

Florida International University
Knight Foundation School of Computing and Information Sciences

Software Engineering Focus

Final Deliverable

Project Title:

Sockify

Team Members: Abel Aguilera, Bora Dibra, Charlotte Williams, Sebastian Nunez

Product Owner(s): Sebastian Nunez

Mentor(s):

Instructor: Masoud Sadjadi

The MIT License (MIT)
Copyright (c) 2016 Florida International University

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Abstract

Sockify is a full-stack e-commerce platform designed to enable users to browse, customize, and purchase socks. The platform delivers a seamless user experience by integrating a modern React-based frontend with a robust Go-powered backend, making it scalable and efficient.

The frontend leverages TypeScript, React (Vite), ShadCN UI, TailwindCSS, and tools like React Query and Axios for data fetching and caching. Form validation is handled with React Hook Form and Zod, while React Hot Toast provides interactive notifications. The backend incorporates PostgreSQL for database management, JWT for secure authentication, Stripe for payment processing, and Firebase for blob storage. API documentation and specifications are managed using OpenAPI (Swagger), and SendGrid powers email notifications.

The project is designed to simplify inventory management for administrators while ensuring a streamlined shopping experience for users. With hosting solutions like Railway and containerization using Docker Compose, Sockify is designed to be highly scalable, reliable, and developer-friendly. This document outlines the system design, features, and implementation details of the platform.

Table of Contents

INTRODUCTION5

 CURRENT SYSTEM5

 PURPOSE OF NEW SYSTEM 5

USER STORIES

 IMPLEMENTED USER STORIES5

 PENDING USER STORIES 6

PROJECT PLAN

 HARDWARE AND SOFTWARE RESOURCES 8

 SPRINTS PLAN 9

Sprint 1 10

Sprint 2 10

Sprint 3 10

Sprint 4 11

Sprint 5 12

Sprint 6 13

Sprint 7 13

SYSTEM DESIGN

 ARCHITECTURAL PATTERNS 14

 SYSTEM AND SUBSYSTEM DECOMPOSITION 15

 DEPLOYMENT DIAGRAM 17

 DESIGN PATTERNS 19

SYSTEM VALIDATION 20

GLOSSARY 20

APPENDIX 21

 APPENDIX A - UML DIAGRAMS 21

 APPENDIX B - USER INTERFACE DESIGN21

 APPENDIX C - SPRINT REVIEW REPORTS 21

 APPENDIX D - USER MANUALS, INSTALLATION/MAINTENANCE DOCUMENT, SHORTCOMINGS/WISHLIST DOCUMENT AND OTHER DOCUMENTS 22

INTRODUCTION

Current System

Currently, no existing system serves the niche for customizable sock products integrated into a streamlined e-commerce experience.

Purpose of New System

The Sockify system aims to provide an accessible and user-friendly platform to enable:

- Efficient browsing and purchasing of socks.
- Inventory management for administrators.
- Seamless scaling for additional product lines.

USER STORIES

The following section provides the detailed user stories that were implemented in this iteration of the project. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

Implemented User Stories

Admin

- As an admin, I want to be able to login into the dashboard with a username and password, so that I can manage the store.
- As an admin, I want to see and browse all of the products in the store, so that I can better keep up with the stock.
- As an admin, I want to add, remove, and update items, so that I can maintain the latest and greatest in the store.
- As an admin, I want to see all orders, so that I can ensure we are able to fulfill them.

- As an admin, I want to update the status of all orders (e.g. pending -> shipped -> delivered), so that I can ensure customers receive their orders.
- (Stretch) As an admin, I want to be able to login using MFA (2-factor authentication), so that I can ensure the store is secure from bad actors.
- (Stretch) As an admin, I want to have enhanced filtering/search capabilities, so that I can look up orders quickly by customer name or invoice number for example.

Customer

- As a customer, I want to browse all socks available within the store, so that I can decide what to purchase.
- As a customer, I want to see the details (available sizes, price, etc.) of an individual/particular item, so that I can make an informed purchase decision.
- As a customer, I want to have the ability to add items to my cart, so that I can keep my order together.
- As a customer, I want to see all of the items in my cart, so that I can review my items before purchasing.
- As a customer, I want to update the items in my cart (remove, update quantity), so that I can ensure I purchase the right items.
- As a customer, I want to be able to check out/purchase the items in my cart (enter payment and shipping info, etc.), so that I can get my items.
- As a customer, I want to be able to pay using Stripe, so that I can ensure my payment information is safeguarded.
- (Stretch) As a customer, I would like to receive a confirmation email when I place an order, so that I can verify my purchase(s).

