CS306 Database Systems

Project Phase III Report

Group Number: 66

Group Members:

1. Mert Güngör - 34159
2. Kaan Berk Karabıyık - 34424

1. Introduction

In this project (Phase 3), we developed a web application for an Airline Operations System. We integrated our database with a web interface using PHP. The project allows users to interact with MySQL triggers and stored procedures via a website. We also built a support ticket system using MongoDB.

The project structure has two main folders:

- **User Side:** For testing triggers, procedures, and creating support tickets.
- **Admin Side:** For managing and replying to support tickets.

2. MySQL Triggers

We integrated two triggers into the web interface. Users can test different scenarios using buttons on the web pages.

2.1. Trigger 1: Check Capacity Before Insert

Responsible Member: Mert Güngör

Description:

This trigger prevents a booking if the flight is full. It checks the aircraft capacity before inserting a new row into the Booking table.

SQL Script:

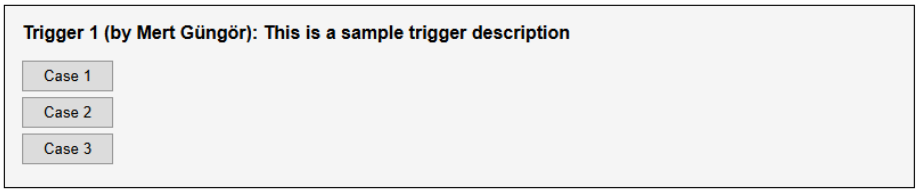
```SQL

CREATE TRIGGER check_capacity_before_insert

BEFORE INSERT ON Booking
```

```
FOR EACH ROW

BEGIN

    DECLARE current_passengers INT;

    DECLARE max_capacity INT;


    SELECT A.capacity INTO max_capacity

    FROM Flight F

    JOIN Aircraft A ON F.aircraft_id_fk = A.aircraft_id

    WHERE F.flight_number = NEW.flight_number_fk;


    SELECT COUNT(*) INTO current_passengers

    FROM Booking

    WHERE flight_number_fk = NEW.flight_number_fk;


    IF current_passengers >= max_capacity THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'ERROR: Flight is full! Cannot accept
new booking.';

    END IF;

END
```

Web Interface Cases:

- **Case 1 Button:** Tries to add a booking to a flight that has empty seats. The result is successful.
- **Case 2 Button:** Shows a table with the capacity and current number of passengers for each flight.

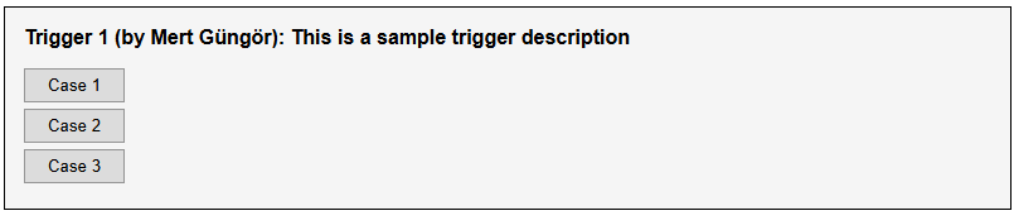
- **Case 3 Button:** Deletes the test booking so we can test it again.



Go to homepage

Result: Showing current flight capacity status

Flight	Capacity	Booked	Available
AA100	248	1	247
AF1390	381	1	380
BA247	469	2	467
EK202	489	1	488
KL1613	408	1	407
LH130	410	1	409
QR240	354	0	354
TK001	349	1	348
TK002	288	1	287
UA110	350	1	349



Go to homepage

Result: SUCCESS: Booking added to flight QR240 (has space available)

after case 1 (photo in the below)

Go to homepage

Result: Showing current flight capacity status

Flight	Capacity	Booked	Available
AA100	248	1	247
AF1390	381	1	380
BA247	469	2	467
EK202	489	1	488
KL1613	408	1	407
LH130	410	1	409
QR240	354	1	353
TK001	349	1	348
TK002	288	1	287
UA110	350	1	349

2.2. Trigger 2: Log Booking Delete

Responsible Member: Kaan Berk Karabıyık

Description:

This trigger automatically saves a record to the Booking_Log table when a booking is deleted. It helps us track cancelled reservations.

