- **AIM:** Deploy full-stack apps using DevOps tools and Docker

- **THEORY**: Technologies used here -

1. DOCKER:

Docker is a **software platform** that simplifies the process of building, running, and managing applications by packaging them into **containers**. This containerization technology uses OS-level virtualization to deliver software in standardized units, which include everything an application needs to run: code, runtime, system tools, and libraries. This makes applications highly portable and ensures they run consistently across different environments, from a developer's local machine to a production server or cloud.

2. DEVOPS TOOL:

A DevOps tool is any software application, platform, or service that supports the **DevOps methodology**. DevOps is a set of cultural philosophies, practices, and tools that automates and integrates the processes between software development (Dev) and IT operations (Ops) teams.

The primary goal of DevOps tools is to break down the traditional "silos" between these teams, enabling them to work together more efficiently to deliver applications and services at a high velocity. This is achieved by automating manual tasks, improving communication and collaboration, and providing visibility into the entire software development lifecycle.
Its features includes:

- **Continuous Integration (CI):** Tools in this phase automatically merge code changes from multiple developers into a single repository, then build and test the new code. ● **Continuous Delivery/Deployment (CD):** These tools automate the process of preparing and deploying the application to testing and production environments.
- **Infrastructure as Code (IaC):** IaC tools allow teams to manage and provision their infrastructure (servers, databases, networks) using code, making the environment consistent and reproducible.
- **Monitoring and Logging:** Once an application is in production, monitoring and logging tools provide real-time visibility into its performance, helping teams quickly identify and resolve issues

HERE WE USED JENKINS AS A DEVOPS TOOL:

Jenkins is an open-source, server-based automation tool that is fundamental to the **DevOps methodology**. Its primary purpose is to automate the various stages of the software development process, particularly those related to **building, testing, and deploying software**.

By automating these tasks, Jenkins helps teams implement **Continuous Integration (CI)** and **Continuous Delivery/Deployment (CD)**, which are core practices of DevOps.

3. SUPABASE (DATABASE):

Supabase is a **Backend-as-a-Service (BaaS)** platform that provides developers with a full suite of tools to build a backend for their web, mobile, and AI applications without having to manage servers. It is often described as an open-source alternative to Firebase.

The key difference lies in its core philosophy: while Firebase uses a NoSQL, document-based database, Supabase is built around a **PostgreSQL relational database**. This means it leverages the power, reliability, and familiarity of a traditional SQL database, giving developers the ability to use complex queries, foreign keys, and a structured schema from the start.

- **SOURCE CODE:**

Jenkinsfile

```
pipeline {
    agent any

    stages {
        stage('Checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/ahujamonica/lost-found-portal.git'
            }
        }

        stage('Build Docker Image') {
            steps {
                bat 'docker-compose build'
            }
        }

        stage('Run Containers') {
            steps {
                echo 'Removing any existing container with a conflicting name...'
                // The || true makes this step not fail if the container doesn't exist
                bat 'docker rm -f lost-and-found-portal || true'
                echo 'Starting new containers...'
                bat 'docker-compose up -d'
            }
        }
    }
}
```

Dockerfile:

```
# Stage 1: Build the React app
FROM node:18-alpine AS builder

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

RUN npm run build
# Add this line to debug your build output
RUN ls -l /app/dist

# Stage 2: Serve the app with Nginx
FROM nginx:alpine

COPY --from=builder /app/dist /usr/share/nginx/html

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Docker-compose.yml:

```
services:
  frontend:
    build: .
    container_name: lost-and-found-portal # Use a fixed name for easier management
    ports:
      - "3000:80"  # This is the crucial line
    environment:
      - VITE_SUPABASE_URL=${VITE_SUPABASE_URL}
      - VITE_SUPABASE_ANON_KEY=${VITE_SUPABASE_ANON_KEY}
```

- **WORKING:**

Project Overview

This project demonstrates the successful deployment of a full-stack web application using a modern CI/CD pipeline. The application is a lost and found portal built with a modern frontend framework and a serverless backend. The entire process from code to deployment is automated using key DevOps tools.

The DevOps Pipeline

The pipeline follows a clear, automated workflow. First, code is written in VS Code and pushed to a GitHub repository. Jenkins automatically detects the new code and starts a build process.

Jenkins then uses Docker to containerize the application and its dependencies. Finally, Jenkins deploys the Docker container, making the application live.

Step 1: Docker & Containerization

Docker's role is to package the application into a single, portable container. This ensures a consistent environment across development and deployment stages.

- **Dockerfile**: The application is built using a multi-stage Dockerfile. It uses a node:18 image to build the application and a lightweight nginx:alpine image to serve the static files.
- **Docker Compose**: The docker-compose.yml file simplifies the process of building and running the container. The docker-compose up --build command was used to initialize the container.
- **Initial Container**: The Nginx server inside the container was able to successfully serve the application's main files, including index.html, index-cHvYlMjY.js, and index-NXh_Dpj-.css, which all returned 200 status codes.

Step 2: Supabase Integration

Supabase was used as a backend-as-a-service to handle the database and API functionality.

- **Database Schema**: A table named lost_items was created in the Supabase database. ● **Data Storage**: A new record for "car keys" was successfully added to the lost_items table. This confirmed the application's ability to connect to and store data in the Supabase database.

Step 3: Jenkins Automation

Jenkins orchestrated the entire CI/CD pipeline, automating the build and deployment process. The pipeline's behavior is defined in a Jenkinsfile.