Developer

- As a developer, I want to learn and become proficient in Go, TypeScript, and PostgreSQL so that I can contribute to the team and work on the project.
- As a developer, I want to make sure the API endpoints that could return thousands of results are paginated (e.g. /socks), so that I ensure the UI is smooth and will not slow down.

Pending User Stories

- (Stretch) As a customer, I would like to receive an email when my order status is updated, so that I can track the progress of my order(s).
- (Stretch) As a customer, I want to have filtering/search capabilities when browsing all items, so that I can find the items I am looking for quicker.

- (Stretch) As a customer, I want to be able to create an account (and login) to the store, so that I can see all my previous orders and track the progress.
- Developer
- (Stretch) As a developer, I want to make sure when purchasing items we do not sell more stock than what we have, so that customers' orders can always be fulfilled.

PROJECT PLAN

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

- Frontend
 - Languages: [TypeScript](#)
 - Frameworks: [React](#) (w/ [Vite](#))
 - UI components: [ShadCN UI](#), [TailwindCSS](#)
 - Icons: [Lucide Icons](#)
 - Data fetching/caching: [React Query](#), [Axios](#)
 - Form validation: [React Hook Form](#), [Zod](#)
 - Notifications/toasts: [React Hot Toast](#)
- Backend:
 - Languages: [Go](#)
 - Authentication: [JSON Web Tokens \(JWT\)](#)
 - Payment processing: [Stripe](#)
 - Blob storage: [Firebase](#)
 - API Specification (UI): [OpenAPI \(Swagger\)](#)
 - Email client: [SendGrid](#)
- Database: [PostgreSQL](#)
- Hosting: [Railway](#), [Docker Compose](#)
- Design: [Figma](#)

Sprints Plan

Sprint 1

1. [\[DB\] Create the `order\_updates` table to store updates made by admins for customer orders.](#)
2. [\[DB\] Create the `orders` table to track customer purchases.](#)
3. [\[DB\] Create the `order\_items` table to link ordered items to orders and `sock\_variants`.](#)
4. [Create initial GitHub organization and README with docs](#)
5. [Develop core project milestones](#)
6. [Create pull request and backlog item templates](#)
7. [\[DB\] Create the `admins` table for admin user information.](#)
8. [\[DB\] Create the `sock\_variants` table to handle sock sizes, prices, and stock levels.](#)
9. [\[DB\] Create the `socks` table for storing product information.](#)
10. [Create the main tasks for the project](#)
11. [Create main MVP CUIs \(user stories\)](#)
12. [Add Swagger UI to the API](#)
13. [Create the initial project scaffold](#)
14. [Create database schema and ER diagram](#)
15. [Create a folder/directory structure for the repo](#)
16. [Move MVP project tasks into the board](#)

Sprint 2

1. [\[API\] Create POST /admins/register endpoint to create new admin credentials](#)
2. [\[API\] Create PATCH /socks/:sock_id to update product details](#)
3. [\[API\] Create GET /socks/:id endpoint to return detailed info on a specific sock](#)
4. [\[API\] Create GET /orders to list all orders](#)
5. [\[API\] Create DELETE /socks/:sock_id to remove a product](#)
6. [\[API\] Create GET /socks endpoint to fetch all products](#)
7. [\[API\] Create POST /socks to add a new sock](#)
8. [\[API\] Create POST /admins/login endpoint to handle admin login \(JWT-based\)](#)
9. [\[API\] Create a WithJWTAuth middleware to safeguard sensitive API endpoints](#)

Sprint 3

1. [\[API\] Create GET /orders/:order_id/updates to retrieve all order_updates for a particular order_id](#)
2. [\[UI, EVERYONE\] Research/explore the ShadCN UI components w/ TailwindCSS](#)
3. [\[UI, EVERYONE\] Complete React mini-course project](#)
4. [\[API\] Create GET /orders/:order_id to view order details and item list](#)
5. [\[API\] Create PATCH /orders/:order_id/status to update order status](#)

