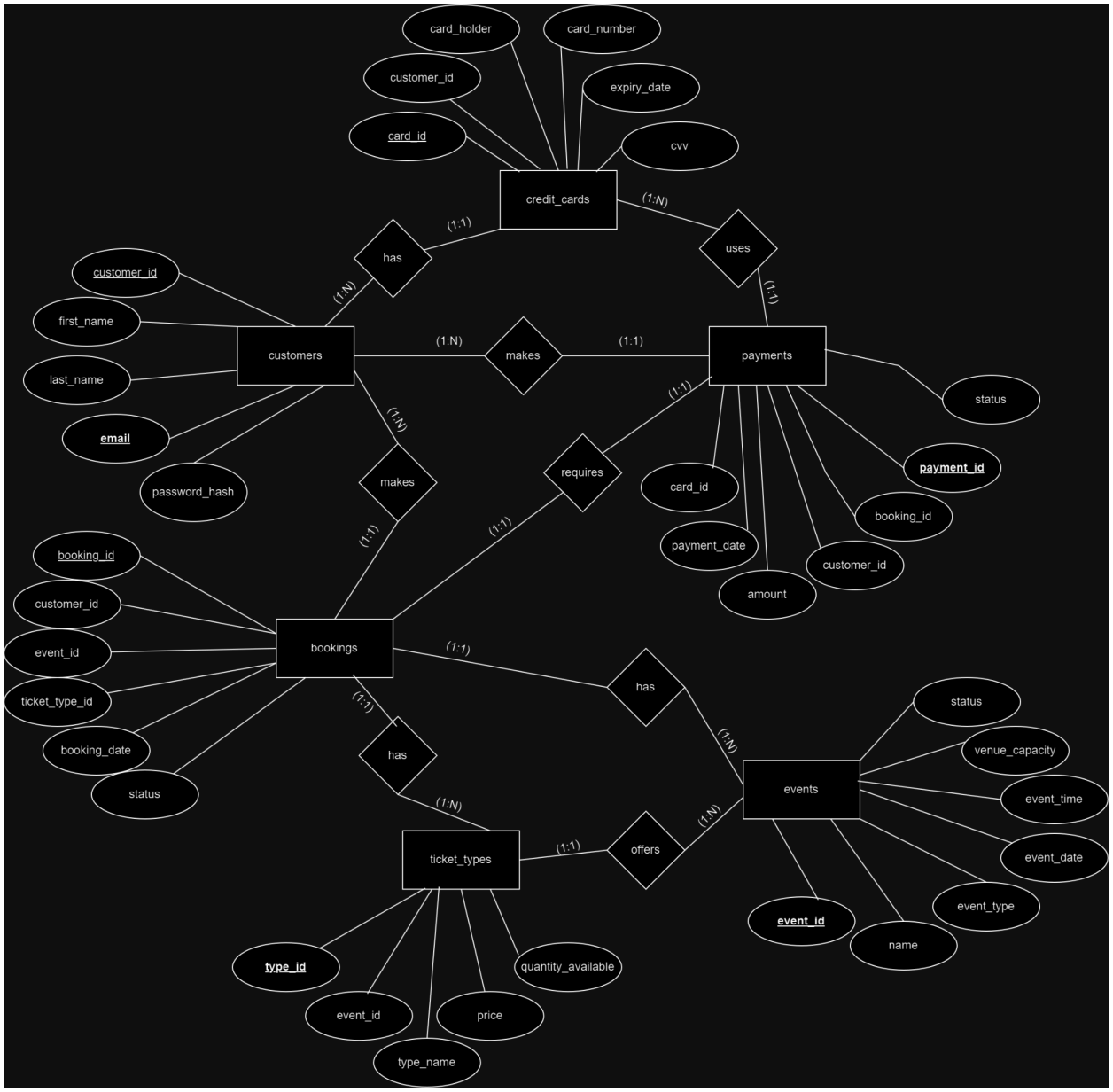
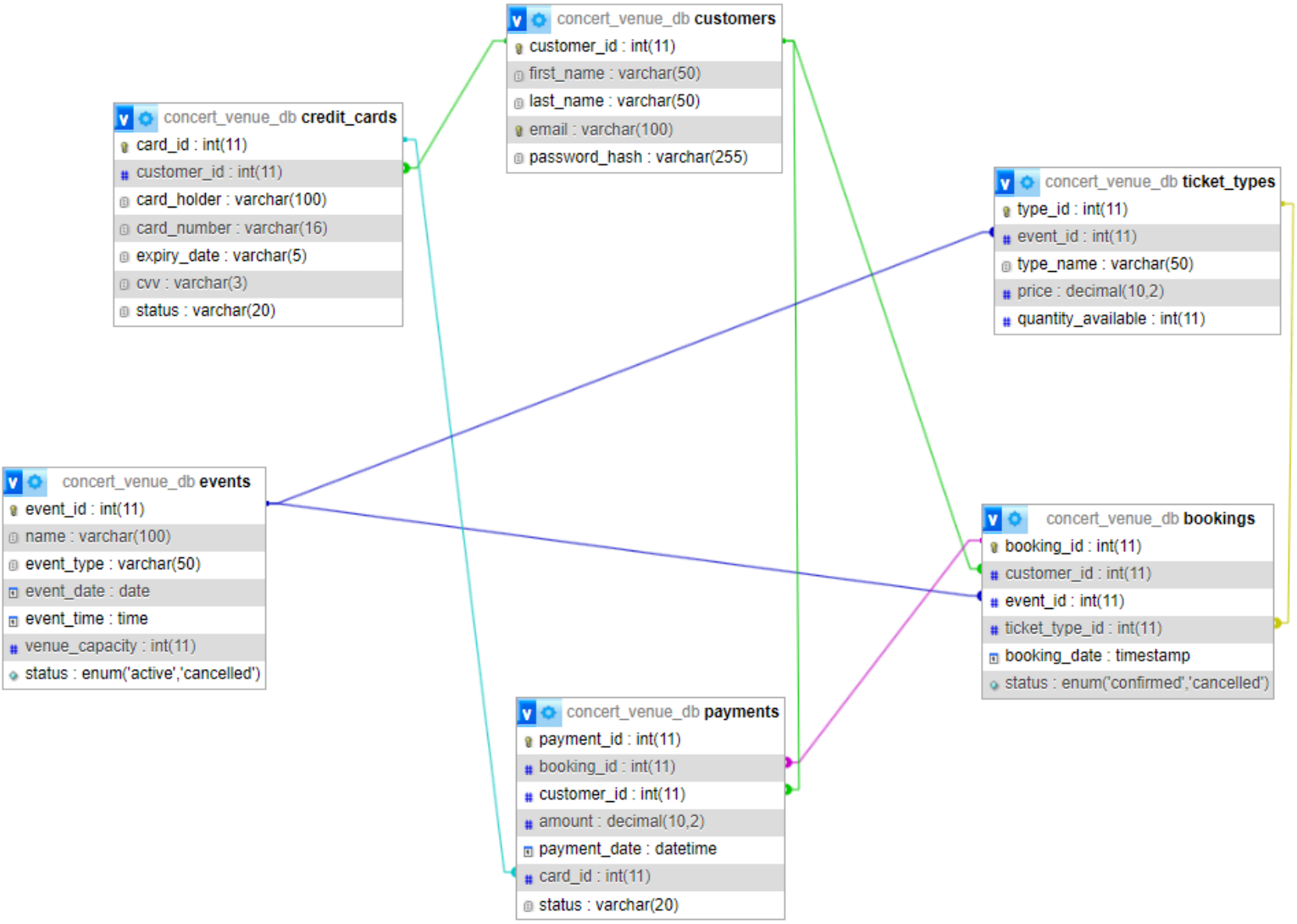
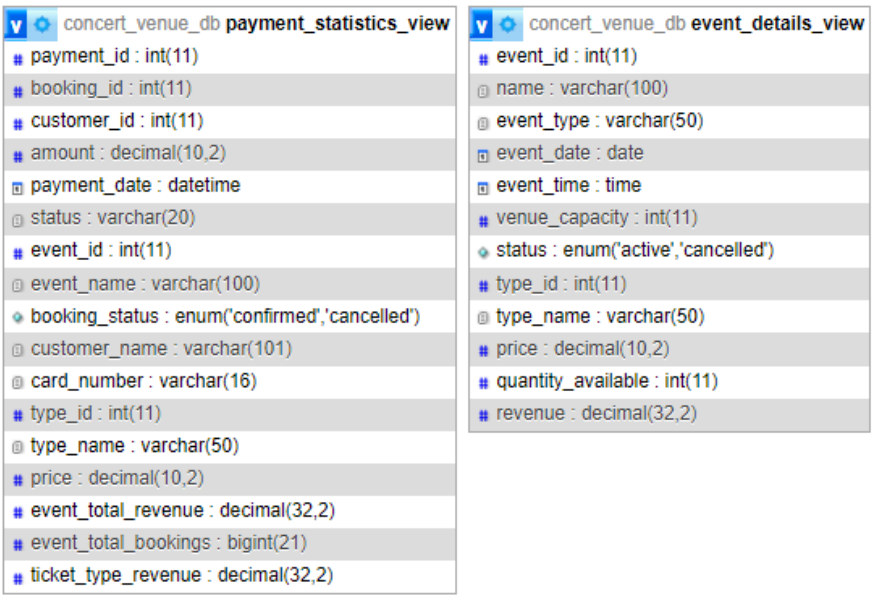
PHASE 2 REPORT
HY360 PROJECT
csd4897 - csd4647 - csd4876
2024-25

