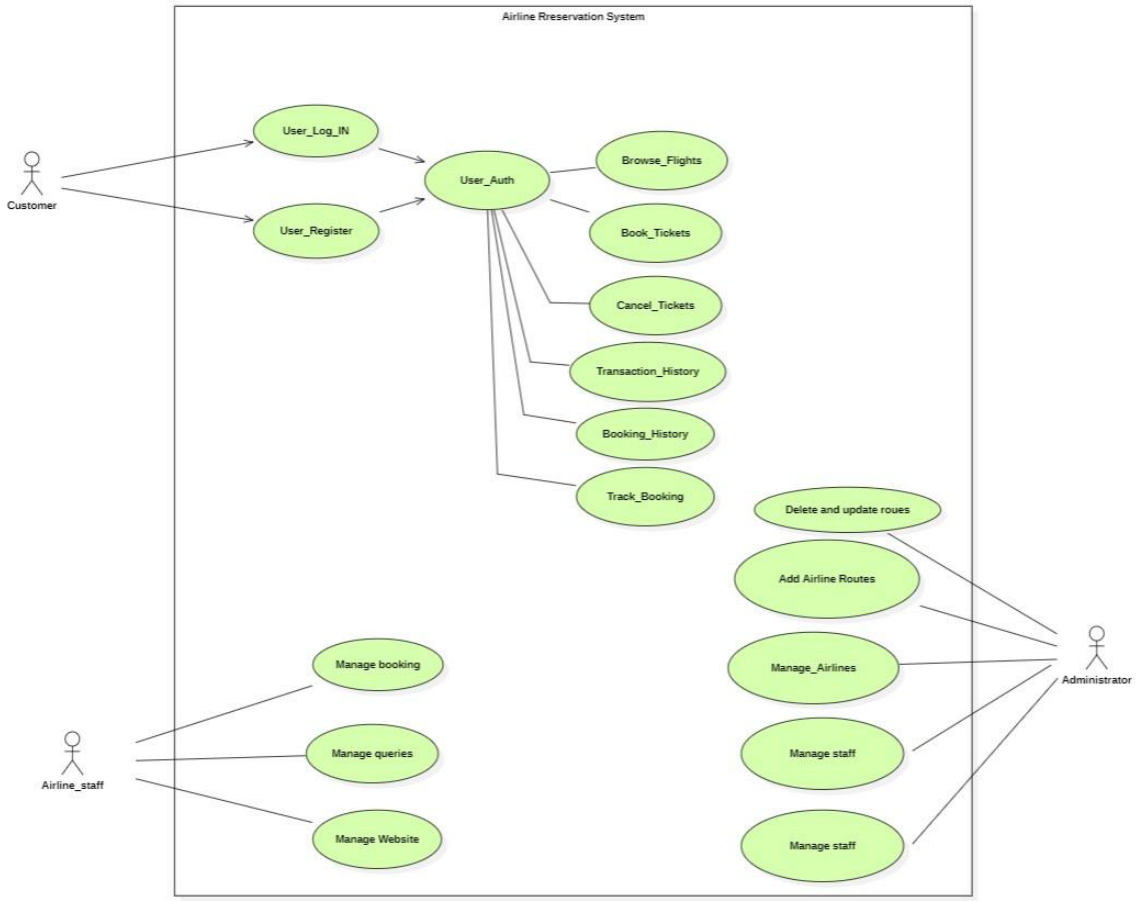
airport.airport_code ORDER BY ticket_num DESC LIMIT 3

ER-DIAGRAM:


login username
user password
admin id
login id
Login
admin password
access
Admin
admin username
user id
start/end time
book id
user mobile
has
book type
flight type
flight
manage
user
booking
canceltime
flight id
user email
Passenger mobile number
Passenger name
flight number
flight name
user address
user name
payment mode
Passenger
no of seat
lands
payment
payment date
Passenger id
class id
class
airport name
payment amount
Passenger password
Passenger address
airport
airport id
payment id
flight fare
class name
airport code
ER DIAGRAM FOR AIRLINE

UML-DIAGRAM:

USE CASE DIAGRAM



Github link for code:

https://github.com/srish-cell/AP_DBMS_Endsem_Project

Conclusion:

A lot of websites today are not secure enough from cyber attacks like SQL injection, which is the root cause for data breaches and leaks. So, in this project, we have implemented a program to prevent SQL injection. This program will secure the data that is being entered in the Airplane Database Management System, so that user information is safe.