

Parts Inventory IPO Charts

Ahmed Al-Taiar

December 8, 2023

1 Enter

1.1 Check If User Is Logged In

| Input | Process | Output |
|-------|--|--|
| | Use the destructured boolean <code>isAuthenticated</code> variable within Redwood's <code>useAuth</code> React hook. | IF <code>isAuthenticated</code> , <code>true</code> ELSE , <code>false</code> |

1.2 Check Theme & Default To Light

| Input | Process | Output |
|-------|---|---|
| | Check the stored <code>theme</code> value in the browser's local storage. IF <code>theme</code> does not exist ⇒ Set <code>theme</code> to "light" | IF <code>theme</code> is "dark", <code>true</code> ELSE , <code>false</code> |

2 404

| Input | Process | Output |
|-------------------------|---------|--------------------|
| "Back to safety" button | | Go to catalog page |

3 Navigation Bar

3.1 Render Navigation Bar

| Input | Process | Output |
|----------------|---|-------------------------------------|
| Child elements | IF screen width is small ⇒ Render menu with all buttons ELSE ⇒ Render all buttons on navigation bar Render child elements | Page with navigation bar at the top |

3.2 Render All Buttons

| Input | Process | Output |
|---|---|---------|
| 1. User is logged in (boolean) | Render “Parts Inventory” button Render “basket” button | Buttons |
| 2. User is admin (false if user is not logged in) | IF user is logged in ⇒ Render “sign out” button ⇒ Render “transactions” button IF user is admin ⇒ Render “parts” button ⇒ Render (admin) “transactions” button | |

3.3 Sign Out

| Input | Process | Output |
|-------------------|---|--------------------------------------|
| “Sign out” button | Use the destructured <code>logout</code> function within Redwood’s <code>useAuth</code> React hook. | Same page the user was on originally |

4 Admin Navigation Bar

4.1 Render Admin Navigation Bar

| Input | Process | Output |
|----------------|--------------------------------------|---|
| Child elements | Render all buttons on navigation bar | Page with admin navigation bar at the top |
| | Render child elements | |

4.2 Render All Buttons

| Input | Process | Output |
|---|--|---------|
| 1. Button label (string) | Render home button that goes to the catalog page (5.2) | Buttons |
| 2. Button destination URL (string) | Render title button with specified title and title destination URL | |
| 3. Title label (string) | | |
| 4. Title destination URL (string) | Render button with specified button destination URL | |

5 Catalog Page

| Input | Process | Output |
|---|---|--------------|
| Any one of: <ul style="list-style-type: none">• Redirect• Enter (1)• “Parts Inventory” button (3.2) | Render navigation bar (3.1) Render static components Render catalog | Catalog page |

5.1 Static Components

| Input | Process | Output |
|-------|--|-------------------|
| | Render “Parts Inventory” Render “Only take what you need” | Static components |

5.2 Catalog

| Input | Process | Output |
|------------------------------------|---|---------|
| 1. Page (default: 1) | Query database for parts with inputted search parameters | Catalog |
| 2. Sort Method (default: ID) | Render a “loading” placeholder while the database is being queried | |
| 3. Sort Order (default: ascending) | | |
| 4. Search query (optional) | IF no parts found \Rightarrow Render “empty” ELSE IF an error occurred \Rightarrow Render “error” ELSE \Rightarrow FOR each part $\Rightarrow\Rightarrow$ Render details $\Rightarrow\Rightarrow$ Render “add to basket” button (15.1.2, quantity of 1) $\Rightarrow\Rightarrow$ IF part’s stock is 0 $\Rightarrow\Rightarrow\Rightarrow$ Disable button $\Rightarrow\Rightarrow$ Redirect to the part’s page if the part is pressed | |
| | Render dropdown menus to change the search parameters, if any change, repeat 5.2 | |

6 Forgot Password Page

6.1 Render Form

| Input | Process | Output |
|-------|--|--------|
| | Render email address input Render submit button | Form |

6.2 Submit

| Input | Process | Output |
|--|---|---|
| 1. “Submit” button 2. Email address (string) | Query database to find account associated with the inputted email address | IF Account exists with email address ⇒ Send email to account’s address with password reset link ELSE ⇒ Return to form, account doesn’t exist (6.1) |

7 Login Page

7.1 Render Form

| Input | Process | Output |
|-------|--|--------|
| | Render email address input Render password input Render “forgot password” link Render “sign up” link Render “login” button | Form |

