

Database Management System : UE23CS351A

Level 2 - Orange Problem

LocateU-Lost-And-Found-System

Team member1:-

Name: Pratheek J Gowda

SRN: PES1UG23CS448

Section: H

Team member2:-

Name: Rajat Ramakrishna Bhat

SRN: PES1UG23CS465

Section: H

Description :

LocateU is a comprehensive Flask + MySQL web application designed to simplify the process of reporting, tracking, and claiming lost or found items within an institution. It features dual-role authentication with separate student and staff dashboards, image upload and storage with BLOB database integration, intelligent automated matching using string similarity algorithms (SequenceMatcher) to suggest matches based on item names, categories, locations, and dates, real-time notifications with role-based filtering for personalized user alerts, and a multi-stage verification workflow where staff can verify item ownership before approving claims. The system ensures data integrity and transparency through stored procedures, database triggers, and comprehensive transaction logging.

User Requirement Specification :

Purpose of the Project

The purpose of this project is to design and develop a university-level Lost and Found Management System that allows students and administrators to record, track, and manage lost and found items efficiently. In most academic institutions, students frequently lose their personal belongings such as books, ID cards, mobile phones, and other valuables. Similarly, found items often remain unclaimed for long periods because of a lack of proper reporting and tracking mechanisms. This project aims to bridge that gap by providing a structured and reliable database-driven system that makes it simple for students to report lost items, submit details of found items, and check the status of their claims.

The project also ensures accountability and transparency in handling lost property within the university campus. By using this system, we seek to reduce the time and effort that students and staff currently spend on searching for lost items, while also minimizing the risk of unclaimed or misplaced belongings being discarded without proper notification. The purpose is to create a digital solution that ensures that lost belongings are returned to their rightful owners in a smooth and organized manner.

Scope of the Project

The scope of the project covers the design and implementation of a fully functional Lost and Found Management System that will be accessible to students and university administrators. Students will be able to register into the system with their details, report lost items by providing descriptions, upload information about items they have found, and track the status of their claims. Administrators will be able to verify reports, manage the database of lost and found items, confirm ownership based on student details, and maintain a proper record of transactions.

The system will be developed with a backend database that ensures data consistency and reliability, while also supporting essential features like unique student records, item categorization, and claim verification. The scope also includes ensuring that duplicate entries are prevented and that there is a secure way to validate student identity before releasing any claimed item. The project does not extend to developing mobile applications or integrating external third-party tools but focuses entirely on building a robust web-based platform supported by a relational database.

Detailed Description

The Lost and Found Management System is a database-driven application designed to handle the process of recording, storing, and tracking lost and found items within the university campus. Students are at the heart of the system, and every student is required to have a unique record that includes their name, email, phone number, department, and year of study. Once registered, students can interact with the system in two main ways. Firstly, if a student loses an item, they can log into the system and submit details of the item such as the name of the item, a description, the location where it was lost, and the approximate date of loss. Secondly, if a student finds an item, they can provide a similar record which includes the category of the item, its condition, the place where it was found, and the date it was retrieved.

All records are stored in the database and can later be searched and filtered by both students and administrators. The system ensures that every lost or found item is assigned a unique identifier, allowing for accurate tracking. When a student comes to claim an item, the administrator verifies the ownership by checking the details provided by both the owner and the finder. Once verified, the administrator updates the database to mark the item as claimed, ensuring that it is no longer listed as available.

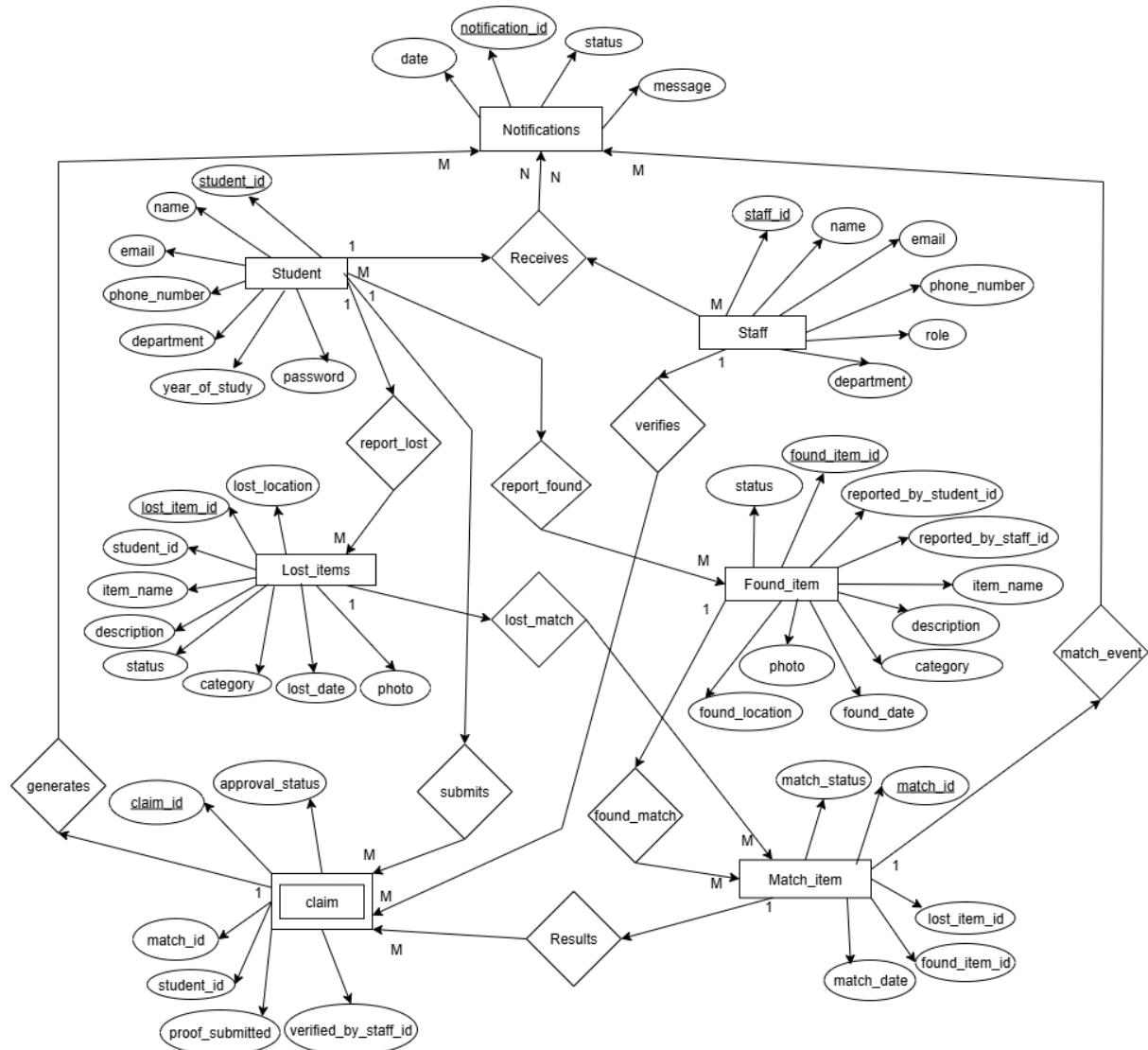
The system also maintains transparency and prevents misuse by ensuring that no two students can register with the same email or phone number. This prevents duplication and makes it easier for administrators to manage student records. Additionally, reports of lost and found items are time-stamped so that administrators can prioritize recent claims. The database is designed to handle multiple categories of items, ranging from books and stationery to electronics and personal belongings, making the system versatile enough to address the diverse needs of the university community.

