

**Case Study
On
LOST AND FOUND MANAGEMENT SYSTEM**

INTRODUCTION

Lost and Found Management System is a web-based application developed to help users report, manage, and recover lost items efficiently. In colleges and public places, many valuable items are lost every day, and manual handling of such cases is difficult. This system provides a digital solution where users can log in, view reported items, add new lost or found items, and search items easily. The application is developed using **Spring Boot** and **Postman** with **MySQL** database support. In the present digital era, managing lost and found items manually has become inefficient due to increased population and frequent movement of people. Educational institutions and public places require a reliable system to track lost belongings. This project addresses this issue by providing a structured and centralized system that helps users easily report and recover items with minimal effort.

ABSTRAT

The **Lost and Found Management System** is designed to automate the process of managing lost and found items. The system allows users to register and log in securely, report lost or found items, and search for items based on name or location. The backend is developed using **Spring Boot with Hibernate and JPA for database interaction, MySQL as the database, and Postman for API testing**. This project improves efficiency, reduces manual effort, and ensures secure data management. The application ensures smooth communication between frontend and backend using RESTful APIs. Hibernate ORM simplifies database operations by reducing SQL dependency. The system also supports search functionality, enabling users to quickly locate items based on name or description. Overall, the project demonstrates practical implementation of Spring Boot concepts.

What we are going to build: Client Requirement

The client also expects the system to be scalable and easy to maintain. The application should support multiple users simultaneously without data inconsistency. The system must provide clear and simple user interfaces and reliable backend services to ensure smooth user experience.

- Users can register and log in securely
- Users can report lost or found items
- Items should include **name, description, image, and Status**
- Users can search items using keywords
- Data should be stored permanently in a database
- Admin/User data should be managed securely

What we are going to build: Some technical terms.

Spring Boot framework helps in rapid application development by providing auto-configuration and embedded server support. MySQL ensures reliable data storage with high performance. Postman is used to validate APIs during development, ensuring correct request-response behavior.

- ❖ **Spring Boot:** Framework for backend development
- ❖ **REST API:** Communication between frontend and backend
- ❖ **Hibernate:** ORM tool for database interaction
- ❖ **JPA:** Java Persistence API
- ❖ **CRUD Operations:** Create, Read, Update, Delete
- ❖ **JSON:** Data format used in API requests and responses

What are the technologies and tools we are going use ?

- ✓ **Backend:** Spring Boot
- ✓ **Language:** Java (JDK 17)
- ✓ **Database:** MySQL
- ✓ **ORM Tool:** Hibernate & JPA
- ✓ **Frontend:** HTML, CSS
- ✓ **IDE:** Spring Tool Suite (STS)
- ✓ **API Testing Tool:** Postman
- ✓ **Server:** Embedded Apache Tomcat

SYSTEM REQUIREMENTS

Software Requirements:

- Windows OS
- Java JDK 17
- Spring Tool Suite (STS)
- MySQL Server
- Postman

Hardware Requirements:

