

**CSE370 : Database Systems**

**Project Report**

**Project Title : Lost & Found Shelf**

|  |  |  |
| --- | --- | --- |
| **Group No : 14, CSE370 Lab Section : 14, Summer 2025** | | |
| **ID** | **Name** | **Contribution** |
| 22299112 | Ahad Islam Shanto |  |
| 24241346 | Binita Ghosh |  |
|  |  |  |

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Section**  **No** | **Content** | **Page No** |
| 1 | Introduction | 3 |
| 2 | Project Features | 4 |
| 3 | ER/EER Diagram | 5 |
| 4 | Schema Diagram | 6 |
| 5 | Normalization | 7 |
| 6 | Frontend Development | 8 |
| 7 | Backend Development | 10 |
| 8 | Source Code Repository | 16 |
| 9 | Conclusion | 16 |
| 10 | References | 18 |

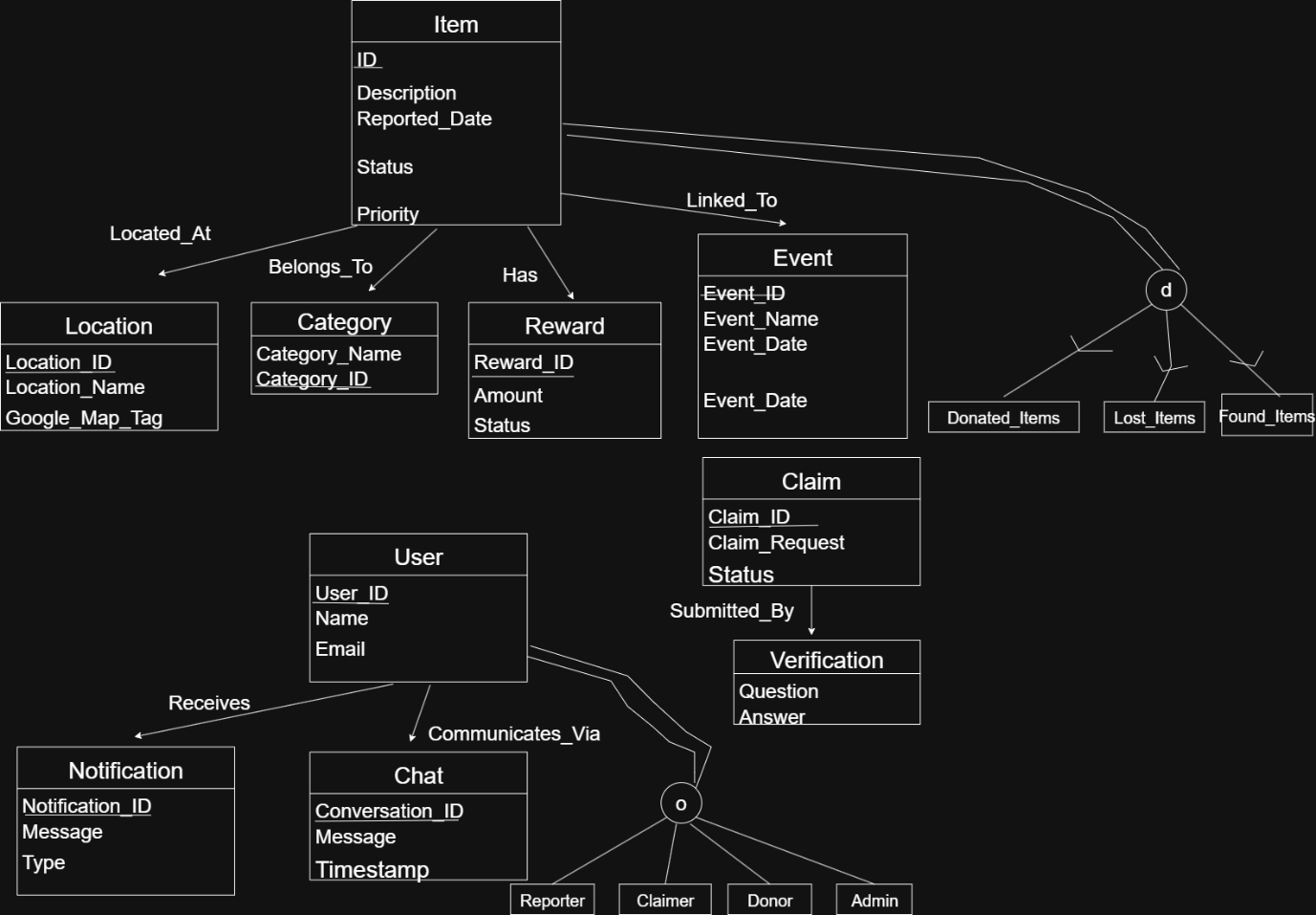
**Introduction**

The name of our project is “Lost & Found Shelf”. The idea behind the project is to create a useful website keep in mind the moto Reunite People with their Belongings. With the help of this website, we can create an account and claim our lost item. We can categorize it and share the location where the item got stolen. We can chat with the users if anyone finds the item and we can claim it by providing enough evidence via question reviewing to find out if the item belongs to them or not. There are a reward system and a donation system. The reward can be announced by either who lost the item or who found the item. The donation system works when the expiry date of an item is over. It is a platform for all where anyone can get their lost products without bare minimum of the hassle.

