

Case Study On

Reclaim - Bringing Things Together

Introduction

- **Reclaim** is a Lost and Found Management System developed to digitally manage lost and found items.
- The system replaces the traditional manual process with a centralized software solution.
- It allows users to report lost and found items and track their status efficiently.
- Administrators can monitor, verify, and manage all item records through the system.
- The application improves efficiency, reduces errors, and simplifies item recovery.

Abstract

- Reclaim is designed to manage lost and found activities in a systematic and organized manner.
- The system helps users register lost and found items through a digital platform.
- Administrators can verify item details and manage records efficiently.
- The application ensures better tracking and transparency in item recovery.
- The system is implemented using Spring Boot, MySQL, and web technologies.

Client Requirements

- The system should allow users to register and log in securely.
- Users should be able to report lost and found items with relevant details.
- The admin should be able to view, update, and manage all item records.
- The system should support searching and filtering of items.
- Item status should be updated and tracked efficiently.

Technical Terms

- The application uses **role-based login** for users and administrators.
- Backend logic is handled using **Spring Boot controllers and services**.
- **REST endpoints** are used for communication between frontend and backend.
- **Input validation and exception handling** are implemented to avoid errors.
- **Maven** is used for dependency management and project build.

Technologies Used

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Java Spring Boot Framework
- **Build Tool:** Maven
- **Database:** MySQL
- **IDE & Tools:** Eclipse / STS, Browser (Chrome)

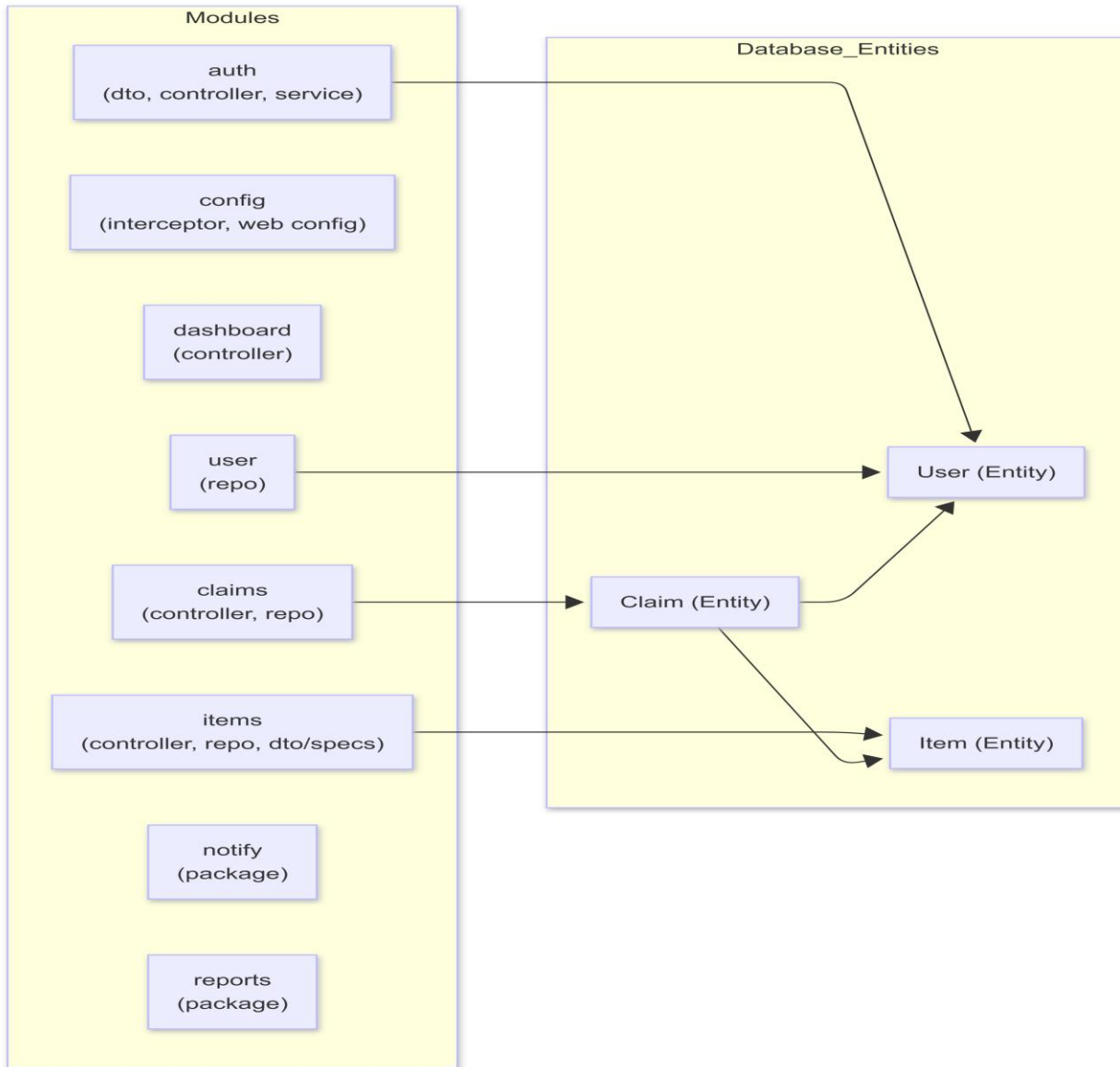
System Requirements

- **Operating System:** Windows / Linux
- **Backend Environment:** Java 8+ with Spring Boot
- **Database Server:** MySQL
- **Web Server:** Apache Tomcat (embedded)
- **Browser:** Google Chrome

