MVP B SOW: Detail

Scope of Work

Project: Results Roofing Online Quote & Homeowner Portal: MVP B
Client: Results Roofing
Vendor: CB.Media
Version: 1.0

1. Project Overview

Results Roofing is building a web-based product that allows homeowners to:

1. Enter their address and basic details
2. Receive a roof quote based on automated roof measurements
3. Select a package (good / better / best or add-ons)
4. Book an appointment
5. Sign an agreement
6. Pay a deposit online
7. Access a portal to monitor status, view documents/photos, and make additional payments

This SOW describes the scope, technical architecture, integrations, and delivery plan for MVP B, which includes:

- Measured quote and deposit flow (MVP A+ scope)
- Homeowner portal
- Advanced analytics with server-side tracking
- Basic dashboards
- Reliability mechanisms (synthetic checks, simple SLOs)
- Experimentation hooks and first A/B tests

2. Objectives and Success Criteria

2.1 Business Objectives

- Provide a differentiated, measurement-driven online quote and deposit experience.
- Reduce time spent on manual estimation for eligible jobs.
- Reduce inbound “what is happening with my job” calls by giving homeowners a self-service portal.
- Establish a tracking and analytics foundation that supports performance marketing.

2.2 Success Criteria at Launch (MVP B)

- A homeowner can complete the online funnel (address → quote → booking → sign → deposit) on mobile and desktop without human intervention in the happy path.
- A homeowner can log into a portal, view job status, view documents and photos, see payments versus balance, and make additional payments.

- Core events (`quote_started`, `quote_completed`, `deposit_paid`, `portal_login`, `payment_made`) are reliably captured in GA4 via sGTM, with events available for use in ad platforms (Meta CAPI, Google where applicable).
- Synthetic checks run the main paths and alert when the experience is broken.
- Internal teams can view a basic dashboard showing funnel performance and online revenue.

3. Scope

3.1 In Scope

- Design and build of web-based quote funnel and homeowner portal under resultsroofing.com (subdomain or subpath).
- Integration with a single roof measurement provider (GAF or equivalent).
- Integration with JobNimbus (or designated CRM/project system) for job records.
- Integration with DocuSign for agreements (single primary template).
- Integration with Stripe for deposit and subsequent payments.
- Implementation of GA4 + server-side GTM + Conversions API for key events.
- Basic Looker Studio (or similar) dashboard for funnel and revenue KPIs.
- Implementation of synthetic checks and alerting for core flows.
- Basic set of feature flags / experimentation hooks and 1–2 initial A/B tests.
- First-pass accessibility and localization support (not full compliance).

3.2 Out of Scope / Phase 2

- Support for multiple measurement vendors and complex routing across them.
- Advanced pricing rules per carrier, region, or complex coverage logic beyond agreed MVP rules.
- Native mobile applications.
- Full accessibility compliance (for example WCAG 2.1 AA with exhaustive testing).
- Full multi-language localization for all copy and portal screens.
- Extensive custom reporting per department beyond the agreed initial dashboards.
- Deep two-way sync with JobNimbus beyond agreed create/update use cases.

4. Assumptions

- Results Roofing will provide access to required third-party platforms (JobNimbus, Stripe, DocuSign, measurement provider, analytics accounts) and designate technical contacts for each.
- Measurement provider exposes an HTTP API to request roof reports and retrieve them asynchronously by job ID or similar identifier.
- JobNimbus offers an API or integration mechanism that supports creating and updating job records and attaching basic metadata.
- Stripe will be used in hosted modes (Checkout and/or Payment Links) rather than fully bespoke payment UIs in MVP.
- Portal authentication will use email-based login or password-based login; single sign-on is not required in MVP.
- A small set of packages (three tiers or equivalent) will be defined by Results Roofing before implementation.

5. Technical Architecture Overview

5.1 High-Level Components

- **Front-end web application**
    - Responsive web application (SPA or SSR) implementing funnel and portal UI.
    - Runs under resultsroofing.com (for example `quote.resultsroofing.com` or `/quote`).
- **Backend application**
    - API endpoints for the front-end to manage leads, quotes, measurements, jobs, users, and payments metadata.
    - Integrations with external systems: measurement provider, JobNimbus, Stripe, DocuSign, analytics (sGTM endpoint).
    - Background job processing for asynchronous tasks (measurement requests, webhooks).
- **Data storage**
    - Application database (for example PostgreSQL or equivalent) to store internal entities: leads, quotes, measurements, users, jobs, events metadata.
- **Analytics stack**
    - GA4 property and web/measurement protocols.
    - Server-side GTM endpoint for event ingestion and transformation.
    - Conversions API for Meta and other platforms.
- **Monitoring and reliability**
    - Synthetic check framework to exercise main flows.
    - Logging and basic alerting for critical paths.