- Minimum 4GB RAM
- Processor: Intel i3 or above

PROJECT MODULE

User Authentication Module

- ✓ User Registration
- ✓ User Login & Logout

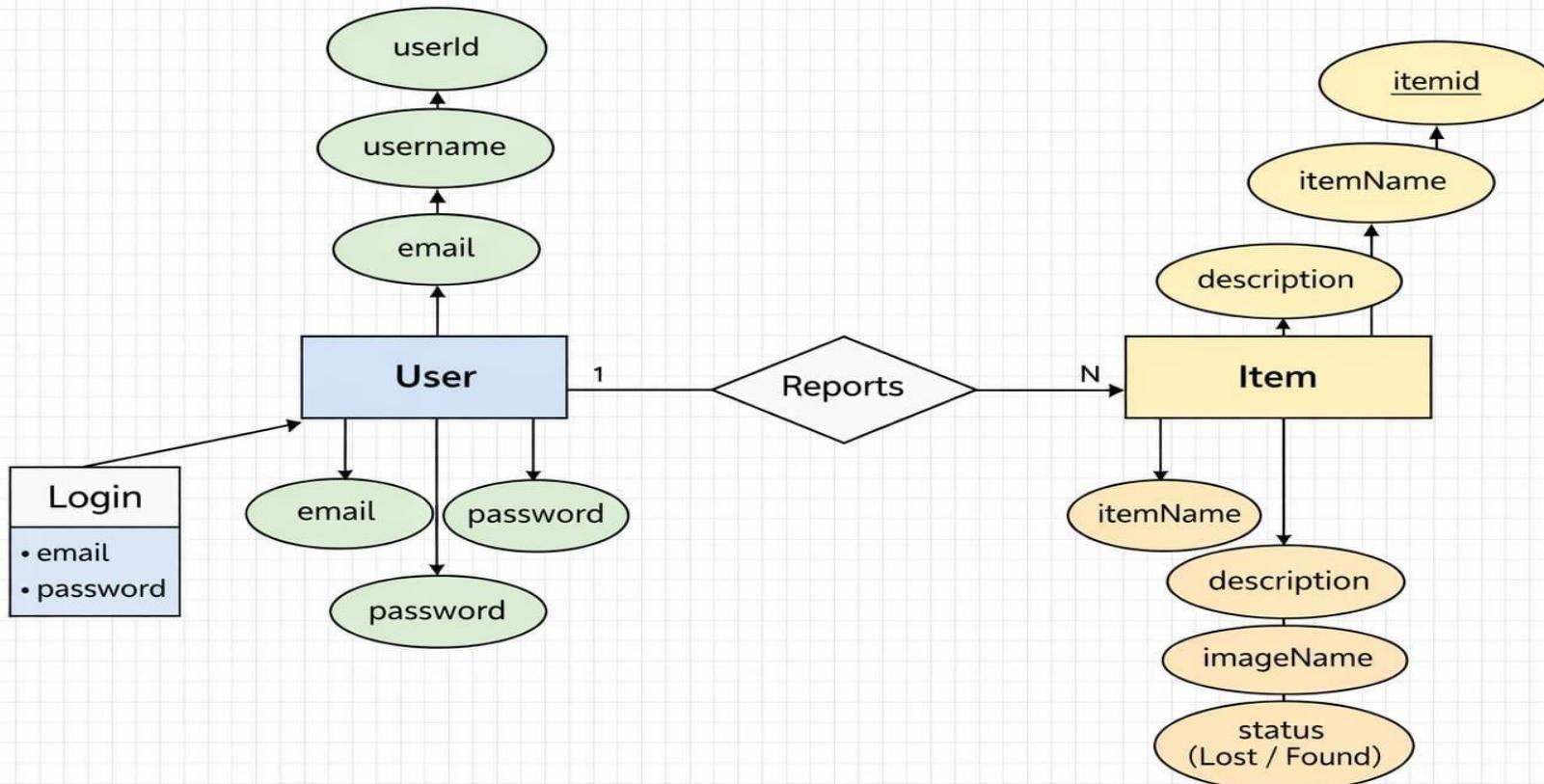
Item Management Module

- ✓ Add Lost/Found Item
- ✓ View Items
- ✓ Search Items

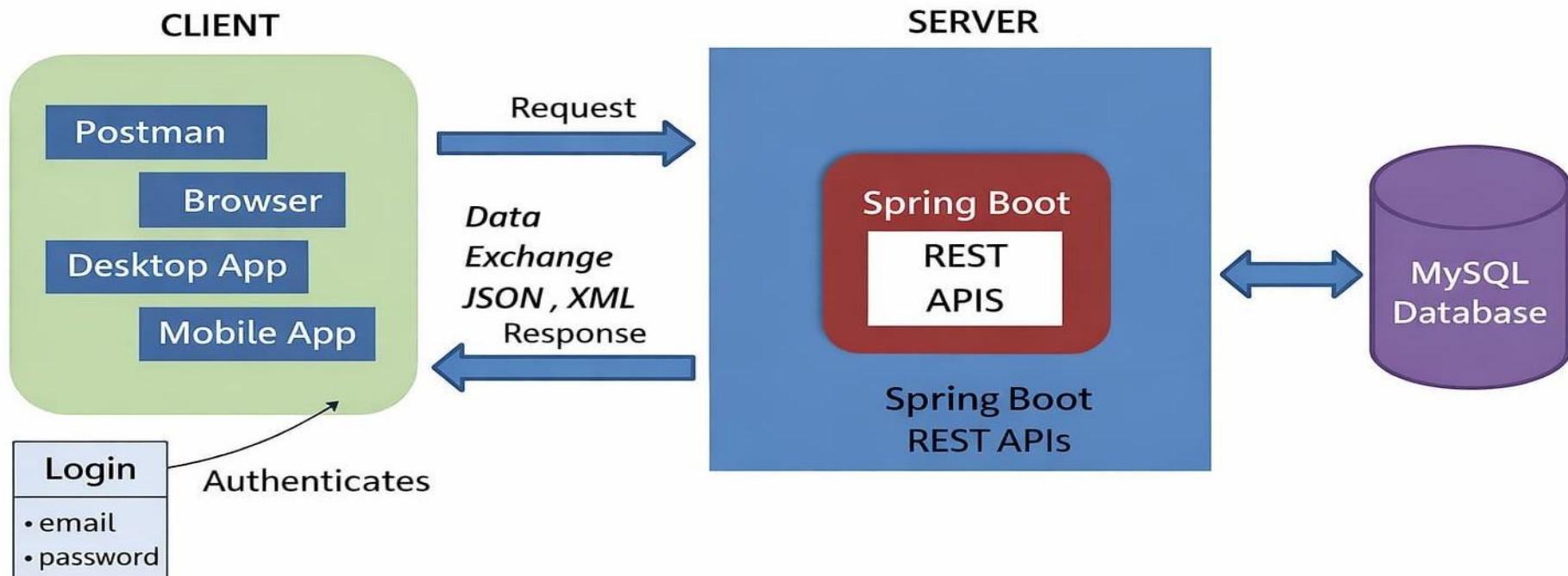
Database Module

- ✓ User Table
- ✓ Items Table

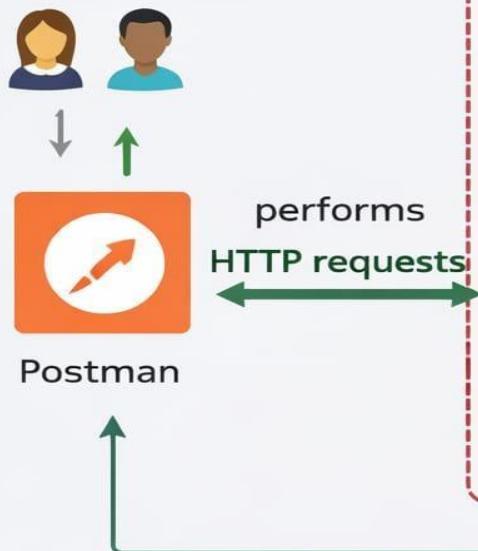
ER DIAGRAM



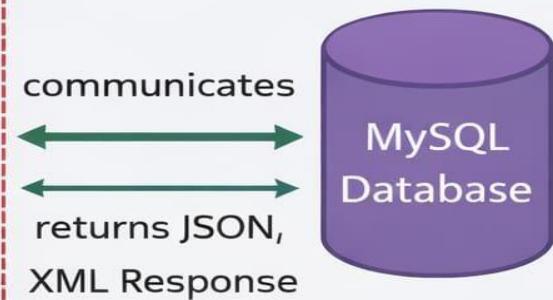
Client – Server Architecture



Client Interaction



Database Layer



returns JSON/XML Response

REGISTER MODUL

The Register Module allows new users to create an account in the Lost and Found Management System. Users are required to provide details such as username, email, and password. The entered details are validated and stored securely in the MySQL database using Spring Boot and Hibernate.

Functionality:

- New user registration
- Email uniqueness validation
- Stores user data in database
- Redirects user to login page after successful registration

LOGIN MODULE

The Login Module authenticates registered users by verifying their email and password. Only valid users are allowed to access the system. Once authenticated, the user is redirected to the home page.

Functionality:

- ❖ User authentication
- ❖ Email and password validation
- ❖ Session handling
- ❖ Logout functionality

Output:

- ❖ Successful login redirects to Index (Home) page
- ❖ Invalid login shows error message

HOME MODULE

The Index Module is the main dashboard of the Lost and Found Management System. After login, users are welcomed and can view the system features. It provides navigation to view items, add items, and logout.

Functionality:

- Displays welcome message with username
- Navigation buttons for Lost & Found Items
- Logout option

Example Output:

“Welcome, Kommi Lekha Sree”

Buttons: Lost & Found Items, Logout

ADD ITEM MODULE

The Add Item Module allows users to report a lost or found item by entering item details. The information is stored in the database and displayed in the items list.

Fields Included:

- Item Name
- Description
- Image (optional)
- Status (Lost / Found)

Functionality:

- Add new lost or found item
- Save item details in MySQL
- Redirect to items list after submission

ITEMS MODULE

The Items Module displays all reported lost and found items in a card-based format. Users can search items using keywords such as item name or place.