In practical terms, the system acts as a bridge between students who have lost belongings and those who have found them. Instead of relying on notice boards, word of mouth, or manual registers, the Lost and Found Management System provides a centralized and secure digital platform that keeps all information in one place. It reduces confusion, saves time, and ensures that students can recover their belongings more efficiently. By implementing this project, we are providing not only a solution to an existing problem but also a model that can be adapted by other institutions to improve campus life through better management of lost and found items.

List of Softwares/Tools/Programming languages used:-

- HTML
- CSS
- JavaScript
- Python
- MySQL
- Flask
- Werkzeug
- mysql.connector
- python-dotenv
- difflib
- io & base64
- MySQL Workbench
- VS Code
- Git and Github

E-R Diagram:



Relational Schema:

Student

Student_id	name	email	Phone_no	department	Year_Study	password
------------	------	-------	----------	------------	------------	----------

Staff

Staff_id	name	Phone_no	email	Role	department
----------	------	----------	-------	------	------------

Lost_items

Lost_item_id	Student_id	Category	Lost_date	Lost_loc	status	Item_name	photo	description
--------------	------------	----------	-----------	----------	--------	-----------	-------	-------------

Found_items

F_I_id	Report_student_id	Report_staff_id	Item_name	description	Category	Found_date	Found_Loc	Status	photo
--------	-------------------	-----------------	-----------	-------------	----------	------------	-----------	--------	-------

Match-items

Match_id	Lost_item_id	F_I_id	Match_date	Status
----------	--------------	--------	------------	--------

Claims

Claim_id	Match_id	Student_id	Proof_submitted	Approval_status	Verified_by_Staff_id
----------	----------	------------	-----------------	-----------------	----------------------

Notifications

Notification_id	message	date	status
-----------------	---------	------	--------

DDL Commands :

```

1 •  DROP DATABASE IF EXISTS LostAndFoundDB;
2 •  CREATE DATABASE LostAndFoundDB;
3 •  USE LostAndFoundDB;
4 •  CREATE TABLE student (
5     student_id INT PRIMARY KEY AUTO_INCREMENT,
6     name VARCHAR(100) NOT NULL,
7     email VARCHAR(100) UNIQUE NOT NULL,
8     phone_number VARCHAR(15) UNIQUE NOT NULL,
9     department VARCHAR(50) NOT NULL,
10    year_of_study INT CHECK (year_of_study BETWEEN 1 AND 5),
11    password VARCHAR(255) NOT NULL
12 );
13 •  CREATE TABLE staff (
14     staff_id INT PRIMARY KEY AUTO_INCREMENT,
15     name VARCHAR(100) NOT NULL,
16     email VARCHAR(100) UNIQUE NOT NULL,
17     phone_number VARCHAR(15) UNIQUE NOT NULL,
18     role VARCHAR(50) NOT NULL,
19     department VARCHAR(50) NOT NULL,
20     password VARCHAR(255) NOT NULL
21 );

```

```

22 • Ⓜ CREATE TABLE lost_items (
23     lost_item_id INT PRIMARY KEY AUTO_INCREMENT,
24     student_id INT,
25     category VARCHAR(50) NOT NULL,
26     lost_date DATE NOT NULL,
27     lost_loc VARCHAR(100) NOT NULL,
28     status VARCHAR(20) DEFAULT 'Unresolved',
29     item_name VARCHAR(100) NOT NULL,
30     photo LONGBLOB NULL,
31     description TEXT,
32     FOREIGN KEY (student_id) REFERENCES student(student_id) ON DELETE CASCADE
33 );
34 • Ⓜ CREATE TABLE found_items (
35     f_i_id INT PRIMARY KEY AUTO_INCREMENT,
36     report_student_id INT NULL,
37     report_staff_id INT NULL,
38     item_name VARCHAR(100) NOT NULL,
39     description TEXT,
40     category VARCHAR(50) NOT NULL,
41     found_date DATE NOT NULL,
42     found_loc VARCHAR(100) NOT NULL,
43     status VARCHAR(20) DEFAULT 'Unclaimed',
44     photo LONGBLOB NULL,
45     FOREIGN KEY (report_student_id) REFERENCES student(student_id) ON DELETE SET NULL,
46     FOREIGN KEY (report_staff_id) REFERENCES staff(staff_id) ON DELETE SET NULL
47 );
48 • Ⓜ CREATE TABLE match_items (
49     match_id INT PRIMARY KEY AUTO_INCREMENT,
50     lost_item_id INT NOT NULL,
51     f_i_id INT NOT NULL,
52     match_date DATE NOT NULL,
53     status VARCHAR(20) DEFAULT 'Pending',
54     FOREIGN KEY (lost_item_id) REFERENCES lost_items(lost_item_id) ON DELETE CASCADE,
55     FOREIGN KEY (f_i_id) REFERENCES found_items(f_i_id) ON DELETE CASCADE
56 );
57 • Ⓜ CREATE TABLE claims (
58     claim_id INT PRIMARY KEY AUTO_INCREMENT,
59     match_id INT NOT NULL,
60     student_id INT NOT NULL,
61     proof_text TEXT NOT NULL,
62     proof_file LONGBLOB NOT NULL,
63     approval_status VARCHAR(20) DEFAULT 'Pending',
64     verified_by_staff_id INT,
65     FOREIGN KEY (match_id) REFERENCES match_items(match_id) ON DELETE CASCADE,
66     FOREIGN KEY (student_id) REFERENCES student(student_id) ON DELETE CASCADE,
67     FOREIGN KEY (verified_by_staff_id) REFERENCES staff(staff_id) ON DELETE SET NULL
68 );
69 • Ⓜ CREATE TABLE notifications (
70     notification_id INT PRIMARY KEY AUTO_INCREMENT,
71     message TEXT NOT NULL,
72     date DATE NOT NULL,
73     status VARCHAR(20) DEFAULT 'Sent'
74 );
75