6. [\[UI\] Generate some basic UI mocks for the main pages](#)
7. [\[API\] Create POST /orders/:order_id/updates to make new updates for an order](#)
8. [\[UI\] Create the core UI routes and auth flow](#)
9. [\[API\] Create PATCH /orders/:order_id/address to update an order's address](#)
10. [\[API\] Update socks deletion to soft delete using is_deleted column](#)
11. [\[API\] Create GET /orders/invoice/:invoice_number to view order details and item list](#)
12. [\[API\] Create PATCH /orders/:order_id/contact to update an order's contact/customer information](#)

Sprint 4

1. [\[UI\] Create useGetOrders in orders service](#)
2. [\[UI\] Create useGetOrderById in order service](#)
3. [\[UI\] Create the AdminOrdersPage](#)
4. [\[UI\] Add pagination to the socks in the HomePage](#)
5. [\[UI\] Create the <SockItem /> and lay it out in a grid within the HomePage](#)
6. [\[UI\] Create the <CartItem /> and render them in the CartPage](#)
7. [\[UI\] Create useGetSocks in inventory service](#)
8. [\[UI\] Create the SockDetailsPage](#)

9. [\[UI\] Create the <RelatedProduct /> component and add the "Related products" section to SockDetailsPage](#)
10. [\[UI\] Create a CartContext to save all items within the cart](#)
11. [\[UI\] Create useGetOrderUpdates in orders service](#)
12. [\[UI\] Admin login page](#)
13. [\[UI\] Create the CheckoutPage](#)
14. [\[UI\] Create footer \(admin + user\)](#)
15. [\[UI\] Create the navbar \(admin + user\)](#)
16. [\[UI\] Create admin profile page with employee directory](#)
17. [\[UI\] Create the <OrderDetails /> and OrderDetailsPage](#)
18. [\[API\] Create POST /cart/checkout/stripe-session to handle order creation and Stripe](#)

Sprint 5

1. [\[API, UI\] Create GET /socks/:sock_id/similar-products endpoint and connect to the UI](#)
2. [\[UI\] Create the OrderConfirmationPage](#)
3. [\[API, UI\] Integrate with Stripe for payment processing](#)
4. [\[UI\] Create /cart/checkout/payment-canceled page](#)
5. [\[API, UI\] Create GET /admins/:admin_id endpoint to get the metadata for a single admin and update UI](#)

Sprint 6

1. [\[UI\] Create the multi-step form to add new socks](#)
2. [\[UI\] Create the <AdminSockDetailsPage />](#)
3. [\[API, UI\] Create the newsletter functionality/endpoints in the API and update the UI](#)
4. [\[UI\] Create the <AdminInventoryPage />](#)