Functionality:

- ✓ Display all items
- ✓ Search items by name or description
- ✓ View item image and status
- ✓ Real-time filtering using search bar

Output:

- ✓ Cards showing item name, place, and image
- ✓ Search bar functionality

Http Request Methods

HTTP defines a set of **request methods** to indicate the desired action to be performed for a given resource

GET	http://localhost:8087/login	Return the list of all lost & found items
GET	http://localhost:8087/items/5	Return item details of item with id = 5
POST	http://localhost:8087/api/items	Add a new lost or found item
PUT	http://localhost:8087/items/mark-found/5	Mark the item with id = 5 as FOUND
DELETE	http://localhost:8087/items/5	Delete item with id = 5

Http
Methods

DATA DICTIONARY

TABLES OF DATABASE

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- Tables
- Views
- Stored Procedures
- Functions
- lafsdb
- lost_and_found
 - Tables
 - Views
 - Stored Procedures
 - Functions
- lost_and_found_db
 - Tables
 - Views
 - Stored Procedures
 - Functions
- lost_found_db
- sys

Administration Schemas

Information

Schema: lost_and_found_db

Object Info Session

SQL File 5* × SQL File 3 SQL File 4

SHOW DATABASES;
USE lost_and_found_db;
SHOW TABLES;

Result Grid | Filter Rows: Export: Wrap Cell Content: □

Tables_in_lost_and_found_db	
▶	items
▶	users

Result 8 × Read Only Context Help Snippets

Action Output

#	Time	Action	Message	Duration / Fetch
4	09:05:13	SHOW TABLES	1 row(s) returned	0.000 sec / 0.000 sec
5	09:05:33	SELECT COUNT(*) FROM items LIMIT 0, 300	1 row(s) returned	0.016 sec / 0.000 sec
6	09:07:50	USE lost_and_found_db	0 row(s) affected	0.000 sec
7	09:07:53	SHOW TABLES	2 row(s) returned	0.000 sec / 0.000 sec
8	09:08:04	SELECT COUNT(*) FROM items LIMIT 0, 300	1 row(s) returned	0.000 sec / 0.000 sec
9	14:56:47	SHOW DATABASES	10 row(s) returned	0.016 sec / 0.000 sec
10	14:56:52	USE lost_and_found_db	0 row(s) affected	0.000 sec
11	14:57:13	SELECT COUNT(*) FROM items LIMIT 0, 300	1 row(s) returned	0.000 sec / 0.000 sec
12	14:59:01	SELECT * FROM item LIMIT 0, 300	Error Code: 1146. Table 'lost_and_found_db.item' doesn't exist	0.031 sec
13	14:59:33	SELECT * FROM user LIMIT 0, 300	Error Code: 1146. Table 'lost_and_found_db.user' doesn't exist	0.016 sec
14	15:00:13	SHOW TABLES	2 row(s) returned	0.031 sec / 0.000 sec

SQLAdditions: Jump to

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

USER DATABASE

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- Tables
- Views
- Stored Procedures
- Functions
- lafldb
- lost_and_found
 - Tables
 - Views
 - Stored Procedures
 - Functions
- lost_and_found_db
 - Tables
 - Views
 - Stored Procedures
 - Functions
- lost_found_db
- sys

Administration Schemas

Information

Schema: lost_and_found_db

SQL File 5* × SQL File 3 SQL File 4

CREATE DATABASE lost_and_found;
SHOW DATABASES;
USE lost_and_found_db;
SHOW TABLES;
SELECT * FROM users;

Result Grid | Filter Rows: Edit: Export/Import: Result Grid Form Editor Field Types

	id	username	email	password
▶	1	admin	admin@123	123
▶	2	Kommi	Kommi@123	123
▶	3	KommiLekhaSree	lekha@123	123
▶	4	Thilaga	thilaga@123	123
*	HULL	HULL	HULL	HULL

users 10 ×

Output

Action Output

#	Time	Action	Message	Duration / Fetch
14	15:00:13	SHOW TABLES	2 row(s) returned	0.031 sec / 0.000 sec
15	15:00:59	SELECT * FROM items LIMIT 0, 300	9 row(s) returned	0.000 sec / 0.000 sec
16	15:01:37	SELECT * FROM user LIMIT 0, 300	Error Code: 1146. Table 'lost_and_found_db.user' doesn't exist	0.000 sec
17	15:01:51	SELECT * FROM users LIMIT 0, 300	4 row(s) returned	0.016 sec / 0.000 sec

Object Info Session

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

ITEMS DATABASE

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- Tables
- Views
- Stored Procedures
- Functions
- lafsdb
- lost_and_found
 - Tables
 - Views
 - Stored Procedures
 - Functions
- lost_and_found_db
 - Tables
 - Views
 - Stored Procedures
 - Functions
- lost_found_db
- sys