1)Entity Relationship Diagram




concert_venue_db customers
customer_id : int(11)
first_name : varchar(50)
last_name : varchar(50)
email : varchar(100)
password_hash : varchar(255)
concert_venue_db credit_cards
card_id : int(11)
customer_id : int(11)
card_holder : varchar(100)
card_number : varchar(16)
expiry_date : varchar(5)
cvv : varchar(3)
status : varchar(20)
concert_venue_db ticket_types
type_id : int(11)
event_id : int(11)
type_name : varchar(50)
price : decimal(10,2)
quantity_available : int(11)
concert_venue_db events
event_id : int(11)
name : varchar(100)
event_type : varchar(50)
event_date : date
event_time : time
venue_capacity : int(11)
status : enum('active','cancelled')
concert_venue_db bookings
booking_id : int(11)
customer_id : int(11)
event_id : int(11)
ticket_type_id : int(11)
booking_date : timestamp
status : enum('confirmed','cancelled')
concert_venue_db payments
payment_id : int(11)
booking_id : int(11)
customer_id : int(11)
amount : decimal(10,2)
payment_date : datetime
card_id : int(11)
status : varchar(20)


concert_venue_db payment_statistics_view
payment_id : int(11)
booking_id : int(11)
customer_id : int(11)
amount : decimal(10,2)
payment_date : datetime
status : varchar(20)
event_id : int(11)
event_name : varchar(100)
booking_status : enum('confirmed','cancelled')
customer_name : varchar(101)
card_number : varchar(16)
type_id : int(11)
type_name : varchar(50)
price : decimal(10,2)
event_total_revenue : decimal(32,2)
event_total_bookings : bigint(21)
ticket_type_revenue : decimal(32,2)
concert_venue_db event_details_view
event_id : int(11)
name : varchar(100)
event_type : varchar(50)
event_date : date
event_time : time
venue_capacity : int(11)
status : enum('active','cancelled')
type_id : int(11)
type_name : varchar(50)
price : decimal(10,2)
quantity_available : int(11)
revenue : decimal(32,2)

2)Attributes of all entities and relationships

Events:
- event_id [Primary Key]
- name
- event_type
- event_date
- event_time
- venue_capacity
- status: active/cancelled
- created_at

Customers:
- customer_id [Primary Key]
- first_name
- last_name
- email
- password_hash
- created_at

Credit Cards:
- card_id [Primary Key]
- customer_id [Foreign Key]
- card_holder
- card_number
- expiry_date
- cvv

Payments:
- payment_id [Primary Key]
- booking_id [Foreign Key]
- customer_id [Foreign Key]
- amount
- payment_date
- card_id [Foreign Key]
- status

Bookings:
- booking_id [Primary Key]
- customer_id [Foreign Key]
- event_id [Foreign Key]
- ticket_type_id [Foreign Key]
- booking_date
- status: confirmed/cancelled

Ticket Types:
- type_id [Primary Key]
- event_id [Foreign Key]
- type_name
- price
- quantity_available

3)Primary keys

events: event_id

customers: customer_id

bookings: booking_id

payments: payment_id

credit_cards: card_id

ticket_types: type_id

4)Explanations for non-obvious attributes and relationships

events table:
- created_at: tracks when the event is added to the system
- status enum('active','cancelled'): tracks cancelled events
- venue_capacity: for ticket sales limits

customers table:
- password_hash: stores encrypted (hash) password instead of plain text
- created_at: tracks when customers registered

credit cards table:
- all attributes are self-explanatory from their names

payments table:
- status: tracks and shows payment stages like pending etc.

bookings table:
- status enum('confirmed','cancelled'): for booking state = cancelled or confirmed

ticket_types:
- all attributes are self-explanatory here as well from their names

5)Cardinality constraints

customers with bookings:
one customer can have multiple bookings (1:N)
each booking must correspond to only one customer (1:1)

events with bookings:
one event can have many bookings (1:N)
each booking must correspond to only one event (1:1)

events with ticket_types:
each event can have multiple ticket types (1:N)
each ticket type belongs to exactly one event (1:1)

bookings with payments:
one booking can have only one payment (1:1)
each payment belongs to only one booking (1:1)

customers with payments:
one customer can have made many payments (1:N)
each payment must belong to exactly one customer (1:1)

customers with credit_Cards:
one customer can have multiple credit cards (1:N)
each credit card belongs to only one customer (1:1)

credit_cards with payments:
one credit card can be used for many payments (1:N)
one payment can be made with only one credit card (1:1)

bookings with ticket_types:
one booking can have one ticket type (1:1)
one ticket type can be used in many bookings (1:N)

6)Translation to relational model

customers	(**customer_id** ,	first_name ,	last_name,	**email**,	password_hash)			
credit_cards	(**card_id** ,	customer_id,	card_holder,	card_number,	expiry_date,	cvv)		
events	(**event_id**,	name,	event_type,	event_date,	event_time,	venue_capacity,	status(active,cancelled))	
ticket_types	(**type_id**,	event_id,	type_name,	price,	quantity_available)			
bookings	(**booking_id,**	customer_id,	event_id,	ticket_type_id,	booking_date,	status(confirmed,cancelled))		
payments	(**payment_id**,	booking_id,	customer_id,	card_id,	amount,	payment_date,	status)	

7)Data Definition Language (DDL) commands for resulting relations

Customers)

```
CREATE TABLE `customers` (
  `customer_id` int(11) NOT NULL AUTO_INCREMENT,
  `first_name` varchar(50) NOT NULL,
  `last_name` varchar(50) NOT NULL,
  `email` varchar(100) NOT NULL,
  `password_hash` varchar(255) NOT NULL,
  PRIMARY KEY (`customer_id`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci
```

Events)

```
CREATE TABLE `events` (
  `event_id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(100) NOT NULL,
  `event_type` varchar(50) NOT NULL,
  `event_date` date NOT NULL,
  `event_time` time NOT NULL,
  `venue_capacity` int(11) NOT NULL,
  `status` enum('active','cancelled') DEFAULT 'active',
  PRIMARY KEY (`event_id`)
) ENGINE=InnoDB AUTO_INCREMENT=24 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci
```

Payments)

CREATE TABLE `payments` (`payment_id` int(11) NOT NULL AUTO_INCREMENT, `booking_id` int(11) DEFAULT NULL, `customer_id` int(11) DEFAULT NULL, `amount` decimal(10,2) DEFAULT NULL, `payment_date` datetime DEFAULT NULL, `card_id` int(11) DEFAULT NULL, `status` varchar(20) DEFAULT NULL, PRIMARY KEY (`payment_id`), KEY `booking_id` (`booking_id`), KEY `customer_id` (`customer_id`), KEY `card_id` (`card_id`), CONSTRAINT `payments_ibfk_1` FOREIGN KEY (`booking_id`) REFERENCES `bookings` (`booking_id`), CONSTRAINT `payments_ibfk_2` FOREIGN KEY (`customer_id`) REFERENCES `customers` (`customer_id`), CONSTRAINT `payments_ibfk_3` FOREIGN KEY (`card_id`) REFERENCES `credit_cards` (`card_id`)) ENGINE=InnoDB AUTO_INCREMENT=36 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci

Ticket_types)

CREATE TABLE `ticket_types` (`type_id` int(11) NOT NULL AUTO_INCREMENT, `event_id` int(11) DEFAULT NULL, `type_name` varchar(50) NOT NULL, `price` decimal(10,2) NOT NULL, `quantity_available` int(11) NOT NULL, PRIMARY KEY (`type_id`), KEY `event_id` (`event_id`), CONSTRAINT `ticket_types_ibfk_1` FOREIGN KEY (`event_id`) REFERENCES `events` (`event_id`)) ENGINE=InnoDB AUTO_INCREMENT=27 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci

Bookings)

```
CREATE TABLE `bookings` (
  `booking_id` int(11) NOT NULL AUTO_INCREMENT,
  `customer_id` int(11) DEFAULT NULL,
  `event_id` int(11) DEFAULT NULL,
  `ticket_type_id` int(11) DEFAULT NULL,
  `booking_date` timestamp NOT NULL DEFAULT current_timestamp(),
  `status` enum('confirmed','cancelled') DEFAULT 'confirmed',
  PRIMARY KEY (`booking_id`),
  KEY `customer_id` (`customer_id`),
  KEY `event_id` (`event_id`),
  KEY `ticket_type_id` (`ticket_type_id`),
  CONSTRAINT `bookings_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `customers`
(`customer_id`),
  CONSTRAINT `bookings_ibfk_2` FOREIGN KEY (`event_id`) REFERENCES `events` (`event_id`),
  CONSTRAINT `bookings_ibfk_3` FOREIGN KEY (`ticket_type_id`) REFERENCES `ticket_types`
(`type_id`)
) ENGINE=InnoDB AUTO_INCREMENT=32 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci
```

Credit_cards)

```
CREATE TABLE `credit_cards` (
  `card_id` int(11) NOT NULL AUTO_INCREMENT,
  `customer_id` int(11) DEFAULT NULL,
  `card_holder` varchar(100) DEFAULT NULL,
  `card_number` varchar(16) DEFAULT NULL,
  `expiry_date` varchar(5) DEFAULT NULL,
  `cvv` varchar(3) DEFAULT NULL,
  `status` varchar(20) DEFAULT 'active',
  PRIMARY KEY (`card_id`),
  KEY `customer_id` (`customer_id`),
  CONSTRAINT `credit_cards_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `customers`
(`customer_id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci
```

