🤖 AI AGENT PROMPT — Supply-Demand Matching Platform UI Completion

CONTEXT & MISSION

You are completing the frontend of a **Supply-Demand Matching Platform** built as a hackathon prototype. A portion of the UI already exists. Your job is to:

1. **Analyse all existing code** before writing a single line
2. **Extract the exact design system** (colors, fonts, spacing, component patterns, CSS conventions)
3. **Build all missing pages and components** that match the existing aesthetic perfectly
4. **Wire up routing** without breaking anything that already works
5. **Never modify existing files** unless you are only adding an import/route entry

---

STEP 1 — MANDATORY ANALYSIS (DO THIS FIRST)

Read every file in the project before proceeding. Specifically extract:

From `BusinessRoom/BusinessRoom.jsx` + `BusinessRoom.css`:

- Primary, secondary, accent color hex values
- Background colors (page bg, card bg, sidebar bg)
- Border radius values used on cards, buttons, inputs
- Box shadow styles
- Button variants (primary CTA, secondary, danger/destructive)
- Font family and font-weight scale
- Spacing rhythm (padding/margin multiples)
- Any CSS custom properties / variables defined
- Animation or transition patterns
- Icon library being used (e.g. lucide-react, react-icons)

From `OrganisationDashboard/index.jsx`, `OrgProfile.jsx`, `OrgDescription.jsx` + `OrganisationDashboard.css`:

- Layout structure (sidebar width, top-nav height, content area)
- Card component anatomy
- How data is displayed (labels, values, badges)
- Navigation pattern (how active states look)
- Any reusable component patterns

From `Supply/Supply.jsx`, `Supply.css`, `Map.jsx`:

- Form field styling (inputs, selects, textareas)
- How listings are rendered (list vs grid, card structure)
- Empty state patterns
- Action button placement conventions

From routing config (`App.jsx` / `main.jsx` / `router.js` — whichever exists):

- Router library in use (React Router v6, TanStack Router, etc.)
- Existing route structure and path naming conventions
- How protected/authenticated routes are handled
- Layout wrapper pattern

Do not proceed to Step 2 until you have documented all of the above.

STEP 2 — ESTABLISH DESIGN TOKENS

After analysis, declare (mentally or in a shared file) the exact design system you found. All new components must use these exact values — no approximations, no new colors, no new fonts.

Example output of what you should have found (fill in actual values from code):

```
PRIMARY_COLOR: #??????
SECONDARY_COLOR: #??????
ACCENT_COLOR: #??????
BACKGROUND_PAGE: #??????
BACKGROUND_CARD: #??????
TEXT_PRIMARY: #??????
TEXT_SECONDARY: #??????
BORDER_COLOR: #??????
BORDER_RADIUS_SM: ?px
BORDER_RADIUS_MD: ?px
BORDER_RADIUS_LG: ?px
BOX_SHADOW: ?
FONT_FAMILY: '?????'
FONT_WEIGHT_NORMAL: ???
FONT_WEIGHT_SEMIBOLD: ???
FONT_WEIGHT_BOLD: ???
SPACING_UNIT: ?px
TRANSITION: ?
ICON_LIBRARY: ?
```

STEP 3 — PAGES & COMPONENTS TO BUILD

Build every item below. Each must be pixel-consistent with the existing design system.

3.1 AUTHENTICATION

`/register` — Organisation Registration Page

File: `src/pages/Auth/Register.jsx` + `Register.css`

Fields (all required unless noted):

- Organisation Name
- Email Address
- Password + Confirm Password (with show/hide toggle)
- Phone Number
- Address (Street), City, State, Country, Postal Code
- Description (textarea, optional)
- Website URL (optional)

UI requirements:

- Two-column layout on desktop (form left, hero/illustration right)
- On mobile: single column, hero hidden
- Inline validation errors below each field
- "Already have an account? Sign in" link at bottom
- Submit button shows loading spinner while request is in-flight
- On success: redirect to `/dashboard`

`/login` — Organisation Login Page

File: `src/pages/Auth/Login.jsx` + `Login.css`

Fields:

- Email Address
- Password (with show/hide toggle)

UI requirements:

- Centered card layout, same width as a modal (~480px max)

- "Forgot password?" link (route to `/forgot-password`, stub page is fine)
- "Don't have an account? Register" link
- Submit button with loading state
- Error banner for invalid credentials (not inline field errors)
- On success: redirect to `/dashboard`

3.2 DEMAND MANAGEMENT

Demand mirrors Supply exactly in structure and styling. Use the Supply page as the direct reference.