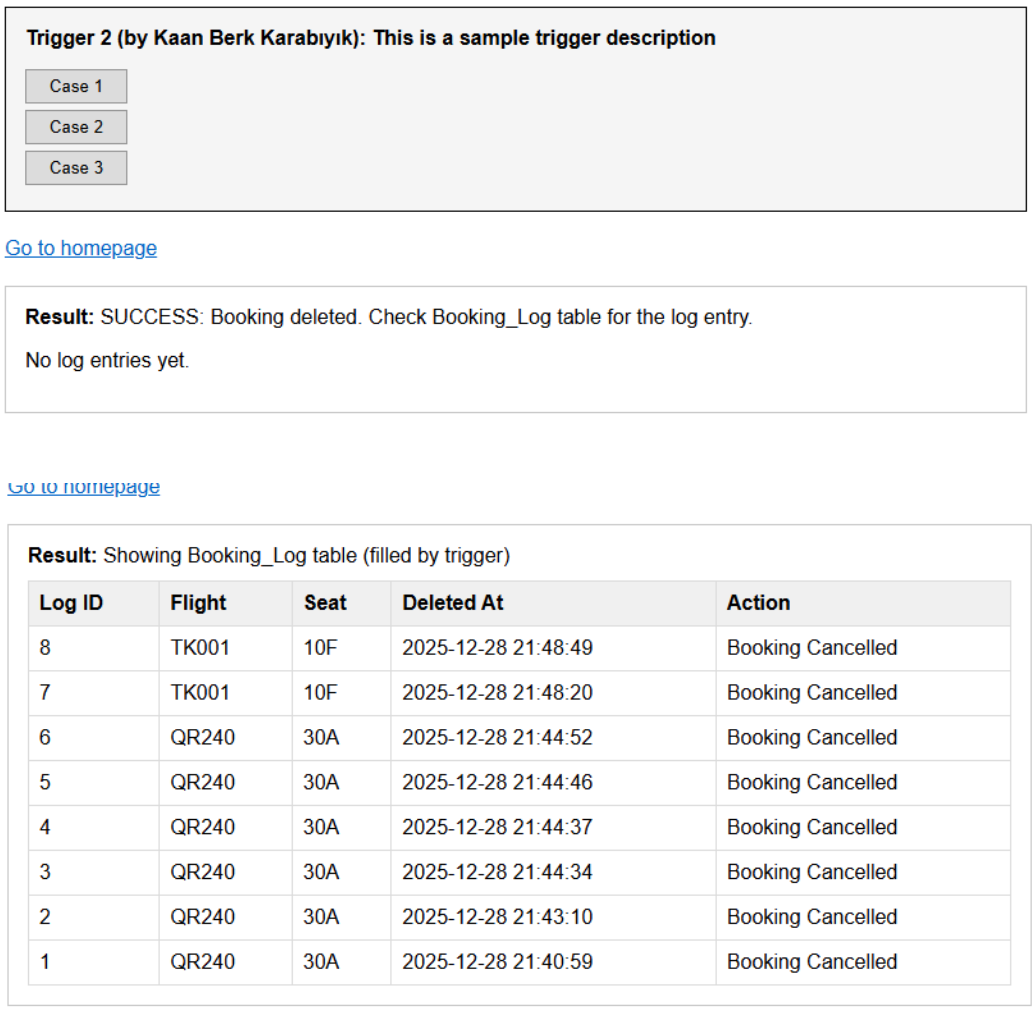
SQL Script:

SQL

```
CREATE TRIGGER log_booking_delete

AFTER DELETE ON Booking

FOR EACH ROW

BEGIN

    INSERT INTO Booking_Log (flight_number, seat_number,
deleted_at, user_action)

    VALUES (OLD.flight_number_fk, OLD.seat_number, NOW(),
'Booking Cancelled');

END
```

Web Interface Cases:

- **Case 1 Button:** Deletes a specific booking from the database to activate the trigger.
- **Case 2 Button:** Shows the Booking_Log table. We can see the deleted booking details here.
- **Case 3 Button:** Restores the deleted booking to the database for future tests.



3. MySQL Stored Procedures

We integrated two stored procedures. Users can enter parameters into input boxes to call these procedures.

3.1. Stored Procedure 1: Get Passenger Manifest

Responsible Member: Mert Güngör

Description:

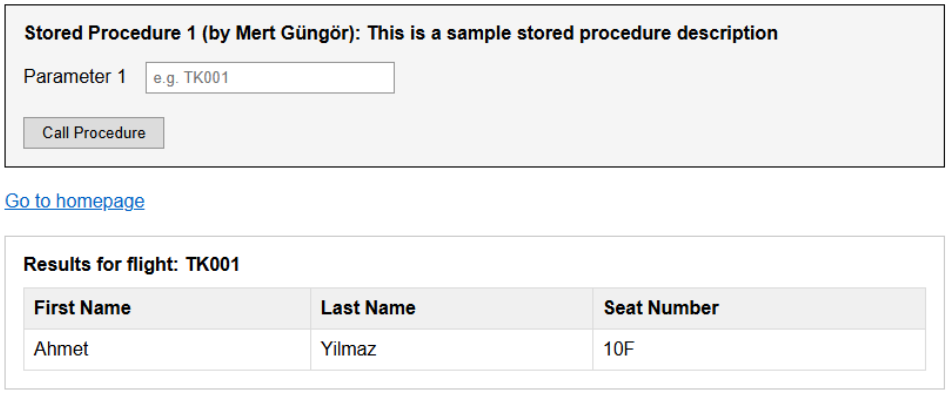
This procedure lists all passengers and their seat numbers for a specific flight number.

SQL Script:

```sql
CREATE PROCEDURE GetPassengerManifest(IN flightNo VARCHAR(10))
BEGIN
    SELECT P.first_name, P.last_name, B.seat_number
    FROM Booking B
    JOIN Passenger P ON B.passenger_id_fk = P.passenger_id
    WHERE B.flight_number_fk = flightNo;
END
```

Web Interface Inputs:

- **Parameter 1 (Input Box):** Flight Number (Example: `TK001`)
- **Button:** When clicked, it displays a table with passenger names and seats.



3.2. Stored Procedure 2: Schedule Flight

Responsible Member: Kaan Berk Karabıyık

Description:

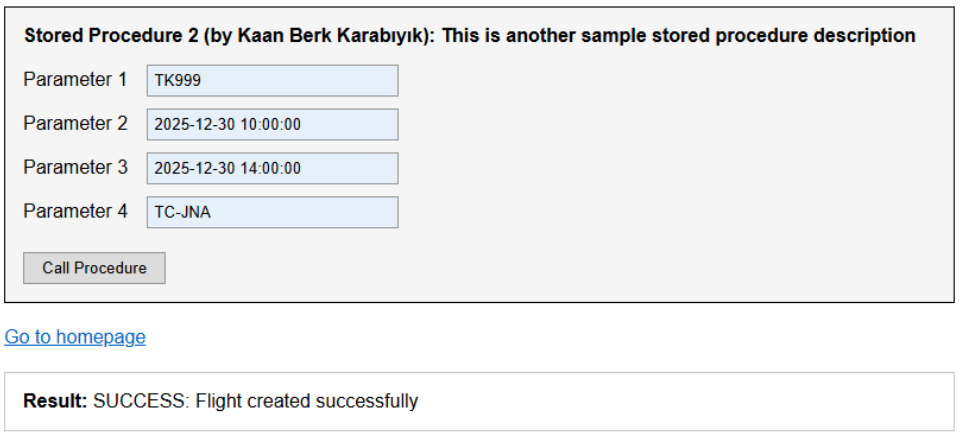
This procedure creates a new flight in the database. It takes flight details as input and saves them.

SQL Script:

```SQL

CREATE PROCEDURE ScheduleFlight(
    IN f_num VARCHAR(10),
    IN f_dep DATETIME,
    IN f_arr DATETIME,
    IN f_plane VARCHAR(10)
)
BEGIN
    INSERT INTO Flight (flight_number, departure_time, arrival_time, airline_code_fk, aircraft_id_fk, origin_airport_fk, dest_airport_fk)
    VALUES (f_num, f_dep, f_arr, 'THY', f_plane, 'IST', 'LHR');

    SELECT 'Flight created successfully' as status;
END
```

Web Interface Inputs:

- **Parameter 1:** Flight Number (e.g., TK999)
- **Parameter 2:** Departure Time (e.g., 2025-12-20 10:00:00)
- **Parameter 3:** Arrival Time (e.g., 2025-12-20 14:00:00)
- **Parameter 4:** Aircraft ID (e.g., TC-JNA)



4. MongoDB Support Ticket System

We used MongoDB to create a support ticket system. We used the `MongoDB\Driver\Manager` class in PHP to connect to the database.

Workflow:

1. **User:** Creates a ticket with a username and message.
2. **User:** Views active tickets and adds comments.
3. **Admin:** Sees all active tickets, replies, and resolves them.

Below are the PHP scripts corresponding to the MongoDB queries we used.

4.1. Reading Tickets (Read Operation)

We use this query to find active tickets (`status: true`) for a user or for the admin.

PHP Script:

```php
$filter = [];
if ($onlyActive) {
    $filter['status'] = true;
}
if ($username) {
    $filter['username'] = $username;
```

```
}

$query = new MongoDB\Driver\Query($filter);

$cursor = $manager->executeQuery("support_tickets.tickets",
$query);
```

4.2. Creating a Ticket (Insert Operation)

When a user submits the form, we insert a new document with the current date and empty comments array.

