CS 306 HOUSING DATABASE SYSTEM GROUP 84 REPORT

Student Name: İdil Sunar
Student Number: 34363

1. INTRODUCTION

This report documents Phase 3 of the Housing Database Management System project. The system demonstrates advanced database features including MySQL triggers, stored procedures, and MongoDB integration for a support ticket system.

Project Components:

- MySQL database with triggers and stored procedures (Phase 2 continuation)
- MongoDB-based support ticket system
- PHP web interface for user and admin interactions

2. IMPLEMENTED TRIGGERS

2.1 Trigger: trg_request_auto_assign

Purpose:
Automatically assigns a default staff member (ID: 501 - Ahmet Usta, Electrician) to maintenance requests when no technician is specified during request creation.

SQL Script:

```
DELIMITER //
CREATE TRIGGER trg_request_auto_assign
BEFORE INSERT ON MaintenanceRequest
FOR EACH ROW
BEGIN
   IF NEW.assignee_staff_id IS NULL THEN
      SET NEW.assignee_staff_id = 501;
   END IF;
END//
DELIMITER ;
```

How it works:
This trigger fires BEFORE each INSERT operation on the MaintenanceRequest table. It checks if the assignee_staff_id field is NULL, and if so, automatically sets it to 501. This ensures all maintenance requests have an assigned technician, preventing orphaned requests.

Web Interface Implementation:

The trigger is tested through `trigger_auto_assign.php` which provides two test cases:

Case 1: Insert WITH Assignee

- **Button: "Run Case 1"**
- **Action: Inserts a maintenance request with assignee_staff_id = 502**
- **Expected Behavior: Trigger does NOT activate; assignee remains 502**

SQL Executed:

```
INSERT INTO MaintenanceRequest
  (request_id, dorm_id, roomno, reporter_stud_id, assignee_staff_id)
  VALUES (5001, 1, 101, 1001, 502)
```

Case 2: Insert WITHOUT Assignee

- **Button: "Run Case 2"**
- **Action: Inserts a maintenance request with assignee_staff_id = NULL**
- **Expected Behavior: Trigger activates; assignee automatically set to 501**

SQL Executed:

```
INSERT INTO MaintenanceRequest
  (request_id, dorm_id, roomno, reporter_stud_id, assignee_staff_id)
  VALUES (5002, 1, 101, 1001, NULL)
```

3. IMPLEMENTED STORED PROCEDURES

3.1 Stored Procedure: sp_room_occupancy_report

Purpose:
Generates a comprehensive occupancy report for a specified dormitory, displaying room-by-room capacity, current student count, and available beds.

SQL Script:

```
DELIMITER //
CREATE PROCEDURE sp_room_occupancy_report(IN p_dorm_id INT)
BEGIN
   SELECT
      r.dorm_id,
      r.roomno,
      r.capacity,
      COUNT(c.c_id) AS current_occupancy,
```

```
      (r.capacity - COUNT(c.c_id)) AS free_beds
   FROM Room r
   LEFT JOIN Contract c
     ON r.dorm_id = c.dorm_id
    AND r.roomno = c.roomno
   WHERE r.dorm_id = p_dorm_id
   GROUP BY r.dorm_id, r.roomno, r.capacity
   ORDER BY r.roomno;
END//
DELIMITER ;
```

This stored procedure accepts one input parameter (dorm ID) and performs a LEFT JOIN between Room and Contract tables to calculate occupancy. The GROUP BY clause aggregates contracts per room, while the COUNT function determines how many students are currently assigned. The calculation (capacity - current_occupancy) provides the number of free beds.

Input Parameters:

Parameter	Type	Description	Web Input
p_dorm_id	INT	Dorm identifier	Dropdown menu populated from Dorm table

Web Interface Implementation:

The procedure is accessed through `procedure_occupancy.php` with the following components:

Input Box:

- Dropdown menu labeled "Select Dorm"
- Dynamically populated with all dorms from the database
- Format: "Dorm ID - Dorm Name" (e.g., "1 - Ada Dorm")

Output Display:

- HTML table with columns: Dorm ID, Room No, Capacity, Current Occupancy, Free Beds
- Results sorted by room number
- Displayed immediately upon form submission

4. MONGODB SUPPORT TICKET SYSTEM

4.1 System Overview

The support ticket system uses MongoDB for flexible document storage, allowing dynamic comments and metadata. The system provides separate interfaces for users and administrators.

Database Structure:

- **Database Name:** support_tickets_db
- **Collection Name:** tickets

Document Schema:

```
{
  _id: ObjectId,                  // Auto-generated MongoDB ID
  username: String,               // Ticket creator's username
  message: String,                // Issue description
  created_at: String,             // Timestamp (YYYY-MM-DD HH:MM:SS)
  status: Boolean,                // true = active, false = resolved
  comments: Array<String>         // Array of comment strings
}
```

4.2 MongoDB Queries and PHP Implementation

Query 1: Create New Ticket

File: create_ticket.php

Purpose: Insert a new support ticket into the database

PHP Script:

```
$client = new MongoDB\Client("mongodb://localhost:27017");
$collection = $client->support_tickets_db->tickets;
```

```
$ticket = [
    'username' => $_POST['username'],
    'message' => $_POST['message'],
    'created_at' => date('Y-m-d H:i:s'),
    'status' => true,
    'comments' => []
];
$result = $collection->insertOne($ticket);
```

Query 2: List Active Tickets with Optional Username Filter

File: tickets.php (User Interface)

Purpose: Retrieve all active tickets, optionally filtered by username

PHP Script:

```
$username_filter = isset($_GET['username']) ? $_GET['username'] : '';

$query = ['status' => true];
if ($username_filter) {
    $query['username'] = $username_filter;
}

$tickets = $collection->find($query, ['sort' => ['created_at' => -1]]);
```

Query 3: Get Distinct Usernames for Filter Dropdown

File: tickets.php

Purpose: Populate username filter dropdown with unique usernames

PHP Script:

```
$usernames = $collection->distinct('username');
```

Query 4: Retrieve Single Ticket by ID

File: ticket_details.php

Purpose: Get complete ticket details for viewing and commenting

PHP Script:

```
$ticket_id = $_GET['id'];


$ticket = $collection->findOne([
    '_id' => new MongoDB\BSON\ObjectId($ticket_id)
]);
```

Query 5: Add Comment to Ticket

File: ticket_details.php

Purpose: Append a user comment to an existing ticket

PHP Script:

```
$comment = $_POST['username'] . ': ' . $_POST['comment'];


$collection->updateOne(
    ['_id' => new MongoDB\BSON\ObjectId($ticket_id)],
    ['$push' => ['comments' => $comment]]
);
```

Query 6: List All Active Tickets (Admin View)

File: admin/index.php

Purpose: Display all active tickets for admin management

PHP Script:

```
$tickets = $collection->find(
    ['status' => true],
    ['sort' => ['created_at' => -1]]
);
```

Query 7: Add Admin Comment

File: admin/ticket_details.php

Purpose: Add an administrative response to a ticket

PHP Script:

```
$comment = 'admin: ' . $_POST['comment'];

$collection->updateOne(
    ['_id' => new MongoDB\BSON\ObjectId($ticket_id)],
    ['$push' => ['comments' => $comment]]
);
```

Query 8: Resolve Ticket

File: admin/ticket_details.php

Purpose: Mark a ticket as resolved

PHP Script:

```
$collection->updateOne(
```

```
    ['_id' => new MongoDB\BSON\ObjectId($ticket_id)],
    ['$set' => ['status' => false]]
);
```

5. SUPPORT TICKET SYSTEM WORKFLOW

5.1 User Workflow

Step 1: Access System

- User navigates to homepage (index.php)
- Clicks "Support Ticket System" button

Step 2: View Existing Tickets

- User sees list of active tickets on tickets.php
- Can filter by username using dropdown menu
- Each ticket shows preview of message and comment count