This SOW is written in a framework-agnostic way. The implementation team will select specific frameworks (such as React/Next.js for front-end and Node/TypeScript or similar for backend) consistent with the vendor's standard stack.

6. Integrations and Interface Contracts

This section defines the expected behavior. Actual vendor-specific field names may differ; the adapter will handle mapping.

6.1 Measurement Provider Integration

Responsibility: Request roof measurement reports based on homeowner address and store the result for pricing.

Key operations:

1. **Create measurement request**
    - Input:
        - `request_id` (internal UUID)
        - `address_line1`
        - `address_line2` (optional)
        - `city`
        - `state`
        - `postal_code`
        - `country`
        - `homeowner_name`
        - `homeowner_email` (optional for provider, always stored internally)
    - Behaviour:
        - Backend sends request to provider API and stores status as `pending`.
        - Provider returns a `provider_measurement_id` or equivalent.
2. **Poll / receive measurement**
    - Polling or webhook-based mechanism to retrieve measurement result.
    - Expected output stored internally:
        - `provider_measurement_id`
        - `status` (`completed`, `failed`, `not_found`)
        - `roof_area_sqft`
        - `perimeter_length` (if available)
        - `story_count` (if available)
        - `pitch` (if available)
        - Raw provider payload as JSON for debugging.
3. **Timeout and fallback**
    - If measurement is not available within a configured time window (for example N minutes), backend marks status as `timeout` and triggers manual fallback logic.
    - Manual fallback will still capture quote intent and hand off to internal team.

6.2 JobNimbus Integration

Responsibility: Keep JobNimbus in sync at key steps.

Key interactions:

- **Create Job/Contact** when:
    - A homeowner completes the flow up to booking (appointment confirmed).
- **Update Job** when:
    - Deposit is paid.
    - Measurement status is updated (for internal notes).

Data mapping (minimum):

- Contact: name, email, phone, address.
- Job:
    - External `quote_id`
    - Measurement details (summary)
    - Selected package and price
    - Deposit amount and status
    - Flags for `manual_estimate_required` if fallback path was used.

Implementation will use JobNimbus API or alternative integration method as agreed, with concrete endpoints and field mappings specified during Phase 0.

6.3 Stripe Integration

Responsibility: Process deposits and subsequent payments.

Approach:

- Use Stripe Checkout or Payment Links for payments to keep PCI scope minimal.
- Use Stripe webhooks to confirm successful charges.

Metadata requirements:

Each payment should include:

- `quote_id`
- `customer_email`
- `type` (`deposit` or `additional_payment`)
- `amount`

On successful webhook, backend will:

- Mark deposit or payment as `paid` in internal DB.
- Update JobNimbus job with payment status.
- Emit `deposit_paid` or `payment_made` events to analytics.

6.4 DocuSign Integration

Responsibility: Handle agreement signing.

Approach:

- Use a single primary template for the MVP.
- Pre-fill template with homeowner details, address, package, and total price.

Flow:

- Backend creates an envelope request when homeowner reaches “sign” step.
- Homeowner is redirected to DocuSign signing ceremony.

- On completion, DocuSign sends webhook or the application polls for envelope status.
- Backend updates quote status to `signed` and stores link or reference to executed document.

6.5 Analytics (GA4, sGTM, Conversions API)

Components:

- GA4 web measurement via gtag or GTM.
- Server-side GTM endpoint to receive events and forward to GA4 and Meta CAPI.

Event flow:

- Front-end sends events to sGTM with standardized payload.
- sGTM transforms and forwards to GA4 and Meta.
- Minimal PII; identifiers to be hashed where required.

7. Data Model

This is a logical model; physical schema may vary.

