Event Ticket Booking System

Event Ticket Booking System

Get Started

F O R   E F F O R T L E S S   B O O K I N G

04 11 2024 Foundations of SRE

Raphaelle Smyth
Askhat Bissembay
Yurii Maisuradze

Problem Statement

Users often struggle with finding, comparing, and securely booking tickets for entertainment events, including challenges with seat selection and ticket availability.

Event managers face difficulties in managing ticket sales, tracking availability, preventing overbooking, and ensuring secure transactions.

04 Nov 2024

Objective

Learning Objective: Python, Flask, mySQL, Docker

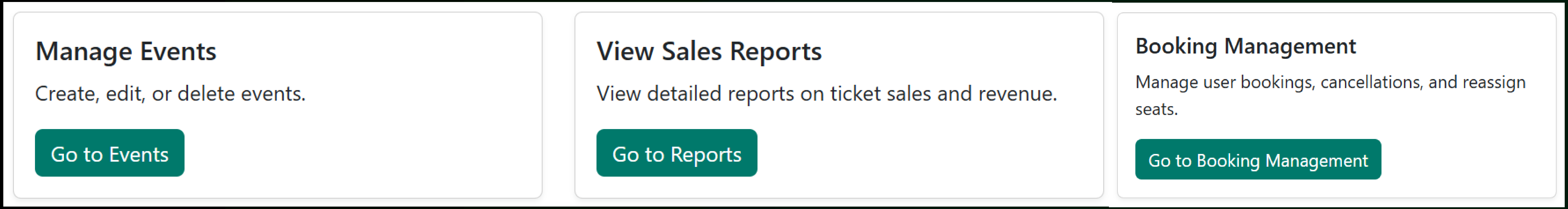
Scalable: Capable of handling multiple events or users.

Security: Implements user authentication, data encryption, and role-based access control.

04 Nov 2024

Our Solution

A web application designed to allows users to browse, search, and book tickets for a variety of entertainment events.



Manage Events

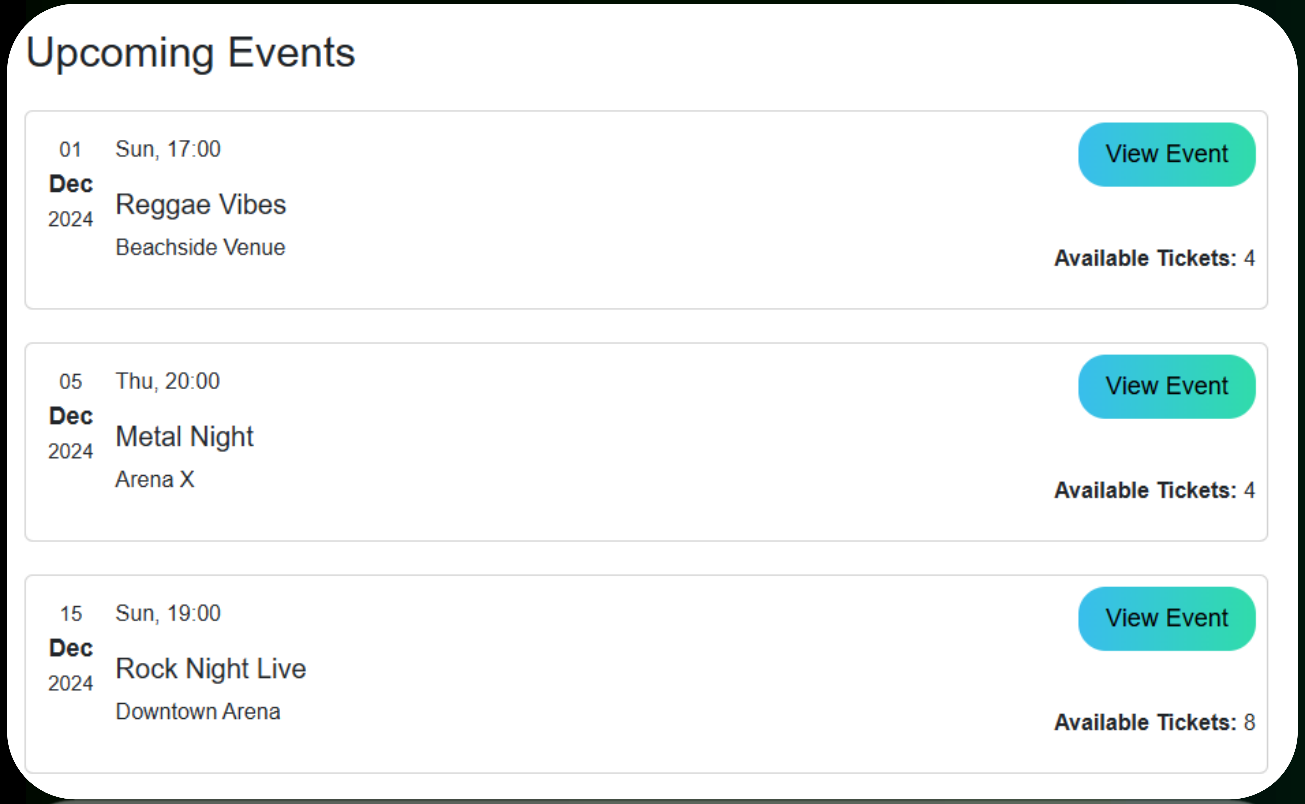
Create, edit, or delete events.

View Sales Reports

View detailed reports on ticket sales and revenue.

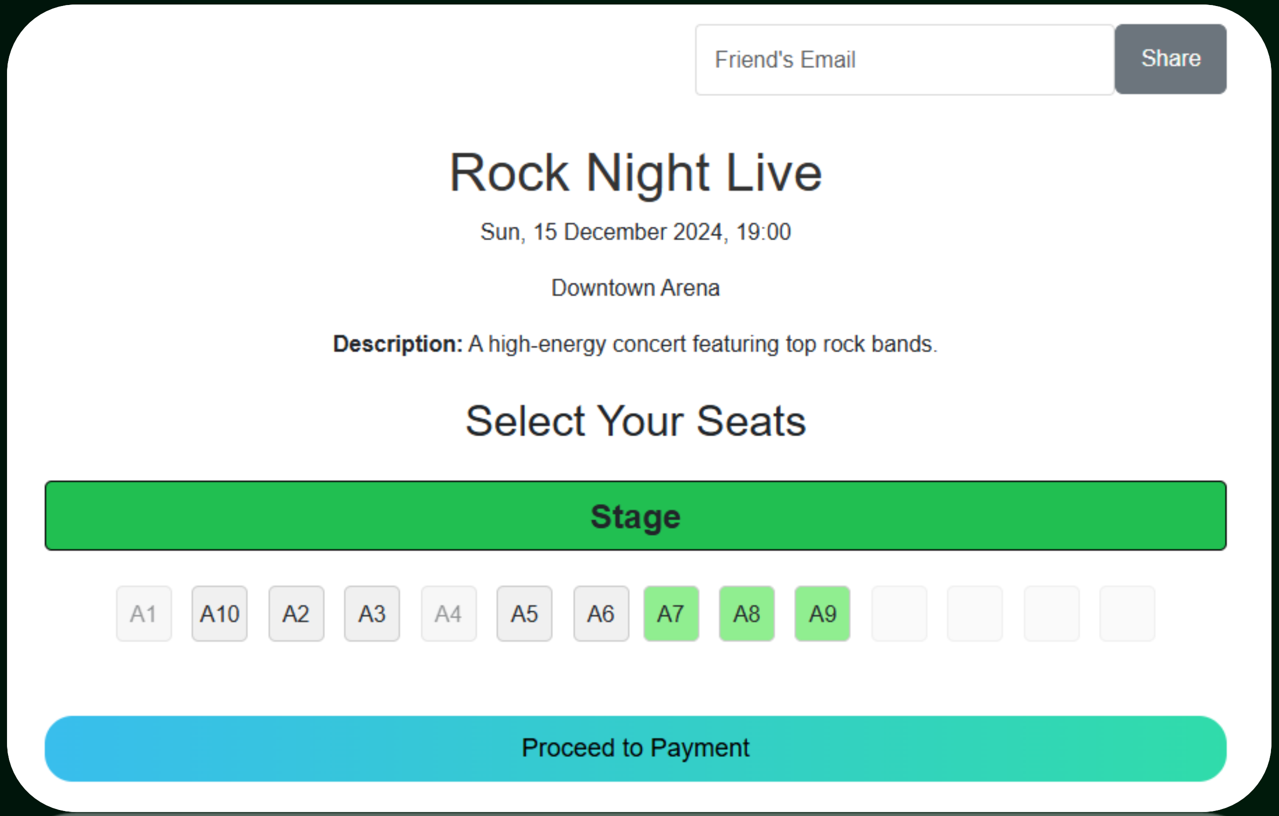
Booking Management

Manage user bookings, cancellations, and reassign seats.



Upcoming Events

01 Dec 2024	Sun, 17:00 Reggae Vibes Beachside Venue	View Event **Available Tickets:** 4
05 Dec 2024	Thu, 20:00 Metal Night Arena X	View Event **Available Tickets:** 4
15 Dec 2024	Sun, 19:00 Rock Night Live Downtown Arena	View Event **Available Tickets:** 8



Friend's Email Share

Rock Night Live

Sun, 15 December 2024, 19:00

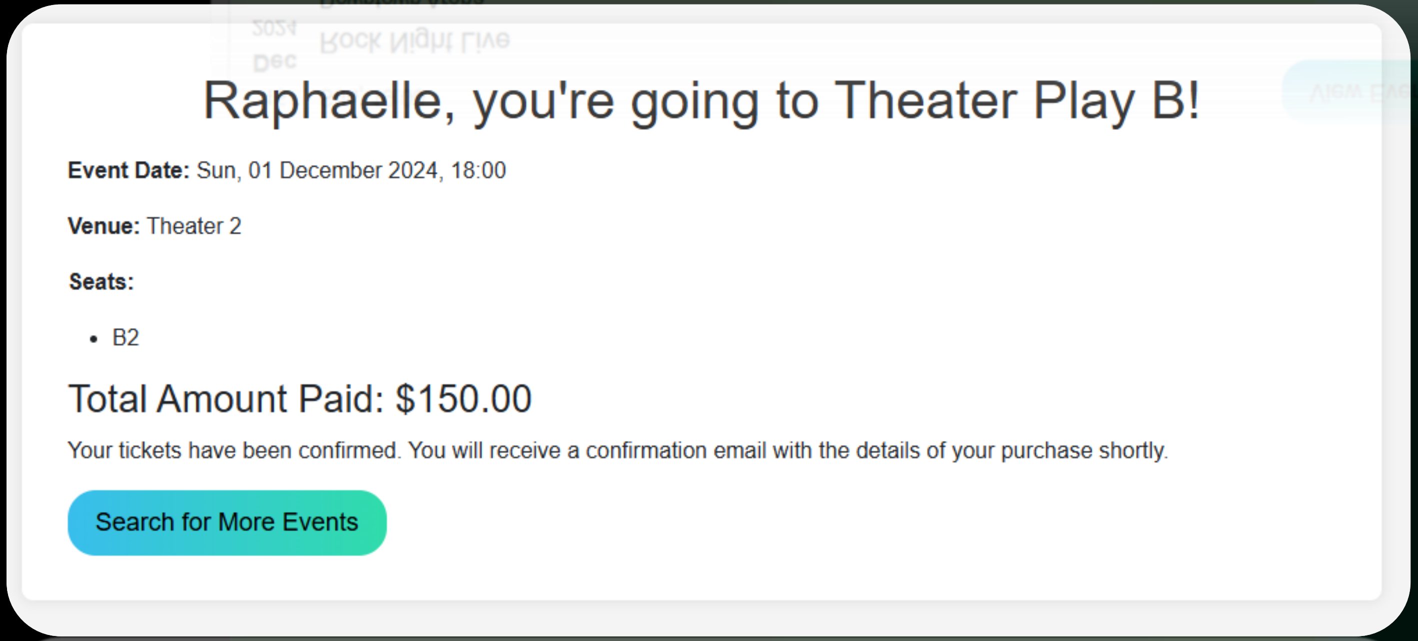
Downtown Arena

Description: A high-energy concert featuring top rock bands.

Select Your Seats

02

03



Raphaelle, you're going to Theater Play B!

Event Date: Sun, 01 December 2024, 18:00

Venue: Theater 2

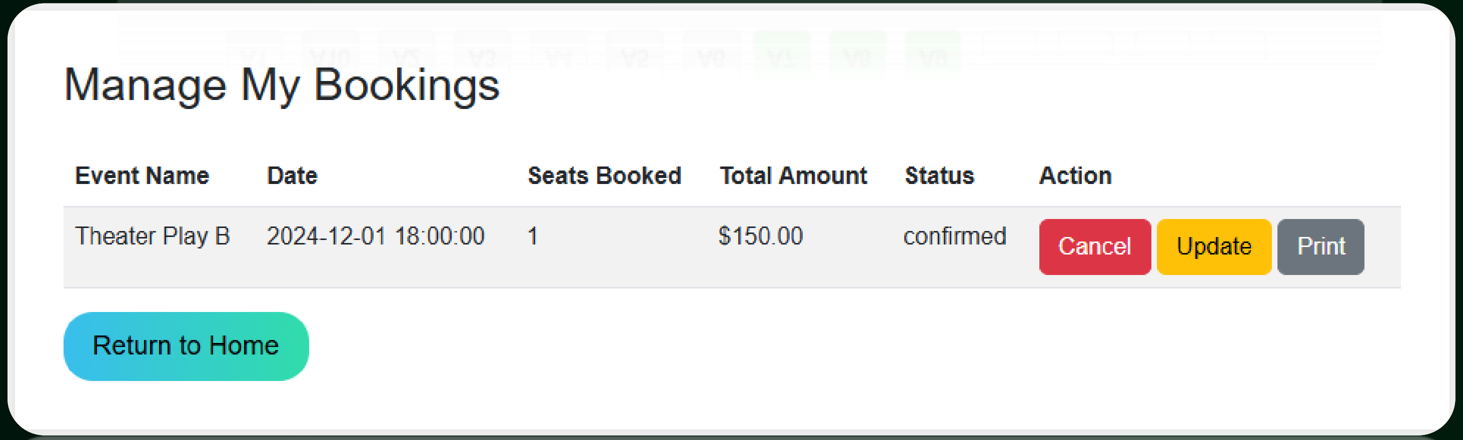
Seats:

- B2

Total Amount Paid: $150.00

Your tickets have been confirmed. You will receive a confirmation email with the details of your purchase shortly.

Search for More Events



Manage My Bookings

Event Name	Date	Seats Booked	Total Amount	Status	Action
Theater Play B	2024-12-01 18:00:00	1	$150.00	confirmed	Cancel Update Print

Return to Home

04 Nov 2024

Tech Stack

Back-End Technologies

Python

Flask web framework

MySQL Relational database management system

SQLAlchemy Object-Relational Mapping (ORM)

Front-End Technologies

HTML/CSS

Bootstrap responsive interfaces

Jinja2

JavaScript

Infrastructure

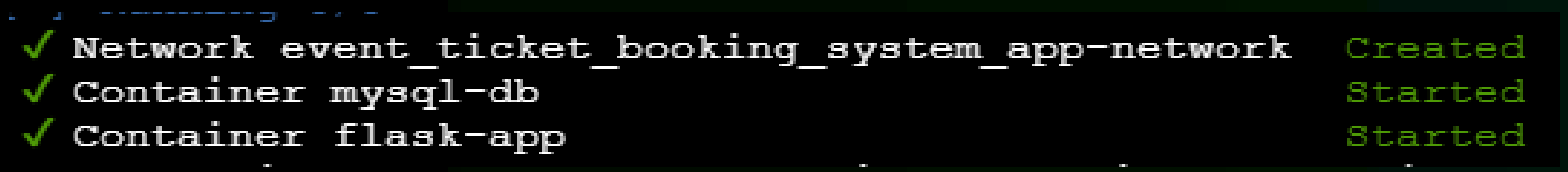
Docker

Other Tools /Platforms

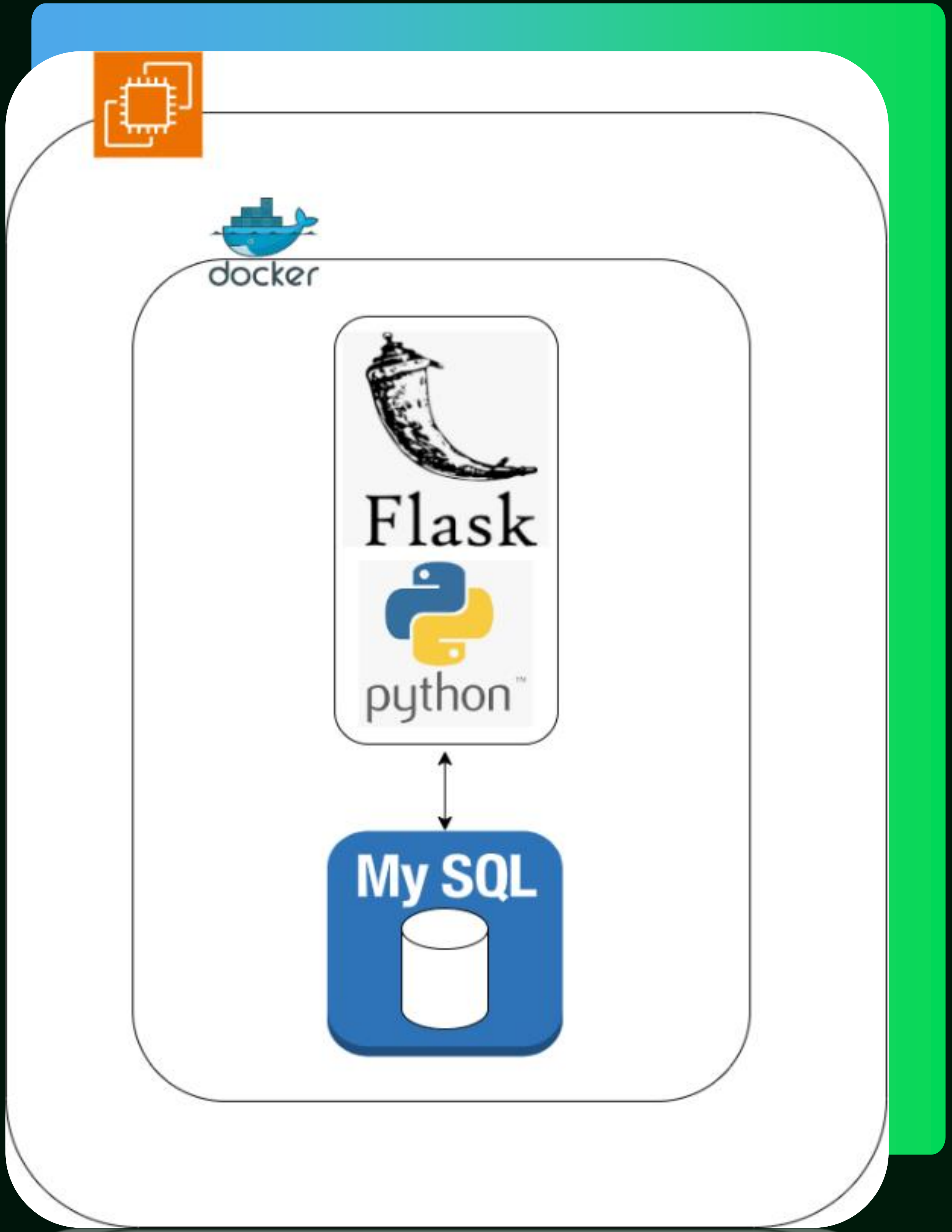
Github

04 Nov 2024

Architecture

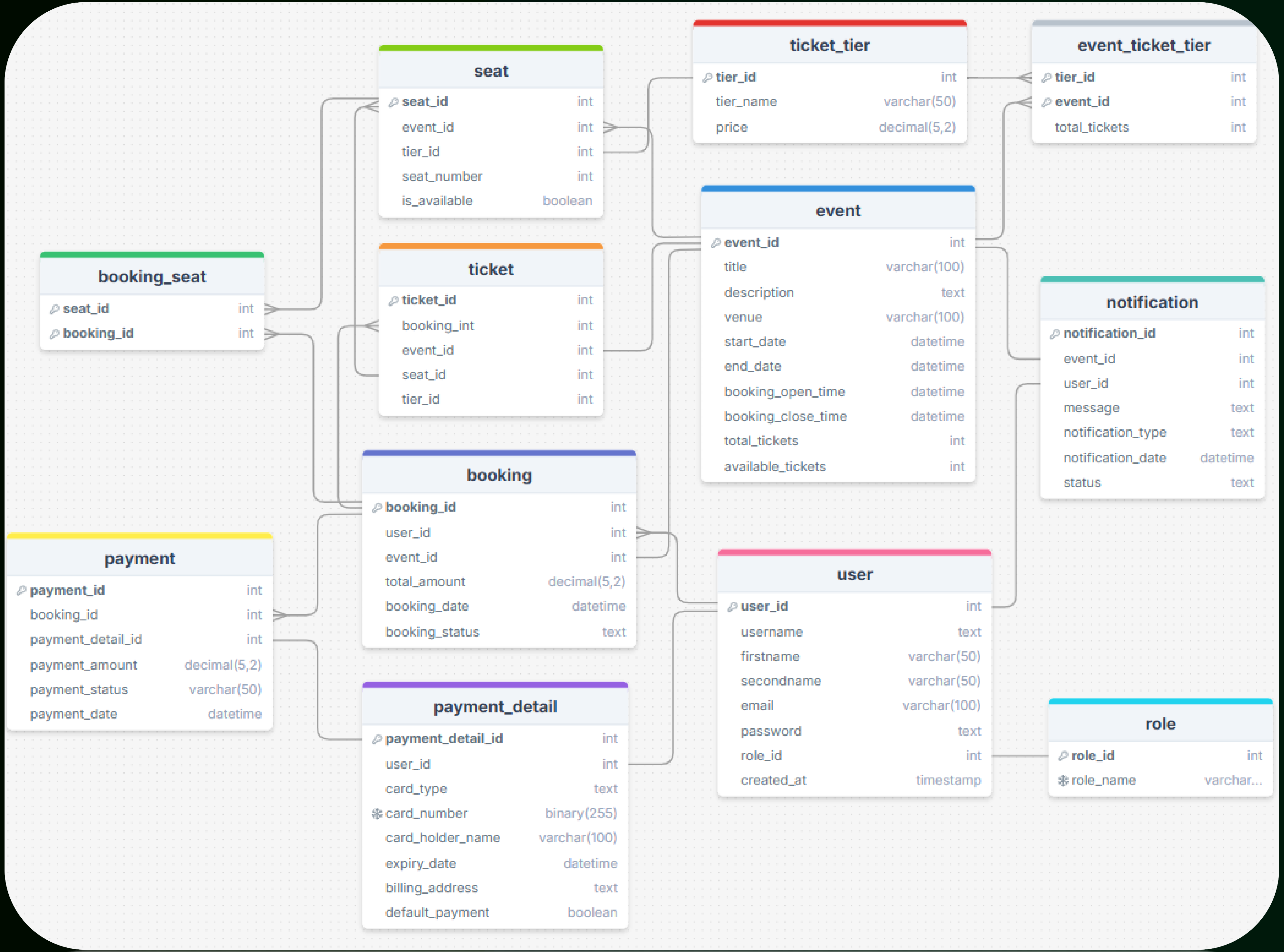


```
✓ Network event_ticket_booking_system_app-network  Created
✓ Container mysql-db                               Started
✓ Container flask-app                              Started
```



04 Nov 2024

Database Design



seat

seat_id	int
event_id	int
tier_id	int
seat_number	int
is_available	boolean

ticket_tier

tier_id	int
tier_name	varchar(50)
price	decimal(5,2)

event_ticket_tier

tier_id	int
event_id	int
total_tickets	int

booking_seat

seat_id	int
booking_id	int

ticket

ticket_id	int
booking_int	int
event_id	int
seat_id	int
tier_id	int

event

event_id	int
title	varchar(100)
description	text
venue	varchar(100)
start_date	datetime
end_date	datetime
booking_open_time	datetime
booking_close_time	datetime
total_tickets	int
available_tickets	int

notification

notification_id	int
event_id	int
user_id	int
message	text
notification_type	text
notification_date	datetime
status	text

booking

booking_id	int
user_id	int
event_id	int
total_amount	decimal(5,2)
booking_date	datetime
booking_status	text

payment

payment_id	int
booking_id	int
payment_detail_id	int
payment_amount	decimal(5,2)
payment_status	varchar(50)
payment_date	datetime

user

user_id	int
username	text
firstname	varchar(50)
secondname	varchar(50)
email	varchar(100)
password	text
role_id	int
created_at	timestamp

payment_detail

payment_detail_id	int
user_id	int
card_type	text
card_number	binary(255)
card_holder_name	varchar(100)
expiry_date	datetime
billing_address	text
default_payment	boolean

role

role_id	int
role_name	varchar...

