```
pizza-pantry/
|— app/                    # App Router pages & layouts
|   ├— layout.tsx            # Root layout
|   ├— page.tsx              # Landing / redirect to login
|   ├— dashboard/            # Protected routes
|   |   ├— layout.tsx
|   |   ├— page.tsx          # Inventory list page
|   |   ├— items/
|   |   |   ├— new/page.tsx     # Add Item form
|   |   |   ├— [id]/edit/page.tsx
|   |   |   ├— [id]/audit/page.tsx
|   |   |   └— [id]/adjust/page.tsx
|   |   └— api/              # Server route handlers
|   |       ├— items/
|   |       |   ├— route.ts     # POST (create), GET (list/search)
|   |       |   └— [id]/route.ts# GET, PATCH, DELETE
|   |       └— adjust/route.ts  # POST stock adjustments
|   └— auth/                 # Clerk auth pages (or custom)
|
|— components/               # Reusable UI
|   ├— forms/                # Zod + react-hook-form forms
|   ├— ui/                   # Buttons, modals, tables, inputs
|   └— inventory/            # ItemCard, ItemTable, etc.
|
|— lib/
|   ├— db.ts                 # MongoDB connection helper
|   ├— models/Item.ts          # Mongoose schema/model
|   ├— validations/          # Zod schemas
|   └— auth.ts               # Clerk middleware helpers
```

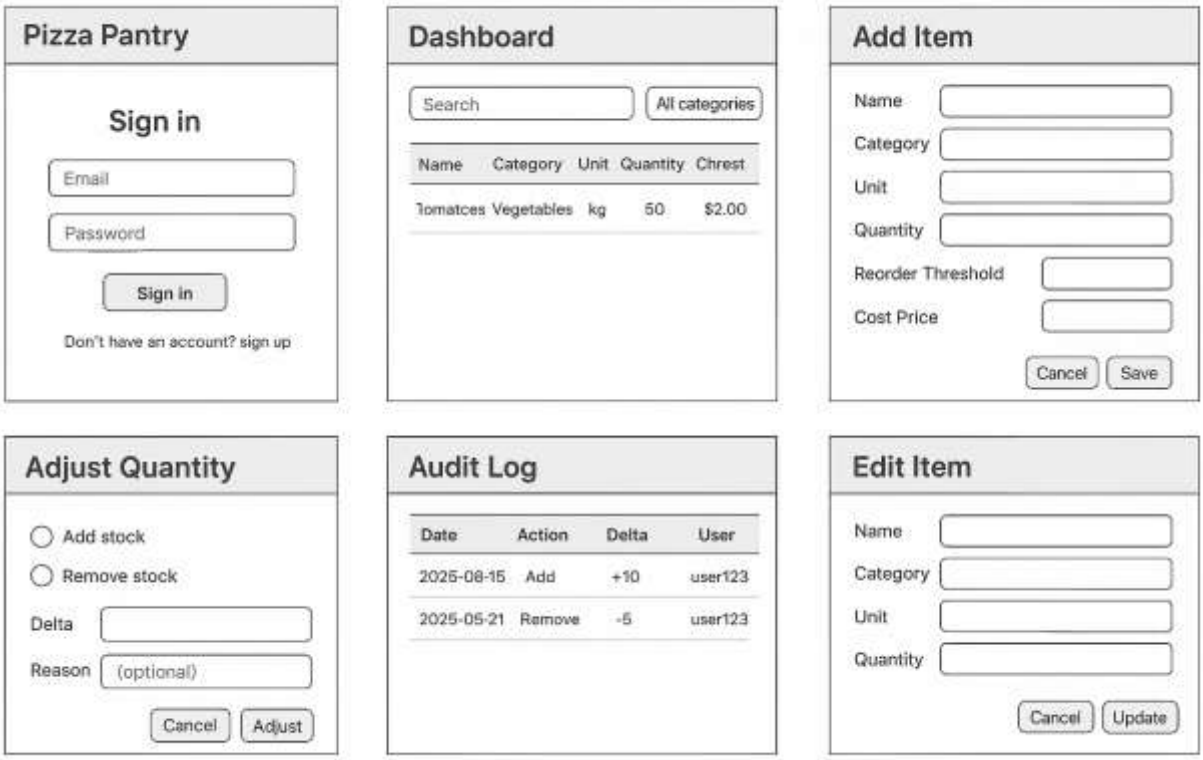
```
|
|— styles/                 # Global styles (Tailwind)
|— utils/                  # Utility functions
|— public/                 # Images/icons
|— .env.example            # Example env file
|— package.json
|— tsconfig.json
|— README.md
```

```
// lib/models/Item.ts
import { Schema, model, models } from "mongoose";

const ItemSchema = new Schema({
  name: { type: String, required: true },
  category: { type: String, required: true },
  unit: { type: String, required: true }, // e.g., "kg", "box"
  quantity: { type: Number, required: true, default: 0 },
  reorderThreshold: { type: Number, default: 5 },
  costPrice: { type: Number, required: true },
  createdBy: { type: String, required: true }, // Clerk userId
}, { timestamps: true });

export default models.Item || model("Item", ItemSchema);
```

==========



app/layout.tsx

```
import "./globals.css";
import { ClerkProvider } from "@clerk/nextjs";

export default function RootLayout({ children }: { children: React.ReactNode }) {
  return (
    <ClerkProvider>
      <html lang="en">
        <body className="bg-gray-50 min-h-screen">{children}</body>
      </html>
    </ClerkProvider>
  );
}
```