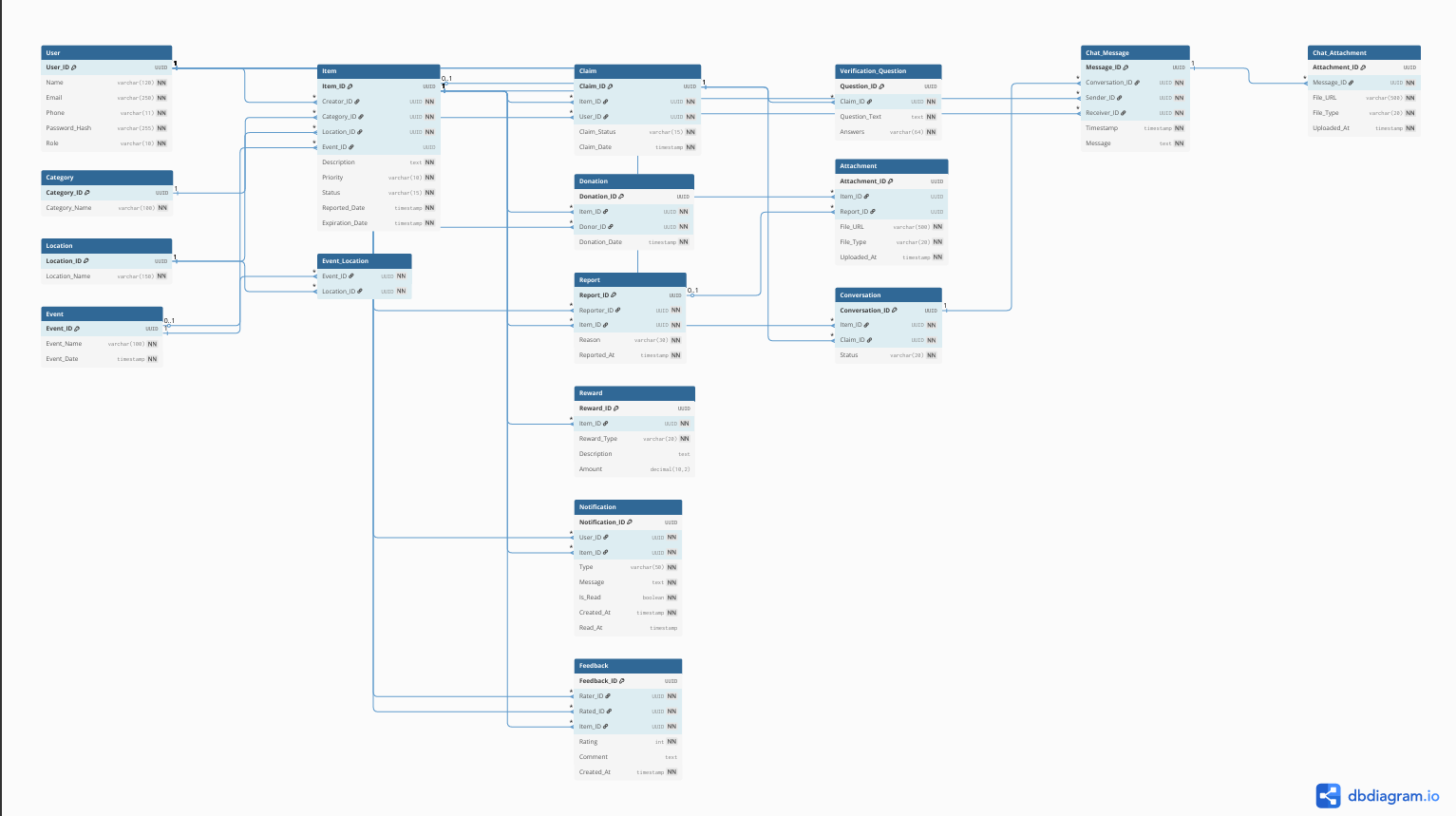
**Project Features**

|  |  |  |
| --- | --- | --- |
| **ID,**  **Name** | **Features [2 member]** | |
| 22299112  Ahad Islam Shanto | Ft 1 | Report Item |
| Ft 2 | Claim Item with Verification question |
| Ft 3 | Bounty and rest of it 🥲 |
| 24241346  Binita Ghosh | Ft 1 | Chat System |
| Ft 2 | Notification |
| Ft 3 | Donation |