8)Integrity constraints and functional dependencies

INTEGRITY CONSTRAINTS:

1. Primary Keys:
- All primary keys must be initialized and unique

2. Foreign Key Constraints:
- credit_cards.customer_id must exist in customers.customer_id
- ticket_types.event_id must exist in events.event_id
- bookings.customer_id must exist in customers.customer_id
- bookings.event_id must exist in events.event_id
- bookings.ticket_type_id must exist in ticket_types.type_id
- payments.booking_id must exist in bookings.booking_id
- payments.card_id must exist in credit_cards.card_id

3. Value Constraints:
- events.venue_capacity must be > 0
- events.status must be 'active' or 'cancelled'
- ticket_types.price must be >= 0
- ticket_types.quantity_available must be >= 0
- bookings.status must be 'confirmed' or 'cancelled'
- payments.amount must be >= 0

FUNCTIONAL DEPENDENCIES:

1. Events:
event_id → name
event_id → event_type
event_id → event_date
event_id → event_time
event_id → venue_capacity
event_id → status
event_id → created_at

2. Customers:
customer_id → first_name
customer_id → last_name
customer_id → email
customer_id → password_hash
customer_id → created_at

3. Credit Cards:
card_id → customer_id
card_id → card_holder
card_id → card_number
card_id → expiry_date
card_id → cvv

4. Ticket Types:
type_id → event_id
type_id → type_name
type_id → price
type_id → quantity_available

5. Bookings:
booking_id → customer_id
booking_id → event_id
booking_id → ticket_type_id
booking_id → booking_date
booking_id → status

6. Payments:
payment_id → booking_id
payment_id → customer_id
payment_id → amount
payment_id → payment_date
payment_id → card_id
payment_id → status

9)Determination of relation keys based on functional dependencies

For the events table:
- Primary key: event_id
- Foreign keys: none

For the customers table:
- Primary key: customer_id
- Foreign keys: none

For the credit_cards table:
- Primary key: card_id
- Foreign keys: customer_id (primary key in the customers table)

For the ticket_types table:
- Primary key: type_id
- Foreign keys: event_id (primary key in the events table)

For the bookings table:
- Primary key: booking_id
- Foreign keys:
  - customer_id (primary key in the customers table)
  - event_id (primary key in the events table)
  - ticket_type_id (primary key in the ticket_types table)

For the payments table:
- Primary key: payment_id
- Foreign keys:
  - booking_id (primary key in the bookings table)
  - customer_id (primary key in the customers table)
  - card_id (primary key in the credit_cards table)

10)Conversion to third normal form while maintaining functional dependencies and without loss of information

```
events (
   event_id (PK),
   name,
   event_type,
   event_date,
   event_time,
   venue_capacity,
   status,
   created_at
)

customers (
   customer_id (PK),
   first_name,
   last_name,
   email,
   password_hash,
   created_at
)

credit_cards (
   card_id (PK),
   customer_id (FK references Customers),
   card_holder,
   card_number,
   expiry_date,
   cvv
)

ticket_types (
   type_id (PK),
   event_id (FK references Events),
   type_name,
   price,
   quantity_available
)

bookings (
   booking_id (PK),
```

```
    customer_id (FK references Customers),
    event_id (FK references Events),
    ticket_type_id (FK references Ticket_Types),
    booking_date,
    status
)

payments (
    payment_id (PK),
    booking_id (FK references Bookings),
    customer_id (FK references Customers),
    amount,
    payment_date,
    card_id (FK references Credit_Cards),
    status
)
```

Definition: A database schema D is in Third Normal Form if all relations in D are in Third Normal Form. We can verify this by checking each table individually.
Given a relation R and a set of functional dependencies F. R is in third normal form (3NF) if for every non-trivial functional dependency of the form X → A that holds in R and belongs to F+, one of the following statements is true:
➢ X is a super-key for R or
➢ A is a prime attribute of R Also, there appear to be no transitive dependencies. Therefore, based on the slides, it is in third normal form.

11)Description of database queries in SQL

```
-- Booking Management
SELECT b.ticket_type_id, tt.price, p.card_id
FROM bookings b
JOIN ticket_types tt ON b.ticket_type_id = tt.type_id
JOIN payments p ON b.booking_id = p.booking_id
WHERE b.booking_id = ? AND b.customer_id = ?
AND b.status = 'confirmed'
AND p.status = 1;

INSERT INTO payments (booking_id, customer_id, amount, payment_date, card_id, status)
VALUES (?, ?, ?, NOW(), ?, 0);

UPDATE ticket_types
SET quantity_available = quantity_available + 1
WHERE type_id = ?;

SELECT e.status as event_status, tt.quantity_available, tt.price
FROM events e
JOIN ticket_types tt ON e.event_id = tt.event_id
WHERE e.event_id = ? AND tt.type_id = ?
FOR UPDATE;

INSERT INTO bookings (customer_id, event_id, ticket_type_id, booking_date, status)
VALUES (?, ?, ?, NOW(), 'confirmed');

INSERT INTO payments (booking_id, customer_id, amount, payment_date, card_id, status)
VALUES (?, ?, ?, NOW(), ?, 1);

UPDATE ticket_types
SET quantity_available = quantity_available - 1
WHERE type_id = ? AND quantity_available > 0;

SELECT b.booking_id, b.status AS booking_status, b.booking_date,
    e.name AS event_name, e.event_date, e.event_time,
    tt.type_name, tt.price, p.status AS payment_status
FROM bookings b
INNER JOIN events e ON b.event_id = e.event_id
INNER JOIN ticket_types tt ON b.ticket_type_id = tt.type_id
LEFT JOIN payments p ON b.booking_id = p.booking_id
WHERE b.customer_id = ?
ORDER BY b.booking_date DESC;
```

```
-- Event Management
INSERT INTO events (name, event_type, event_date, event_time, venue_capacity, status)
VALUES (?, ?, ?, ?, ?, ?);

SELECT e.status as event_status, b.booking_id, b.customer_id,
       tt.price, p.card_id
FROM events e
LEFT JOIN bookings b ON e.event_id = b.event_id AND b.status = 'confirmed'
LEFT JOIN ticket_types tt ON b.ticket_type_id = tt.type_id
LEFT JOIN payments p ON b.booking_id = p.booking_id AND p.status = 1
WHERE e.event_id = ?;

UPDATE bookings
SET status = 'cancelled'
WHERE booking_id = ?;

UPDATE events
SET status = 'cancelled'
WHERE event_id = ?;

SELECT * FROM event_details_view
WHERE status != 'cancelled'
ORDER BY event_date, event_time, event_id;

SELECT * FROM event_details_view
WHERE event_id = ? AND status != 'cancelled';

-- User Management
SELECT customer_id, first_name, last_name, email
FROM customers
WHERE email = ? AND password_hash = ?;

SELECT customer_id
FROM customers
WHERE email = ?;

SELECT customer_id, first_name, last_name, email
FROM customers
WHERE customer_id = ?;

-- Payment Methods
SELECT * FROM credit_cards
WHERE customer_id = ? AND (status = 'active' OR status IS NULL);
```

```
INSERT INTO credit_cards (customer_id, card_holder, card_number, expiry_date, cvv)
VALUES (?, ?, ?, ?, ?);

SELECT COUNT(*)
FROM payments
WHERE card_id = ?;

UPDATE credit_cards
SET status = 'inactive'
WHERE card_id = ? AND customer_id = ?;

DELETE FROM credit_cards
WHERE card_id = ? AND customer_id = ?;
```

-- Payment Statistics

```
SELECT * FROM payment_statistics_view
WHERE 1=1
AND status = ?
AND DATE(payment_date) = ?
ORDER BY payment_date DESC;
```

-- Ticket Types

```
INSERT INTO ticket_types (event_id, type_name, price, quantity_available)
VALUES (?, ?, ?, ?);
```

12) Views

The system uses two views to support reporting and statistics functionality:

1. event_details_view
Purpose:
- Provides comprehensive event information including ticket types
- Calculates net revenue per ticket type accounting for both payments and refunds
- Used for event management interface

Structure:
- Events basic information (id, name, type, date, time, capacity, status)
- Associated ticket types (type, price, availability)
- Calculated revenue per ticket type (payments minus refunds)

```
CREATE TABLE `event_details_view` (
`event_id` int(11)
,`name` varchar(100)
,`event_type` varchar(50)
,`event_date` date
,`event_time` time
,`venue_capacity` int(11)
,`status` enum('active','cancelled')
,`type_id` int(11)
,`type_name` varchar(50)
,`price` decimal(10,2)
,`quantity_available` int(11)
,`revenue` decimal(32,2)
);
```

2. payment_statistics_view
Purpose:
- Provides detailed payment and booking statistics
- Used for financial reporting and event performance analysis
- Supports admin dashboard statistics

Structure:
- Complete payment information including customer details
- Event and booking status tracking
- Revenue calculations:
  * Total revenue per event
  * Total bookings per event