```

Outputs:-

Output			
#	Time	Action	Message
1	14:21:47	DROP DATABASE IF EXISTS LostAndFoundDB	0 row(s) affected, 1 warning(s): 1008 Can't drop database `lostandfounddb'; ...
2	14:21:47	CREATE DATABASE LostAndFoundDB	1 row(s) affected
3	14:21:47	USE LostAndFoundDB	0 row(s) affected
4	14:21:47	CREATE TABLE student (student_id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(50), age INT, gender CHAR(1), address TEXT)	0 row(s) affected
5	14:21:47	CREATE TABLE staff (staff_id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(50), age INT, gender CHAR(1), address TEXT)	0 row(s) affected
6	14:21:47	CREATE TABLE lost_items (lost_item_id INT PRIMARY KEY AUTO_INCREMENT, item_name VARCHAR(50), category VARCHAR(50), lost_date DATE, lost_loc VARCHAR(100), status CHAR(10))	0 row(s) affected
7	14:21:47	CREATE TABLE found_items (f_i_id INT PRIMARY KEY AUTO_INCREMENT, item_name VARCHAR(50), category VARCHAR(50), found_date DATE, found_loc VARCHAR(100), status CHAR(10))	0 row(s) affected
8	14:21:47	CREATE TABLE match_items (match_id INT PRIMARY KEY AUTO_INCREMENT, item_name VARCHAR(50), category VARCHAR(50), match_date DATE, match_loc VARCHAR(100), status CHAR(10))	0 row(s) affected
9	14:21:47	CREATE TABLE claims (claim_id INT PRIMARY KEY AUTO_INCREMENT, item_name VARCHAR(50), category VARCHAR(50), lost_date DATE, lost_loc VARCHAR(100), status CHAR(10))	0 row(s) affected
10	14:21:48	CREATE TABLE notifications (notification_id INT PRIMARY KEY AUTO_INCREMENT, message TEXT)	0 row(s) affected

Total no. of Tables

Tables_in_lostandfounddb	
▶	claims
	found_items
	lost_items
	match_items
	notifications
	staff
	student

CRUD operation Screenshots:

- Create operation on lost_items:

```
• INSERT INTO lost_items (
    student_id,
    category,
    lost_date,
    lost_loc,
    status,
    item_name,
    photo,
    description
) VALUES (
    3,
    'Electronics',
    '2025-01-12',
    'College Ground',
    'Unresolved',
    'Smartwatch',
    NULL,
    'Black strap smartwatch lost during sports event'
);
```

Output:-

- Read operation on lost_items table:

76 • `select * from lost_items;`

Output:-

	lost_item_id	student_id	category	lost_date	lost_loc	status	item_name	photo	description
▶	1	1	ID Cards	2025-11-14	inform of BE block	Unresolved	College ID card	HULL	It has my name Satwik Hegde and my SRN
●	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

✓	38	15:51:41	UPDATE lost_items SET category = 'Electronics', lost_loc = 'Library' WHERE lost_item_id = 1
---	----	----------	---

- Update operation on lost_items table

```
78 •     UPDATE lost_items  
79         SET  
80             category = 'Electronics',  
81             lost_loc = 'Library'  
82     WHERE  
83         lost_item_id = 1;
```

Output:-

	lost_item_id	student_id	category	lost_date	lost_loc	status	item_name	photo	description
▶	1	1	Electronics	2025-11-14	Library	Unresolved	College ID card	NULL	It has my name Satwik Hegde and my SRN
●	HULL	HULL	HULL	HULL	NULL	NULL	NULL	NULL	NULL

- Delete operation on lost_items table

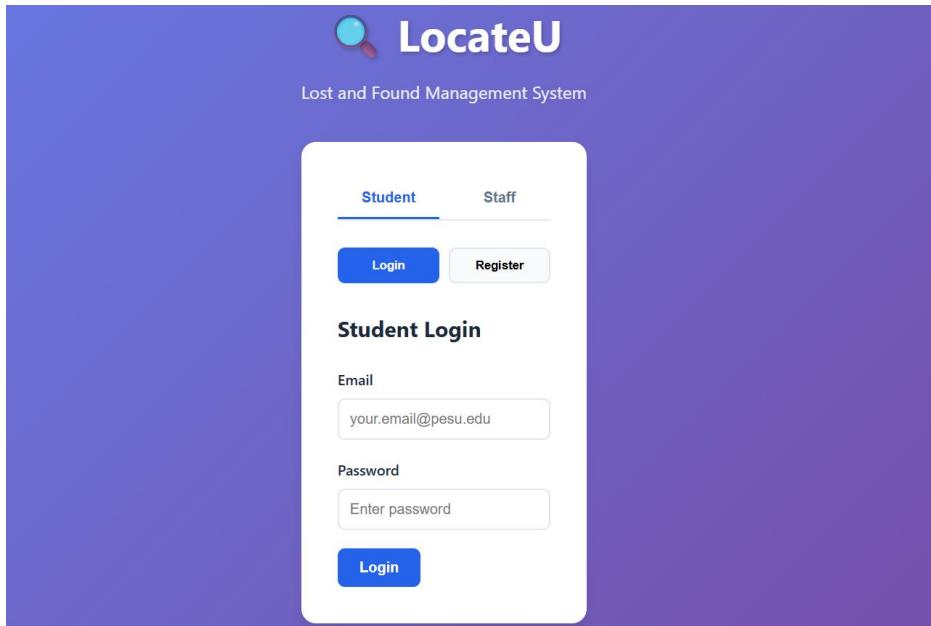
```
DELETE FROM lost_items  
WHERE lost_item_id = 3;
```