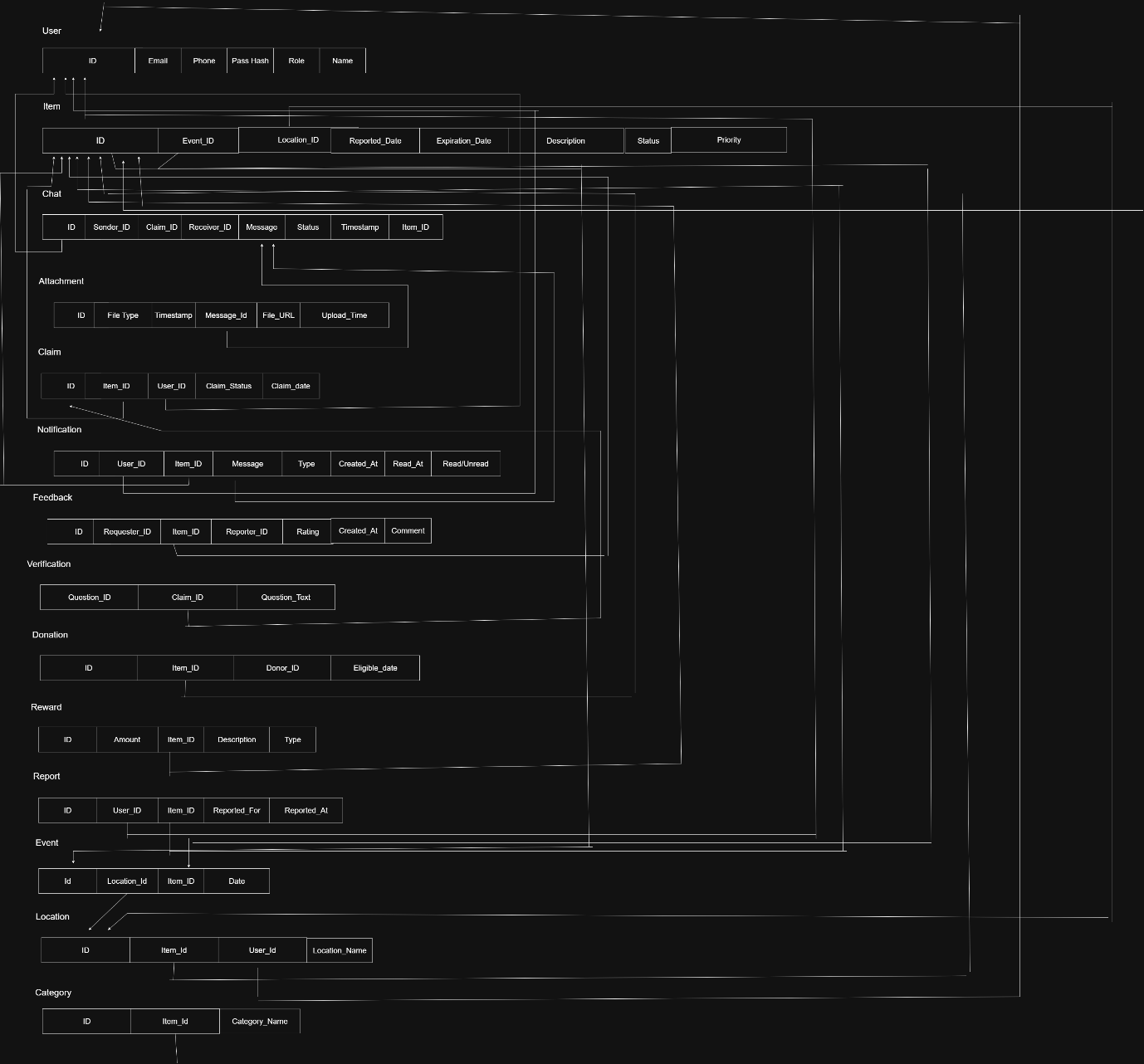
**ER/EER Diagram**



**Schema Diagram**



**Normalization**



It is converted into 3NF but has transitive dependencies. For an instance, the Donor\_ID and Reward\_ID.

**Frontend Development**

**Contribution of ID:** 22299112**, Name: Ahad Islam Shanto**

1. ***Category Dropdown in report\_lost.php***This HTML and PHP code snippet shows how a responsive dropdown menu for categories is generated.

<div class="col-md-6">

<div class="mb-3">

<label for="category\_id" class="form-label">Category \*</label>

<select class="form-select" id="category\_id" name="category\_id" required>

<option value="">Select Category</option>

<?php foreach ($categories as $category): ?>

<option value="<?php echo $category['Category\_ID']; ?>">

<?php echo htmlspecialchars($category['Category\_Name']); ?>

</option>

<?php endforeach; ?>

</select>

</div>

</div>

1. **Claim Item with Verification Questions**

To ensure items are returned to their rightful owners, I implemented a secure and detailed claim process.

* claim\_item.php  
  The user with a three-question verification form. The frontend clearly displays the item being claimed and provides three text areas for the user to answer questions that help prove their ownership. This form is a crucial part of our item claiming.

1. 3. Bounty / Reward System

To initiate the return of lost items faster a bounty system was implemented.

* **report\_lost.php & view\_item.php**

On the report item page, a dedicated "Bounty/Reward" section to the form, allowing users to specify a reward type (e.g., money, services and other type) with or without amount .

On the item details page (view\_item.php), alert box is added to inform the users that a bounty is being offered for the item, making it highly lookrative.

**Contribution of ID: 24241346, Name: Binita Ghosh**

**1. Chat System**

The chat system is one of the most dynamic and interactive parts of our website.

* **chat.php**This file contains the complete code for the chat interface. The front end is a two-paned layout created with Bootstrap 5’s grid system.
  + Left Pane: Lists all of the user's conversations, with unread message badges.
  + Right Pane: Displays the active chat window, with messages styled differently for the sender and receiver.

The front end heavily relies on automatically refreshing. The chat panes every 5 seconds, handle message submissions without a page reload, and manage file attachment previews before sending.