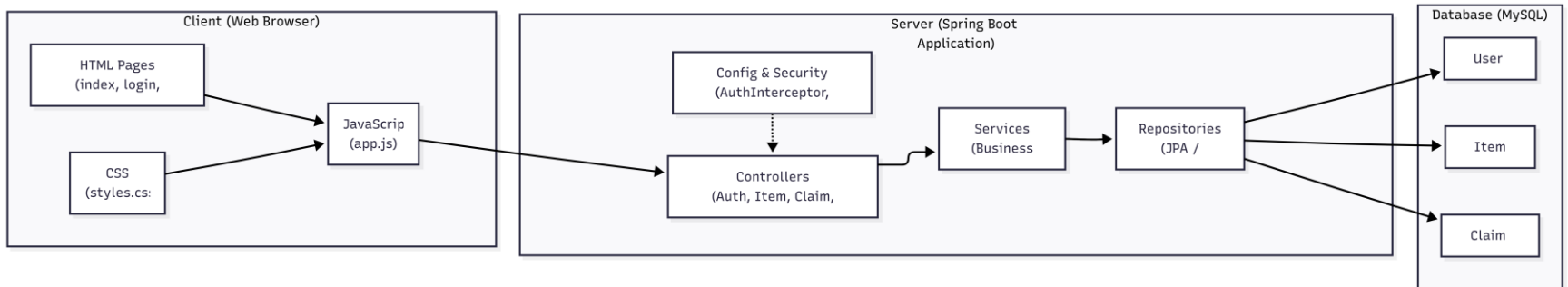
Project Modules

- **Authentication Module** - handles user login and registration
- **Admin Module** - handles overall controls of the app.
- **User Module** - manages user details and user-related operations
- **Items Module** - manages lost and found item details and status
- **Claims Module** - handles claiming and verification of items
- **Dashboard Module** - displays summary and overview information

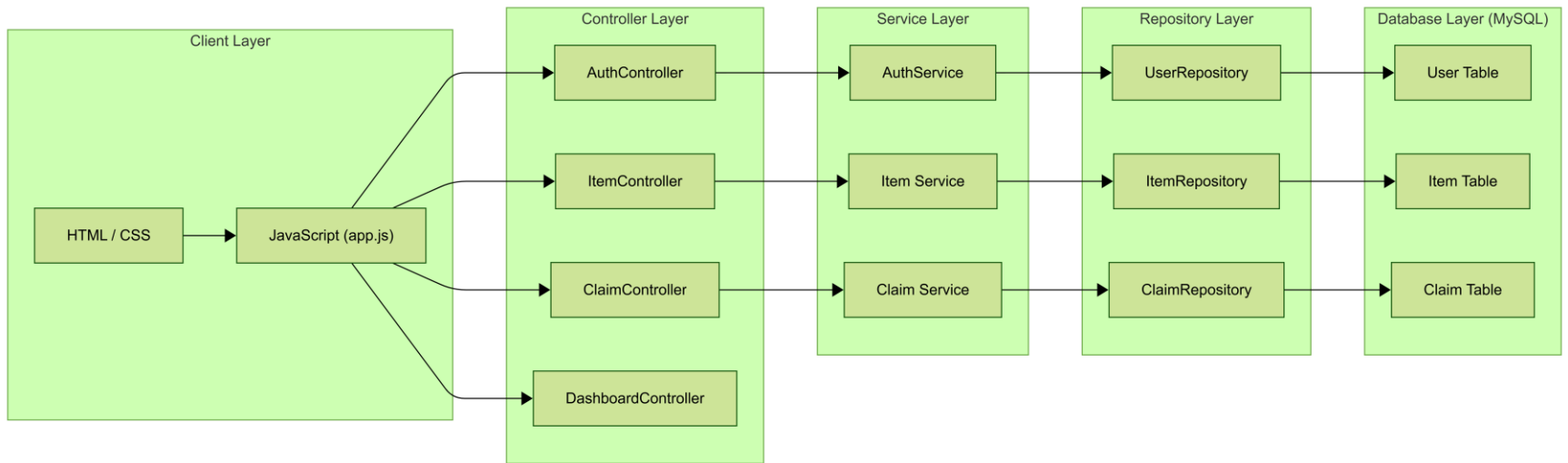
ER Diagram



Client-Server Architecture



Application Layers



Admin Module

- The **Admin Module** provides centralized control over the entire application.
- Admin can **view, resolve, approve, or reject** item claims.
- Admin has the authority to **delete inappropriate items or claims**.
- Admin can **block or manage users** to prevent misuse of the system.
- This module ensures **security, moderation, and smooth functioning** of the application.

User Module

- Manages **user profile details** and basic user operations.
- Allows users to **view and manage their information** after login.
- Connects users with the items they create or claim.
- Works with the authentication module for secure access.
- Stores and retrieves user data using **JPA repositories**.

Item Module

- Manages **lost and found item details** in the system.
- Allows users to **add, update, and delete item information**.
- Supports classification of items based on **category and type**.
- Tracks the **status of items** such as lost, found, or returned.
- Stores item data securely in the database using JPA.

Dashboard Module

- Provides a **summary view** of lost, found, and claimed items.
- Displays **counts and statistics** for quick understanding.
- Helps administrators **monitor system activity** efficiently.
- Presents consolidated information in a **user-friendly format**.
- Improves decision-making by giving a clear overview of data.

Claim Module

- Handles the process of **claiming lost or found items**.
- Allows users to submit a **claim request** for a specific item.
- Associates each claim with a **user and an item**.
- Enables tracking of claim **status** such as pending, approved, or rejected.
- Stores and manages claim information using JPA repositories.

Item Status Flow

Represents the **lifecycle of an item** in the system:

- **Lost** - Item is reported as missing by a user
- **Found** - Item is found and added to the system
- **Verified** - Claim details are checked and approved
- **Returned** - Item is successfully returned to the owner

Flow:

Lost → Found → Verified → Returned

Validation Handling

- Input validation is performed at the **controller level**
- Ensures required fields are **not null or empty**
- Prevents invalid or incomplete data from being processed
- Sends **proper error responses** for invalid requests
- Improves system reliability and data consistency

Security Handling

- User authentication handled through **login and registration**
- Passwords stored in **encrypted/hashed format**
- Restricted access to protected pages like **dashboard**
- Session-based validation to prevent unauthorized access
- Ensures data safety for users and items

Data Dictionary

- Defines the **structure of database tables** used in the system
- Describes important fields for **User, Item, and Claim** entities
- Helps understand how data is **stored and related**
- Improves clarity for developers and evaluators
- Acts as a reference for database design

User Table

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

reclaim_db

- Tables
 - claims
 - items
 - users
- Columns
- Indexes
- Foreign Keys
- Triggers

Views

- Stored Procedures
- Functions

sys

Administration Schemas

Information

Schema: reclaim_db

Object Info Session

SQL File 3* x

Limit to 1000 rows

```
1 CREATE DATABASE IF NOT EXISTS reclaim_db;
2 SELECT * FROM CLAIMS;
3 SELECT * FROM ITEMS;
4 SELECT * FROM USERS;
5
```

Result Grid

	id	email	name	password_hash
▶	1	abc@abc.com	abc	\$2a\$10\$1zpZXsvkFpfaDgzRE6a/5Oiof5XKWlmk...
	2	hi@hi.com	hi	\$2a\$10\$VyTOKS/hNs5ndk1mzzkAWeNk2A0Snu...
	3	hi@n.com	hi	\$2a\$10\$5w6TPwemxKtflb/8y8irWe87uK/qfq6T...
	4	shreya@gmail.com	shreya	\$2a\$10\$WjGEXyAZGfaltLE/ZOzW.mtk9Qz8L9...
	5	test@a.com	test	\$2a\$10\$PAlKf8e3HYUB5mRTvb94e03j.KoSgYG...
	6	test@gmail.com	test	\$2a\$10\$Jk/CIQFL4n7b1rt4kpgVxONBvUJzMSIzy...
	7	dudu@bubu.com	dudububu	\$2a\$10\$S4H2ZjotGNWl2xhRcPEYtUPuZHQq7gN...