PHP Script:

```php

$bulk = new MongoDB\Driver\BulkWrite();

$document = [

    'username' => $username,

    'message' => $message,

    'created_at' => date('Y-m-d H:i:s'),

    'status' => true,

    'comments' => []

];

$bulk->insert($document);

$manager->executeBulkWrite("support_tickets.tickets", $bulk);
```

4.3. Adding a Comment (Update Operation - Push)

When a user or admin replies, we use `$push` to add the comment to the `comments` array inside the document.

PHP Script:

PHP

```php
$bulk = new MongoDB\Driver\BulkWrite();
$commentData = [
    'username' => $author,
    'comment' => $comment,
    'created_at' => date('Y-m-d H:i:s')
];
$bulk->update(
    ['_id' => new MongoDB\BSON\ObjectId($ticketId)],
    ['$push' => ['comments' => $commentData]]
);
$manager->executeBulkWrite("support_tickets.tickets", $bulk);
```

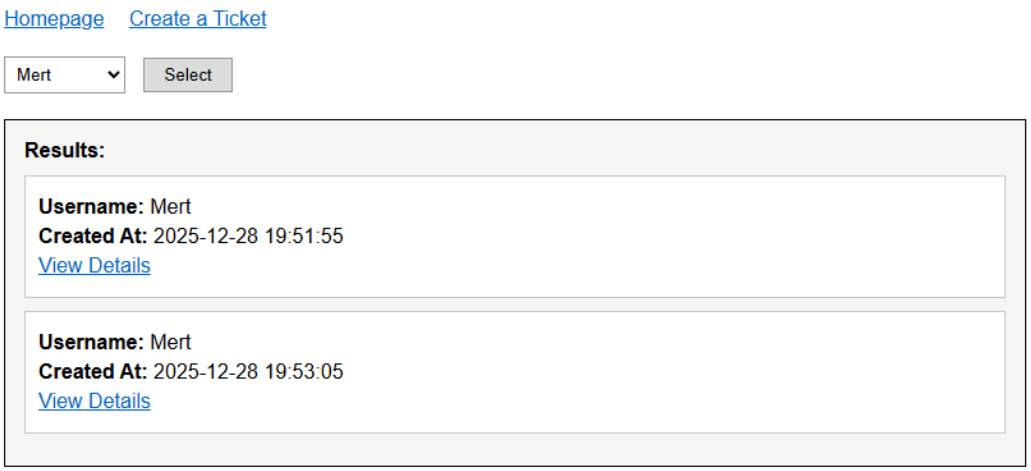
4.4. Resolving a Ticket (Update Operation - Set)

Admins can resolve a ticket. We use `$set` to change the `status` field to `false`.

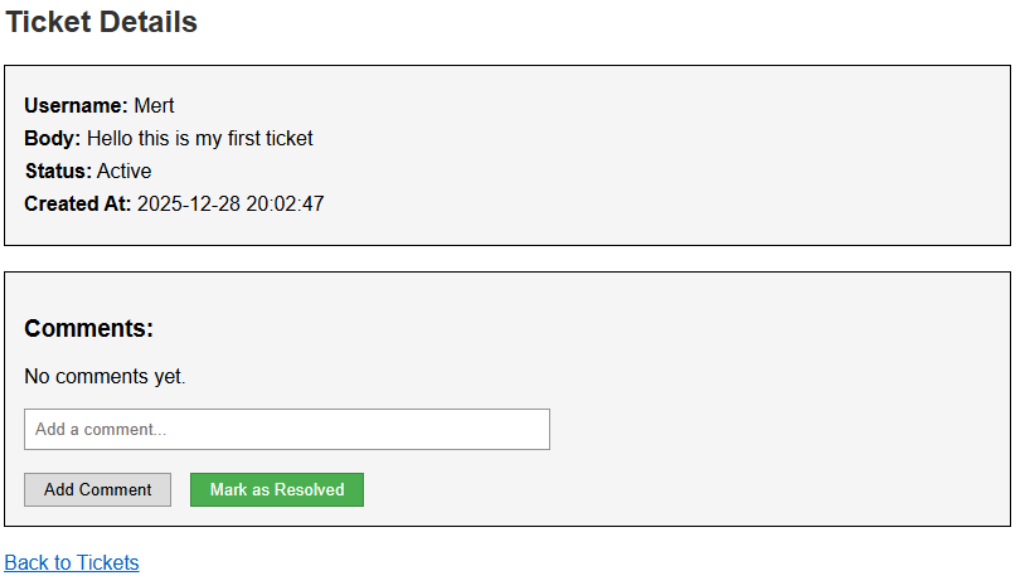
PHP Script:

PHP

```php
$bulk = new MongoDB\Driver\BulkWrite();
$bulk->update(
    ['_id' => new MongoDB\BSON\ObjectId($ticketId)],
    ['$set' => ['status' => false]]
);
$manager->executeBulkWrite("support_tickets.tickets", $bulk);
```



Admin view(below images)



5. Assumptions, Challenges, and Design Decisions

- **Challenge: MongoDB Driver Configuration:** The most significant challenge was integrating MongoDB with XAMPP on Windows. We had to ensure the `php_mongodb.dll` version matched our PHP version (Thread Safe vs. Non-Thread Safe) and manually edit the `php.ini` file to enable the extension.

Challenge: SQL Import and Constraints: During the database import of `CS306_GROUP_66_PHASE3_SQLDUMP.sql`, we encountered compatibility issues with `CHECK` constraints (Error #1901) and internal phpMyAdmin user warnings ("Access denied for user 'pma'"). The MariaDB version used by XAMPP did not support the specific syntax for column comparisons (e.g., `arrival_time > departure_time`) defined in the `Flight` and `Aircraft` tables.

- **Solution:** To resolve this, we manually edited the SQL dump file to remove the incompatible `CHECK` constraint lines from the `CREATE TABLE` statements. We determined that the `pma` user warning was unrelated to our project schema and disregarded it. After dropping the partially created tables to ensure a clean state, we successfully re-imported the modified SQL script without errors.

- **Design Decision: Low-Level MongoDB Driver:** Although high-level libraries (like the pure MongoDB PHP Library) offer easier syntax, we utilized the low-level `MongoDB\Driver\Manager` class for database connections and bulk writes. This decision was made to strictly adhere to the project implementation requirements.
- **Assumption: Data Integrity:** We assumed that the `Aircraft` capacity in the relational database remains constant during the booking process. The trigger logic relies on the static capacity value from the `Aircraft` table to validate insertions into the `Booking` table.
- **Assumption: Input Validation:** For the stored procedure `ScheduleFlight`, we assumed that the user (admin) provides valid airport codes (e.g., 'IST', 'LHR') that already exist in the `Airport` table, as the foreign key constraints will reject invalid codes.

6. Conclusion

In this project, we successfully integrated SQL and NoSQL databases into a web application. We learned how to trigger database events from a web page and how to manage document-based data with MongoDB in PHP. All requirements, including triggers, stored procedures, and the ticket system, are fully functional.

During the development of Phase 3, we encountered several technical challenges and made specific design decisions to ensure the system met the requirements:

Appendices:

Appendix A: SQL Dump Below is the complete SQL script used to create the relational schema, insert sample data, and define the triggers and stored procedures.

```sql
--
-- ========================================================================

-- CS306 GROUP 66 - PHASE 3 SQL DUMP

-- Airline Operations and Booking System

--
-- ========================================================================


SET FOREIGN_KEY_CHECKS = 0;


-- Drop tables if exist

DROP TABLE IF EXISTS Flight_Crew;

DROP TABLE IF EXISTS Booking_Log;

DROP TABLE IF EXISTS Booking;

DROP TABLE IF EXISTS Flight;

DROP TABLE IF EXISTS Pilot;

DROP TABLE IF EXISTS Aircraft;

DROP TABLE IF EXISTS Passenger;

DROP TABLE IF EXISTS Airline;

DROP TABLE IF EXISTS Airport;


SET FOREIGN_KEY_CHECKS = 1;
```

```sql
SET NAMES utf8mb4;
SET time_zone = '+03:00';


--
================================================================
-- TABLE CREATION
--
================================================================


-- Airport Table
CREATE TABLE Airport (
  airport_code VARCHAR(3) PRIMARY KEY,
  name         VARCHAR(100) NOT NULL,
  city         VARCHAR(50),
  country      VARCHAR(50)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


-- Airline Table
CREATE TABLE Airline (
  airline_code VARCHAR(3) PRIMARY KEY,
  name         VARCHAR(100) NOT NULL UNIQUE,
  hub_airport  VARCHAR(3) NULL,
  CONSTRAINT fk_airline_hub
    FOREIGN KEY (hub_airport) REFERENCES Airport(airport_code)
```

```
    ON UPDATE CASCADE
    ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Passenger Table
CREATE TABLE Passenger (
  passenger_id    INT PRIMARY KEY AUTO_INCREMENT,
  first_name      VARCHAR(50)  NOT NULL,
  last_name       VARCHAR(50)  NOT NULL,
  email           VARCHAR(100) NOT NULL UNIQUE,
  passport_number VARCHAR(20)  UNIQUE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Aircraft Table
CREATE TABLE Aircraft (
  aircraft_id     VARCHAR(10) PRIMARY KEY,
  model           VARCHAR(50) NOT NULL,
  capacity        INT         NOT NULL,
  airline_code_fk VARCHAR(3)  NOT NULL,
  CONSTRAINT fk_aircraft_airline
    FOREIGN KEY (airline_code_fk) REFERENCES
Airline(airline_code)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


-- Pilot Table
CREATE TABLE Pilot (
  pilot_id        INT PRIMARY KEY AUTO_INCREMENT,
  first_name      VARCHAR(50) NOT NULL,
  last_name       VARCHAR(50) NOT NULL,
  license_number  VARCHAR(20) NOT NULL UNIQUE,
  airline_code_fk VARCHAR(3)  NOT NULL,
  CONSTRAINT fk_pilot_airline
    FOREIGN KEY (airline_code_fk) REFERENCES
Airline(airline_code)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


-- Flight Table
CREATE TABLE Flight (
  flight_number     VARCHAR(10) PRIMARY KEY,
  departure_time    DATETIME NOT NULL,
  arrival_time      DATETIME NOT NULL,
  airline_code_fk   VARCHAR(3)  NOT NULL,
  aircraft_id_fk    VARCHAR(10) NOT NULL,
  origin_airport_fk VARCHAR(3)  NOT NULL,
```

```
  dest_airport_fk   VARCHAR(3)  NOT NULL,
  CONSTRAINT fk_flight_airline
    FOREIGN KEY (airline_code_fk)   REFERENCES
Airline(airline_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_flight_aircraft
    FOREIGN KEY (aircraft_id_fk)    REFERENCES
Aircraft(aircraft_id)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_flight_origin
    FOREIGN KEY (origin_airport_fk) REFERENCES
Airport(airport_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_flight_dest
    FOREIGN KEY (dest_airport_fk)   REFERENCES
Airport(airport_code)
    ON UPDATE CASCADE ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


-- Booking Table
CREATE TABLE Booking (
  passenger_id_fk  INT,
  flight_number_fk VARCHAR(10),
  seat_number      VARCHAR(4) NOT NULL,
  booking_date     DATE,
```

```
  PRIMARY KEY (passenger_id_fk, flight_number_fk),

  CONSTRAINT uq_booking_seat UNIQUE (flight_number_fk,
seat_number),

  CONSTRAINT fk_booking_passenger

    FOREIGN KEY (passenger_id_fk)  REFERENCES
Passenger(passenger_id)

    ON UPDATE CASCADE ON DELETE RESTRICT,

  CONSTRAINT fk_booking_flight

    FOREIGN KEY (flight_number_fk) REFERENCES
Flight(flight_number)

    ON UPDATE CASCADE ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


-- Flight_Crew Table

CREATE TABLE Flight_Crew (

  pilot_id_fk       INT,

  flight_number_fk  VARCHAR(10),

  role              VARCHAR(20) NOT NULL,

  PRIMARY KEY (pilot_id_fk, flight_number_fk),