7.1 Entities

1. **User**
    - `id` (UUID)
    - `email`
    - `password_hash` or equivalent (if password-based auth)
    - `name`
    - `created_at`, `updated_at`
2. **Lead**
    - `id` (UUID)
    - `user_id` (nullable; may be pre-login)
    - `first_name`, `last_name`
    - `email`
    - `phone`
    - `address_line1`, `address_line2`, `city`, `state`, `postal_code`, `country`
    - `source` (utm_source, utm_medium, utm_campaign, referrer)
    - `created_at`
3. **Quote**
    - `id` (UUID)
    - `lead_id`
    - `measurement_id` (nullable until completed)
    - `status` (`pending_measurement`, `ready`, `signed`, `closed`, etc.)
    - `selected_package` (`good`, `better`, `best` or custom identifier)
    - `price_total`
    - `deposit_amount`
    - `appointment_datetime`
    - `manual_estimate_required` (boolean)

- `jobnimbus_job_id` (nullable)
- `created_at`, `updated_at`

4. **Measurement**
   - `id` (UUID)
   - `provider_measurement_id`
   - `status` (`pending`, `completed`, `failed`, `timeout`, `not_found`)
   - `roof_area_sqft`
   - `perimeter_length`
   - `story_count`
   - `pitch`
   - `raw_payload` (JSON)
   - `requested_at`, `completed_at`

5. **Payment**
   - `id` (UUID)
   - `quote_id`
   - `stripe_payment_intent_id`
   - `amount`
   - `currency`
   - `type` (`deposit`, `additional_payment`)
   - `status` (`pending`, `succeeded`, `failed`)
   - `created_at`, `updated_at`

6. **Job**
   - `id` (UUID)
   - `quote_id`
   - `jobnimbus_job_id`
   - `status` (`scheduled`, `in_progress`, `completed`, etc.)
   - `created_at`, `updated_at`

7. **Document**
   - `id` (UUID)
   - `job_id`
   - `type` (`agreement`, `scope_of_work`, `other`)
   - `title`
   - `url`
   - `uploaded_at`

8. **Photo**
   - `id` (UUID)
   - `job_id`
   - `title`
   - `url`
   - `uploaded_at`

8. Event Taxonomy

Events are emitted from front-end to sGTM (and sometimes from backend). Each event includes timestamp, user/session identifiers, and relevant properties.

8.1 Core Events

1. **`quote_started`**
    - Trigger: When homeowner begins the quote process after entering an address.
    - Properties:
        - `quote_id`
        - `lead_id`
        - `postal_code`
        - `source`, `utm_source`, `utm_medium`, `utm_campaign`
        - `device_type` (`mobile`, `desktop`, `tablet`)

2. **`measurement_requested`**
    - Trigger: When backend sends measurement request to provider.
    - Properties:
        - `quote_id`
        - `measurement_id`
        - `postal_code`

3. **`measurement_completed`**
    - Trigger: When measurement is retrieved successfully.
    - Properties:
        - `quote_id`
        - `measurement_id`
        - `roof_area_sqft` (binned or range if needed)
        - `duration_seconds`

4. **`quote_completed`**
    - Trigger: When homeowner reaches pricing summary before payment.
    - Properties:
        - `quote_id`
        - `selected_package`
        - `price_total`

5. **`deposit_paid`**
    - Trigger: Successful deposit payment via Stripe webhook.
    - Properties:
        - `quote_id`
        - `amount`
        - `currency`
6. **`portal_login`**
    - Trigger: Successful login to portal.
    - Properties:
        - `user_id`
7. **`payment_made`**
    - Trigger: Successful subsequent payment via portal.
    - Properties:
        - `quote_id`
        - `amount`
        - `type` (`additional_payment`)
8. **`measurement_timeout_fallback`**
    - Trigger: When measurement does not complete and system triggers manual fallback.
    - Properties:
        - `quote_id`
        - `timeout_seconds`

9. Functional Requirements

9.1 Quote Funnel

- Responsive UI for the entire funnel from address entry to payment.
- Steps:
    - Address and contact details
    - Automated measurement request (asynchronous)

    - Package selection (good / better / best or add-ons)
    - Appointment booking
    - Agreement signing (DocuSign)
    - Deposit payment (Stripe)
    - Confirmation page with summary

- Fallback behavior:

    - If measurement fails or times out, system informs user that an estimator will complete quote manually and finish the booking flow, collecting contact details and appointment preferences.

9.2 Homeowner Portal

- Login and account creation (email-based).

- Dashboard that shows:

    - Job status in clear stages
    - Documents list (agreement, key docs)
    - Photos list (pre, during, post job)
    - Payment history and remaining balance
- Ability to make additional payments using Stripe.

9.3 Admin / Internal Functionality (MVP level)

- Internal view is not a full admin panel in MVP. Assumed management via:

    - JobNimbus for full job management.
    - Dashboard views via Looker Studio for performance.

10. Non-Functional Requirements

10.1 Performance

- Primary funnel pages must load first meaningful content within 3 seconds on a typical 4G mobile connection.