* Revenue per ticket type
* Customer payment history

```
CREATE TABLE `payments` (
  `payment_id` int(11) NOT NULL,
  `booking_id` int(11) DEFAULT NULL,
  `customer_id` int(11) DEFAULT NULL,
  `amount` decimal(10,2) DEFAULT NULL,
  `payment_date` datetime DEFAULT NULL,
  `card_id` int(11) DEFAULT NULL,
  `status` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

13)Servlets Pseudocode

BookingCancellationServlet:
Handles cancellation of bookings, updates booking status and creates refund payment
Check booking exists and status → Update booking status to cancelled → Create refund payment

BookingServlet:
Handles new booking creation with ticket and payment processing
Validate ticket availability → Create booking record → Process payment → Update ticket quantities

CreateEventServlet:
Creates new events with input validation
Validate event details → Create event record → Initialize ticket types if provided

DeleteEventServlet:
Handles event cancellation and associated refunds
Update event status → Process refunds for all bookings → Update booking statuses

EventServlet:
Retrieves event information for display
Query events (all or single) → Join with ticket types → Return formatted event data

LoginServlet:
Handles user authentication and session creation
Validate credentials → Create session → Return user role and redirect

LogoutServlet:
Handles user session termination
Find session → Invalidate session → Redirect to login

PaymentMethodServlet:
Manages user payment methods (credit cards)
CRUD operations for credit cards → Validate card details → Update card status

PaymentServlet:
Processes payments and retrieves payment history
Process payment → Update booking status → Return payment confirmation

RegisterServlet:
Handles new user registration
Validate user details → Hash password → Create user record

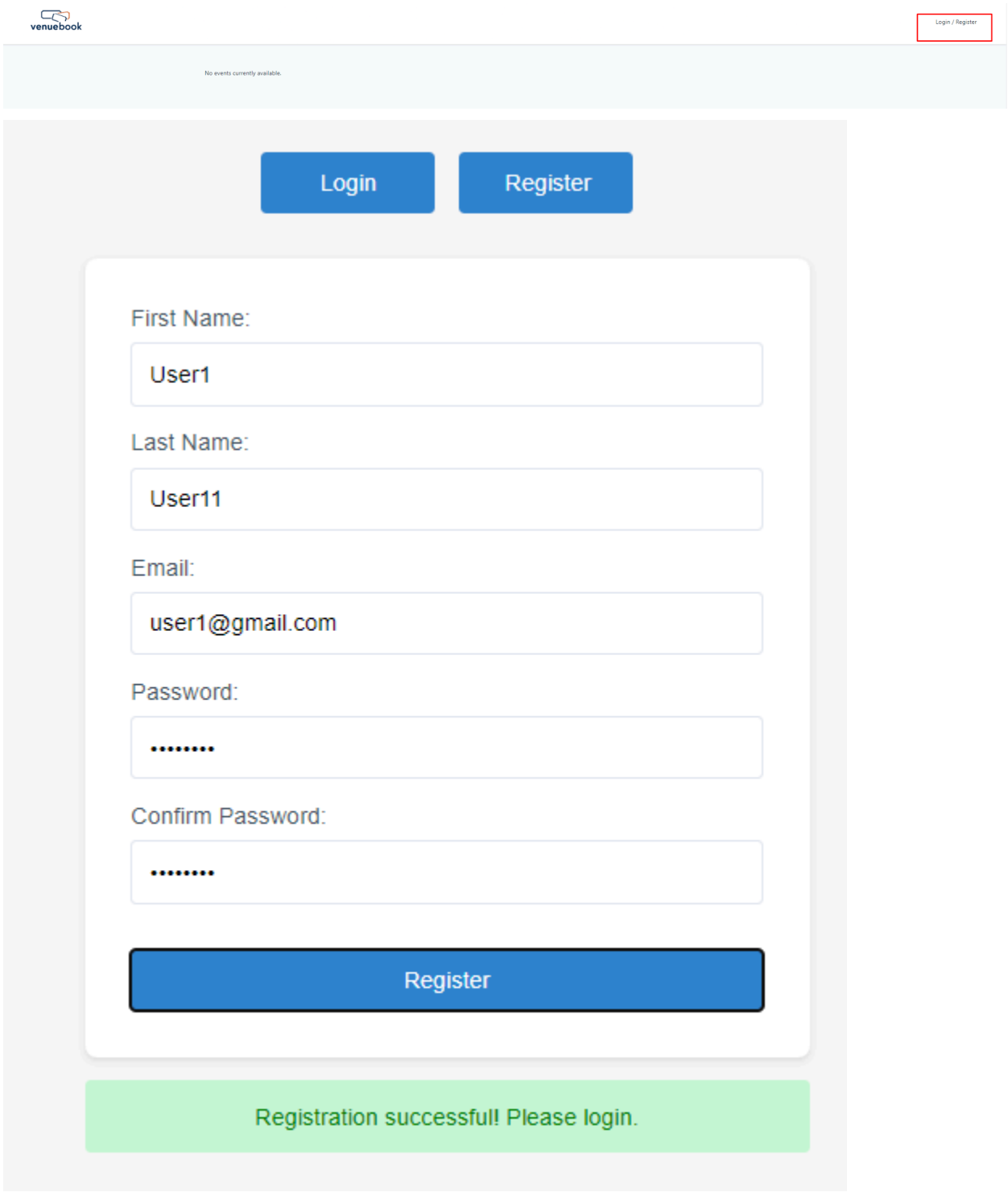
TicketTypeServlet:
Manages ticket types for events
Create/Update ticket types → Validate quantities → Update availability