  CONSTRAINT fk_crew_pilot

    FOREIGN KEY (pilot_id_fk)      REFERENCES Pilot(pilot_id)

    ON UPDATE CASCADE ON DELETE RESTRICT,

  CONSTRAINT fk_crew_flight

    FOREIGN KEY (flight_number_fk) REFERENCES
Flight(flight_number)
```

```
    ON UPDATE CASCADE ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


-- Booking_Log Table (for Trigger 2)
CREATE TABLE Booking_Log (
  log_id        INT PRIMARY KEY AUTO_INCREMENT,
  flight_number VARCHAR(10),
  seat_number   VARCHAR(4),
  deleted_at    DATETIME,
  user_action   VARCHAR(50)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;



--
================================================================
-- SAMPLE DATA
--
================================================================


-- Airports (10)
INSERT INTO Airport VALUES
('IST','Istanbul Airport','Istanbul','Turkey'),
('SAW','Sabiha Gokcen','Istanbul','Turkey'),
('LHR','London Heathrow','London','UK'),
```

```
('JFK','John F. Kennedy','New York','USA'),

('FRA','Frankfurt Airport','Frankfurt','Germany'),

('CDG','Charles de Gaulle','Paris','France'),

('AMS','Schiphol','Amsterdam','Netherlands'),

('DXB','Dubai International','Dubai','UAE'),

('SFO','San Francisco Intl','San Francisco','USA'),

('DOH','Hamad Intl','Doha','Qatar');


-- Airlines (10)

INSERT INTO Airline VALUES

('THY','Turkish Airlines','IST'),

('BAW','British Airways','LHR'),

('LUF','Lufthansa','FRA'),

('AFR','Air France','CDG'),

('KLM','KLM Royal Dutch','AMS'),

('UAE','Emirates','DXB'),

('AAL','American Airlines','JFK'),

('UAL','United Airlines','SFO'),

('QTR','Qatar Airways','DOH'),

('PGT','Pegasus Airlines','SAW');


-- Passengers (10)

INSERT INTO Passenger
(first_name,last_name,email,passport_number) VALUES
```

```
('John','Doe','john.doe@example.com','A123456'),
('Jane','Smith','jane.smith@example.com','B789012'),
('Ahmet','Yilmaz','ahmet.yilmaz@example.com','T345678'),
('Elif','Demir','elif.demir@example.com','T987654'),
('David','Brown','david.brown@example.com','C112233'),
('Emily','Clark','emily.clark@example.com','D445566'),
('Hans','Muller','hans.muller@example.com','E778899'),
('Pierre','Dubois','pierre.dubois@example.com','F101112'),
('Mert','Kaplan','mert.kaplan@example.com','T556677'),
('Zeynep','Acar','zeynep.acar@example.com','T889900');

-- Aircraft (10)
INSERT INTO Aircraft VALUES
('TC-JNA','Boeing 777',349,'THY'),
('TC-LPA','Airbus A330',288,'THY'),
('G-XLEA','Airbus A380',469,'BAW'),
('D-ABYA','Boeing 747',410,'LUF'),
('F-GZNP','Boeing 777',381,'AFR'),
('PH-BVA','Boeing 777',408,'KLM'),
('A6-EOA','Airbus A380',489,'UAE'),
('N-123AA','Boeing 787',248,'AAL'),
('N-987UA','Boeing 777',350,'UAL'),
('A7-BEB','Boeing 777',354,'QTR');
```

```sql
-- Pilots (10)
INSERT INTO Pilot
(first_name,last_name,license_number,airline_code_fk) VALUES
('Mehmet','Oz','TR-PL-001','THY'),
('Ayse','Kaya','TR-PL-002','THY'),
('David','Brown','UK-PL-001','BAW'),
('Hans','Mueller','DE-PL-001','LUF'),
('Pierre','Dubois','FR-PL-001','AFR'),
('Jan','de Vries','NL-PL-001','KLM'),
('Ahmed','Al Maktoum','AE-PL-001','UAE'),
('John','Smith','US-PL-001','AAL'),
('Sarah','Lee','US-PL-002','UAL'),
('Khalid','Hassan','QA-PL-001','QTR');

-- Flights (10)
INSERT INTO Flight VALUES
('TK001','2025-10-22 08:30:00','2025-10-22
11:00:00','THY','TC-JNA','IST','LHR'),
('TK002','2025-10-22 13:00:00','2025-10-22
16:30:00','THY','TC-LPA','LHR','IST'),
('BA247','2025-10-22 09:00:00','2025-10-22
12:30:00','BAW','G-XLEA','LHR','JFK'),
('LH130','2025-10-22 10:00:00','2025-10-22
11:20:00','LUF','D-ABYA','FRA','CDG'),
```

```
('AF1390','2025-10-22 12:00:00','2025-10-22
13:10:00','AFR','F-GZNP','CDG','AMS'),

('KL1613','2025-10-22 14:00:00','2025-10-22
17:00:00','KLM','PH-BVA','AMS','IST'),

('EK202','2025-10-23 08:00:00','2025-10-23
12:00:00','UAE','A6-EOA','DXB','LHR'),

('AA100','2025-10-23 09:30:00','2025-10-23
12:45:00','AAL','N-123AA','JFK','LHR'),

('UA110','2025-10-23 07:00:00','2025-10-23
15:15:00','UAL','N-987UA','SFO','JFK'),

('QR240','2025-10-23 06:00:00','2025-10-23
09:00:00','QTR','A7-BEB','DOH','IST');


-- Bookings (10)

INSERT INTO Booking VALUES

(1,'BA247','22A','2025-09-01'),

(2,'BA247','22B','2025-09-02'),

(3,'TK001','10F','2025-09-03'),

(4,'TK002','11C','2025-09-05'),

(5,'LH130','05A','2025-09-05'),

(6,'AF1390','07D','2025-09-07'),

(7,'KL1613','14C','2025-09-08'),

(8,'EK202','18A','2025-09-09'),

