CS306 Project Phase III

Web Integrations of SQL and NoSQL Databases

Contents

1. Introduction

This project focuses on extending our previously developed database system into a fully functional web application, supporting both user and administrator interfaces. Using PHP and XAMPP as our development stack, the goal is to integrate core database logic—specifically, triggers and stored procedures—with a dynamic and interactive website. In addition, we introduce a support ticket system that allows users to request assistance and administrators to manage and respond to those requests.

The application is divided into two primary components:

- **User Side:** A homepage where users can access features that interact with database triggers and stored procedures. It also includes a support section where users can open tickets for help.
- **Admin Side:** A dedicated interface for administrators to view and manage active support tickets submitted by users. Admins can comment on tickets and mark them as resolved.

The web application is hosted locally using XAMPP, and PHP is used to handle both MySQL and MongoDB database interactions. The triggers and stored procedures used in this project were designed and implemented in earlier phases, and this phase focuses on making them accessible and testable via the web interface. Meanwhile, the support ticket system is backed by MongoDB.

2. System Design and Features

This section describes the key components of the web application, detailing how users and administrators interact with the system. Each part of the interface is designed to connect with the backend logic and database operations.

2.1. User Interface

The user side of the system is accessible via the localhost/user URL. This entry point leads to a homepage where users can navigate to triggers, stored procedures, or the support section. Each of these features is accessible through clearly labeled hyperlinks.

Trigger Integration

Each group member is responsible for integrating one database trigger with the web interface. The homepage displays a list of all triggers, with a brief description and the name of the person responsible for that integration. Clicking on a trigger link opens a dedicated page

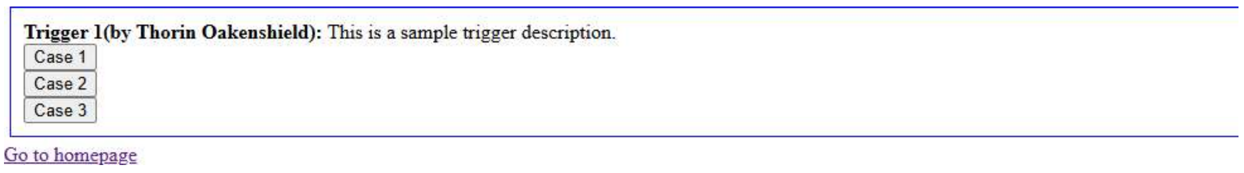

Trigger 1(by Thorin Oakenshield): This is a sample trigger description.

Figure 1: An example Trigger Page

describing how it works and showing interactive buttons that simulate different conditions under which the trigger would execute. **After a button is clicked, the output message should be displayed within the page.** Figure 1 shows an example page dedicated to a trigger.

As an example, a trigger that behaves differently depending on two input conditions will have two buttons on its page. Each button initiates a case that tests a specific behavior of the trigger through database operations. At the bottom of the page, there is a link to return to the homepage.

Stored Procedure Integration

Similarly, each member integrates a stored procedure from the previous phase. The homepage displays these procedures along with descriptions and responsible group members. Clicking on a procedure link takes the user to a form where they can provide input values for the procedure's parameters. Figure 2 shows an example page dedicated to a stored procedure.

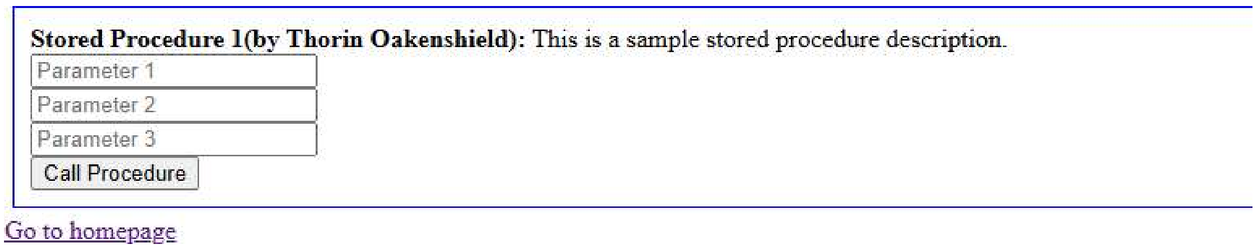

Stored Procedure 1(by Thorin Oakenshield): This is a sample stored procedure description.

