Dokumentation der technisch und nicht-technisch Lösungen

❖ App.js

Funktion zum Starten der Bearbeitung eines Elements:

- Problem: Das Problem mit der Funktion war, dass wir zweimal eine Warnmeldung erhielten, einmal zu Beginn der Funktion und das zweite Mal, als das Produkt geändert wurde.

- Lösung: Die Funktionsschleife so geändert, dass sie nur zu Beginn eine Bestätigung verlangt.

- Endgültiger Code:

```
// Funktion zum Starten der Bearbeitung eines Produkts
function startEditing(item) {
 // Bestätigungsdialog für den Start der Bearbeitung anzeigen
 const isConfirmed = window.confirm("Möchten Sie das Element wirklich bearbeiten?");
 if (!isConfirmed) {
  return;
 }
```

Funktion zum Speichern der Einkaufsliste des Benutzers:

- Problem:
    - Das erste Problem bestand darin, dass die Funktion den Namen des Käufers nicht überschrieb und ihn zwei oder mehr Mal neu schrieb, wobei die vorherigen Änderungen gespeichert blieben;
    - Das zweite Problem bestand darin, dass der Name des Käufers nach dem Speichern der Liste nicht mehr korrekt angezeigt wurde;
- Lösung:
    - Erstens den Code so zu schreiben, dass der Name des Käufers nicht kopiert, sondern mit demselben Namen überschrieben wird.
    - Zweitens eine kommentierte Codezeile.
- Endgültiger Code:

```
// Funktion zum Speichern der Einkaufsliste des Benutzers
 function saveUserShopping() {
  if (userName.trim() === "") {
   alert("Bitte geben Sie einen Benutzernamen ein.");
   return;
  }

  const userShopping = {
   userName,
   items: [...items],
  };
// Zurücksetzen von Benutzername und Liste
 //setUserName("");
```

Funktion zum Löschen der aktuellen Liste des Benutzers:

- Problem: Die aktuelle Einkaufsliste wurde gelöscht, aber der Name des Käufers wurde nicht.
- Lösung: Den aktuellen Namen aus dem Selektor löschen.
- Endgültiger Code:

```
// Liste und ausgewählten Benutzer zurücksetzen
setItems([]);
setSelectedUser(null);
setSavedUsers(savedUsers.filter(user => user.userName !== userName));
```

Funktion zum Herunterladen der Einkaufsliste als PDF (Download):

- Problem:

Bei der Funktion des Herunterladens der Einkaufsliste traten Probleme bei der Änderung auf, da das Programm zunächst ein einfaches Textdokument und später eine PDF-Datei mit einem Logo und einem Hintergrund in der Farbe des Anwendungs Hintergrunds herunterlädt. Die Probleme ergaben sich aus der Tatsache, dass Änderungen an dem Dokument nicht in Echtzeit vorgenommen werden konnten, da die Liste jedes Mal heruntergeladen werden musste, um die neuen Änderungen an den Koordinaten des Logos und der darin geschriebenen Texte zu sehen.

- Endgültiger Code:

```
function downloadList() {
  // Erstelle eine Zeile mit dem Benutzernamen
  const userNameLine = `Name des Käufers: ${userName}`;
  // Erstelle den Inhalt der Liste, indem die Elemente formatiert werden
  const listContent = items.map(item => (
    `${item.value}: ${item.quantity} ${item.unit}, ${item.category}`
  )).join('\n');

  // Erstelle ein neues PDF-Dokument mit jsPDF
  const pdf = new jsPDF();
  pdf.setFillColor(104, 190, 244);
  pdf.rect(0, 0, pdf.internal.pageSize.width, pdf.internal.pageSize.height, 'F');


  // Füge das Logo deines Projekts zum Dokument hinzu
  const logoUrl = 'shopwise-high-resolution-logo-transparent.png'; // Ersetze dies durch den Pfad zu deinem Logo
  pdf.addImage(logoUrl, 'PNG',150, 10, 56, 40); // Passe die Koordinaten und Größe des Logos an
  pdf.setFont('helvetica');
  pdf.setFontSize(20);
  // Füge den Inhalt der Liste zum Dokument hinzu  const listContent: string
  pdf.text(`${userNameLine}\n\nEinkaufsliste:\n${listContent}`, 10, 50);
  pdf.text(`Danke, dass Sie unsere App nutzen`,10 ,250);


  // Speichere das PDF-Dokument
  pdf.save('shopping-list.pdf');
}
```

❖ App.css

Farbe der Schaltflächen:

- Problem: Es gibt keine orangefarbene Schaltfläche Vorlage in Bootstrap.
- Lösung: Button btn-outline-warning vorlage benutzen und dazu eine orangefarbene Schaltfläche in CSS programmieren.
- Endgültiger Code:

```css
/* Buttons */
.btn-outline-warning {
  text-align: center;
  width: 15%;
  margin: auto;
  margin-right: 1%;
  background-color: orange;
  color: white;
  border: 0cap;
  font-size: 20px;
  cursor: pointer;
  }
```

Positionierung der ersten Reihe:

- Problem: Die erste Reihe von Feldern: Käufer, Produkt, Menge, Einheit, Kategorie und Hinzufügen-Taste, wurde nicht an das Tabellenlayout angepasst.
- Lösung: Definieren der Breite des Containers, der Komponenten und des Feldes des Käufers.
- Endgültiger Code:

```css
.container {
  width: 80%;
  margin: auto;
}

.input-group {
  width: 100%;
  text-align: center;
}

.einkäufer {
  width: 320px !important;
}
```

Ändern der Textgröße mit der Bootstrap-Vorlage:

- Problem: Die erste Reihe von Feldern: Käufer, Produkt, Menge, Einheit, Kategorie und Hinzufügen-Taste, wurde nicht an das Tabellenlayout angepasst.
- Lösung: Unter “className” den Bootstrap code: `"col fs-4 rounded text-center border-0 einkaufslisten"` im fs-Teil ändern. Es war vorher fs-6 und wurde in fs-4 geändert.
- Endgültiger Code:

```
{/* Dropdown-Menü für die Auswahl gespeicherter Benutzer */}
<select
  className="col fs-4 rounded text-center border-0 einkaufslisten"
  value={selectedUser ? selectedUser.userName : ""}
  onChange={(e) => selectUser(savedUsers.find(user => user.userName === e.target.value))}
>
```

Positionierung der Schaltfläche "Hinzufügen”

- Problem: Der Abstand zwischen der Schaltfläche "Hinzufügen" und dem Kategoriefeld ist zu groß.
- Lösung: Programmieren der Abmessungen der Schaltfläche "Hinzufügen" in CSS
- Endgültiger Code:

```
{/* Button zum Hinzufügen oder Aktualisieren */}
<div className="col-2 input-group-append">
  <button
    onClick={updateItemList}
    className="btn btn-outline-warning add-button"
  >
```

```
  /* Add Button */
.add-button {
  width: 200px !important;
  height: 50px;
  margin-left: -10px;
}
```