04 Nov 2024

Data Integrity

ON DELETE CASCADE

```
FOREIGN KEY (role_id) REFERENCES role(role_id) ON DELETE CASCADE
```

NOT NULL

```
title VARCHAR(255) NOT NULL,
```

COMPOSITE UNIQUES

```
UNIQUE(event_id, seat_number)
```

ENCRYPTED DATA STORAGE

```
card_number VARBINARY(255) NOT NULL,
```

TRIGGERS

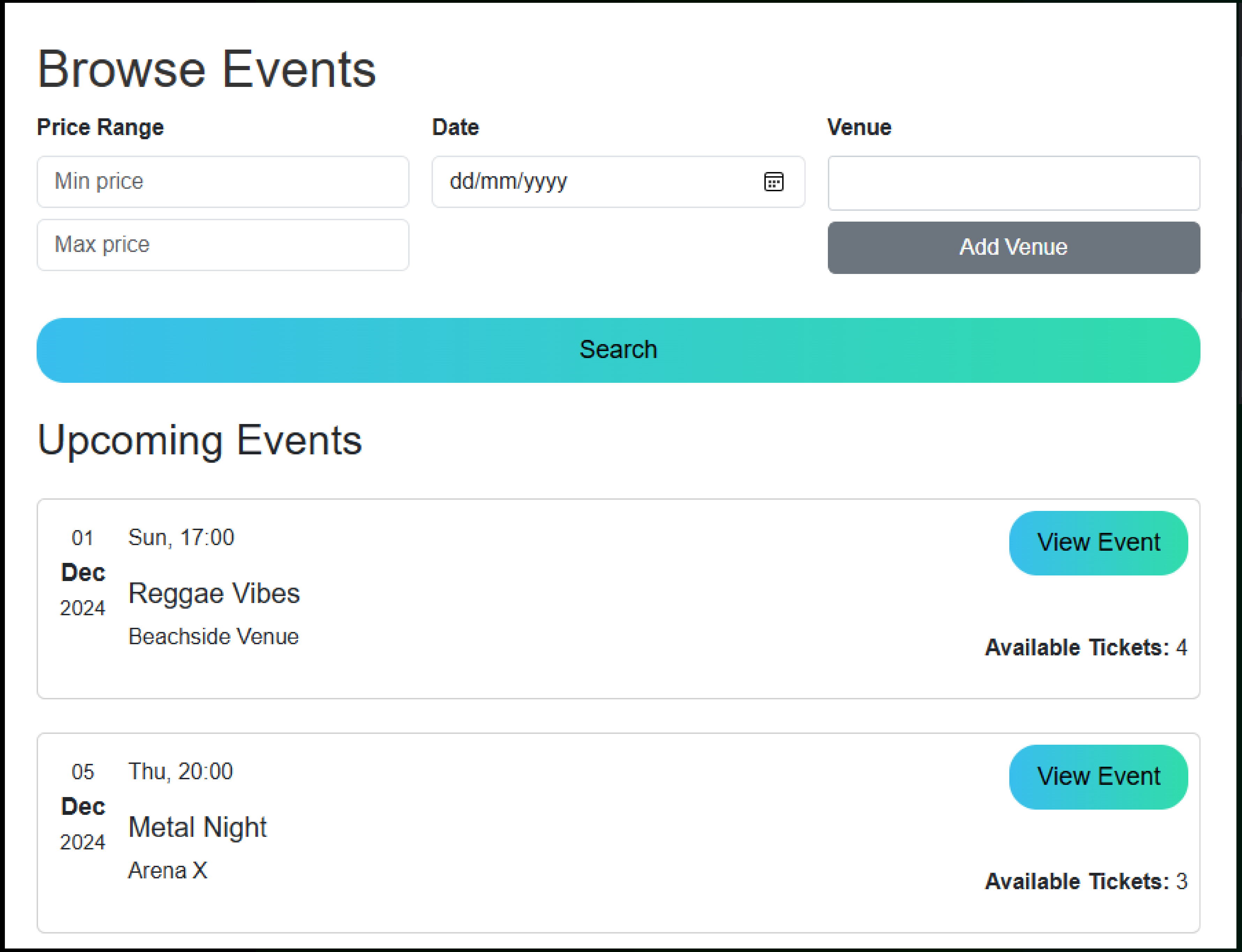
```
CREATE TRIGGER after_ticket_insert
AFTER INSERT ON ticket
FOR EACH ROW
BEGIN
    UPDATE seat
    SET is_available = FALSE
    WHERE seat_id = NEW.seat_id;
```

04 Nov 2024

User Features

04 Nov 2024

Filter & Browse



Browse Events

Price Range

Min price

Max price

Date

dd/mm/yyyy

Venue

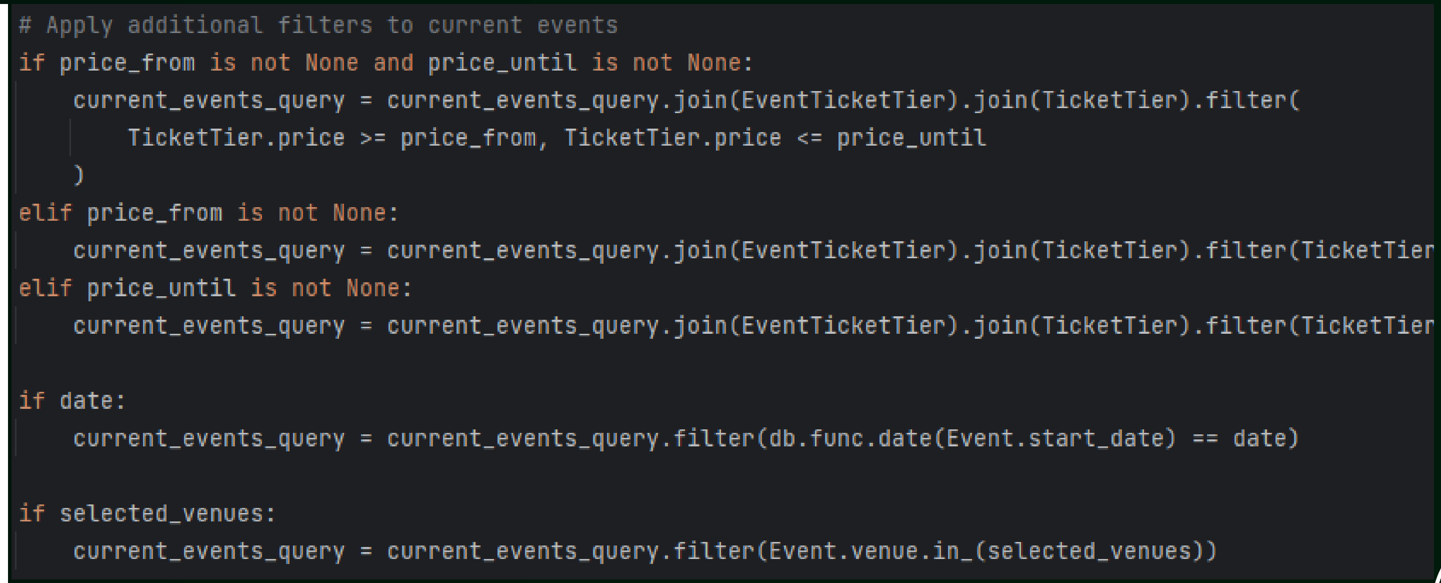
Add Venue

Search

Upcoming Events

01 **Dec** 2024 — Sun, 17:00 — Reggae Vibes — Beachside Venue — View Event — **Available Tickets:** 4

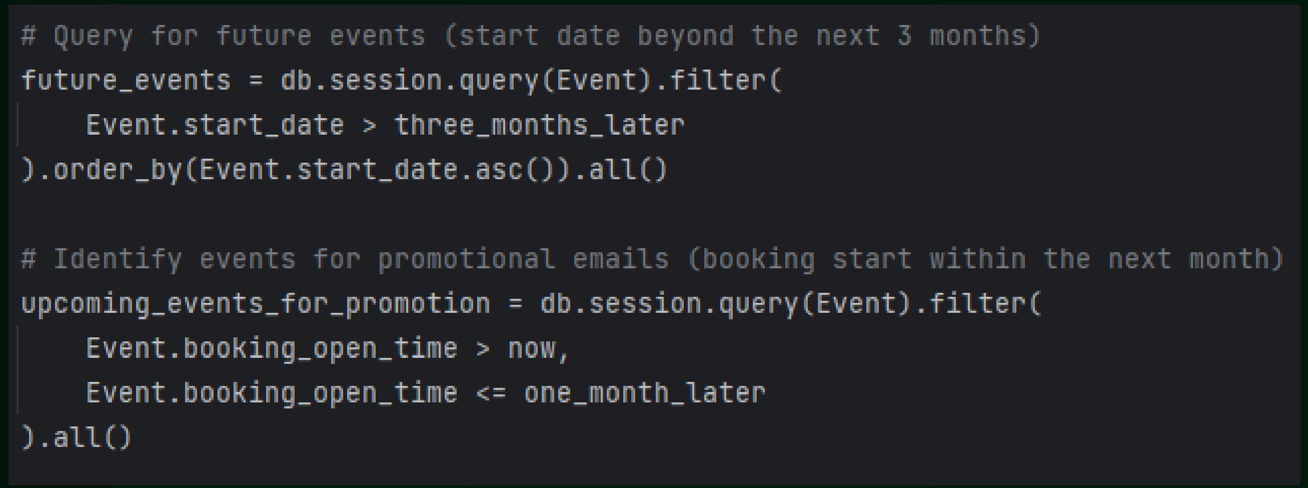
05 **Dec** 2024 — Thu, 20:00 — Metal Night — Arena X — View Event — **Available Tickets:** 3



```python
# Apply additional filters to current events
if price_from is not None and price_until is not None:
    current_events_query = current_events_query.join(EventTicketTier).join(TicketTier).filter(
        TicketTier.price >= price_from, TicketTier.price <= price_until
    )
elif price_from is not None:
    current_events_query = current_events_query.join(EventTicketTier).join(TicketTier).filter(TicketTier
elif price_until is not None:
    current_events_query = current_events_query.join(EventTicketTier).join(TicketTier).filter(TicketTier

if date:
    current_events_query = current_events_query.filter(db.func.date(Event.start_date) == date)

if selected_venues:
    current_events_query = current_events_query.filter(Event.venue.in_(selected_venues))
```

01

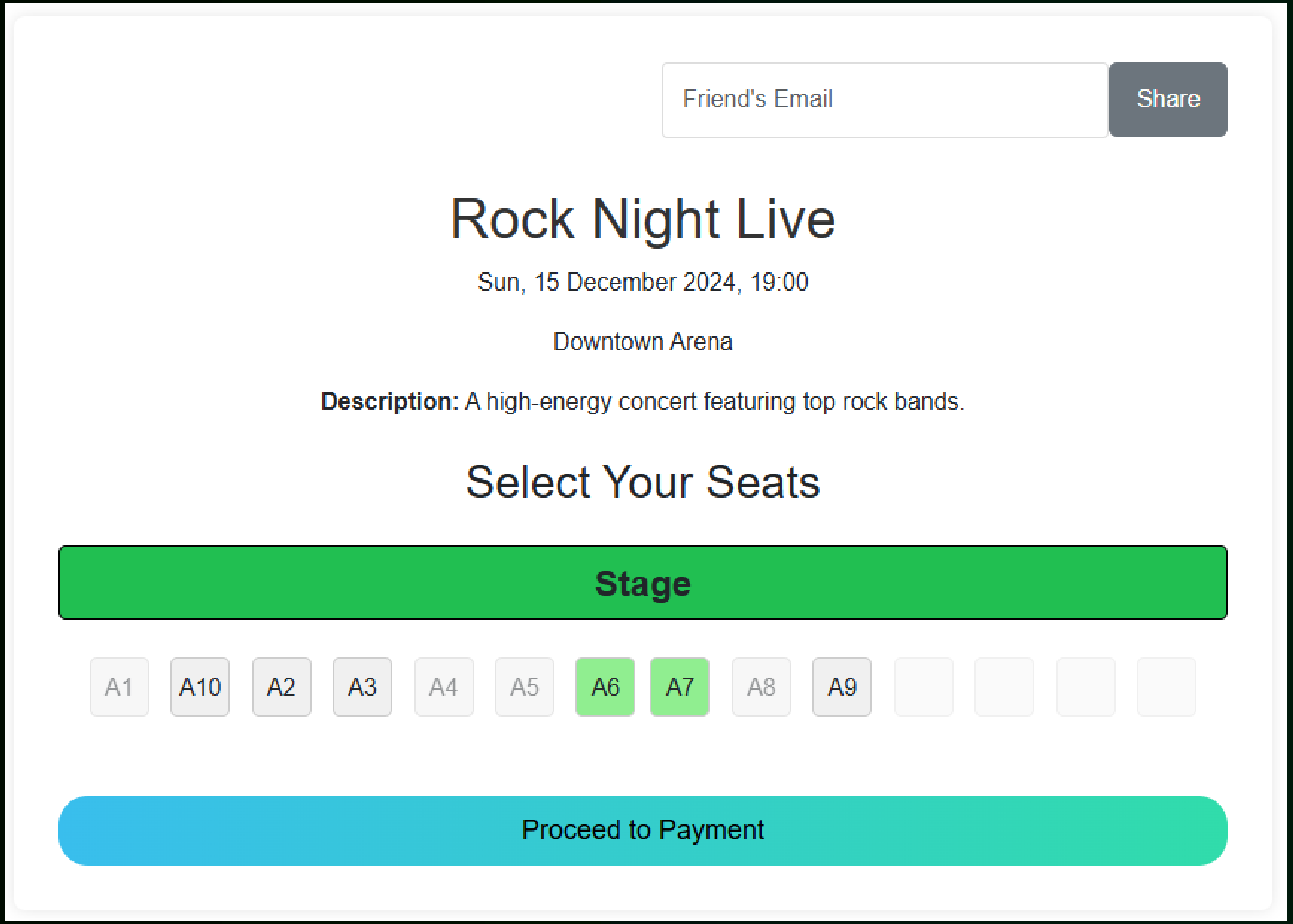
02



```python
# Query for future events (start date beyond the next 3 months)
future_events = db.session.query(Event).filter(
    Event.start_date > three_months_later
).order_by(Event.start_date.asc()).all()

# Identify events for promotional emails (booking start within the next month)
upcoming_events_for_promotion = db.session.query(Event).filter(
    Event.booking_open_time > now,
    Event.booking_open_time <= one_month_later
).all()
```

04 Nov 2024

Select & Send Event



Friend's Email

Share

Rock Night Live

Sun, 15 December 2024, 19:00

Downtown Arena

Description: A high-energy concert featuring top rock bands.

Select Your Seats

Stage

A1 A10 A2 A3 A4 A5 A6 A7 A8 A9

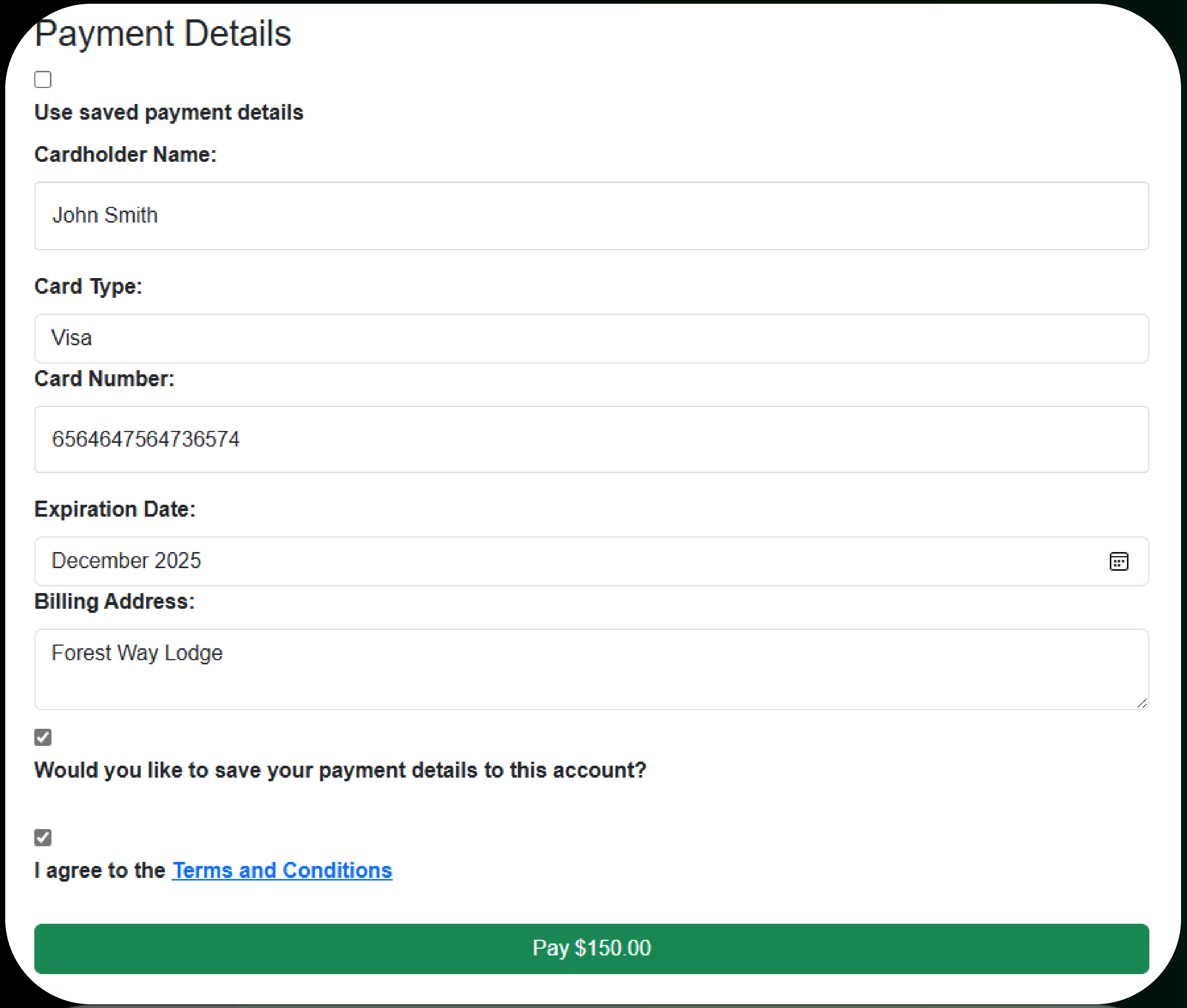
Proceed to Payment

01

02

04 Nov 2024

Secure Payment Process



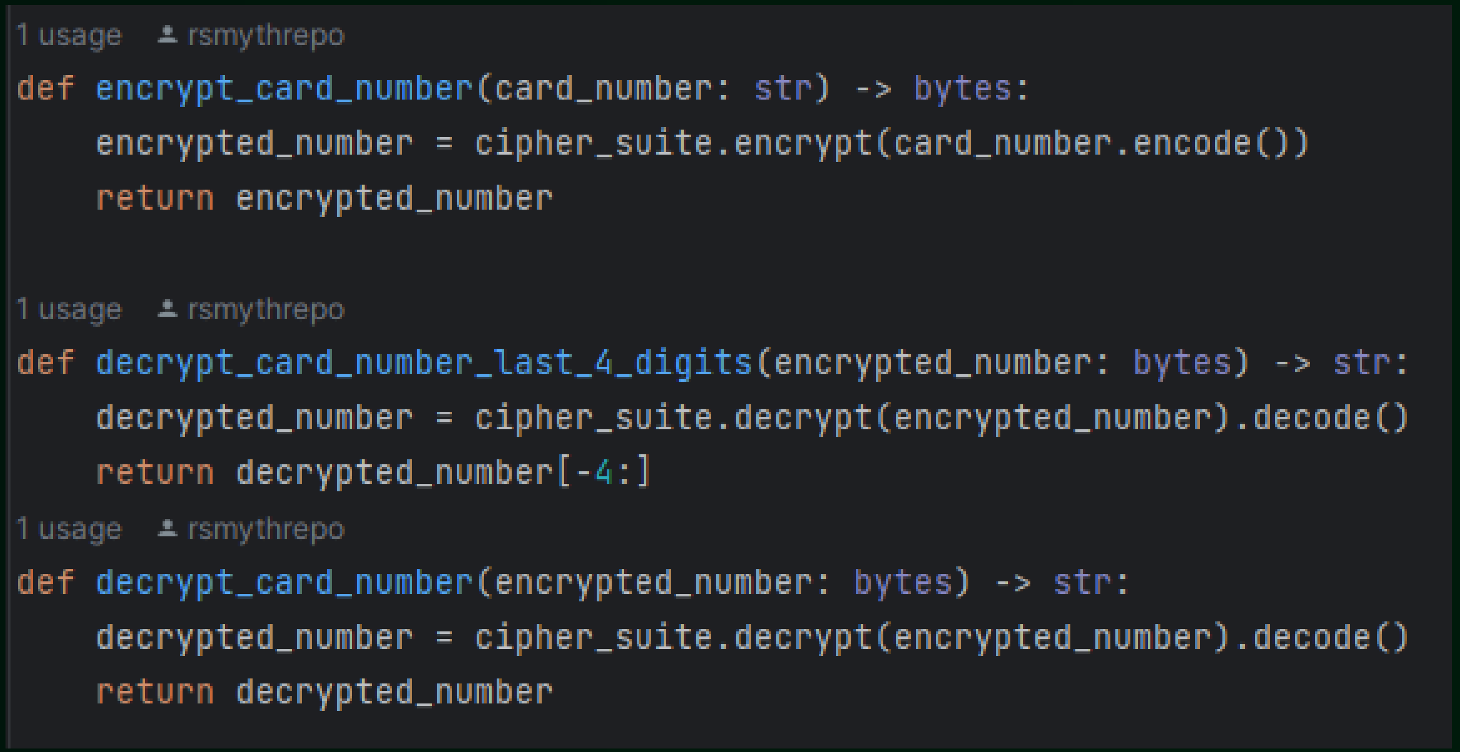
Payment Details

☐ Use saved payment details

Cardholder Name:	John Smith
Card Type:	Visa
Card Number:	6564647564736574
Expiration Date:	December 2025
Billing Address:	Forest Way Lodge