`/demand` — Demand Listings Page

File: `src/pages/Demand/Demand.jsx` + `Demand.css`

Mirrors `Supply.jsx` completely. Differences:

- "Supply" → "Demand" in all labels
- Fields: Name, Description, Max Price Per Unit, Currency, Quantity, Quantity Unit, Required By Date, Delivery Location, Category
- No supplier contact fields (supply-specific)
- Delete triggers by listing name (same as supply)

Actions per listing card:

- **Edit** — opens inline edit form (same pattern as Supply if it has one, otherwise a modal)
- **Delete** — confirmation prompt, then delete
- **Find Matching Supplies** — triggers AI search, navigates to `/match-results?type=demand&id={demand_id}`

3.3 AI SEMANTIC MATCH RESULTS

`/match-results` — Match Results Page

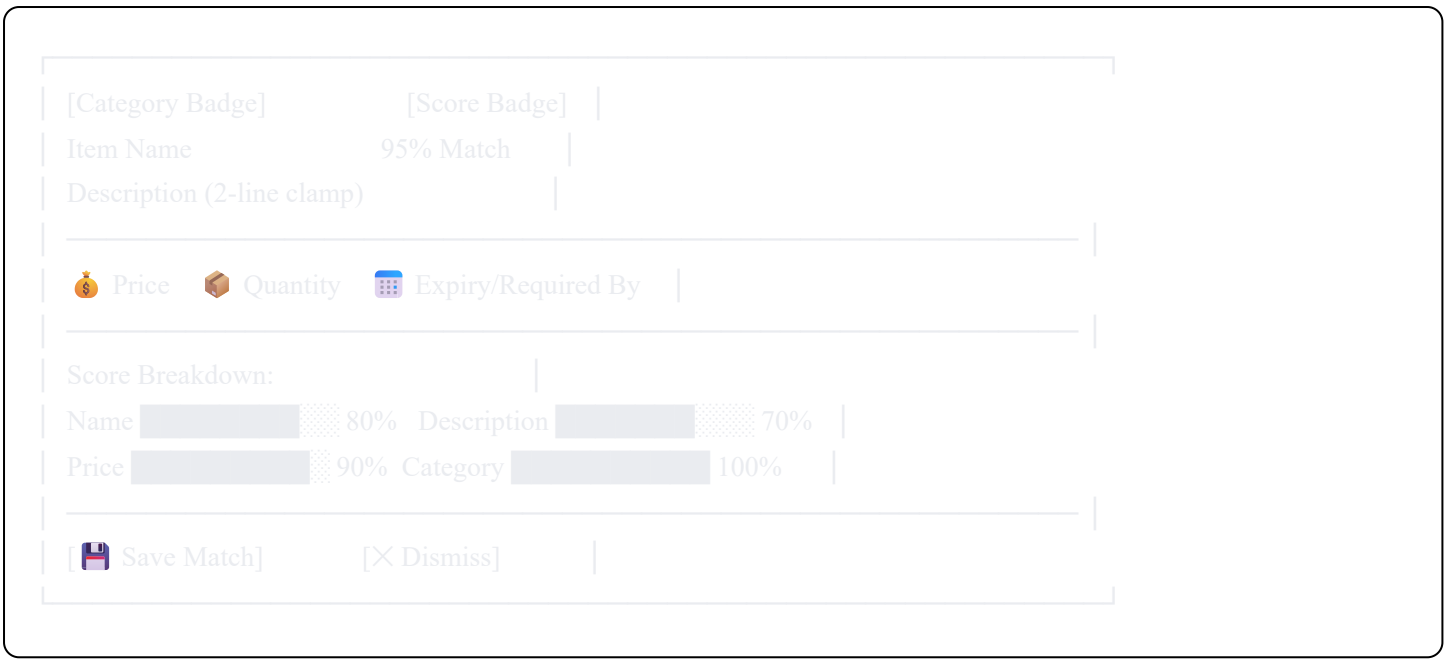
File: `src/pages/MatchResults/MatchResults.jsx` + `MatchResults.css`

