Saltify Brainstorming Summary

■ PRODUCT DIRECTION

1. Saltify will offer both drag-and-drop and code-first experience.
2. Forms and Landing Pages can be created via GUI or by switching to 'SaltScript' mode.
3. SaltScript = Next.js + TypeScript + TailwindCSS + JS logic — all in a React-like DSL.
4. Builder UI will allow switching back and forth between GUI and SaltScript seamlessly.
5. AI-based Service Bot will allow prompt-based generation of branded, dynamic content.

■ MULTI-TENANT SAAS CAPABILITIES

1. System is designed to be multi-tenant from the start.
2. Each tenant (business) will have its own isolated resources, configurations, and branding.
3. Forms, landing pages, automation flows will be tenant-specific.

■ FLEXIBLE DATA MODEL (DATA EXTENSIONS + ATTRIBUTES)

1. Tenants can create 'Data Extensions' (DE) which are free-form tables.
2. Each DE can have many 'Attributes' (fields). These attributes are selected or created globally.
3. Attributes are globally unique per tenant to enable consistent data capture.
4. Each form/landing page must link to a DE to determine data schema.
5. Submissions to forms will insert rows in DE tables.

■ PROSPECT OBJECT — UNIFIED CUSTOMER PROFILE

1. All DE submissions are mapped to a single Prospect object.
2. Prospect is uniquely identified via: Email OR Mobile OR External ID OR Unique ID.
3. Prospect is the universal identity object across DEs and workflows.

■ SUBSCRIPTION + FEATURE FLAGGING

1. Feature access will be governed by subscription tier.
2. Feature flags will be used to enable/disable capabilities per tenant.
3. Will support monthly/annual plans, trials, usage-based billing.

■ NEXT STEPS

- Finalize SaltScript parser/compiler design
- Implement Form Builder tabs: Details, Fields, Styling
- Integrate AI Service Agent into onboarding/building flow
- Architect Data Extension/Attribute/Prospect DB layer
- Add layout/theme manager for brand control