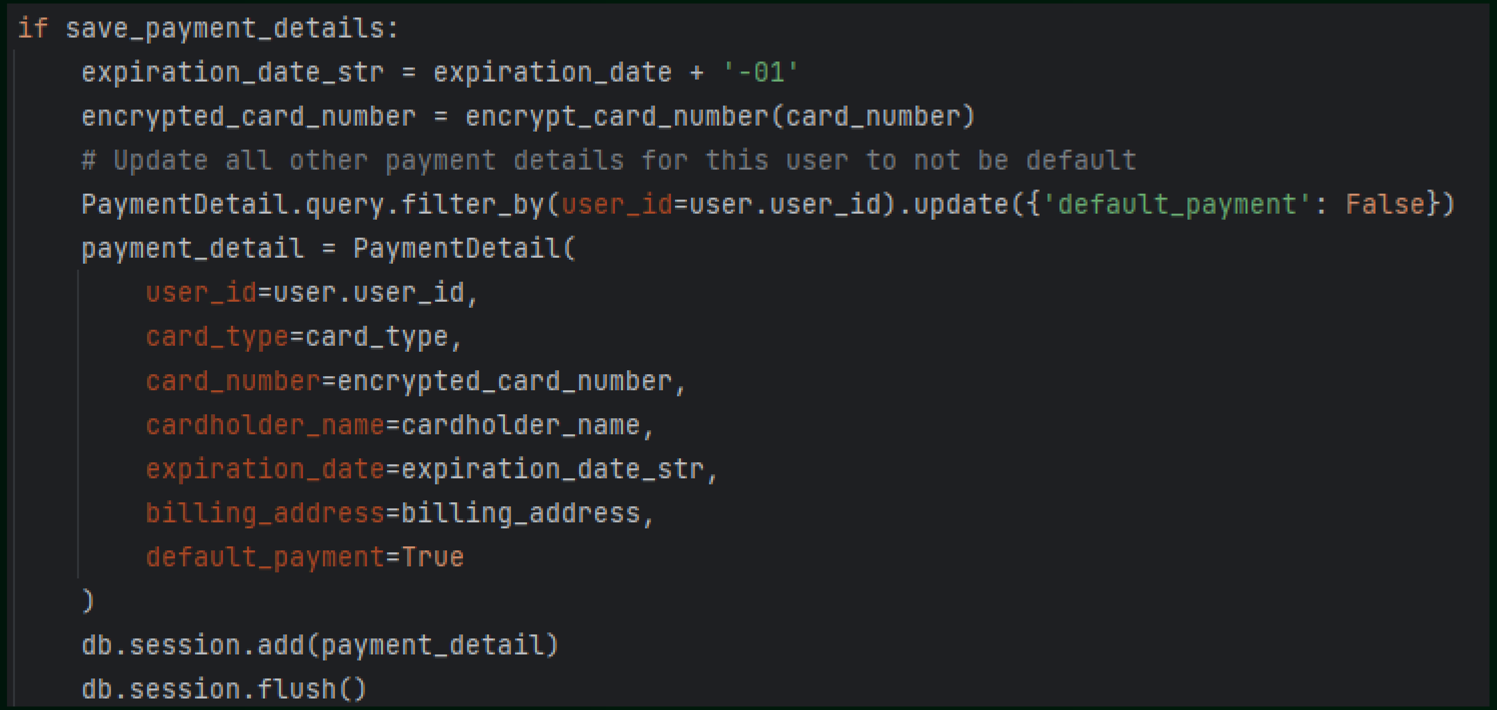
☑ Would you like to save your payment details to this account?

☑ I agree to the Terms and Conditions



```
1 usage  rsmythrepo
def encrypt_card_number(card_number: str) -> bytes:
    encrypted_number = cipher_suite.encrypt(card_number.encode())
    return encrypted_number

1 usage  rsmythrepo
def decrypt_card_number_last_4_digits(encrypted_number: bytes) -> str:
    decrypted_number = cipher_suite.decrypt(encrypted_number).decode()
    return decrypted_number[-4:]
1 usage  rsmythrepo
def decrypt_card_number(encrypted_number: bytes) -> str:
    decrypted_number = cipher_suite.decrypt(encrypted_number).decode()
    return decrypted_number
```

01

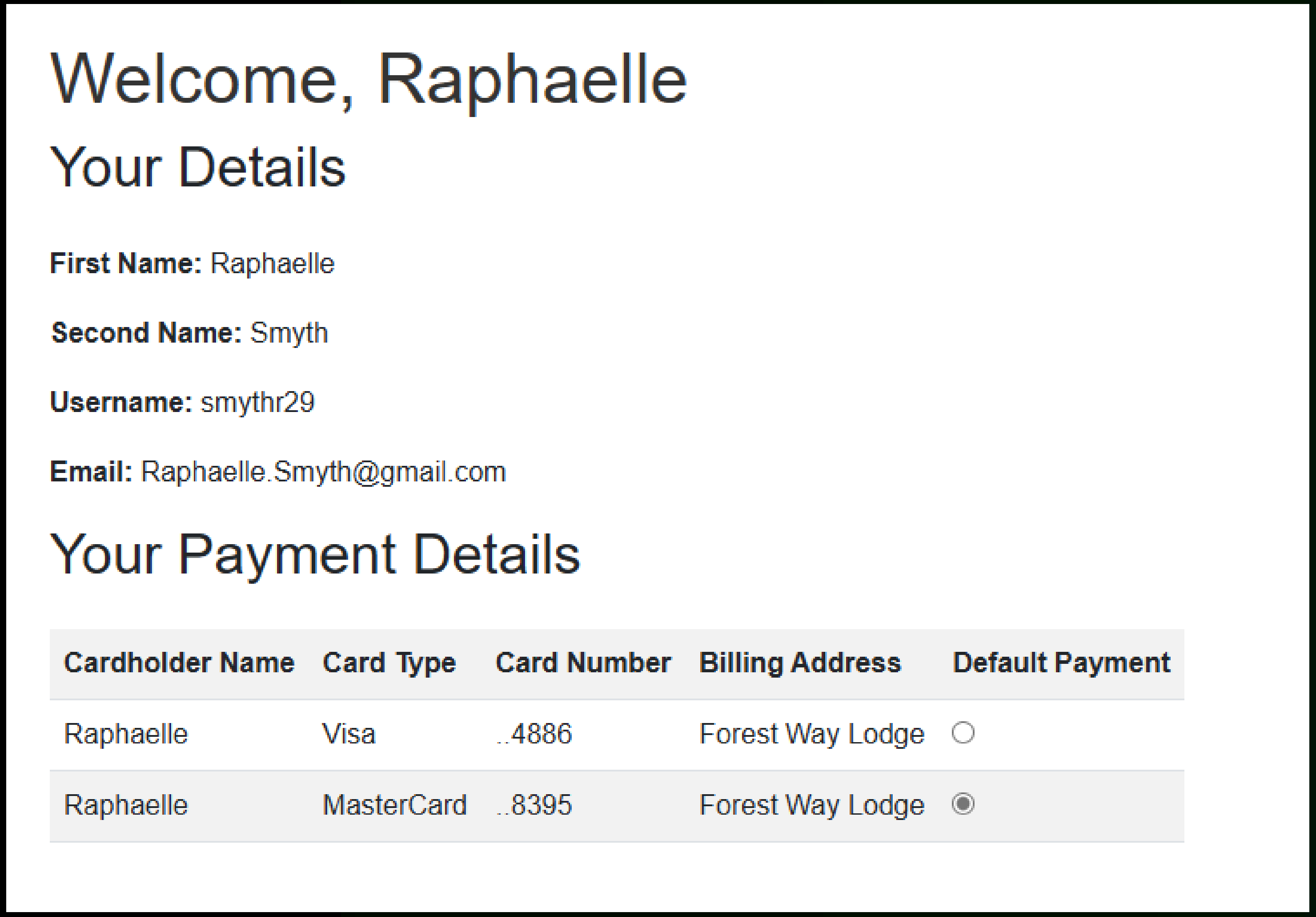


```
if save_payment_details:
    expiration_date_str = expiration_date + '-01'
    encrypted_card_number = encrypt_card_number(card_number)
    # Update all other payment details for this user to not be default
    PaymentDetail.query.filter_by(user_id=user.user_id).update({'default_payment': False})
    payment_detail = PaymentDetail(
        user_id=user.user_id,
        card_type=card_type,
        card_number=encrypted_card_number,
        cardholder_name=cardholder_name,
        expiration_date=expiration_date_str,
        billing_address=billing_address,
        default_payment=True
    )
    db.session.add(payment_detail)
    db.session.flush()
```

02

04 Nov 2024

Manage Profile



Welcome, Raphaelle

Your Details

First Name: Raphaelle

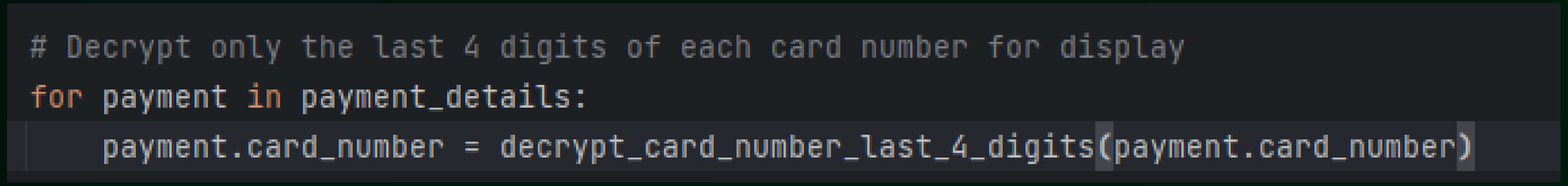
Second Name: Smyth

Username: smythr29

Email: Raphaelle.Smyth@gmail.com

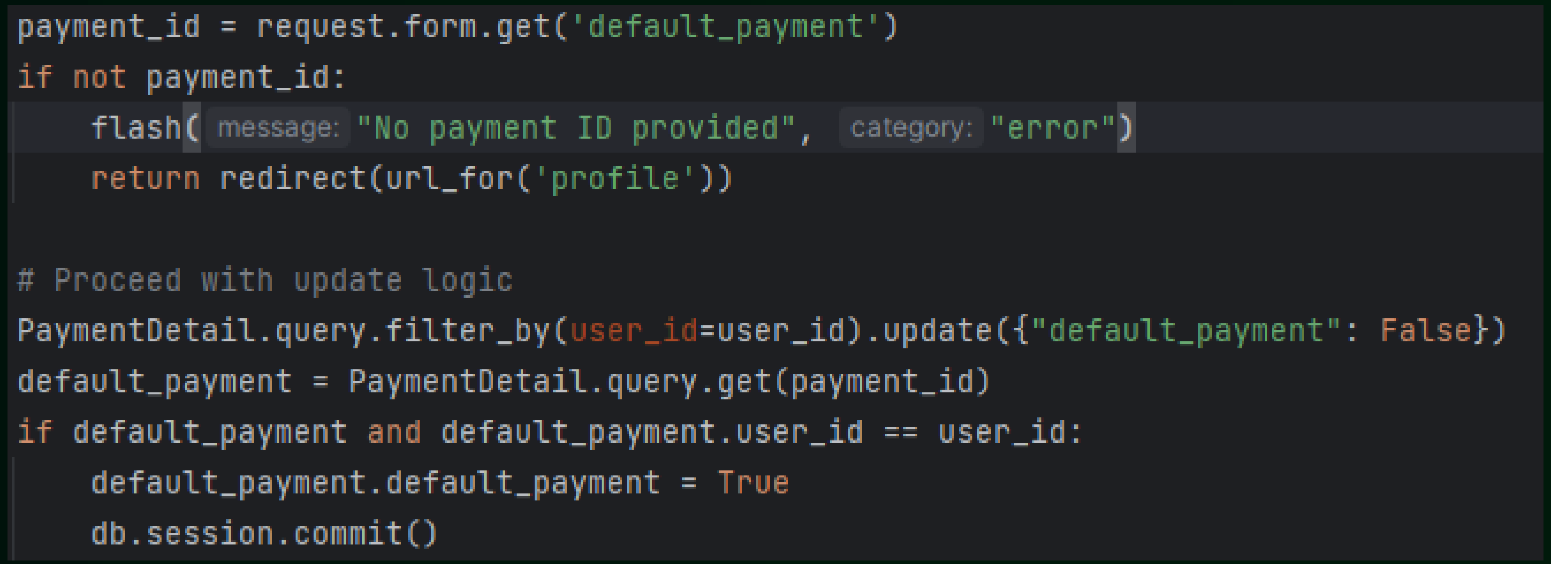
Your Payment Details

Cardholder Name	Card Type	Card Number	Billing Address	Default Payment
Raphaelle	Visa	..4886	Forest Way Lodge	○
Raphaelle	MasterCard	..8395	Forest Way Lodge	●



```
# Decrypt only the last 4 digits of each card number for display
for payment in payment_details:
    payment.card_number = decrypt_card_number_last_4_digits(payment.card_number)
```

01

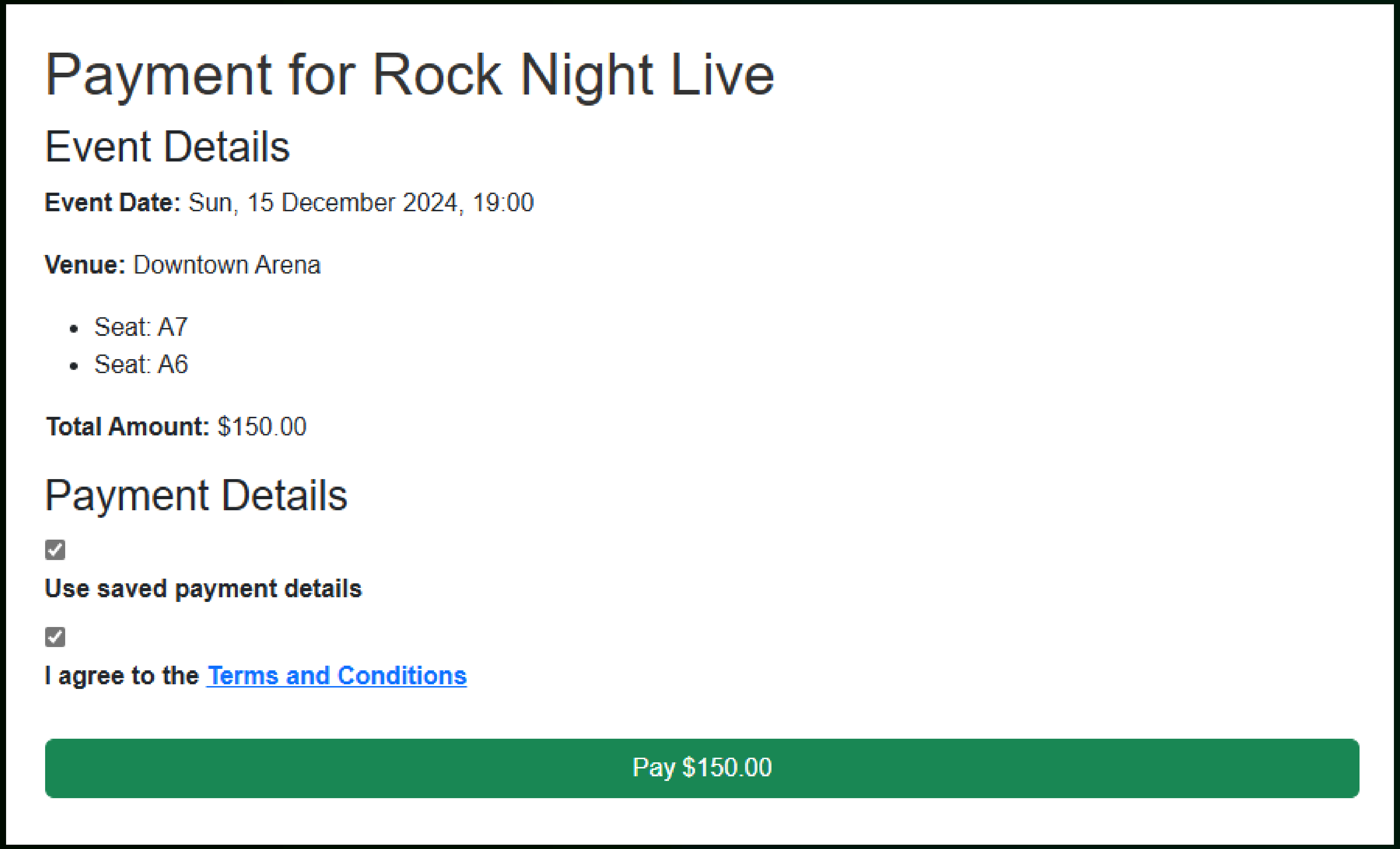


```
payment_id = request.form.get('default_payment')
if not payment_id:
    flash( message: "No payment ID provided",  category: "error")
    return redirect(url_for('profile'))

# Proceed with update logic
PaymentDetail.query.filter_by(user_id=user_id).update({"default_payment": False})
default_payment = PaymentDetail.query.get(payment_id)
if default_payment and default_payment.user_id == user_id:
    default_payment.default_payment = True
    db.session.commit()
```