USERS 3 x

Apply Revert Context Help Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	21:38:16	SELECT * FROM CLAIMS LIMIT 0, 1000	2 row(s) returned	0.015 sec / 0.000 sec
✓ 2	21:38:48	SELECT * FROM ITEMS LIMIT 0, 1000	4 row(s) returned	0.031 sec / 0.000 sec
✓ 3	21:39:17	SELECT * FROM USERS LIMIT 0, 1000	8 row(s) returned	0.016 sec / 0.000 sec

SQLAdditions

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

Item Table

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

reclaim_db

Tables

claims

items

Columns

Indexes

Foreign Keys

Triggers

users

Views

Stored Procedures

Functions

sys

Administration Schemas

Information

Column: **id**

Definition:

id bigint AI PK

Object Info Session

SQL File 3* x

Limit to 1000 rows

```
1 • CREATE DATABASE IF NOT EXISTS reclaim_db;
2 • SELECT * FROM CLAIMS;
3 • SELECT * FROM ITEMS;
```

Result Grid

	id	category	contact_method	contact_value	created_at	description	event_date	ima
▶	1	Document	EMAIL	nana@gmail.com	2025-12-16 12:02:19.838506	Found ID card named Sara	2025-12-16	/up
	2	Phone	PHONE	9876543210	2025-12-16 20:29:13.600911	Brand new orange iphone 17 pro found	2025-12-15	/up
	3	Keys	EMAIL	test@gmail.com	2025-12-16 20:43:22.520319	Bunch of keys found in floor	2025-12-16	NULL
	4	Toy	PHONE	9876543210	2025-12-16 20:56:13.501392	Found a bright pink flaming toy	2025-12-16	/up
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid

Form Editor

SQLAdditions

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

ITEMS 2 x

Apply Revert Context Help Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	21:38:16	SELECT * FROM CLAIMS LIMIT 0, 1000	2 row(s) returned	0.015 sec / 0.000 sec
✓ 2	21:38:48	SELECT * FROM ITEMS LIMIT 0, 1000	4 row(s) returned	0.031 sec / 0.000 sec

Claim Table

The screenshot displays the MySQL Workbench interface. On the left, the 'Navigator' pane shows the 'reclaim_db' database schema with tables 'claims', 'items', and 'users'. The 'claims' table is selected, showing its columns: 'id', 'created_at', 'status', 'claimant_id', and 'item_id'. The 'Information' pane at the bottom left shows the definition for the 'id' column: 'id bigint AI PK'. The main editor shows a SQL query: 'CREATE DATABASE IF NOT EXISTS reclaim_db;' followed by 'SELECT * FROM CLAIMS'. The 'Result Grid' displays the query results in a table with 5 columns and 3 rows. The first two rows contain data, and the third row is a summary row with 'NULL' values. The 'Output' pane at the bottom shows the execution of the query, indicating that 2 rows were returned in 0.015 seconds.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

reclaim_db

Tables

claims

Columns

id

created_at

status

claimant_id

item_id

Indexes

Foreign Keys

Triggers

items

users

Views

Stored Procedures

Functions

Administration Schemas

Information

Column: id

Definition:

id bigint AI PK

Object Info Session

SQL File 3*

Limit to 1000 rows

```
1 • CREATE DATABASE IF NOT EXISTS reclaim_db;
2 • SELECT * FROM CLAIMS
3
```

Result Grid

id	created_at	status	claimant_id	item_id
1	2025-12-16 20:30:50.874073	PENDING	5	1
2	2025-12-16 20:31:31.505266	PENDING	5	1
NULL	NULL	NULL	NULL	NULL