7.2 Log In

| Input | Process | Output |
|---|---|---|
| 1. “Login” button | Use the destructured <code>logIn</code> function within Redwood’s <code>useAuth</code> React hook | IF login successful ⇒ Go to the page the user was previously on |
| 2. Email address (<code>string</code>) | IF credentials match ⇒ Save session in browser’s cookies, so user is still logged in | |
| 3. Password (<code>string</code>) | ELSE ⇒ Reject login | |

7.3 Forgot Password

| Input | Process | Output |
|------------------------|---------|--------------------------------|
| “Forgot password” link | | Go to forgot password page (6) |

7.4 Sign Up

| Input | Process | Output |
|----------------|---------|------------------------|
| “Sign up” link | | Go to sign up page (8) |

8 Sign Up Page

8.1 Render Form

| Input | Process | Output |
|-------|--|--------|
| | Render first name input Render last name input Render email address input Render password input Render “login” link Render “sign up” button | Form |

8.2 Sign Up

| Input | Process | Output |
|------------------------------------|--|---|
| 1. “Sign up” button | Use the destructured signUp function within Redwood’s useAuth React hook | Go to the page the user was previously on |
| 2. First name (string) | | |
| 3. Last name (string) | Save session in browser’s cookies, so user is immediately logged in | |
| 4. Email address (string) | | |
| 5. Password (string) | | |

8.3 Log In

| Input | Process | Output |
|---------------|---------|----------------------|
| “Log in” link | | Go to login page (7) |

9 Reset Password Page

9.1 Render Form

| Input | Process | Output |
|-------------------------------|--|--------|
| Reset token (string) | Render new password input Render “submit” button Match reset token with associated account IF reset token is not valid ⇒Disable “submit” button | Form |

9.2 Reset Password

| Input | Process | Output |
|---|--|----------------------|
| 1. “Submit” button 2. New password (string) | Use the destructured <code>resetPassword</code> function within Redwood’s <code>useAuth</code> React hook | Go to login page (7) |

10 Part Management Page

| Input | Process | Output |
|---|--|----------------------|
| Either: <ul style="list-style-type: none">• Redirect• “Parts” button | Render admin navigation bar (4.1) Render part management | Part management page |

10.1 Part Management

| Input | Process | Output |
|--------------------------|--|------------|
| Database query for parts | Render a “loading” placeholder while the database is being queried IF no parts found ⇒ Render “empty” ELSE IF an error occurred ⇒ Render “error” ELSE ⇒ FOR each part ⇒ ⇒ Render part ID, name, description, stock, thumbnail, and creation date in a table row ⇒ ⇒ Render “show” button, that goes to its part details page (11) ⇒ ⇒ Render “edit” button that goes to its edit page (12) ⇒ ⇒ Render “delete” button that deletes the part (10.1.1) | Parts list |

10.1.1 Delete Part

| Input | Process | Output |
|---|---|--------------------|
| <ul style="list-style-type: none">Part (object) | Confirm if the admin wants to delete the part | Refresh parts list |
| <ul style="list-style-type: none">“Delete” button | IF admin confirms yes ⇒ Delete part | |

11 Admin Part Page

| Input | Process | Output |
|--------------------|--|-----------------|
| Part “show” button | Render admin navigation bar (4.1) Render admin part | Admin part page |

11.1 Admin Part

| Input | Process | Output |
|---------------|--|--------------|
| Part (object) | Render complete part ID, name, description, stock, image, and creation date Render “edit” button that goes to its edit page (12) Render “delete button” that deletes the part (10.1.1) | Part details |

12 Edit Part Page

| Input | Process | Output |
|--------------------|---|----------------|
| Part “edit” button | Render admin navigation bar (4.1) Render edit form | Edit part page |

12.1 Edit Part

| Input | Process | Output |
|---------------|--|-----------|
| Part (object) | Render part name input (initial value: part's name) Render part description input (initial value: part's description) Render stock number input (initial value: part's stock, must be ≥ 0) Render current image with "replace image" button Render "save" button | Edit form |

12.2 Replace Image

| Input | Process | Output |
|------------------------|------------------------------|--|
| "Replace image" button | Render Filestack file upload | IF new image file is up- loaded \Rightarrow Filestack CDN URL of the new image ELSE \Rightarrow Current image URL |

12.3 Save

| Input | Process | Output |
|--|--|-------------------------------|
| <ul style="list-style-type: none">• Part (object) | Overwrite the part's fields with the new values | Go to part management page |
| <ul style="list-style-type: none">• New name (string) | Update part in database | |
| <ul style="list-style-type: none">• New description (string) | | |
| <ul style="list-style-type: none">• New stock (int) | | |
| <ul style="list-style-type: none">• New image URL (string) | | |

13 New Part Page

| Input | Process | Output |
|-------------------|---|---------------|
| “New part” button | Render admin navigation bar (4.1) Render new part form | New part page |