Administration Schemas

Information

Schema: lost_and_found_db

SQL File 5* × SQL File 3 SQL File 4

```
3 • USE lost_and_found_db;
4 • SHOW TABLES;
5 • SELECT * FROM items;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Result Grid | Form Editor | Field Types

	id	description	item_name	status	image_name
▶	18	dass Room	Watch	LOST	1765943763970_watch.jpg
19	Canteen	Bangle	LOST	1765943820871_images.jpg	
20	Ground	Hat	FOUND	1765943885820_images (1).jpg	
21	Library	Keys	LOST	1765943944829_images (2).jpg	
22	Corridor	Ring	FOUND	1765943996917_download.jpg	
23	Canteen	Bottle	FOUND	17659 1765943996917_download.jpg	
24	Ground	Bag	LOST	1765944459132_download (1).jpg	
25	dass Room	Lunch Box	LOST	1765944548625_download (2).jpg	
26	Library	Watch girl	LOST	1765949584201_watch.jpg	
*	HULL	HULL	HULL	HULL	HULL

items 9 × Apply Revert Context Help Snippets

Action Output

#	Time	Action	Message	Duration / Fetch
5	09:05:33	SELECT COUNT(*) FROM items LIMIT 0, 300	1 row(s) returned	0.016 sec / 0.000 sec
6	09:07:50	USE lost_and_found_db	0 row(s) affected	0.000 sec
7	09:07:53	SHOW TABLES	2 row(s) returned	0.000 sec / 0.000 sec
8	09:08:04	SELECT COUNT(*) FROM items LIMIT 0, 300	1 row(s) returned	0.000 sec / 0.000 sec
9	11:55:17	SHOW DATABASES	12 row(s) returned	0.016 sec / 0.000 sec

Object Info Session

COUNT DATABASE

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- Tables
- Views
- Stored Procedures
- Functions
- lafsdb
- lost_and_found
 - Tables
 - Views
 - Stored Procedures
 - Functions
- lost_and_found_db
 - Tables
 - Views
 - Stored Procedures
 - Functions
- lost_found_db
- sys

Administration Schemas

Information

Schema: lost_and_found_db

SQL File 5* × SQL File 3 SQL File 4

```
4 • SHOW TABLES;
5 • SELECT COUNT(*) FROM items;
```

Result Grid | Filter Rows: Result Grid

COUNT(*)
3

Result 5 × Read Only

Action Output

#	Time	Action	Message	Duration / Fetch
1	08:48:52	SHOW DATABASES	10 row(s) returned	0.000 sec / 0.000 sec
2	09:01:24	SELECT COUNT(*) FROM items LIMIT 0, 300	Error Code: 1046. No database selected Select the default D...	0.000 sec
3	09:04:58	USE lost_and_found	0 row(s) affected	0.000 sec
4	09:05:13	SHOW TABLES	1 row(s) returned	0.000 sec / 0.000 sec
5	09:05:33	SELECT COUNT(*) FROM items LIMIT 0, 300	1 row(s) returned	0.016 sec / 0.000 sec
6	09:07:50	USE lost_and_found_db	0 row(s) affected	0.000 sec
7	09:07:53	SHOW TABLES	2 row(s) returned	0.000 sec / 0.000 sec
8	09:08:04	SELECT COUNT(*) FROM items LIMIT 0, 300	1 row(s) returned	0.000 sec / 0.000 sec

SQLAdditions: Jump to

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Context Help Snippets

Object Info Session

POST METHOD FOR ADD DATA

The screenshot shows the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', 'API Network', a search bar, and various icons for user profile, invite, settings, and upgrade.

The left sidebar contains sections for 'Collections', 'Environments', 'History', 'Flows', and 'Files' (BETA).

The main workspace displays a POST request to `http://localhost:8087/api/items`. The request details show the method as 'POST' and the URL as `http://localhost:8087/api/items`. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   "description": "Lost near library",  
3   "id": 27,  
4   "imageName": null,  
5   "itemName": "Wallet",  
6   "status": "LOST"  
7 }
```

The response status is '200 OK' with a duration of '244 ms' and a size of '260 B'. Below the response, there are icons for copy, refresh, search, and other actions.

At the bottom, the footer includes links for 'Cloud View', 'Find and replace', 'Console', 'Terminal', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and a help icon.

GET METHOD FOR GET ALL DATA

The screenshot shows the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', 'API Network', a search bar, and various icons for authentication, invite, settings, and upgrade.

The left sidebar contains icons for 'Overview', 'Collections', 'Environments', 'History', 'Flows', and 'Files (BETA)'. The 'Overview' section is active, showing a recent request to 'http://localhost:8087/api/items'.