Output:-

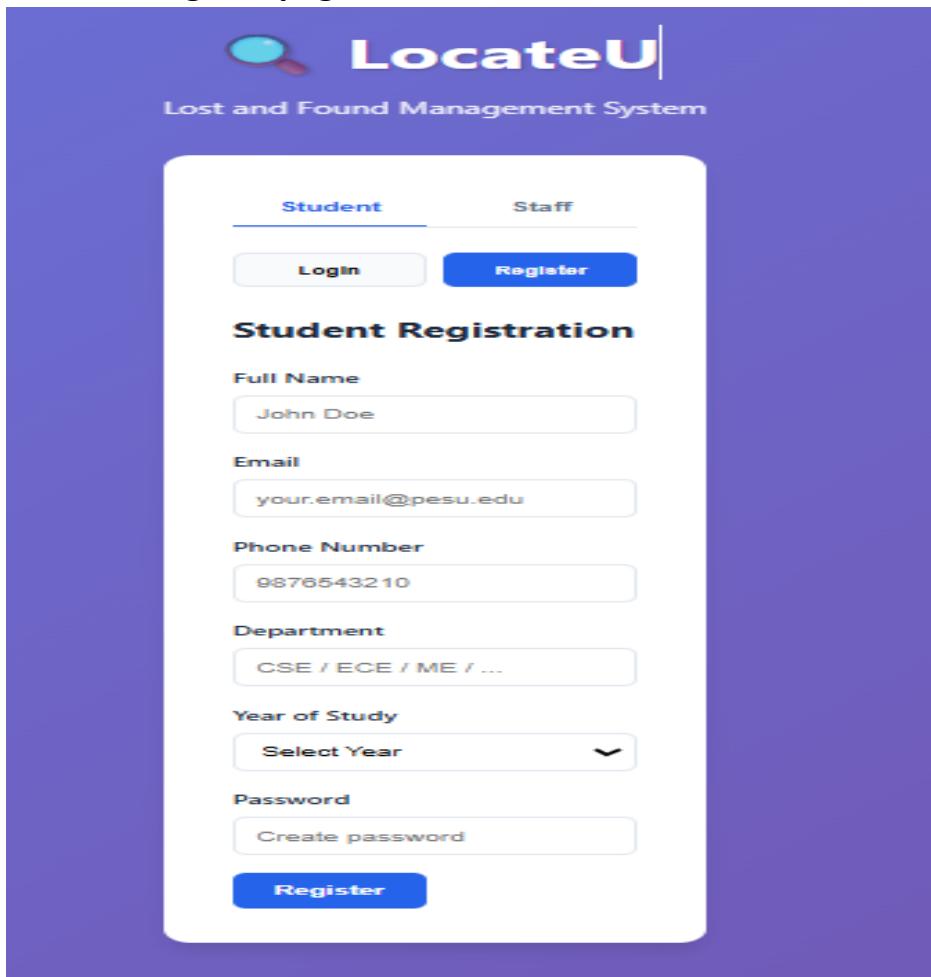
43 16:02:38 DELETE FROM lost_items WHERE lost_item_id = 3

List of functionalities/features of the application :

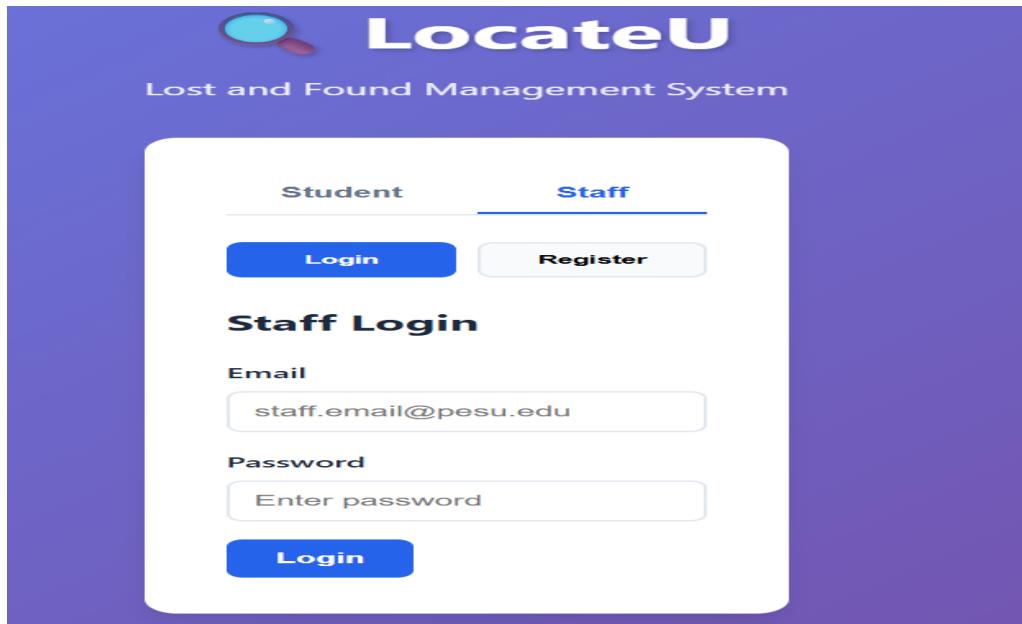
- Student login page



- Student Register page



- Staff registration and login page



- Staff registration page

The image shows the staff registration page of the LocateU system. At the top, there is a logo of a magnifying glass and the word "LocateU". Below it, the text "Lost and Found Management System" is displayed. A navigation bar at the top right includes "Student" and "Staff" tabs, with "Staff" being the active tab. Below the navigation bar are two buttons: "Login" (in white) and "Register" (in blue). The main section is titled "Staff Registration" and contains several input fields:

- Full Name: Dr. John Doe
- Email: staff.email@pesu.edu
- Phone Number: 9876543210
- Role: e.g., Admin, Security
- Department: e.g., CSE, Admin
- Password: Create password

A "Register" button is located at the bottom of this section.

○ Student Dashboard

The dashboard features a sidebar menu with options: Dashboard (selected), Report Lost Item, Report Found Item, and Claim Item. The main area includes a 'Recent Notifications' section with two items: 'Lost item reported: College ID card' (date 2025-11-15) and 'Lost item reported: SmartWatch' (date 2025-11-15). A 'My Lost Items' section shows a single item: 'College ID card' (Category: Electronics, Lost Date: 2025-11-14, Location: Library, status: Unresolved). Quick actions include 'Report Lost Item' (red circle), 'Report Found Item' (green circle), and 'Claim an Item' (yellow hand icon).

○ Report Lost Item form

The form is titled 'Report Lost Item' and prompts the user to 'Fill in the details of the item you lost'. It includes fields for 'Item Name *' (e.g., Blue Water Bottle), 'Category *' (Select Category dropdown), 'Lost Date *' (15-11-2025), 'Lost Location *' (e.g., MRD Auditorium), and a 'Description *' text area (Provide detailed description including color, brand, distinguishing features...). An optional 'Upload Photo' field with a 'Choose Photo' button is also present. The form has 'Submit Report' and 'Cancel' buttons.

○ Report Found Item form

The screenshot shows the 'Report Found Item' form. At the top right, there are 'Welcome, Satwik' and 'Logout' buttons. On the left, a sidebar menu includes 'Dashboard', 'Report Lost Item', 'Report Found Item' (which is highlighted in blue), and 'Claim Item'. The main form area has a title 'Report Found Item' with a subtitle 'Help someone find their lost item'. It contains fields for 'Item Name *' (e.g., Black Wallet), 'Category *' (Select Category dropdown), 'Found Date *' (15-11-2025), 'Found Location *' (e.g., Library 2nd Floor), 'Description *' (Provide detailed description to help owner identify...), and an 'Upload Photo (Optional)' button ('Choose Photo'). At the bottom are 'Submit Report' and 'Cancel' buttons.

