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

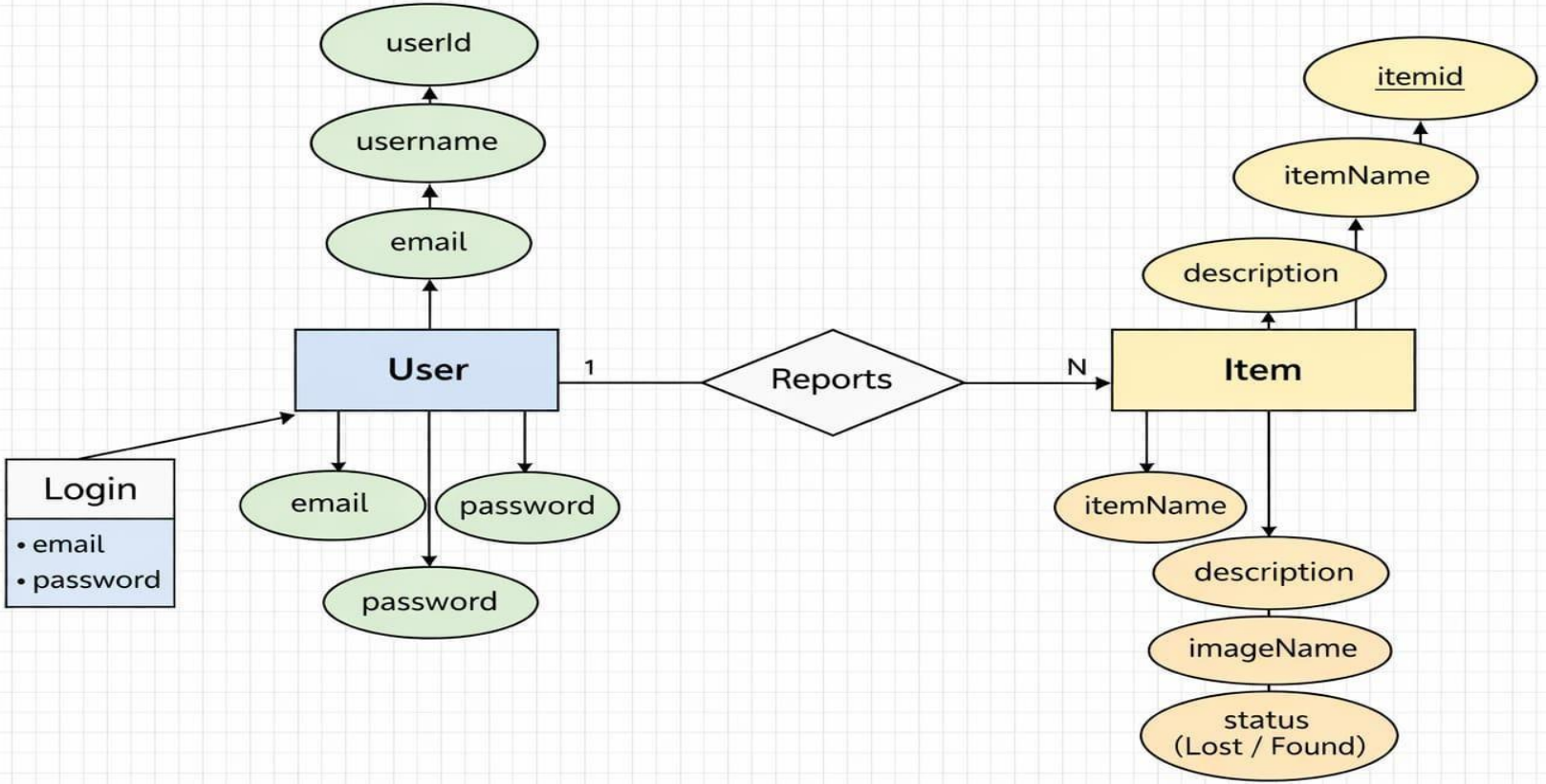
PROJECT MODULE

- **User Authentication Module**
  - ✓ User Registration
  - ✓ User Login & Logout

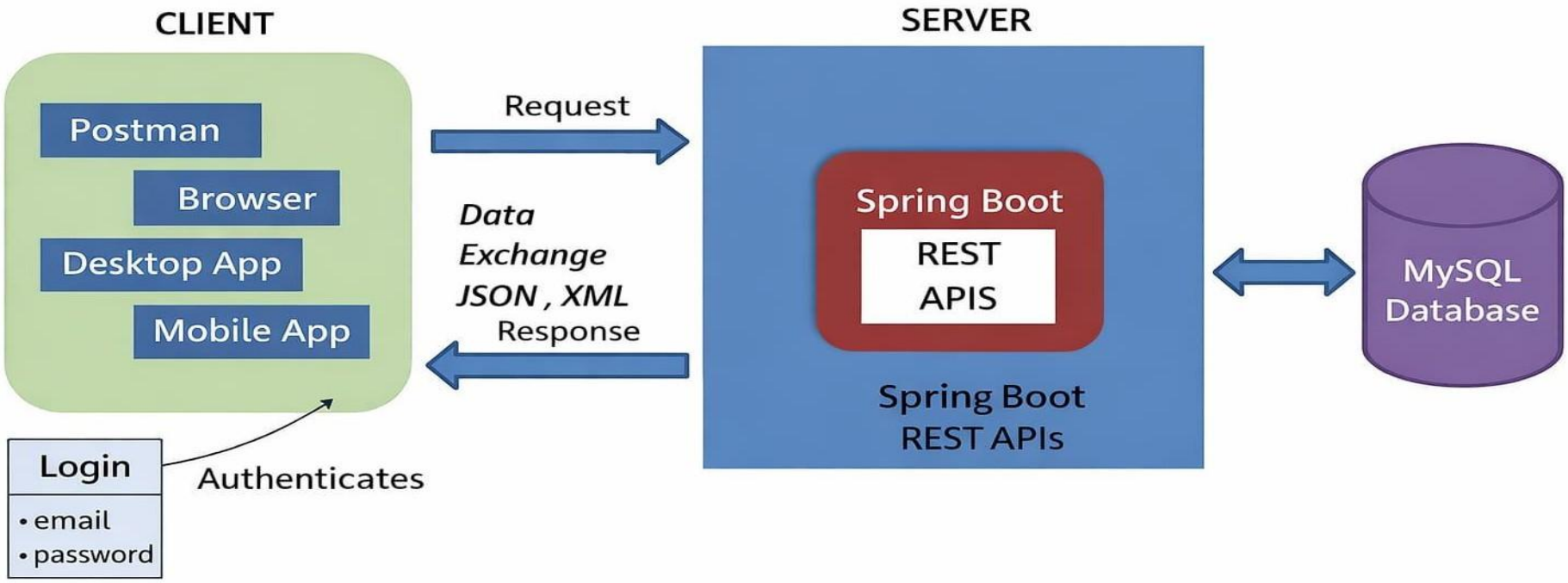
- **Item Management Module**
  - ✓ Add Lost/Found Item
  - ✓ View Items
  - ✓ Search Items