Form Editor

CLAIMS 1

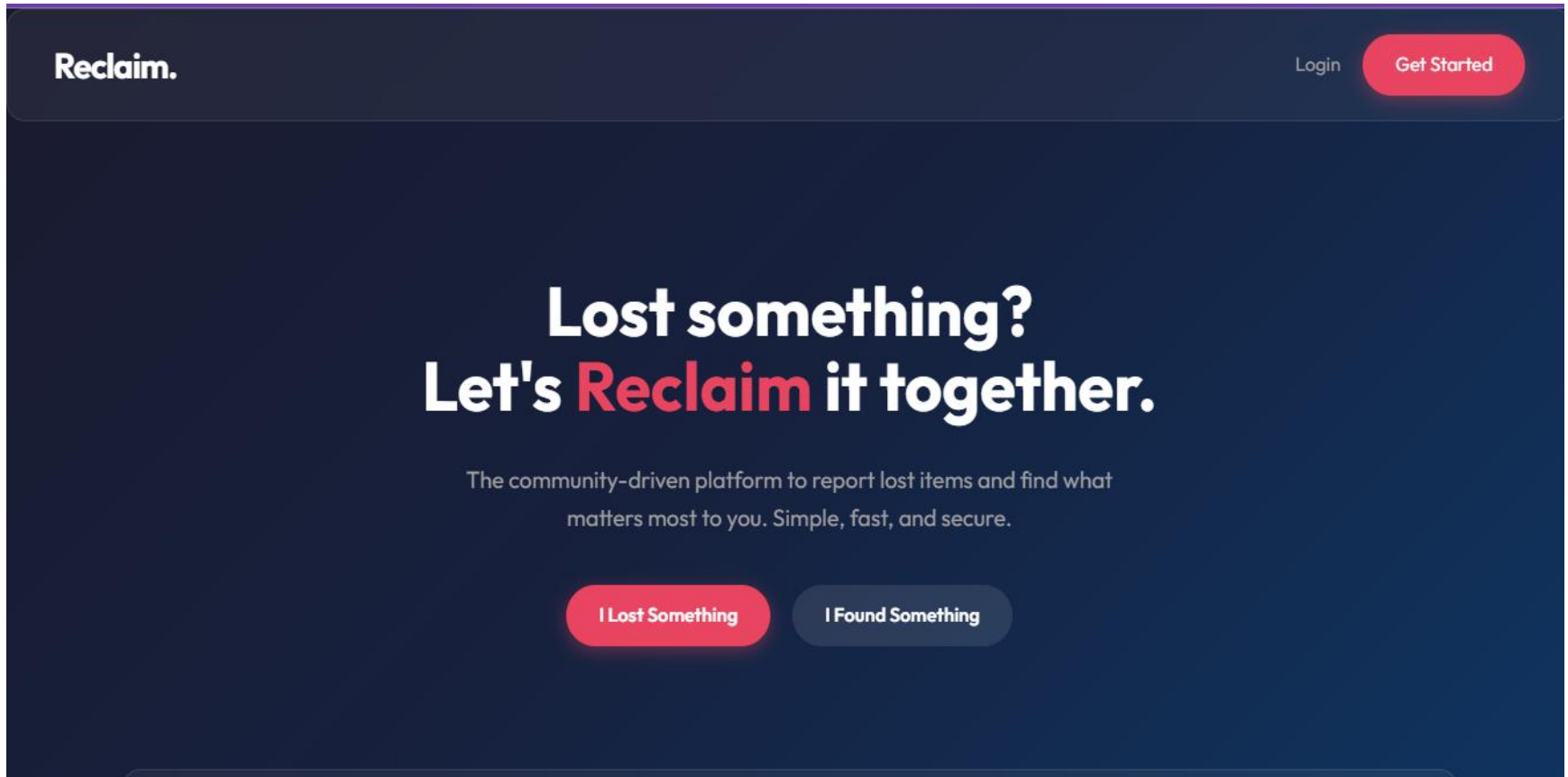
Apply Revert Context Help Snippets

Output

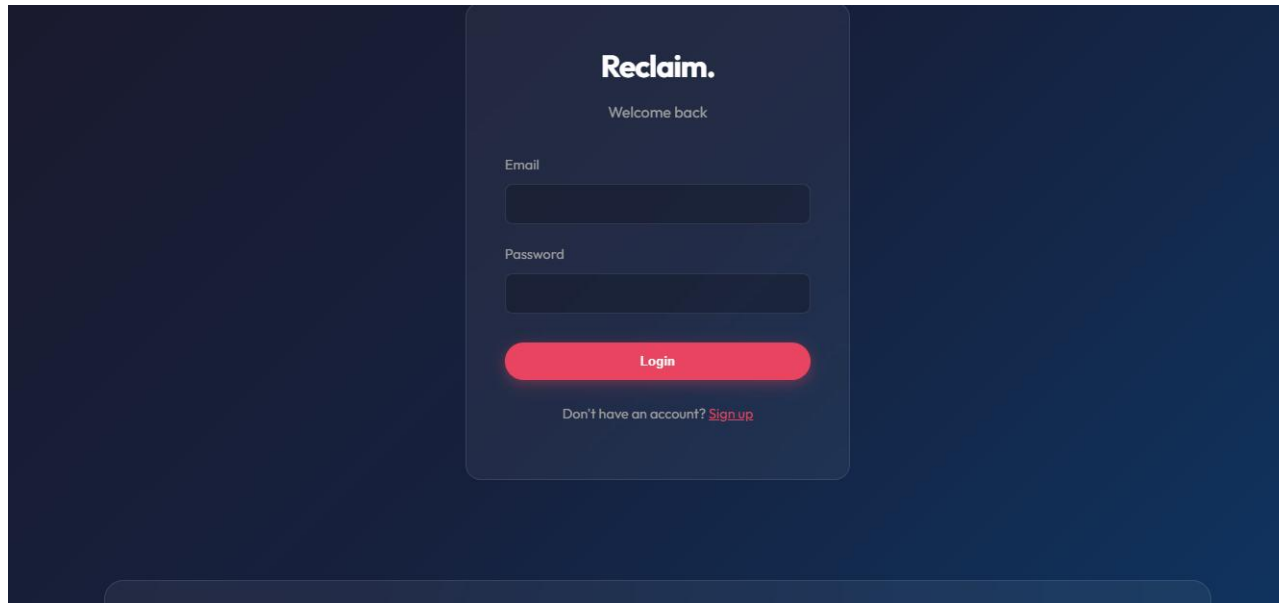
Action Output

#	Time	Action	Message	Duration / Fetch
1	21:38:16	SELECT * FROM CLAIMS LIMIT 0, 1000	2 row(s) returned	0.015 sec / 0.000 sec

User Interface - Home



User Interface – Login/Register



The login form is centered on a dark blue background. It features the 'Reclaim.' logo at the top, followed by the text 'Welcome back'. Below this are two input fields: 'Email' and 'Password'. A prominent red 'Login' button is positioned below the password field. At the bottom, there is a link that says 'Don't have an account? [Sign up](#)'.

Reclaim.