<div class="message <?php echo $message['Sender\_ID'] == $\_SESSION['user\_id'] ? 'sent' : 'received'; ?>">

<div class="message-bubble">

<!-- Message text and attachments go here -->

</div>

</div>

**2. Notification System**

To keep users updated, a notification system was implemented throughout the application.

* **index.php & notifications.php**
  + On the dashboard (index.php) there is added a section to display the most recent unread notifications, giving users a quick overview of recent activity.
  + The dedicated notifications page (notifications.txt) displays a full history of all notifications, with icons and colors corresponding to the notification type (e.g., green for approved, red for rejected).

**3. Donation Feature**

This feature allows users to donate unclaimed items to charity.

* **donate\_item.php**This file contains the frontend for the donation process. The interface clearly informs the user why the item is eligible for donation (e.g., it has expired). It includes a form to specify a recipient organization and requires the user to check a confirmation box, which is a key UI element to prevent accidental donations.

**Backend Development**

**Contribution of ID:** 22299112**, Name: Ahad Islam Shanto**

**The backend of this application is the engine that powers all of its features. It is developed using PHP and connects to a MariaDB database. Individual PHP scripts have been used, each responsible for a specific feature, making the system modular and easier to manage.**

1. **PDO database connection (safe defaults)**

Database Interaction: Securely creating, reading, updating, and deleting data from the database using PDO (PHP Data Objects).

// PDO connection

$dsn = "mysql:host=localhost;dbname=lost\_found\_db;charset=utf8mb4";

$options = [

    PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION,

    PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_ASSOC,

    PDO::ATTR\_EMULATE\_PREPARES => false,

];

$db = new PDO($dsn, $dbUser, $dbPass, $options);

1. Auth middleware (session check)

<?php

// require-auth.php (include at top of protected pages)

session\_start();

if (!isset($\_SESSION['user\_id'])) {

    header('Location: auth.php');

    exit();

}

1. **Safe SELECT / UPDATE with prepared statements**

<?php

// Fetch user

$stmt = $db->prepare("SELECT User\_ID, Email, Password\_Hash, Push\_Notifications FROM User WHERE User\_ID = ?");

$stmt->execute([$userId]);

$user = $stmt->fetch()

// Update push preference

$stmt = $db->prepare("UPDATE User SET Push\_Notifications = ? WHERE User\_ID = ?");

$stmt->execute([$pushValue, $userId]);

1. **Password Hashing**

register.php script handles the new user registration process. Its backend logic validates all user inputs (name, email, phone), checks if the email or phone number already exists in the database to prevent duplicates and uses PHP's password\_hash() function to securely store passwords.

try {

$new\_password\_hash = password\_hash($new\_password, PASSWORD\_DEFAULT);

$stmt = $db->prepare("UPDATE User SET Password\_Hash = ?, Updated\_At = CURRENT\_TIMESTAMP WHERE User\_ID = ?");

if ($stmt->execute([$new\_password\_hash, $\_SESSION['user\_id']])) {

$success = 'Password updated successfully!';

$password\_changed = true;

} else {

$error = 'Failed to update password. Please try again.';

}

} catch (Exception $e) {

$error = 'An error occurred while updating your password.';

}

}

**Item and Claim Management**

This is the central functionality of the application allowing users to report and view their items.

* report\_lost.php & report\_found.php  
  These two files contain the backend logic for adding new items to the system. They handle form submissions, validate all the data (description, category, location), process optional image uploads for the item, and insert new records into the Item, Attachment, and Reward tables.
* **index.php (Dashboard)**  
  The backend logic in the dashboard is responsible for fetching all the dynamic data a user sees when they log in. It runs multiple optimized SQL queries to get the user's name, a list of the 10 most recent items, the user's unread notifications, and key statistics (like total items and personal claims).
* **view\_item.php**  
  When a user views a single item, this script's backend fetches all related information for that specific Item\_ID. It uses complex SQL JOIN statements to retrieve the item's details, category name, location name, creator's name, and all associated attachments and claims from different tables in a single, efficient operation.