Step 3: Create New Ticket

- User clicks "Create New Ticket" button
- Fills out form with username and issue description
- Submits form to create ticket in MongoDB

Step 4: View Ticket Details

- User clicks on a ticket from the list
- Views complete message and all comments
- Sees creation date and status

Step 5: Add Comments

- User enters name and comment in form
- Submits to add comment to ticket thread
- Comment appears immediately in list

Step 6: Track Resolution

- User can see admin responses in comments

- When admin resolves ticket, it disappears from active list

5.2 Admin Workflow

Step 1: Access Admin Panel

- Admin navigates to http://localhost/admin/
- Sees all active tickets from all users (no filtering)

Step 2: Review Tickets

- Admin reviews ticket messages and priorities
- Can see creation dates and comment counts

Step 3: View Ticket Details

- Admin clicks "View & Manage" on any ticket
- Sees full ticket information and comment history

Step 4: Respond to User

- Admin adds comment with solution or update
- Comment is prefixed with "admin:" identifier

Step 5: Resolve Ticket

- When issue is addressed, admin clicks "Mark as Resolved"
- Ticket status changes to false
- Ticket is removed from both admin and user active lists

6. ASSUMPTIONS

1. Default Staff Assignment: All unassigned maintenance requests should be handled by the same default technician (Ahmet Usta, ID: 501) who serves as a general maintenance coordinator
2. Username-Based Identification: The system uses username strings for identification rather than implementing a full authentication system, appropriate for a demonstration project

3. Comment Format: Comments are stored as simple strings with username prefix (e.g., "john: This is my comment") rather than as separate document objects, prioritizing simplicity
4. Ticket Resolution: Resolved tickets are soft-deleted (status set to false) rather than physically deleted, preserving historical data for potential reporting or auditing
5. Single Database Server: Both user and admin interfaces connect to the same MongoDB instance on localhost:27017
6. Active-Only Views: Both user and admin interfaces display only active tickets (status = true) in list views; resolved tickets are hidden but not deleted
7. No Ticket Priority: All tickets are treated with equal priority; no categorization or urgency system implemented
8. MongoDB Default Port: MongoDB runs on default port 27017 without authentication for local development

7. CHALLENGES AND DESIGN DECISIONS

7.1 Challenge: MySQL Port Conflict

Problem: Windows MySQL80 service was already running on port 3306, preventing XAMPP MySQL from starting.

Solution:

1. Identified the conflicting process using netstat -ano | findstr :3306
2. Terminated the MySQL80 service using Task Manager
3. Disabled MySQL80 from starting automatically: sc config MySQL80 start= disabled
4. Successfully started XAMPP MySQL on port 3306

Design Decision: Use XAMPP's bundled MariaDB instead of external MySQL installation for easier project portability.

7.2 Challenge: MariaDB Collation Compatibility

Problem: MySQL Workbench exported database using utf8mb4_0900_ai_ci collation, which is not supported by MariaDB 10.4.

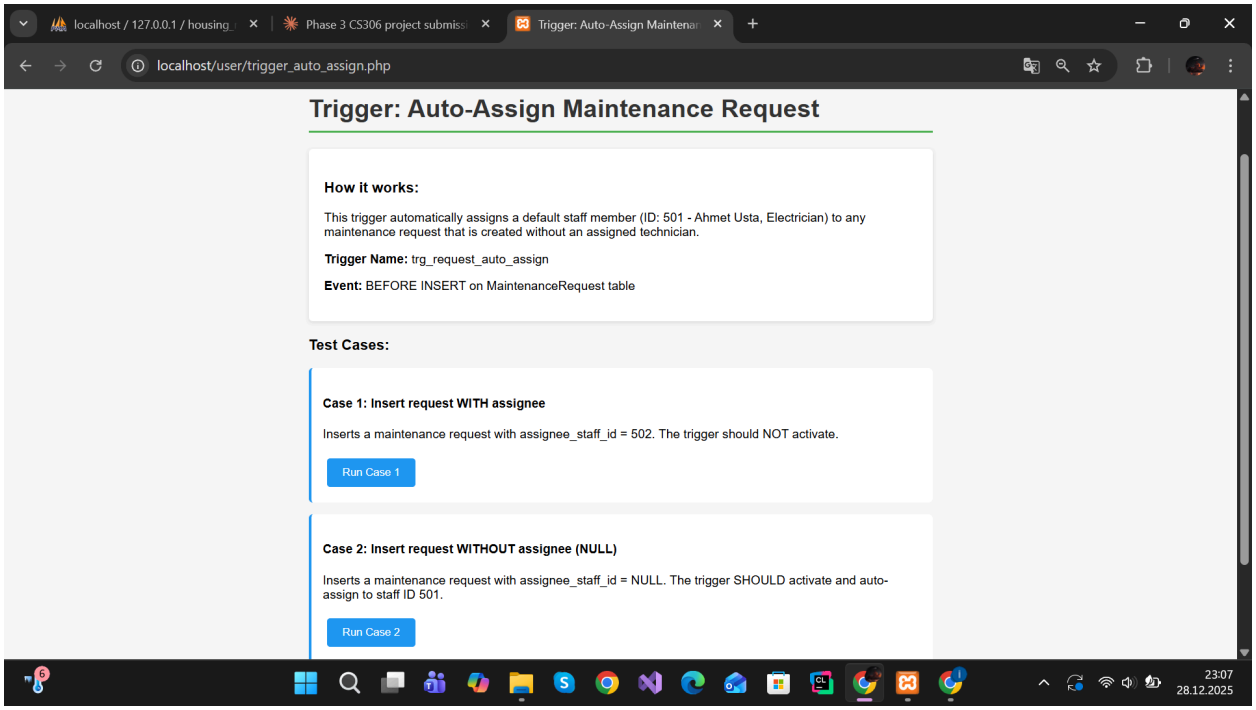
Error Message: Unknown collation: 'utf8mb4_0900_ai_ci'

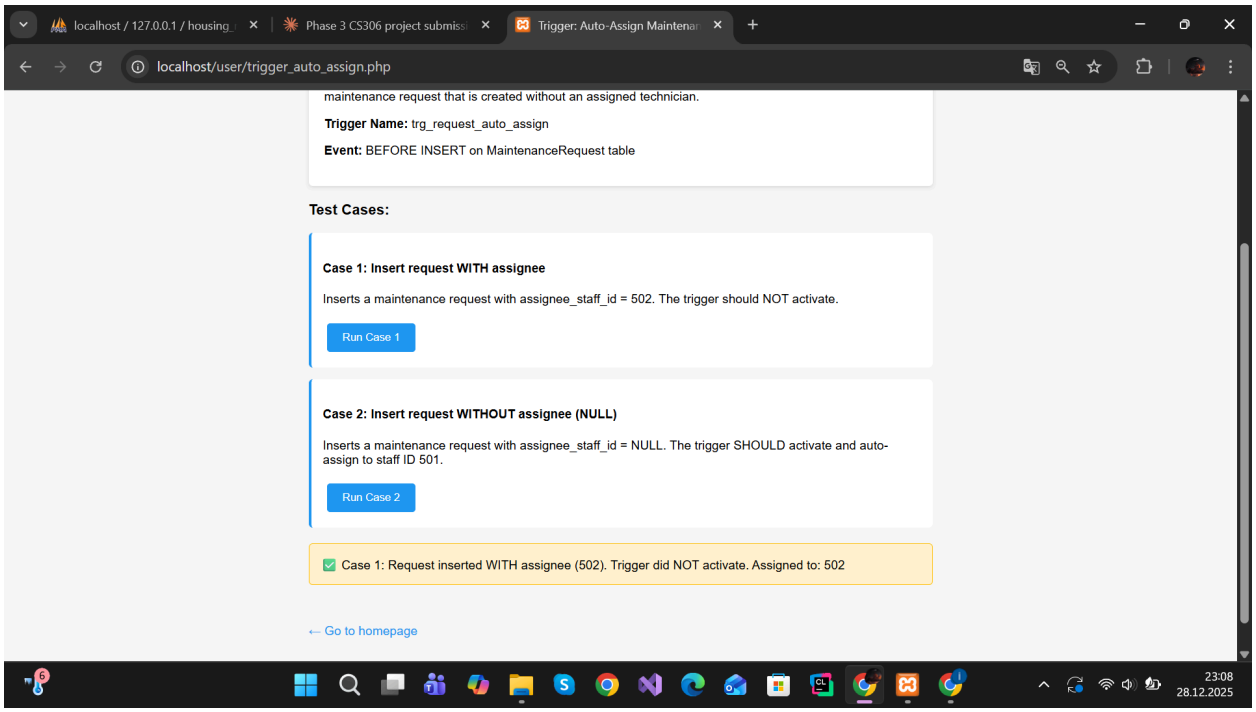
Solution: Modified the CREATE DATABASE statement to use MariaDB-compatible collation:

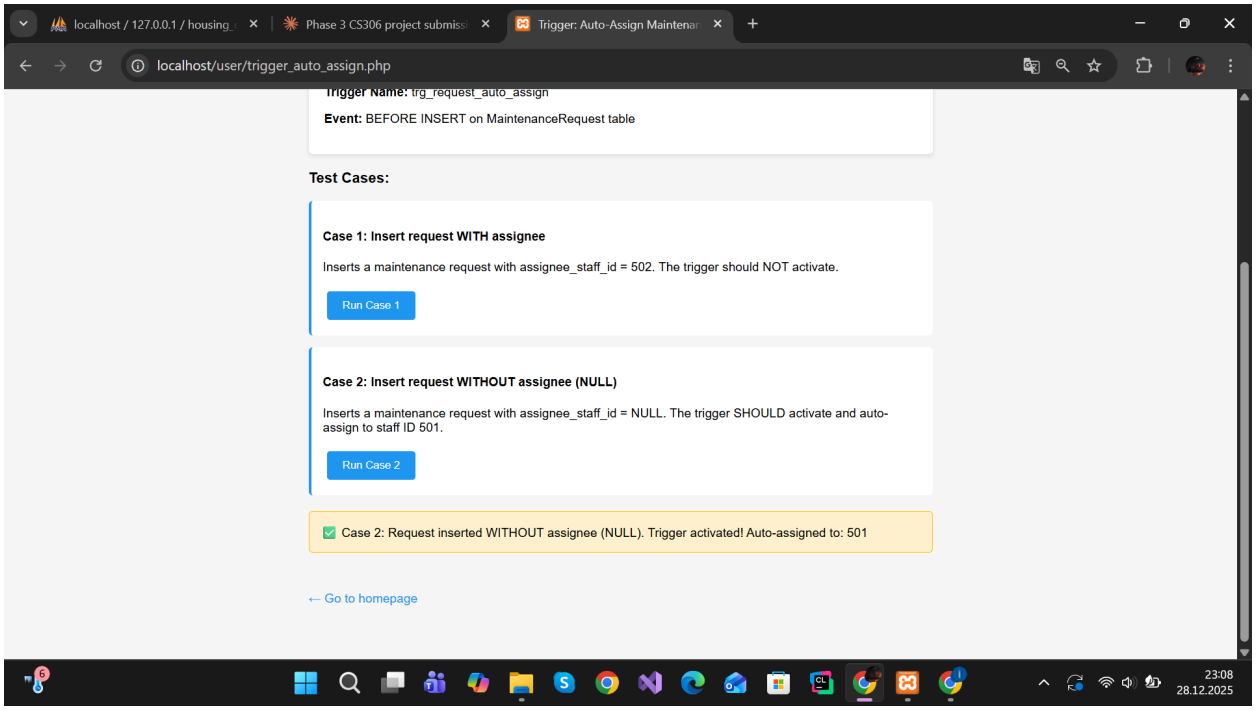
CREATE DATABASE housing_mgmt CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

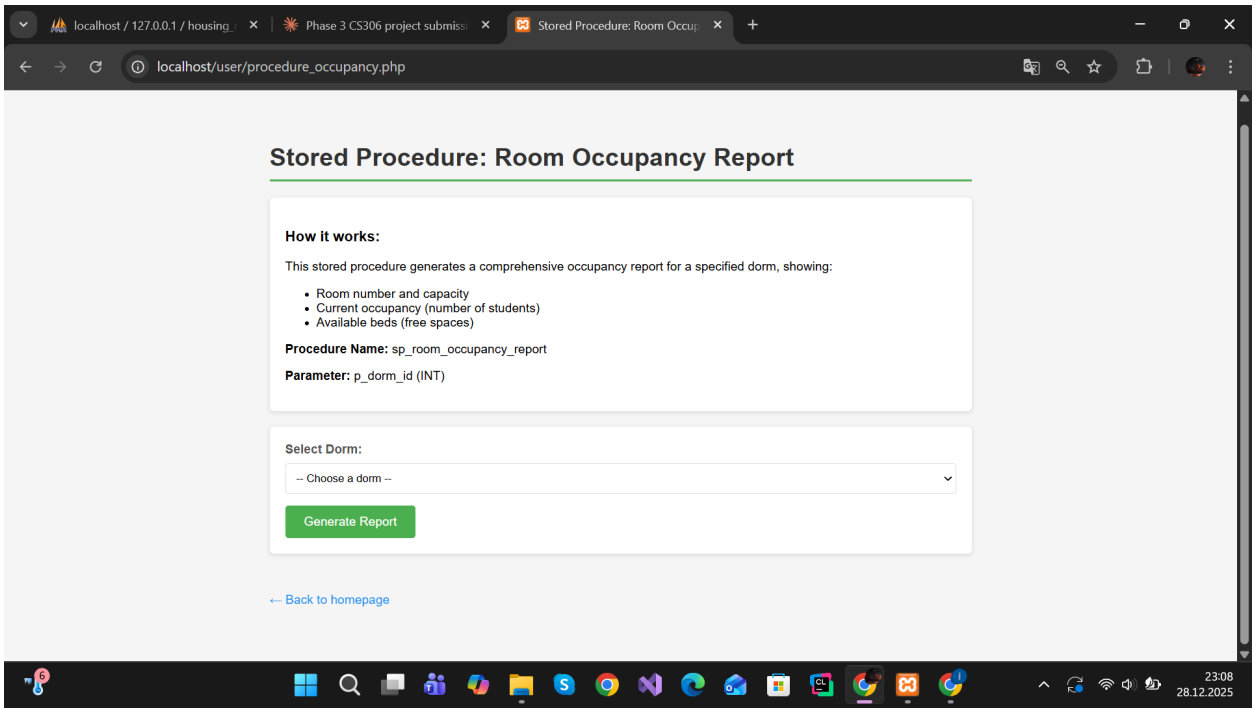
Design Decision: Standardize on `utf8mb4_unicode_ci` collation for maximum compatibility across MySQL and MariaDB versions.