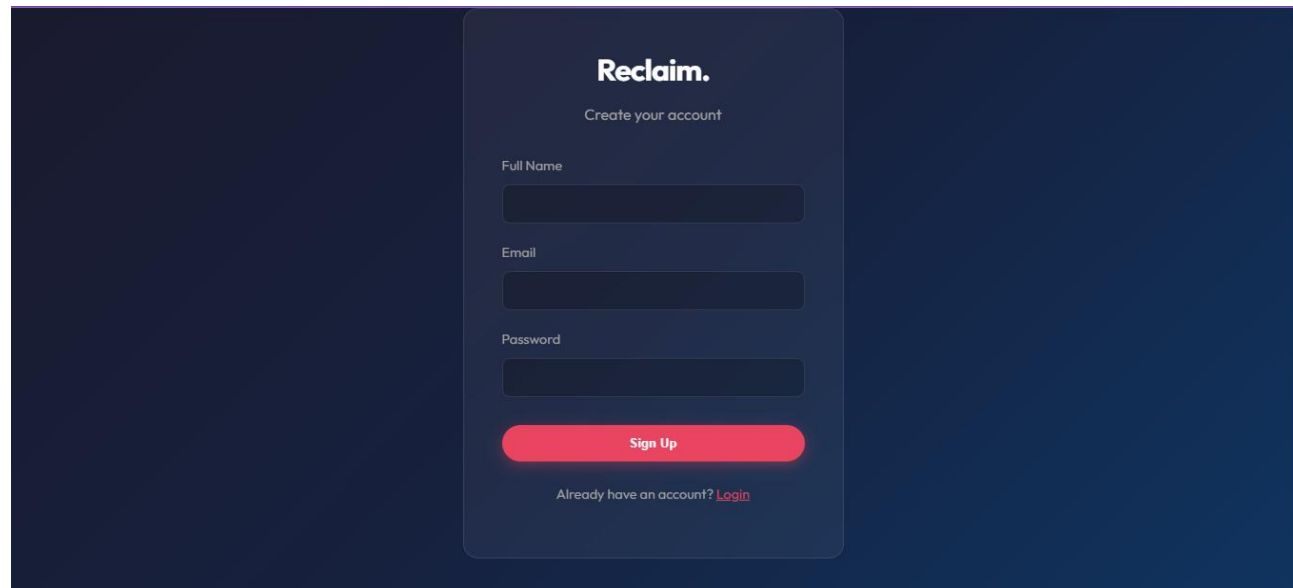
Welcome back

Email

Password

Login

Don't have an account? [Sign up](#)



The register form is centered on a dark blue background. It features the 'Reclaim.' logo at the top, followed by the text 'Create your account'. Below this are three input fields: 'Full Name', 'Email', and 'Password'. A prominent red 'Sign Up' button is positioned below the password field. At the bottom, there is a link that says 'Already have an account? [Login](#)'.

Reclaim.