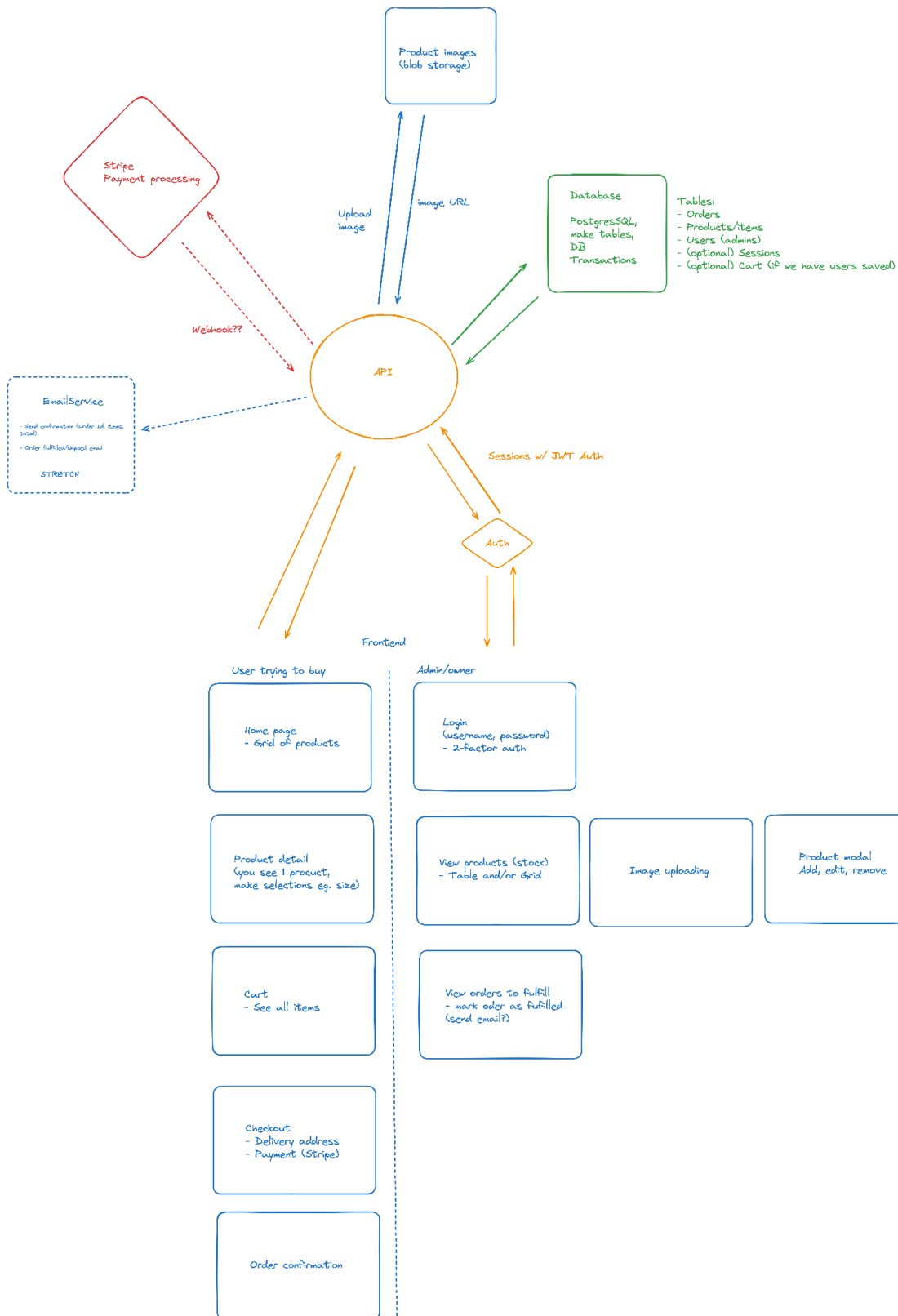
Sprint 7

1. [EVERYONE!! \[DB\] Write SQL to add 20 socks, 10 orders for our "demo" sessions](#)
2. [\[Due 12/02 @ 5PM\] Work on the individual project posters](#)
3. [\[UI\] Dark mode toggle via button in header](#)
4. [\[API\] Integrate an email client to send order confirmations](#)
5. [\[Due 12/04 @ 5PM\] Create final demo presentation recording with walkthrough](#)
6. [\[On 12/06 @ 7:30AM - 12:00PM\] Attend final presentation](#)
7. [\[Due 12/16 @ 5PM\] Final ZIP deliverable \(4 videos + All docs + Code\)](#)