UserProfileServlet:
Manages user profile information
Retrieve/Update user details → Manage user preferences → Return profile data

(There were no changes to the phase 1 report - part)

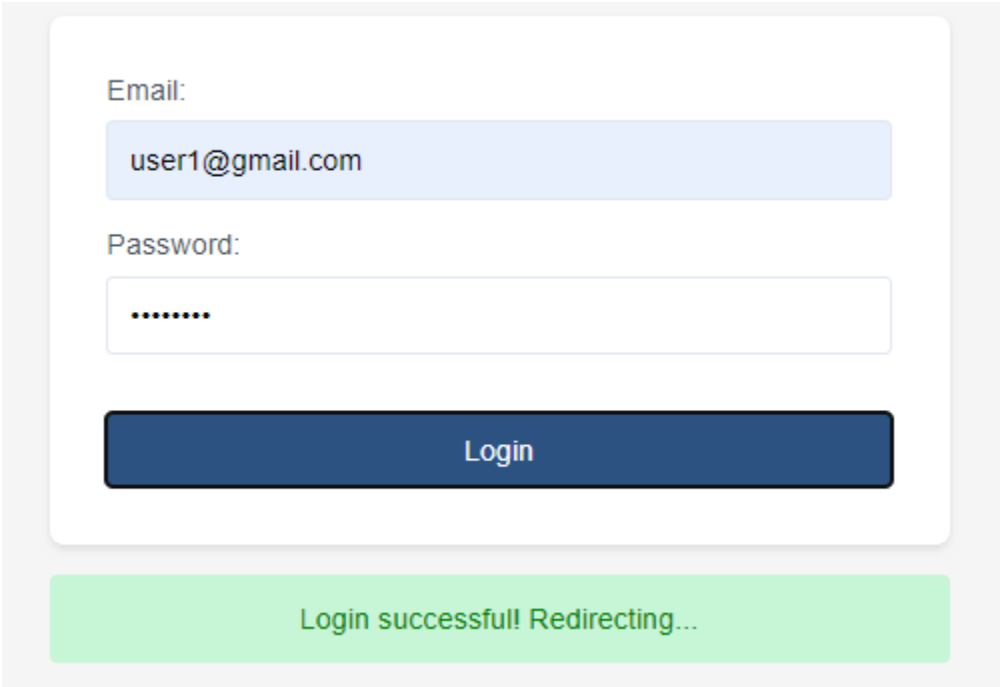
-EXAMPLES OF SERVICES WORKING

Register

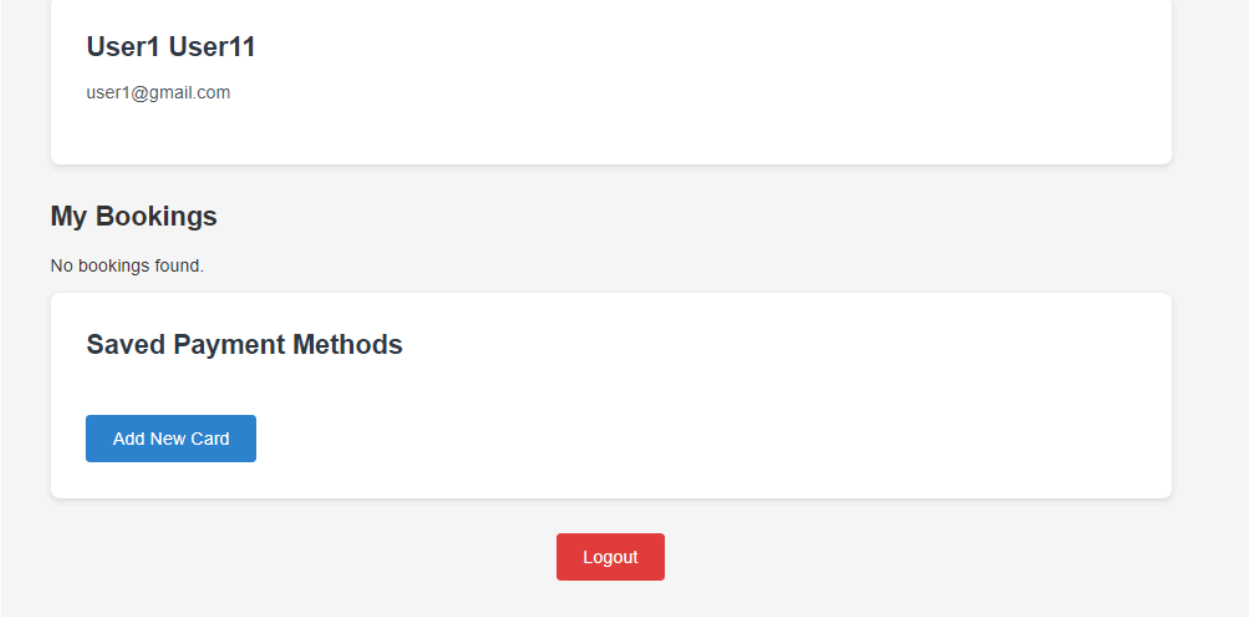


←T→	customer_id	first_name	last_name	email	password_hash
☐ Edit Copy Delete	11	User1	User11	user1@gmail.com	XohImNooBHFR0OVvjcYpJ3NgPQ1qq73WKhHvch0VQtg=

Login

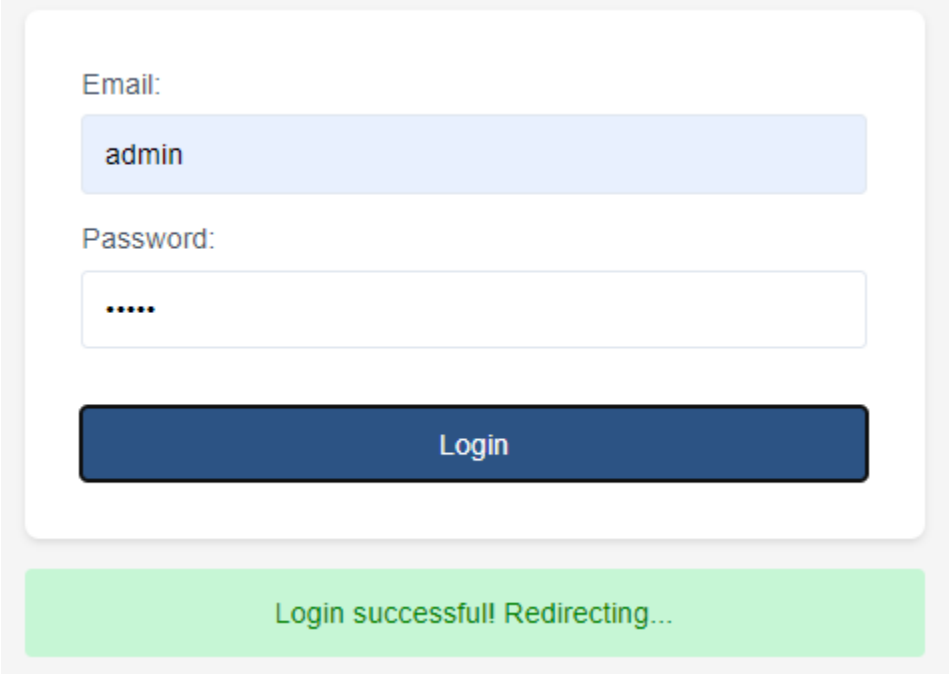


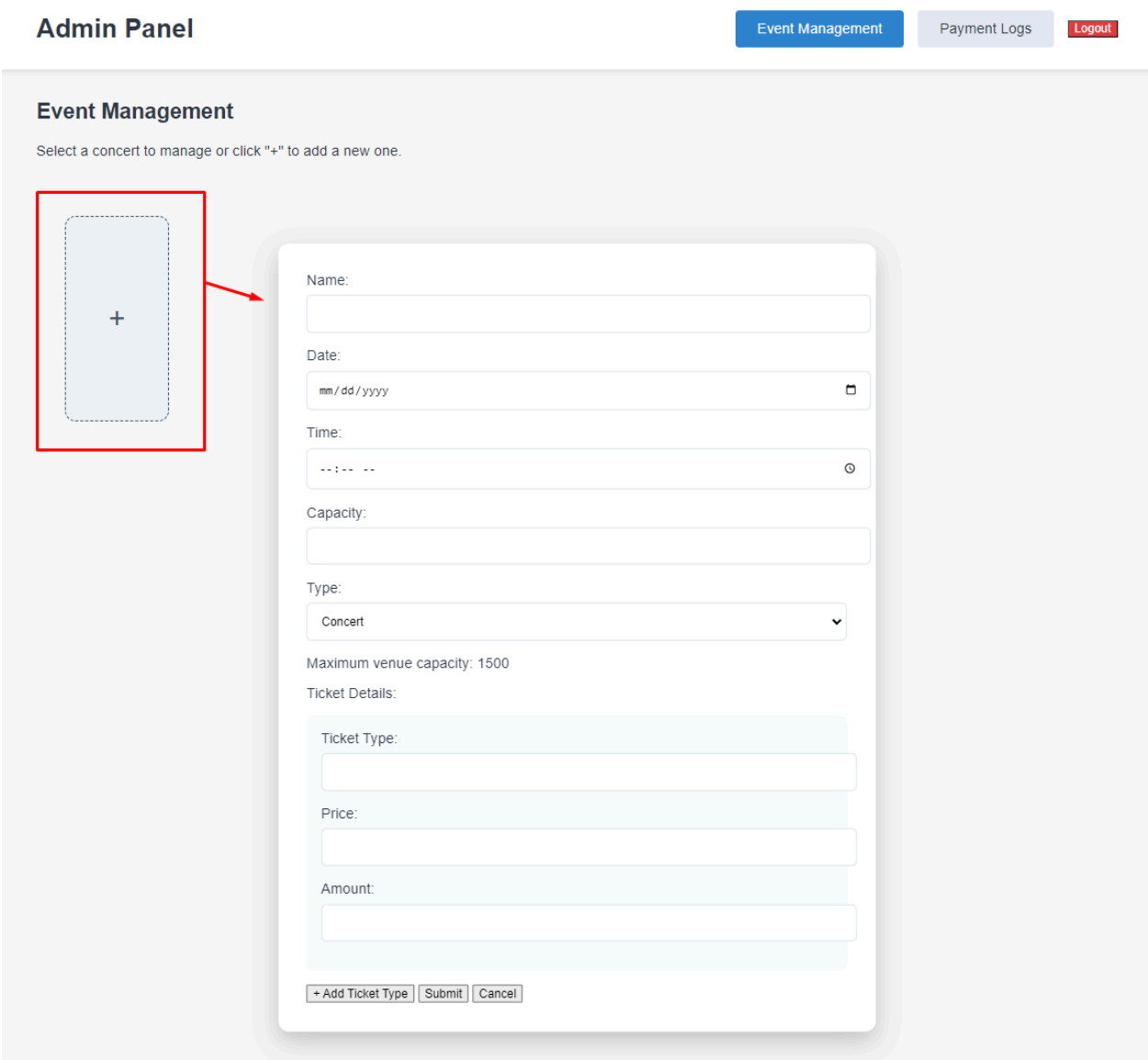
Now we can access the user tab



Event Creation / Ticket type creation

Logging into admin page




Admin Panel
Event Management
Payment Logs
Logout
Event Management
Select a concert to manage or click "+" to add a new one.
+
Name:
Date:
mm/dd/yyyy
Time:
--:-- --
Capacity:
Type:
Concert
Maximum venue capacity: 1500
Ticket Details:
Ticket Type:
Price:
Amount:
+ Add Ticket Type
Submit
Cancel

Name:

Event1

Date:

12/28/2024

Time:

08:07 PM

Capacity:

100

Type:

Concert

Maximum venue capacity: 1500

Ticket Details:

Ticket Type:

type1

Price:

10

Amount:

90

Ticket Type:

type2

Price:

20

Amount:

10

Remove

+ Add Ticket Type | Submit | Cancel

Event1

Date: 12/28/2024

Time: 20:07

Type: Concert

Capacity: 100

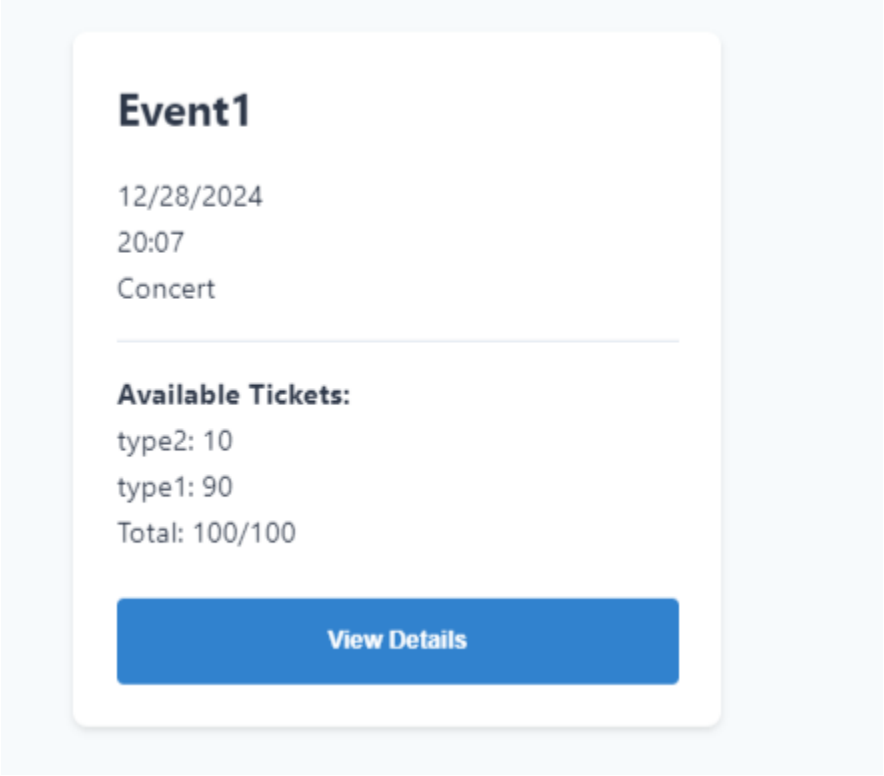
Ticket Types:

type2

←T→	event_id	name	event_type	event_date	event_time	venue_capacity	status
Edit Copy Delete	24	Event1	Concert	2024-12-28	20:07:00	100	active

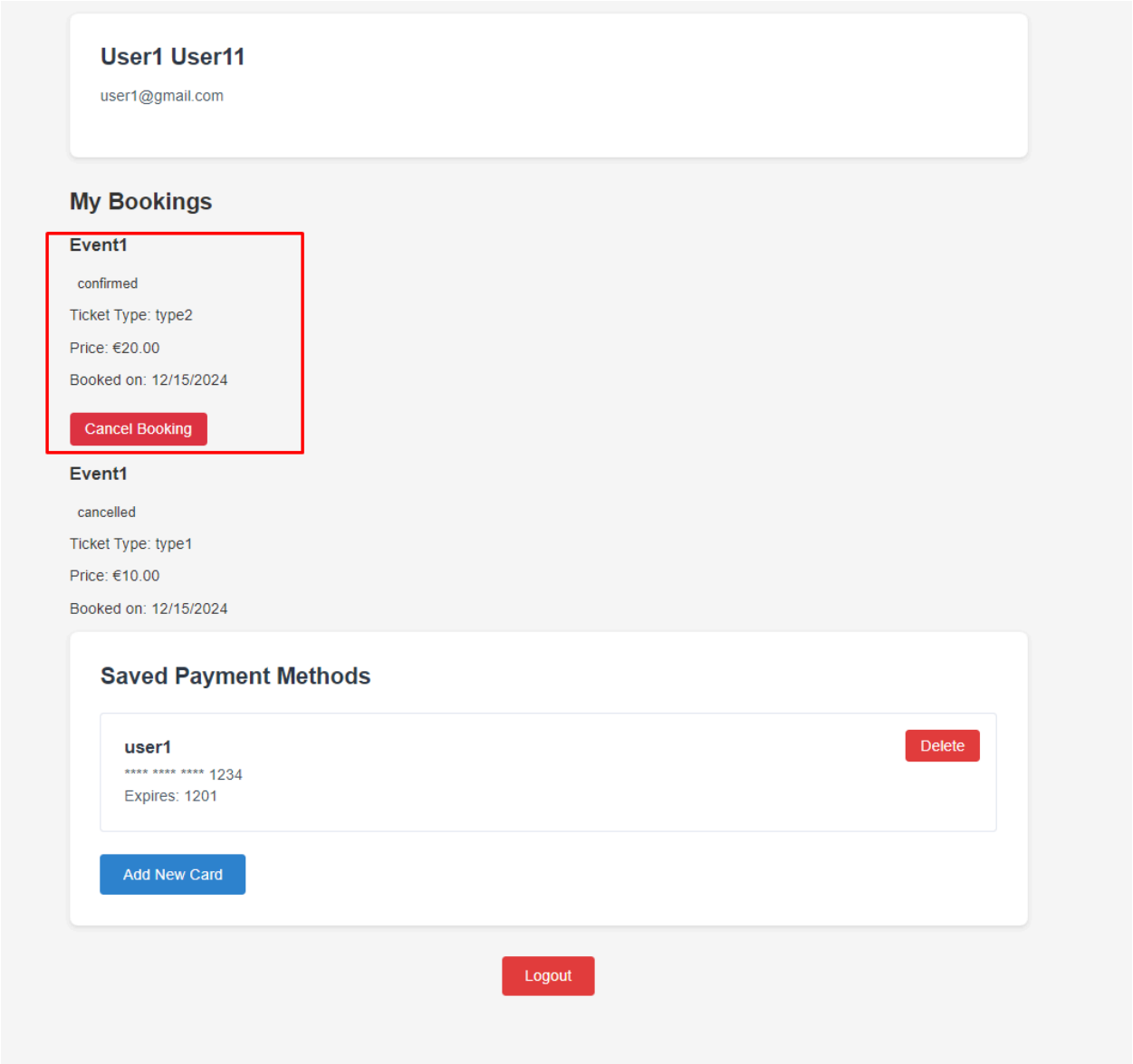
←T→	type_id	event_id	type_name	price	quantity_available
Edit Copy Delete	27	24	type1	10.00	90
Edit Copy Delete	28	24	type2	20.00	10

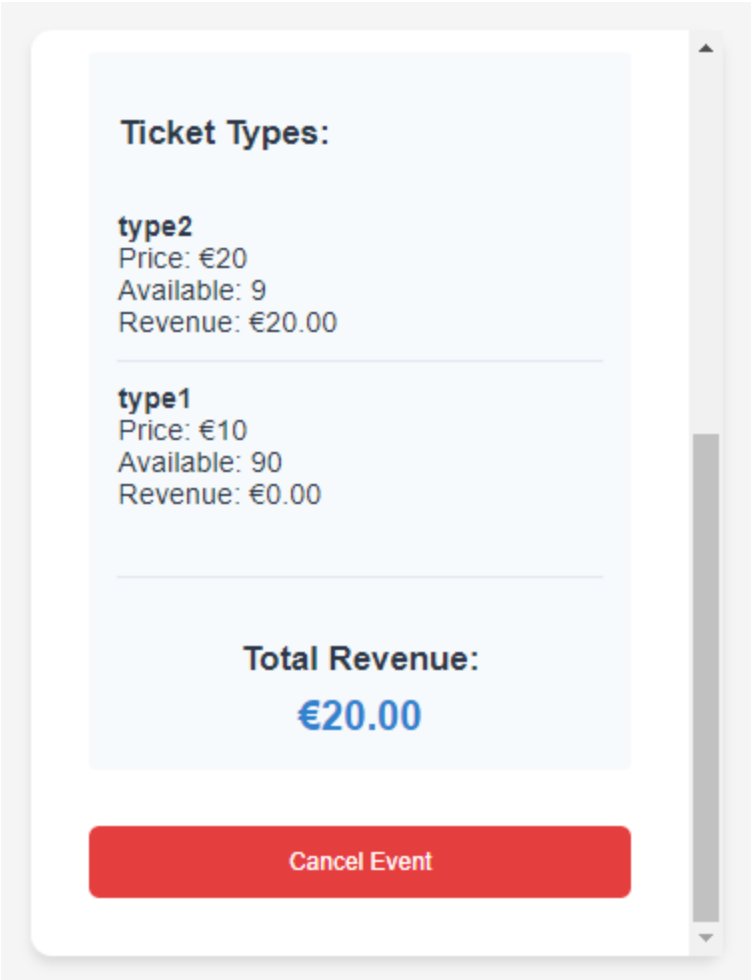
At index


Event1
12/28/2024
20:07
Concert
Available Tickets:
type2: 10
type1: 90
Total: 100/100
View Details

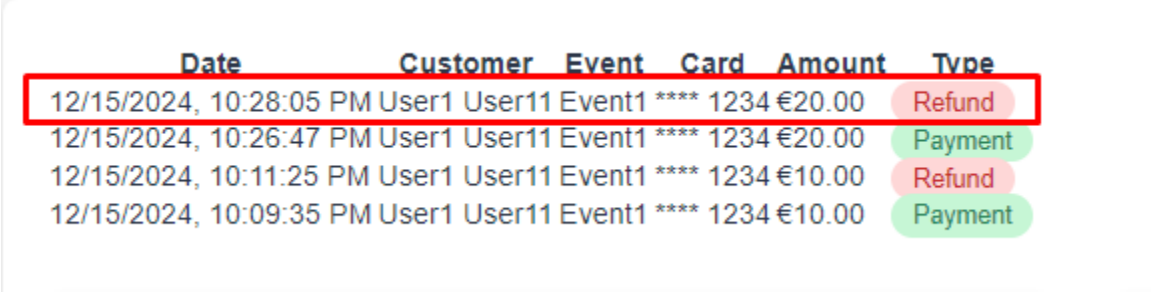
Event deletion / auto ticket refund

(with 1 ticket booked for the event)





Autorefund on the ticket