Figure 2: An example Stored Procedure Page

The stored procedure page includes:

- A description of the procedure's functionality.
- Input boxes for all required parameters.
- A submit button that triggers the procedure using PHP and MySQL. **After the**

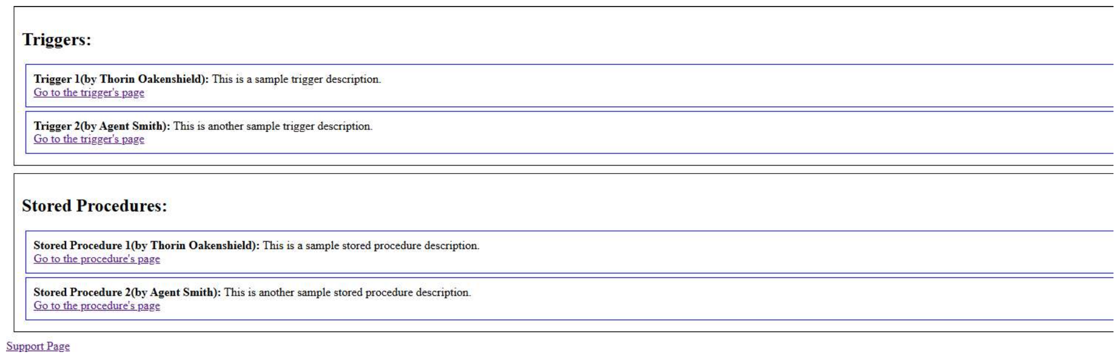
procedure is called using the interactive button, the output of the procedure should be displayed within the page.

- A hyperlink back to the homepage.

Navigation

At the bottom of the user homepage, there is a hyperlink to the support section, where users can create and manage support tickets. This ensures smooth navigation throughout the system.

Figure 3 shows an example layout for the home page of the user.



Triggers:

Trigger 1(by Thorin Oakenshield): This is a sample trigger description.
Go to the trigger's page

Trigger 2(by Agent Smith): This is another sample trigger description.
Go to the trigger's page

Stored Procedures:

Stored Procedure 1(by Thorin Oakenshield): This is a sample stored procedure description.
Go to the procedure's page

Stored Procedure 2(by Agent Smith): This is another sample stored procedure description.
Go to the procedure's page

Support Page

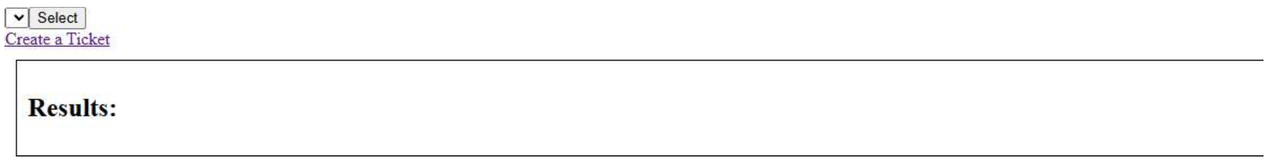
Figure 3: The layout of the users' home page

2.2. Support Ticket System (User Side)

The support ticket system allows users to request help and track the status of their issues. All ticket data is stored in a MongoDB collection, and the interface is designed to be simple but functional.