SYSTEM DESIGN

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

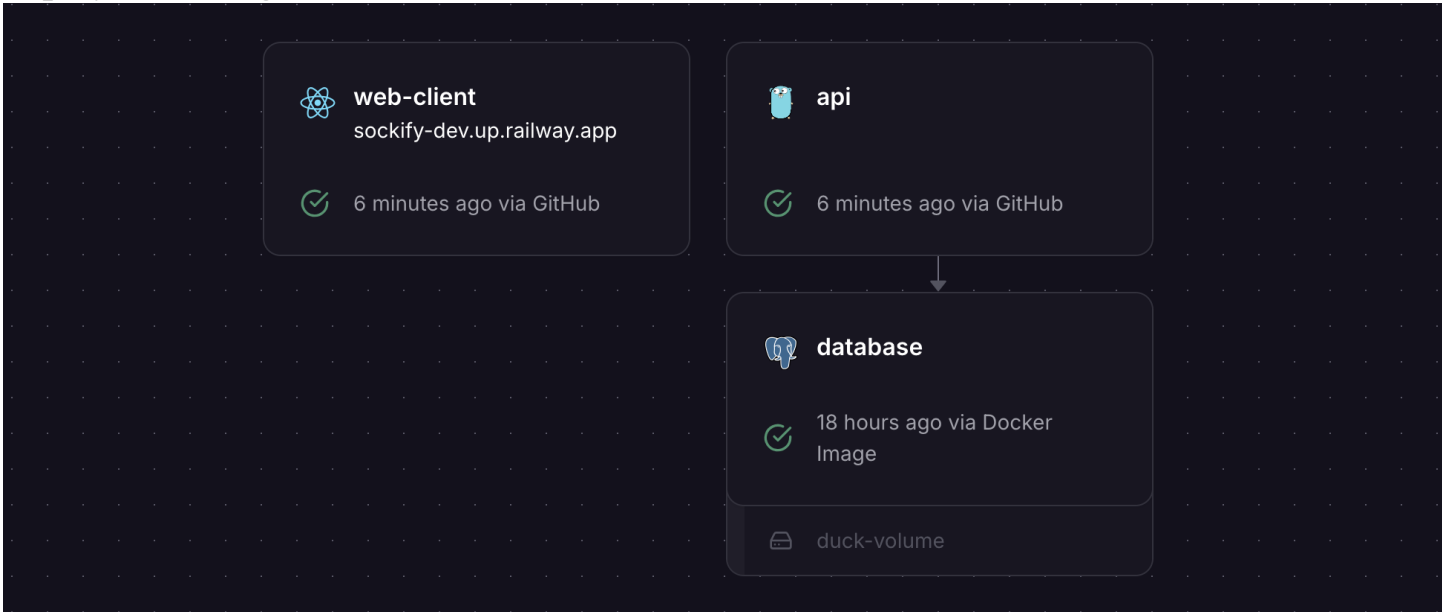
Architectural Patterns



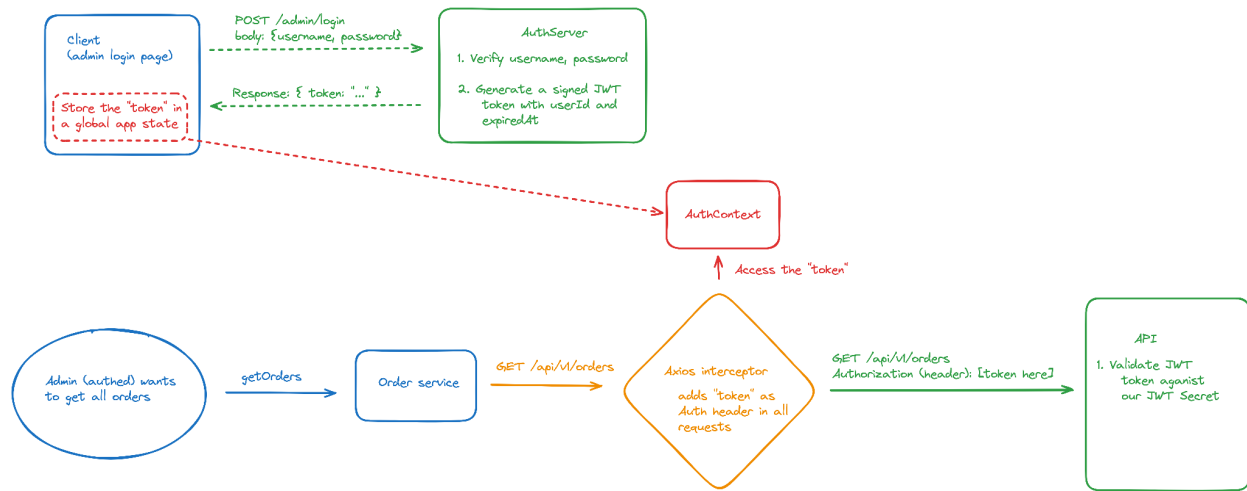
System and Subsystem Decomposition



Deployment Diagram



Design Patterns



SYSTEM VALIDATION

All MVP features were completed and were verified to work as expected.