The main workspace displays a GET request to 'http://localhost:8087/api/items'. The response status is '200 OK' with a duration of '165 ms' and a size of '1.15 KB'. The response body is shown as JSON:

```
[{"id": 18, "itemName": "Watch", "description": "class Room", "status": "LOST"}, {"id": 19, "itemName": "Bangle", "description": "Canteen", "status": "LOST"}, {"id": 20, "itemName": "Hat", "description": "Ground", "status": "FOUND"}]
```

At the bottom, there are links for 'Cloud View', 'Find and replace', 'Console', 'Terminal', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and a help icon.

PUT METHOD FOR UPDATE DATA

The screenshot shows the Postman application interface. On the left, there's a sidebar with icons for Overview, Collections, Environments, History, Flows, and Files. The main area has a header with 'Overview' and a search bar. Below the header, a request card is displayed:

- Method:** PUT
- URL:** <http://localhost:8087/api/items/{id}/found>
- Send** button

The 'Params' tab is selected in the request details. Under 'Query Params', there are two columns: 'Key' and 'Value'. The first row has 'Key' and 'Value' both empty. The second row has 'Key' as 'Description' and 'Value' as 'Description'.

Below the request details, the 'Body' tab is selected. It shows a JSON response with the following content:

```
1  {
2    "timestamp": "2025-12-17T10:03:41.307Z",
3    "status": 400,
4    "error": "Bad Request",
5    "path": "/api/items/%7Bid%7D/found"
6 }
```

The response status is shown as 400 Bad Request with a timestamp of 2025-12-17T10:03:41.307Z, a status of 400, an error message of 'Bad Request', and a path of '/api/items/%7Bid%7D/found'.

At the bottom, there are navigation links for Cloud View, Find and replace, Console, Terminal, Runner, Start Proxy, Cookies, Vault, Trash, and Help.

DELETE METHOD FOR DELETE DATA

The screenshot shows the Postman application interface. At the top, the title bar reads "DELETE METHOD FOR DELETE DATA". Below the title bar, the navigation bar includes "Home", "Workspaces", "API Network", a search bar "Search Postman", and various icons for "Invite", "Settings", and "Upgrade". On the left side, there are sidebar panels for "Collections", "Environments", "History", "Flows", and "Files" (BETA). The main workspace displays an API endpoint:

- HTTP Method:** DELETE
- URL:** http://localhost:8087/api/items/{id}/found
- Headers:** (8)
- Body:** (Green dot)
- Scripts:**
- Settings:**

Query Params:

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body: (Grey dot)

Test Results: (6) | ⏱

The results section shows the following details:

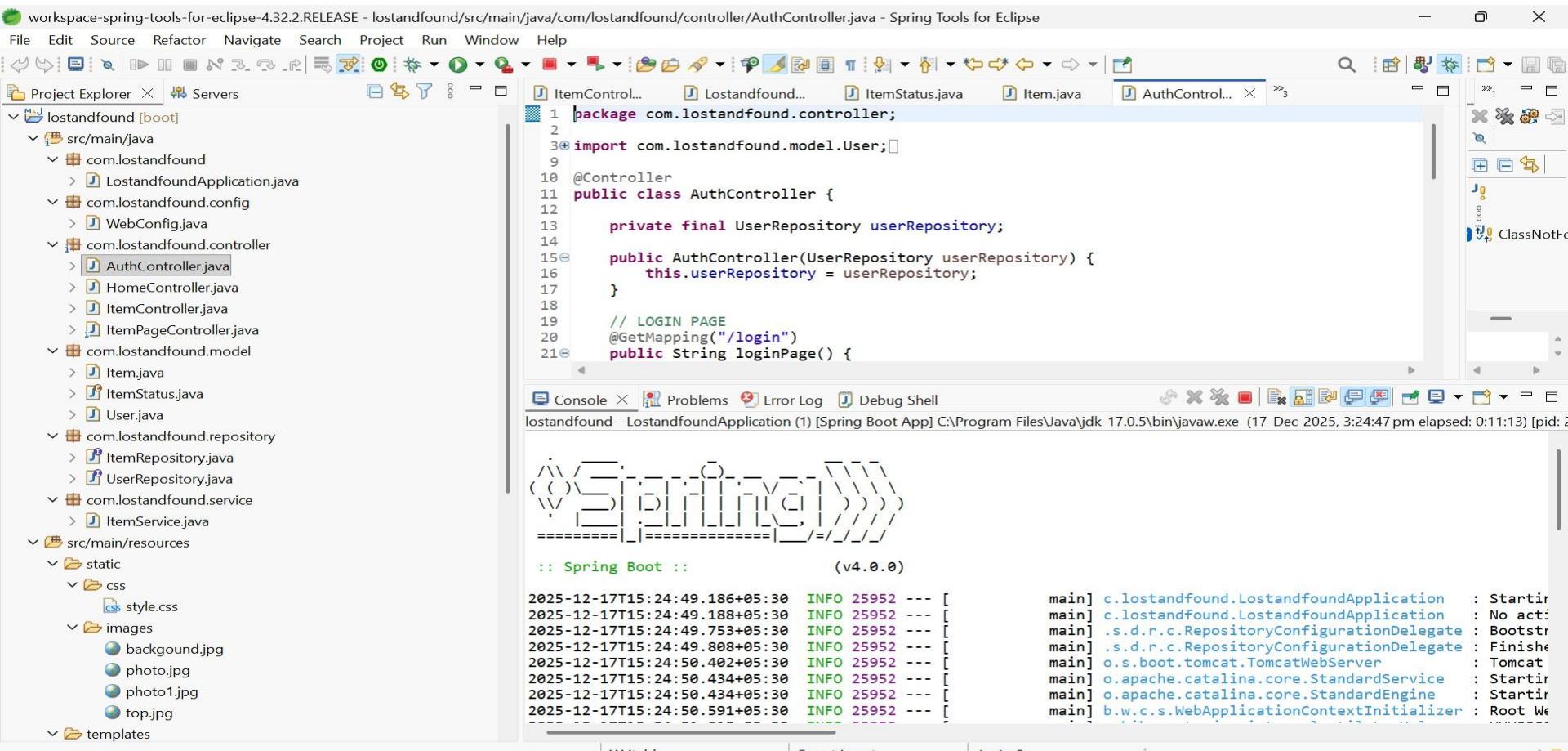
- Status: 405 Method Not Allowed
- Time: 136 ms
- Size: 309 B
- Global: (Globe icon)
- More: (Three dots icon)