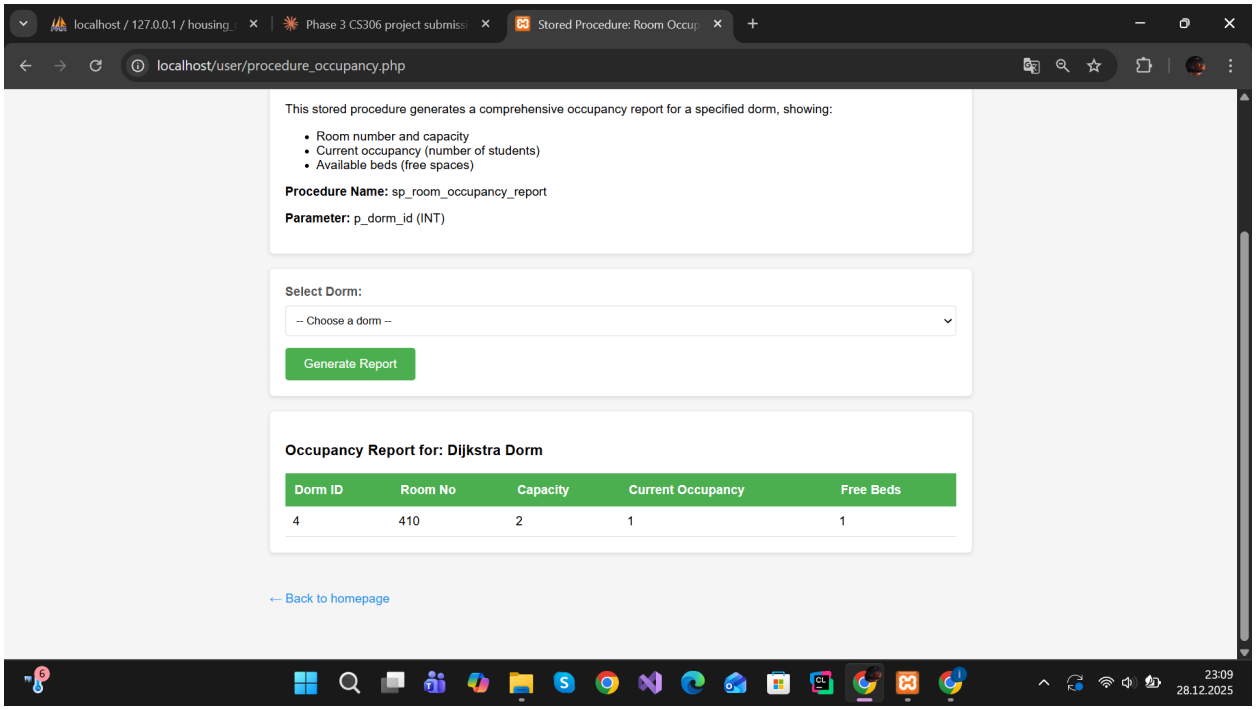
8. Screenshots

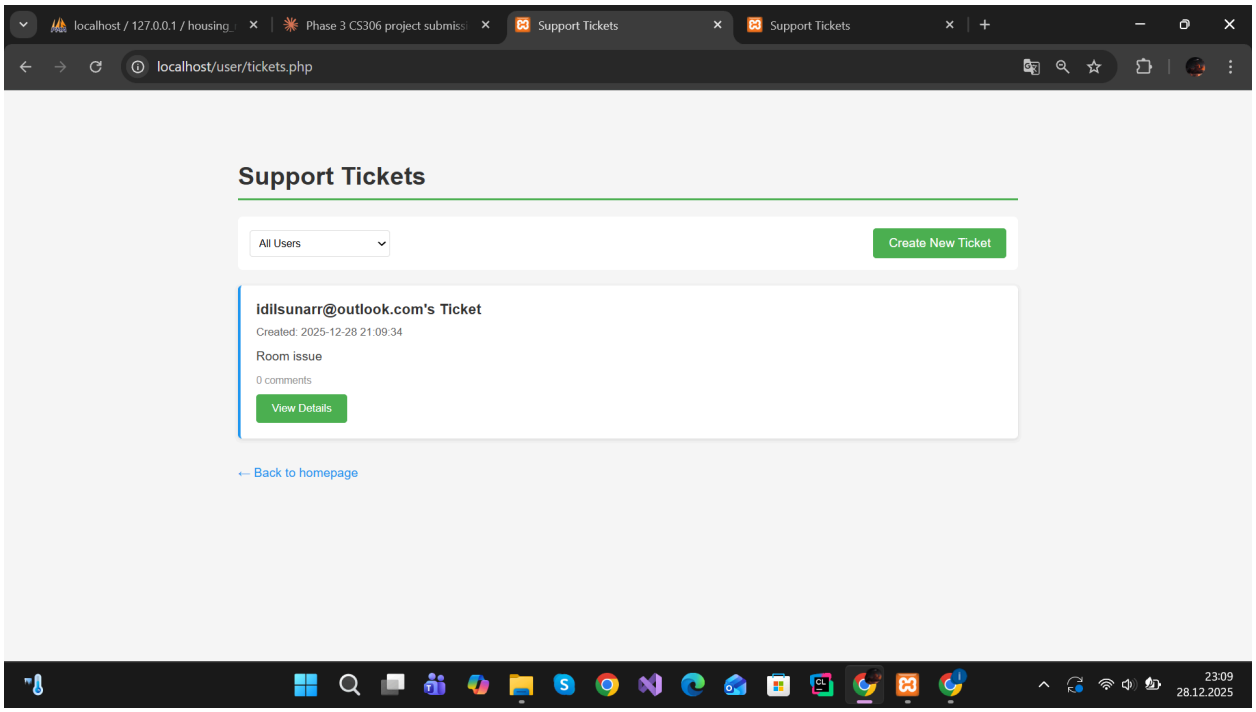

localhost / 127.0.0.1 / housing
Phase 3 CS306 project submiss
Trigger: Auto-Assign Maintenar
localhost/user/trigger_auto_assign.php
Trigger: Auto-Assign Maintenance Request
How it works:
This trigger automatically assigns a default staff member (ID: 501 - Ahmet Usta, Electrician) to any maintenance request that is created without an assigned technician.
Trigger Name: trg_request_auto_assign
Event: BEFORE INSERT on MaintenanceRequest table
Test Cases:
Case 1: Insert request WITH assignee
Inserts a maintenance request with assignee_staff_id = 502. The trigger should NOT activate.
Run Case 1
Case 2: Insert request WITHOUT assignee (NULL)
Inserts a maintenance request with assignee_staff_id = NULL. The trigger SHOULD activate and auto-assign to staff ID 501.
Run Case 2
23:07
28.12.2025