- **Pipeline Stages**:
  - **Checkout**: Jenkins pulls the latest code from the GitHub repository.
  - **Build Docker Image**: Jenkins runs the docker-compose build command to create the Docker image of the application.
  - **Run Containers**: Jenkins first runs docker rm -f lost-and-found-portal || true to remove any existing containers, and then docker-compose up -d to start a new container.
- **Successful Build**: The Jenkins pipeline completed successfully with a "Finished: SUCCESS" status.

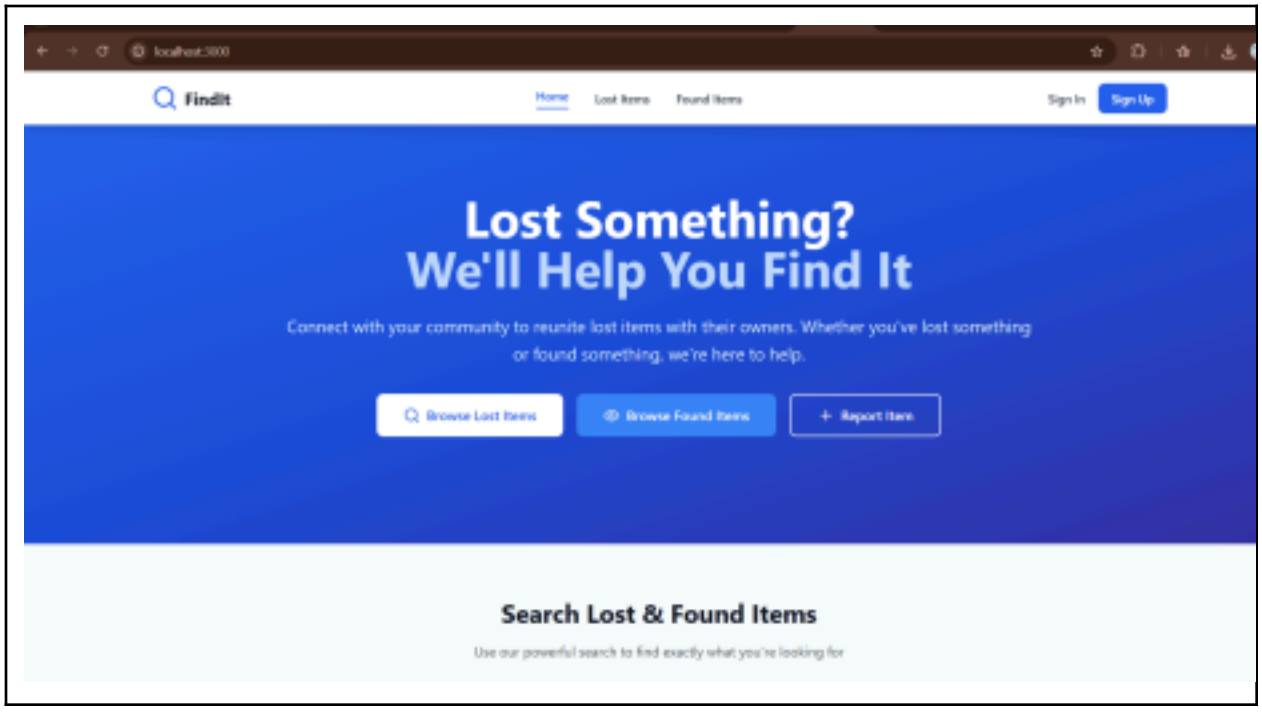
- **OUTPUT:**

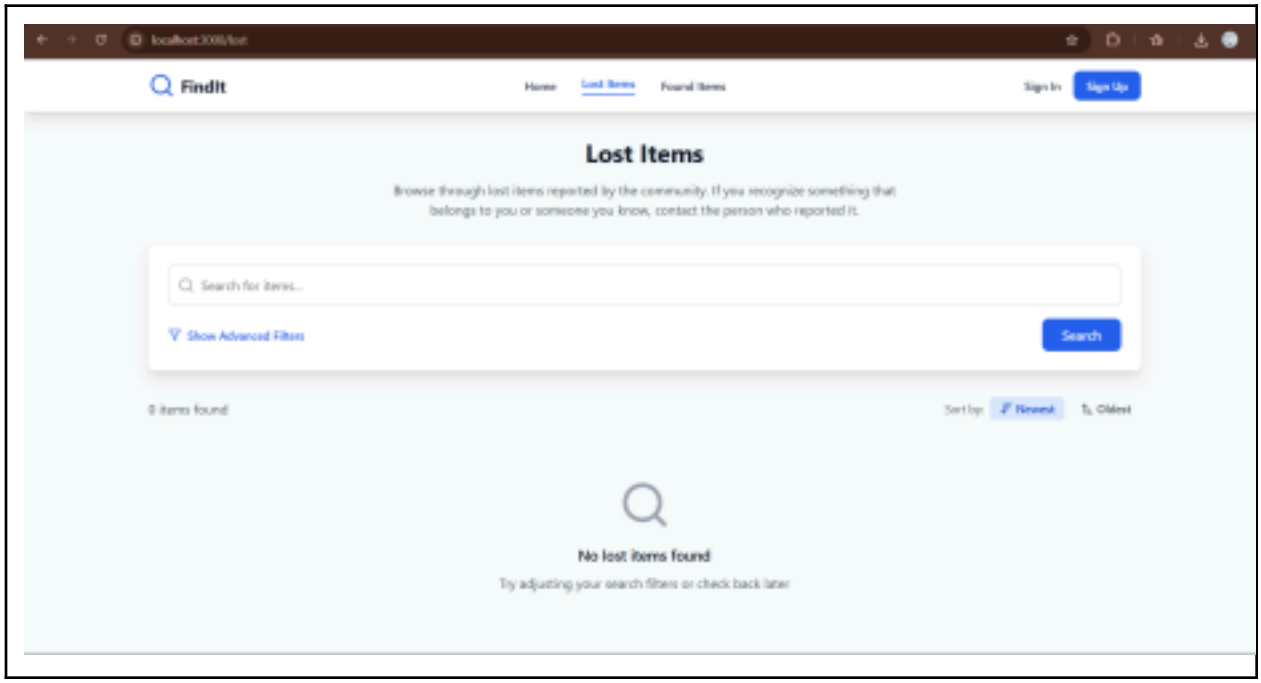
INITIALIZING CONTAINER AT VSC FOR OUR WEBSITE IN DOCKER:

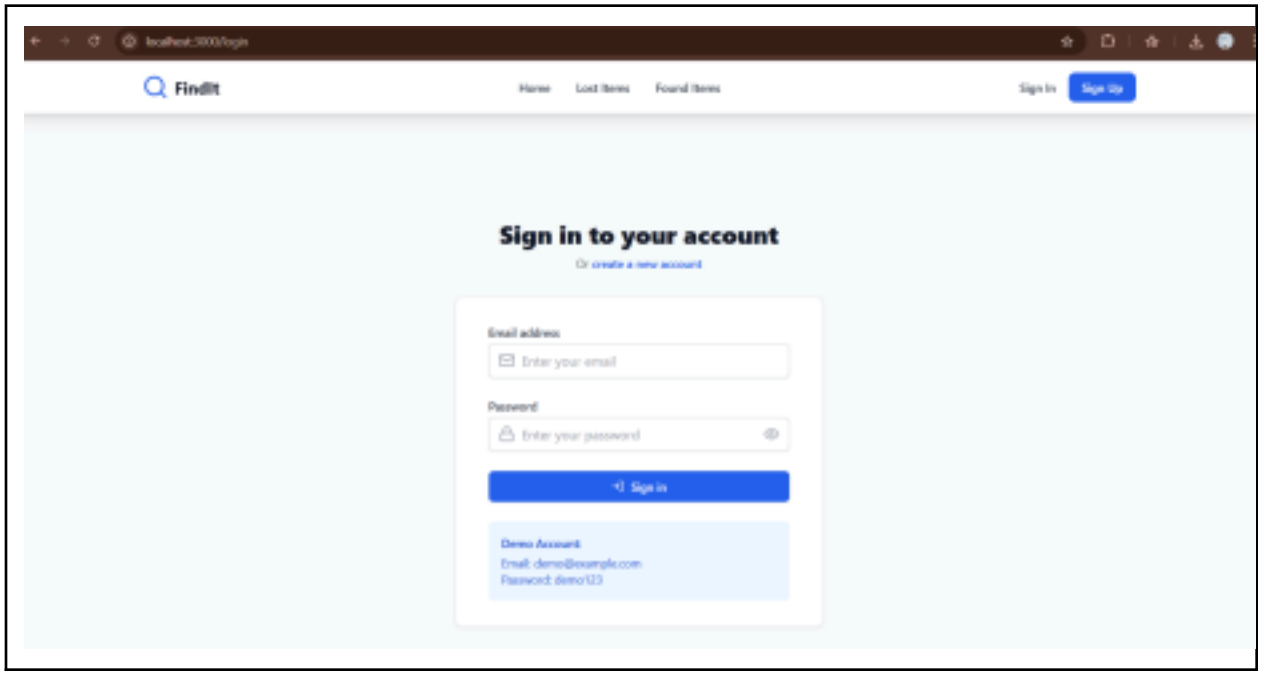
```
PS C:\Users\Monica\Downloads\lostFound\project> docker-compose up --build
lost-and-found-portal  | 2025/09/23 18:28:22 [notice] 1#1: start worker process 39
lost-and-found-portal  | 2025/09/23 18:28:22 [notice] 1#1: start worker process 40
lost-and-found-portal  | 2025/09/23 18:28:22 [notice] 1#1: start worker process 41
lost-and-found-portal  | 172.18.0.1 - - [23/Sep/2025:18:30:13 +0000] "GET / HTTP/1.1" 200 485 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Ap
pleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36" "-"
lost-and-found-portal  | 172.18.0.1 - - [23/Sep/2025:18:30:13 +0000] "GET /assets/index-N0h_Opj-.css HTTP/1.1" 200 22911 "http://localhost:3000
/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36" "-"
lost-and-found-portal  | 172.18.0.1 - - [23/Sep/2025:18:30:13 +0000] "GET /assets/index-gc-EcoeM.js HTTP/1.1" 200 484964 "http://localhost:3000
/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36" "-"
lost-and-found-portal  | 172.18.0.1 - - [23/Sep/2025:18:30:13 +0000] "GET /vite.svg HTTP/1.1" 404 555 "http://localhost:3000/" "Mozilla/5.0 (Wi
ndows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36" "-"
lost-and-found-portal  | 2025/09/23 18:30:13 [error] 31#31: *2 open() "/usr/share/nginx/html/vite.svg" failed (2: No such file or directory), c
lient: 172.18.0.1, server: localhost, request: "GET /vite.svg HTTP/1.1", host: "localhost:3000", referrer: "http://localhost:3000/"
```

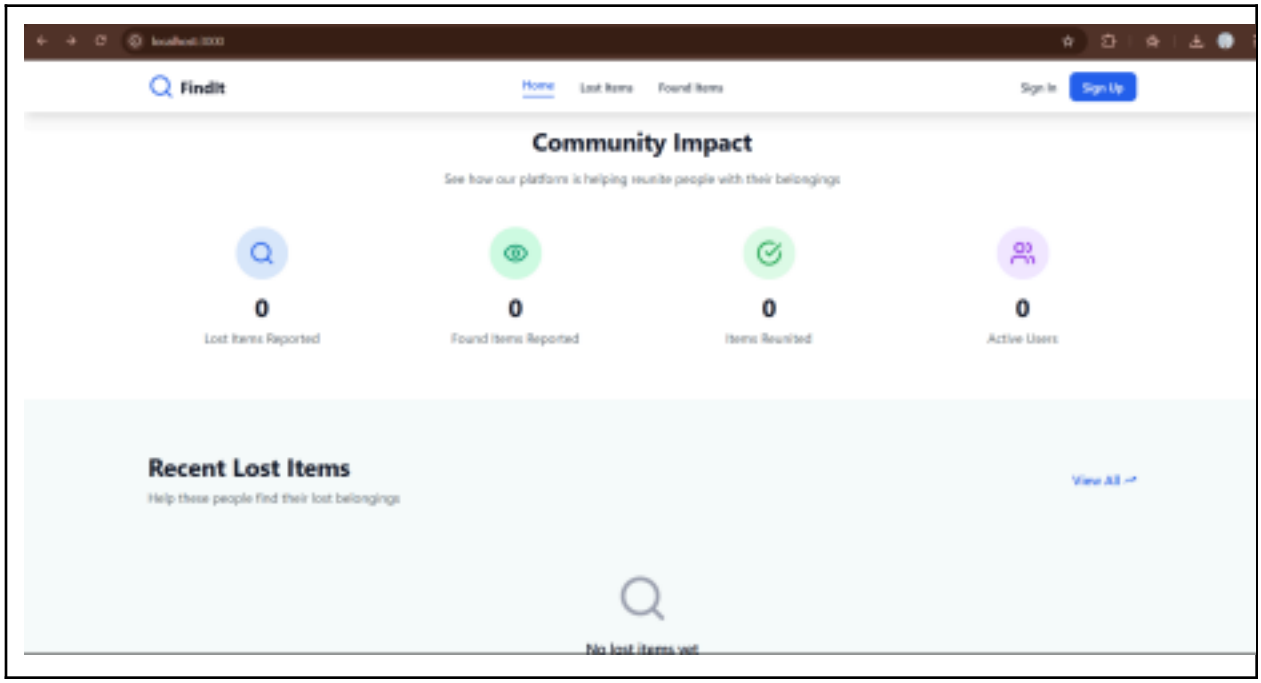
View in Docker Desktop View Config Enable Watch

DOCKER LOCALLY HOSTED WEBSITE WHEN JENKINS BUILD WAS SUCCESSFUL:

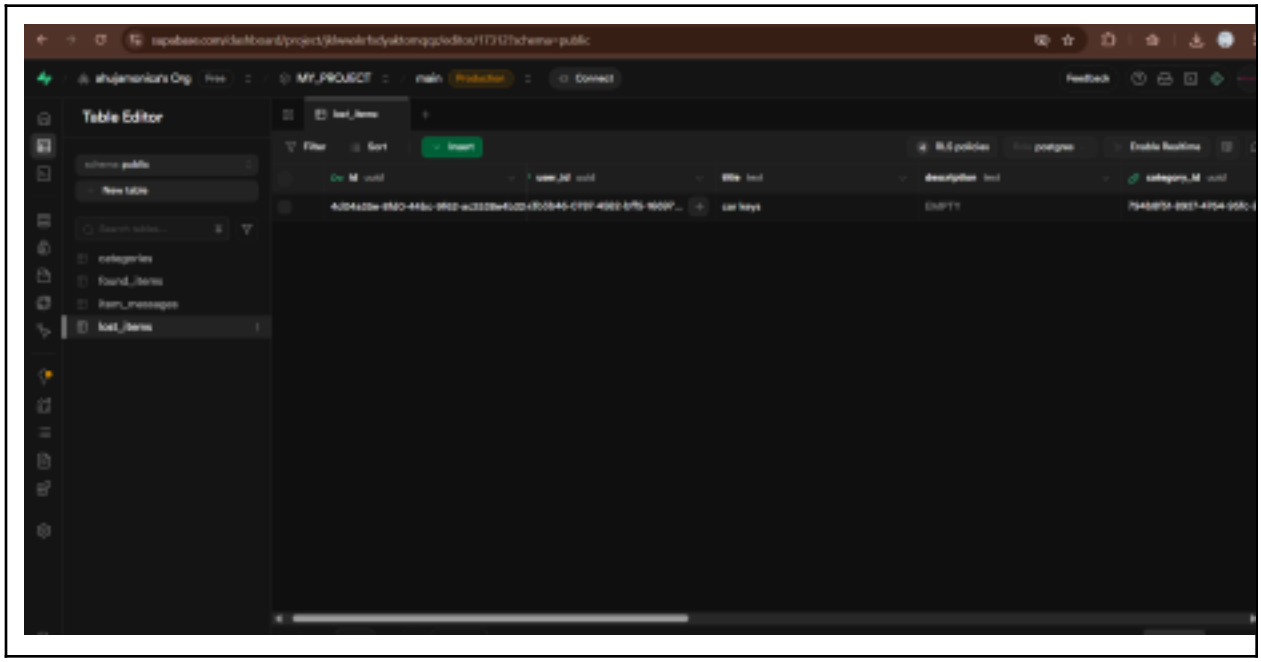



FindIt
Home
Lost Items
Found Items
Sign In
Sign Up
Lost Items
Browse through lost items reported by the community. If you recognize something that belongs to you or someone you know, contact the person who reported it.
Search for items...
Show Advanced Filters
Search
0 items found
Sort by:
Newest
Oldest
No lost items found
Try adjusting your search filters or check back later

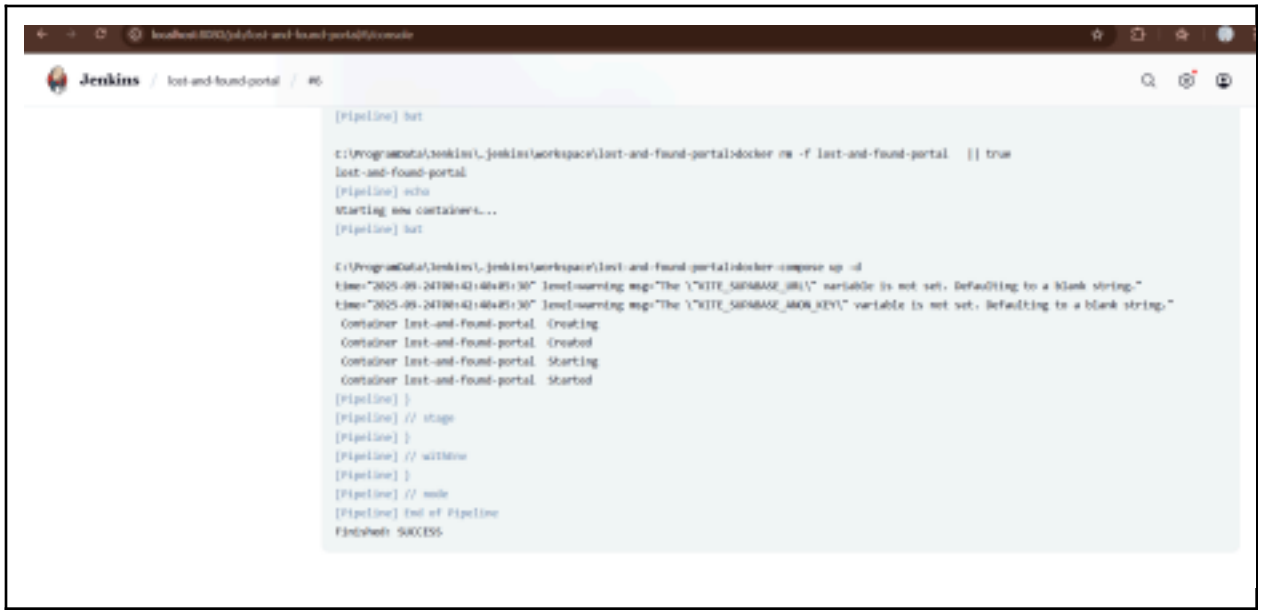

FindIt
Home
Lost Items
Found Items
Sign In
Sign Up
Sign in to your account
Or create a new account
Email address
Enter your email
Password
Enter your password
Sign in
Demo Account
Email: demo@example.com
Password: demo123



SUPABASE DATABASE CONNECTIVITY SHOWING THE ADDED LOST ITEM:



JENKINS OUTPUT:



- **Conclusion**

The project successfully demonstrates an end-to-end DevOps pipeline.

- **Docker** provided a consistent environment for the application.
- **Jenkins** automated the build and deployment process, from a code commit to a running application.
- The application is fully functional and accessible at http://localhost:3000. It correctly displays pages and connects to the Supabase database for data management