The response body is displayed as JSON:

```
1 {  
2   "timestamp": "2025-12-17T10:01:31.941Z",  
3   "status": 405,  
4   "error": "Method Not Allowed",  
5   "path": "/api/items/%7Bid%7D/found"  
6 }
```

At the bottom, there are links for "Cloud View", "Find and replace", "Console", "Terminal", "Runner", "Start Proxy", "Cookies", "Vault", "Trash", and a question mark icon.

Spring Boot Module WORKSPACE



File Edit Source Navigate Search Project Run Window Help

Project Explorer X Servers

index.html Lostandfound... ItemStatus.java Item.java AuthControl... »3

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Lost & Found</title>
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <link rel="stylesheet" href="/css/style.css">
8 </head>
9 <body>
10 <!-- ===== HEADER ===== -->
11 <div class="header">
12   <h1>Lost & Found System</h1>
13
14   <!-- Show welcome message if logged in -->
15   <p th:if="${username != null}">
16     Welcome, ${username}!

```

Console X Problems Error Log Debug Shell

lostandfound - LostandfoundApplication (1) [Spring Boot App] C:\Program Files\Java\jdk-17.0.5\bin\javaw.exe (17-Dec-2025, 3:24:47 pm elapsed: 0:11:30) [pid: 2025-12-17T15:24:58.860+05:30 INFO 25952 --- [main] o.s.b.w.a.WelcomePageHandlerMapping : Adding
2025-12-17T15:24:59.791+05:30 INFO 25952 --- [main] o.s.boot.tomcat.TomcatWebServer : Tomcat
2025-12-17T15:24:59.803+05:30 INFO 25952 --- [main] c.lostandfound.LostandfoundApplication : Started
2025-12-17T15:25:04.140+05:30 INFO 25952 --- [nio-8087-exec-2] o.a.c.c.C.[Tomcat].[localhost].[] : Initia...
2025-12-17T15:25:04.141+05:30 INFO 25952 --- [nio-8087-exec-2] o.s.web.servlet.DispatcherServlet : Initia...
2025-12-17T15:25:04.142+05:30 INFO 25952 --- [nio-8087-exec-2] o.s.web.servlet.DispatcherServlet : Comple...
Hibernate: select i1_0.id,i1_0.description,i1_0.image_name,i1_0.item_name,i1_0.status from items i1_0
Hibernate: select i1_0.id,i1_0.description,i1_0.image_name,i1_0.item_name,i1_0.status from items i1_0 where i1_0...
2025-12-17T15:31:31.921+05:30 WARN 25952 --- [nio-8087-exec-6] .w.s.m.s.DefaultHandlerExceptionResolver : Resolve...
