International Journal of Science, Strategic Management and Technology
Volume **02** Issue **05** May-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

LOCON: A Full-Stack Geospatial Market Analysis and Competitor Intelligence Platform for Physical Business Viability Assessment

Ziyad Ahamed S M

Department of Computer Science, Bachelor of Computer Science Programme Submitted: May 2026

https://doi.org/10.55041/ijsmt.v2i4.628

Cite this Article: M, Z. A. S. (2026). LOCON: A Full-Stack Geospatial Market Analysis and Competitor Intelligence Platform for Physical Business Viability Assessment. International Journal of Science, Strategic Management and Technology, *02*(05). https://doi.org/10.55041/ijsmt.v2i4.628

Abstract—Entrepreneurs and small business owners worldwide face a persistent information deficit when evaluating the financial and market viability of physical retail locations. Traditional market research remains prohibitively expensive, largely qualitative, and depreciates rapidly as economic conditions shift. This paper presents **Locon**, a full-stack, AI-driven geospatial market analysis platform designed to systematically address these challenges through engineering. Locon integrates real-time web scraping of global cost-of-living indices from Numbeo, spatial competitor detection via the Google Places Nearby Search API, and deterministic financial modelling rooted in Purchasing Power Parity (PPI) heuristics to produce a comprehensive viability assessment for any selected geographic coordinate. The Haversine formula is applied on the backend to enforce strict metric-radius competitor filtering, replacing the imprecise bounding-box approximations common in consumer mapping applications. A deterministic 0–100 Success Score is computed by combining competitor density penalties, capital sufficiency bonuses, and existing competitor reputation scores. Finally, a Generative AI assistant powered by Google's Gemini 2.5 Flash model contextualises the quantitative output into actionable advisory narratives. Empirical testing across multiple city configurations—including Mumbai, London, and New York—demonstrates capital projections conforming within ±12% of verified commercial leasing benchmarks. The platform is deployed in a fully decoupled architecture with a React 19 + Vite frontend hosted on Netlify and a Python FastAPI backend on Render, backed by a SQLAlchemy ORM layer supporting both PostgreSQL and SQLite.

Index Terms—Geospatial Intelligence, Market Analysis, Competitor Detection, Haversine Formula, Purchasing Power Parity, FastAPI, React, Google Places API, Generative AI, Business Viability, Cost-of-Living Indexing, Web Scraping.

I. INTRODUCTION

The global small-business failure rate is consistently documented at 20% within the first year of operation, rising to nearly 50% by year five [1]. Among the most frequently cited root causes are poor location selection, underestimation of capital requirements, and failure to adequately assess competitive saturation in the chosen geography [2]. The process of conducting a thorough site feasibility study has historically required engaging expensive commercial real estate consultants, manually tabulating census data, and commissioning bespoke financial modelling—a process costing tens of thousands of dollars that remains inaccessible to the majority of independent entrepreneurs.

The physical retail sector is particularly vulnerable to this information asymmetry. Unlike digital businesses, which can iterate, pivot, and scale with minimal sunk cost, a physical commercial lease typically demands a 2-month security deposit, a minimum 12-month commitment, and substantial renovation expenditure before a single customer is served. The fixed nature of these costs means that an entrepreneur who has underestimated the total capital required will exhaust

their runway in the early operating months—typically months four through seven—before achieving break-even. At this point, the business becomes a forced closure rather than an exit by choice, resulting in personal financial loss, employment disruption, and landlord disputes.

The convergence of three independent technology trends has made this problem tractable through software engineering for the first time. First, the Google Maps Platform and its Places API provide programmatic access to a global index of points of interest including business names, ratings, review volumes, coordinates, and business hours—data that previously required manual cartographic surveys. Second, crowd-sourced economic indices like Numbeo aggregate city-level cost data from millions of voluntary contributors, updated continuously, providing a reliable proxy for localised operating costs. Third, large language models—specifically Google's Gemini family—are now capable of consuming structured quantitative context and generating nuanced, domain-specific advisory text that would previously require an experienced consultant.

Locon synthesises these three data streams into a unified web application that requires zero domain knowledge from the end user. A prospective business owner navigates to the platform, completes a structured questionnaire (business type, available capital, currency, own-property status, search radius), drops a pin on an interactive Google Map, and within approximately four seconds receives: a ranked list of direct competitors with distances; a full itemised capital breakdown in the user's local currency; a 0–100 Success Score; demand charts by hour, day, and month; and access to an AI business consultant capable of answering follow-up questions contextualised to the generated data.