○ Claim Item page

The screenshot shows the 'Claim an Item' page. At the top right, there are 'Welcome, Satwik' and 'Logout' buttons. On the left, a sidebar menu includes 'Dashboard', 'Report Lost Item', 'Report Found Item', and 'Claim Item' (which is highlighted in blue). The main area has a title 'Claim an Item' with a subtitle 'Browse found items and claim yours'. It features a placeholder image of a brown box. Below it, a card displays 'College ID card' information: Category: ID Cards, Found Date: 2025-11-14, Location: BE block, Reported by: Satwik. It also shows 'Name written as Satwik Hegde' and a 'Claim This Item' button.

○ Staff Dashboard

The screenshot shows the 'Staff Dashboard'. At the top right, there are 'Welcome, Rajat' and 'Logout' buttons. On the left, a sidebar menu includes 'Dashboard' (highlighted in blue), 'Review Claims', and 'Match Items'. The main area has a title 'Staff Dashboard' with a subtitle 'Manage lost and found items'. It features four cards: 'Pending Claims' (0), 'Unresolved Lost Items' (1), 'Unclaimed Found Items' (1), and 'Pending Matches' (0). Below these is a section titled 'Recent System Notifications' with three entries: 'Lost item reported: College ID card 2025-11-15', 'Lost item reported: SmartWatch 2025-11-15', and 'New found item reported by Satwik: College ID card at BE block on 2025-11-14 2025-11-15'.

○ Staff Claims Management page

The screenshot shows the 'Review Claims' section of the LocateU - Staff Portal. A claim for 'Claim #2' is displayed, which is a lost item. The student information shows Name: Satwik and Email: satwikhedgeley11@gmail.com. The lost item details show Item: College ID card and Category: Electronics. Below this, found item details show Item: College ID card and Location: BE block. A photograph of the ID card is shown, along with a note: 'My id card has my name and sm'. At the bottom, there are 'Approve' and 'Reject' buttons.

○ Staff Match Items page with suggestions

The screenshot shows the 'Match Lost & Found Items' section of the LocateU - Staff Portal. It displays two items for matching: 'Lost Items (Unresolved)' and 'Found Items (Unclaimed)'. In the 'Lost Items (Unresolved)' section, there is a 'cricket bat' listed. In the 'Found Items (Unclaimed)' section, there is a 'Blue water bottle'. A modal window titled 'Match Lost Item' is open, showing the 'Lost: cricket bat (ID 6)' and the 'Select a Found item:' dropdown. The dropdown contains 'choose found item --'. Below the dropdown are 'Confirm Match' and 'Cancel' buttons. The background shows the main interface with the same two items listed under their respective sections.

○ Notifications panel

Recent System Notifications	
Lost item reported: College ID card	2025-11-15
Lost item reported: SmartWatch	2025-11-15
New found item reported by Satwik: College ID card at B6 block on 2025-11-14	2025-11-15
New claim submitted by Satwik for found item 'College ID card' (found_id=1)	2025-11-15

Code snippets for invoking the Procedures/Functions/Trigger :

■ Procedures

- `DROP PROCEDURE IF EXISTS RegisterLostItem;`
DELIMITER //
- `CREATE PROCEDURE RegisterLostItem(IN p_student_id INT, IN p_category VARCHAR(50), IN p_item_name VARCHAR(100), IN p_description TEXT, IN p_lost_date DATE, IN p_lost_loc VARCHAR(100))`
BEGIN
 `INSERT INTO lost_items(student_id, category, item_name, description, lost_date, lost_loc, status) VALUES (p_student_id, p_category, p_item_name, p_description, p_lost_date, p_lost_loc, 'Unresolved');`
END;
//
DELIMITER ;
- `DROP PROCEDURE IF EXISTS MatchLostFound;`
DELIMITER //
- `CREATE PROCEDURE MatchLostFound(IN p_lost_id INT, IN p_found_id INT)`
BEGIN
 `INSERT INTO match_items(lost_item_id, f_i_id, match_date, status) VALUES (p_lost_id, p_found_id, CURDATE(), 'Pending');`
 `UPDATE lost_items SET status='Matched' WHERE lost_item_id=p_lost_id;`
 `UPDATE found_items SET status='Matched' WHERE f_i_id=p_found_id;`
END;
//

■ Functions

- `DROP FUNCTION IF EXISTS CountLostItems;`
DELIMITER //
- `CREATE FUNCTION CountLostItems(p_student_id INT)`
RETURNS INT DETERMINISTIC
BEGIN
 `DECLARE lost_count INT;`
 `SELECT COUNT(*) INTO lost_count FROM lost_items WHERE student_id = p_student_id;`
 `RETURN lost_count;`
END;
//
DELIMITER ;
- `DROP FUNCTION IF EXISTS GetClaimStatus;`
DELIMITER //
- `CREATE FUNCTION GetClaimStatus(p_claim_id INT)`
RETURNS VARCHAR(20) DETERMINISTIC
BEGIN
 `DECLARE status_val VARCHAR(20);`
 `SELECT approval_status INTO status_val FROM claims WHERE claim_id = p_claim_id;`
 `RETURN status_val;`
END;
//

■ Trigger

```

//  

DELIMITER ;  

DROP TRIGGER IF EXISTS after_lost_item_insert;  

DELIMITER //  

CREATE TRIGGER after_lost_item_insert AFTER INSERT ON lost_items FOR EACH ROW  

BEGIN INSERT INTO notifications (message, date, status) VALUES (CONCAT('Lost item  

reported: ', NEW.item_name), CURDATE(), 'Unread'); END;  

//  

DELIMITER ;  

DROP TRIGGER IF EXISTS after_claim_update_notify;  

DELIMITER //  

CREATE TRIGGER after_claim_update_notify AFTER UPDATE ON claims FOR EACH  

ROW BEGIN IF NEW.verified_by_staff_id IS NOT NULL THEN INSERT INTO notifications  

(message, date, status) VALUES (CONCAT('Claim ', NEW.claim_id, ',  

NEW.approval_status), CURDATE(), 'Unread'); END IF; END;  

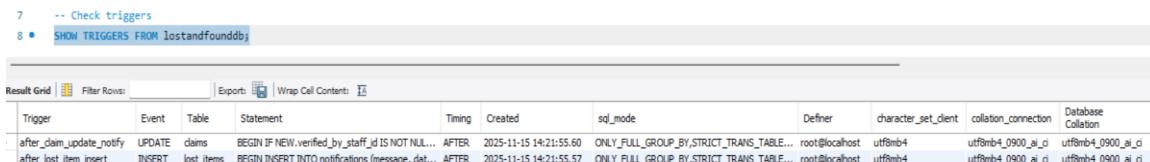
//  

DELIMITER ;

```

Triggers, Procedures/Functions, Nested query, Join, Aggregate queries:

☞ Triggers :- here trigger update the notification after lost item inserting



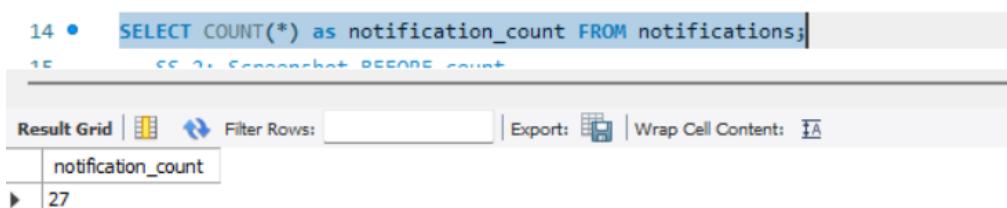
The screenshot shows the MySQL Workbench interface with the following details:

- Session: 7 - ... Check triggers
- Session: 8 - SHOW TRIGGERS FROM lostandfounddb;

Result Grid:

Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client	collation_connection	Database	Collation
after_claim_update_notify	UPDATE	claims	BEGIN IF NEW.verified_by_staff_id IS NOT NUL...	AFTER	2025-11-15 14:21:55.60	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci	
after_lost_item_insert	INSERT	lost_items	BEGIN INSERT INTO notifications (message, dat...	AFTER	2025-11-15 14:21:55.57	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci	

Before inserting lost item the count of notification



The screenshot shows the MySQL Workbench interface with the following details:

- Session: 14 - SELECT COUNT(*) as notification_count FROM notifications;
- Session: 15 - COUNT(*) as notification_count

Result Grid:

notification_count
27

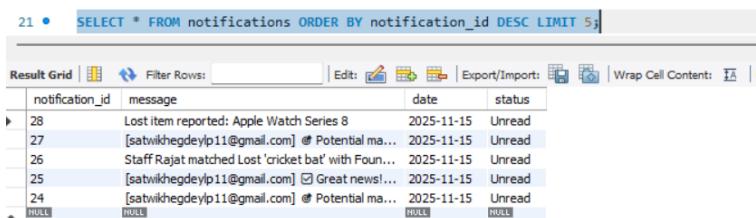
after inserting lost item:

```

• INSERT INTO lost_items (student_id, category, lost_date, lost_loc, status, item_name, description)
VALUES (1, 'Electronics', '2025-11-15', 'Library 3rd Floor', 'Unresolved', 'Apple Watch Series 8', 'Silver watch');

```

The notification goes to 27 to 28 after inserting



The screenshot shows the MySQL Workbench interface with the following details:

- Session: 21 - SELECT * FROM notifications ORDER BY notification_id DESC LIMIT 5;

Result Grid:

notification_id	message	date	status
28	Lost item reported: Apple Watch Series 8	2025-11-15	Unread
27	[satwikhegdeylp11@gmail.com] ⚡ Potential ma...	2025-11-15	Unread
26	Staff Rajat matched Lost 'cricket bat' with Foun...	2025-11-15	Unread
25	[satwikhegdeylp11@gmail.com] 🎉 Great news!...	2025-11-15	Unread
24	[satwikhegdeylp11@gmail.com] ⚡ Potential ma...	2025-11-15	Unread
*	NULL	NULL	NULL

↗ Procedures :-

There are two procedures, one is MatchLostFound and another is RegisterLostItem

```
6    -- Check procedures
7 • SHOW PROCEDURE STATUS WHERE Db='lostandfounddb';
8 |
9
```

Result Grid Filter Rows: Export: Wrap Cell Content:										
Db	Name	Type	Definer	Modified	Created	Security_type	Comment	character_set_client	collation_connection	Database Collation
lostandfounddb	MatchLostFound	PROCEDURE	root@localhost	2025-11-15 14:21:55	2025-11-15 14:21:55	DEFINER		utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
lostandfounddb	RegisterLostItem	PROCEDURE	root@localhost	2025-11-15 14:21:55	2025-11-15 14:21:55	DEFINER		utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

For example I show RegisterLostItem working.

Before calling the RegisterLostItem procedure, the number of lost item

```
38 •   SELECT COUNT(*) AS after_count FROM lost_items;
39 |
40
```

Result Grid Filter Rows: Export: Wrap Cell Content:	
	after_count
▶	6

Call the RegisterLostItem procedure

```
29 • CALL RegisterLostItem(
30     1,
31     'Electronics',
32     'iPhone 17 pro max',
33     'Orange Colour with three camera and profile photo has virat kohli',
34     '2025-11-10',
35     'Canteen'
36 );
```

```
67 17:33:20 CALL RegisterLostItem( 1, 'Electronics', 'iPhone 17 pro max', 'Orange Colour with three camera and profile photo has virat k... 1 row(s) affected 0.016 sec
```

After calling the procedure it triggers the number of lost item increased by 1

```
38 •   SELECT COUNT(*) AS after_count FROM lost_items;
39
```

Result Grid Filter Rows: Export: Wrap Cell Content:	
	after_count
▶	7

↗ Functions:-

There is two function is there

The screenshot shows the MySQL Workbench interface with the SQL editor containing the command:

```
8 • -- Check functions
9 • SHOW FUNCTION STATUS WHERE Db='lostandfounddb';
```

The results are displayed in a grid:

Db	Name	Type	Definer	Modified	Created	Security_type	Comment	character_set_client	collation_connection	Database	Collation
lostandfounddb	CountLostItems	FUNCTION	root@localhost	2025-11-15 14:21:55	2025-11-15 14:21:55	DEFINER		utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
lostandfounddb	GetClaimStatus	FUNCTION	root@localhost	2025-11-15 14:21:55	2025-11-15 14:21:55	DEFINER		utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

One is counting the lost item which gives total number of lost item

```
23 • SELECT CountLostItems(1) AS total_lost_items;
```

24

The screenshot shows the MySQL Workbench interface with the SQL editor containing the command:

```
23 • SELECT CountLostItems(1) AS total_lost_items;
```

The results are displayed in a grid:

total_lost_items
5

You can see in the website too.

The screenshot shows a web-based dashboard titled "Dashboard". On the left, there is a sidebar with a "Menu" section containing links for "Dashboard", "Report Lost Item", "Report Found Item", and "Claim Item". The main area is titled "Dashboard" and displays the message "Welcome to your Lost & Found dashboard". Below this, there is a box titled "My Lost Items" with an icon of a clipboard and the number "5".