- Actions that trigger measurement requests must return a UI response (loading state) within 1 second.

10.2 Reliability

- Synthetic checks must run the following flows on a schedule (for example every 5–15 minutes):

    - Happy path: quote → deposit.

    - Portal login and basic page load.

- If checks fail consecutively more than a configured threshold, alerts must be sent to designated contacts.

10.3 Security

- All traffic must be served over HTTPS.
- Secrets and keys stored in secure environment configuration, not in source control.
- Rate limiting applied to sensitive endpoints (login, quote creation) to mitigate abuse.

10.4 Browser Support

- Fully supported: latest two major versions of Chrome, Safari, Edge, and Firefox.
- Mobile support: Safari on iOS and Chrome on Android (recent major versions).

10.5 Accessibility

- Basic keyboard navigability and screen-reader-friendly structure for primary flows.
- Colors and contrast should meet reasonable readability standards, but full WCAG audit is not part of MVP.

11. Delivery Plan and Phase Breakdown (with Acceptance Criteria)

Phase 0 – Kickoff and Discovery (10–14 hours)

- Finalize tech stack, environments, and dev workflows.
- Confirm all third-party accounts and access.
- Lock measurement provider, payment provider, e-sign provider, and JobNimbus integration approach.
- Deliverable: Discovery document summarizing decisions and confirmed flows.

Acceptance criteria:

- All vendors and credentials identified and accessible.
- Primary user flows and constraints are documented and signed off.

Phase 1 – UX and Architecture (24–32 hours)

- Create wireframes and prototype for funnel and portal.
- Produce system architecture diagram and data model outline.
- Define integration patterns and sequence diagrams for key flows.

Acceptance criteria:

- UX prototype reviewed and approved.

- Architecture and data model documented, with dev team sign-off.

Phase 2 – Foundations and Enablers (20–28 hours)

- Set up repo(s), environments, CI/CD pipeline, and basic monitoring tools.
- Implement baseline security headers and performance budgets.
- Create GA4 property, GTM containers, and sGTM endpoint skeleton.
- Implement data layer format for core events.

Acceptance criteria:

- Ability to deploy to staging and production environments.
- Events from a test page visible in GA4 via sGTM.

Phase 3 – Core Funnel and Portal Build (78–98 hours)

- Implement front-end and backend endpoints for the full funnel.
- Implement measurement adapter, async job handling, and fallback behavior.
- Implement homeowner portal UI and API: status, docs, photos, payments.
- Integrate with JobNimbus, Stripe, DocuSign as per scope.

Acceptance criteria:

- A test user can complete the full funnel on staging (using test environments for third-party systems).
- A test user can log into portal and see a mocked job with documents and payments.

Phase 4 – Analytics, Reliability, and Hardening (24–32 hours)

- Implement full event mapping for core events.
- Set up dashboards for funnel and revenue KPIs.
- Implement synthetic checks and alerting.
- Harden endpoints and refine error handling.

Acceptance criteria:

- Dashboards show real test data for key events and revenue.
- Synthetic checks run and trigger alerts when intentionally broken in staging.

Phase 5 – UAT, Training, and Content (10–16 hours)

- Execute UAT scenarios.
- Refine copy for errors, disclosures, and portal content.
- Deliver training sessions and simple runbooks.

Acceptance criteria:

- All high-priority UAT issues addressed.
- Internal stakeholders trained and sign off on UAT.

Phase 6 – Launch and Stabilization (10–14 hours)

- Deploy to production.
- Monitor behavior and fix critical issues.
- Adjust timeouts, fallbacks, and error messaging based on live usage.

Acceptance criteria:

- Stable production behavior over agreed stabilization period.
- No critical bugs open impacting core flows.

Phase 7 – Enhancements and Experiments (4–6 hours)

- Implement feature flags and A/B hooks.
- Launch 1–2 initial tests (for example CTA copy, package layout).
- Capture backlog items for Phase 2.

Acceptance criteria:

- At least one experiment live in production.
- Phase 2 backlog documented.

12. Risks and Mitigation

- **Measurement provider API limitations**
    - Mitigation: validate capabilities in Phase 0; if inadequate, adjust scope or provider.
- **Third-party integration latency or outages**
    - Mitigation: implement timeouts and clear fallback flows; monitor error rates.
- **JobNimbus API changes or limitations**
    - Mitigation: minimize reliance on advanced features; isolate integration logic.
- **Data privacy and consent**
    - Mitigation: implement consent messaging for tracking; limit PII in analytics payloads.