Create your account

Full Name

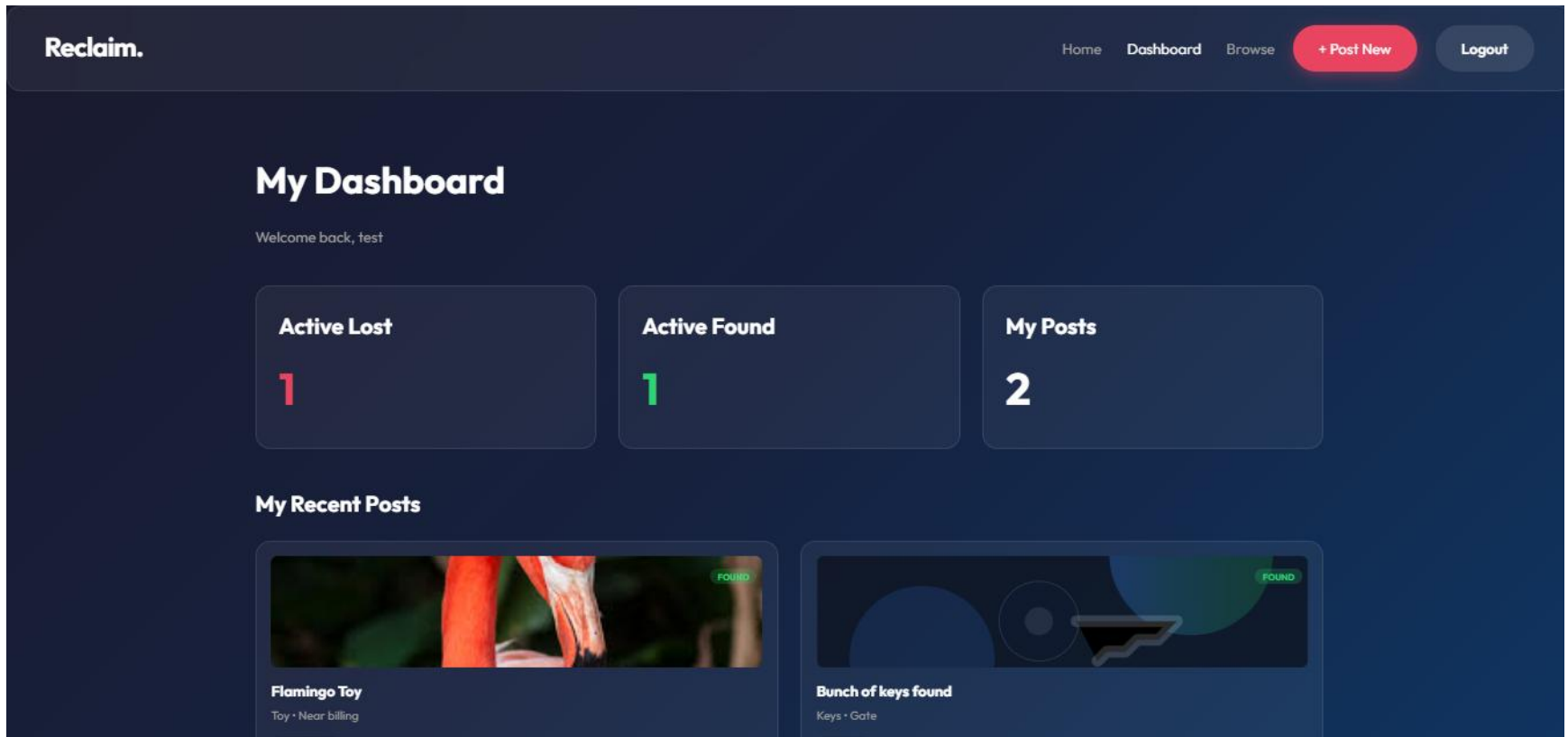
Email

Password

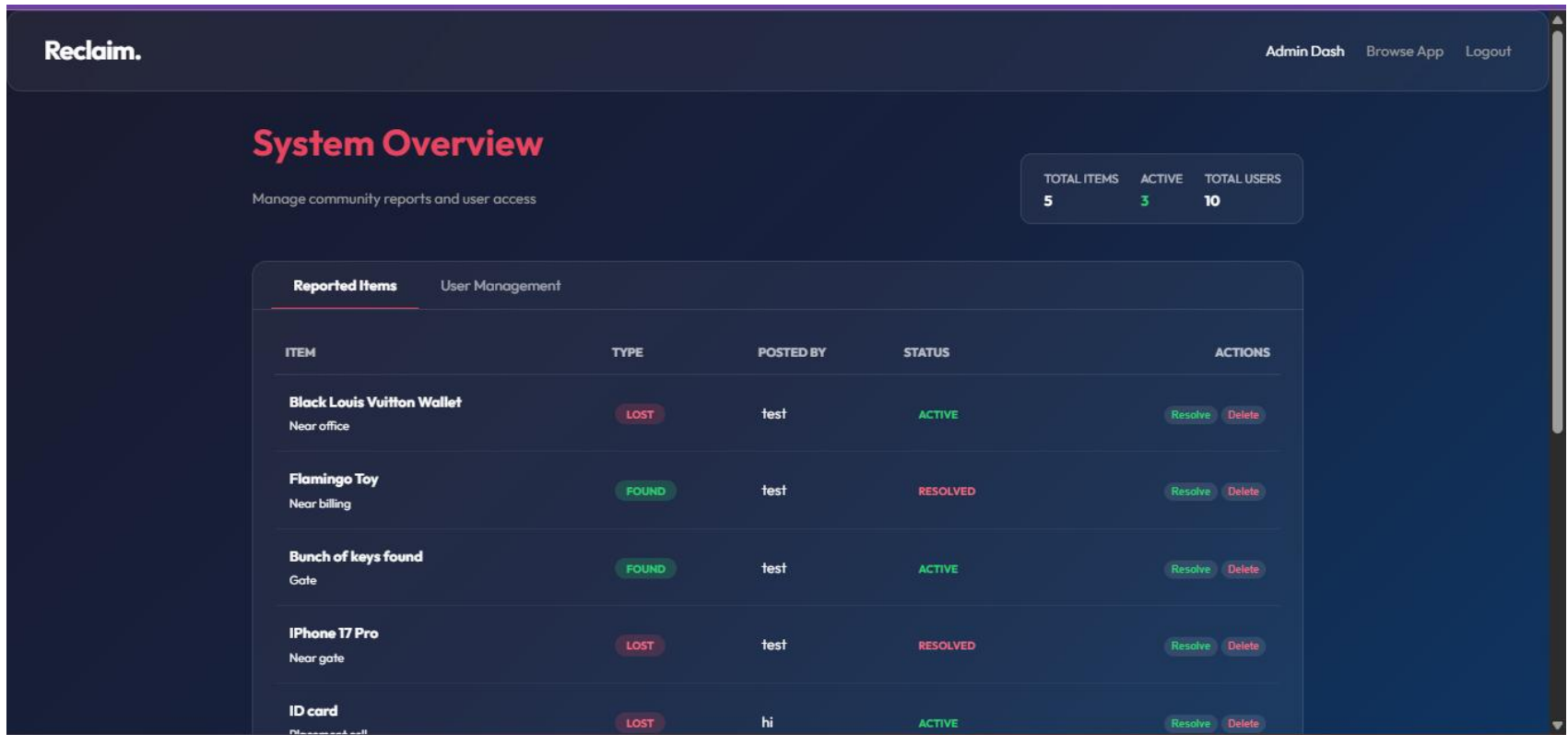
Sign Up

Already have an account? [Login](#)

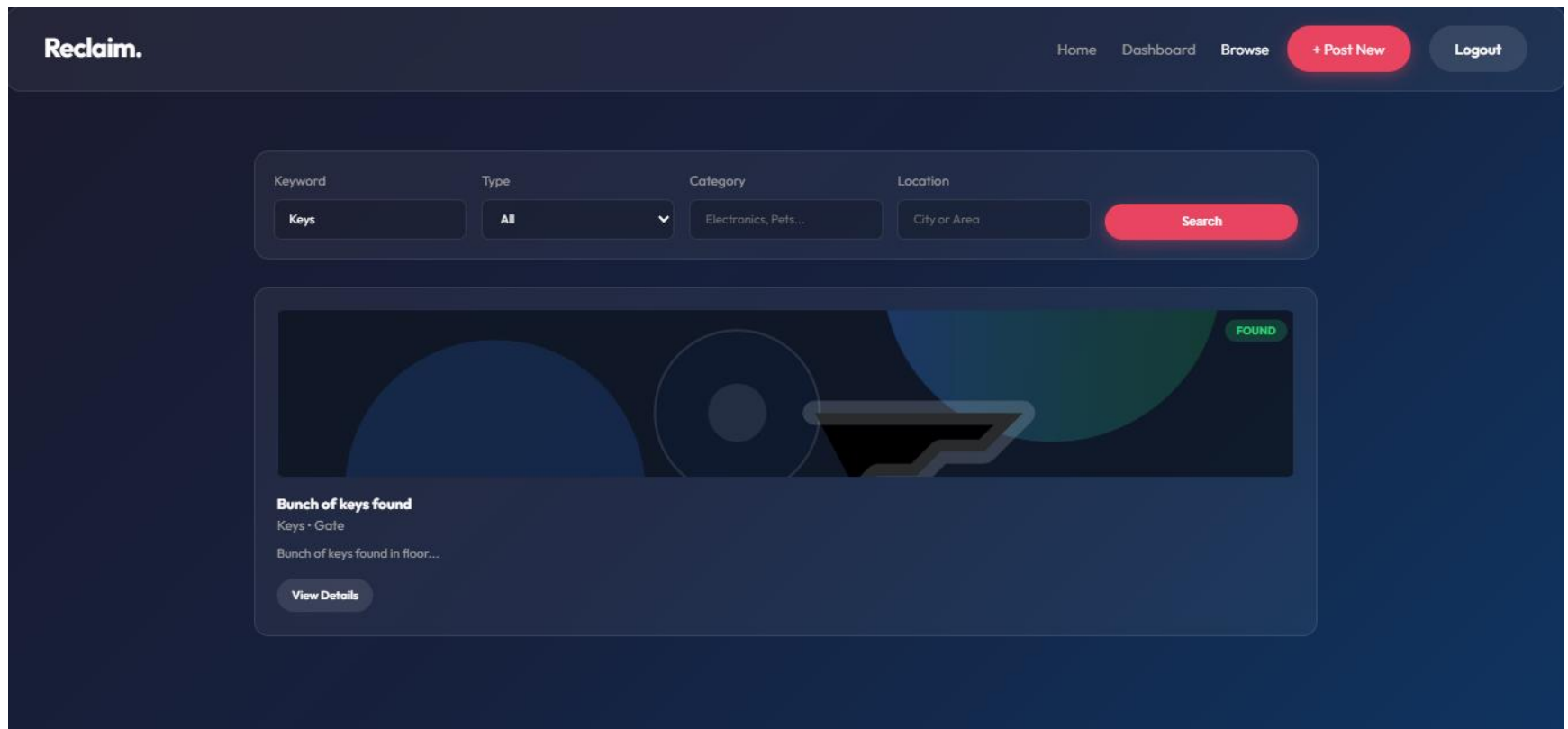
User Interface - Dashboard



User Interface - Admin Controls



User Interface - Browse Items



User Interface - Post Lost/Found

Post an Item

I...