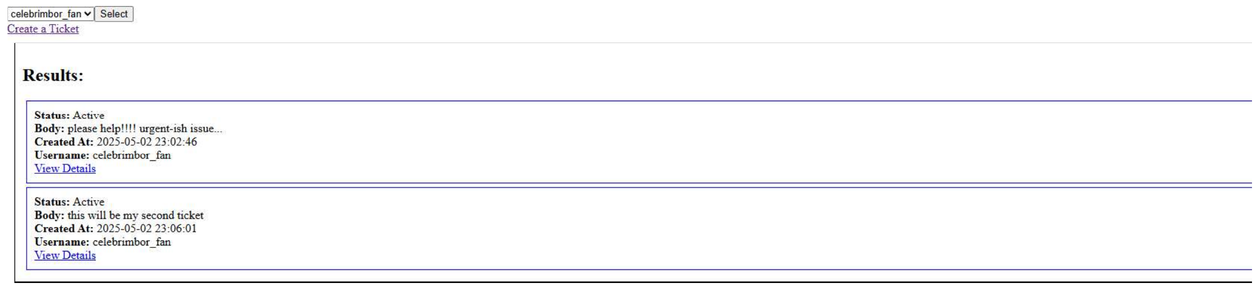
Ticket List Page

The first page of the support section presents a dropdown menu to select a username. This acts as a lightweight login system. Only users who have at least one active ticket appear in this list. Figure 4 shows how the Ticket List page should look like if there is no active ticket in the system. After selecting a username, the system displays their active tickets along with a link to create a new ticket. Figure 6 shows the usernames that have active tickets, whereas Figure 5 shows the tickets of the username *celebrimbor_fan* when it is selected from the dropdown menu.



Results:

Figure 4: An example Ticket List page when there is no active tickets

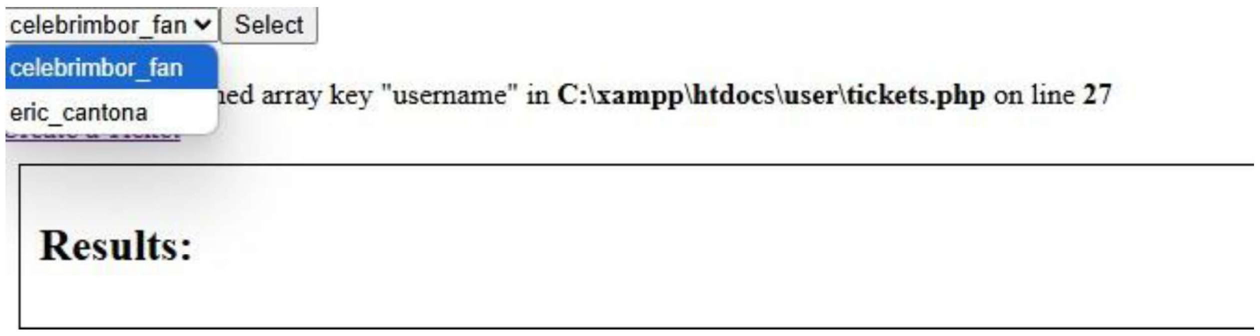


Results:

Status: Active
Body: please help!!!! urgent-ish issue...
Created At: 2025-05-02 23:02:46
Username: celebrimbor_fan
View Details

Status: Active
Body: this will be my second ticket
Created At: 2025-05-02 23:06:01
Username: celebrimbor_fan
View Details

Figure 5: An example Ticket List page with the username filter



ed array key "username" in **C:\xampp\htdocs\user\tickets.php** on line **27**

Results:

Figure 6: Dropdown menu when there are active tickets

Ticket Creation Page

The ticket creation page includes:

- Two navigation links at the top: one to return to the homepage and another to return to the ticket list page.
- Input fields for the username and the body of the ticket message.
- A submit button to create the ticket.

Each ticket also stores additional information automatically:

- The creation date and time (as a string).
- A boolean status indicating whether the ticket is active.
- An array of comments (initially empty).

Figure 7 shows an example page layout for ticket creation.