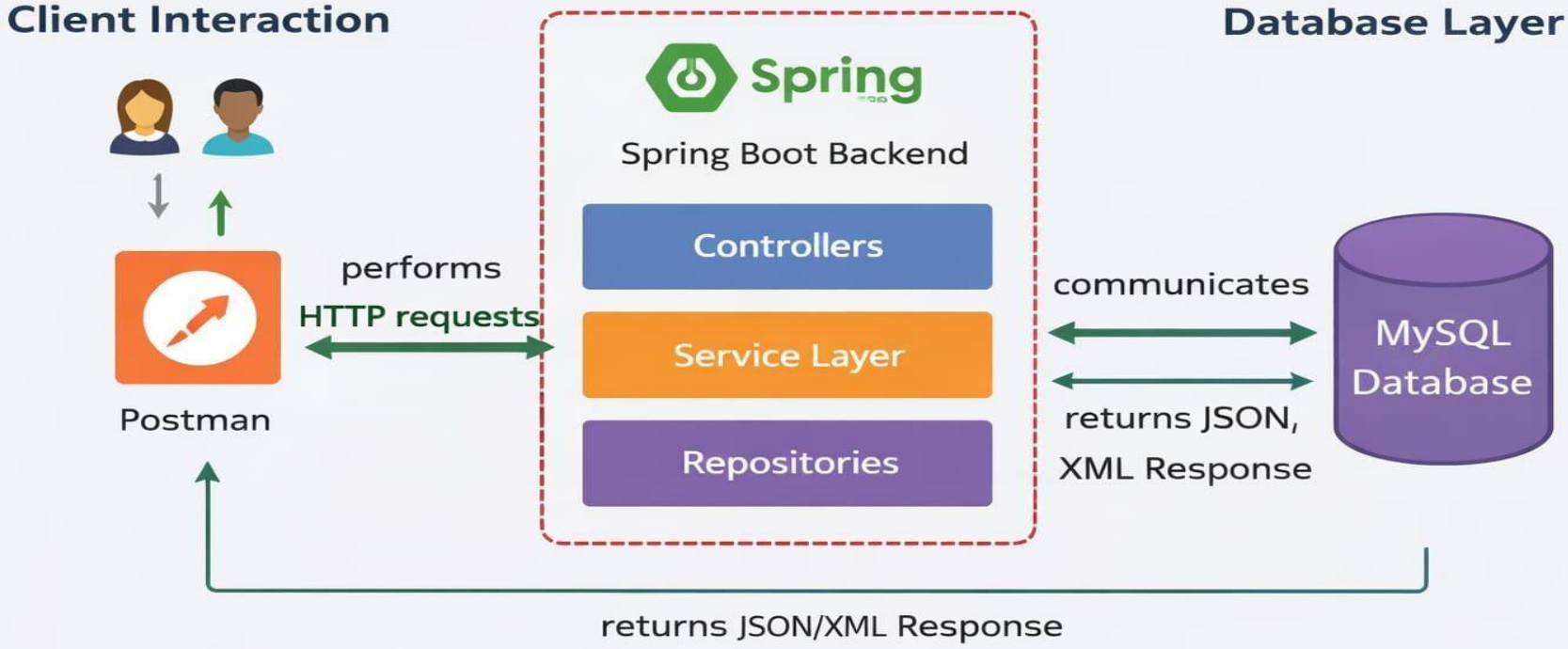
- **Database Module**
  - ✓ User Table
  - ✓ Items Table

ER DIAGRAM


userId
username
email
User
1
Reports
N
Item
itemid
itemName
description
Login
• email
• password
email
password
password
itemName
description
imageName
status
(Lost / Found)

Client – Server Architecture




Client Interaction
Database Layer
Spring
Spring Boot Backend
Controllers
Service Layer
Repositories
Postman
performs
HTTP requests
communicates
returns JSON,
XML Response
MySQL
Database
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

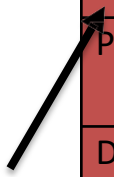
Output:

- ✓ 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

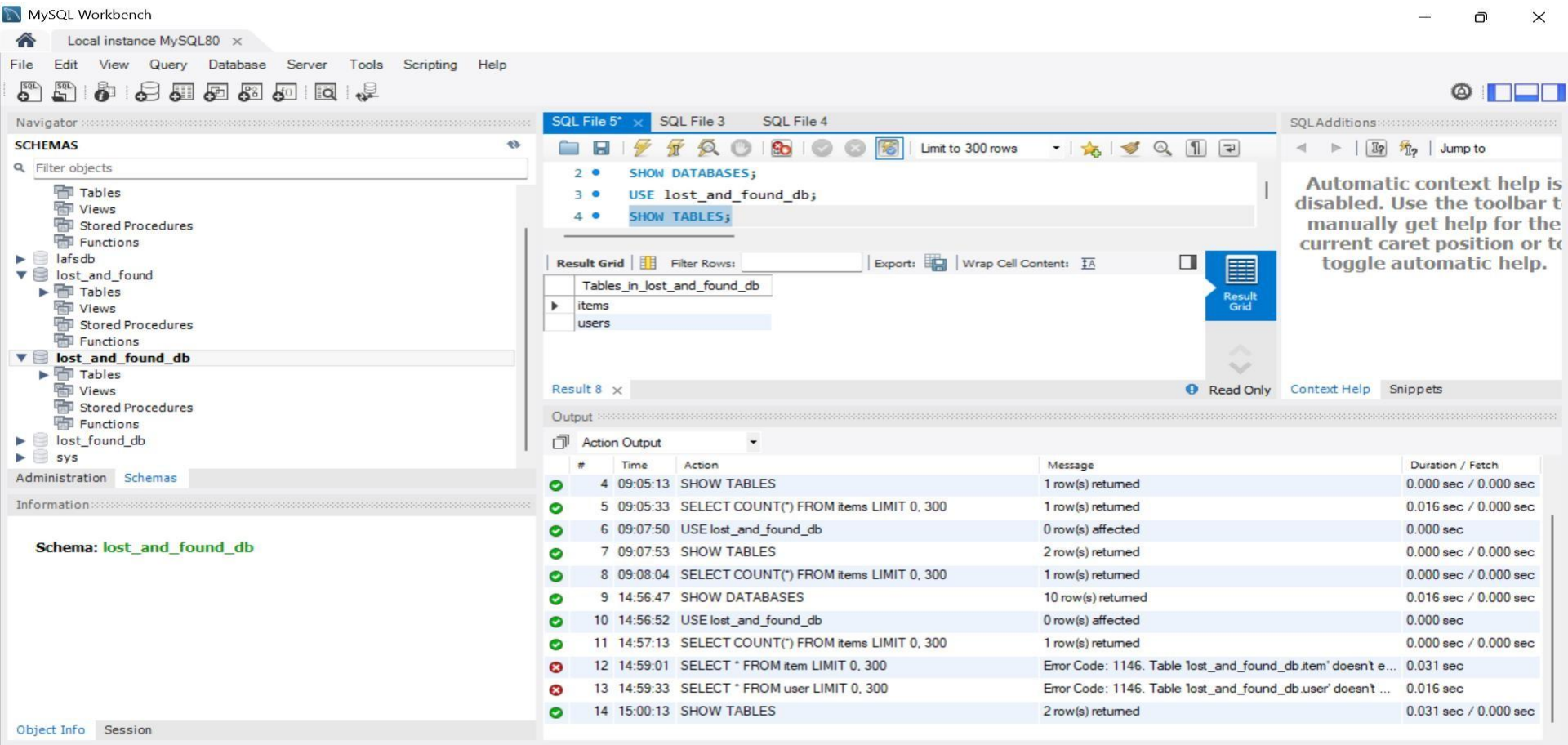
GET	http://localhost:8087/login	Return the list of all lost & found items
GET	http://localhost:8087/items/5	Return item details of item with id = 5
POST	http://localhost:8087/api/items	Add a new lost or found item
PUT	http://localhost:8087/items/mark-found/5	Mark the item with id = 5 as FOUND
DELETE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