Lost something

Lost something

Found something

Black Louis Vuitton Wallet

Category

Wallet

Description

Wallet with cash and cards

Location

Near office

Date

17-12-2025

Contact Method

Phone

Contact Details

9876543210

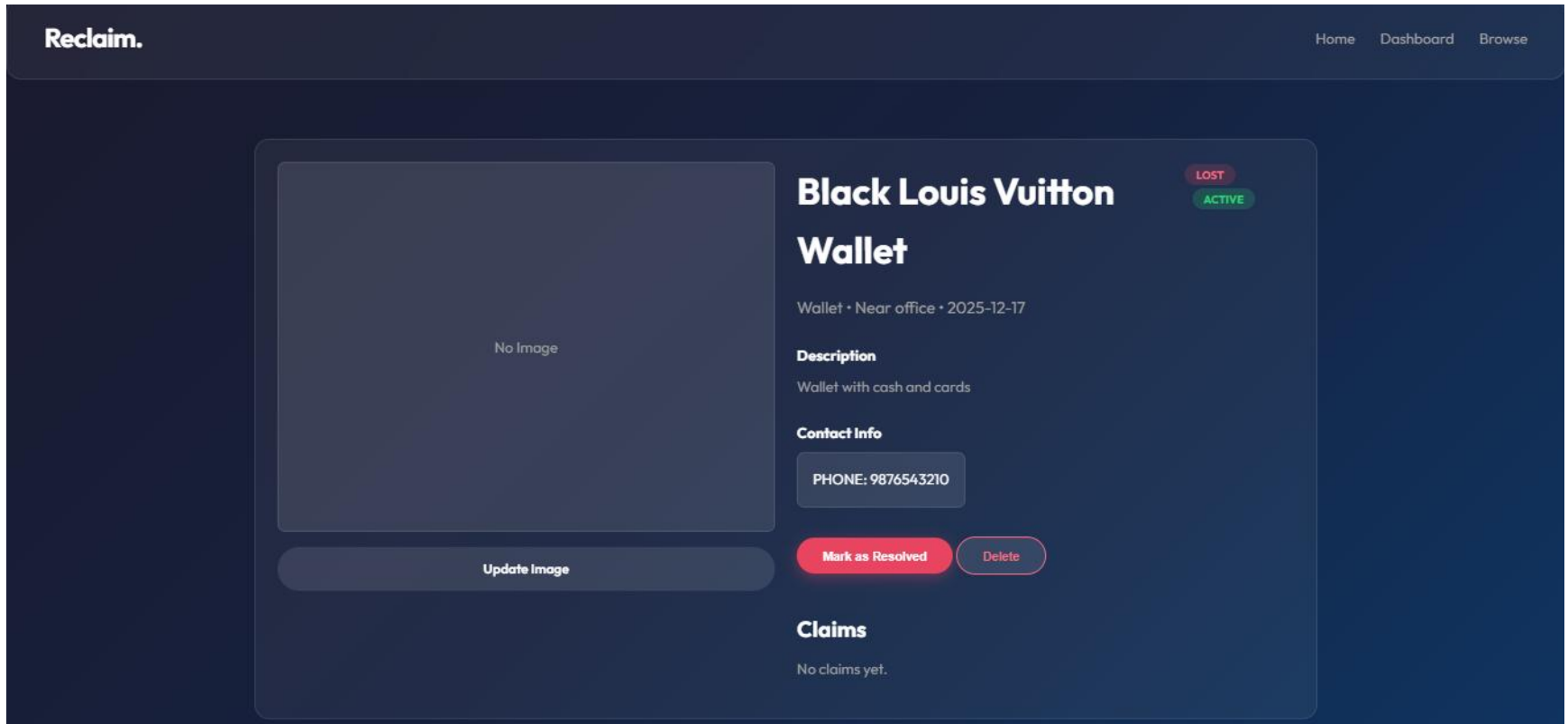
Upload Image (Optional)