(9,'AA100','19F','2025-09-10'),

(10,'UA110','21B','2025-09-11');
```

```sql
-- Flight_Crew (10)
INSERT INTO Flight_Crew VALUES
(1,'TK001','Captain'),
(1,'TK002','Captain'),
(3,'BA247','Captain'),
(4,'LH130','Captain'),
(5,'AF1390','Captain'),
(6,'KL1613','Captain'),
(7,'EK202','Captain'),
(8,'AA100','Captain'),
(9,'UA110','Captain'),
(10,'QR240','Captain');



--
================================================================
-- TRIGGERS
--
================================================================


-- Drop triggers if exist
DROP TRIGGER IF EXISTS check_capacity_before_insert;
DROP TRIGGER IF EXISTS log_booking_delete;
```

```sql
-- TRIGGER 1: Check Capacity Before Insert
-- This trigger checks if the flight is full before adding a new booking
DELIMITER //
CREATE TRIGGER check_capacity_before_insert
BEFORE INSERT ON Booking
FOR EACH ROW
BEGIN
    DECLARE current_passengers INT;
    DECLARE max_capacity INT;

    -- Get aircraft capacity for this flight
    SELECT A.capacity INTO max_capacity
    FROM Flight F
    JOIN Aircraft A ON F.aircraft_id_fk = A.aircraft_id
    WHERE F.flight_number = NEW.flight_number_fk;

    -- Count current passengers on this flight
    SELECT COUNT(*) INTO current_passengers
    FROM Booking
    WHERE flight_number_fk = NEW.flight_number_fk;

    -- If flight is full, prevent the booking
    IF current_passengers >= max_capacity THEN
```

```
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'ERROR: Flight is full! Cannot accept
new booking.';
    END IF;
END //
DELIMITER ;



-- TRIGGER 2: Log Booking Delete
-- This trigger logs every deleted booking to the Booking_Log
table
DELIMITER //
CREATE TRIGGER log_booking_delete
AFTER DELETE ON Booking
FOR EACH ROW
BEGIN
    -- Save deleted booking info to log table
    INSERT INTO Booking_Log (flight_number, seat_number,
deleted_at, user_action)
    VALUES (OLD.flight_number_fk, OLD.seat_number, NOW(),
'Booking Cancelled');
END //
DELIMITER ;
```

```
--
=============================================================
-- STORED PROCEDURES
--
=============================================================


-- Drop procedures if exist
DROP PROCEDURE IF EXISTS GetPassengerManifest;
DROP PROCEDURE IF EXISTS ScheduleFlight;


-- PROCEDURE 1: GetPassengerManifest
-- Returns the passenger list for a specific flight
DELIMITER //
CREATE PROCEDURE GetPassengerManifest(IN flightNo VARCHAR(10))
BEGIN
    -- Get all passengers with their seats for this flight
    SELECT P.first_name, P.last_name, B.seat_number
    FROM Booking B
    JOIN Passenger P ON B.passenger_id_fk = P.passenger_id
    WHERE B.flight_number_fk = flightNo;
END //
DELIMITER ;
```

```sql
-- PROCEDURE 2: ScheduleFlight

-- Creates a new flight with given parameters (uses THY airline
and IST->LHR route as default)

DELIMITER //

CREATE PROCEDURE ScheduleFlight(

    IN f_num VARCHAR(10),

    IN f_dep DATETIME,

    IN f_arr DATETIME,

    IN f_plane VARCHAR(10)

)

BEGIN

    -- Insert new flight with default route

    INSERT INTO Flight (flight_number, departure_time,
arrival_time, airline_code_fk, aircraft_id_fk, origin_airport_fk,
dest_airport_fk)

    VALUES (f_num, f_dep, f_arr, 'THY', f_plane, 'IST', 'LHR');



    SELECT 'Flight created successfully' as status;

END //

DELIMITER ;

--
========================================================================

-- END OF SQL DUMP

--
========================================================================
```

Appendix B: PHP Integration Notes

To enable the database connectivity required for this project, the following integration steps were performed in the XAMPP environment:

1. **MySQL Integration:** We used the built-in `mysqli` extension in PHP. No additional configuration was required as XAMPP enables this by default.
2. **MongoDB Integration:**
    - We determined the correct PHP version and architecture (Thread Safe) using the `phpinfo()` command.
    - We downloaded the compatible `php_mongodb.dll` file and placed it in the `C:\xampp\php\ext` directory.
    - We edited the `php.ini` file to add the line `extension=mongodb`.
    - We installed the MongoDB library using Composer in the project root directory with the command: `composer require mongodb/mongodb`.
    - After restarting the Apache server via the XAMPP Control Panel, the MongoDB driver was successfully loaded.