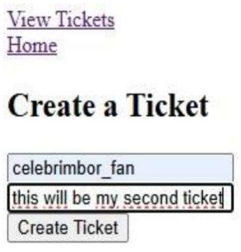


Figure 7: An example Ticket Creation Page

Ticket Confirmation

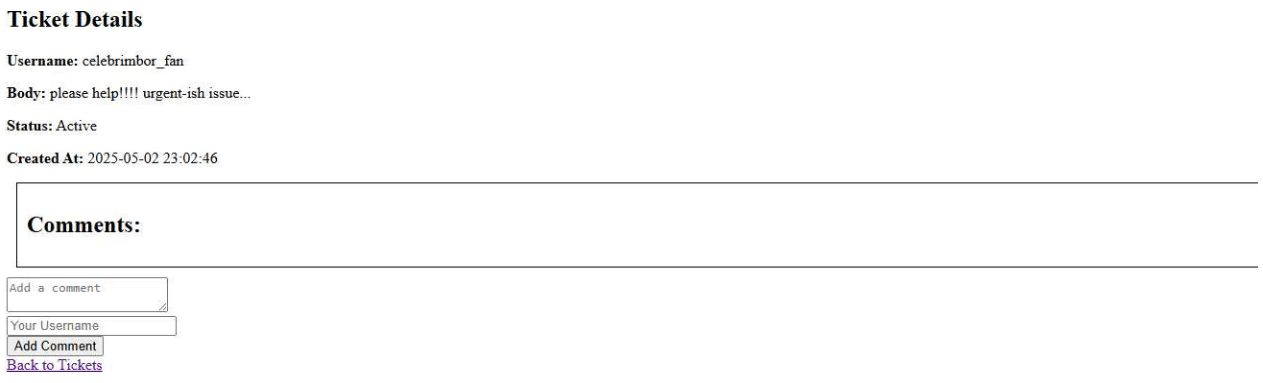
After submission, the system displays a confirmation page indicating whether the ticket was successfully created. This page also includes links to both create another ticket or return to the list of tickets.

Ticket Detail View

From the ticket list, users can click on a specific ticket to view its details. The detail page shows:

- Username of the ticket creator
- Creation date
- Ticket body
- Status (active/inactive)
- Comments

Users can also add new comments to the ticket. A link is provided at the bottom to return to the ticket list page. Figure 8 shows an example layout for the ticket creation page.



Ticket Details

Username: celebrimbor_fan

Body: please help!!!! urgent-ish issue...

Status: Active

Created At: 2025-05-02 23:02:46

Comments:

Figure 8: An example ticket details page

2.3. Admin Interface

The admin interface is accessed via the localhost/admin URL and is functionally similar to the user's support system but with broader permissions.

Admin Ticket List

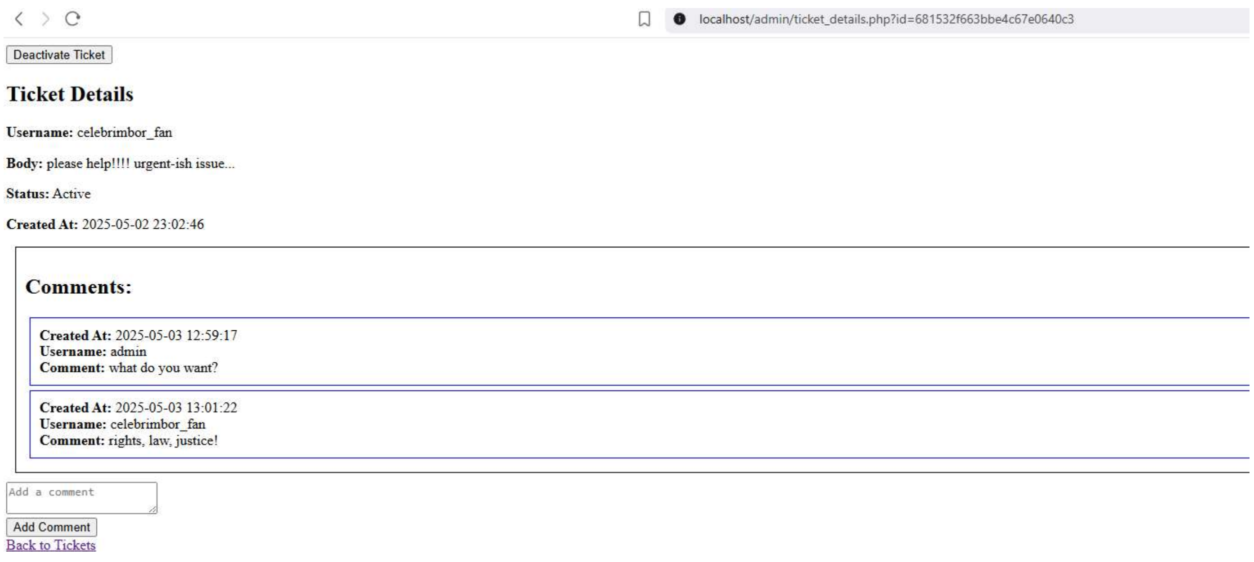
Unlike regular users, admins can see all active tickets from all users. The layout and features are similar, but the query in the backend retrieves a broader set of results. This page serves as the admin homepage.