Choose File

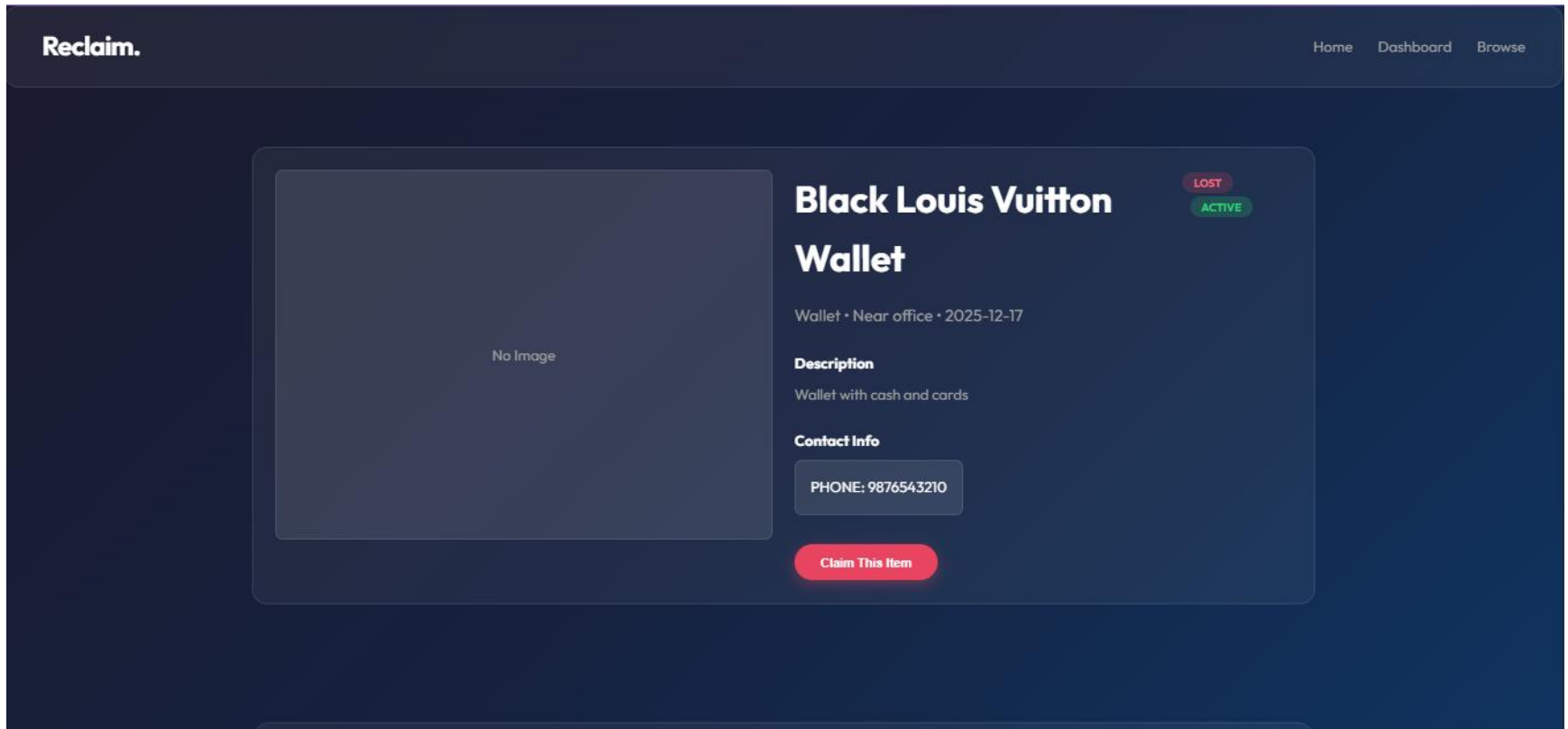
No file chosen

Submit Post

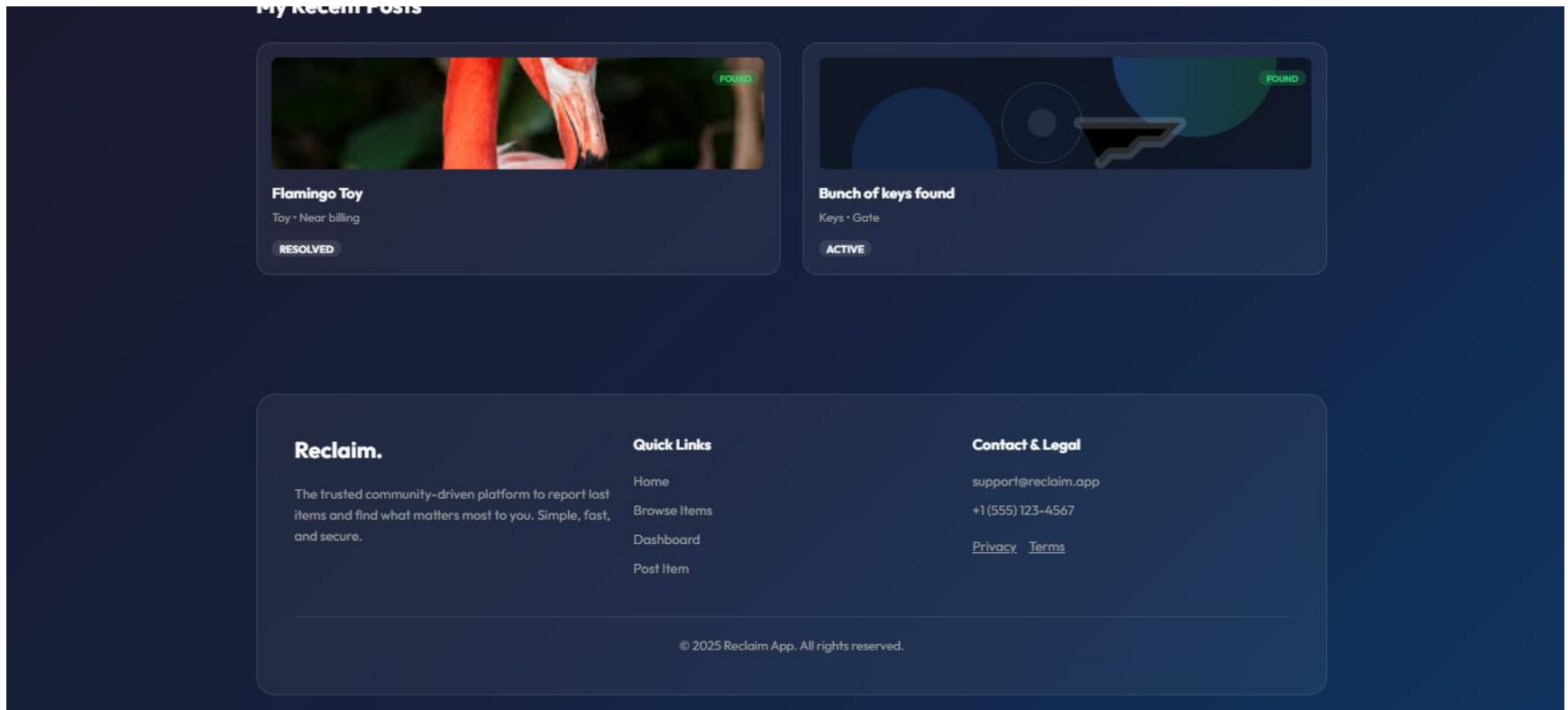
User Interface – User Controls View



User Interface – Non-login View



User Interface - Footer



Testing

- Manual testing performed for all UI pages
- API endpoints tested using browser
- Database operations verified using MySQL queries
- End-to-end flow tested for item posting and claiming
- Basic validation testing handled at controller level

Security

- Authentication implemented using **login and registration**
- User credentials are stored securely using **hashed passwords**
- Access to protected pages is restricted after login
- Basic authorization ensures only valid users perform actions
- Security logic handled at **controller and interceptor level**

Error Handling

- Centralized exception handling using Spring Boot
- Validation errors handled at controller level
- Meaningful error messages returned to the client
- Prevents application crashes due to invalid inputs

Limitations

- Manual verification required for claim approval
- No automated image or AI-based matching
- Web-based application only (no mobile app)
- Depends on user-provided item details
- Limited notification mechanisms

Future Enhancements

- Mobile application support (Android / iOS)
- Image-based item matching using AI
- Automated claim verification system
- Real-time notifications (email / SMS)
- Admin analytics dashboard

Conclusion

- Reclaim provides an efficient solution for lost and found management
- Simplifies item reporting and claim verification
- Ensures secure data handling using authentication
- Demonstrates full-stack development using Java Spring Boot
- Can be extended further with advanced features

Links

GitHub Link:

<https://github.com/Shreyasri30/Placement>

Demo Video Link:

<https://drive.google.com/file/d/182vKVCugOwUtNgbv2bWj3YMao1wwwD5G/view?usp=sharing>

Thank You