Date	Customer	Event	Card	Amount	Type
12/15/2024, 10:28:05 PM	User1 User11	Event1	**** 1234	€20.00	Refund
12/15/2024, 10:26:47 PM	User1 User11	Event1	**** 1234	€20.00	Payment
12/15/2024, 10:11:25 PM	User1 User11	Event1	**** 1234	€10.00	Refund
12/15/2024, 10:09:35 PM	User1 User11	Event1	**** 1234	€10.00	Payment

Booking

×

Event1

12/28/2024 20:07 Concert

Select Tickets

type1
€10.00
90 tickets available

type2
€20.00
10 tickets available

Book Now

Booking successful!

OK

×

Select Payment Method

**** **** **** 1234

Expires: 1201

Back to Tickets

Event1

12/28/2024
20:07
Concert

Available Tickets:

type2: 10
type1: 89
Total: 99/100

View Details

User1 User11

user1@gmail.com

My Bookings

Event1

confirmed
Ticket Type: type1
Price: €10.00
Booked on: 12/15/2024

Cancel Booking

Saved Payment Methods

user1
**** **** **** 1234
Expires: 1201

Delete

Add New Card

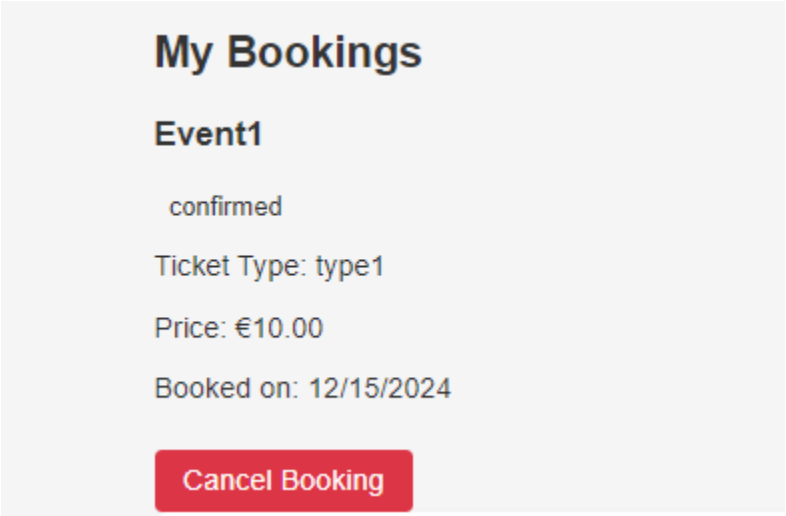
Logout

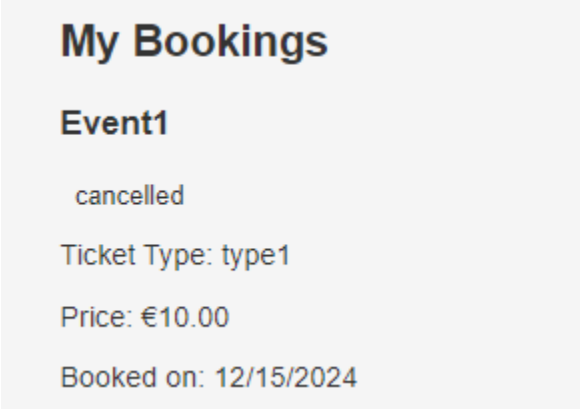
←T→	booking_id	customer_id	event_id	ticket_type_id	booking_date	status
☐ Edit Copy Delete	32	11	24	27	2024-12-15 20:09:35	confirmed

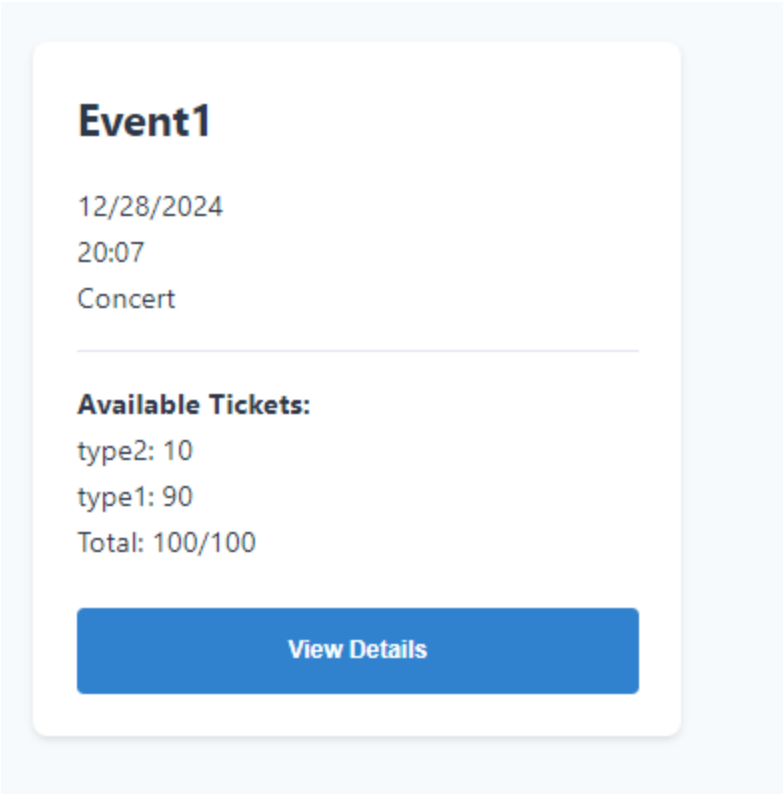
At payments tab

12/15/2024, 10:09:35 PM User1 User11 Event1 **** 1234 €10.00 Payment

Booking cancelation





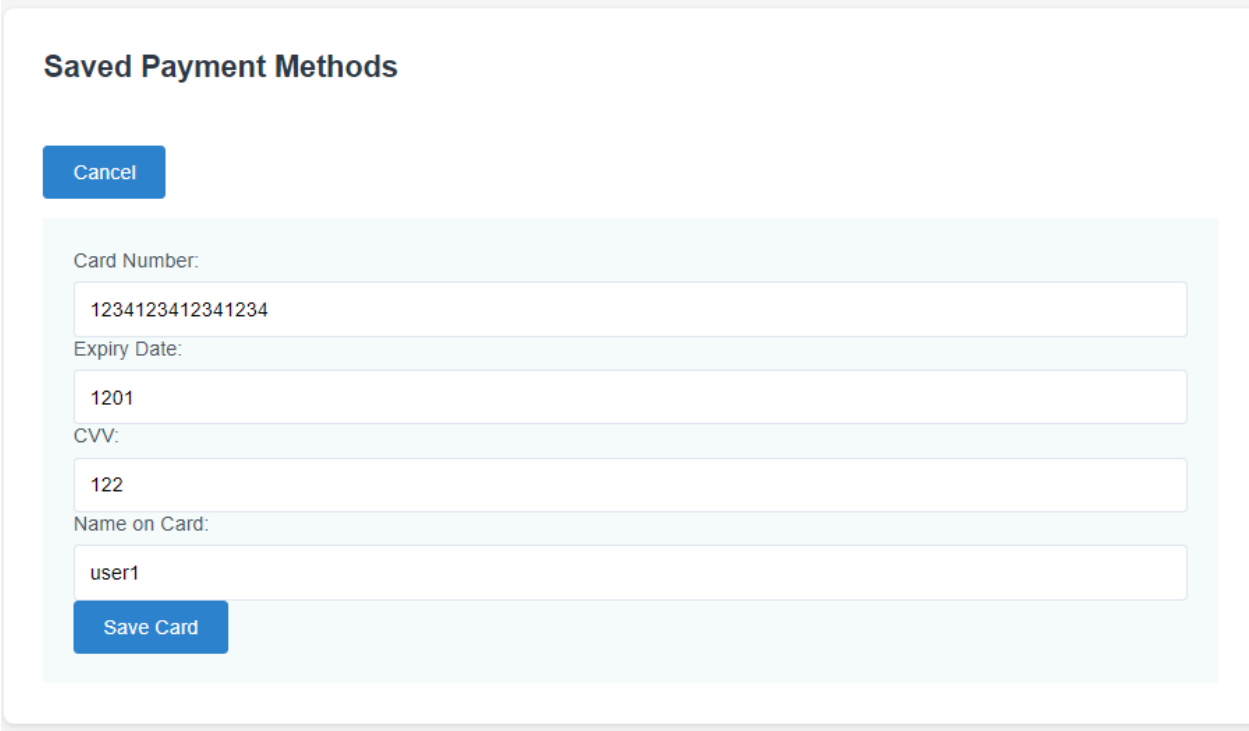


←T→	booking_id	customer_id	event_id	ticket_type_id	booking_date	status
Edit Copy Delete	32	11	24	27	2024-12-15 20:09:35	cancelled

At payments

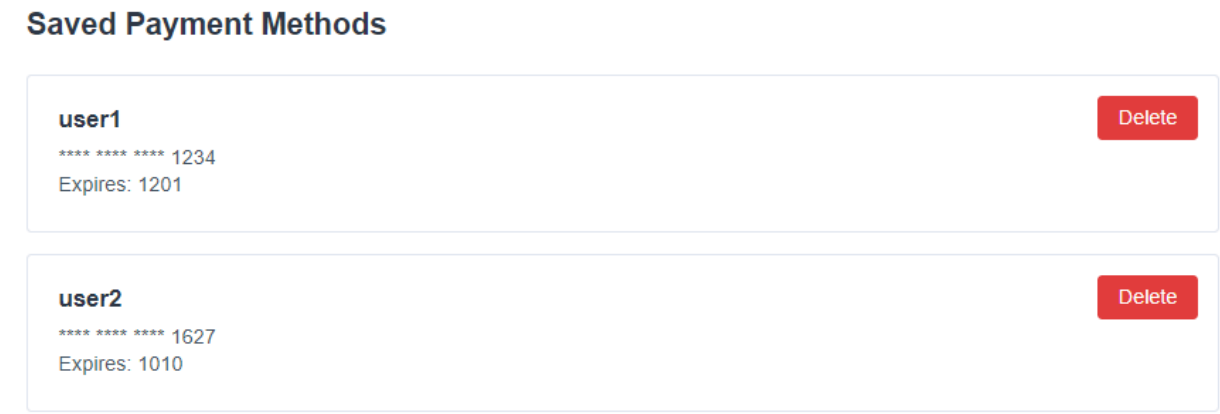
Date	Customer	Event	Card	Amount	Type
12/15/2024, 10:11:25 PM	User1 User11	Event1	**** 1234	€10.00	Refund
12/15/2024, 10:09:35 PM	User1 User11	Event1	**** 1234	€10.00	Payment

Card creation

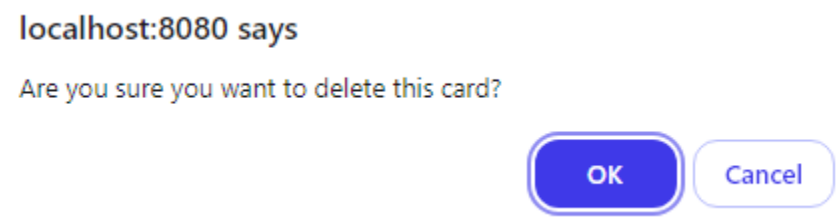


←T→	card_id	customer_id	card_holder	card_number	expiry_date	cvv	status
Edit Copy Delete	9	11	user1	1234123412341234	1201	122	active

Card deletion



←T→	card_id	customer_id	card_holder	card_number	expiry_date	cvv	status
Edit Copy Delete	9	11	user1	1234123412341234	1201	122	active