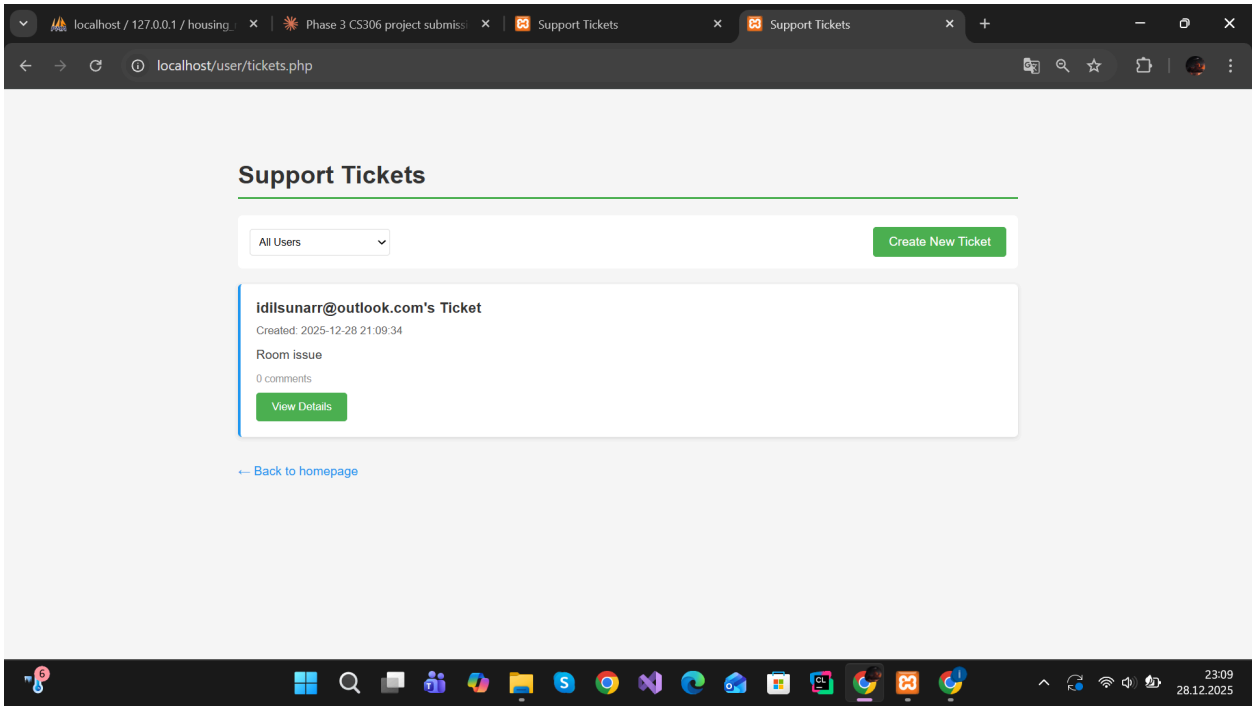

localhost / 127.0.0.1 / housing
Phase 3 CS306 project submiss
Trigger: Auto-Assign Maintenar
localhost/user/trigger_auto_assign.php
maintenance request that is created without an assigned technician.
Trigger Name: trg_request_auto_assign
Event: BEFORE INSERT on MaintenanceRequest table
Test Cases:
Case 1: Insert request WITH assignee
Inserts a maintenance request with assignee_staff_id = 502. The trigger should NOT activate.
Run Case 1
Case 2: Insert request WITHOUT assignee (NULL)
Inserts a maintenance request with assignee_staff_id = NULL. The trigger SHOULD activate and auto-assign to staff ID 501.
Run Case 2
Case 1: Request inserted WITH assignee (502). Trigger did NOT activate. Assigned to: 502
← Go to homepage
23:08
28.12.2025

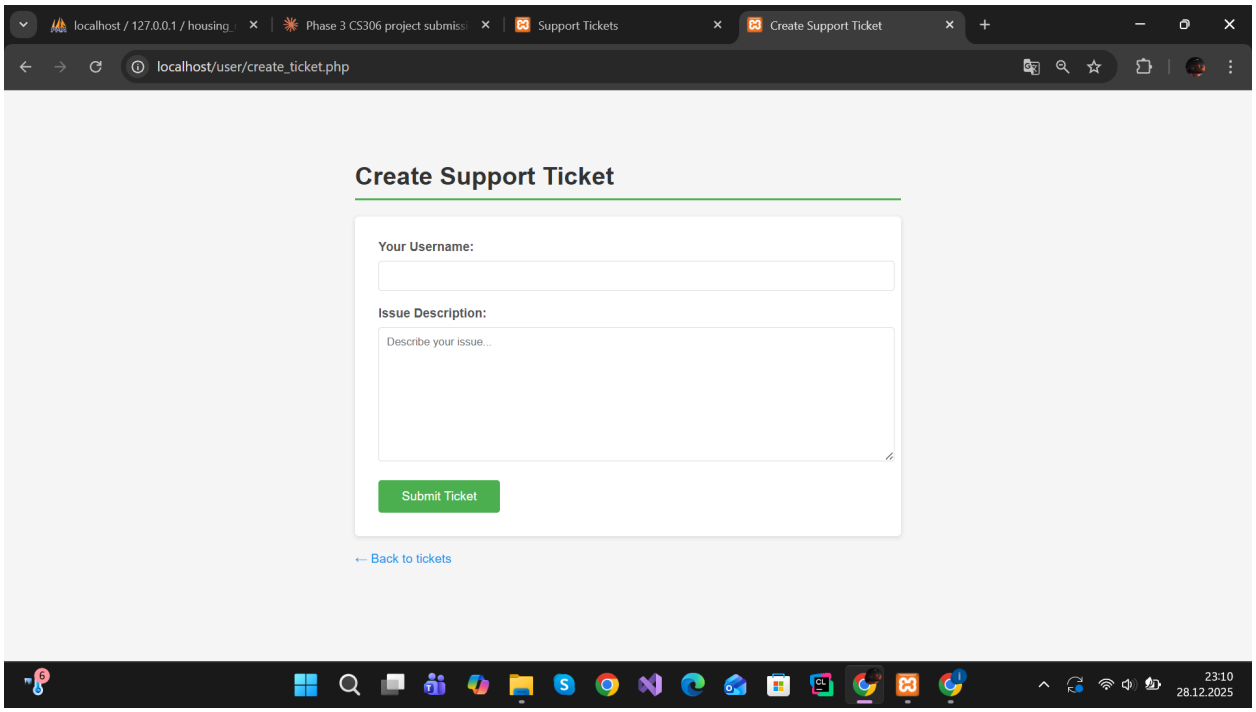

localhost / 127.0.0.1 / housing
Phase 3 CS306 project submiss
Trigger: Auto-Assign Maintenar
localhost/user/trigger_auto_assign.php
Trigger Name: trg_request_auto_assign
Event: BEFORE INSERT on MaintenanceRequest table
Test Cases:
Case 1: Insert request WITH assignee
Inserts a maintenance request with assignee_staff_id = 502. The trigger should NOT activate.
Run Case 1
Case 2: Insert request WITHOUT assignee (NULL)
Inserts a maintenance request with assignee_staff_id = NULL. The trigger SHOULD activate and auto-assign to staff ID 501.
Run Case 2
Case 2: Request inserted WITHOUT assignee (NULL). Trigger activated! Auto-assigned to: 501
← Go to homepage
23:08
28.12.2025