Query params: `?type=supply|demand&id={listing_id}`

Layout:

- Left panel (30%): The source listing card (the supply or demand you searched from). Clearly labelled "Searching From". Non-interactive, display only.
- Right panel (70%): Scrollable list of matched results

Match Result Card (each matched listing):



States:

- **Loading** — skeleton cards (3 of them) while AI processes
- **Empty** — "No matches found" illustration + message
- **Error** — retry button
- **Saved** — card border changes to success color, "Saved ✓" button replaces Save
- **Dismissed** — card slides out / fades out of list

Confidence score badge color:

- ≥ 80%: green/success
- 60–79%: amber/warning
- < 60%: red/danger

After saving a match → a "Send Request" button appears on the saved card.

3.4 REQUEST MANAGEMENT

`/requests` — Requests Dashboard

File: `src/pages/Requests/Requests.jsx` + `Requests.css`

Two tabs: **Sent Requests** | **Received Requests**

Request Card:

Status badge colors:

- Pending: amber
- Accepted: green
- Rejected: red
- Cancelled: grey

Reject flow: clicking Reject expands an inline textarea for rejection reason, then a confirm button.

On Accept → immediately navigate or show link to the newly created Business Room.

3.5 BUSINESS ROOM ENHANCEMENTS

The BusinessRoom component exists. Extend it with the following **without touching existing code structure**:

`/rooms` — Business Rooms List Page

File: `src/pages/BusinessRoom/RoomList.jsx`

Grid of Room Cards:

Statuses: Active, In Progress, Deal Closed (success), Deal Failed (muted/red)

`/rooms/:roomId` — Individual Business Room

This already exists as `BusinessRoom.jsx`. Ensure routing points here.

Add to the existing room UI (in a sidebar panel or header area, matching existing layout):

- Room status display with current status badge
- "Mark as Deal Success" button (green, prominent) — only if status is not already closed
- "Mark as Deal Failed" button (red, outlined) — only if status is not already closed
- Confirmation modal before status change
- Link to QR/barcode once deal is closed

3.6 DEAL & BARCODE

`/deals` — Deals Overview

File: `src/pages/Deals/Deals.jsx` + `Deals.css`

Table or card list of all finalised deals:

- Supply name ↔ Demand name
- Partner organisation
- Agreed price × quantity = total value
- Currency
- Deal status badge
- Date finalised
- [View Barcode] button → navigates to `/deals/:dealId/barcode`

`/deals/:dealId/barcode` — Barcode / QR Page

File: `src/pages/Deals/DealBarcode.jsx`

Layout:

Verify QR Code sub-section (below the above, or separate route `/verify`):