Admin Ticket Detail View

Admins can view the details of any active ticket. In addition to the user's comments and metadata, admins can:

- Submit comments as the user admin (automatically assigned).
- Mark the ticket as resolved, which sets its status to false.

Once a ticket is resolved, it will no longer appear in either the user or admin homepage views, as both interfaces filter out inactive tickets. Figure 9 shows the admin's view of a ticket's details


localhost/admin/ticket_details.php?id=681532f663bbe4c67e0640c3
Deactivate Ticket
Add a comment
Add Comment
Back to Tickets

Ticket Details

Username: celebrimbor_fan

Body: please help!!!! urgent-ish issue...

Status: Active

Created At: 2025-05-02 23:02:46

Comments:

Created At: 2025-05-03 12:59:17
Username: admin
Comment: what do you want?

Created At: 2025-05-03 13:01:22
Username: celebrimbor_fan
Comment: rights, law, justice!

Figure 9: Admin's view of the ticket details

3. Implementation Requirements

This section covers the technical requirements for setting up and running the project. It includes configuration steps, technology stack, folder structure, and guidelines for integrating both MySQL and MongoDB with PHP through XAMPP.

3.1. Project Structure

The project is hosted locally using the XAMPP server. The directory structure is organized under the htdocs folder in XAMPP as follows:

- htdocs/user: Contains all PHP files for the user interface, including trigger and stored procedure pages and the support ticket system.

- htdocs/admin: Contains the admin's interface for viewing and managing support tickets.

Users can access the homepage via localhost/user, and admins can access their dashboard via localhost/admin.

3.2. Trigger and Stored Procedure Responsibilities

Each group member is responsible for integrating one trigger and one stored procedure. The triggers and procedures must be the ones developed in earlier phases of the database project. Integration responsibilities are displayed on the user homepage next to each feature's description.

3.3. MySQL Integration

MySQL is used to manage the core relational data of the system, including the logic behind triggers and stored procedures. All database operations (e.g., calling stored procedures, activating triggers via events) are implemented using PHP's MySQLi library.

Example tasks performed using MySQL:

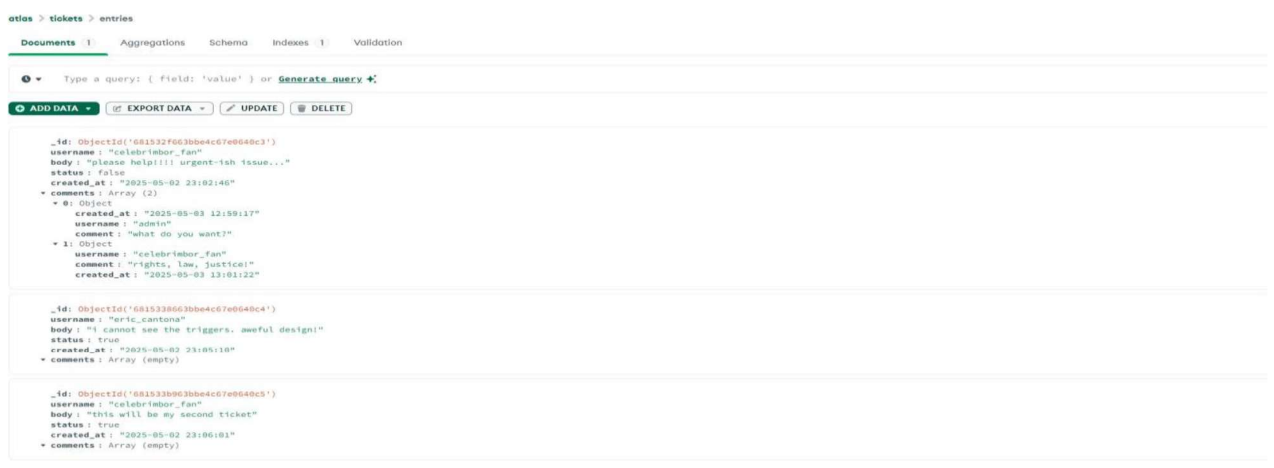
- Triggering actions based on database inserts/updates
- Executing stored procedures with user inputs

3.4. MongoDB Integration for Support Tickets

Support ticket operations are handled using MongoDB, with a single collection storing all ticket-related data. Each document in the collection contains:

- username: Creator of the ticket
- message: Body of the ticket
- created_at: Timestamp (stored as a string)
- status: Boolean value (true = active, false = resolved)
- comments: Array of comment strings

Figure 10 shows the final structure of the MongoDB collections.



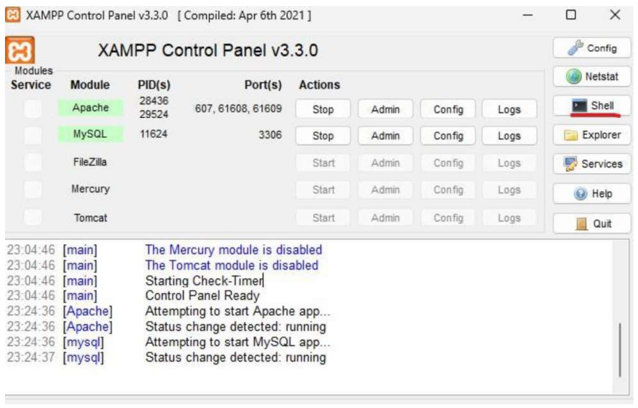
```
_id: ObjectId('681532f663bbe4c67e0640c3')
username : "celebrimbor_fan"
body : "please help!!!! urgent-ish issue..."
status : false
created_at : "2025-05-02 23:02:46"
comments : Array (2)
  0: Object
    created_at : "2025-05-03 12:59:17"
    username : "admin"
    comment : "what do you want?"
  1: Object
    username : "celebrimbor_fan"
    comment : "rights, law, justice!"
    created_at : "2025-05-03 13:01:22"

_id: ObjectId('6815338663bbe4c67e0640c4')
username : "eric_cantona"
body : "i cannot see the triggers. aweful design!"
status : true
created_at : "2025-05-02 23:05:10"
comments : Array (empty)

_id: ObjectId('681533b963bbe4c67e0640c5')
username : "celebrimbor_fan"
body : "this will be my second ticket"
status : true
created_at : "2025-05-02 23:06:01"
comments : Array (empty)
```

Figure 10: MongoDB Compass screenshot showing the documents in the collection

Installing the MongoDB Driver in XAMPP (Windows)

This section describes how to add MongoDB extension into PHP for Windows machines, In the next sections there will be links that may help Mac/Linux users. To enable MongoDB functionality in XAMPP on Windows:

1. Launch the terminal from the XAMPP control panel.


XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]
XAMPP Control Panel v3.3.0
Modules
Config
Netstat
Shell
Explorer
Services
Help
Quit

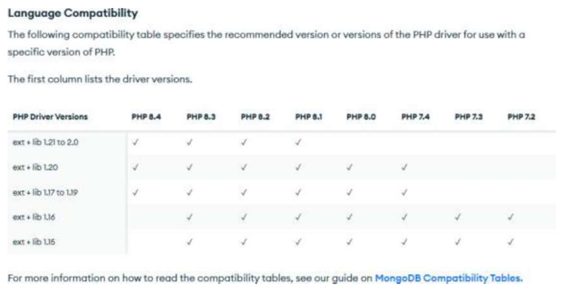
Service	Module	PID(s)	Port(s)	Actions			
	Apache	28436 29524	607, 61608, 61609	Stop	Admin	Config	Logs
	MySQL	11624	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
23:04:46  [main]     The Mercury module is disabled
23:04:46  [main]     The Tomcat module is disabled
23:04:46  [main]     Starting Check-Timer
23:04:46  [main]     Control Panel Ready
23:24:36  [Apache]   Attempting to start Apache app...
23:24:36  [Apache]   Status change detected: running
23:24:36  [mysql]    Attempting to start MySQL app...
23:24:37  [mysql]    Status change detected: running
```

2. Type composer -V to check the PHP and Composer versions. The output should be similar to the following:

```
Composer version 2.8.8 2025-04-04 16:56:46
PHP version 8.2.12 (C:\xampp\php\php.exe)
Run the "diagnose" command to get more detailed diagnostics output.
```

3. Visit the official PHP MongoDB driver download page to find a version matching your setup. You can find the compatible version from the following table:



Language Compatibility

The following compatibility table specifies the recommended version or versions of the PHP driver for use with a specific version of PHP.

The first column lists the driver versions.

PHP Driver Versions	PHP 8.4	PHP 8.3	PHP 8.2	PHP 8.1	PHP 8.0	PHP 7.4	PHP 7.3	PHP 7.2
ext + lib 1.21 to 2.0	✓	✓	✓	✓				
ext + lib 1.20	✓	✓	✓	✓	✓	✓		
ext + lib 1.17 to 1.19	✓	✓	✓	✓	✓	✓		
ext + lib 1.16		✓	✓	✓	✓	✓	✓	✓
ext + lib 1.15		✓	✓	✓	✓	✓	✓	✓

For more information on how to read the compatibility tables, see our guide on MongoDB Compatibility Tables.

4. Download the appropriate zip file and extract the php_mongodb.dll file.
5. Copy php_mongodb.dll into the xampp/php/ext directory.
6. Edit php.ini (located in xampp/php) and add the following line just above extension=curl:

```
extension = mongodb
```

7. In your project directory (inside htdocs/user and/or htdocs/admin), run:

```
composer require mongodb/mongodb
```

3.5. Focus on Functionality

While visual design is not graded, functional correctness is essential. Pages should include appropriate links for navigation, interactive elements for database integration, and meaningful status or error messages. The goal is to build a working prototype, not a polished interface.

3.6. Resources and References

The following tutorials and documentation may assist during implementation:

1. PHP + XAMPP setup: https://www.youtube.com/watch?v=jLqBiSDNXO0
2. PHP basics: https://www.youtube.com/watch?v=SI69DXbYHXU
3. _GET and _POST usage: https://www.youtube.com/watch?v=6AzAYU8AOhQ
4. PHP functions: https://www.youtube.com/watch?v=SBFO5bIDYHQ
5. MySQL + PHP: https://www.youtube.com/watch?v=-1DTYAQ25bY
6. Creating tables with phpMyAdmin: https://www.youtube.com/watch?v=BO6XOcfTyhU
7. MySQL insertion: https://www.youtube.com/watch?v=tq0ghtZsHJ0

8. Executing MySQL queries in PHP: https://www.youtube.com/watch?v=om_GHl6BrdM
9. MongoDB PHP Library: https://www.mongodb.com/docs/php-library/current/get-started/ (You are expexted to use MongoDB\Driver\Manager class, do not use the code sample provided in the link)
10. MongoDB\Driver\Manager reference: https://www.php.net/mongodb-driver-manager
11. MongoDB Compass: https://www.mongodb.com/products/tools/compass
12. PHP add extension on Ubuntu: https://stackoverflow.com/questions/78765759/add-extension-in-ubuntu-2204-to-xampp
13. PHP add extension on Mac: https://stackoverflow.com/questions/27886117/php-intl-installation-on-xampp

4. Submission and Evaluation

The final submission of the project must be packaged as a single zip archive. This archive will include both the project documentation and the accompanying SQL file required to recreate the database.

4.1. Contents of the Submission

- Package all files into a ZIP archive with the following structure:

```
|-- CS306_GROUP_X_PHASE3_REPORT.pdf
|-- CS306_GROUP_X_PHASE3_Demo_Video.mpg
|-- CS306_GROUP_X_PHASE3_SQLDUMP.sql
|-- scripts
|    |-- user
|    |    |-- index.php
|    |    |-- script_1.php
|    |    |-- .
|    |    |-- .
|    |    |-- script_n.php
|    |-- admin
|    |    |-- index.php
|    |    |-- script_1.php
|    |    |-- .
|    |    |-- .
|    |    |-- script_m.php
```

- **PDF Report**: A comprehensive document detailing the structure and functionality of the system.

The **FIRST PAGE** of the PDF report must include the course name, project title (Phase 3), group number, and the names of **ALL group members** (with roll numbers).

- For each trigger;
  - provide the SQL script that will create it,
  - state the different cases and corresponding buttons in the web page of it.
- For each stored procedure;
  - provide the SQL script that will create it,
  - state the input parameters and corresponding input boxes in the web page of it.
- For each MongoDB query implemented for the support system;
  - provide the PHP script that corresponds to the query.

- **SQL File**: A single SQL script file containing the statements for:
  - Creation of the database and relevant tables.
  - Insertion of mock data to test the project.
  - Definitions of all triggers and stored procedures used in the project.
- **Scripts**: All the PHP scripts of the web application. *script i.php* is a placeholder, you can name your scripts freely.

4.2. Demo and Grading Policy

Project evaluation will be based on both the submitted PDF report and the recorded demo video.

Recorded Demo Video (70%):
Each group must submit one recorded video, with a maximum length of 3 minutes, demonstrating:
- The correct execution of each trigger and stored procedure
- The full workflow of the support ticket system (ticket creation, listing, management)
- All required components running on the student's own development environment

Speaker could be one of the group members appear in the video and clearly speak about what he/she does during video.
The video must show the actual environment, including the running servers, database connections, and web interfaces.

PDF Report (30%):
Students must submit a well-structured PDF report containing:
- A brief explanation of each implemented trigger and stored procedure
- Screenshots of successful executions
- A description of the support ticket system workflow
- Any assumptions, challenges, or design decisions
- The final SQL file and PHP integration notes (as appendices if needed)

Both components will be evaluated for clarity, correctness, completeness, and demonstration of understanding.

4.3. Demo Video Requirements

Before recording the demo, each student must ensure that the following software and configurations are correctly installed and visible in the video:

- XAMPP server with Apache and MySQL running
- MongoDB Compass (or any interface used to inspect the MongoDB ticket database)
- Configured MongoDB PHP extension and functional project folder
- Browsers or tools used to display web interfaces
- The recorded demo video must clearly demonstrate:
- Working web interfaces connected to each trigger and stored procedure
- Functional ticket creation, listing, and management
- Active connections between PHP → MySQL and PHP → MongoDB

Although the support ticket system is developed collaboratively, the video must show that any group member can present and run it.