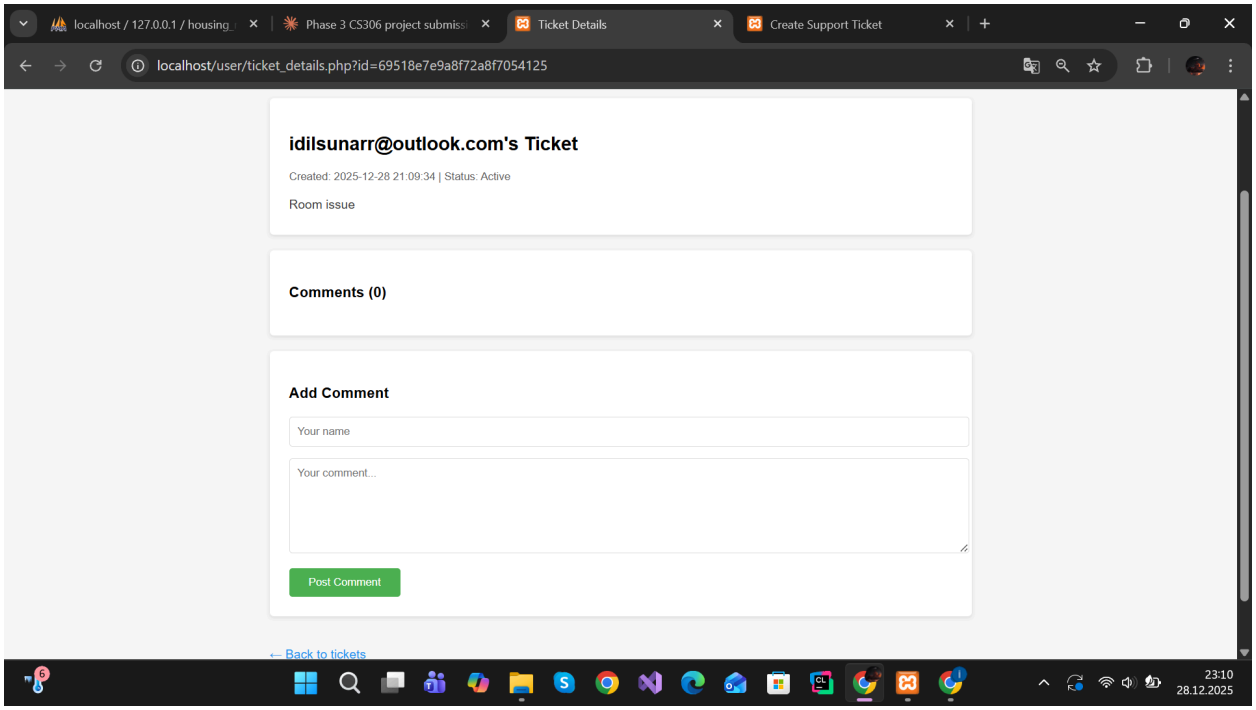

localhost / 127.0.0.1 / housing
Phase 3 CS306 project submiss
Stored Procedure: Room Occup
localhost/user/procedure_occupancy.php
Stored Procedure: Room Occupancy Report
How it works:
This stored procedure generates a comprehensive occupancy report for a specified dorm, showing:
Room number and capacity
Current occupancy (number of students)
Available beds (free spaces)
Procedure Name: sp_room_occupancy_report
Parameter: p_dorm_id (INT)
Select Dorm:
-- Choose a dorm --
Generate Report
← Back to homepage
23:08
28.12.2025

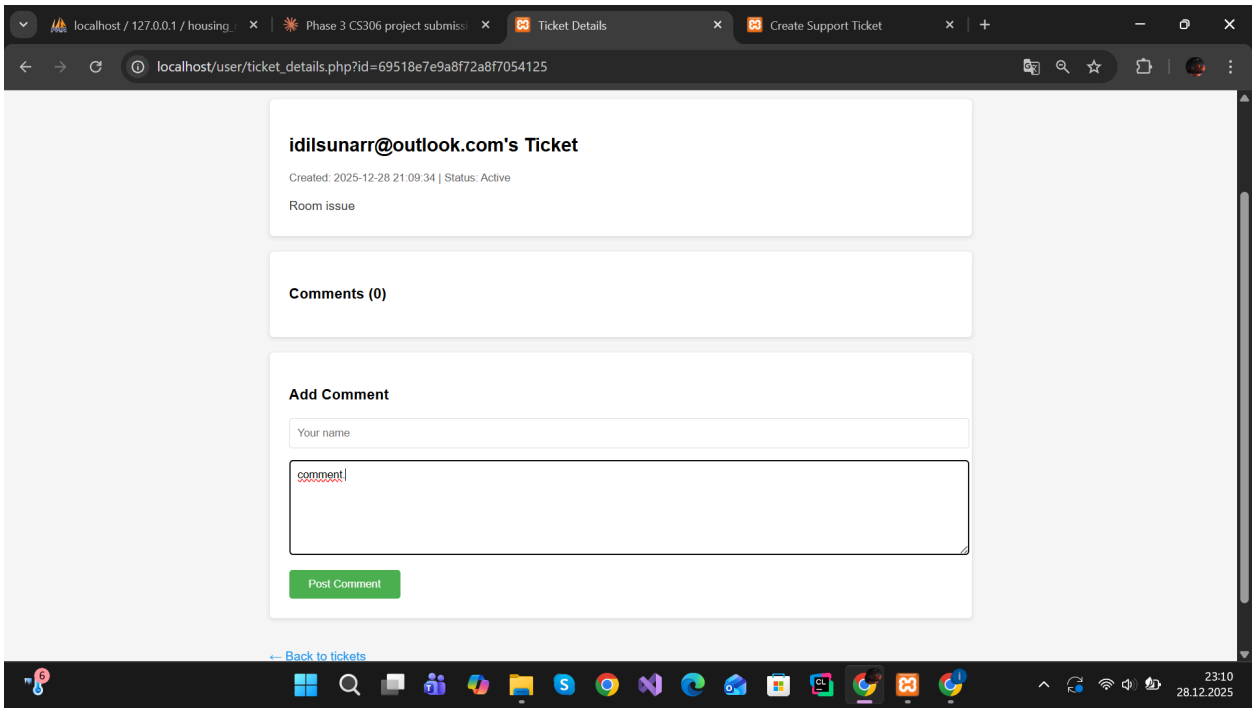

localhost / 127.0.0.1 / housing
Phase 3 CS306 project submissi
Stored Procedure: Room Occup
localhost/user/procedure_occupancy.php
This stored procedure generates a comprehensive occupancy report for a specified dorm, showing:
Room number and capacity
Current occupancy (number of students)
Available beds (free spaces)
Procedure Name: sp_room_occupancy_report
Parameter: p_dorm_id (INT)
Select Dorm:
-- Choose a dorm --
Generate Report
Occupancy Report for: Dijkstra Dorm
Dorm ID
Room No
Capacity
Current Occupancy
Free Beds
4
410
2
1
1
← Back to homepage
23:09
28.12.2025

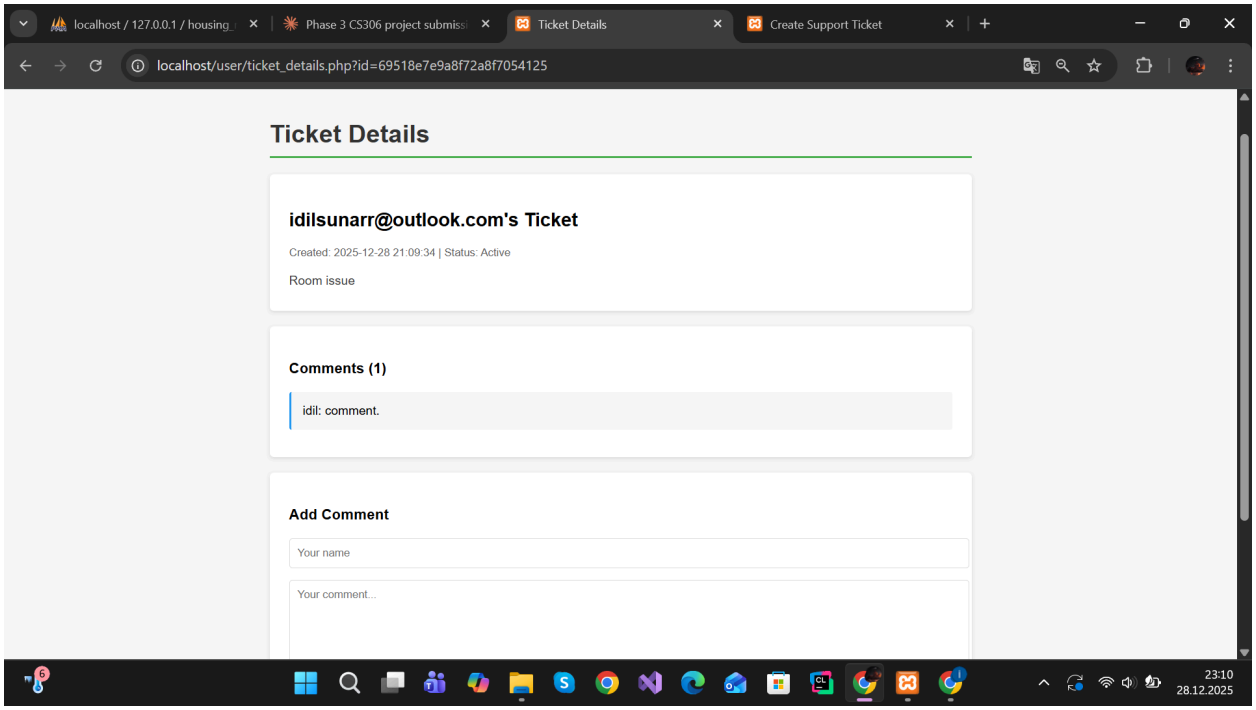

localhost / 127.0.0.1 / housing
Phase 3 CS306 project submissi
Support Tickets
Support Tickets
localhost/user/tickets.php
Support Tickets
All Users
Create New Ticket
idilsunarr@outlook.com's Ticket
Created: 2025-12-28 21:09:34
Room issue
0 comments
View Details
← Back to homepage
23:09
28.12.2025