`app/page.tsx` (redirect to dashboard or login)

```
import { redirect } from "next/navigation";
import { auth } from "@clerk/nextjs";

export default function HomePage() {
  const { userId } = auth();
  if (!userId) redirect("/sign-in");
  else redirect("/dashboard");
}
```

`app/dashboard/page.tsx` (Inventory List)

```
import InventoryTable from "@/components/inventory/InventoryTable";
import InventoryFilters from "@/components/inventory/InventoryFilters";

export default function DashboardPage() {
  return (
    <main className="p-6 space-y-4">
      <h1 className="text-2xl font-bold">Inventory</h1>
      <InventoryFilters />
      <InventoryTable items={[]} /> {/* placeholder */}
    </main>
  );
}
```

`app/dashboard/items/new/page.tsx` (Add Item)

```
import ItemForm from "@/components/inventory/ItemForm";

export default function AddItemPage() {
  return (
    <main className="p-6">
      <h1 className="text-xl font-bold">Add Item</h1>
      <ItemForm mode="create" />
    </main>
  );
}
```

`app/dashboard/items/[id]/edit/page.tsx` (Edit Item)

```
import ItemForm from "@/components/inventory/ItemForm";

export default function EditItemPage() {
  return (
    <main className="p-6">
      <h1 className="text-xl font-bold">Edit Item</h1>
      <ItemForm mode="edit" />
    </main>
  );
}
```

`app/dashboard/items/[id]/audit/page.tsx` (Audit Log)

```
import AuditLogList from "@/components/inventory/AuditLogList";

export default function AuditLogPage() {
  return (
    <main className="p-6">
      <h1 className="text-xl font-bold">Audit Log</h1>
      <AuditLogList logs={[]} />
    </main>
  );
}
```

components/inventory/InventoryTable.tsx

```
"use client";

type Item = {
  _id: string;
  name: string;
  category: string;
  unit: string;
  quantity: number;
  reorderThreshold: number;
  costPrice: number;
```

```
};

export default function InventoryTable({ items }: { items: Item[] }) {
  return (
    <table className="w-full border">
      <thead className="bg-gray-100">
        <tr>
          <th>Name</th>
          <th>Category</th>
          <th>Unit</th>
          <th>Quantity</th>
          <th>Threshold</th>
          <th>Cost</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        {items.length === 0 ? (
          <tr>
            <td colSpan={7} className="text-center p-4">No items found.</td>
          </tr>
        ) : (
          items.map((item) => (
            <tr key={item._id}>
              <td>{item.name}</td>
              <td>{item.category}</td>
              <td>{item.unit}</td>
              <td>{item.quantity}</td>
              <td>{item.reorderThreshold}</td>
```

```
            <td>{item.costPrice}</td>
            <td>
              <button>Edit</button>
              <button>Adjust</button>
              <button>Audit</button>
              <button>Delete</button>
            </td>
          </tr>
        ))
      )}
    </tbody>
  </table>
 );
}
```

components/inventory/InventoryFilters.tsx

```
"use client";

export default function InventoryFilters() {
 return (
  <div className="flex gap-2 mb-4">
   <input
    type="text"
    placeholder="Search..."
    className="border px-3 py-2 rounded w-1/3"
   />
   <select className="border px-3 py-2 rounded">
    <option value="">All categories</option>
    <option value="vegetables">Vegetables</option>
    <option value="meat">Meat</option>
```

```
      <option value="dairy">Dairy</option>
    </select>
   </div>
  );
}
```

components/inventory/ItemForm.tsx

```
"use client";

type ItemFormProps = {
  mode: "create" | "edit";
};

export default function ItemForm({ mode }: ItemFormProps) {
  return (
   <form className="space-y-4 max-w-md">
    <input type="text" placeholder="Name" className="w-full border p-2" />
    <input type="text" placeholder="Category" className="w-full border p-2" />
    <input type="text" placeholder="Unit (e.g. kg, box)" className="w-full border p-2" />
    <input type="number" placeholder="Quantity" className="w-full border p-2" />
    <input type="number" placeholder="Reorder Threshold" className="w-full border p-2" />
    <input type="number" placeholder="Cost Price" className="w-full border p-2" />

    <div className="flex gap-2">
     <button type="button" className="px-4 py-2 border rounded">Cancel</button>
     <button type="submit" className="px-4 py-2 bg-blue-600 text-white rounded">
      {mode === "create" ? "Save" : "Update"}
     </button>
    </div>
```

```
    </form>
  );
}
```

components/inventory/AuditLogList.tsx

```
"use client";

type Log = {
  _id: string;
  action: "add" | "remove";
  delta: number;
  userId: string;
  createdAt: string;
};

export default function AuditLogList({ logs }: { logs: Log[] }) {
  return (
    <table className="w-full border">
      <thead className="bg-gray-100">
        <tr>
          <th>Date</th>
          <th>Action</th>
          <th>Delta</th>
          <th>User</th>
        </tr>
      </thead>
      <tbody>
```

```
      {logs.length === 0 ? (
        <tr>
          <td colSpan={4} className="text-center p-4">No logs available.</td>
        </tr>
      ) : (
        logs.map((log) => (
          <tr key={log._id}>
            <td>{new Date(log.createdAt).toLocaleString()}</td>
            <td>{log.action}</td>
            <td>{log.delta}</td>
            <td>{log.userId}</td>
          </tr>
        ))
      )}
    </tbody>
  </table>
 );
}
```

==========