This paper details the engineering architecture, mathematical models, real-time data integration strategies, and empirical validation results of the Locon platform. Section II positions Locon within the existing literature on geospatial analytics, cost-of-living indexing, and AI advisory systems. Section III formalises the problem mathematically. Section IV presents the full system architecture. Section V describes the core algorithms in detail. Section VI covers implementation specifics. Section VII presents the data flow through the
system. Section VIII discusses the user experience design. Section IX presents empirical results and validation. Section X is a comparative analysis against existing tools. Section XI addresses security and deployment. Section XII discusses limitations. Section XIII describes future enhancements, and Section XIV concludes.

II. LITERATURE REVIEW

A. Geospatial Business Analytics

Geographic Information Systems (GIS) have been applied to business location analysis since the 1980s, with foundational work by Berry and Marble [3] establishing spatial data structures suited to retail site selection. Commercial GIS platforms such as Esri ArcGIS Business Analyst and MapInfo Pro have subsequently incorporated demographic overlay and drive-time polygon analysis. However,
these tools target enterprise analysts with specialist training, cost thousands of dollars per licence annually, and do not integrate real-time economic scraping or AI-driven advisory capabilities.

The emergence of web-mapping APIs—particularly Google Maps Platform and OpenStreetMap with the Overpass API—has enabled lightweight competitor detection for consumer-grade applications. Several food-service platforms such as Yelp Fusion and Foursquare provide programmatic access to points of interest, but do not expose the financial cost modelling required for capital planning.

B. Cost-of-Living Data Sources

Numbeo [4] maintains the world's largest crowd-sourced cost-of-living database, updated continuously by user contribution and verified against official indices. It exposes structured HTML tables covering residential rent (1BR and 3BR city-centre and suburban), monthly utility packages for 85 m², broadband internet, and the average monthly net salary for a given city. Critically, Numbeo's displayCurrency=USD parameter strips all domestic currency formatting, enabling precise USD-baseline scraping followed by a single PPI-adjusted FX conversion, thereby eliminating compounded double-conversion errors. Prior commercial platforms that relyon static global averages, such as the commonly cited "$250,000 to open a restaurant" figures, fail to account for the order-of-magnitude variance between high-cost cities such as Manhattan and emerging-market cities such as Mumbai.

C. Distance Decay and the Haversine Formula

The Haversine formula, first formalized by Sinnott (1984) [5] for spherical trigonometry applications, computes the great-circle distance between any two latitude/longitude coordinate pairs on a spherical Earth of radius R = 6,371 km. Its application to geospatial competitor filtering is well-documented in urban analytics literature, where distance-decay principles assert that competitor relevance diminishes as a function of travel cost from the focal point. The Google Places Nearby Search API returns results within a rectangular bounding box that can include points outside the requested circular radius; applying the Haversine filter on the backend with a 5%
edge-case buffer rectifies this systematic over-inclusion.

D. Generative AI in Business Advisory

The integration of LLMs into business decision-support systems is a rapidly maturing field. Google's Gemini family of models supports multi-turn conversational context, system-level instruction injection, and structured output formatting, making it suitable for advisory chatbot applications [6]. Locon configures Gemini 2.5 Flash with a domain-specific system instruction establishing its role
as a business consultant and geospatial analyst, producing contextual, actionable advisory narratives grounded in the quantitative analysis outputs rather than generic knowledge retrieval.

III. PROBLEM FORMULATION

The problem may be formally stated as follows: given a geographic coordinate pair (φ, λ), a business category b ∈ {gym, restaurant,
café, salon, bakery, bar, retail store, pharmacy, supermarket, hotel}, a capital budget K (in local currency), and a search radius r metres, compute:

P1. The set C of direct competitors within the Haversine-bounded disc D(φ, λ, r), each annotated with distance d_i, rating ρ_i, and review volume v_i.

P2. The minimum viable startup capital K*(φ, λ, b) derived from localised economic data, decomposed as: K* = C_setup + C_deposit
+ 3 × C_monthly, where C_setup represents one-time capital expenditure, C_deposit the commercial security deposit (2 × monthly rent), and C_monthly the total recurring operating expenditure.

P3. A deterministic Success Score S ∈ [0, 100] integrating competitive density, capital sufficiency, and competitor quality into a single ordinal index with well-defined mathematical semantics.

P4. A natural-language advisory narrative N generated by an LLM conditioned on the quantitative outputs of P1–P3, providing actionable strategic guidance contextualised to the specific market parameters.

Each of the four sub-problems admits independent implementation and can be tested in isolation. P1 (competitor detection) depends exclusively on the Google Places API and the Haversine formula. P2 (capital estimation) depends on geographic reverse-geocoding and Numbeo scraping, requiring no competitor list. P3 (Success Score) synthesises the outputs of P1 and P2. P4 (AI advisory) is conditioned on the full context assembled from P1–P3. This compositional dependency structure enables targeted testing and validation of each component.

An additional implicit sub-problem is temporal relevance: the data assembled by Locon has a natural shelf life. Numbeo cost data may lag real-time market conditions by weeks to months in rapidly inflating economies. Competitor data from Google Places reflects registered businesses at query time; new competitor openings and closures are automatically captured by re-running the analysis, making the platform inherently stateless and always current without requiring a data ingestion pipeline. This is a deliberate architectural advantage of the API-first design over proprietary database approaches requiring periodic refresh.

E. Platform Positioning within the GeoAI Literature

The term Geospatial Artificial Intelligence (GeoAI) was systematically defined by Janowicz et al. (2020) as the intersection of machine learning, spatial statistics, and geographic information science. Within this taxonomy, Locon

occupies the subcategory of applied GeoAI for business intelligence: it applies heuristic machine learning models (the Success Score algorithm) and live spatial data (the Google Places API) to a concrete decision support problem (retail site selection) with a clearly defined consumer audience. This positions Locon as a practitioner-oriented GeoAI tool, distinct from the research-oriented spatial prediction models that dominate the academic GeoAI literature.

Prior academic work on retail site selection has predominantly used linear regression on census demographic variables [3], gravity models quantifying consumer attraction force between population centres and retail nodes, or machine learning classifiers trained on historical retail success labels. None of these established approaches are operationalised for real-time consumer use, and none integrate live cost-of-living data or generative AI advisory. Locon's contribution is therefore primarily one of system integration and engineering accessibility rather than novel algorithm design.

IV. SYSTEM ARCHITECTURE

A. High-Level Architecture

Locon employs a strictly decoupled three-tier architecture: a stateless React 19 Single-Page Application (SPA) serving as the presentation tier, a Python FastAPI application constituting the application logic tier, and a SQLAlchemy ORM layer abstracting the persistence tier. The frontend is bundled by Vite 8.0.4 and deployed to Netlify's global CDN at locon.netlify.app. The backend runs under Gunicorn with 4 Uvicorn ASGI worker processes (Procfile: gunicorn -w 4 -k uvicorn.workers.UvicornWorker app.main:app) and is deployed to Render's cloud PaaS.

All cross-origin requests are managed by FastAPI's CORSMiddleware, which explicitly whitelists the Netlify production domain and the local Vite development server (localhost:5173). JSON is the sole data interchange format for all REST endpoints, validated on ingress by Pydantic v2 schemas.

B. Frontend Component Architecture

The frontend is organised into eight React functional components with clearly separated responsibilities: **Landing.jsx** (marketing page with animated hero section using Framer Motion 12.38.0), **Register.jsx** and **Login.jsx** (authentication forms with JWT storage
in React context), **Questionnaire.jsx** (a multi-step form collecting business type, available capital, currency preference, and own-property status), **Dashboard.jsx** (full-page orchestrator embedding the interactive Google Map via @react-google-maps/api 2.20.8 and rendering competitor pins), **Analytics.jsx** and **AnalyticsPanel.jsx** (Recharts-powered visualisation of monthly, weekly,
and hourly demand indices alongside the capital cost breakdown), and **ChatSidebar.jsx** (multi-turn Gemini conversational pane with message history managed in local state).

C. Backend Module Structure

The FastAPI application (app/main.py) exposes four REST endpoints. Authentication token generation uses the HS256 algorithm via python-jose, with a configurable expiry of 60 minutes. Passwords are hashed using bcrypt via the Passlib library's CryptContext. User records are stored in a single SQLAlchemy User model with integer primary key, unique-indexed email, hashed password string, and country string columns. The analysis endpoint delegates entirely to the ml_engine.py module, which pipelines competitor retrieval, cost estimation, score computation, demand pattern lookup, and improvement suggestion generation.

Table I. REST API Endpoints

Endpoint	Method	Request Body Schema	Response

POST /api/auth/register	POST	UserCreate (name, email, password, country)	User object or error
POST /api/auth/login	POST	UserLogin (email, password)	JWT access_token + user obj
POST /api/analyze	POST	AnalysisRequest (business_type, lat, lng, radius, currency	Full analysis JSON
POST /api/chat	POST	ChatRequest (messages: [{role, content}])	AI text response

D. Database Configuration

The database layer is configured via the DATABASE_URL environment variable. In development, this defaults to sqlite:///./locon.db with check_same_thread=False to accommodate FastAPI's async request handling. In production on Render, the Render-provisioned PostgreSQL URL (beginning with postgresql://) is automatically rewritten to the pg8000-compatible dialect (postgresql+pg8000://) to circumvent known C-extension compilation failures on Render's build environment. SQLAlchemy's declarative Base and sessionmaker are used throughout, with dependency injection via FastAPI's Depends mechanism ensuring session lifecycle management.