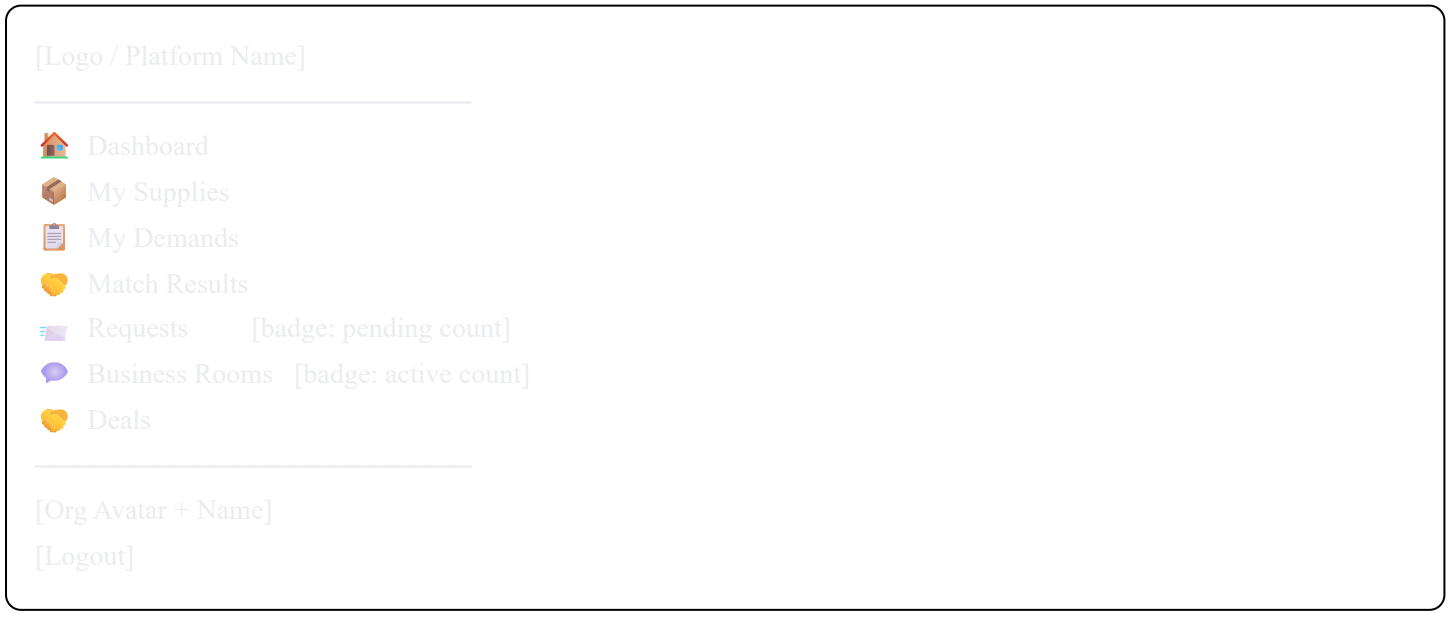
- Text input to paste a QR payload string
- OR a "Scan QR" button (use a QR scanning library or just the text input for prototype)
- On verify: show ✅ Verified / ❌ Invalid result with details

3.7 NAVIGATION & LAYOUT

Global Layout Wrapper

File: `src/components/Layout/AppLayout.jsx`

Sidebar navigation (matches existing dashboard layout):



- Active nav item uses the same highlight style as existing OrganisationDashboard nav
- Sidebar collapses to icon-only on medium screens, hidden with hamburger on mobile
- Top bar on mobile shows logo + hamburger

`/dashboard` — Dashboard/Home

File: `src/pages/Dashboard/Dashboard.jsx`

Stats overview cards:

- Active Supplies (count)
- Active Demands (count)
- Pending Requests (count)
- Open Business Rooms (count)
- Completed Deals (count)

Recent activity feed below cards (last 5 notifications in chronological order).

Quick-action buttons: "+ New Supply", "+ New Demand", "View Requests"

3.8 NOTIFICATIONS

Notification Bell (in AppLayout top-right)

File: `src/components/Notifications/NotificationBell.jsx`

- Bell icon with red badge showing unread count
- Dropdown panel on click showing last 10 notifications
- Each notification: icon by type, title, message, time ago
- "Mark all as read" button
- "View all" link → `/notifications`

`/notifications` — Full Notifications Page

File: `src/pages/Notifications/Notifications.jsx`

Full list with filter tabs: All | Unread | Requests | Rooms | Deals

---

STEP 4 — ROUTING

Update the router (App.jsx or equivalent) with these routes. **Do not remove or modify existing routes.**

```
/                    → redirect to /login (if unauth) or /dashboard (if auth)
/login               → Login
/register            → Register

/dashboard           → Dashboard          [PROTECTED]
/supply              → Supply (existing)  [PROTECTED]
/demand              → Demand             [PROTECTED]
/match-results       → MatchResults       [PROTECTED]
/requests            → Requests           [PROTECTED]
/rooms               → RoomList           [PROTECTED]
/rooms/:roomId       → BusinessRoom       [PROTECTED]
/deals               → Deals              [PROTECTED]
/deals/:dealId/barcode → DealBarcode      [PROTECTED]
/notifications       → Notifications      [PROTECTED]

/verify              → QR Verify (public) — anyone can verify a QR
```

Protected route wrapper: if no auth token in localStorage/context, redirect to `/login`.