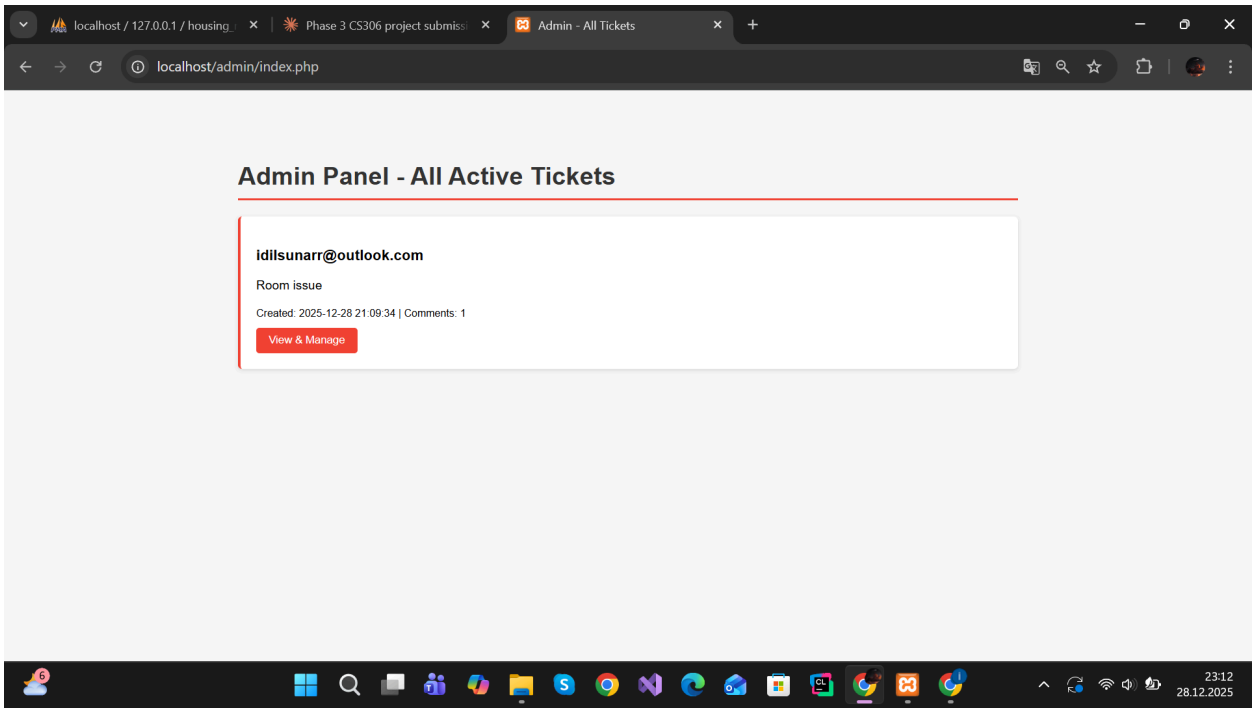

localhost / 127.0.0.1 / housing
Phase 3 CS306 project submiss
Support Tickets
Support Tickets
localhost/user/tickets.php
Support Tickets
All Users
Create New Ticket
idilsunarr@outlook.com's Ticket
Created: 2025-12-28 21:09:34
Room issue
0 comments
View Details
← Back to homepage
23:09
28.12.2025

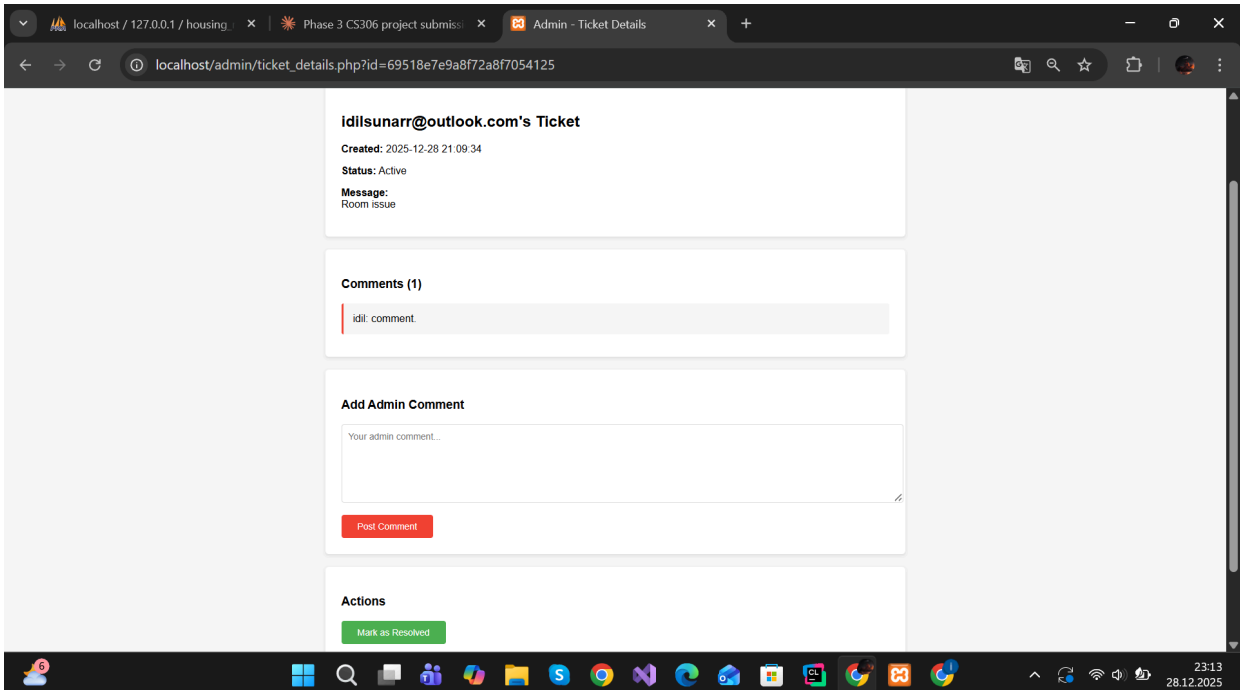

localhost / 127.0.0.1 / housing
Phase 3 CS306 project submiss
Support Tickets
Create Support Ticket
localhost/user/create_ticket.php
Create Support Ticket
Your Username:
Issue Description:
Describe your issue...
Submit Ticket
← Back to tickets
23:10
28.12.2025