13.1 New Part Form

| Input | Process | Output |
|-------|--|--------|
| | Render part name input Render part description input Render stock number input (initial value: 0, must be ≥ 0) Render Filestack file upload Render “save” button | Form |

13.2 Save

| Input | Process | Output |
|---|-----------------------------|----------------------------|
| <ul style="list-style-type: none">• Name (string)• Description (string)• Stock (int)• Image URL (string) | Create new part in database | Go to part management page |

14 Part Details Page

| Input | Process | Output |
|--------------------|--|-------------------|
| Catalog part click | Render navigation bar (3.1) Render part details | Part details page |

14.1 Part Details

| Input | Process | Output |
|------------------------------|---|--------------|
| Part (<code>object</code>) | Render complete part name, description, stock, and image Render quantity selector (range from 1 to part's stock) Render “add to basket” button with specified quantity (15.1.2) | Part details |

15 Basket Page

| Input | Process | Output |
|-----------------|--|-------------|
| “Basket” button | Render navigation bar (3.1) Render basket | Basket page |

15.1 Basket

| Input | Process | Output |
|--|--|--------|
| Basket entry in browser's local storage (<code>string</code> or <code>null</code>) | IF basket is <code>null</code> OR empty ⇒Render “empty” ELSE ⇒Parse basket to <code>object[]</code> ⇒ FOR each part in basket ⇒⇒Render thumbnail & title ⇒⇒Render quantity selector (range from 1 to part's stock) ⇒⇒Render “delete” button ⇒Render “clear basket” button ⇒Render “checkout” button | Basket |

15.1.1 Delete From Basket

| Input | Process | Output |
|--|---|------------|
| <ul style="list-style-type: none">• Basket entry in browser's local storage (string or null)• Basket part• "Delete" button | Parse basket to an object[] , or create a new array if null or empty Remove part & quantity from basket array Convert basket array back to string Overwrite basket in browser's local storage with new basket string | New basket |

15.1.2 Add To Basket

| Input | Process | Output |
|--|--|------------|
| <ul style="list-style-type: none">• Basket entry in browser's local storage (string or null)• Part (object)• Quantity (int)• "Add to basket" button | Parse basket to an object[] , or create a new array if null or empty Add part & quantity to basket array Convert basket array back to string Overwrite basket in browser's local storage with new basket string | New basket |

15.1.3 Clear Basket

| Input | Process | Output |
|-----------------------|---|--------|
| "Clear basket" button | Delete basket entry from browser's local storage | |

15.1.4 Checkout

| Input | Process | Output |
|--|--|----------------------------------|
| <ul style="list-style-type: none">• User is logged in (boolean)• Parsed basket entry from browser's local storage (<code>object[]</code>)• "Checkout" button | IF user is not logged in ⇒Reject transaction ELSE ⇒Get user's ID ⇒ FOR each part in basket ⇒⇒Get up-to-date part details from database ⇒⇒ IF specified quantity for part > part's stock ⇒⇒⇒Reject transaction ⇒⇒ ELSE ⇒⇒⇒Decrement part's stock by quantity ⇒Create new transaction in database, with basket, user's ID, and "out" as the transaction type ⇒Delete basket entry from browser's local storage | Rejection message or transaction |

16 Transactions Page

| Input | Process | Output |
|-----------------------|--|-------------------|
| "Transactions" button | Render navigation bar (3.1) Render transactions | Transactions page |

16.1 Transactions

| Input | Process | Output |
|--|---|--------------|
| <ul style="list-style-type: none">• User ID (<code>int</code>)• Filter by (<code>enum</code>) | <p>Query database for transactions linked to specified user ID and filter</p> <p>Render “filter by” selection</p> <p>IF transactions is empty ⇒Render “empty”</p> <p>ELSE ⇒FOR each transaction ⇒⇒Render item count, relative time, and type (“in”/“out”) ⇒⇒On press, reveal quantity and title of each part</p> | Transactions |

17 Admin Transactions Page

| Input | Process | Output |
|-------------------------------|---|-------------------------|
| (Admin) “Transactions” button | <p>Render admin navigation bar (4.1)</p> <p>Render admin transactions</p> | Admin transactions page |

17.1 Admin Transactions

| Input | Process | Output |
|---------------------------|---|--------------------|
| Filter by (enum) | Query database for all transactions and filter by specified filter Render “filter by” selection IF transactions is empty ⇒Render “empty” ELSE ⇒ FOR each transaction ⇒⇒Render item count, relative time, type (“in”/“out”), and user’s full name ⇒⇒On press, reveal quantity and title of each part | Admin transactions |