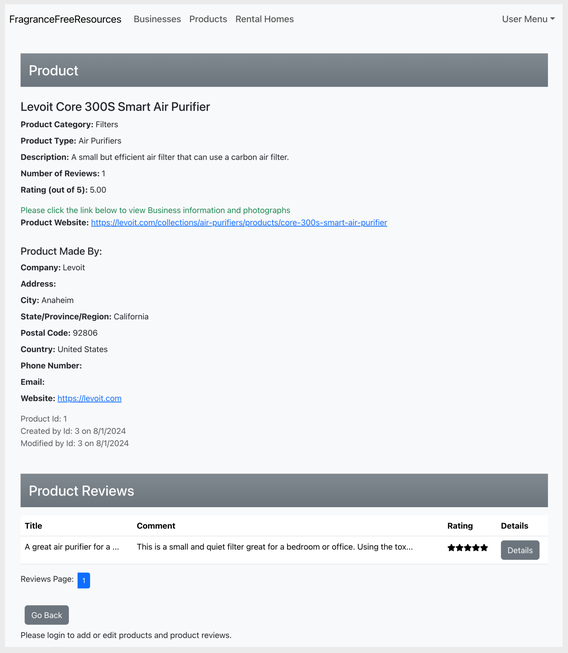
COLLEGE OF ENGINEERING

Electrical Engineering and Computer Science

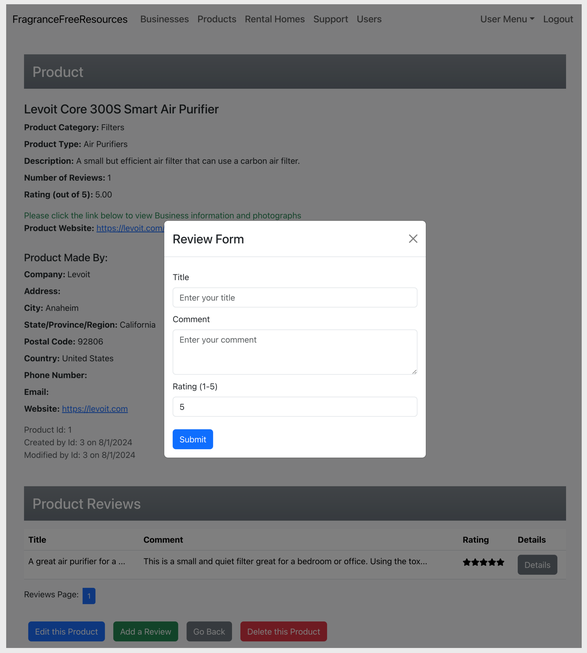
www.FragranceFreeResources.com

Joel Strong
406 Personal Project

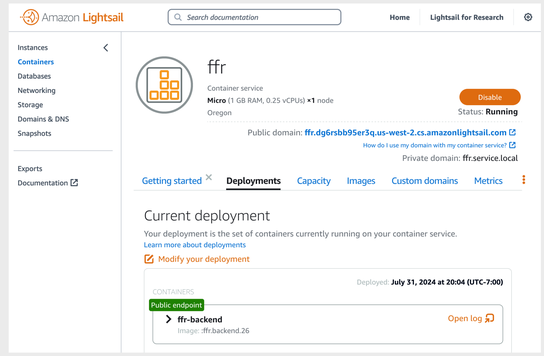
A full stack web application for people to share their knowledge and experiences about Fragrance Free businesses, products, and rental homes.



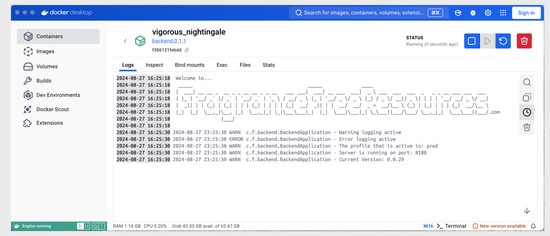
Visitor Product View



Administrator Product View with Review Add Modal



Amazon Lightsail Container Hosting



Containerized Spring Boot Application

MOTIVATION

One of my family members is affected by a severe sensitivity to fragrances and struggles with isolation having to avoid scented environments. There is large international community people with similar struggles, and I wanted to create a resource for them to share their individual knowledge of safe businesses, products, and rental homes with the community. My hope is that this web application will be a resource to improve the quality of life for people that have a sensitivity to fragrances.

LEARNING GOALS

This project was intended to coalesce my Computer Science education to build a more advanced full stack MVC web application. For this 10-week project, I had the following learning goals:

- Implement Spring Security, Users, Sessions, and JWT usage in React.
- Deploy and maintain a containerized Spring Boot application to AWS
- Learn the Typescript language
- Learn and implement advanced React concepts such as Component architecture, useContext, useReducer, and other tools.
- Deploy a usable application to the public.

RETROSPECTIVE

I was able to accomplish all the goals of the project except my desired deployment timeline. Implementation of an international solution compliant with GDPR, support-moderation features, and nested Product to Company relationships were desired, but non-critical features. I learned the valuable lesson of focusing on producing the Minimum Viable Product first, before extending the application with additional features. This project was an invaluable learning opportunity, and I am grateful to Professor Pam Van Londen for giving me this opportunity and being my project sponsor.

To see more examples of my work, skills, certifications, and resume visit my personal website at www.jdstrongpdx.com.

Oregon State University

FRONTEND

Overview:
- Development of a SPA/PWA frontend using React, Typescript, and React-Bootstrap. Hosted on AWS Amplify.

React Features:
- Component architecture to break the application into functional pieces.
- Conditional rendering based on User/Entity data to generate NavBar features, buttons, and page text.
- Employed useContext to cache application wide data.
- Utilized Effects, Reducers, and Fetch to load and track changes to data.
- Used Typescript to enforce strict naming and typing consistent with backend entities.
- Implemented development and production environment configurations.

Bootstrap Features:
- Use of a collapsable NavBar for compatibility across screen sizes.
- Consistent page layouts, color themes, and fields for uniformity.
- Intuitive NavBar, Buttons, and Links to view and access information.
- Using Modals for entering entity reviews or confirming delete actions.
- Providing user messages using non-obtrusive Touch alerts.
- Efficient page design and handling for user Signup, Login and Logout with auto logout on token expiration.
- Implementation of Paging to limit the number of responses per page, improve latency, and decrease server load.
- Showing ratings in a familiar five-star rating system.

BACKEND

Overview:
- Development of a RESTful API backend using Spring Boot, Java, and Docker.
- Hosted using AWS Lightsail Container and deployed using Docker and AWS CLI.

Features:
- Used Spring Web to implement RESTful APIs for 40 API endpoints on 9 entities.
- Implemented Spring Security for user roles, authentication, authorization, and sessions using JWT. Created security configurations for access permissions at each endpoint.
- Utilized Spring Data JPA to perform database transactions and paginated returns.
- Created custom DTO classes to return only necessary information from endpoints.
- Implemented development and production configurations to easy switch between local test and remote production environments.
- Testing performed using Postman and Newman collections on API endpoints.
- Used Maven to manage dependencies, test, and build the Java/Spring Boot application.
- Employed Docker to containerize, test, and deploy the application.

DATABASE

Overview:
- Utilized MySQL for data storage and MySQL Workbench for administration and queries.
- Used a local server for development and testing activities.
- Deployed a remote server hosted by AWS Lightsail Database for production data.
- Tables and fields automatically generated by Spring Data JPA using entity annotations.