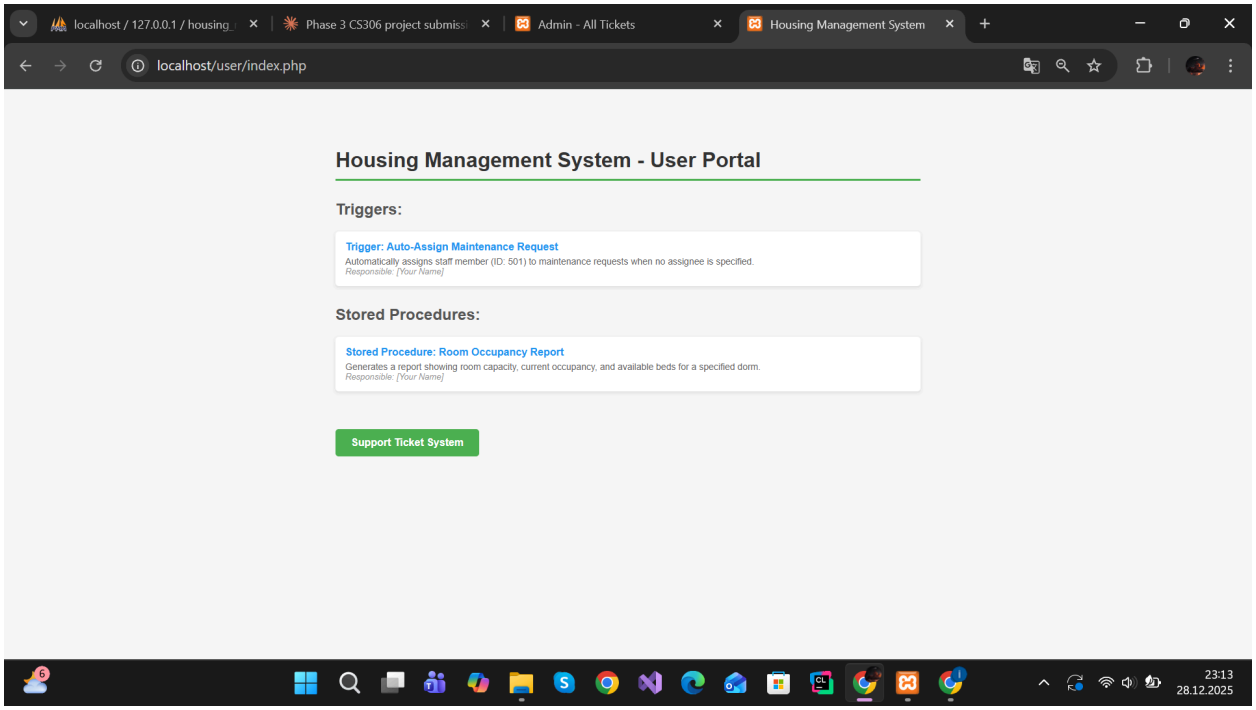

localhost / 127.0.0.1 / housing
Phase 3 CS306 project submiss
Ticket Details
Create Support Ticket
localhost/user/ticket_details.php?id=69518e7e9a8f72a8f7054125
idilsunarr@outlook.com's Ticket
Created: 2025-12-28 21:09:34 | Status: Active
Room issue
Comments (0)
Add Comment
Your name
Your comment...
Post Comment
← Back to tickets
23:10
28.12.2025

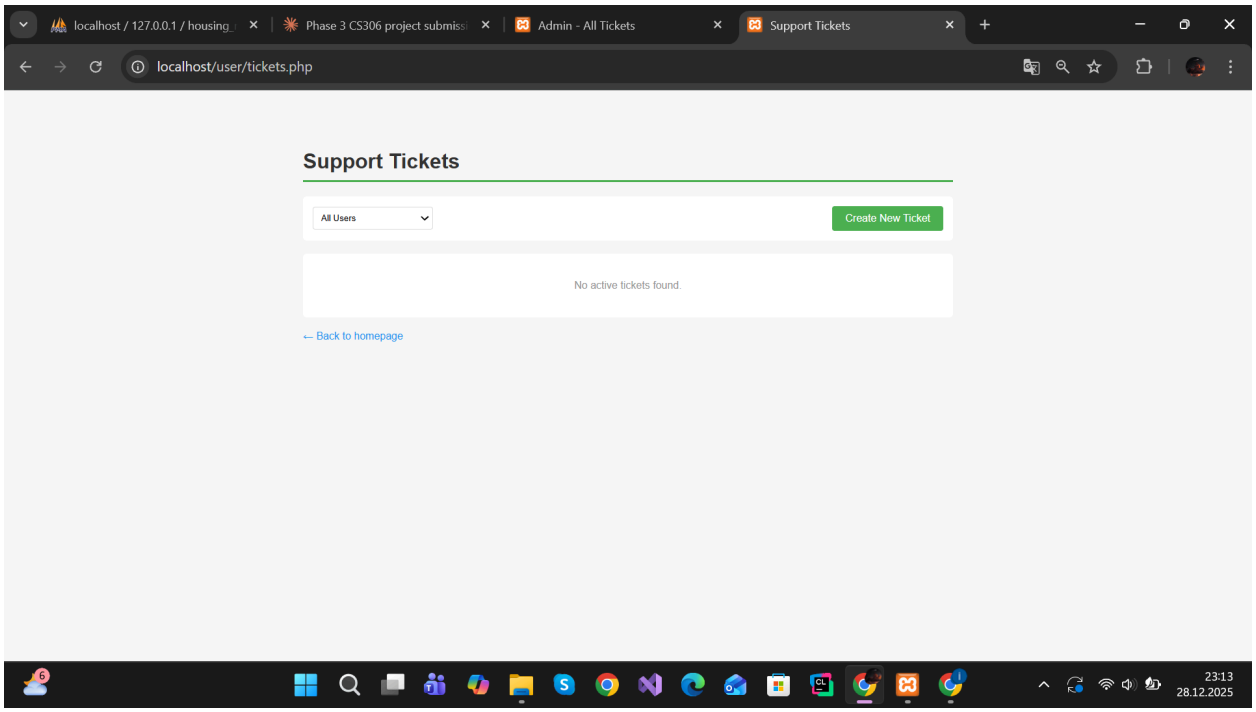

localhost / 127.0.0.1 / housing
Phase 3 CS306 project submiss
Ticket Details
Create Support Ticket
localhost/user/ticket_details.php?id=69518e7e9a8f72a8f7054125
idilsunarr@outlook.com's Ticket
Created: 2025-12-28 21:09:34 | Status: Active
Room issue
Comments (0)
Add Comment
Your name
comment
Post Comment
← Back to tickets
23:10
28.12.2025


localhost / 127.0.0.1 / housing
Phase 3 CS306 project submiss
Ticket Details
Create Support Ticket
localhost/user/ticket_details.php?id=69518e7e9a8f72a8f7054125
Ticket Details
idilsunarr@outlook.com's Ticket
Created: 2025-12-28 21:09:34 | Status: Active
Room issue
Comments (1)
idil: comment.
Add Comment
Your name
Your comment...
23:10
28.12.2025


localhost / 127.0.0.1 / housing
Phase 3 CS306 project submiss
Admin - All Tickets
localhost/admin/index.php
Admin Panel - All Active Tickets
idilsunarr@outlook.com
Room issue
Created: 2025-12-28 21:09:34 | Comments: 1
View & Manage
23:12
28.12.2025


localhost / 127.0.0.1 / housing
Phase 3 CS306 project submissi
Admin - Ticket Details
localhost/admin/ticket_details.php?id=69518e7e9a8f72a8f7054125
idilsunarr@outlook.com's Ticket
Created: 2025-12-28 21:09:34
Status: Active
Message:
Room issue
Comments (1)
idil: comment.
Add Admin Comment
Your admin comment...
Post Comment
Actions
Mark as Resolved
23:13
28.12.2025





9. PHP Integration Notes

Composer Dependency Installation:

cd C:\xampp\htdocs\user
composer require mongodb/mongodb

cd C:\xampp\htdocs\admin
composer require mongodb/mongodb

Key Integration Points:

1. Autoloading: All MongoDB scripts include require_once 'vendor/autoload.php' to load Composer dependencies
2. Error Handling: Try-catch blocks wrap all MongoDB operations to gracefully handle connection failures
3. ObjectId Conversion: URL-passed ticket IDs are converted to MongoDB ObjectIds using: