APRIL 2026

PROPOSAL

Web Platform Development
Accessible Travel & Leisure Database

A structured, scalable platform designed for discoverability, accessibility, and long-term content management. Built from real systems, not templates.

This proposal is grounded in platforms I have already built: structured directories, filter-heavy search systems, map-led interfaces, and editor-friendly CMS tools that make large content libraries manageable over time.

5	24+	3,000	8–9
RELEVANT PLATFORMS	SYSTEMS BUILT	ENTRY SCALE TARGET	WEEKS ESTIMATED

ABOUT

I build platforms and the systems that keep them usable.

SSTIEM is a one-person studio focused on complete product systems: the public interface, the data architecture underneath it, and the operational tooling that allows a team to keep the platform accurate after launch. I do not work from themes or site builders. I build **custom structures that match the shape of the problem**, especially when content, filtering, and long-term maintainability matter.

For your travel and leisure database, that matters because this is not just a website. It is a **living content product** with thousands of entries, multiple filtering paths, editorial workflows, accessibility considerations, and a public experience that must stay coherent as the database grows. The strongest fit for this brief is not a visual designer alone or a backend engineer alone. It is someone who can design the system from both directions at once.

RECORD LOGIC

The destination record has to be strong enough to power listing views, map behaviour, search states, and detail pages without duplicating content across the platform.

DISCOVERY DESIGN

Category browsing, regional navigation, accessibility filters, and the map all need to work as one joined discovery system rather than separate interface features.

EDITORIAL RELIABILITY

Editors need structured forms, permissions, and clear update paths so the database stays trustworthy as more entries, regions, and contributors are added.

Those three decisions determine whether the platform still feels simple after the first thousand records. If the record model, the discovery logic, or the editorial workflow is weak, the public experience becomes harder to trust and the internal team inherits unnecessary manual work.

BRIEF ALIGNMENT

How each RFP requirement maps to proven capability

YOUR REQUIREMENT	HOW THIS PROPOSAL ADDRESSES IT
3,000+ Entries	PostgreSQL relational model with dedicated tables for entries, categories, regions, and attributes. Proven at directory scale in World of Doors (74+ auto-generated pages) and LeadGen (40+ modules processing multi-source records into one clean dataset).
Multi-Filter Search	API-driven combinable filters with persistent, shareable state and sub-500ms response. Tradezyx proves this at 18 filter dimensions. The same architecture powers your category, geography, and accessibility filtering.
Interactive Map	Leaflet or Mapbox with clustered markers, click-to-detail routing, and results synced to the list view. ATX Notary demonstrates this as the primary product surface, not a decorative add-on.
CMS & User Roles	Structured editorial forms with field validation, tiered permissions, and preview workflows for non-technical teams. AVLUX proves this at 30+ content models across 4 user roles with clear editorial governance.
Scalability	Architecture where growth means adding records and taxonomy, not redesigning pages. World of Doors generates 74+ pages from one clean data source without any manual page-building process.
Multilingual	Content model structured to support locale-aware fields and language-specific routing from the data layer, so multilingual capacity activates without a platform rebuild or migration.

PLATFORM REQUIREMENTS

What this platform has to do from day one

A strong outcome here depends on more than launching a polished public interface. The platform has to support **clear discovery for visitors, structured publishing for editors, and dependable system logic underneath both**. If any one of those layers is weak, the product becomes harder to trust and harder to maintain as it grows.

PUBLIC DISCOVERY

Search, filters, and map behaviour that stay intuitive

Visitors should be able to move between category browsing, geographic exploration, accessibility filters, and destination detail without losing context or restarting the journey.

EDITORIAL CONTROL

A CMS shaped around real destination data

Your team needs structured forms for place details, certifications, media, accessibility features, and status updates so content quality remains consistent across thousands of entries.

SHARED LOGIC

One source of truth behind every public view

Listings, map markers, search states, and destination pages should all reflect the same record structure so the product feels coherent rather than stitched together.

LONG-TERM SCALE

Growth without operational drag

As the database expands, growth should come from adding records, categories, and regions, not from redesigning pages or introducing one-off exceptions into the system.

PORTFOLIO

Relevant systems already in production

WORLD OF DOORS

Scalable Directory Platform

A multi-city directory generating 74+ structured pages from a category and region model. New records create new public pages automatically, turning growth into a data operation rather than a design bottleneck.

DIRECTORIES • STRUCTURED CONTENT

ATX NOTARY PLATFORM

Map-Driven Service Finder

An interactive Leaflet interface with filterable markers, profile routing, and a role-based admin panel. The map is not decorative; it is one of the primary ways the product is understood and used.

MAPS • FILTERING • ROLES

AVLUX

Visual CMS Engine

A modular CMS with 30+ schema models, structured field types, drag-and-drop editing, and admin tooling designed so non-technical teams can confidently manage complex content.

CMS • CONTENT ARCHITECTURE

LEADGEN PLATFORM

Data Enrichment Pipeline

A record-processing engine that ingests from multiple sources, normalises data, resolves duplicates, and scores completeness before publishing clean entries for search and export.

DATA • QUALITY • SCALE

TRADEZYX

Multi-Filter Analytics Platform

A 39+ microservice platform with combinable real-time filtering across structured datasets. Users can layer category, region, status, and feature filters with sub-500ms response while administrators control access, data views, and configuration.

REAL-TIME FILTERING • SCALABILITY • ACCESS CONTROL

Together, these projects cover the main demands of your brief: **large-scale structured content, editorial workflows, geographic discovery, multi-filter search, and an admin experience that remains usable after launch**. The proposal below is based on those working patterns rather than hypothetical capability claims.

TECHNICAL APPROACH

How I would build your platform

COMPONENT	APPROACH
CMS	Custom headless CMS shaped around the actual editorial job: destination records, accessibility features, location data, media, certifications, and supporting notes. Editors get clear structured forms rather than a generic page builder that invites inconsistency.
Data Model	PostgreSQL schema with dedicated tables for entries, categories, regions, accessibility attributes, certifications, and related content. The goal is a relational structure that remains understandable and fast as the platform grows beyond the initial 3,000-entry target.
Filtering	API-driven filtering that combines category, region, accessibility features, and keyword search without page reloads. Filter state stays shareable and persistent, which improves usability for members, staff, and public visitors alike.
Map	Leaflet or Mapbox with clustered markers, click-to-detail routing, and results synced to the same logic as the list view. The map and directory behave like one product rather than two disconnected discovery tools.
Accessibility	Semantic HTML, keyboard-first interaction patterns, screen reader support, clear focus states, and strong information hierarchy built into the system from the beginning rather than checked in at the end.

In practical terms, this means a platform that is easier to trust because the public experience, the editorial workflow, and the system architecture all reinforce the same goal: **clear information, dependable records, and growth without operational clutter**. The delivery sequence below follows that same logic.

PHASE 1

Foundation

Weeks 1–2

Content model, taxonomy, accessibility needs, filter logic, and map behaviour defined before interface build begins.

PHASE 2

Core Build

Weeks 3–5

CMS backend, API layer, search engine, entry templates, and role-based editorial workflow.

PHASE 3

Interface

Weeks 6–7

Public UI, responsive behaviour, synced map/list views, and accessibility review across the main journeys.

PHASE 4

Launch

Weeks 8–9

Testing, content import support, deployment, editor training, handover documentation, and launch stabilisation.

UNDERSTANDING

This is not a website project. It is a structured information product.

Your brief describes a platform that has to do several things at once: store complex destination data, make it discoverable through multiple paths, support an editorial team that grows over time, and present everything with the kind of clarity that earns long-term trust. That combination is what makes this project interesting — and it is also what makes it difficult to get right without the right foundations.

I have built every part of this kind of system before — not in a single project, but across five separate platforms that each tested a different piece of the same puzzle. **A large-scale structured directory. A map-led discovery system. An editorial CMS for non-technical teams. A data pipeline that cleans and scores incoming records. A filter system that handles complex search without breaking.** Your brief brings all of those pieces together, and I understand how they connect because I have already built each one independently.

01

Structured Data at Scale

The foundation of this platform is the record model. If destination entries, categories, accessibility features, and regional taxonomy are not structured correctly from the start, every surface built on top becomes harder to trust and harder to maintain.

02

Editorial Usability

A CMS is only useful if the team actually wants to use it. I build editorial interfaces shaped around the real job — structured forms, clear workflows, field validation — not generic page builders that invite inconsistency at scale.

03

Discovery That Connects

Filtering, search, map, and category browsing should behave as one joined system. When a visitor moves between those paths, the experience should feel coherent — not like switching between disconnected features.

I do not approach this as someone who builds websites. I approach it as someone who builds the system that makes the platform trustworthy — the data model, the editorial workflow, and the public interface, designed as one connected product.

HOW I WORK

A thinking partner, not just execution

ARCHITECTURE FIRST

I define the system shape early so content, search, and admin workflows are stable before visual polish takes over. Structural decisions made in week one determine whether the platform still feels clear in year two.

INDEPENDENT & ACCOUNTABLE

I work autonomously, with visible progress. Regular checkpoints and working prototypes at each phase mean the project never disappears into a black box or drifts from your expectations.

FULL OWNERSHIP FOR YOU

Hosting, code, data, and deployment remain yours. No proprietary lock-in, no hidden platform dependency. You own the product and can evolve it on your own terms.

PRACTICAL DELIVERY

I optimise for working software — clear editorial tools, dependable system logic, and a platform your team can keep using without developer dependency after handoff.

I work as a one-person studio because it means every architectural decision, every editorial interface choice, and every delivery milestone has the same person behind it. There is no handoff risk between what is planned and what gets built. That matters most on projects where the data model, the public interface, and the CMS have to reinforce each other — which is exactly what your brief requires.

WHY THIS MATTERS FOR YOUR PROJECT

The brief asks for more than a public-facing website. It asks for a **durable content product** that can store complex destination information, make it discoverable in multiple ways, and remain manageable for editors over time. That is why this proposal emphasises CMS design, data architecture, and filter behaviour as strongly as the visual interface. Those are the decisions that determine whether the platform still feels clear, trustworthy, and usable a year after launch.

I want to build something your team is proud of and your users actually rely on — *not a short-term launch asset that becomes fragile once the content base expands.*

If selected, I would approach this as a long-term destination information product from the first week of work. The strongest version of this platform is one where clarity, usability, and scalability are designed in from the beginning — not added later.

INVESTMENT & DELIVERABLES

Three approaches depending on depth, refinement, and future needs

This project is best approached as a structured platform build rather than a standard website. The complexity lies in the data architecture, filtering system, CMS usability, and long-term scalability. To keep things flexible, I have outlined three tiers.

Core Platform

$4,500 – $5,500

A complete, functional platform with all core requirements in place.

DELIVERABLES

- ✓ Structured content system (entries, categories, regions)
- ✓ Multi-parameter filtering & basic search
- ✓ Interactive map with location markers
- ✓ Detail pages for each entry
- ✓ CMS with admin/editor roles
- ✓ Media and document upload support
- ✓ Responsive layout (desktop, tablet, mobile)
- ✓ Deployment and hosting setup

RECOMMENDED

Full Platform

$6,000 – $8,000

The best balance of functionality, usability, and long-term scalability.

EVERYTHING IN CORE, PLUS

- ✓ Refined filtering UX & navigation structure
- ✓ Map interaction synced with filters
- ✓ Scalable architecture for ~3,000 entries
- ✓ CMS workflows for non-technical users
- ✓ Multilingual-ready structure
- ✓ UI refinement & layout consistency
- ✓ System optimisation for maintainability

Extended Platform

$8,500 – $10,000

Designed for long-term growth beyond the initial launch.

EVERYTHING IN FULL, PLUS

- ✓ Advanced filtering interactions & enhanced UX
- ✓ Map clustering & richer interactions
- ✓ Content validation & structured data
- ✓ Groundwork for ratings & reviews
- ✓ Preparation for certification workflows
- ✓ Community contribution readiness
- ✓ Additional performance considerations

Ongoing Support — Hosting, infrastructure management, monitoring, bug fixes, and platform stability. Does not include new feature development. $300 – $500 / mo

HOW THE PROJECT RUNS

Four phases — each with a checkpoint before moving forward

PHASE 1

Alignment & Discovery

Weeks 1 – 2

We define scope, priorities, and success criteria together. You review the proposed structure and confirm direction before any development begins.

PHASE 2

Build & First Review

Weeks 3 – 5

Core functionality is developed and shared as a working prototype. You walk through the system, test workflows, and provide structured feedback.

PHASE 3

Refinement & Feedback

Weeks 6 – 7

The platform is refined based on your input — UI, usability, editorial experience. A polished version is presented for look, feel, and interaction review.

PHASE 4

Delivery & Handoff

Weeks 8 – 9

Final testing, deployment, CMS walkthrough, and documentation. A final review ensures everything is complete and aligned before handoff.

PAYMENT STRUCTURE

30 – 50%	30 – 40%	Remaining
Upfront at project start	After core system completion	Upon final delivery

WHAT TO EXPECT

COMMUNICATION

Regular check-ins throughout each phase. You will always know where the project stands, what is being worked on, and what is coming next.

REVISIONS

Minor revisions are included within each phase. Larger scope changes can be discussed and scoped separately so the timeline stays clear.

COLLABORATION

This is a partnership, not a handoff. Every phase includes a dedicated review so we stay aligned on priorities, direction, and quality.

The goal is to build a system that is *clear, maintainable, scalable, and usable long-term* — not something that only works at launch.

Thomas — SSTIEM — sstiem.com — sstiem.pro@gmail.com

CASE STUDY 1 OF 5 — WORLD OF DOORS

Scalable Directory Platform

A structured directory system that turns one clean database into city pages, service pages, and landing pages without manual page building.

PROJECT OVERVIEW

A directory that grows by adding records, not by redesigning pages

World of Doors was built as a regional service directory spanning multiple cities and service categories. Instead of designing each page by hand, the platform is driven by a structured model that connects **entry data, category data, and location data** to reusable templates. That means the public experience stays consistent even as the dataset expands.

For the user, it feels curated and editorial. For the operator, it behaves like a publishing engine: add a new service, assign its category and region, and the platform knows where it belongs. That separation between **content structure and page presentation** is what makes it relevant to a large directory brief.

The system was designed from the beginning to treat each entry as a **reusable record** rather than a standalone page. That architectural choice is what lets the directory generate city landing pages, category browse pages, and detail pages from the same underlying data without duplicating content or introducing inconsistency as the database grows.

WHO IT SERVES

People browsing by city, service type, and local need who want a clear route from discovery to detail.

CORE BEHAVIOUR

Structured records feed repeatable templates so the directory expands in a controlled, predictable way.

OPERATIONAL OUTCOME

New regions and categories can be introduced without rebuilding the product or breaking navigation consistency.

74+	50	6	0
PAGES GENERATED	CITIES SERVED	SERVICE CATEGORIES	CODE CHANGES TO REBRAND

HOW GROWTH WORKS

Record Drives Output

One structured entry automatically appears in every relevant city page, category listing, and browse path without manual placement.

Template Inheritance

Public pages are generated from reusable templates that enforce layout consistency as the directory scales beyond the original scope.

Taxonomy Expansion

New categories, cities, and groupings can be added as content operations without design work or codebase changes.

What Users Experience

- Browse by **city, category, or intent** through a clear regional information structure.
- Land on **consistent detail pages** with service descriptions, features, and contact routes.
- Move through the directory without feeling like each page was built separately or patched together over time.
- Use a mobile-friendly experience that still preserves the hierarchy and clarity of a larger desktop directory.
- Navigate between city views, category views, and individual detail pages using the same **intuitive information hierarchy**.

What Admin Controls

- Create and edit entries through **structured forms** rather than manually formatted pages.
- Assign category, region, and supporting metadata once so the record appears everywhere it belongs.
- Manage descriptions, media, and taxonomy from one system without fragmenting content across tools.
- Rebrand the presentation layer without rebuilding the underlying directory logic.
- Monitor publishing activity and **track which categories and regions** are growing fastest.

WHAT MAKES THE DIRECTORY SCALABLE

The important decision was to treat the directory as a **structured record system**, not a collection of manually designed pages. Once the listing, category, and regional relationships were defined cleanly, the public experience could stay consistent even as the number of pages increased. That is what turns directory growth from a design burden into a manageable publishing workflow. The same logic that powers 50 cities today could support hundreds without changing the underlying architecture.

That principle — structure the record correctly, and the platform handles the rest — is exactly what a large-scale travel directory depends on. It is also why this project is the closest architectural parallel to what your brief requires.

STRUCTURED ONCE

Each listing is entered once, then reused across regional, category, and destination views instead of being rebuilt page by page.

CONSISTENT EVERYWHERE

The same content rules keep landing pages aligned as the directory expands into more places, categories, and editorial paths.

READY TO EXPAND

New destination groupings, service types, or branded variations can be introduced without reworking the entire front-end structure.

How the directory expands without losing clarity

The directory only remains manageable if one listing can drive many outputs without duplication. In this build, the listing itself stays separate from the templates that render city pages, category views, and detail pages. That separation is what turns expansion into a controlled publishing process instead of ongoing manual page production.

From a client perspective, that means growth stops being a design bottleneck. The team can focus on improving records and taxonomy, while the system takes responsibility for rendering the right pages in the right places with consistent presentation.

In practice, this architecture means the directory publishing workflow scales independently of the design workflow. New content follows the same rules as existing content, so quality stays predictable even as **multiple contributors, new categories, and additional geographic areas** are introduced over time.

What One Listing Holds

- A core profile with title, description, contact details, imagery, and supporting editorial context.
- Category and regional assignments so the record knows which landing pages, result sets, and browse paths it belongs inside.
- Optional metadata that can later shape filters, trust signals, badges, or supporting page sections without rebuilding the platform.
- A structured base that keeps content reusable instead of locking each update into one manually edited page.

What The Platform Generates

- A consistent destination page built from the same template logic every time, which protects the public experience from fragmentation.
- Placement inside the right city and category views without editors needing to manually rebuild the same information in multiple places.
- A directory that can grow by adding records, expanding taxonomy, and refining rules rather than redesigning the site for each new location.
- A publishing model that stays useful even when the content library becomes much larger than the original launch set.

DIRECTORY GENERATION FLOW

Once the taxonomy is in place, every new record can appear in the right views automatically. That reduces publishing effort, improves consistency, and keeps growth operationally sustainable.

KEY DESIGN DECISIONS

Three architectural choices made this system commercially useful rather than technically impressive:

Single Source of Truth

Every listing lives in one place. City pages, category views, and detail pages all read from the same structured record, eliminating sync issues and editor confusion.

Template Separation

Presentation logic is fully decoupled from record data. The directory can be reskinned, white-labelled, or restructured without touching the content database.

Taxonomy-Driven Routing

URLs, navigation, and browse paths are all generated from the category and region taxonomy. Adding a new city creates new routes automatically.

RFP CONNECTION

Your accessible travel database will need the same pattern demonstrated here: a structured entry feeding multiple public views (country pages, category listings, destination detail) while keeping editorial effort concentrated in one place. This project proves that model works at scale.

Rapid Expansion

New cities or destination types can be added as content, not as bespoke design work, which keeps the platform commercially flexible.

Consistent Structure

Public pages stay coherent because every listing follows the same content rules, hierarchy, and presentation logic.

White-Label Ready

Brand presentation can shift without undoing the directory system that powers the content underneath it.

WHY THIS MATTERS FOR YOUR PLATFORM

Your accessible travel database needs to handle **thousands of destinations across categories and regions** without turning content management into manual page production. This project proves the exact principle your brief depends on: if the record structure is right, the public platform can scale cleanly. The same pattern that generated 74 regional directory pages here can support **thousands of travel entries, filtered pathways, and destination detail pages** while keeping editorial effort controlled and the public experience consistent. For your brief, that same logic is what keeps country views, category views, and individual destination pages aligned as the database becomes broader and more detailed. It also means your team can focus on improving content quality rather than managing page production, which is where the real long-term value of an information product lives.

CASE STUDY 2 OF 5 — ATX NOTARY PLATFORM

Map-Driven Service Platform

An interactive map connected to live structured data so users can discover nearby services, compare options, and move directly into contact or booking.

PROJECT OVERVIEW

A location-first interface where the map is the product, not an add-on

This platform was built for a service category where relevance depends on **place, availability, and trust**. The public experience opens on an interactive map, and each marker represents a real record in the system. Users can filter by service type, area, and other attributes, then move from a map result into a full profile with contact details and supporting information.

What makes the platform stronger than a simple directory is the way the map, data model, and admin system work together. Discovery is fast for the public, while the back office stays controlled through clear roles, structured content, and a workflow that keeps the location layer accurate.

The map is not an add-on feature attached at the end. It is the **primary product surface**. The entire information architecture was designed so that geographic coordinates, service metadata, profile data, and administrative controls all feed the same interactive layer. That makes the map feel like a dependable discovery tool rather than a visual decoration.

WHO IT SERVES

People who need to find the right provider in the right area quickly, with enough context to make a confident decision.

CORE BEHAVIOUR

Map markers, filters, profile pages, and admin editing all connect to the same structured location data.

OPERATIONAL OUTCOME

Teams can manage a geographically distributed service network without losing control over permissions or content quality.

<800ms	3	13	10
LOAD TIME	USER ROLES	ADMIN MODULES	DASHBOARD VIEWS

HOW DISCOVERY WORKS

Map-First Entry

Users begin with a geographic view. Clustered markers resolve into individual providers as users zoom in or filter by service type.

Filter Refinement

Service type, area, and availability filters narrow the map in real time without reloading the page or losing the user's zoom level.

Marker to Detail

Clicking a marker opens a brief summary, then links directly into a full profile page with contact information and service details.

What Users Experience

- Open an **interactive map** that responds to real filters rather than a static location image.
- Filter by service, area, and supporting attributes without losing context or needing to restart the search.
- Click a marker and move directly into a **detail page** with profile information and action routes.
- Use the same journey on mobile with an interface that keeps discovery fast and understandable.
- Compare multiple providers by location, availability, and **supporting trust signals** within the same map session.

What Admin Controls

- Manage providers, areas served, content, and supporting records from a **13-module dashboard**.
- Assign role-based access so administrators, editors, and staff each see the tools they actually need.
- Update listings through structured forms that protect consistency across the public map and profile pages.
- Maintain oversight with permission boundaries and a cleaner operational workflow than ad hoc map plugins allow.
- Review **dashboard views** that track provider distribution, coverage gaps, and content freshness across the network.

WHY THE MAP STAYS IN SYNC

Location is treated as a **real field in the record**, not a decorative layer added at the end. Because the map, listing views, and profile pages all read from the same structured data, users can move between them without losing context or seeing conflicting information. That is what makes the map feel like a dependable discovery tool rather than a visual extra. It also means editorial updates propagate immediately — change a location record once, and every surface that references it reflects the change.

MAP-LED DISCOVERY

Users begin with place and then move naturally into comparison, profile review, and action without switching mental models.

ONE LOCATION RECORD

Coordinates, service area, listing content, and profile detail stay tied to the same source record instead of drifting apart.

OPERATIONAL TRUST

Editors and administrators can update the network confidently because the public map and the managed data stay aligned.

SYSTEM DETAIL

How the map, listings, and admin system stay aligned

Location only becomes a real product advantage when it is embedded in the record itself. In this platform, geographic coordinates, service area, profile data, and supporting filters all belong to the same structured entity. That is why the map feels reliable: it is not a visual layer sitting on top of the data model, it is one expression of the data model.

That matters because location-based products usually break when the map, the listing view, and the admin panel drift apart. Here, they stay anchored to the same record logic, which is what keeps the experience believable for both the public and the team maintaining it.

From a governance standpoint, the three-role structure means administrators control the system shape, editors maintain content quality, and staff can use the platform operationally without accidentally modifying records. That layered approach keeps the **location data trustworthy** as more people interact with the system.

What One Provider Record Contains

- Geographic coordinates, service area, profile content, and structured attributes used by search and map behaviour.
- Media and supporting details that appear consistently on both the marker-driven browse layer and the full detail page.
- Operational status fields so the platform can stay current without requiring editors to update multiple disconnected surfaces.
- A clean source record that makes map-led discovery feel as trustworthy as a traditional list or directory view.

What The Platform Keeps Aligned

- Marker placement, listing results, and profile routes all update from the same source record so users do not encounter mismatched information.
- Editors, administrators, and staff work with different permissions without breaking the publishing flow or confusing operational responsibility.
- The public map remains a genuine discovery tool rather than a decorative feature layered on after the rest of the platform is built.
- The same logic can support future member views, internal dashboards, or regional expansions without rethinking the location model.

ROLE STRUCTURE

ROLE	ACCESS LEVEL	CAPABILITIES
Administrator	Full	System configuration, user management, module oversight, exports, and operational settings across the entire platform.
Editor	Content	Add and maintain provider records, update media, manage supporting details, and keep listings accurate and current.
Staff	Operational	View dashboards, check records, follow status changes, and use the system without accessing sensitive controls or editing capabilities.

KEY DESIGN DECISIONS

Map as Record View

The map renders directly from structured records, not from a separate geo layer. That keeps markers, profiles, and the admin panel permanently aligned.

Progressive Disclosure

Users see clustered markers first, then detailed pins, then summary cards, then full profiles. Each step adds detail without overwhelming.

Permission Boundaries

Role separation means data quality improves as the team grows because each person has a clear operational scope.

RFP CONNECTION

Your travel database needs map-led discovery where users can browse destinations geographically, filter by accessibility features, and move into detailed destination pages. This project proves that exact pattern: interactive map, structured data underneath, and an editorial system that keeps the location layer trustworthy at scale.

Place-Led Discovery

Users can browse geographically without losing the richness of the underlying record data or the clarity of the journey.

Controlled Publishing

Location records stay accurate because editing is structured, permissions are separated, and the data model is consistent.

Operational Clarity

The same product can support public browsing, internal coordination, and future expansion without splitting into separate tools.

WHY THIS MATTERS FOR YOUR PLATFORM

Your brief calls for an **interactive map with clickable markers** connected to structured filtering and detailed destination pages. This project demonstrates that exact pattern: map-led discovery, live data underneath it, and an editor workflow that keeps the location layer trustworthy. For an accessible travel database, the map is not just decorative. It becomes one of the most intuitive ways for users to understand where destinations are, how they compare, and which places match their needs. It also proves that map browsing can sit inside a governed editorial system rather than operating as a separate plugin or one-off interface. The same pattern supports future regional expansion, partner overlays, and accessibility-filtered map views without rebuilding the core location model.

CASE STUDY 3 OF 5 — AVLUX PLATFORM

Visual Content Management Platform

A CMS built for people who are responsible for content, operations, and publishing quality, not for people who want to think like developers.

PROJECT OVERVIEW

A content system designed around editors, structure, and repeatable control

AVLUX began as a product for a specific hospitality context, then expanded into a **full modular CMS** that supports content editing, SEO structure, media handling, operations, and reporting. The turning point in the project was recognising that most CMS tools are optimised for technical flexibility, while most real teams need **editorial clarity**.

This system was therefore built around structured fields, predictable roles, and interface patterns that make sense to managers, editors, and business owners. The goal was not to expose every possibility. It was to create a content environment where the right actions are obvious and the wrong ones are difficult.

What makes this CMS distinct is its understanding that editorial tools fail when they give teams too many options without enough structure. The system enforces **content schemas, field validation, and role-based governance** so that publishers can focus on improving information quality rather than managing formatting, permissions, or consistency by hand.

WHO IT SERVES

Content editors, managers, and operators who need to maintain a large structured site without developer intervention.

CORE BEHAVIOUR

Structured models, clear field types, and role-aware editing create a CMS that stays usable as content grows.

OPERATIONAL OUTCOME

Publishing becomes safer, faster, and more consistent because the system enforces good structure by default.

30+	9	4	12
CONTENT MODELS	FIELD TYPES	USER ROLES	ADMIN MODULES

EDITORIAL PHILOSOPHY

Schema-First Editing

Every content type has a defined schema. Editors fill structured fields rather than free-form pages, which protects consistency automatically.

Role-Aware Interfaces

Each user role sees only the controls they need. Editors see forms, managers see reporting, administrators see system configuration.

Validation by Design

Required fields, format rules, and relationship constraints catch errors before they reach the public site, not after.

What Editors Experience

- Structured forms with clear field intent so editors do not have to guess what belongs where.
- Reusable modules and media handling that make large pages feel manageable rather than intimidating.
- Preview and publishing flows that support confidence before changes go live.
- A consistent editing logic whether the team is managing a small catalogue or a much larger structured database.
- Clear **status indicators** showing which entries are published, in draft, or awaiting review.

What Admin Controls

- Content schemas, field relationships, and visibility rules that shape how the CMS behaves.
- Role management for administrators, editors, managers, and viewers with different levels of authority.
- Operational views that connect content management with reporting, KPIs, and broader business oversight.
- A system architecture that can support multiple brands or deployments without rebuilding the CMS from scratch.
- Configuration tools that let **non-developers adjust taxonomy, categories, and field rules** without touching code.

WHY EDITORS GET CLEANER RECORDS

The value of the CMS is not only that it stores content. It shapes how content is entered, validated, and published. By giving each record a **clear schema and field logic**, the platform reduces editorial drift and makes quality easier to maintain across a large content library. The result is a system that helps editors make stronger decisions instead of simply giving them more fields to fill in. That design principle — guide rather than expose — is what keeps the CMS reliable as the team and content base grow.

CLEAR EDITING LOGIC

Editors are guided by field intent and structured forms, which reduces hesitation and lowers the chance of inconsistent publishing.

STRUCTURED GOVERNANCE

Roles, permissions, and content relationships make quality control part of the system rather than a manual clean-up exercise.

REPEATABLE PUBLISHING

The same editorial model can support a small curated set or a much larger content operation without changing how the team works.

How the CMS protects quality as content grows

The value of a CMS like this is not simply that it can store entries. Its value is that it makes the right editing behaviour easier than the wrong one. Structured relationships, field rules, permissions, and publishing logic give teams a system they can trust even when the content base becomes much larger and more operationally complex.

In practice, that gives leadership, editors, and operators different levels of confidence at the same time. Managers know the model is governed, editors know the forms are understandable, and the public side benefits from cleaner, more dependable source material.

The system also protects against a common failure mode in content platforms: the gradual erosion of quality as more people contribute. By enforcing structure at the entry level, the CMS ensures that the **thousandth record is as clean and navigable** as the first, which is critical when content powers public discovery and trust.

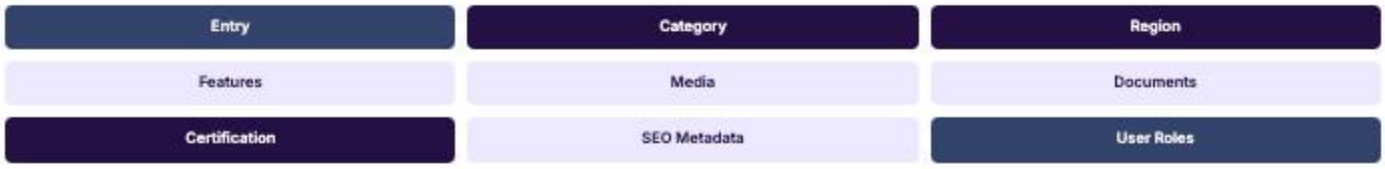
What One Entry Can Hold

- Core content, categories, regional assignments, and descriptive accessibility or feature data that shape how the public experience behaves.
- Media, documents, and certification material that need to stay attached to the same record instead of being scattered across tools.
- SEO and publishing metadata so the public output stays consistent without hidden formatting work behind the scenes.
- A record structure that remains useful whether the content set is small and curated or large and continuously updated.

What The CMS Enforces

- Field types and validation rules that prevent editors from entering inconsistent or incomplete information.
- Role-based permissions so contributors only see the controls they actually need to do their work safely.
- A repeatable publishing structure that makes large datasets easier to govern, review, and improve over time.
- A cleaner separation between content ownership, editorial workflow, and technical implementation.

CONTENT ARCHITECTURE



Those relationships matter because they let the CMS behave like an editorial product rather than a storage panel. The field types below are not technical decoration. They are what make the system understandable and repeatable for the people who maintain it every day.

SUPPORTED FIELD TYPES

Rich Text · Single Line · Image Upload · Document Upload · Dropdown Select · Multi-Select Tags · Date/Time · Map Coordinates · Boolean Toggle

KEY DESIGN DECISIONS

Schema Enforcement

Every content type has a schema that governs what fields are available, required, and validated. This prevents editorial drift as teams scale.

Progressive Disclosure

Editors see the right level of complexity for their role. Advanced fields and system configuration are only visible to administrators.

Multi-Brand Architecture

The same CMS engine can power multiple deployments with different branding, field sets, and publishing rules without forking the codebase.

RFP CONNECTION

Your platform needs a CMS where non-technical editors can manage thousands of destination entries with accessibility attributes, media, certifications, and regional assignments. This project proves that structured editorial environments keep data quality high as the team and content set grow.

Faster Publishing

Editors can add and update structured records without waiting on design or development support for routine content operations.

Cleaner Data

Field types and validation rules improve consistency across the whole content set, which matters more as the system scales.

Safer Permissions

Different users can contribute without all of them carrying the same publishing risk or administrative authority.

WHY THIS MATTERS FOR YOUR PLATFORM

Your platform needs a CMS that **non-technical editors can use with confidence** while managing thousands of destination entries, accessibility features, location information, media, and certification details. This project demonstrates exactly that kind of structured editorial environment. The field logic maps naturally to your entry schema. The role system maps to your administrator and editor model. Most importantly, it proves that the CMS can be treated as a **product for editors**, not just a backend storage tool. That matters because editorial confidence is what protects long-term data quality once more people, more destinations, and more publishing activity enter the system. For your brief, this is the most direct evidence that CMS design and editorial usability are treated as first-class priorities, not afterthoughts.

CASE STUDY 4 OF 5 — LEADGEN PLATFORM

Data Enrichment & Discovery Engine

A system that collects from many sources, resolves duplicates, scores quality, and turns inconsistent raw records into clean, searchable structured data.

PROJECT OVERVIEW

A record pipeline built to turn scattered inputs into one trustworthy dataset

This platform was built to solve a common but difficult data problem: important records exist across **many partial sources**, each with different structures, quality levels, and gaps. The system ingests raw leads from scraping, public APIs, file imports, partner feeds, and manual entry, then processes each record through a multi-stage workflow that makes the data usable.

What matters is not only ingestion. It is the quality layer afterwards: matching duplicates, filling missing attributes, applying confidence scores, and deciding when a record is ready to publish. The result is a database that does not just collect information, but **improves it as it moves through the system**.

The critical insight behind this system is that raw data has almost no value on its own. Its value comes from the processing steps between ingestion and publication: standardisation, entity resolution, quality scoring, and editorial review gates. Those steps are what turn **a pile of records into a credible information product** that people can search, compare, and trust.

WHO IT SERVES

Teams that need a reliable, searchable dataset rather than a pile of raw records from disconnected sources.

CORE BEHAVIOUR

Ingest, normalise, deduplicate, enrich, and score records before they appear in the public or internal experience.

OPERATIONAL OUTCOME

Editors and analysts spend less time cleaning data manually and more time working from a trusted base record.

40+	12	6	<2S
SERVICE MODULES	DATA SOURCES	ENRICHMENT STAGES	SEARCH LATENCY

HOW QUALITY WORKS

Source Normalisation

Incoming data from APIs, scrapers, and CSV imports is standardised into a common format before it enters the processing pipeline.

Entity Resolution

Duplicate records from different sources are matched and merged into a single clean entity using name, location, and attribute matching.

Confidence Scoring

Each record receives a quality score based on completeness, source reliability, and verification status so editors can prioritise review work.

What Users Experience

- Search across clean records using category, region, or quality filters without seeing the messy data sources underneath.
- Use **confidence signals** to understand whether a record is complete, verified, or still needs review.
- Export and share structured records once the platform has resolved duplicates and normalised formats.
- Benefit from a dataset that keeps improving as new source information arrives over time.
- Trust that the records presented have been **validated, deduplicated, and scored** before appearing in search results.

What Operators Control

- Manage data sources, pause feeds, add connectors, and adjust how incoming records enter the system.
- Set thresholds for completeness, match sensitivity, and publish readiness based on operational needs.
- Review duplicate logic and override edge cases where human judgment is needed.
- Target specific geographic regions or segments so the dataset stays aligned with business priorities.
- Monitor **pipeline health and throughput** to understand how quickly new records move from ingestion to publication.

WHAT HAPPENS BEFORE A RECORD GOES LIVE

The platform treats data quality as an active process, not a manual afterthought. Incoming records are standardised, checked against existing entities, improved where possible, and scored so editors can quickly tell which entries are ready for use. That quality layer is what separates a large raw dataset from a dependable information product. Without it, the database grows in volume but not in usability — which is exactly the failure mode that makes large directory products frustrating for both users and editors.

MANY SOURCES

The platform can absorb information from multiple source types without assuming those sources will agree with each other.

QUALITY BEFORE PUBLISH

Records move through validation, matching, and scoring before they become part of the trusted working dataset.

CONFIDENCE AT SCALE

As more data arrives, teams can still prioritize what is ready, what needs review, and what should not be published yet.

SYSTEM DETAIL

How the record pipeline improves what enters the database

The commercial value of this platform comes from what happens between ingestion and publication. Records do not simply arrive and appear. They move through a quality process that standardises formats, resolves duplication, enriches missing context, and assigns a readiness signal. That is what turns a large data operation into a credible information product.

That distinction matters commercially because once teams stop trusting the dataset, the public experience suffers quickly. A strong enrichment layer protects the product from becoming large but unreliable as new sources, categories, and geographies are added.

The pipeline also supports editorial oversight. Records do not move from ingestion to publication without passing through defined quality gates. That means editors always know the difference between a **verified entry and a pending one**, which is essential when the dataset powers a public-facing discovery experience.

What Comes Into The System

- Records arriving from APIs, scraped sources, CSV imports, partner feeds, and manual entry.
- Partial or conflicting information that often describes the same business, place, or entity in different formats.
- Geographic and category clues that help group information around one destination or business instead of leaving it fragmented.
- A constant stream of raw material that would otherwise require large amounts of manual clean-up before anyone could trust it.

What Gets Improved Before Publish

- Formats are normalised so names, addresses, and supporting fields become consistent enough to search, compare, and export cleanly.
- Duplicate records are merged into one cleaner entity instead of polluting the directory with near-identical entries.
- Additional signals and supporting attributes can be layered in to improve completeness and decision quality.
- Each record receives a quality indicator so editors know whether it is ready, partial, or still needs review.

RECORD ENRICHMENT PIPELINE

Ingest → Normalise → Deduplicate → Enrich → Score → Publish

This pipeline is what makes a large database usable. It gives teams a repeatable way to improve trust in the dataset instead of relying on manual clean-up every time new information arrives.

SUPPORTED DATA SOURCES

Public APIs · Web Scraping · CSV Import · RSS Feeds · Partner Feeds · Manual Entry · Government Data · Social Profiles · Maps / Geo · Directory Listings · Review Sites · Email Verification

KEY DESIGN DECISIONS

Quality Gates

Records must pass completeness and confidence thresholds before entering the public dataset. This prevents low-quality data from polluting the user experience.

Source Agnostic

The pipeline accepts data from any format or origin. New source connectors can be added without modifying the enrichment or scoring logic.

Continuous Improvement

Records are not static after publication. New source data can update existing entries, improving completeness over time without manual re-entry.

RFP CONNECTION

Your travel database will ingest destination information from tourism boards, certification bodies, partner directories, and user submissions. This project proves that multi-source data can be normalised, deduplicated, and scored into one trustworthy dataset that editors and users can depend on.

Duplicate Resolution

Many partial records can be merged into one cleaner entity instead of cluttering the platform or confusing users.

Quality Scoring

Confidence levels make it easier to separate ready records from entries that still need editorial review.

Geographic Precision

Data can be filtered, prioritised, and published based on place as well as category, source quality, or completeness.

WHY THIS MATTERS FOR YOUR PLATFORM

Building a database of accessible travel destinations means drawing information from tourism boards, certification schemes, partner directories, user submissions, and review ecosystems that all describe destinations differently. This project proves the exact capability your brief needs: ingest from many sources, resolve to one clean destination record, and expose quality signals that help editors trust what they are publishing. That is what makes a large directory credible rather than merely large. It also provides a model for keeping trust intact as source inputs change, expand, or conflict over time. For your platform, this means the initial 3,000-entry import and all future data additions follow the same quality process.

CASE STUDY 5 OF 5 — TRADEZYX PLATFORM

Multi-Filter Analytics Platform

A data-dense application where users combine filters in real time to narrow thousands of records into precise, useful answers in seconds.

PROJECT OVERVIEW

A platform where filtering is the main interaction model

Tradezyx was built for a domain where the value of the product depends on how quickly users can move from a large dataset to a very specific subset. The interface is designed so that **filters feel like navigation**: users refine by category, status, region, feature, or time range, and the system responds fast enough that the search process feels fluid rather than transactional.

That matters because large information products often fail at the moment of discovery. They contain useful records, but the interface makes them hard to reach. This platform solves that by treating response time, filter combinations, and access control as core product decisions rather than secondary technical concerns.