02

04 Nov 2024

Use Default Payment



Payment for Rock Night Live

Event Details

Event Date: Sun, 15 December 2024, 19:00

Venue: Downtown Arena

- Seat: A7
- Seat: A6

Total Amount: $150.00

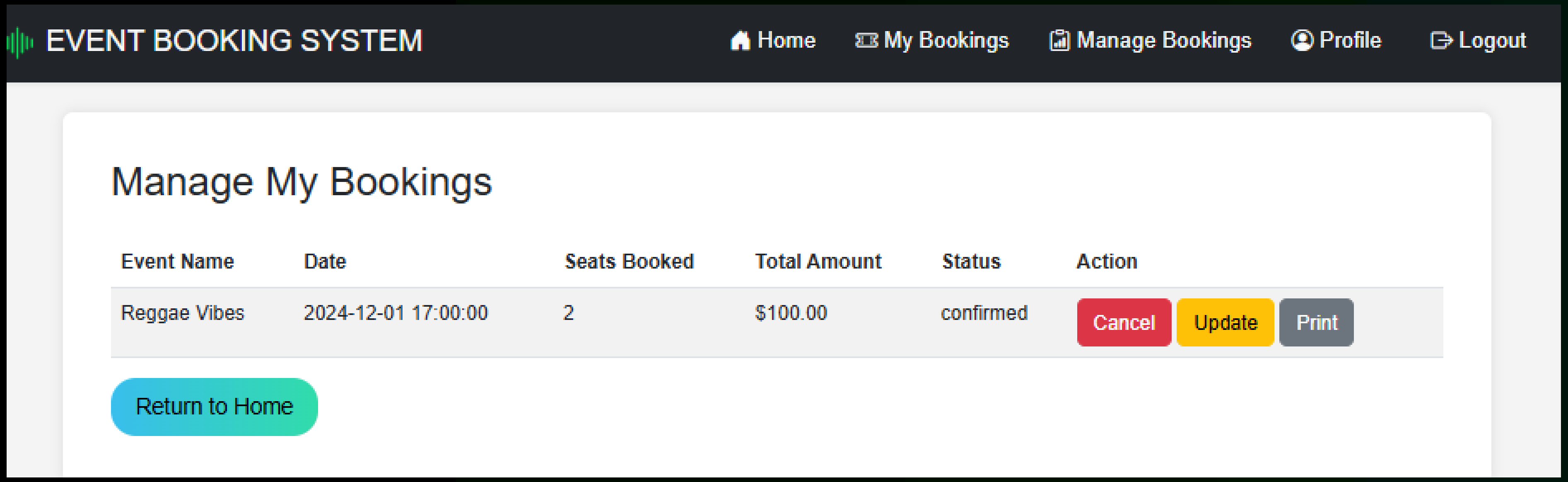
Payment Details

☑ **Use saved payment details**

☑ **I agree to the Terms and Conditions**

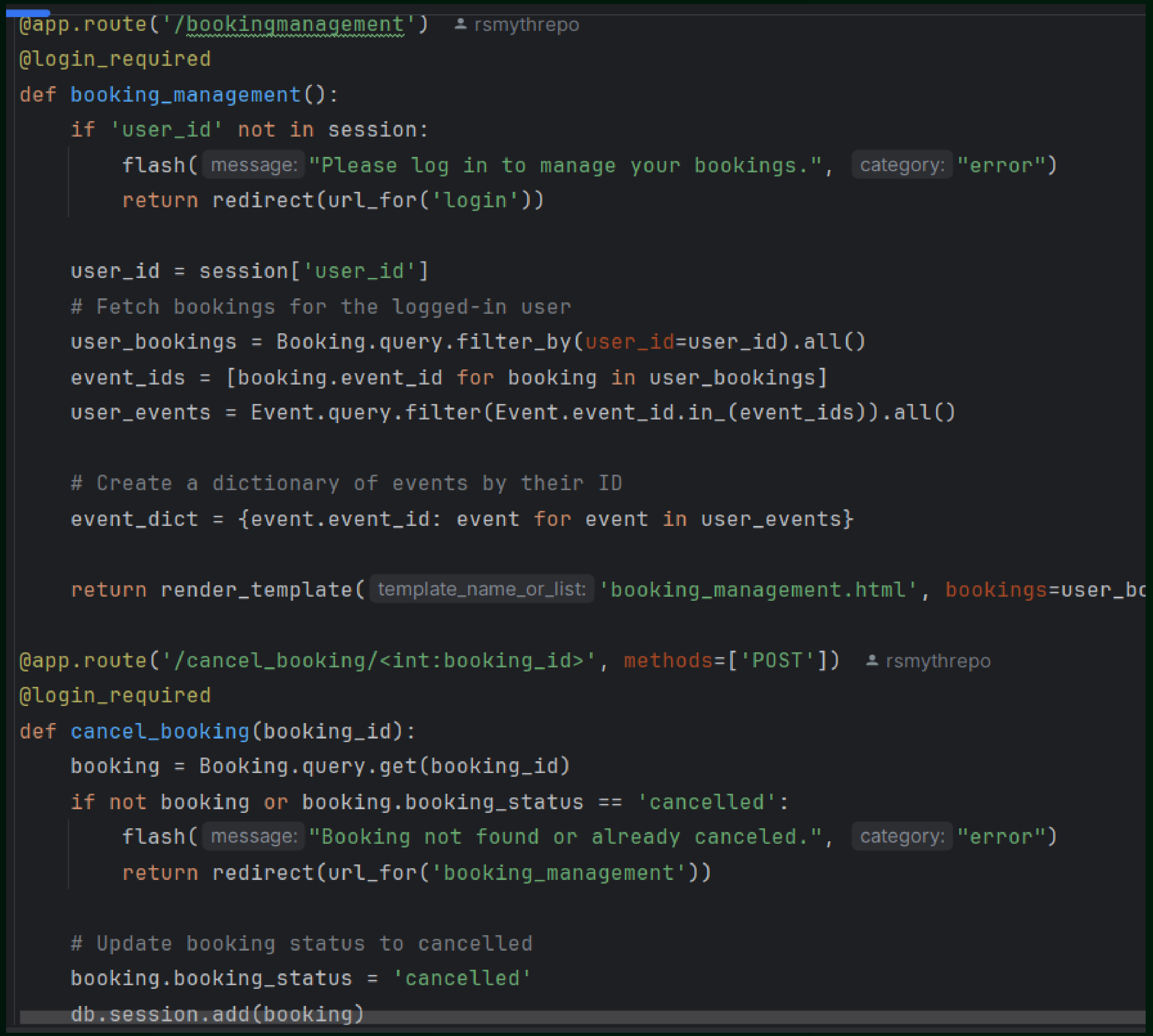
04 Nov 2024

Manage My Bookings



Manage My Bookings

Event Name	Date	Seats Booked	Total Amount	Status	Action
Reggae Vibes	2024-12-01 17:00:00	2	$100.00	confirmed	Cancel Update Print



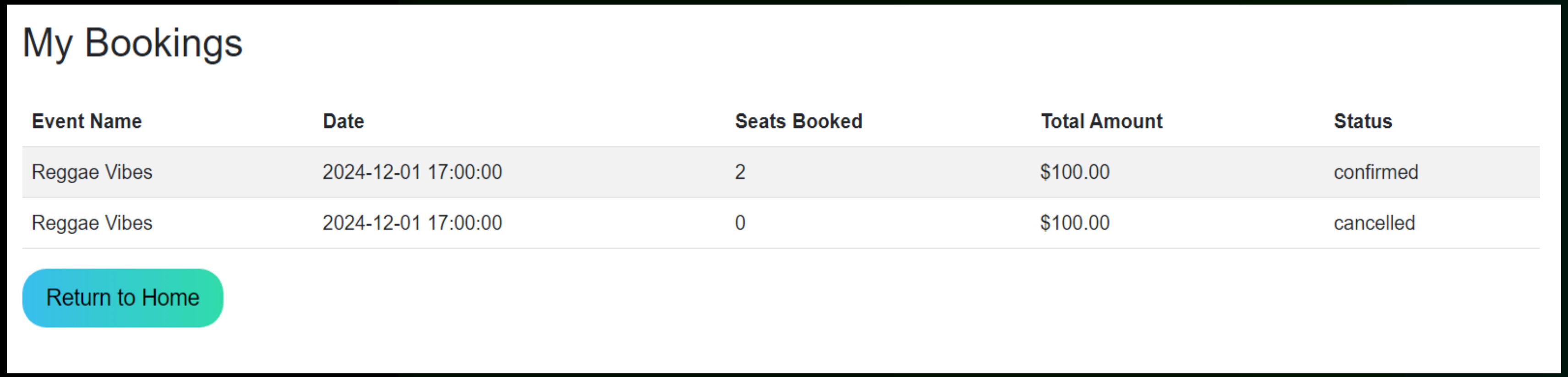
```
@app.route('/bookingmanagement')  rsmythrepo
@login_required
def booking_management():
    if 'user_id' not in session:
        flash( message: "Please log in to manage your bookings.",  category: "error")
        return redirect(url_for('login'))

    user_id = session['user_id']
    # Fetch bookings for the logged-in user
    user_bookings = Booking.query.filter_by(user_id=user_id).all()
    event_ids = [booking.event_id for booking in user_bookings]
    user_events = Event.query.filter(Event.event_id.in_(event_ids)).all()

    # Create a dictionary of events by their ID
    event_dict = {event.event_id: event for event in user_events}

    return render_template( template_name_or_list: 'booking_management.html', bookings=user_bo

@app.route('/cancel_booking/<int:booking_id>', methods=['POST'])  rsmythrepo
@login_required
def cancel_booking(booking_id):
    booking = Booking.query.get(booking_id)
    if not booking or booking.booking_status == 'cancelled':
        flash( message: "Booking not found or already canceled.",  category: "error")
        return redirect(url_for('booking_management'))

    # Update booking status to cancelled
    booking.booking_status = 'cancelled'
    db.session.add(booking)
```

01

02

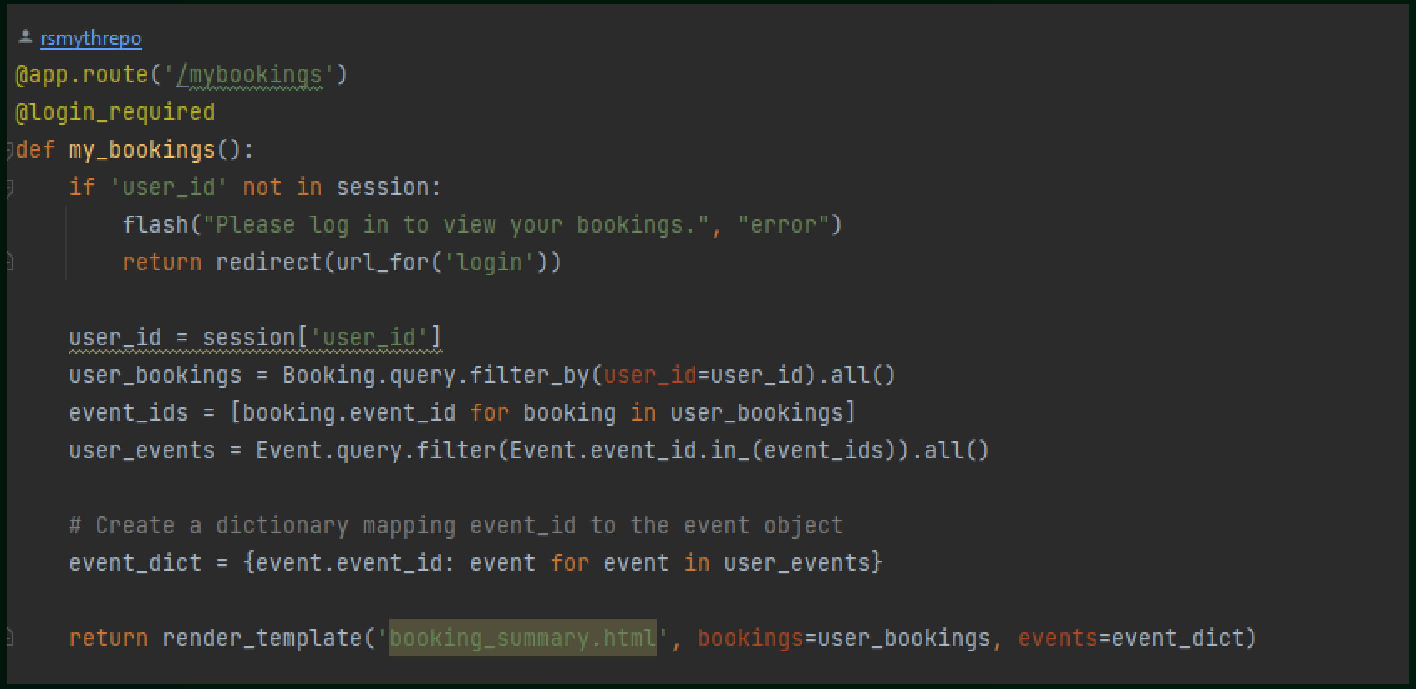
04 Nov 2024

View my bookings



My Bookings

Event Name	Date	Seats Booked	Total Amount	Status
Reggae Vibes	2024-12-01 17:00:00	2	$100.00	confirmed
Reggae Vibes	2024-12-01 17:00:00	0	$100.00	cancelled



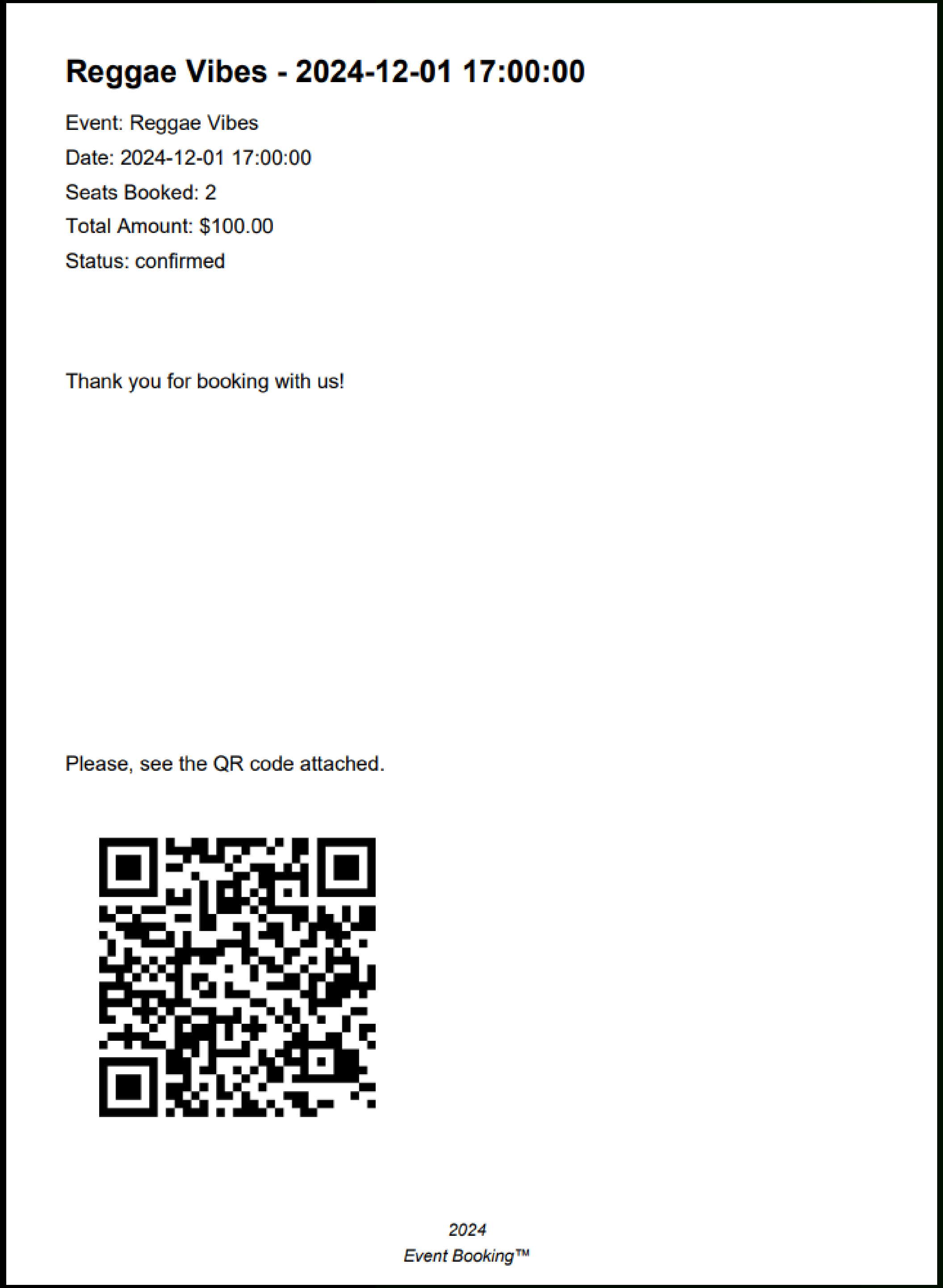
```
rsmythrepo
@app.route('/mybookings')
@login_required
def my_bookings():
    if 'user_id' not in session:
        flash("Please log in to view your bookings.", "error")
        return redirect(url_for('login'))

    user_id = session['user_id']
    user_bookings = Booking.query.filter_by(user_id=user_id).all()
    event_ids = [booking.event_id for booking in user_bookings]
    user_events = Event.query.filter(Event.event_id.in_(event_ids)).all()

    # Create a dictionary mapping event_id to the event object
    event_dict = {event.event_id: event for event in user_events}

    return render_template('booking_summary.html', bookings=user_bookings, events=event_dict)
```

01

02

04 Nov 2024

Ticket printing



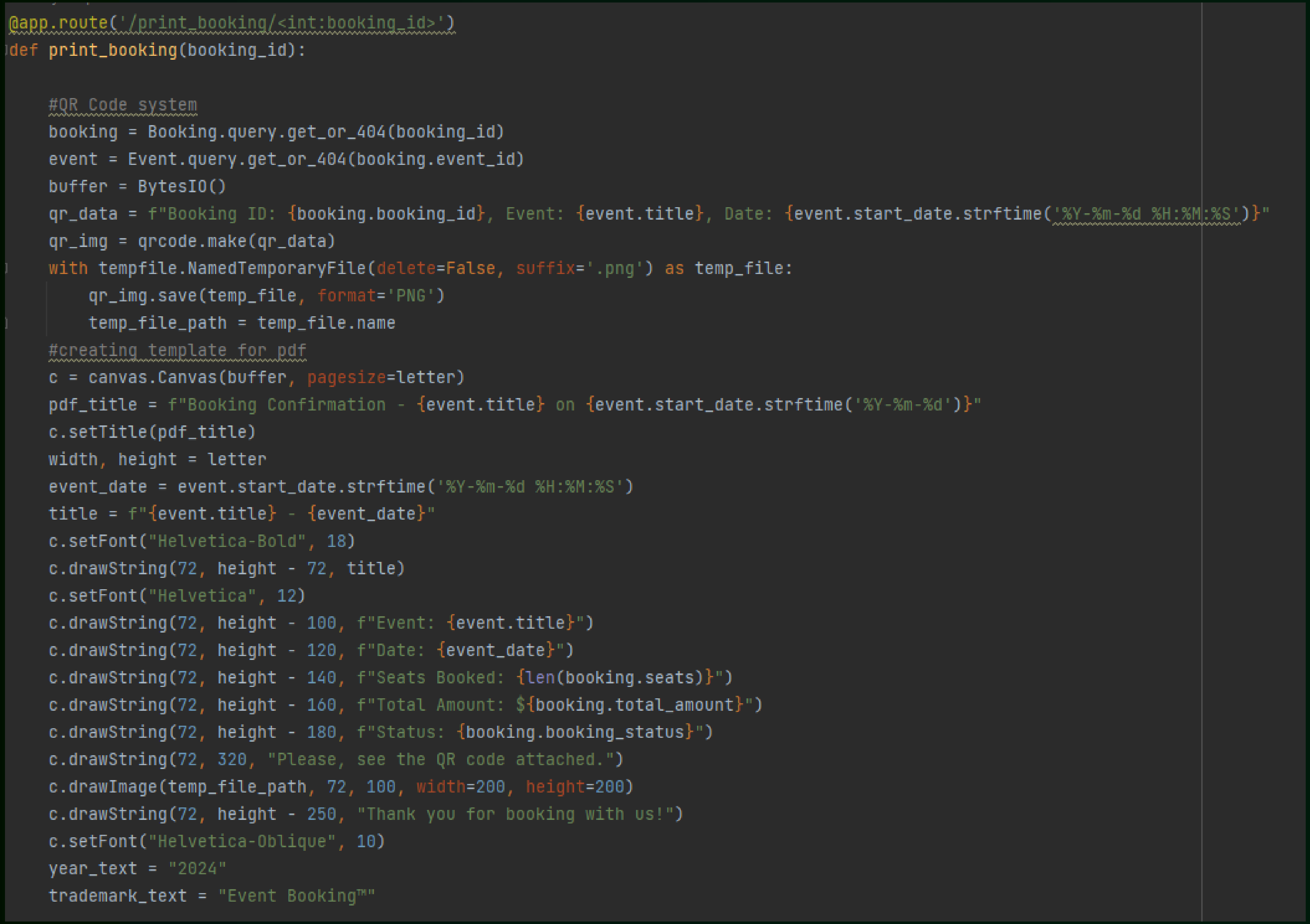
Reggae Vibes - 2024-12-01 17:00:00

Event: Reggae Vibes
Date: 2024-12-01 17:00:00
Seats Booked: 2
Total Amount: $100.00
Status: confirmed

Thank you for booking with us!

Please, see the QR code attached.