GLOSSARY

MVP - Minimum viable product

CUJ - Critical user journey

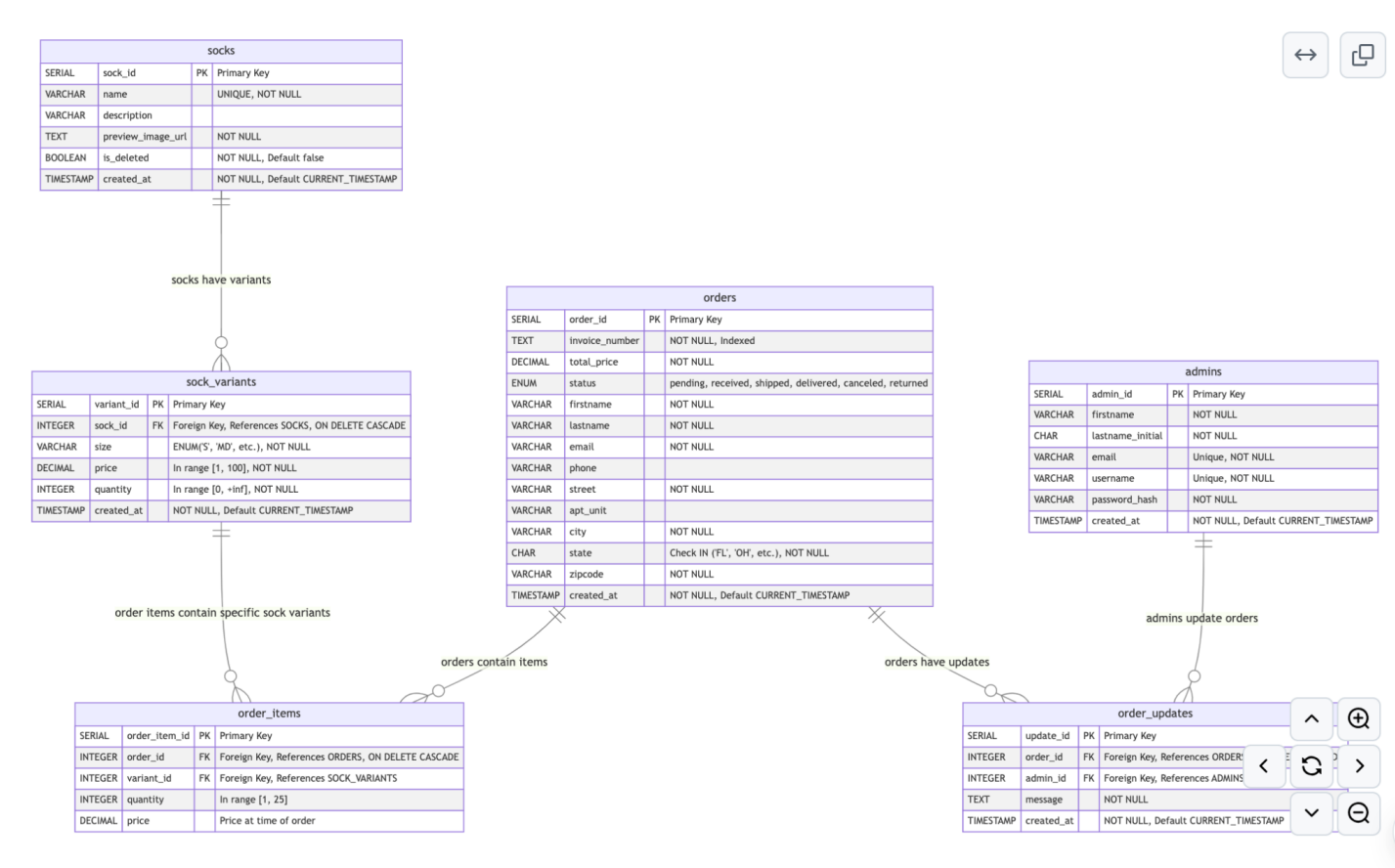
APPENDIX

Appendix A - UML Diagrams

Database schema can be found here:

https://github.com/sockify/sockify/blob/main/docs/planning/database_schema.md

Entity-relationship (ER) diagram



Appendix B - User Interface Design

Can be found here: https://github.com/sockify/sockify/blob/main/docs/planning/ui_mockups.md

Appendix C - Sprint Review Reports

Documents are found here:
https://drive.google.com/drive/folders/16r1wBDVO5GkDRZdZu1Ky_8WRa_hm_oBe?usp=drive_link

Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents

All documents can be found in this Google drive folder:

<https://drive.google.com/drive/folders/1hZvBZaI-ZNuhVaHg59FM1U-9I3i-4s39?usp=sharing>