The filter system was designed as a **shared platform capability** rather than a front-end widget. The same filter logic powers public search, member views, editorial dashboards, and API integrations. That means every surface stays consistent, and new capabilities can be added without duplicating or fragmenting the search infrastructure.

WHO IT SERVES

Users who need to move through large structured datasets quickly and return to precise result sets without friction.

CORE BEHAVIOUR

Multiple filters can be combined in real time while counts, results, and access rules stay in sync.

OPERATIONAL OUTCOME

The same platform supports public browsing, saved searches, editorial control, and restricted access tiers.

39+	<500ms	6	18
MICROSERVICES	FILTER RESPONSE	ACCESS TIERS	COMBINABLE FILTERS

HOW FILTERING WORKS

Shared Filter State

All filters contribute to one unified state object. Adding or removing any filter immediately recalculates results without restarting the search.

API-Driven Logic

Filter behaviour lives in the API layer, not the front end. That means the same search logic works across public UI, dashboards, and third-party integrations.

Sub-500ms Response

The architecture is optimised for speed. Users can layer filters rapidly without waiting for each refinement to process before making the next one.

What Users Experience

- Apply multiple filters at once without page reloads or disjointed query steps.
- See counts and result sets update live, which makes the search process feel exploratory rather than brittle.
- Save filter combinations and return to them later, reducing repetitive search work for frequent users.
- Move from overview to rich detail pages without losing the context of how the result set was produced.
- Share filter states via URL so **saved searches and curated pathways** can be bookmarked or linked externally.

What Operators Control

- Define filters, labels, ordering, and visibility without rewriting the entire interface layer.
- Apply tier-based access so different users can see different levels of detail or functionality.
- Compose dashboards and analytical views from the same structured data powering search.
- Expose filter logic through the API so the system can support integrations and future extensions.
- Monitor **search usage patterns** to understand which filter combinations are most common and optimise relevance.

WHY SEARCH STILL FEELS MANAGEABLE

Every filter is treated as part of a connected state rather than a standalone query. That makes it possible to combine category, geography, accessibility signals, and keyword search while keeping the interface responsive. Users can keep refining their search without feeling like the platform is making them start over each time. That continuity is what makes a dense search product feel usable instead of exhausting. It also means users naturally discover more of the database because the cost of exploring is low.

CONNECTED FILTERS

Every search control contributes to one shared state, so refinement feels cumulative rather than fragmented.

IMMEDIATE FEEDBACK

Counts and results react quickly enough that the interface feels exploratory, which is crucial in large data environments.

GOVERNED ACCESS

The same core filter logic can support public users, members, editors, and integrations without splitting into separate systems.

SYSTEM DETAIL

How the filter model stays fast, useful, and commercially flexible

A filter-heavy product only becomes valuable when speed, clarity, and access rules all work together. This platform treats filters as a structured system rather than a front-end convenience. That is why search remains responsive, why the same logic can power dashboards and APIs, and why different user groups can see different layers of the same dataset without splitting the product into separate experiences.

The practical advantage is that the product can serve multiple audiences without multiplying search systems. Public users, members, editors, and integrations can all rely on the same filter intelligence while seeing different levels of depth and control.

Access control is not bolted on as a restriction layer. It is **designed into the filter model** so that each user tier receives an appropriate experience. Public visitors see high-level discovery, members get deeper search and saved filters, editors access record management, and administrators control system configuration — all through the same underlying architecture.

FILTER-TO-RFP MAPPING

PLATFORM FEATURE	RFP REQUIREMENT	STATUS
Multi-category faceted search	Filter by category and accommodation type	Built
Geographic and region filter	Filter by country, region, and area	Built
Tag-based filtering	Accessibility features and certifications	Built
Quality or readiness score filter	Verification and completeness logic	Built
Keyword full-text search	Search bar across all fields	Built
Saved search and bookmarks	User saved destinations or preferred result sets	Built

The important point is not only that each feature exists. It is that the filter logic behaves like a shared platform capability. The same model can support public browsing, member depth, editor workflow, dashboards, and future integrations without rebuilding the core search behaviour each time.

ACCESS CONTROL MODEL

PUBLIC
Browse listings, use basic search, and access limited destination detail.

MEMBER
Full search, saved filters, and broader access to destination records.

EDITOR
Add and update entries, manage media, and improve record quality.

MANAGER
Approve changes, oversee categories, and monitor reporting views.

ADMIN
Full system control, configuration, permissions, and operational oversight.

API
Programmatic access for integrations, partner services, and future extensions.

KEY DESIGN DECISIONS

Filter as Infrastructure

Search lives in the API layer, not the UI. That means the same filter logic supports web interfaces, dashboards, partner integrations, and future mobile clients.

Persistent State

Filter combinations are shareable via URL. Users can bookmark, share, and return to specific search states without re-entering criteria.

Tier-Aware Results

The same query engine adjusts what is returned based on the user's access tier. No separate search systems are needed for different audience levels.

RFP CONNECTION

Your platform needs combinable filtering across category, country, accessibility level, and certification while keeping the interface fast and the experience consistent. This project proves that exact pattern at 18 filter dimensions and sub-500ms response — with the same architecture supporting multiple access tiers.

Fast Narrowing

Users can move from a broad universe of records to a useful shortlist with very little friction, which is the core promise of this kind of product.

Shared Logic

Search, detail pages, dashboards, and API responses all reflect the same structured filter model instead of diverging over time.

Tiered Experience

Different user groups can receive different levels of detail and capability without duplicating the platform or weakening governance.

WHY THIS MATTERS FOR YOUR PLATFORM

Your users need to **find accessible destinations quickly** by combining filters like country, category, accessibility level, and certification without waiting on slow result pages or losing track of their search state. This platform proves that those interactions can be handled cleanly at scale. It also demonstrates the value of pairing search logic with **role-based access and structured editorial control**, which is important if your platform needs to serve public visitors, registered members, editors, and administrators inside one coherent system. In other words, it shows how discovery, saved behaviour, and governance can live inside one product instead of becoming separate tools. For your brief, this is the strongest evidence that filtering, access control, and scale can be solved together rather than traded against each other.