2024
Event Booking™



```python
@app.route('/print_booking/<int:booking_id>')
def print_booking(booking_id):

    #QR Code system
    booking = Booking.query.get_or_404(booking_id)
    event = Event.query.get_or_404(booking.event_id)
    buffer = BytesIO()
    qr_data = f"Booking ID: {booking.booking_id}, Event: {event.title}, Date: {event.start_date.strftime('%Y-%m-%d %H:%M:%S')}"
    qr_img = qrcode.make(qr_data)
    with tempfile.NamedTemporaryFile(delete=False, suffix='.png') as temp_file:
        qr_img.save(temp_file, format='PNG')
        temp_file_path = temp_file.name
    #creating template for pdf
    c = canvas.Canvas(buffer, pagesize=letter)
    pdf_title = f"Booking Confirmation - {event.title} on {event.start_date.strftime('%Y-%m-%d')}"
    c.setTitle(pdf_title)
    width, height = letter
    event_date = event.start_date.strftime('%Y-%m-%d %H:%M:%S')
    title = f"{event.title} - {event_date}"
    c.setFont("Helvetica-Bold", 18)
    c.drawString(72, height - 72, title)
    c.setFont("Helvetica", 12)
    c.drawString(72, height - 100, f"Event: {event.title}")
    c.drawString(72, height - 120, f"Date: {event_date}")
    c.drawString(72, height - 140, f"Seats Booked: {len(booking.seats)}")
    c.drawString(72, height - 160, f"Total Amount: ${booking.total_amount}")
    c.drawString(72, height - 180, f"Status: {booking.booking_status}")
    c.drawString(72, 320, "Please, see the QR code attached.")
    c.drawImage(temp_file_path, 72, 100, width=200, height=200)
    c.drawString(72, height - 250, "Thank you for booking with us!")
    c.setFont("Helvetica-Oblique", 10)
    year_text = "2024"
    trademark_text = "Event Booking™"
```

04 Nov 2024

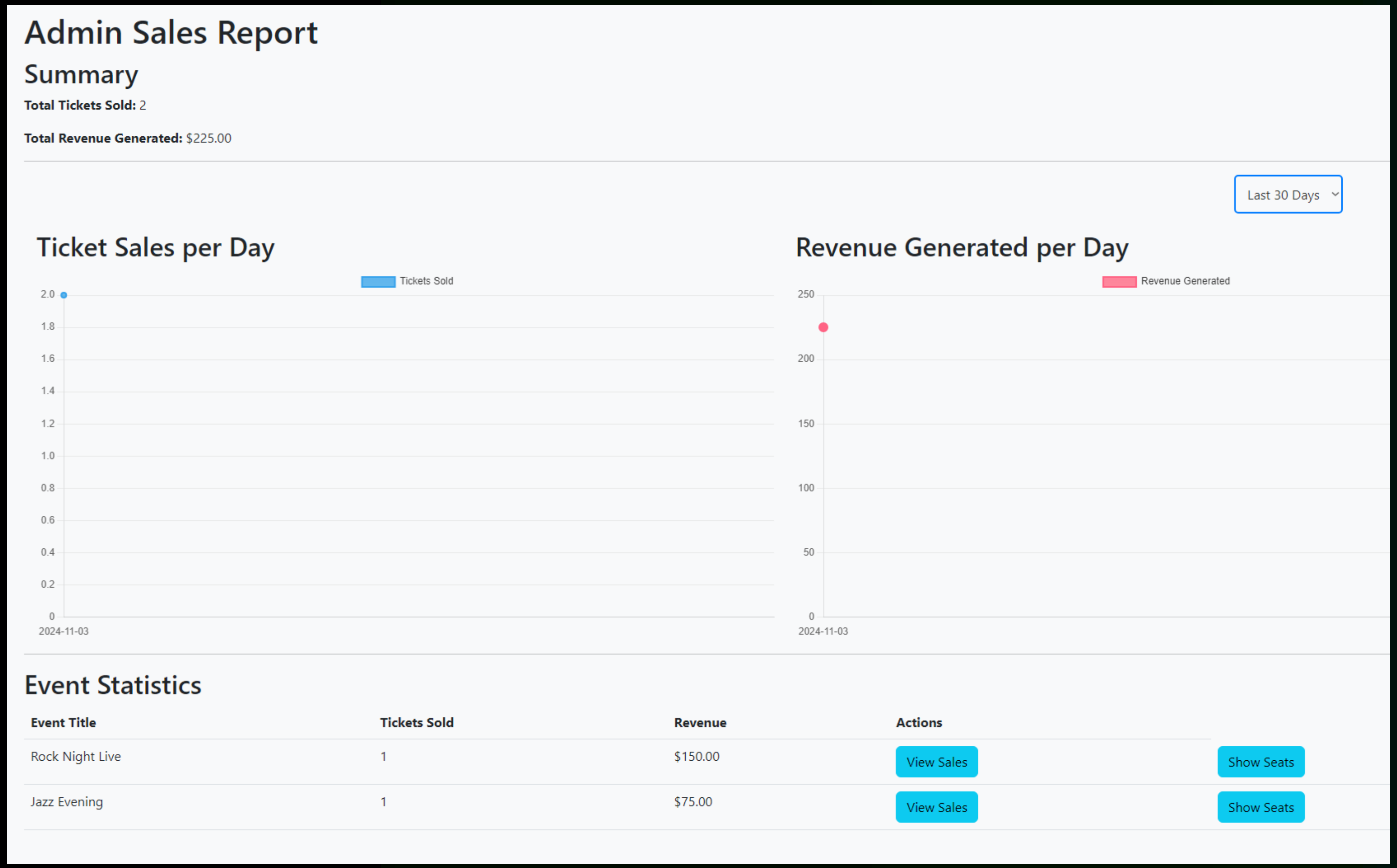
01

02

Admin Features

04 Nov 2024

Sales report



Admin Sales Report

Summary

Total Tickets Sold: 2

Total Revenue Generated: $225.00

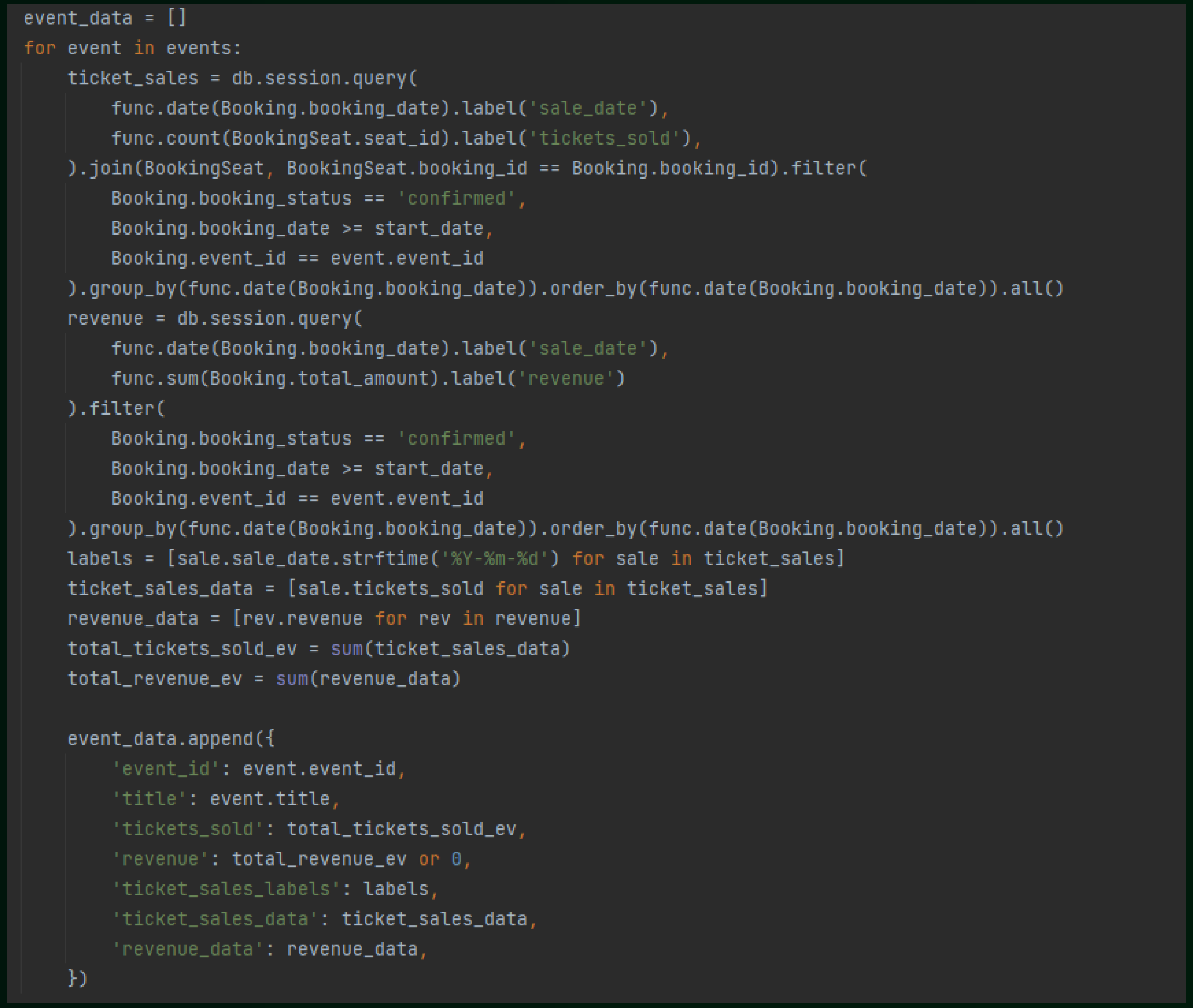
Last 30 Days

Ticket Sales per Day

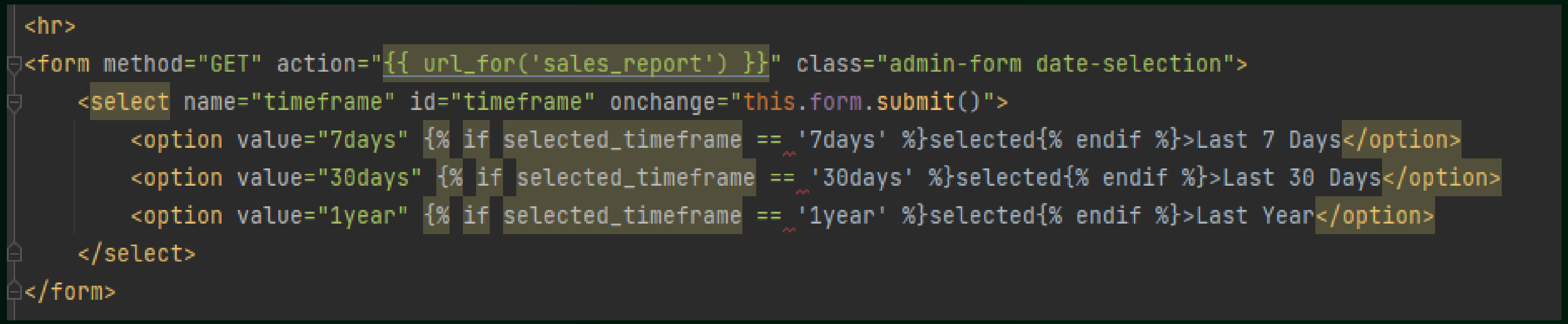
Revenue Generated per Day

Event Statistics

Event Title	Tickets Sold	Revenue	Actions	
Rock Night Live	1	$150.00	View Sales	Show Seats
Jazz Evening	1	$75.00	View Sales	Show Seats



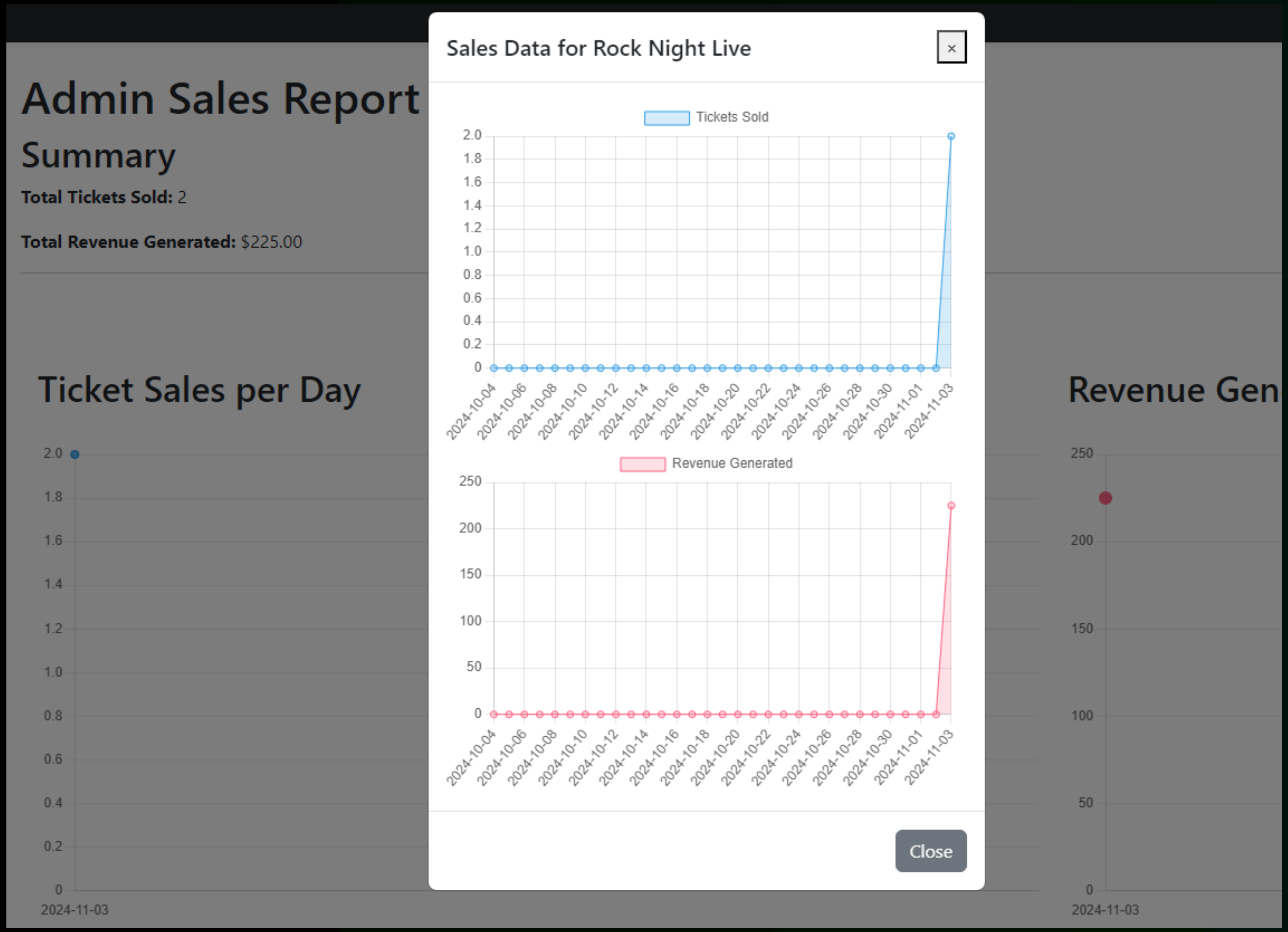
```
event_data = []
for event in events:
    ticket_sales = db.session.query(
        func.date(Booking.booking_date).label('sale_date'),
        func.count(BookingSeat.seat_id).label('tickets_sold'),
    ).join(BookingSeat, BookingSeat.booking_id == Booking.booking_id).filter(
        Booking.booking_status == 'confirmed',
        Booking.booking_date >= start_date,
        Booking.event_id == event.event_id
    ).group_by(func.date(Booking.booking_date)).order_by(func.date(Booking.booking_date)).all()
    revenue = db.session.query(
        func.date(Booking.booking_date).label('sale_date'),
        func.sum(Booking.total_amount).label('revenue')
    ).filter(
        Booking.booking_status == 'confirmed',
        Booking.booking_date >= start_date,
        Booking.event_id == event.event_id
    ).group_by(func.date(Booking.booking_date)).order_by(func.date(Booking.booking_date)).all()
    labels = [sale.sale_date.strftime('%Y-%m-%d') for sale in ticket_sales]
    ticket_sales_data = [sale.tickets_sold for sale in ticket_sales]
    revenue_data = [rev.revenue for rev in revenue]
    total_tickets_sold_ev = sum(ticket_sales_data)
    total_revenue_ev = sum(revenue_data)

    event_data.append({
        'event_id': event.event_id,
        'title': event.title,
        'tickets_sold': total_tickets_sold_ev,
        'revenue': total_revenue_ev or 0,
        'ticket_sales_labels': labels,
        'ticket_sales_data': ticket_sales_data,
        'revenue_data': revenue_data,
    })
```

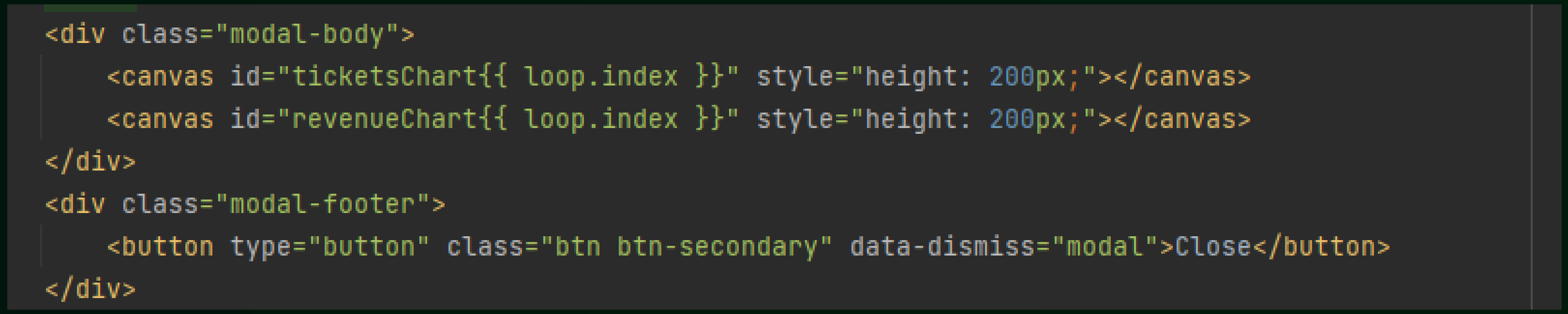


```
<hr>
<form method="GET" action="{{ url_for('sales_report') }}" class="admin-form date-selection">
    <select name="timeframe" id="timeframe" onchange="this.form.submit()">
        <option value="7days" {% if selected_timeframe == '7days' %}selected{% endif %}>Last 7 Days</option>
        <option value="30days" {% if selected_timeframe == '30days' %}selected{% endif %}>Last 30 Days</option>
        <option value="1year" {% if selected_timeframe == '1year' %}selected{% endif %}>Last Year</option>
    </select>
</form>
```

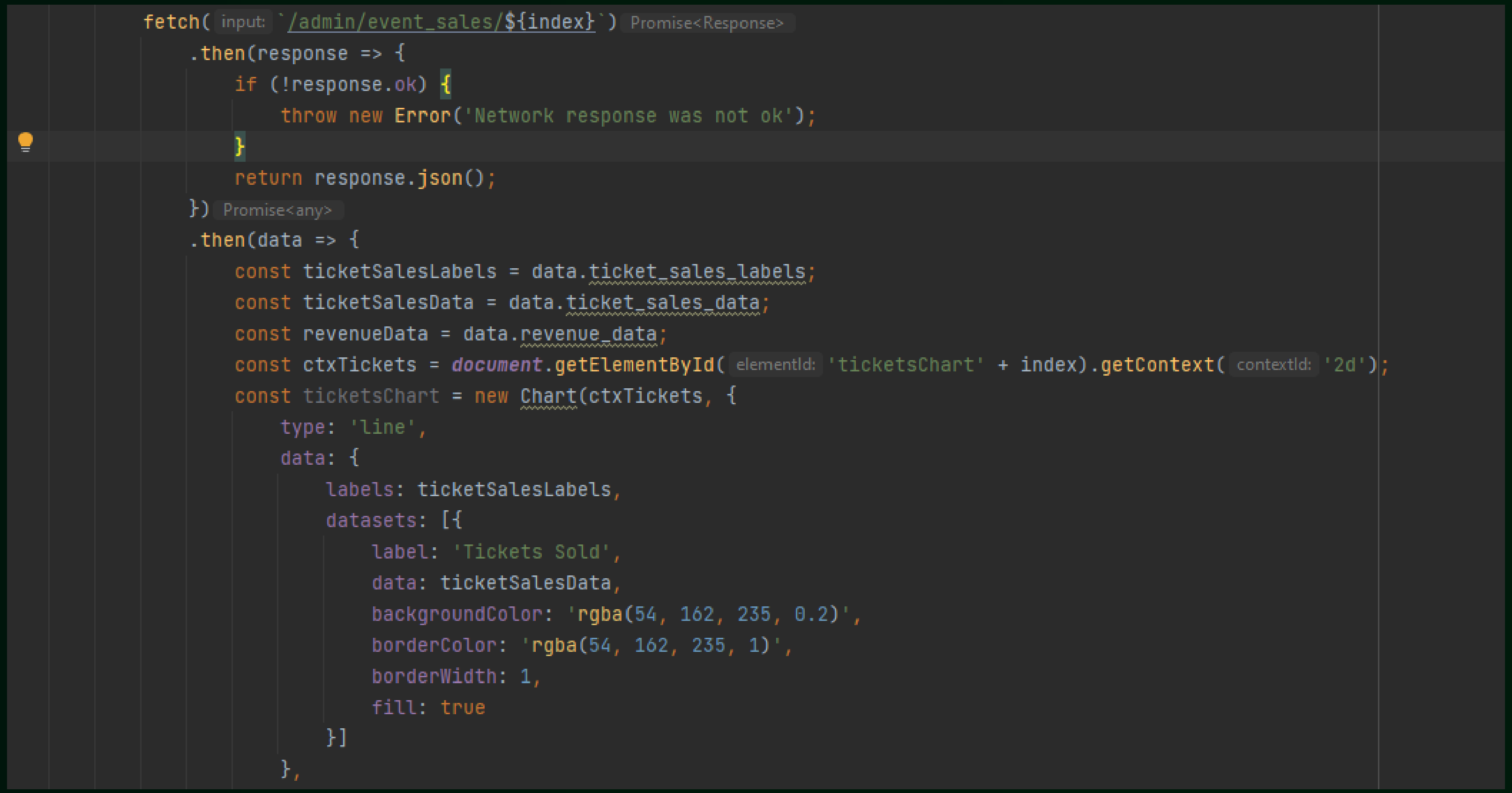
04 Nov 2024

Sales report





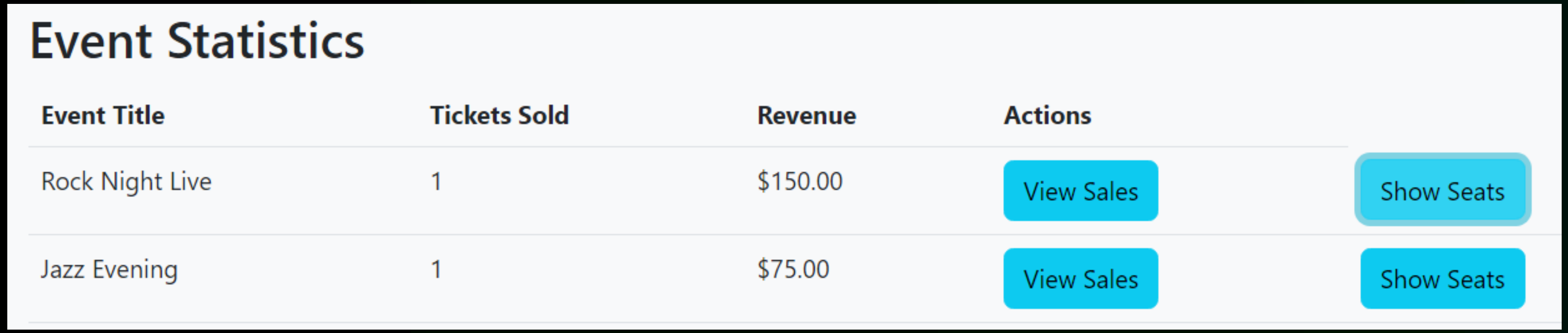
```html
<div class="modal-body">
    <canvas id="ticketsChart{{ loop.index }}" style="height: 200px;"></canvas>
    <canvas id="revenueChart{{ loop.index }}" style="height: 200px;"></canvas>
</div>
<div class="modal-footer">
    <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
</div>
```



```javascript
fetch(`/admin/event_sales/${index}`)
    .then(response => {
        if (!response.ok) {
            throw new Error('Network response was not ok');
        }
        return response.json();
    })
    .then(data => {
        const ticketSalesLabels = data.ticket_sales_labels;
        const ticketSalesData = data.ticket_sales_data;
        const revenueData = data.revenue_data;
        const ctxTickets = document.getElementById('ticketsChart' + index).getContext('2d');
        const ticketsChart = new Chart(ctxTickets, {
            type: 'line',
            data: {
                labels: ticketSalesLabels,
                datasets: [{
                    label: 'Tickets Sold',
                    data: ticketSalesData,
                    backgroundColor: 'rgba(54, 162, 235, 0.2)',
                    borderColor: 'rgba(54, 162, 235, 1)',
                    borderWidth: 1,
                    fill: true
                }]
            },
```