STEP 5 — COMPONENT CONVENTIONS

Follow these rules strictly for every new component:

1. **CSS files are co-located** with their component (same folder, same name). Do NOT use Tailwind or inline styles unless the existing code already uses them.
2. **CSS variables** — if the existing code defines CSS custom properties (e.g. `--primary-color`), use those exact variable names everywhere.
3. **No new dependencies** unless the existing package.json already includes them. Exception: if a QR code renderer is needed, you may add `qrcode.react`.
4. **Component structure** mirrors existing files: same import order, same functional component pattern, same prop naming style.
5. **Loading states** use the same spinner/skeleton pattern as any existing loading state in the codebase.
6. **Error states** display the same way as any existing error handling.
7. **Empty states** follow the same pattern — look for any "no data" screen in existing components.
8. **Form validation** — if the existing code uses a form library (react-hook-form, formik), use the same one. If vanilla, stay vanilla.
9. **API calls** — match the existing HTTP client pattern (axios instance, fetch wrapper, etc.). Use the same base URL config.
10. **File/folder naming** — match exactly: PascalCase components, camelCase utilities, kebab-case for route paths.

STEP 6 — WORKFLOW CORRECTNESS

Ensure the full user journeys work end-to-end in the UI:

Journey A: Supply → Match → Request → Room → Deal → QR

1. Org A creates a Supply listing (`/supply`)
2. Org A clicks "Find Matching Demands" → goes to `/match-results?type=supply&id=X`
3. Org A sees ranked results, saves a match
4. Org A clicks "Send Request" on saved match → sends request
5. Org B sees request in `/requests` Received tab → Accepts
6. Both orgs see new room in `/rooms`
7. They chat in `/rooms/:roomId`
8. Either org marks "Deal Success"
9. QR code auto-displayed at `/deals/:dealId/barcode`
10. Either org downloads or verifies QR

Journey B: Demand → Match → Request

Same as above but starting from Demand listing and searching for Supplies.

CRITICAL CONSTRAINTS

- ❌ Do NOT modify `BusinessRoom.jsx`, `BusinessRoom.css`, `OrganisationDashboard/*`, `Supply.jsx`, `Supply.css`, `Map.jsx`
- ❌ Do NOT introduce a new CSS framework if the project doesn't already use one
- ❌ Do NOT change existing route paths
- ❌ Do NOT add a new state management library if one isn't already used
- ✅ DO add new routes to the existing router config
- ✅ DO create new CSS files for each new page
- ✅ DO reuse any shared components you find (buttons, cards, modals, badges)
- ✅ DO extract shared components into `src/components/` if you find yourself repeating patterns across 2+ new pages
- ✅ DO make every new page responsive (mobile-first)

OUTPUT CHECKLIST

Before considering this task complete, verify:

- [ ] All 12 routes are registered and navigable
- [ ] Auth flow (register → login → dashboard) works with redirects
- [ ] Supply CRUD + AI search trigger works (existing, just verify routing)
- [ ] Demand CRUD + AI search trigger works
- [ ] Match results display with confidence scores and save/dismiss
- [ ] Request send, receive, accept, reject flows work
- [ ] Business Room list and individual room accessible
- [ ] Deal Success / Failed marking works from room
- [ ] QR code displayed on deal success
- [ ] QR verification page functional
- [ ] Notification bell shows in layout
- [ ] All new pages match the existing color palette, fonts, and component style exactly
- [ ] No existing file has been broken or deleted

This prompt was generated to guide an AI coding agent in completing the Supply-Demand Matching Platform frontend. All design decisions must be derived from analysing the existing codebase first.