Second is the to know the claim status

```
25 • SELECT GetClaimStatus(1) AS claim_status;
```

26

The screenshot shows the MySQL Workbench interface with the SQL editor containing the command:

```
25 • SELECT GetClaimStatus(1) AS claim_status;
```

The results are displayed in a grid:

claim_status
NULL

↗ Nested query

This query help to find Find students who have reported more than 1 lost item

```
1 • SELECT s.student_id, s.name
2   FROM student s
3 WHERE (SELECT COUNT(*) FROM lost_items li WHERE li.student_id = s.student_id) > 1;
4
```

The screenshot shows the MySQL Workbench interface with the SQL editor containing the command:

```
1 • SELECT s.student_id, s.name
2   FROM student s
3 WHERE (SELECT COUNT(*) FROM lost_items li WHERE li.student_id = s.student_id) > 1;
4
```

The results are displayed in a grid:

student_id	name
1	Satwik

This nested query shows the found items that were reported by students who are in the CSE department.

```
5 •   SELECT f.f_i_id, f.item_name, f.found_loc  
6     FROM found_items f  
7     WHERE f.report_student_id IN (  
8       SELECT s.student_id FROM student s WHERE s.department = 'CSE'  
9     );  
10
```

Result Grid		
f_i_id	item_name	found_loc
1	College ID card	BE block
2	College ID card	BE block
3	Blue water bottle	BE block
4	cricket bat	college ground
NULL	NULL	NULL

Join queries

This is the inner join help to display claims with student name, lost item, found item, and claim status

```
12    -- Join  
13    -- Inner join  
14 •   SELECT c.claim_id, s.name AS student_name,  
15          li.item_name AS lost_item, fi.item_name AS found_item,  
16          c.approval_status  
17    FROM claims c  
18    JOIN student s ON c.student_id = s.student_id  
19    JOIN match_items m ON c.match_id = m.match_id  
20    JOIN lost_items li ON m.lost_item_id = li.lost_item_id  
21    JOIN found_items fi ON m.f_i_id = fi.f_i_id  
22    ORDER BY c.claim_id DESC  
23    LIMIT 10;
```

Result Grid				
claim_id	student_name	lost_item	found_item	approval_status
3	Satwik	SmartWatch	Blue water bottle	Pending
2	Satwik	College ID card	College ID card	Approved

This is the outer join which helps to list the found items with match info if it exists (shows unclaimed/unmatched too).

```
25    -- Outer join
26 •  SELECT f.f_i_id, f.item_name, f.found_loc, m.match_id, m.status AS match_status
27  FROM found_items f
28  LEFT JOIN match_items m ON f.f_i_id = m.f_i_id
29  ORDER BY f.found_date DESC
30  LIMIT 20;
31
```

Result Grid				
f_i_id	item_name	found_loc	match_id	match_status
2	College ID card	BE blook	2	Approved
3	Blue water bottle	BE blook	3	Pending
4	cricket bat	college ground	4	Pending
4	cricket bat	college ground	5	Pending
1	College ID card	BE blook	1	Rejected
1	College ID card	BE blook	6	Pending

Aggregate queries:-

This query helps to count lost items per category

```
32    -- Aggregate query
33 •  SELECT category, COUNT(*) AS total_lost
34  FROM lost_items
35  GROUP BY category
36  ORDER BY total_lost DESC;
37
```

Result Grid	
category	total_lost
Electronics	5
Other	2
Accessories	1
Clothing	1

This query helps to count found items per day (time-based aggregation)

```
38 •  SELECT found_date, COUNT(*) AS found_count
39  FROM found_items
40  GROUP BY found_date
41  ORDER BY found_date DESC
42  LIMIT 30;
```

Result Grid	
found_date	found_count
2025-11-15	3
2025-11-14	1

SQL queries(Create, Insert, Triggers, Procedures/Functions, Nested query, Join, Aggregate queries) used in the project in the form of .sql file :

```
DROP DATABASE IF EXISTS LostUDB;
```

```
CREATE DATABASE LostUDB;
```

```
USE LostUDB;
```

```
-- CREATE TABLES
```

```
CREATE TABLE student (
    student_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone_number VARCHAR(15) UNIQUE NOT NULL,
    department VARCHAR(50) NOT NULL,
    year_of_study INT CHECK (year_of_study BETWEEN 1 AND 5),
    password VARCHAR(255) NOT NULL
);
```

```
CREATE TABLE staff (
    staff_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone_number VARCHAR(15) UNIQUE NOT NULL,
    role VARCHAR(50) NOT NULL,
    department VARCHAR(50) NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

```
CREATE TABLE lost_items (
    lost_item_id INT PRIMARY KEY AUTO_INCREMENT,
```

```
student_id INT,  
category VARCHAR(50) NOT NULL,  
lost_date DATE NOT NULL,  
lost_loc VARCHAR(100) NOT NULL,  
status VARCHAR(20) DEFAULT 'Unresolved',  
item_name VARCHAR(100) NOT NULL,  
photo LONGBLOB NULL,  
description TEXT,  
FOREIGN KEY (student_id) REFERENCES student(student_id) ON DELETE CASCADE  
);
```

```
CREATE TABLE found_items (  
f_i_id INT PRIMARY KEY AUTO_INCREMENT,  
report_student_id INT NULL,  
report_staff_id INT NULL,  
item_name VARCHAR(100) NOT NULL,  
description TEXT,  
category VARCHAR(50) NOT NULL,  
found_date DATE NOT NULL,  
found_loc VARCHAR(100) NOT NULL,  
status VARCHAR(20) DEFAULT 'Unclaimed',  
photo LONGBLOB NULL,  
FOREIGN KEY (report_student_id) REFERENCES student(student_id) ON DELETE SET NULL,  
FOREIGN KEY (report_staff_id) REFERENCES staff(staff_id) ON DELETE SET NULL  
);
```

```
CREATE TABLE match_items (  
match_id INT PRIMARY KEY AUTO_INCREMENT,  
lost_item_id INT NOT NULL,
```

```
f_i_id INT NOT NULL,  
match_date DATE NOT NULL,  
status VARCHAR(20) DEFAULT 'Pending',  
FOREIGN KEY (lost_item_id) REFERENCES lost_items(lost_item_id) ON DELETE CASCADE,  
FOREIGN KEY (f_i_id) REFERENCES found_items(f_i_id) ON DELETE CASCADE  
);
```

```
CREATE TABLE claims (  
claim_id INT PRIMARY KEY AUTO_INCREMENT,  
match_id INT NOT NULL,  
student_id INT NOT NULL,  
proof_text TEXT NOT NULL,  
proof_file LONGBLOB NOT NULL,  
approval_status VARCHAR(20) DEFAULT 'Pending',  
verified_by_staff_id INT,  
FOREIGN KEY (match_id) REFERENCES match_items(match_id) ON DELETE CASCADE,  
FOREIGN KEY (student_id) REFERENCES student(student_id) ON DELETE CASCADE,  
FOREIGN KEY (verified_by_staff_id) REFERENCES staff(staff_id) ON DELETE SET NULL  
);
```

```
CREATE TABLE notifications (  
notification_id INT PRIMARY KEY AUTO_INCREMENT,  
message TEXT NOT NULL,  
date DATE NOT NULL,  
status VARCHAR(20) DEFAULT 'Sent'  
);
```