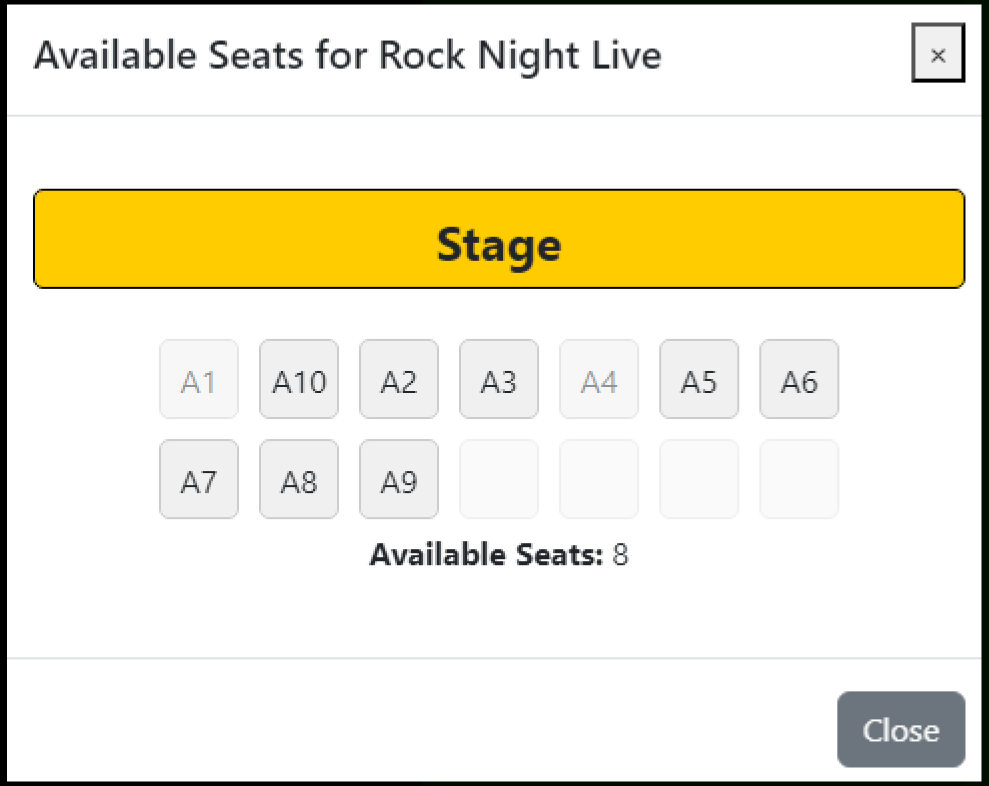
04 Nov 2024

Seat monitoring



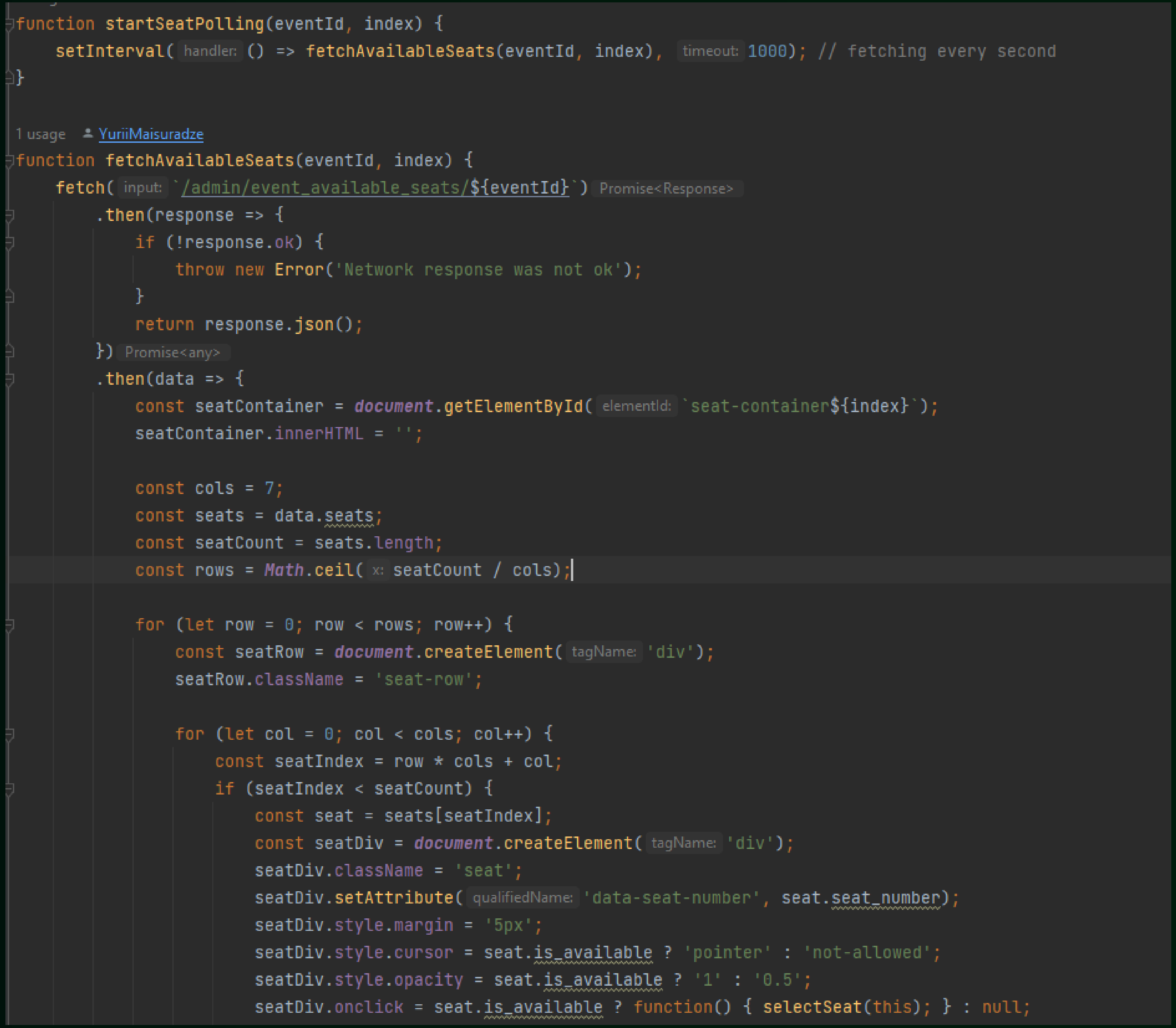
Event Statistics

Event Title	Tickets Sold	Revenue	Actions	
Rock Night Live	1	$150.00	View Sales	Show Seats
Jazz Evening	1	$75.00	View Sales	Show Seats



Available Seats for Rock Night Live

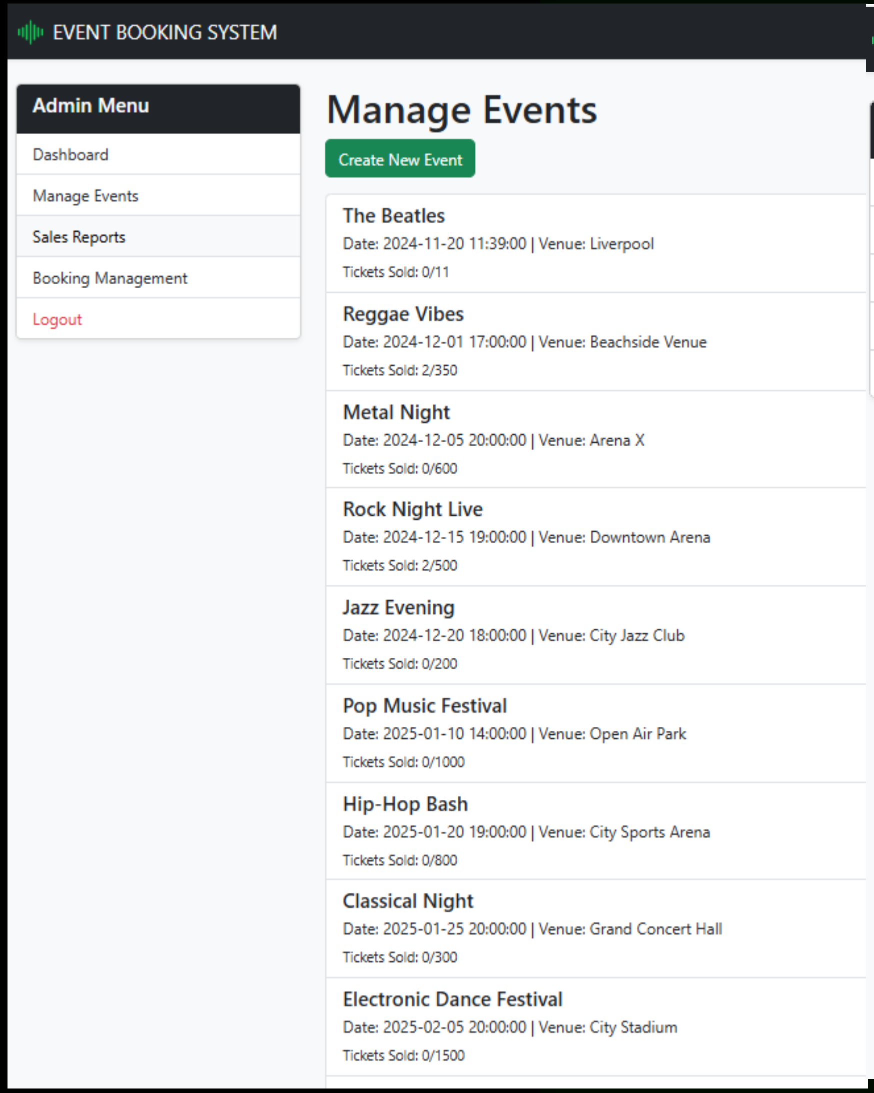
Available Seats: 8



```
function startSeatPolling(eventId, index) {
    setInterval( handler: () => fetchAvailableSeats(eventId, index),  timeout: 1000); // fetching every second
}

1 usage  YuriiMaisuradze
function fetchAvailableSeats(eventId, index) {
    fetch( input: `/admin/event_available_seats/${eventId}`) Promise<Response>
        .then(response => {
            if (!response.ok) {
                throw new Error('Network response was not ok');
            }
            return response.json();
        }) Promise<any>
        .then(data => {
            const seatContainer = document.getElementById( elementId: `seat-container${index}`);
            seatContainer.innerHTML = '';

            const cols = 7;
            const seats = data.seats;
            const seatCount = seats.length;
            const rows = Math.ceil( x: seatCount / cols);

            for (let row = 0; row < rows; row++) {
                const seatRow = document.createElement( tagName: 'div');
                seatRow.className = 'seat-row';

                for (let col = 0; col < cols; col++) {
                    const seatIndex = row * cols + col;
                    if (seatIndex < seatCount) {
                        const seat = seats[seatIndex];
                        const seatDiv = document.createElement( tagName: 'div');
                        seatDiv.className = 'seat';
                        seatDiv.setAttribute( qualifiedName: 'data-seat-number', seat.seat_number);
                        seatDiv.style.margin = '5px';
                        seatDiv.style.cursor = seat.is_available ? 'pointer' : 'not-allowed';
                        seatDiv.style.opacity = seat.is_available ? '1' : '0.5';
                        seatDiv.onclick = seat.is_available ? function() { selectSeat(this); } : null;
```

04 Nov 2024

Manage Events



EVENT BOOKING SYSTEM

Admin Menu

- Dashboard
- Manage Events
- Sales Reports
- Booking Management
- Logout

Manage Events

Create New Event

The Beatles
Date: 2024-11-20 11:39:00 | Venue: Liverpool
Tickets Sold: 0/11

Reggae Vibes
Date: 2024-12-01 17:00:00 | Venue: Beachside Venue
Tickets Sold: 2/350

Metal Night
Date: 2024-12-05 20:00:00 | Venue: Arena X
Tickets Sold: 0/600

Rock Night Live
Date: 2024-12-15 19:00:00 | Venue: Downtown Arena
Tickets Sold: 2/500

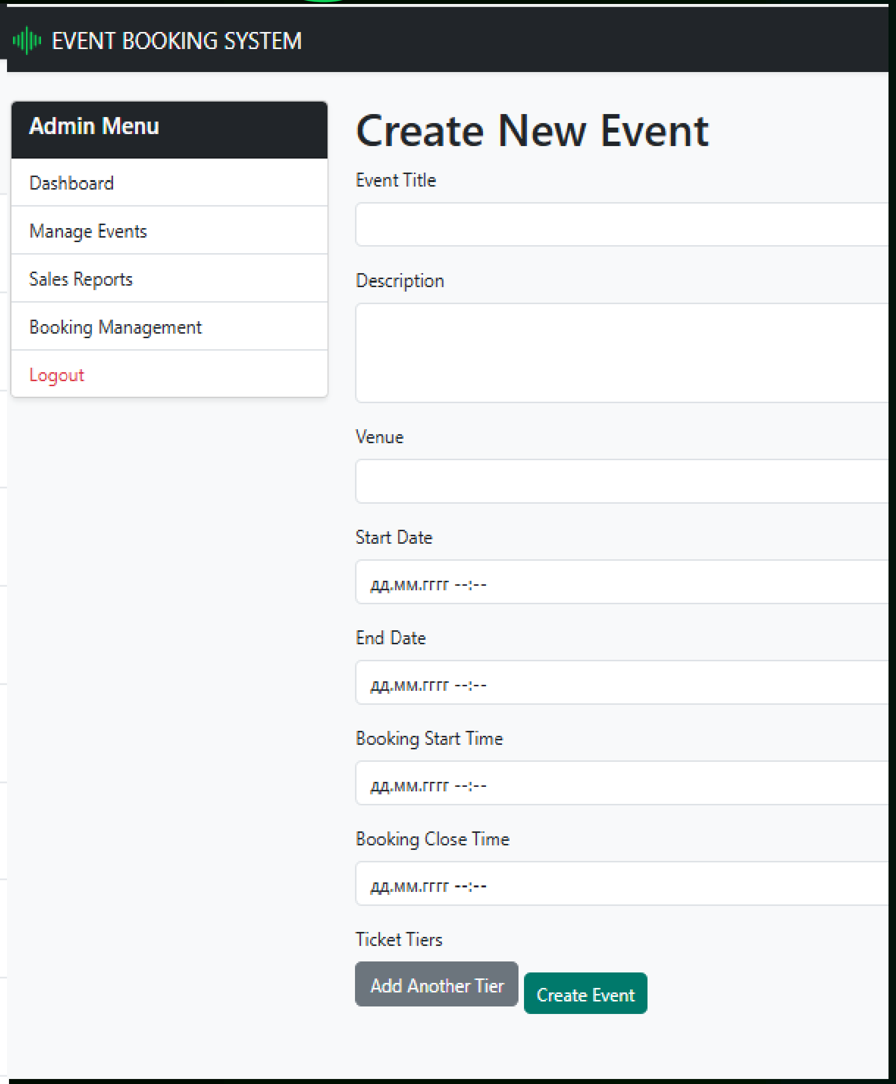
Jazz Evening
Date: 2024-12-20 18:00:00 | Venue: City Jazz Club
Tickets Sold: 0/200

Pop Music Festival
Date: 2025-01-10 14:00:00 | Venue: Open Air Park
Tickets Sold: 0/1000

Hip-Hop Bash
Date: 2025-01-20 19:00:00 | Venue: City Sports Arena
Tickets Sold: 0/800

Classical Night
Date: 2025-01-25 20:00:00 | Venue: Grand Concert Hall
Tickets Sold: 0/300

Electronic Dance Festival
Date: 2025-02-05 20:00:00 | Venue: City Stadium
Tickets Sold: 0/1500



EVENT BOOKING SYSTEM

Admin Menu

- Dashboard
- Manage Events
- Sales Reports
- Booking Management
- Logout

Create New Event

Event Title

Description

Venue

Start Date
дд.мм.гггг --:--

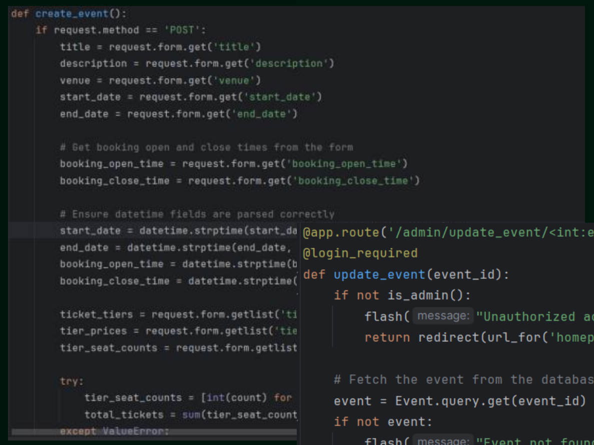
End Date
дд.мм.гггг --:--

Booking Start Time
дд.мм.гггг --:--

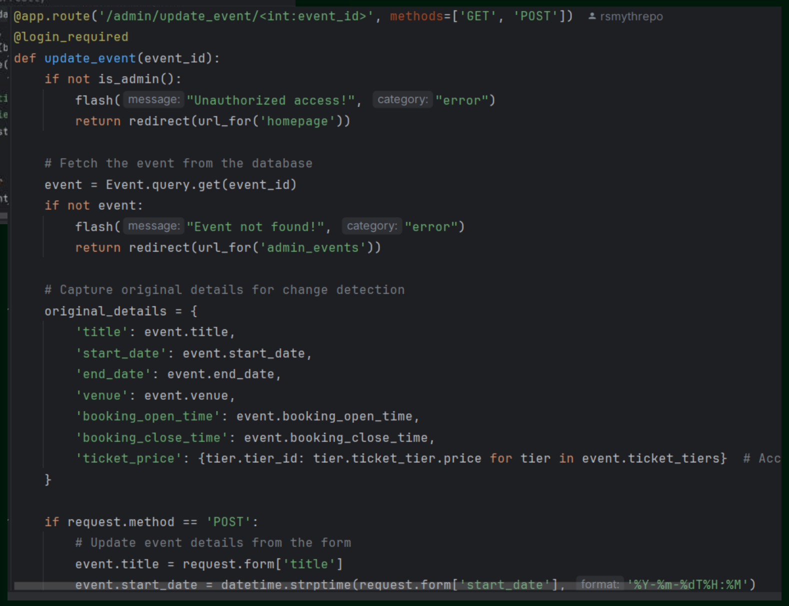
Booking Close Time
дд.мм.гггг --:--

Ticket Tiers

Add Another Tier | Create Event



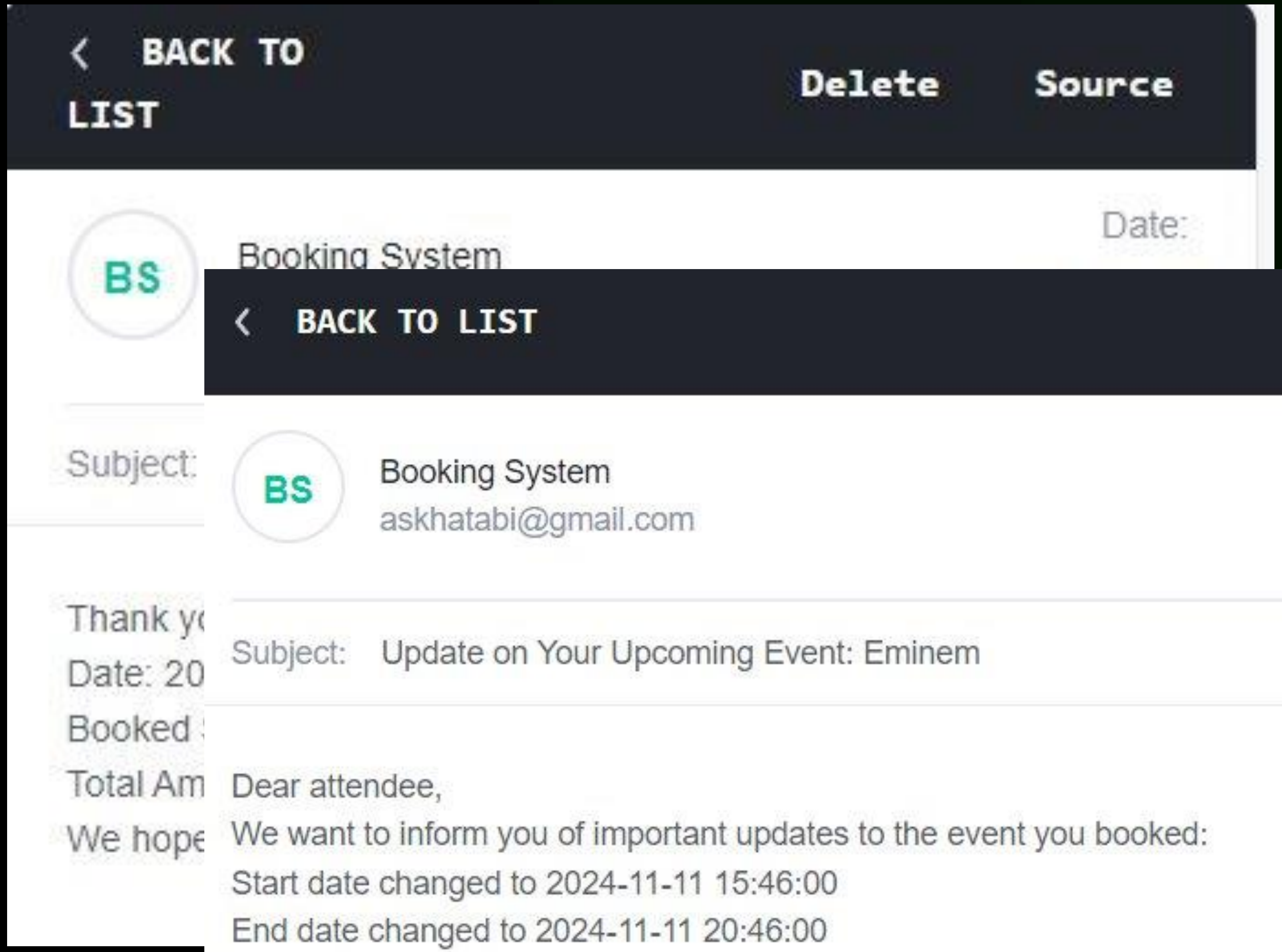
```
def create_event():
    if request.method == 'POST':
        title = request.form.get('title')
        description = request.form.get('description')
        venue = request.form.get('venue')
        start_date = request.form.get('start_date')
        end_date = request.form.get('end_date')

        # Get booking open and close times from the form
        booking_open_time = request.form.get('booking_open_time')
        booking_close_time = request.form.get('booking_close_time')

        # Ensure datetime fields are parsed correctly
        start_date = datetime.strptime(start_da
        end_date = datetime.strptime(end_date,
        booking_open_time = datetime.strptime(b
        booking_close_time = datetime.strptime(

        ticket_tiers = request.form.getlist('ti
        tier_prices = request.form.getlist('tie
        tier_seat_counts = request.form.getlist

        try:
            tier_seat_counts = [int(count) for
            total_tickets = sum(tier_seat_count
        except ValueError:
```



```
@app.route('/admin/update_event/<int:event_id>', methods=['GET', 'POST'])  rsmythrepo
@login_required
def update_event(event_id):
    if not is_admin():
        flash( message: "Unauthorized access!",  category: "error")
        return redirect(url_for('homepage'))

    # Fetch the event from the database
    event = Event.query.get(event_id)
    if not event:
        flash( message: "Event not found!",  category: "error")
        return redirect(url_for('admin_events'))

    # Capture original details for change detection
    original_details = {
        'title': event.title,
        'start_date': event.start_date,
        'end_date': event.end_date,
        'venue': event.venue,
        'booking_open_time': event.booking_open_time,
        'booking_close_time': event.booking_close_time,
        'ticket_price': {tier.tier_id: tier.ticket_tier.price for tier in event.ticket_tiers}  # Acc
    }

    if request.method == 'POST':
        # Update event details from the form
        event.title = request.form['title']
        event.start_date = datetime.strptime(request.form['start_date'],  format: '%Y-%m-%dT%H:%M')
```