Edit Copy Delete	10	11	user2	1243526178271627	1010	111	active



←T→	card_id	customer_id	card_holder	card_number	expiry_date	cvv	status
Edit Copy Delete	9	11	user1	1234123412341234	1201	122	active

Admin statistics

Booked from the same user

←T→	booking_id	customer_id	event_id	ticket_type_id	booking_date	status
Edit Copy Delete	32	11	24	27	2024-12-15 20:09:35	cancelled
Edit Copy Delete	33	11	24	28	2024-12-15 20:26:47	cancelled
Edit Copy Delete	34	11	25	29	2024-12-15 20:36:46	confirmed
Edit Copy Delete	35	11	26	31	2024-12-15 20:36:50	confirmed
Edit Copy Delete	36	11	26	31	2024-12-15 20:36:54	confirmed
Edit Copy Delete	37	11	27	32	2024-12-15 20:36:58	confirmed
Edit Copy Delete	38	11	27	32	2024-12-15 20:37:01	confirmed
Edit Copy Delete	39	11	27	32	2024-12-15 20:37:05	confirmed
Edit Copy Delete	40	11	27	32	2024-12-15 20:37:10	confirmed
Edit Copy Delete	41	11	25	30	2024-12-15 20:37:14	confirmed
Edit Copy Delete	42	11	25	30	2024-12-15 20:37:17	confirmed
Edit Copy Delete	43	11	25	30	2024-12-15 20:37:21	confirmed

Ticket Types:

1
Price: €5
Available: 4
Revenue: €5.00

2
Price: €10
Available: 2
Revenue: €30.00

Total Revenue:
€35.00

Cancel Event

Capacity: 15

Ticket Types:

11
Price: €15
Available: 13
Revenue: €30.00

Total Revenue:
€30.00

Cancel Event

Capacity: 100

Ticket Types:

standard
Price: €5
Available: 96
Revenue: €20.00

Total Revenue:
€20.00

Cancel Event

Date	Customer	Event	Card	Amount	Type
12/15/2024, 10:37:21 PM	User1 User11	Event2	**** 1234	€10.00	Payment
12/15/2024, 10:37:17 PM	User1 User11	Event2	**** 1234	€10.00	Payment
12/15/2024, 10:37:14 PM	User1 User11	Event2	**** 1234	€10.00	Payment
12/15/2024, 10:37:10 PM	User1 User11	Event4	**** 1234	€5.00	Payment
12/15/2024, 10:37:05 PM	User1 User11	Event4	**** 1234	€5.00	Payment
12/15/2024, 10:37:01 PM	User1 User11	Event4	**** 1234	€5.00	Payment
12/15/2024, 10:36:58 PM	User1 User11	Event4	**** 1234	€5.00	Payment
12/15/2024, 10:36:54 PM	User1 User11	Event3	**** 1234	€15.00	Payment