This SQL query demonstrates how the backend joins multiple tables to gather all necessary information for the item details page.

SELECT i.\*, c.Category\_Name, l.Location\_Name, u.Name as Creator\_Name

FROM Item i

LEFT JOIN Category c ON i.Category\_ID = c.Category\_ID

LEFT JOIN Location l ON i.Location\_ID = l.Location\_ID

LEFT JOIN User u ON i.Creator\_ID = u.User\_ID

WHERE i.Item\_ID = ?

**Contribution of ID: 24241346, Name: Binita Ghosh**

**Chat System**

The backend of the chat system is designed to facilitate real-time communication between users, in context of a lost/found item claim. The system's foundation lies in a relational database schema that tracks conversations, messages, and participants.

CREATE TABLE IF NOT EXISTS ChatMessage (

MessageID VARCHAR(191) PRIMARY KEY,

ConversationID VARCHAR(191) NOT NULL,

SenderID VARCHAR(191) NOT NULL,

MessageText TEXT NOT NULL,

MessageType ENUM('text', 'image', 'file') NOT NULL DEFAULT 'text',

IsRead BOOLEAN NOT NULL DEFAULT FALSE,

SentAt TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (ConversationID) REFERENCES Conversation(ConversationID) ON DELETE CASCADE,

FOREIGN KEY (SenderID) REFERENCES User(UserID) ON DELETE CASCADE,

INDEX idx\_conversation (ConversationID),

INDEX idx\_sent\_at (SentAt)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

* Fetching Conversations: The system retrieves a list of all conversations for the currently logged-in user. Then it joins the Conversation, Item, and User tables to display relevant information like the other participant's name and the last exchanged message.
* Sending Messages: When a user sends a message, the backend script receives the text and any attachments. It inserts the new message into the ChatMessage table and handles file uploads to the ChatAttachment table.

**// Get user's conversations**

**$stmt = $db->prepare("**

**SELECT c.\*, i.Description as ItemDescription,**

**CASE WHEN c.Participant1ID = ? THEN p2.Name ELSE p1.Name END as OtherParticipantName,**

**(SELECT cm.MessageText FROM ChatMessage cm WHERE cm.ConversationID = c.ConversationID ORDER BY cm.SentAt DESC LIMIT 1) as LastMessage**

**FROM Conversation c**

**LEFT JOIN Item i ON c.ItemID = i.ItemID**

**LEFT JOIN User p1 ON c.Participant1ID = p1.UserID**

**LEFT JOIN User p2 ON c.Participant2ID = p2.UserID**

**WHERE c.Participant1ID = ? OR c.Participant2ID = ?**

**ORDER BY c.UpdatedAt DESC**

**");**

**$stmt->execute([$\_SESSION['userid'], $\_SESSION['userid'], $\_SESSION['userid']]);**

**$conversations = $stmt->fetchAll();**

**Notification System**

The notification system is designed to keep users informed about important events and actions related to their items and claims.

// Create notification for item owner

$notif\_id = 'notif\_' . uniqid();

$message = "Someone has claimed your item: " . substr($item['Description'], 0, 50) . "...";

$stmt = $db->prepare("

INSERT INTO Notification (NotificationID, UserID, ItemID, Type, Title, Message, IsRead, CreatedAt)

VALUES (?, ?, ?, 'claim\_received', 'New Claim Received', ?, 0, ?)

");

$stmt->execute([$notif\_id, $item['CreatorID'], $itemid, $message, $claim\_date]);

**Donation System**

The Donation table records all donation activities. It stores the ID of the donated item, the ID of the donor, the date of the donation.