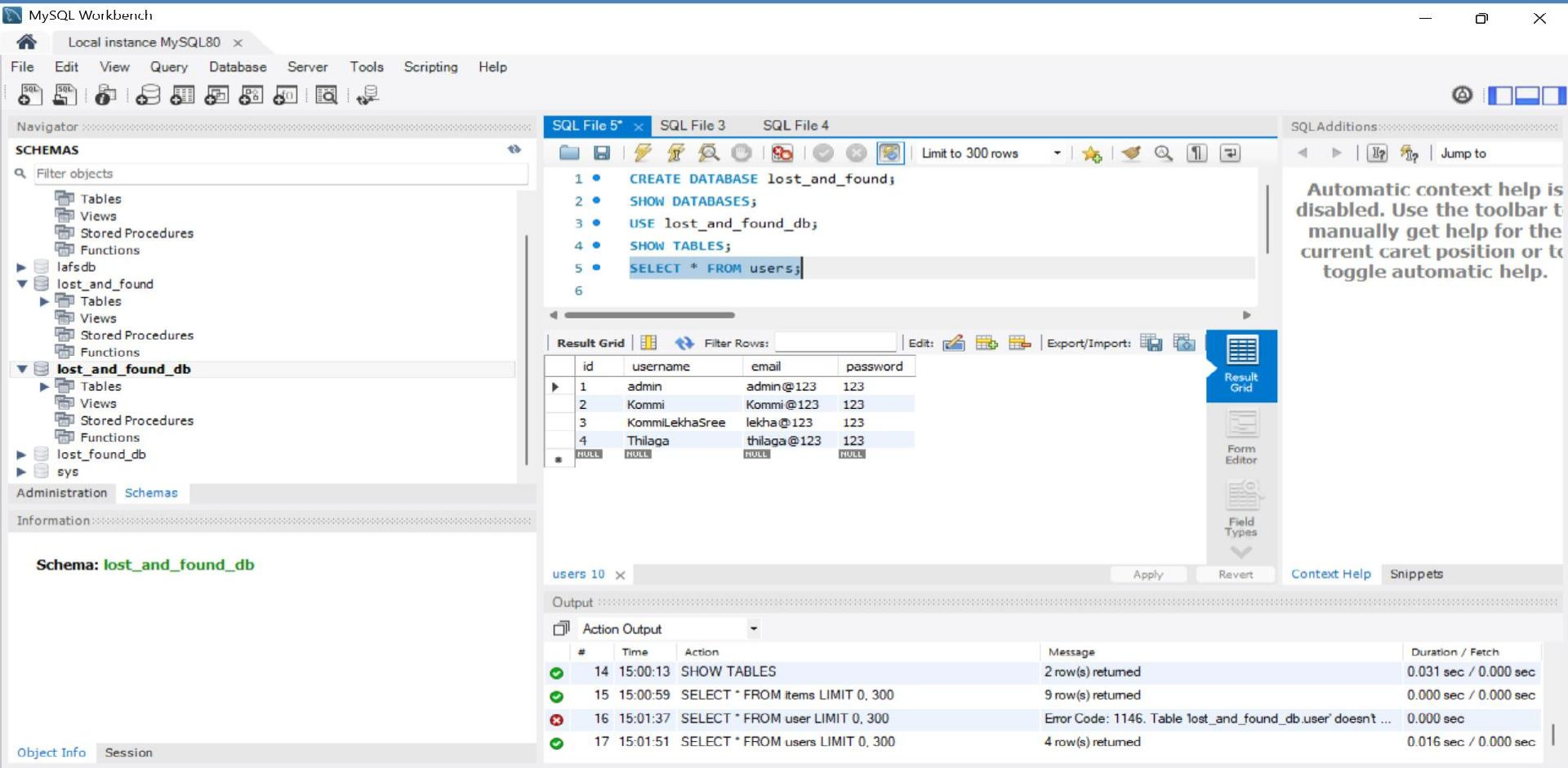
Http Methods

DATA DICTIONARY

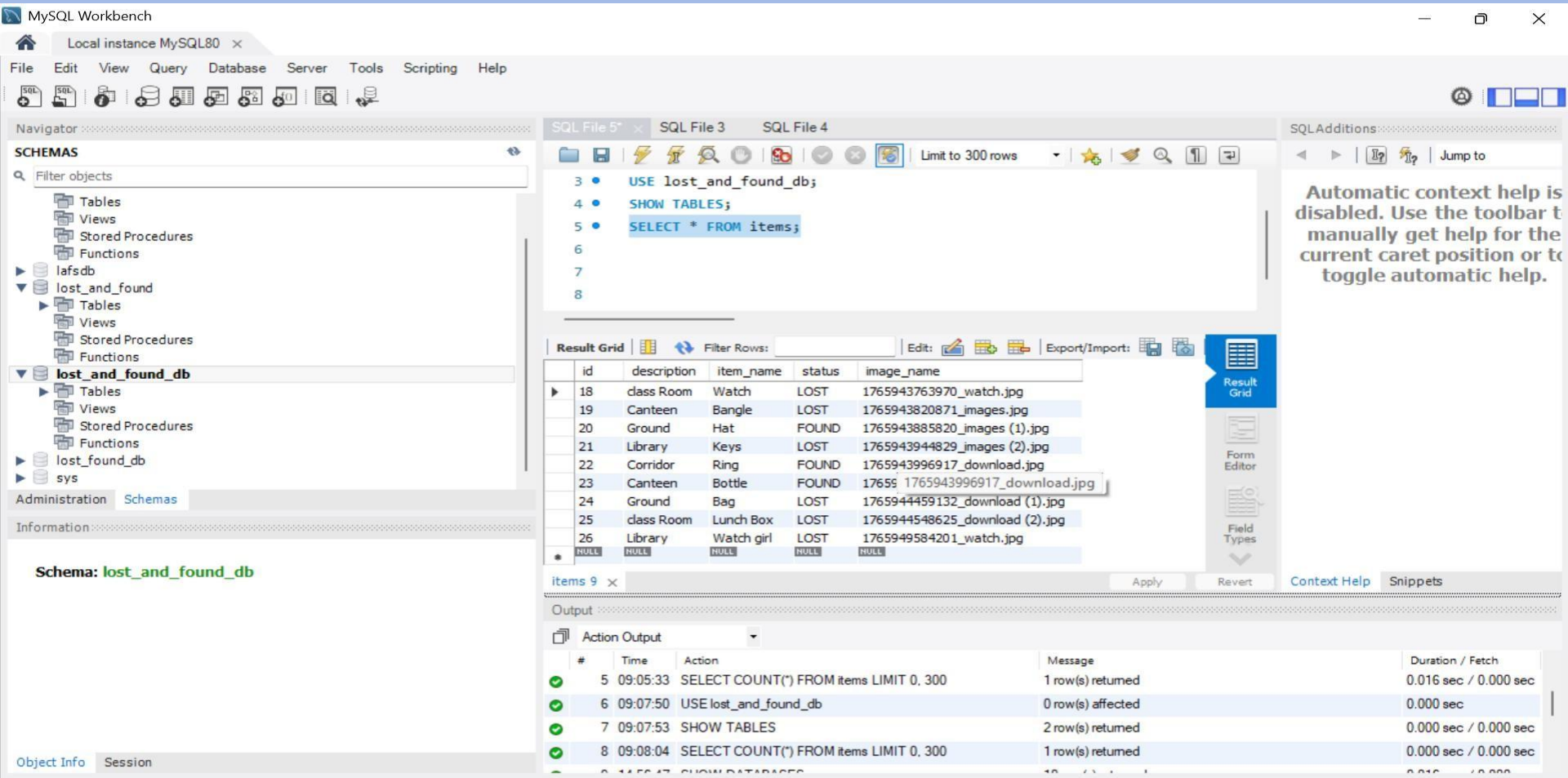
TABLES OF DATABASE



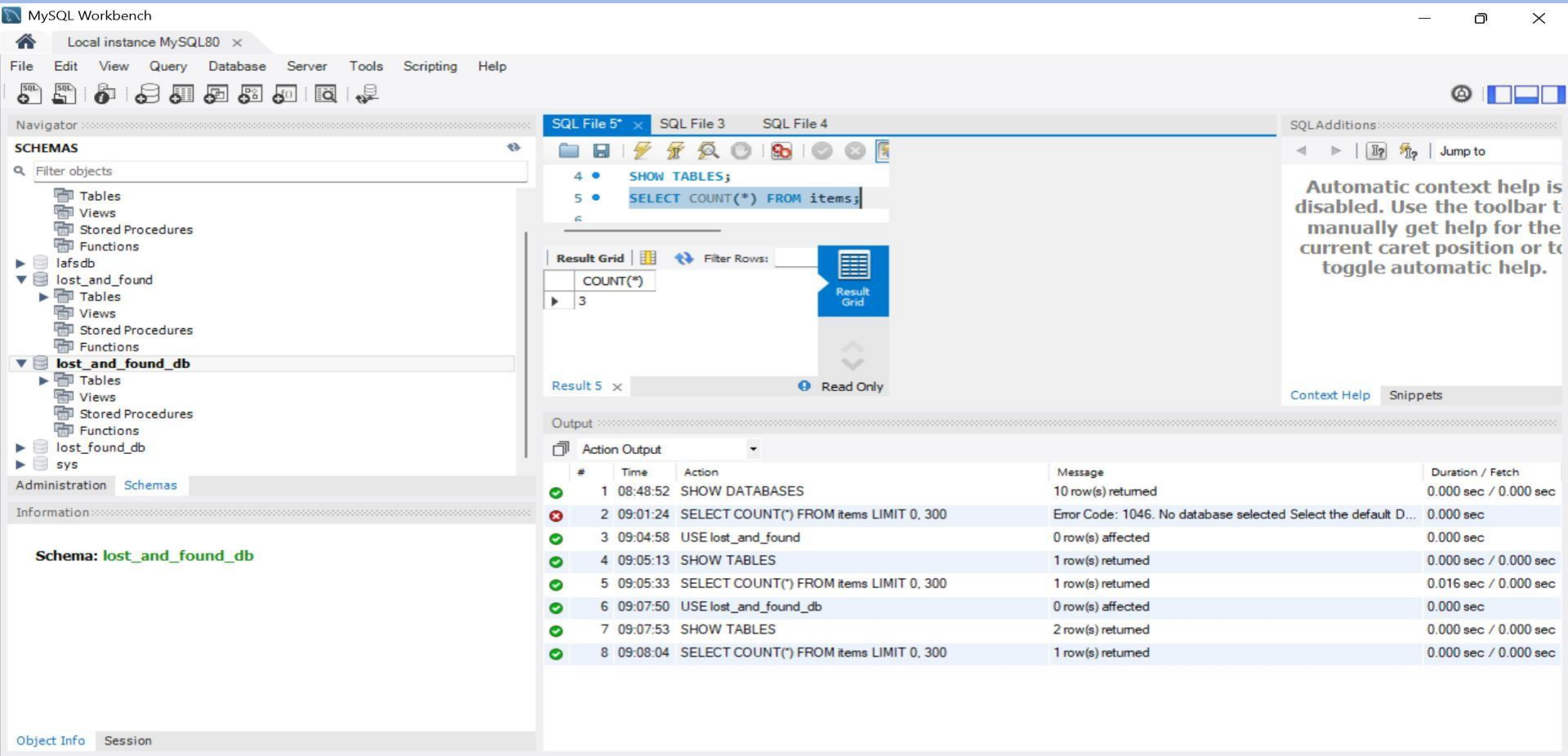
USER DATABASE



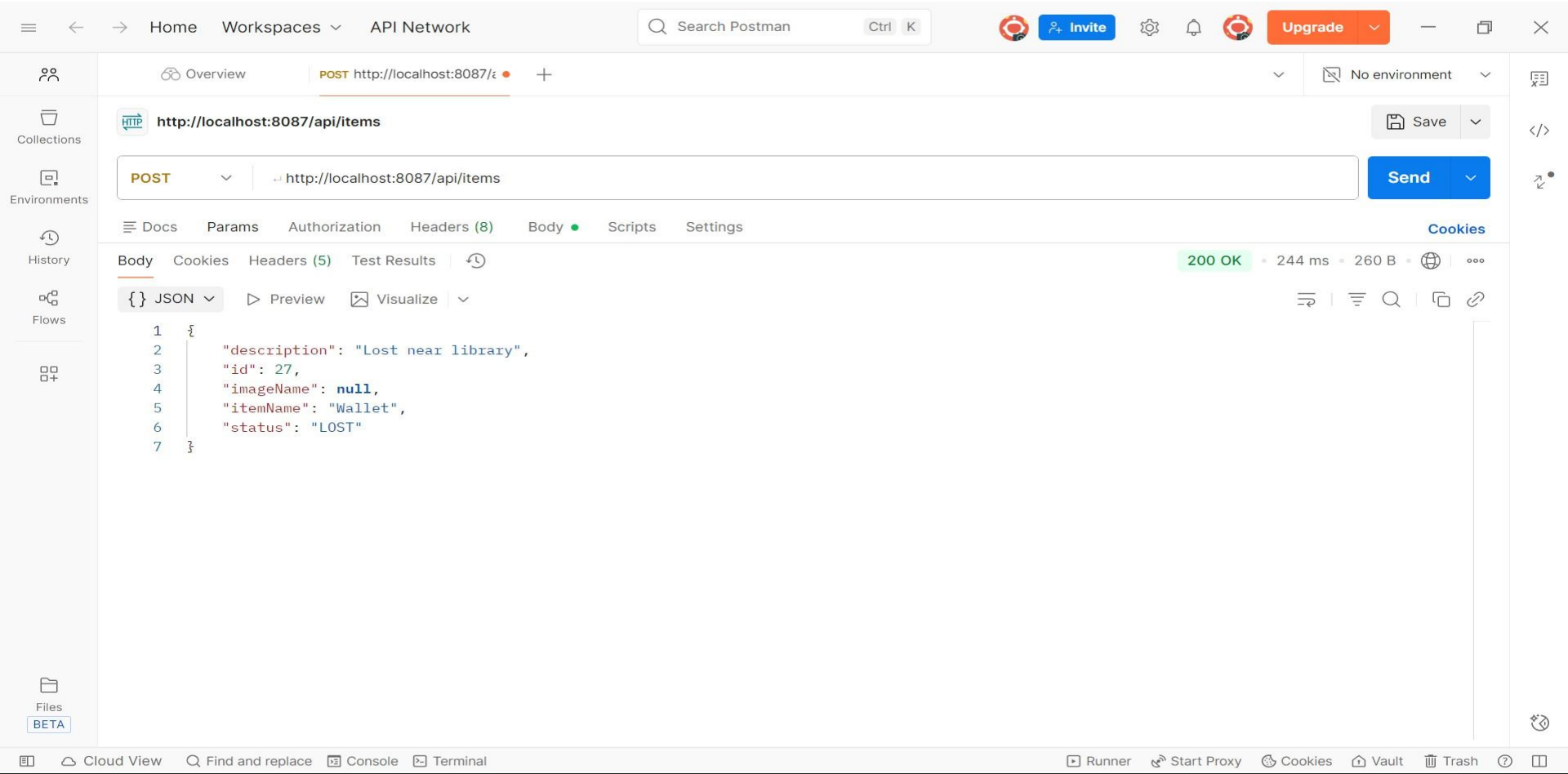
ITEMS DATABASE



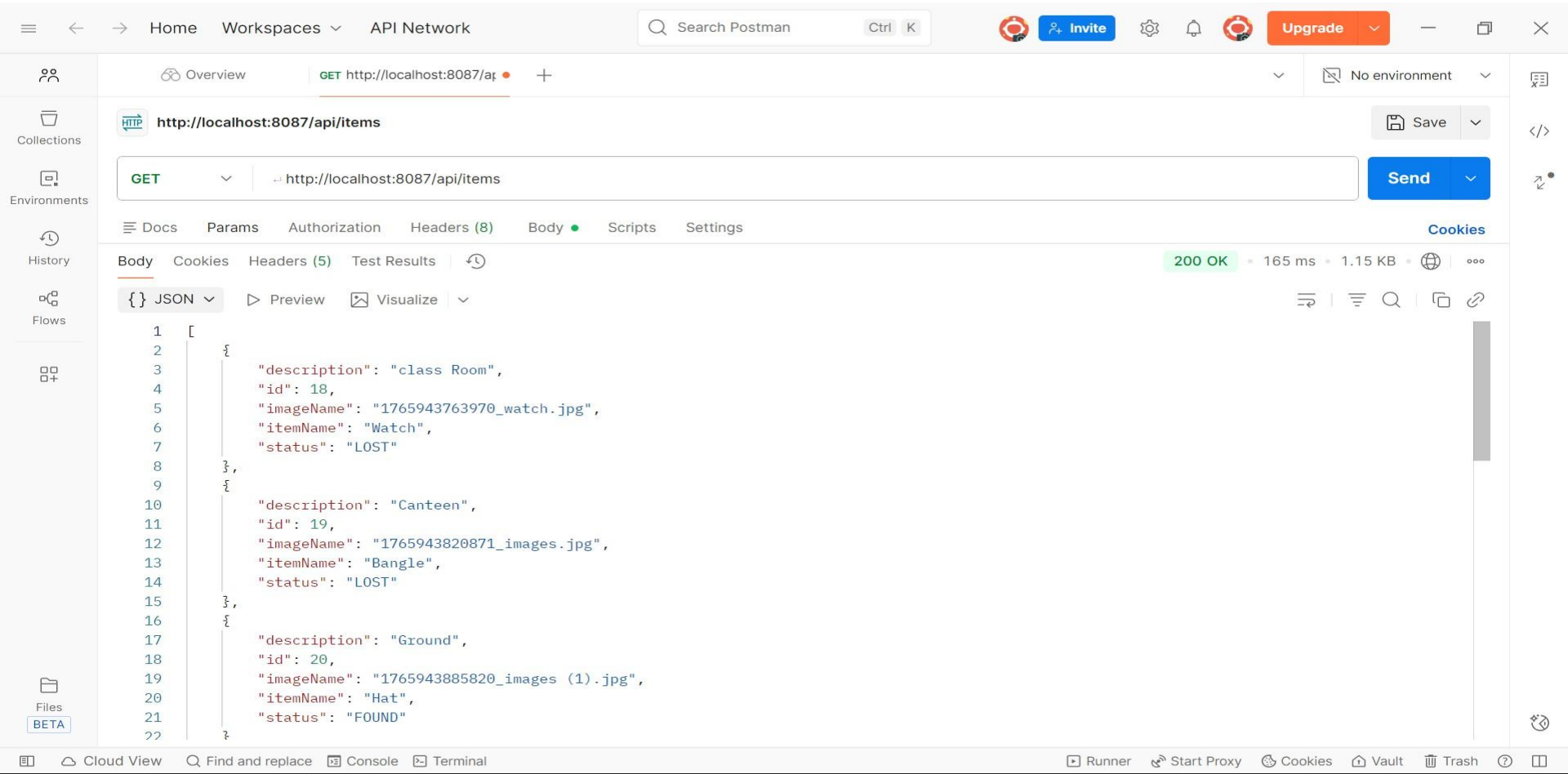
COUNT DATABASE



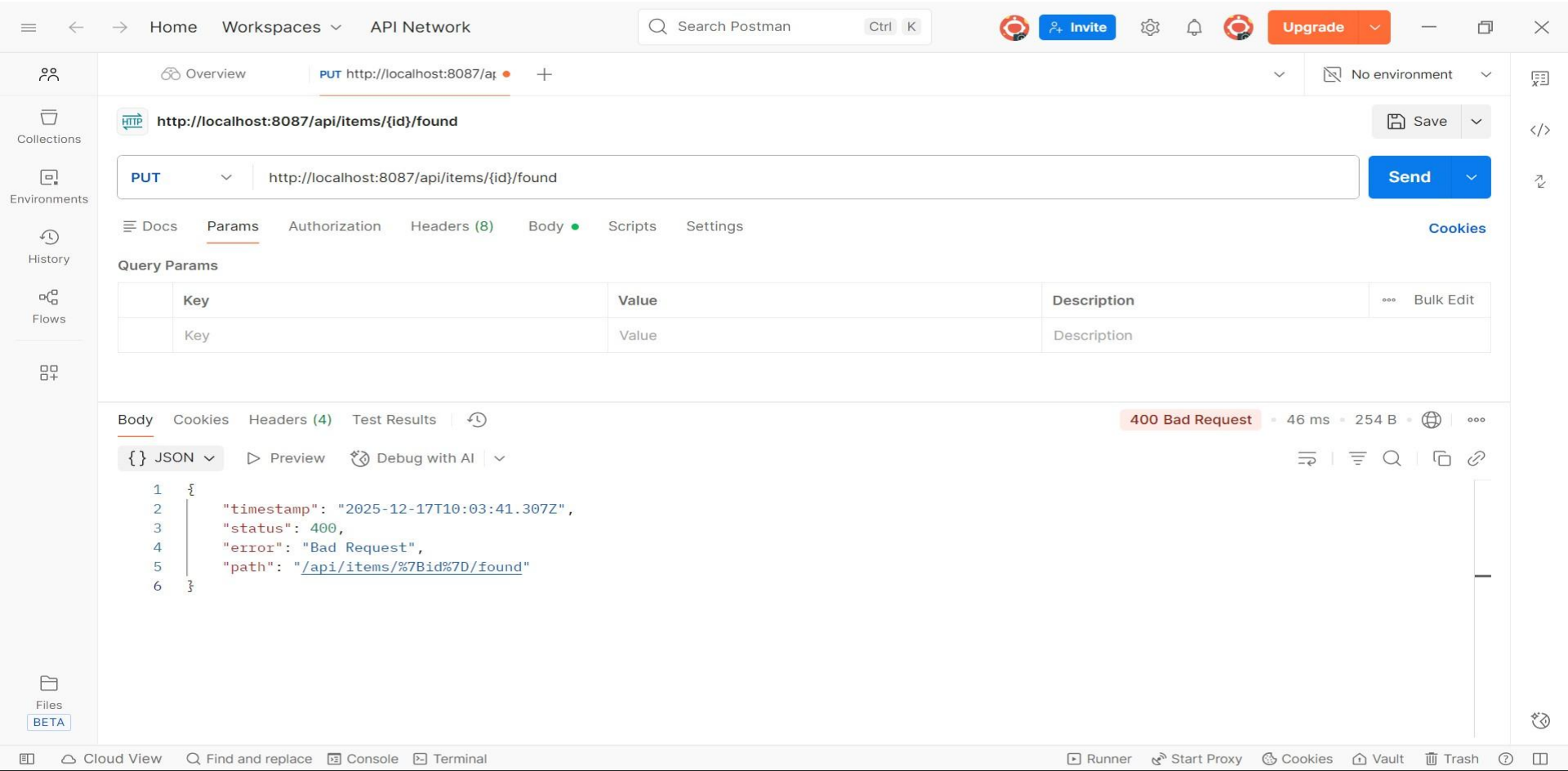
POST METHOD FOR ADD DATA



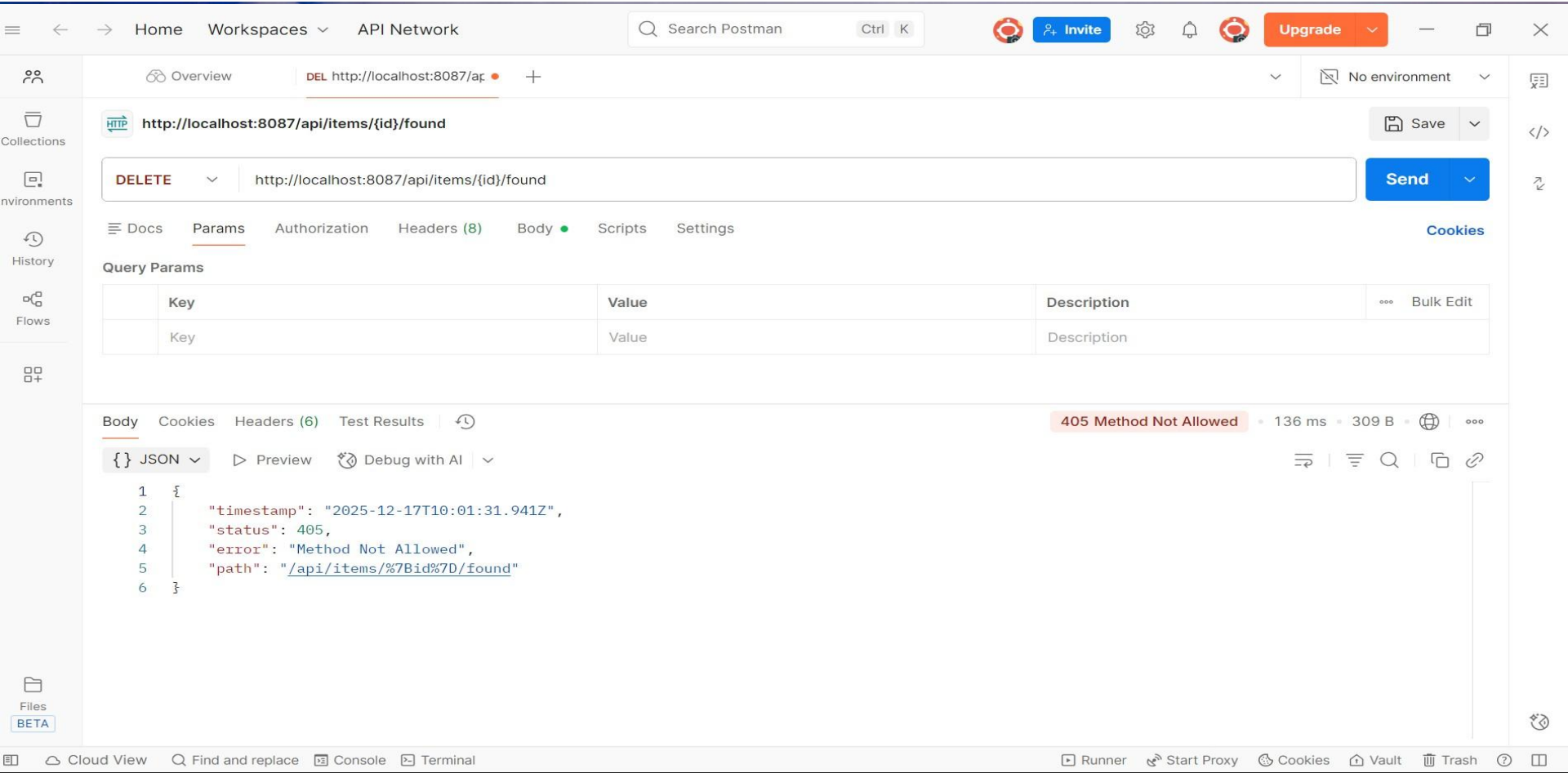
GET METHOD FOR GET ALL DATA



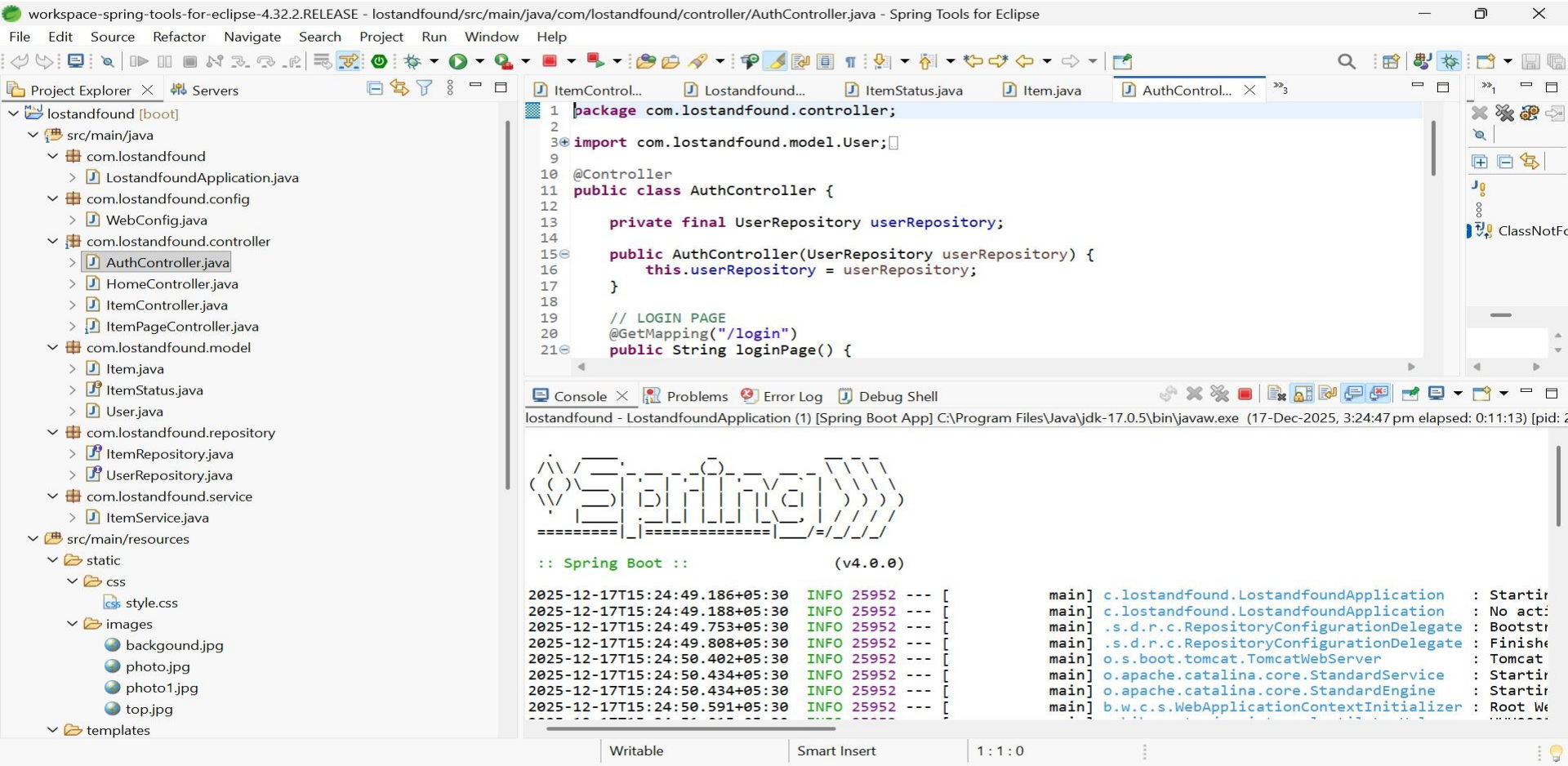
PUT METHOD FOR UPDATE DATA

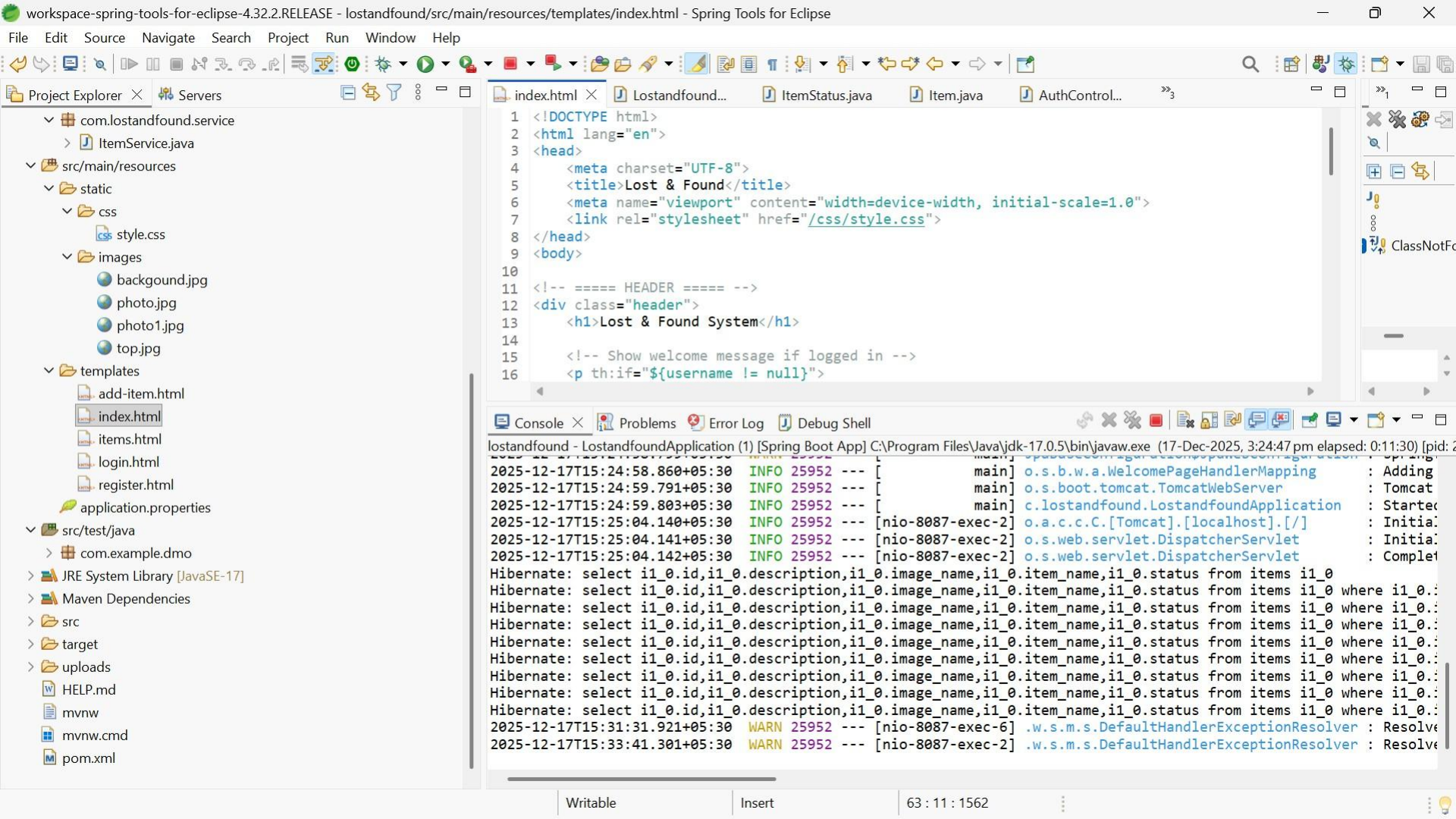


DELETE METHOD FOR DELETE DATA