```
-- REQUIRED INSERTS
```

```
INSERT INTO student (name, email, phone_number, department, year_of_study, password)
VALUES
('Rajat Bhat', 'rajat@example.com', '9999999991', 'CSE', 2, 'pwdhash1'),
('Alice Kumar', 'alice@example.com', '9999999992', 'ECE', 3, 'pwdhash2'),
('Bob Sharma', 'bob@example.com', '9999999993', 'CSE', 2, 'pwdhash3');
```

```
INSERT INTO staff (name, email, phone_number, role, department, password)
```

```
VALUES
```

```
('S. Patel', 'patel@example.com', '8888888801', 'Security', 'Admin', 'pwdhashs1'),
('M. Singh', 'msingh@example.com', '8888888802', 'Helper', 'Admin', 'pwdhashs2');
```

```
INSERT INTO lost_items (student_id, category, lost_date, lost_loc, status, item_name,
description)
```

```
VALUES
```

```
(1, 'Electronics', '2025-11-10', 'Library 1st Floor', 'Unresolved', 'Apple Watch Series 8', 'Silver
watch'),
(2, 'Stationery', '2025-11-12', 'Block A Corridor', 'Unresolved', 'Calculator', 'Casio fx-991'),
(1, 'Electronics', '2025-11-11', 'Cafeteria', 'Unresolved', 'AirPods Pro', 'White case');
```

```
INSERT INTO found_items (report_student_id, report_staff_id, item_name, description,
category, found_date, found_loc, status)
```

```
VALUES
```

```
(NULL, 1, 'Smartwatch', 'Found near library desk', 'Electronics', '2025-11-10', 'Library 1st
Floor', 'Unclaimed'),
(3, NULL, 'Calculator', 'Found near Block A', 'Stationery', '2025-11-12', 'Block A Corridor',
'Unclaimed'),
(NULL, 2, 'Earbuds', 'White earbuds in case', 'Electronics', '2025-11-11', 'Cafeteria',
'Unclaimed');
```

```
-- FUNCTIONS
```

```
USE LostAndFoundDB;

DROP FUNCTION IF EXISTS CountLostItems;
DELIMITER //

CREATE FUNCTION CountLostItems(p_student_id INT)
RETURNS INT DETERMINISTIC
BEGIN
    DECLARE lost_count INT;
    SELECT COUNT(*) INTO lost_count FROM lost_items WHERE student_id = p_student_id;
    RETURN lost_count;
END;
//

DELIMITER ;

DROP FUNCTION IF EXISTS GetClaimStatus;
DELIMITER //
CREATE FUNCTION GetClaimStatus(p_claim_id INT)
RETURNS VARCHAR(20) DETERMINISTIC
BEGIN
    DECLARE status_val VARCHAR(20);
    SELECT approval_status INTO status_val FROM claims WHERE claim_id = p_claim_id;
    RETURN status_val;
END;
//

DELIMITER ;

-- PROCEDURES

DROP PROCEDURE IF EXISTS RegisterLostItem;
```

```

DELIMITER //

CREATE PROCEDURE RegisterLostItem(
    IN p_student_id INT,
    IN p_category VARCHAR(50),
    IN p_item_name VARCHAR(100),
    IN p_description TEXT,
    IN p_lost_date DATE,
    IN p_lost_loc VARCHAR(100))
BEGIN
    INSERT INTO lost_items(student_id, category, item_name, description, lost_date, lost_loc, status)
    VALUES (p_student_id, p_category, p_item_name, p_description, p_lost_date, p_lost_loc, 'Unresolved');

END;
//

DELIMITER ;

DROP PROCEDURE IF EXISTS MatchLostFound;

DELIMITER //

CREATE PROCEDURE MatchLostFound(IN p_lost_id INT, IN p_found_id INT)
BEGIN
    INSERT INTO match_items(lost_item_id, f_i_id, match_date, status)
    VALUES (p_lost_id, p_found_id, CURDATE(), 'Pending');

    UPDATE lost_items SET status='Matched' WHERE lost_item_id=p_lost_id;
    UPDATE found_items SET status='Matched' WHERE f_i_id=p_found_id;
END;
//

DELIMITER ;

```

```

-- TRIGGERS

DROP TRIGGER IF EXISTS after_lost_item_insert;
DELIMITER //

CREATE TRIGGER after_lost_item_insert
AFTER INSERT ON lost_items
FOR EACH ROW
BEGIN
    INSERT INTO notifications (message, date, status)
    VALUES (CONCAT('Lost item reported: ', NEW.item_name), CURDATE(), 'Unread');
END;
// 
DELIMITER ;

DROP TRIGGER IF EXISTS after_claim_update_notify;
DELIMITER //
CREATE TRIGGER after_claim_update_notify
AFTER UPDATE ON claims
FOR EACH ROW
BEGIN
    IF NEW.verified_by_staff_id IS NOT NULL THEN
        INSERT INTO notifications (message, date, status)
        VALUES (CONCAT('Claim ', NEW.claim_id, ' ', NEW.approval_status), CURDATE(),
        'Unread');
    END IF;
END;
// 
DELIMITER ;

```

```
-- NESTED QUERIES
```

```
SELECT s.student_id, s.name  
FROM student s  
WHERE (SELECT COUNT(*) FROM lost_items li WHERE li.student_id = s.student_id) > 1;
```

```
SELECT f.f_i_id, f.item_name, f.found_loc  
FROM found_items f  
WHERE f.report_student_id IN (  
    SELECT s.student_id FROM student s WHERE s.department = 'CSE'  
);
```

```
-- JOIN QUERIES
```

```
-- Inner join
```

```
SELECT c.claim_id, s.name AS student_name,  
    li.item_name AS lost_item, fi.item_name AS found_item,  
    c.approval_status  
FROM claims c  
JOIN student s ON c.student_id = s.student_id  
JOIN match_items m ON c.match_id = m.match_id  
JOIN lost_items li ON m.lost_item_id = li.lost_item_id  
JOIN found_items fi ON m.f_i_id = fi.f_i_id  
ORDER BY c.claim_id DESC  
LIMIT 10;
```

```
-- Outer join
```

```
SELECT f.f_i_id, f.item_name, f.found_loc, m.match_id, m.status AS match_status  
FROM found_items f  
LEFT JOIN match_items m ON f.f_i_id = m.f_i_id  
ORDER BY f.found_date DESC
```

```
LIMIT 20;
```

```
-- AGGREGATE QUERIES
```

```
SELECT category, COUNT(*) AS total_lost  
FROM lost_items  
GROUP BY category  
ORDER BY total_lost DESC;
```

```
SELECT found_date, COUNT(*) AS found_count  
FROM found_items  
GROUP BY found_date  
ORDER BY found_date DESC  
LIMIT 30;
```

Github repo link :

<https://github.com/Pratheek22/LocateU-Lost-And-Found-System.git>