SmartStock – Schedule

Team: JUME (Max, Jay, Erfan, Usman)

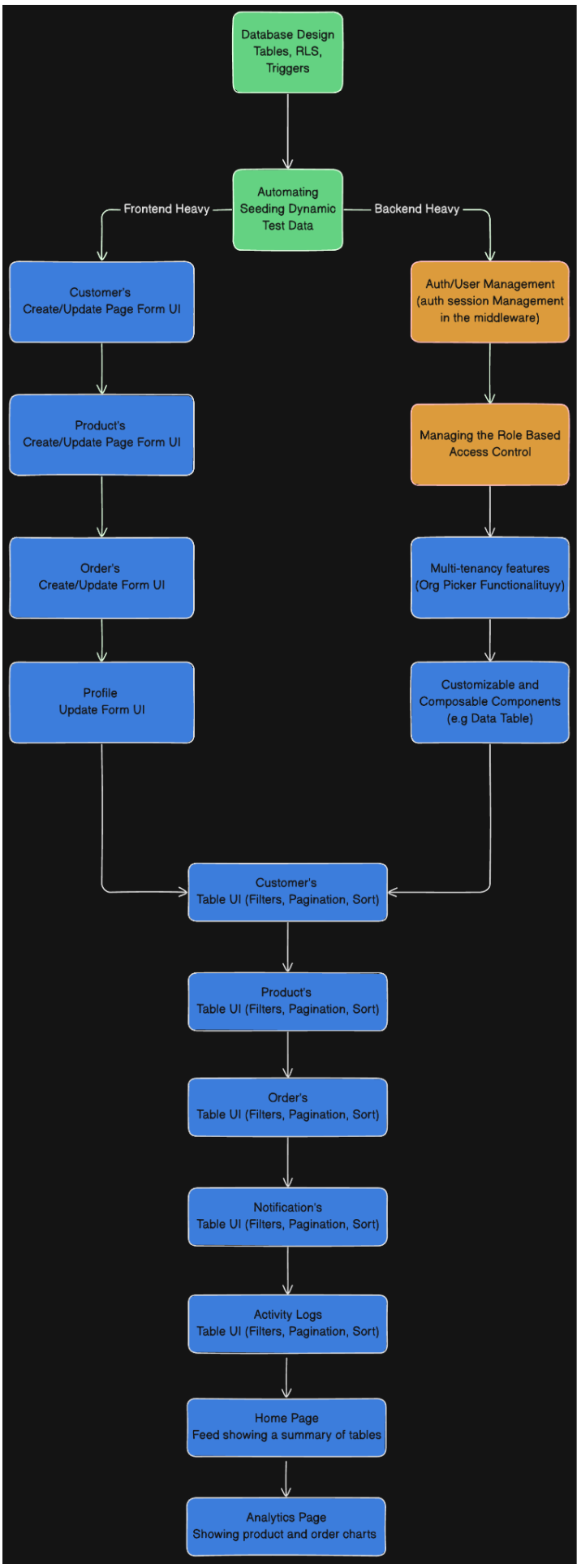
Task Dependencies and Critical Path

Key Dependencies Identified:

1. **Authentication, Database, and Multi-Tenancy Setup**
    - The initial setup of Supabase Auth, role-based access, and multi-org database seeding formed the foundation for all modules.
    - Delays in RLS policies, seed data, or schema creation directly impacted the start of downstream features like Orders, Products, and Activity Logs.
2. **Sadman Table Convention Integration**
    - To support real-time filtering, pagination, and scoped queries, the table infrastructure had to be completed early.
    - Many UI components—especially Orders and Products—relied on this convention being stable before functional development could proceed.

Critical Path

The critical path was defined by backend infrastructure: setting up authentication, seeding demo orgs, implementing RLS, and integrating Sadman Table conventions. Without this backbone, UI progress across modules like Customers, Orders, and Analytics would have stalled. Ensuring this backend base was operational early was crucial for unblocking frontend parallel work.


Database Design
Tables, RLS,
Triggers
Automating
Seeding Dynamic
Test Data
Frontend Heavy
Backend Heavy
Customer's
Create/Update Page Form UI
Product's
Create/Update Page Form UI
Order's
Create/Update Form UI
Profile
Update Form UI
Auth/User Management
(auth session Management
in the middleware)
Managing the Role Based
Access Control
Multi-tenancy features
(Org Picker Functionalityy)
Customizable and
Composable Components
(e.g Data Table)
Customer's
Table UI (Filters, Pagination, Sort)
Product's
Table UI (Filters, Pagination, Sort)
Order's
Table UI (Filters, Pagination, Sort)
Notification's
Table UI (Filters, Pagination, Sort)
Activity Logs
Table UI (Filters, Pagination, Sort)
Home Page
Feed showing a summary of tables
Analytics Page
Showing product and order charts

Keeping the Sprint on Schedule

1. **Proactive Communication:**
    - Regular standups helped surface blockers around Supabase permissions and syncing.
    - Max led the backend infrastructure and ensured that the rest of the team had access to real-time data and seeded environments, allowing independent progress.
    - Peer debugging—especially around Activity Logging and Notification Triggers—was critical in resolving tricky Supabase-side issues and syncing bugs.
    - **Discord** was the central hub for daily collaboration, async bug solving, and sharing UI previews, schema updates, and architecture diagrams.
    - Outside of scheduled sessions, team members checked in informally, helping each other with RLS errors, pagination bugs, and role-based visibility issues.
2. **Parallel Tasking:**
    - While backend elements like multi-tenancy logic and real-time analytics queries were in development, team members advanced sections with fewer dependencies.
    - Progress continued on:
        - Static layouts and placeholder logic for Orders, Products, and Customers
        - Activity Log UI and Profile Editing
        - Analytics page charts and dashboards using dummy data
        - Sidebar and routing enhancements for final polishing

---

Challenges & Lessons Learned

Challenges:

- Setting up Supabase RLS policies correctly took more time than expected, creating delays for real-time testing across roles.
- Analytics integration was slightly slowed due to the need to derive aggregate queries scoped per organization.
- Having one core backend owner (Max) meant temporary bottlenecks during schema design and seeding.

Lessons Learned:

- **Backend infrastructure should be finalized at the start** of the sprint to avoid team-wide stalls.
- **More thorough planning of Supabase roles, policies, and functions** in the first sprint would have smoothed Sprint 2 development.
- **Design and backend teams should be decoupled**—future work will benefit from separating responsibilities clearly between UI layout and backend logic setup.

On Being Behind Schedule

While the sprint goals were completed, some tasks—such as edge-case validation and minor UI polish—were compressed toward the end due to upstream delays.

What helped avoid overruns:

- Clear fallback priorities (e.g., Profile Edit, Activity Log)
- Focusing on low-dependency areas when blocked
- Having seeded data early enough to test and demo core flows

---

Conclusion

Sprint 2 of SmartStock was a pivotal phase where the database and real-time infrastructure were solidified, enabling multi-tenant functionality and secure org-level access. Despite a few timing challenges with backend setup, the team delivered a complete, stable MVP across all major modules: Orders, Products, Customers, Home Dashboard, and Analytics.

The team’s adaptive workflow, cross-support debugging, and structured communication ensured steady progress. Lessons around dependency planning and backend-first task structuring will guide the approach for Sprint 3, especially as SmartStock scales with more advanced features and user roles.