// Process the donation

$donation\_id = 'don\_' . uniqid();

$stmt = $db->prepare("

INSERT INTO Donation (DonationID, ItemID, DonorID, RecipientOrganization, Notes, DonationDate)

VALUES (?, ?, ?, ?, ?, NOW())

");

$stmt->execute([$donation\_id, $item\_id, $\_SESSION['userid'], $recipient, $notes]);

// Update the item status to 'donated'

$stmt = $db->prepare("UPDATE Item SET Status = 'donated' WHERE ItemID = ?");

$stmt->execute([$item\_id]);

**Source Code Repository**

[**https://github.com/BR4CKUD3R/lost-and-found-shelf/tree/master**](https://github.com/BR4CKUD3R/lost-and-found-shelf/tree/master)

[**https://drive.google.com/drive/u/1/folders/1fNxX9\_RksMc1E9Bsk7n06Ln3ihimJShZ**](https://drive.google.com/drive/u/1/folders/1fNxX9_RksMc1E9Bsk7n06Ln3ihimJShZ)

**Conclusion**

The Lost & Found Shelf Management System is built with PHP and optimized for MariaDB. Which stands as a comprehensive and well-rounded application designed to streamline the process of managing lost and found items and both of them are open-sourced. The system's strength lies in its thoughtful combination of a robust backend, a feature-rich database, and a modern, user-friendly frontend. We tried to use the power of Bootstrap 5. The user interface is not only visually appealing but also fully responsive while ensuring a good experience across all web browsers.

The project excels in its core functionalities. The item reporting process is intuitive, allowing users to provide detailed descriptions, categorize their items, specify locations, and even upload images, which significantly increases the chances of a successful recovery . One of the standout features is the secure, three-question verification system for claims, which ensures that items are returned only to their rightful owners. This, combined with the integrated chat system, provides a secure and direct line of communication between the finder and the claimant which ensures trust and efficiency in the recovery process.

Furthermore, the real-time notification system keeps users instantly informed of any updates on their items or claims, while the optional bounty system introduces a powerful incentive for the community to actively participate in returning lost items. The dashboard provides users with an at-glance overview of their activity, including their reported items, active claims, and unread notifications, making the platform easy to navigate and manage. From a technical standpoint, the database schema is well-designed and optimized for performance, utilizing MariaDB-specific features like proper indexing and appropriate data types to ensure fast and efficient queries. The backend is built on the foundation of PHP. Overall, this project is a useful around user-centric solution that effectively addresses the minimal real-world challenge of managing lost and found items in any community. There is headroom for further optimization and update. The project delivers a practical lost-and-found management solution that meets the course requirements and provides a clear foundation for further enhancements. Our project delivers a practical lost-and-found management solution that meets the course requirements and provides a foundation for further changes/ updates.

**References**

[**https://eflairwebtech.com/php-and-sql/**](https://eflairwebtech.com/php-and-sql/)

[**https://phpbestpractices.org/**](https://phpbestpractices.org/)

[**https://phpdelusions.net/pdo\_examples/connect\_to\_mysql**](https://phpdelusions.net/pdo_examples/connect_to_mysql)

[**https://www.w3schools.com/php/php\_mysql\_connect.asp**](https://www.w3schools.com/php/php_mysql_connect.asp)

[**https://www.geeksforgeeks.org/php/what-is-pdo-in-php/**](https://www.geeksforgeeks.org/php/what-is-pdo-in-php/)

[**https://stackoverflow.com/questions/19648760/project-structure-for-php**](https://stackoverflow.com/questions/19648760/project-structure-for-php)

[**https://www.vaadata.com/blog/php-security-best-practices-vulnerabilities-and-attacks/**](https://www.vaadata.com/blog/php-security-best-practices-vulnerabilities-and-attacks/)

[**https://www.repoapp.com/features/**](https://www.repoapp.com/features/)

[**https://www.lucidchart.com/pages/tutorial/database-design-and-structure**](https://www.lucidchart.com/pages/tutorial/database-design-and-structure)

[**https://www.youtube.com/watch?v=Zar\_nkweCKs**](https://www.youtube.com/watch?v=Zar_nkweCKs)