04 Nov 2024

Email notifications

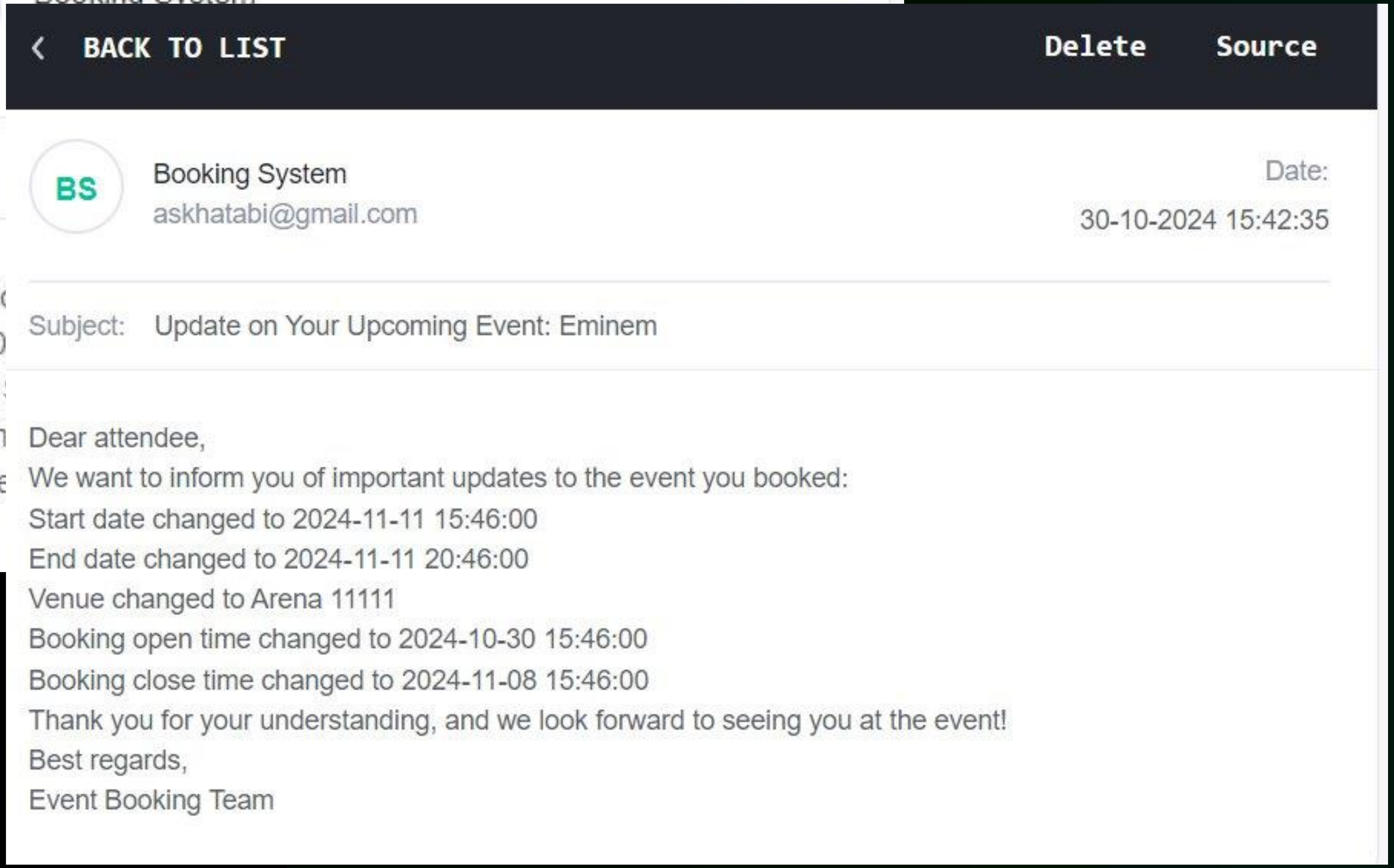


Booking System

Date:

Subject:

Thank y
Date: 20
Booked
Total Am
We hope



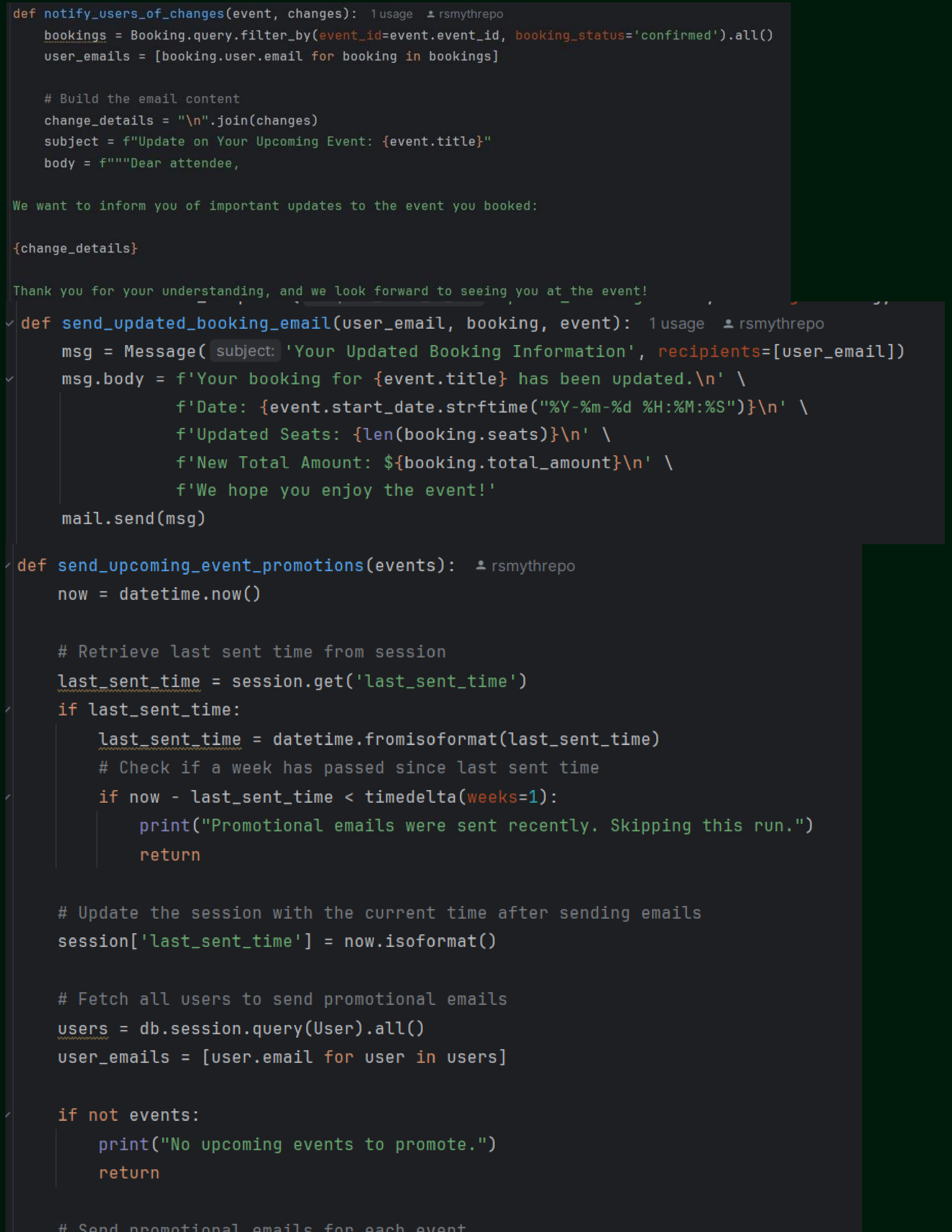
Booking System
askhatabi@gmail.com

Date:
30-10-2024 15:42:35

Subject: Update on Your Upcoming Event: Eminem

Dear attendee,
We want to inform you of important updates to the event you booked:
Start date changed to 2024-11-11 15:46:00
End date changed to 2024-11-11 20:46:00
Venue changed to Arena 11111
Booking open time changed to 2024-10-30 15:46:00
Booking close time changed to 2024-11-08 15:46:00
Thank you for your understanding, and we look forward to seeing you at the event!
Best regards,
Event Booking Team

04 Nov 2024



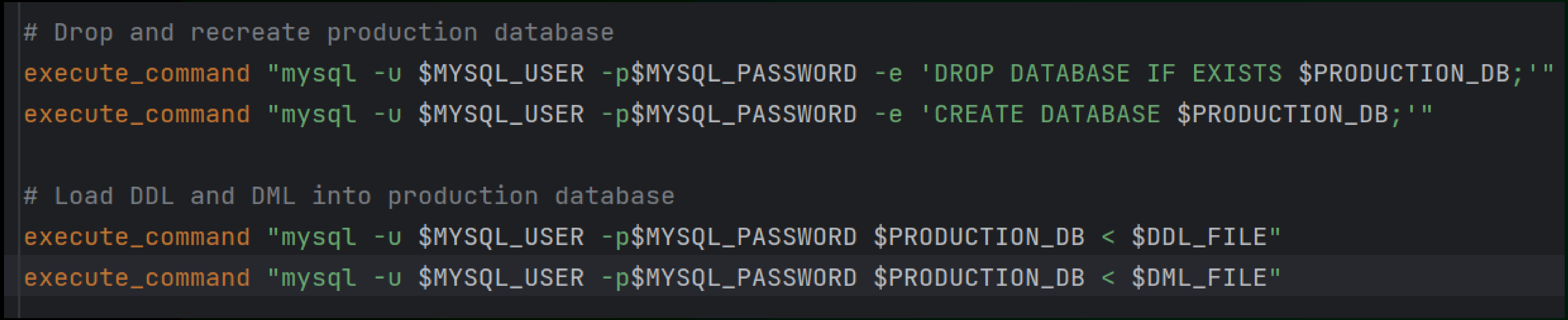
```
def notify_users_of_changes(event, changes):  1 usage  rsmythrepo
    bookings = Booking.query.filter_by(event_id=event.event_id, booking_status='confirmed').all()
    user_emails = [booking.user.email for booking in bookings]

    # Build the email content
    change_details = "\n".join(changes)
    subject = f"Update on Your Upcoming Event: {event.title}"
    body = f"""Dear attendee,

We want to inform you of important updates to the event you booked:

{change_details}

Thank you for your understanding, and we look forward to seeing you at the event!
```

```
def send_updated_booking_email(user_email, booking, event):  1 usage  rsmythrepo
    msg = Message( subject: 'Your Updated Booking Information', recipients=[user_email])
    msg.body = f'Your booking for {event.title} has been updated.\n' \
               f'Date: {event.start_date.strftime("%Y-%m-%d %H:%M:%S")}\n' \
               f'Updated Seats: {len(booking.seats)}\n' \
               f'New Total Amount: ${booking.total_amount}\n' \
               f'We hope you enjoy the event!'
    mail.send(msg)
```

```
def send_upcoming_event_promotions(events):  rsmythrepo
    now = datetime.now()

    # Retrieve last sent time from session
    last_sent_time = session.get('last_sent_time')
    if last_sent_time:
        last_sent_time = datetime.fromisoformat(last_sent_time)
        # Check if a week has passed since last sent time
        if now - last_sent_time < timedelta(weeks=1):
            print("Promotional emails were sent recently. Skipping this run.")
            return

    # Update the session with the current time after sending emails
    session['last_sent_time'] = now.isoformat()

    # Fetch all users to send promotional emails
    users = db.session.query(User).all()
    user_emails = [user.email for user in users]

    if not events:
        print("No upcoming events to promote.")
        return

    # Send promotional emails for each event
```

Toil

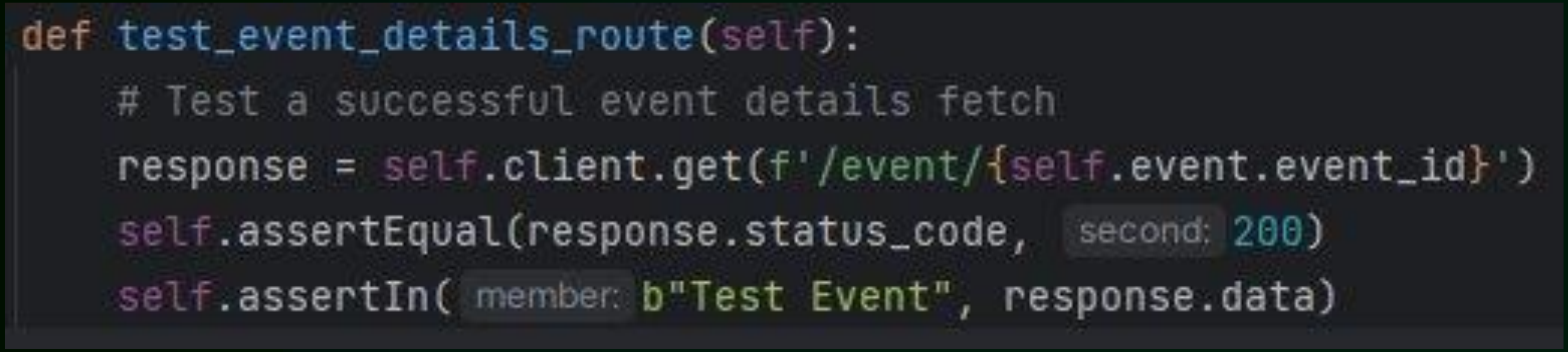
Automation of database reload



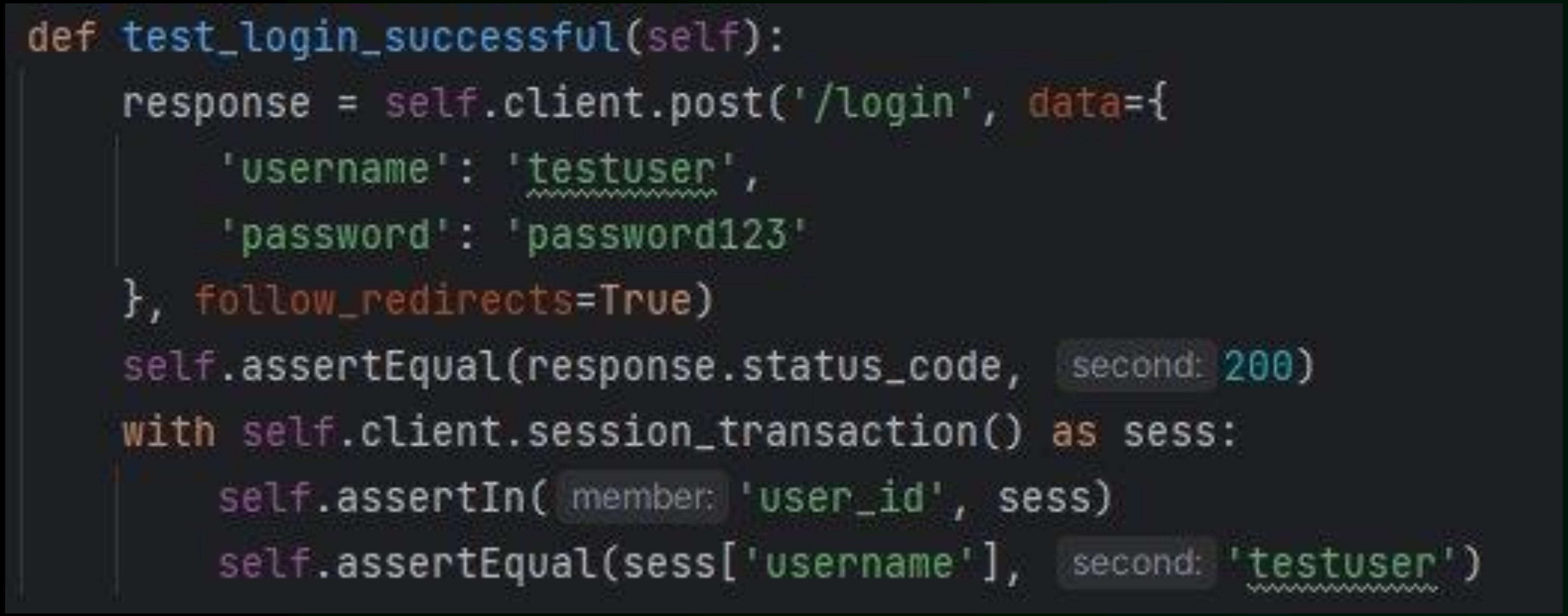
```
# Drop and recreate production database
execute_command "mysql -u $MYSQL_USER -p$MYSQL_PASSWORD -e 'DROP DATABASE IF EXISTS $PRODUCTION_DB;'"
execute_command "mysql -u $MYSQL_USER -p$MYSQL_PASSWORD -e 'CREATE DATABASE $PRODUCTION_DB;'"

# Load DDL and DML into production database
execute_command "mysql -u $MYSQL_USER -p$MYSQL_PASSWORD $PRODUCTION_DB < $DDL_FILE"
execute_command "mysql -u $MYSQL_USER -p$MYSQL_PASSWORD $PRODUCTION_DB < $DML_FILE"
```

04 Nov 2024

Testing



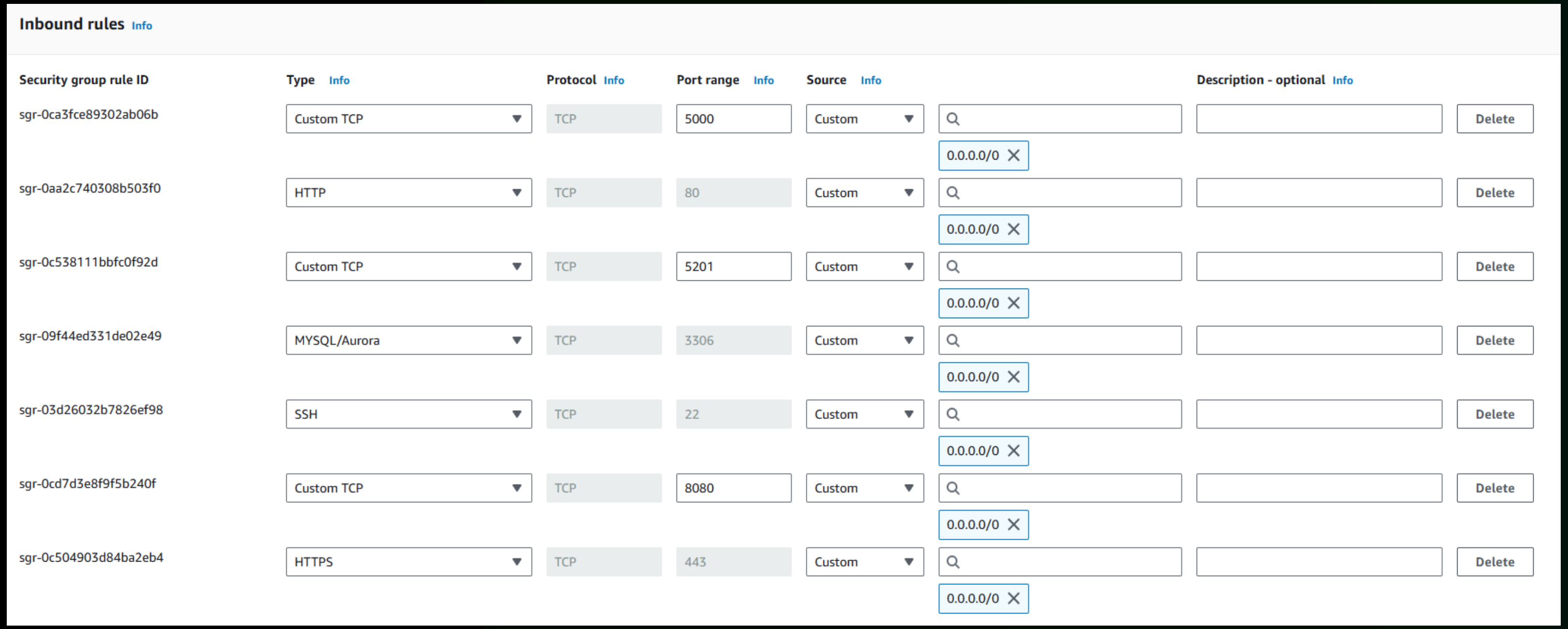
```python
def test_event_details_route(self):
    # Test a successful event details fetch
    response = self.client.get(f'/event/{self.event.event_id}')
    self.assertEqual(response.status_code,  second: 200)
    self.assertIn( member: b"Test Event", response.data)
```



```python
def test_login_successful(self):
    response = self.client.post('/login', data={
        'username': 'testuser',
        'password': 'password123'
    }, follow_redirects=True)
    self.assertEqual(response.status_code,  second: 200)
    with self.client.session_transaction() as sess:
        self.assertIn( member: 'user_id', sess)
        self.assertEqual(sess['username'],  second: 'testuser')
```