Spring Boot Module WORKSPACE




workspace-spring-tools-for-eclipse-4.32.2.RELEASE - lostandfound/src/main/resources/templates/index.html - Spring Tools for Eclipse
File Edit Source Navigate Search Project Run Window Help
Project Explorer
Servers
com.lostandfound.service
ItemService.java
src/main/resources
static
css
style.css
images
backgound.jpg
photo.jpg
photo1.jpg
top.jpg
templates
add-item.html
index.html
items.html
login.html
register.html
application.properties
src/test/java
com.example.dmo
JRE System Library [JavaSE-17]
Maven Dependencies
src
target
uploads
HELP.md
mvnw
mvnw.cmd
pom.xml
index.html
Lostandfound...
ItemStatus.java
Item.java
AuthControl...
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
11 <!-- ===== HEADER ===== -->
12
13 <h1>Lost & Found System</h1>
14
15 <!-- Show welcome message if logged in -->
16 <p th:if="${username != null}">
Console
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].[/] : Initial
2025-12-17T15:25:04.141+05:30 INFO 25952 --- [nio-8087-exec-2] o.s.web.servlet.DispatcherServlet : Initial
2025-12-17T15:25:04.142+05:30 INFO 25952 --- [nio-8087-exec-2] o.s.web.servlet.DispatcherServlet : Complet
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.
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.
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.
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.
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.
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.
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

workspace-spring-tools-for-eclipse-4.32.2.RELEASE - lostandfound/uploads/1765943763970_watch.jpg - Spring Tools for Eclipse

File Edit Navigate Search Project Run Window Help

Project Explorer | Servers

- 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

index.html | AuthControl... | photo1.jpg ... | top.jpg (700... | 176594376397... | »6

file:///C:/Users/kommi/Documents/workspace-spring-tools-for-eclipse-4.32.2.RELEASE/lostandfound/uploac

ClassNotFc

Console | 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: 2

```
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   : Startec
2025-12-17T15:25:04.140+05:30  INFO 25952 --- [nio-8087-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initial
2025-12-17T15:25:04.141+05:30  INFO 25952 --- [nio-8087-exec-2] o.s.web.servlet.DispatcherServlet        : Initial
2025-12-17T15:25:04.142+05:30  INFO 25952 --- [nio-8087-exec-2] o.s.web.servlet.DispatcherServlet        : Complet
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.:
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.:
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.:
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.:
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.:
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.:
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.:
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