[**https://stackoverflow.com/questions/8705195/how-do-i-select-a-mysql-database-to-use-with-pdo-in-php**](https://stackoverflow.com/questions/8705195/how-do-i-select-a-mysql-database-to-use-with-pdo-in-php)

[**https://www.youtube.com/watch?v=fg1rjQOOWoQ**](https://www.youtube.com/watch?v=fg1rjQOOWoQ)

[**https://echost.net/knowledge-base/web-hosting-help/ftp-file-manager/cpanel-file-permissions-%E0%A6%A0%E0%A6%BF%E0%A6%95-%E0%A6%95%E0%A6%B0%E0%A6%AC%E0%A7%87%E0%A6%A8-%E0%A6%95%E0%A7%80%E0%A6%AD%E0%A6%BE%E0%A6%AC%E0%A7%87-644-vs-755-%E0%A6%8F%E0%A6%AC%E0%A6%82/**](https://echost.net/knowledge-base/web-hosting-help/ftp-file-manager/cpanel-file-permissions-%E0%A6%A0%E0%A6%BF%E0%A6%95-%E0%A6%95%E0%A6%B0%E0%A6%AC%E0%A7%87%E0%A6%A8-%E0%A6%95%E0%A7%80%E0%A6%AD%E0%A6%BE%E0%A6%AC%E0%A7%87-644-vs-755-%E0%A6%8F%E0%A6%AC%E0%A6%82/)

[**https://stackoverflow.com/questions/59986082/php-how-to-properly-check-mime-type-of-a-file**](https://stackoverflow.com/questions/59986082/php-how-to-properly-check-mime-type-of-a-file)

[**https://forum.bricksbuilder.io/t/solved-fallback-displayed-instead-of-source-image/25509**](https://forum.bricksbuilder.io/t/solved-fallback-displayed-instead-of-source-image/25509)

[**https://support.cloudinary.com/hc/en-us/community/posts/4790106624018-Get-URL-in-Callback-Response-with-Simple-Upload-Form**](https://support.cloudinary.com/hc/en-us/community/posts/4790106624018-Get-URL-in-Callback-Response-with-Simple-Upload-Form)

[**https://muz.li/inspiration/chat-ui/**](https://muz.li/inspiration/chat-ui/)

[**https://www.rst.software/blog/chat-app-architecture**](https://www.rst.software/blog/chat-app-architecture)

[**https://www.behance.net/search/projects/Chat%20App%20UI**](https://www.behance.net/search/projects/Chat%20App%20UI)

[**https://www.cometchat.com/blog/chat-app-design-best-practices**](https://www.cometchat.com/blog/chat-app-design-best-practices)

[**https://github.com/rosell-dk/image-mime-type-guesser**](https://github.com/rosell-dk/image-mime-type-guesser)

[**https://www.youtube.com/watch?v=7T-HnTE6v64**](https://www.youtube.com/watch?v=7T-HnTE6v64)

[**https://hackmd.io/@oidf-wg-authzen/S1inmizEa**](https://hackmd.io/@oidf-wg-authzen/S1inmizEa)

[**https://vertabelo.com/blog/user-authentication-module/**](https://vertabelo.com/blog/user-authentication-module/)

[**https://getstream.io/chat/docs/react/tokens\_and\_authentication/**](https://getstream.io/chat/docs/react/tokens_and_authentication/)

[**https://stackoverflow.com/questions/62468691/how-to-authenticate-websocket-chat-by-jwt**](https://stackoverflow.com/questions/62468691/how-to-authenticate-websocket-chat-by-jwt)

[**https://docs.aws.amazon.com/apigateway/latest/developerguide/websocket-api-chat-app.html**](https://docs.aws.amazon.com/apigateway/latest/developerguide/websocket-api-chat-app.html)

[**https://github.com/YKLoh666/chat-app**](https://github.com/YKLoh666/chat-app)

[**https://spring.io/guides/gs/messaging-stomp-websocket**](https://spring.io/guides/gs/messaging-stomp-websocket)