
WAREHOUSE AUTOMATION DASHBOARD – DESIGN & ARCHITECTURE DOCUMENT

---

1. Introduction

This document explains the design, architecture, and implementation details of the **Warehouse Automation Dashboard** built as part of the internship assignment.
The main objective of the project is to provide a clean and simple UI to monitor warehouse robots, track tasks, and view inventory status.

The application is built using:

- React (Vite)
- Tailwind CSS
- Recharts (local installation)
- Motion One (basic animations)
- JavaScript ES6

The project focuses on clarity, responsiveness, and meeting all functional requirements given in the assignment.

---

2. High-Level System Overview

The system follows a **component-based architecture** commonly used in modern frontend development.
The application is divided into separate logical layers:

1. **Pages Layer** – Handles individual screens
2. **Components Layer** – Reusable UI components
3. **Data Layer** – Static mock data for robots, tasks, inventory
4. **Routing Layer** – Navigation between pages
5. **Utilities/Styles** – Tailwind and motion animations

This modular structure makes the project easy to maintain, scale, and understand.

---

3. Project Folder Structure

src/

├── components/

│ ├── Sidebar.jsx

│ ├── TopBar.jsx

│ └── RobotCard.jsx

│

├── pages/

│ ├── Dashboard.jsx

│ ├── Robots.jsx

│ ├── Tasks.jsx

│ └── Inventory.jsx

│

├── data/

│ ├── robots.js

│ ├── tasks.js

│ └── inventory.js

│

├── App.jsx

└── main.jsx

This structure keeps pages, components, and data clearly separated.

---

4. UI/UX Design

4.1 Layout

The layout consists of:

- A left-side **Sidebar** for navigation
- A **TopBar** showing page title and user avatar

- A main content area for dashboard pages

This layout is commonly used in admin dashboards for clarity and professional appearance.

4.2 Colors & Styling

The UI uses:

- Blue (primary)
- White card-based layout
- Soft shadows
- Rounded corners

These choices help maintain a clean, modern interface.

4.3 Responsiveness

Tailwind CSS ensures responsiveness using utility classes:

- md:grid-cols-3
- sm:grid-cols-2
- p-6, gap-6
- Automatic stacking on mobile screens

4.4 Animations

Motion One is used for:

- Fade-in page transitions
- Hover interactions on cards

This improves user experience without making the UI heavy.

---

5. Feature-Level Design

5.1 Dashboard Page

The dashboard shows:

- Robot metrics (Total, Idle, Busy, Low Battery)
- Task metrics (Pending, In-Progress, Completed)
- Inventory metrics (Total Items, Low Stock)
- A bar chart showing robot status distribution

This gives a quick high-level overview of the warehouse.

5.2 Robots Page

Features:

- List of all robots
- Search robots by ID
- Battery level bar
- Location indicator
- Status-based metric cards
- Robot status chart (Idle, Busy, Charging)

This helps understand real-time robot activity.

5.3 Tasks Page

Includes:

- Task list (ID, description, robot assigned, status)
- Search functionality
- “Add Task” popup form
- Toast notification on task creation

This allows fast task assignment and monitoring.

5.4 Inventory Page

Shows:

- Item name
- Quantity
- Warehouse location
- Low-stock highlighting
- KPI summary cards
- Search bar

Useful for warehouse stock monitoring.

---

6. Data Flow

Robots Data

Stored in robots.js, containing:

- id
- status
- battery level
- location

Used in Dashboard and Robots pages.

Tasks Data

Stored in tasks.js, containing:

- id
- description
- robot assigned
- status

Used in Tasks page with state updates.

Inventory Data

Stored in inventory.js, containing:

- id
- item name
- quantity
- location

Used in Inventory page list rendering.

---

7. Routing Architecture

React Router handles navigation:

/ → Dashboard

/robots → Robots page

/tasks → Tasks page

/inventory → Inventory page

This keeps navigation simple and clean.

---

8. Docker Architecture

A lightweight Dockerfile is included:

- Base image: node:18-alpine
- Copies project files
- Installs dependencies
- Exposes port 5173
- Runs Vite dev server using --host

This makes the project container-friendly and easy to run in any environment.

---

9. Design Decisions & Trade-Offs

Why Tailwind?

Faster development, responsive utilities, minimal CSS management.

Why Recharts?

Simple charting library that integrates well with React.

Why static data?

Assignment does not require backend; mock data keeps it simple.

Why Vite?

Faster build and dev environment compared to CRA.

---

10. Conclusion

The Warehouse Automation Dashboard meets all assignment requirements:

- Robot overview

- Task management
- Inventory management
- Charts
- Responsive design
- Docker support
- Clean code structure
- Clear documentation
