Parts Inventory Specification

Ahmed Al-Taiar

November 24, 2023

1 Technologies

- Redwood Web Framework
  - React
  - GraphQL
  - Prisma
  - TypeScript
  - Jest
  - Storybook
- TailwindCSS Utility Classes
- DaisyUI Component Library

2 Database

The website's data will be hosted on a local PostgreSQL server with the exception for the part images, those will be stored on Filestack.

2.1 Schemas

2.1.1 Part

Field	Required	Type	Default Value
ID	Yes	Int	Automatically increment
Name	Yes	String	
Description	No	String	No description provided
Available stock	Yes	Int	0
Image URL	Yes	String	Local placeholder image path
Date of creation	Yes	Date	Now
Transaction ID Relation	No	Int	

2.1.2 User

Field	Required	Type	Default Value
ID	Yes	Int	Automatically increment
First name	Yes	String	
Last name	Yes	String	
Email address	Yes	String	
Encrypted password	Yes	String	
Password salt	Yes	String	
Password reset token	No	String	
Password reset token expiry date	No	Date	
Role	Yes	String	user
Transactions	Yes	Transaction[]	

2.1.3 Transaction

Field	Required	Type	Default Value
ID	Yes	Int	Automatically increment
Transaction Date	Yes	Date	Now
User ID relation	Yes	Int	
Type	Yes	TransactionType	
Parts	Yes	Json[]	

TransactionType is an enum of values “in” or “out”.

3 Account System

Use Redwood's built in authentication system then generate pages (using the Redwood command line interface) for signing up, logging in, forgot password, etc. Change the CSS stylesheets to use DaisyUI components instead, for uniformity across pages.

3.1 Sign Up Page

Modify the sign-up page to include text fields for entering the user's first and last name, then include the values in the payload when creating a new account.

4 Part Management

Generate the pages needed to create, retrieve, update, and delete parts using Redwood's scaffold generator. Pass in the Part schema as input. Change the CSS stylesheets to use DaisyUI components instead, for uniformity across pages.

4.1 Part Form

Modify the form for used for creating or updating parts to include Filestack's image uploader component instead of a text field for inputting the image URL, use the outputted image URL from the uploader component in the payload for creating/updating parts.

4.2 Deleting Parts

After a part is deleted, automatically delete the image from Filestack as well.

4.3 Retrieving Parts

When retrieving a part's image through the image URL, use Filestack's transformation parameters to resize the image to the appropriate size in order to conserve bandwidth. Also create a new GraphQL query, for retrieving parts based on a page and filter, that takes in the following parameters:

Parameter	Required	Type	Default Value
Page	Yes	Int	
Sort Method	Yes	SortMethod	
Sort Order	Yes	SortOrder	
Search query	No	String	Nothing

SortMethod is an enum of any one of these values:

- ID

- Name
- Description
- Available stock
- Date of creation

SortOrder is an enum of any one of these values:

- Ascending
- Descending

5 Basket

5.1 Adding To Basket

When a user adds a part to the basket, save the part as a Json string and also include the quantity. If the same part is already in the basket, increment the quantity instead. Then save the entire basket as a string in the browser's local storage.

5.2 Clearing Basket

Delete the string from the browser's local storage.

5.3 Deleting From Basket

Delete the part & quantity element based on the index.

5.4 Editing Basket

The quantity of each element can be modified directly in the basket page.

6 Transactions

6.1 Creating Transactions

When a user checks out their order, create a transaction in the database. Save the basket and its contents, the user's ID, and date. When the transaction is created, decrement the available stock of each specified part by the quantity. Then clear the basket from the browser's local storage.

6.2 Retrieving Transactions

Administrator accounts get access to a second transactions page. It is the exact same as the normal transactions' page with the exception that it lists all transactions instead of just the user's.

6.3 Returning

Create a new GraphQL mutation which takes the transaction's ID and user's ID, and then marks a transaction's TransactionType to "in", then increment the available stock of each part by the quantity. This mutation can be accessed in the user's transactions page when they want to return parts.

7 Miscellaneous

7.1 Theme

Use DaisyUI's themes and create a theme toggle component in the navigation bar that switches between light and dark theme. The theme should be saved in the browser's local storage so the theme persists across sessions.

7.2 Navigation Bar

On mobile, hide the links on the navigation bar and put them in a list, that is shown with the press of one button on the navigation bar. It should also have the theme toggle component.