12/15/2024, 10:36:50 PM	User1 User11	Event3	**** 1234	€15.00	Payment
12/15/2024, 10:36:46 PM	User1 User11	Event2	**** 1234	€5.00	Payment
12/15/2024, 10:28:05 PM	User1 User11	Event1	**** 1234	€20.00	Refund
12/15/2024, 10:26:47 PM	User1 User11	Event1	**** 1234	€20.00	Payment
12/15/2024, 10:11:25 PM	User1 User11	Event1	**** 1234	€10.00	Refund
12/15/2024, 10:09:35 PM	User1 User11	Event1	**** 1234	€10.00	Payment

Revenue Summary

Total Payments: €115.00

Total Refunds: €30.00

Net Revenue: €85.00

Event Performance

Most Popular: Event2 (4 active bookings)

Highest Revenue: Event2 (€35.00)

Booking Analysis

12/01/2024 to 01/16/2025 Analyze Bookings

Event Bookings:

Event2 (12/18/2024)	4 bookings
Event3 (12/27/2024)	2 bookings

Most Profitable Event:

Event2

Revenue: €35.00

- INSTALLATION GUIDE

- Required Software:
  - XAMPP (with MySQL 8.0.0)
  - JDK 8 or higher
  - Maven 3.8 or higher
  - VS Code
- Install Java:

  sudo apt update

  sudo apt install openjdk-8-jdk

  update-alternatives --list java

  export JAVA_HOME= dirname

  export PATH=$JAVA_HOME/bin:$PATH
- Install Maven:

  sudo apt install maven
- Start XAMPP (as admin):
  - Start Apache & MySQL.
- Import or Recreate Database via phpMyAdmin:
  - Export to `.sql` file.
  - Import into phpMyAdmin.
- Build and Deploy Application:

  1. Navigate to the project directory:

     cd hy360-project-2024

  2. Clean and package the project:

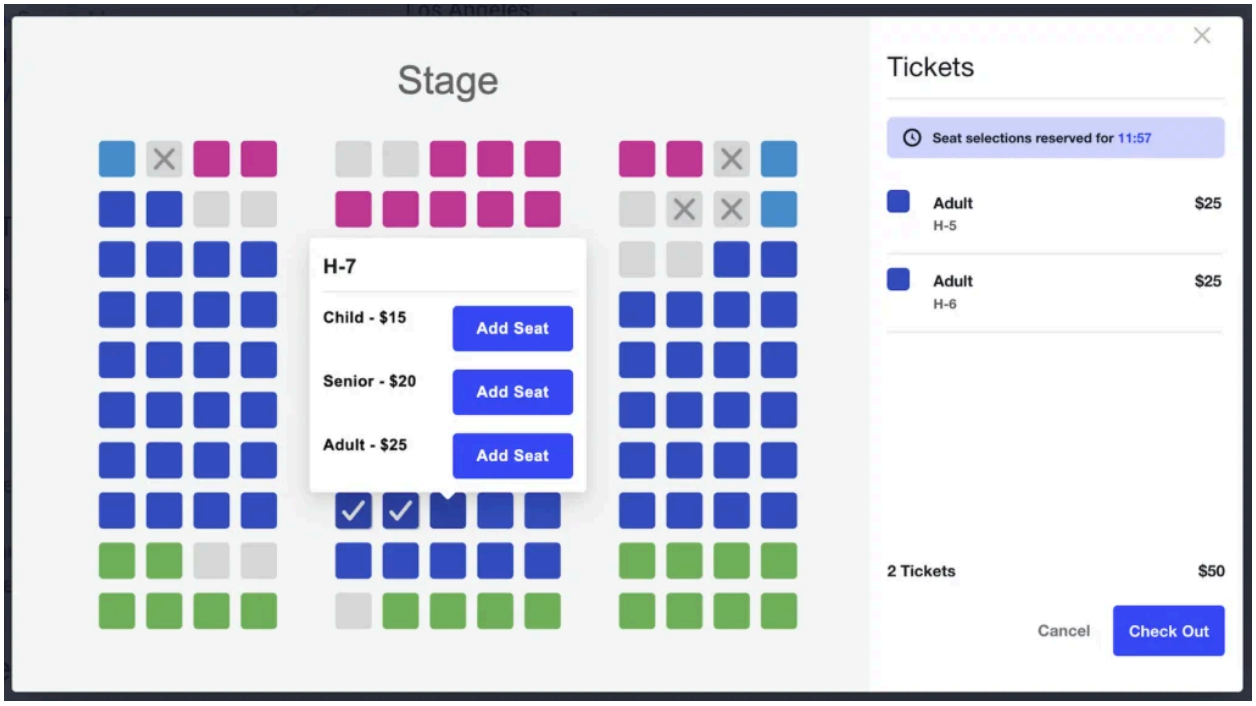
mvn clean package

3. Go to Tomcat Admin.
4. List Applications.
5. Undeploy the old version.
6. Select the WAR file to upload.
7. Deploy and then click on the application.

-Restrictions + improvements

The project is 100% covering the requests of the assignment plus some extra payment handling , if we would add something more in the project would be closing tickets like real event sites , so choosing the exact seat in a event seat map

Example image



Also make a system where events get cancelled past their actual date and appear as “past date events”