04 Nov 2024

Containerization



Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional	
sgr-0ca3fce89302ab06b	Custom TCP	TCP	5000	Custom	0.0.0.0/0		Delete
sgr-0aa2c740308b503f0	HTTP	TCP	80	Custom	0.0.0.0/0		Delete
sgr-0c538111bbfc0f92d	Custom TCP	TCP	5201	Custom	0.0.0.0/0		Delete
sgr-09f44ed331de02e49	MYSQL/Aurora	TCP	3306	Custom	0.0.0.0/0		Delete
sgr-03d26032b7826ef98	SSH	TCP	22	Custom	0.0.0.0/0		Delete
sgr-0cd7d3e8f9f5b240f	Custom TCP	TCP	8080	Custom	0.0.0.0/0		Delete
sgr-0c504903d84ba2eb4	HTTPS	TCP	443	Custom	0.0.0.0/0		Delete

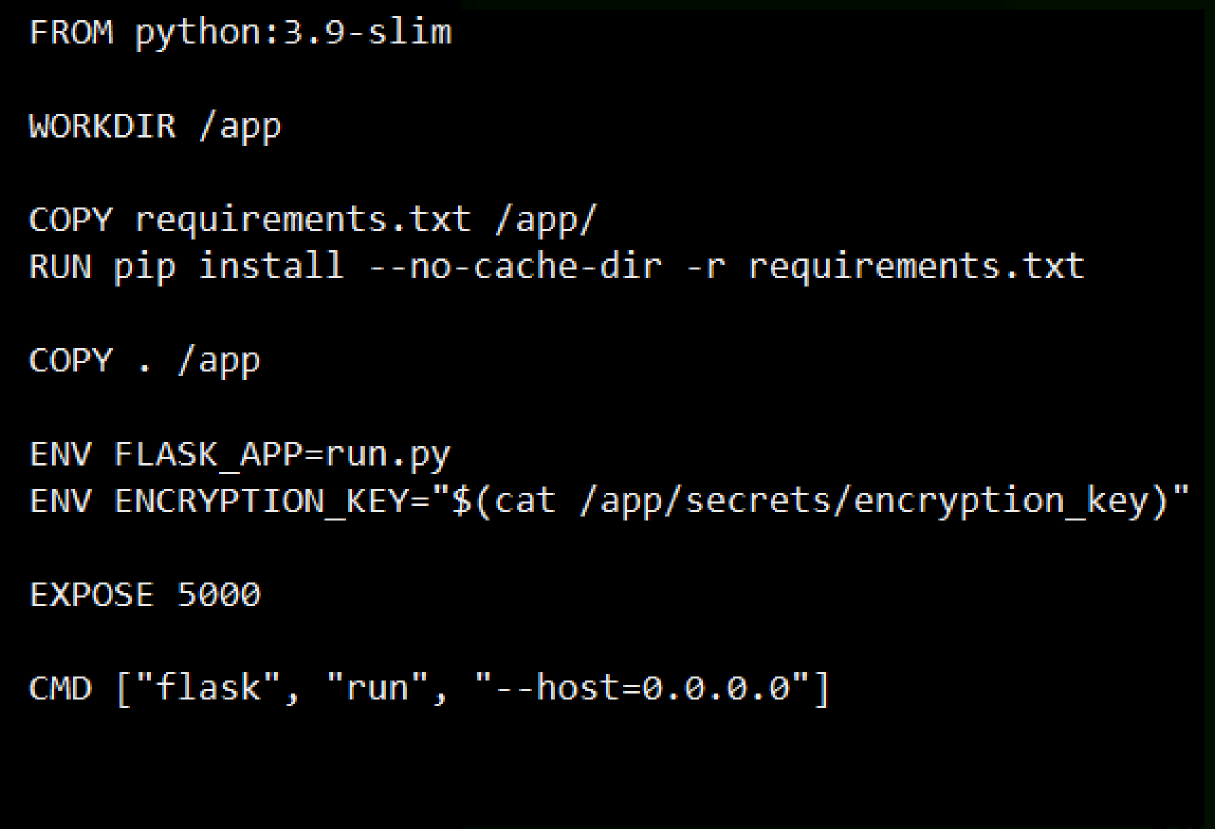
Configure Security Groups:
- Allow **SSH (port 22)**, **HTTP (port 80)**, and **Custom TCP (e.g., port 5000)**.

```
sudo amazon-linux-extras install docker -y sudo service docker start sudo
usermod -a -G docker ec2-user # Add user to Docker group
```

Project 2

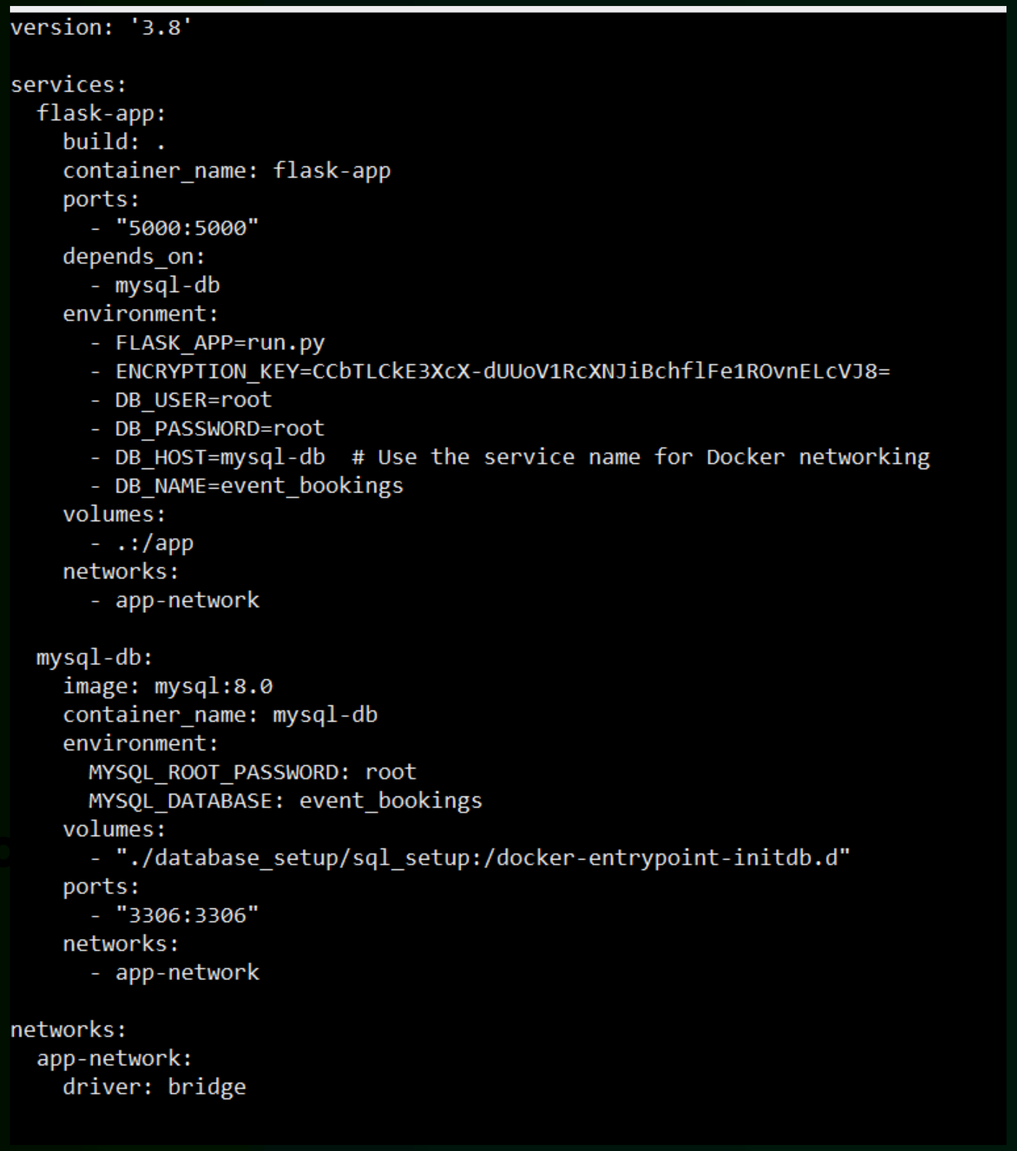
04 Nov 2024

Containerization



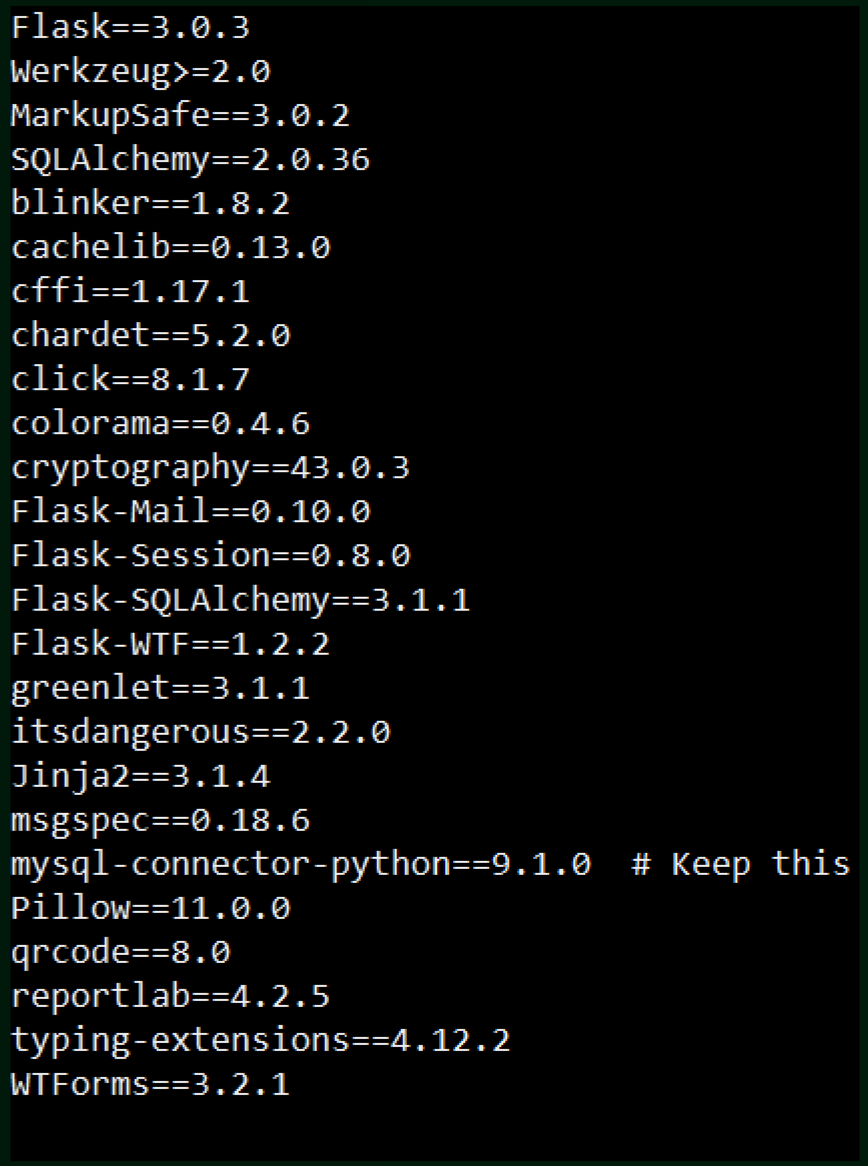
```
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt /app/
RUN pip install --no-cache-dir -r requirements.txt

COPY . /app

ENV FLASK_APP=run.py
ENV ENCRYPTION_KEY="$(cat /app/secrets/encryption_key)"

EXPOSE 5000

CMD ["flask", "run", "--host=0.0.0.0"]
```

Dockerfile



```
version: '3.8'

services:
  flask-app:
    build: .
    container_name: flask-app
    ports:
      - "5000:5000"
    depends_on:
      - mysql-db
    environment:
      - FLASK_APP=run.py
      - ENCRYPTION_KEY=CCbTLCkE3XcX-dUUoV1RcXNJiBchflFe1ROvnELcVJ8=
      - DB_USER=root
      - DB_PASSWORD=root
      - DB_HOST=mysql-db  # Use the service name for Docker networking
      - DB_NAME=event_bookings
    volumes:
      - .:/app
    networks:
      - app-network

  mysql-db:
    image: mysql:8.0
    container_name: mysql-db
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: event_bookings
    volumes:
      - "./database_setup/sql_setup:/docker-entrypoint-initdb.d"
    ports:
      - "3306:3306"
    networks:
      - app-network

networks:
  app-network:
    driver: bridge
```

Docker-compose.yml

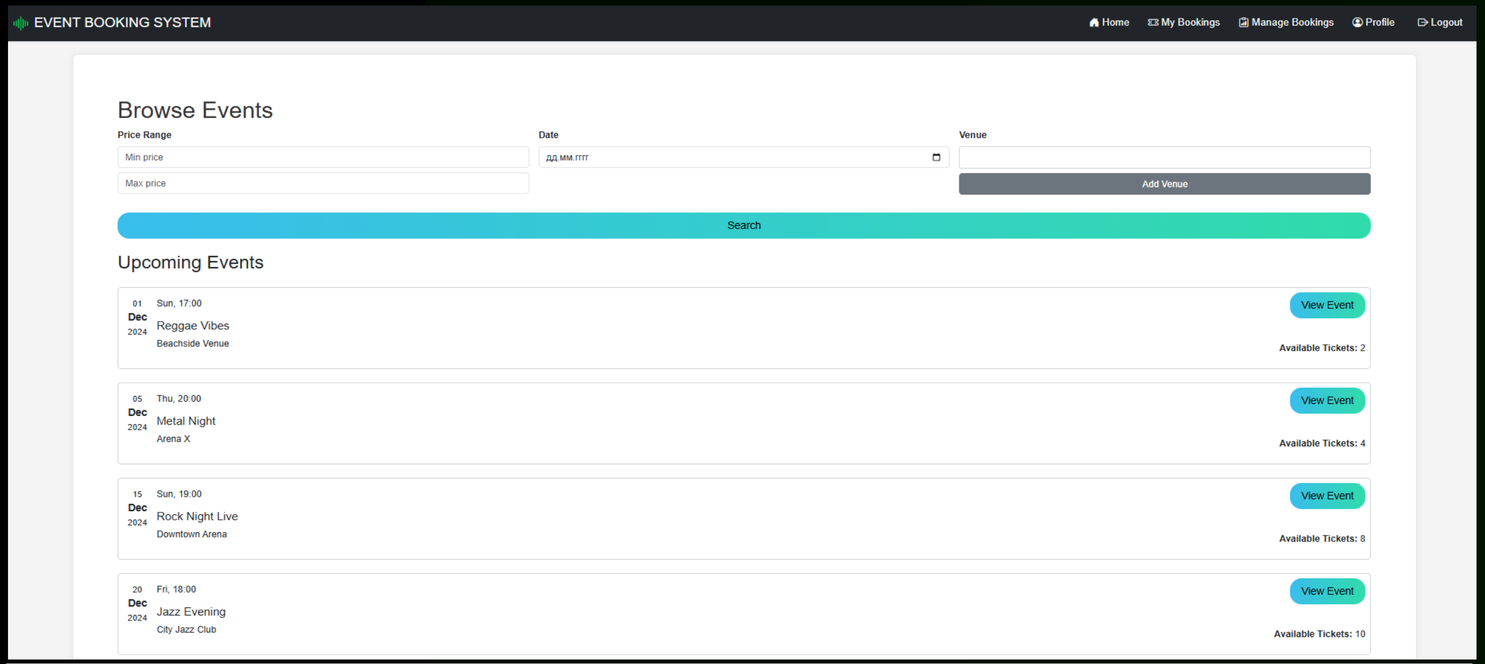


```
Flask==3.0.3
Werkzeug>=2.0
MarkupSafe==3.0.2
SQLAlchemy==2.0.36
blinker==1.8.2
cachelib==0.13.0
cffi==1.17.1
chardet==5.2.0
click==8.1.7
colorama==0.4.6
cryptography==43.0.3
Flask-Mail==0.10.0
Flask-Session==0.8.0
Flask-SQLAlchemy==3.1.1
Flask-WTF==1.2.2
greenlet==3.1.1
itsdangerous==2.2.0
Jinja2==3.1.4
msgspec==0.18.6
mysql-connector-python==9.1.0  # Keep this
Pillow==11.0.0
qrcode==8.0
reportlab==4.2.5
typing-extensions==4.12.2
WTForms==3.2.1
```

requirements.txt

04 Nov 2024

Containerization

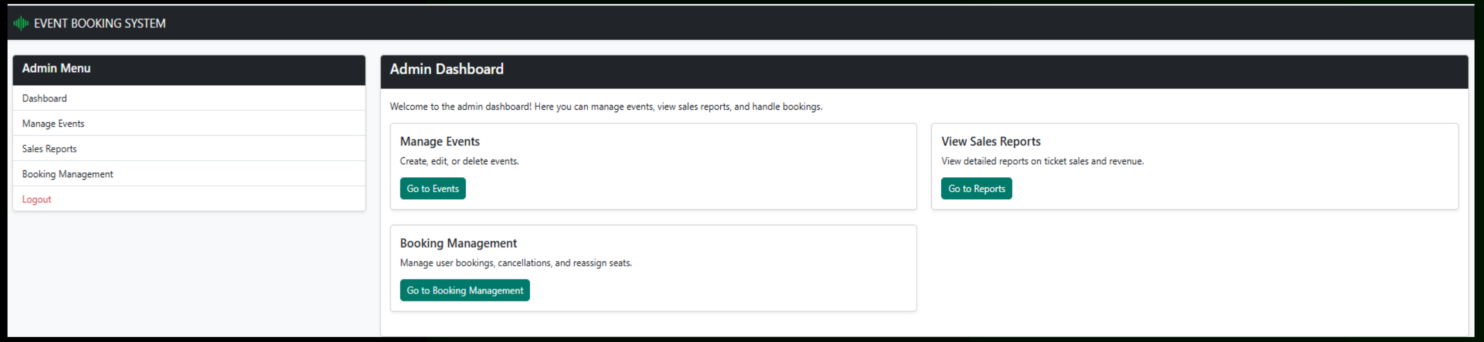

EVENT BOOKING SYSTEM

Browse Events

Price Range: Min price / Max price
Date: дд.мм.гггг
Venue:

Upcoming Events

01 Dec 2024	Sun, 17:00 Reggae Vibes Beachside Venue	View Event Available Tickets: 2
05 Dec 2024	Thu, 20:00 Metal Night Arena X	View Event Available Tickets: 4
15 Dec 2024	Sun, 19:00 Rock Night Live Downtown Arena	View Event Available Tickets: 8
20 Dec 2024	Fri, 18:00 Jazz Evening City Jazz Club	View Event Available Tickets: 10


EVENT BOOKING SYSTEM

Admin Menu

Admin Dashboard

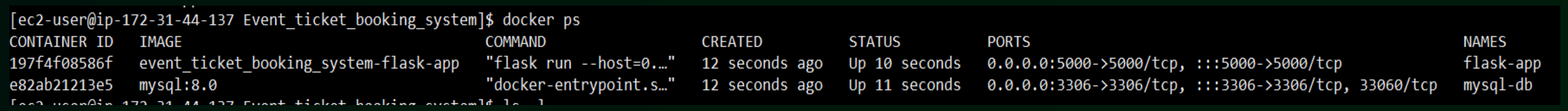
Welcome to the admin dashboard! Here you can manage events, view sales reports, and handle bookings.

Manage Events
Create, edit, or delete events.

View Sales Reports
View detailed reports on ticket sales and revenue.

Booking Management
Manage user bookings, cancellations, and reassign seats.

sudo docker-compose up --build -d


```
[ec2-user@ip-172-31-44-137 Event_ticket_booking_system]$ docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
197f4f08586f   event_ticket_booking_system-flask-app  "flask run --host=0.…"   12 seconds ago   Up 10 seconds   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp              flask-app
e82ab21213e5   mysql:8.0                              "docker-entrypoint.s…"   12 seconds ago   Up 11 seconds   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   mysql-db
```

04 Nov 2024

Demo

04 Nov 2024

MUSIC PARTY

GitHub

Academy

mThree

Contributors

Raphaelle Smyth

Askhat Bissembay

Yurii Maisuradze

2024 Music Presentation

MUSIC PARTY

Questions?

04 Nov 2024