2025-12-17T15:33:41.301+05:30 WARN 25952 --- [nio-8087-exec-2] .w.s.m.s.DefaultHandlerExceptionResolver : Resolve...

Writable Insert 63 : 11 : 1562

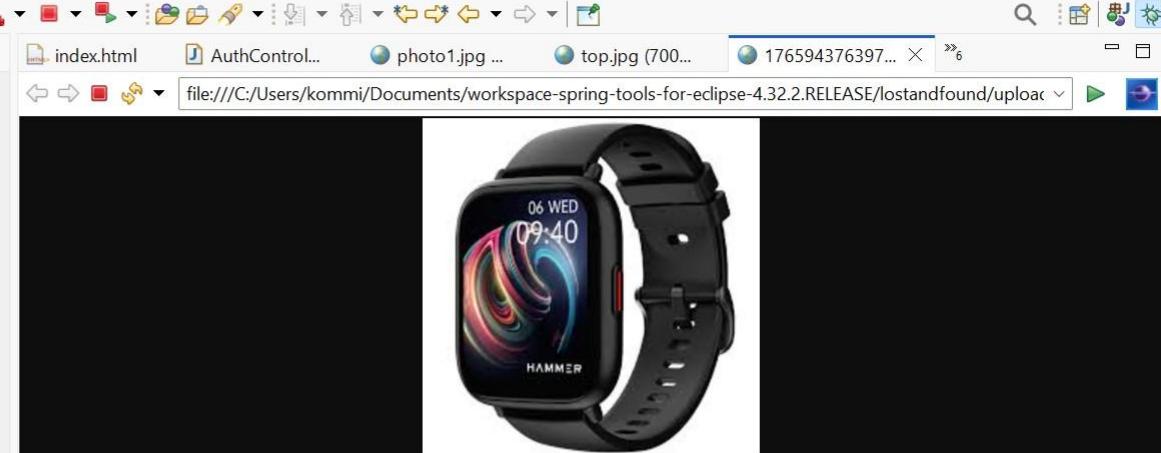


templates
 add-item.html
 index.html
 items.html
 login.html
 register.html
 application.properties

src/test/java
 JRE System Library [JavaSE-17]
 Maven Dependencies
 src
 target

uploads
 1765880979770_watch.jpg
 1765882204042_watch.jpg
 1765882444503_watch.jpg
 1765941338381_photo1.jpg
 1765942056087_watch.jpg
 1765942106537_top.jpg
 1765942117735_photo1.jpg
 1765943763970_watch.jpg
 1765943820871_images.jpg
 1765943885820_images (1).jpg
 1765943944829_images (2).jpg
 1765943996917_download.jpg
 1765944061694_images (3).jpg
 1765944459132_download (1).jpg
 1765944548625_download (2).jpg
 1765949584201_watch.jpg

HELP.md



Console X Problems Error Log Debug Shell

```
lostandfound - LostandfoundApplication (1) [Spring Boot App] C:\Program Files\Java\jdk-17.0.5\bin\javaw.exe (17-Dec-2025, 3:24:47 pm elapsed: 0:11:53) [pid: 2025-12-17T15:24:58.860+05:30 INFO 25952 --- [main] o.s.b.w.a.WelcomePageHandlerMapping : Adding
2025-12-17T15:24:59.791+05:30 INFO 25952 --- [main] o.s.boot.tomcat.TomcatWebServer : Tomcat
2025-12-17T15:24:59.803+05:30 INFO 25952 --- [main] c.lostandfound.LostandfoundApplication : Started
2025-12-17T15:25:04.140+05:30 INFO 25952 --- [nio-8087-exec-2] o.a.c.c.C.[Tomcat].[localhost].[] : Initia
2025-12-17T15:25:04.141+05:30 INFO 25952 --- [nio-8087-exec-2] o.s.web.servlet.DispatcherServlet : Initia
2025-12-17T15:25:04.142+05:30 INFO 25952 --- [nio-8087-exec-2] o.s.web.servlet.DispatcherServlet : Comple
Hibernate: select i1_0.id,i1_0.description,i1_0.image_name,i1_0.item_name,i1_0.status from items i1_0
Hibernate: select i1_0.id,i1_0.description,i1_0.image_name,i1_0.item_name,i1_0.status from items i1_0 where i1_0.
2025-12-17T15:31:31.921+05:30 WARN 25952 --- [nio-8087-exec-6] .w.s.m.s.DefaultHandlerExceptionResolver : Resolve
2025-12-17T15:33:41.301+05:30 WARN 25952 --- [nio-8087-exec-2] .w.s.m.s.DefaultHandlerExceptionResolver : Resolve
```

WEB SITE DRIVE LINK

LINK : <https://drive.google.com/file/d/12Pq1VAiDf5QZ-8fnbQyFhWzspiTf9DUO/view?usp=sharing>