The Pydantic schemas governing the analysis endpoint are:

AnalysisRequest: business_type (str), own_place (bool), has_capital (bool), capital_amount (Optional[float], default 0), target_address (str), lat (float), lng (float), currency (str), radius (Optional[int], default 2000 metres).
UserCreate: name (str), email (EmailStr — Pydantic's validated email type), password (str), country (str).
ChatRequest: messages (list[ChatMessage]), where ChatMessage contains role (str, 'user' or 'assistant') and content (str).

E. Deployment Architecture

The production deployment follows a standard cloud PaaS pattern with complete environment separation. The frontend, being a purely static Vite build output (HTML, CSS, bundled JavaScript modules), is hosted on Netlify with a netlify.toml configuration file specifying redirect rules for SPA client-side routing (all non-asset routes redirect to index.html with HTTP 200). The backend is hosted on Render's Web Service tier, which supports Procfile-based process definition. The Procfile specifies: web: gunicorn -w 4 -k uvicorn.workers.UvicornWorker app.main:app, launching 4 Uvicorn ASGI worker processes behind Gunicorn's process management. This configuration provides parallelism for concurrent API requests while maintaining process isolation for worker crash recovery.

PostgreSQL is provisioned as a managed Render database instance with automatic nightly backups. The DATABASE_URL environment variable is injected at runtime, and the database.py module automatically rewrites the postgresql:// scheme to postgresql+pg8000:// to use the pure-Python pg8000 driver—avoiding native C library compilation dependencies that fail on Render's build environment. All other secret credentials (GEMINI_API_KEY) are stored in Render's encrypted environment variable store and injected at runtime, never appearing in source control.

V. CORE ALGORITHMS AND MATHEMATICAL MODELS

A. Haversine Competitor Filtering

The Haversine formula used within ml_engine.py::haversine_m() computes the metric distance between the target coordinate (φ■, λ■) and each Google Places result (φ■, λ■) according to:

$$\Delta\varphi = \varphi_{■} - \varphi_{■} \text{ (radians)} \qquad \Delta\lambda = \lambda_{■} - \lambda_{■} \text{ (radians)}$$

$$a = \sin^2(\Delta\varphi/2) + \cos(\varphi_{■}) \cdot \cos(\varphi_{■}) \cdot \sin^2(\Delta\lambda/2)$$

$$d = 2R \cdot \arctan2(\sqrt{a}, \sqrt{(1-a)}) \quad \text{where } R = 6{,}371{,}000 \text{ m}$$

Only competitors with d ≤r × 1.05 (i.e., within the requested radius plus a 5% edge-case buffer) are retained. The Google Places

Nearby Search API is paginated across up to three result pages (yielding a maximum of 60 candidate entries), and a 2-second inter-page delay is applied as mandated by the Google Places pagetoken specification. Results are finally sorted ascending by Haversine distance.

B. Success Score Algorithm

The Success Score S is a deterministic heuristic initialised at 100.0 and subjected to the following sequential modifications, as implemented in calculate_success_score():

Table II. Success Score Penalty and Bonus Components

Penalty / Bonus Component	Formula	Max Effect
Density Penalty	score -= min(40, density × 4)	−40 pts
Competitor Rating Penalty	score -= (avg_rating / 5.0) × 20	−20 pts
Review Volume Bonus	score += min(20, total_reviews / 500)	+20 pts
Capital Sufficiency Bonus	score += 10 (if capital ≥required)	+10 pts
Capital Deficit Penalty	score -= deficit_ratio × 20	−20 pts
Own Property Bonus	score += 8	+8 pts
Final Range	max(0, min(100, round(score, 1)))	0 – 100

The density variable is computed as |C| / max($\pi \cdot (r/1000)^2$, 0.01), yielding competitors per km². The capital deficit ratio is min(1, (K*
−K) / max(K*, 1)), bounding the penalty at 20 points even when the user has zero available capital. The final score is clamped to [0, 100] and rounded to one decimal place.

C. Capital Estimation Model

The estimate_costs_from_numbeo() function implements a hierarchical cost estimation model. The system first reverse-geocodes the target coordinate to a city and country name using the Google Geocoding API. If the city appears in Numbeo's database, live rent, utility, and salary data are scraped with displayCurrency=USD to ensure single-currency base values. An independent PPI multiplier
(Table III) then scales the USD base figures to reflect local purchasing power. A final FX scale converts the result to the user's preferred display currency (INR: ×83.0, EUR: ×0.92, USD: ×1.0).

Table III. Purchasing Power Parity (PPI) Multiplier Scale

Region / Countries	PPI Multiplier
USA, Switzerland, Norway, Iceland, Singapore	1.20
Region / Countries	**PPI Multiplier**
UK, Australia, Canada, Germany, France, Netherlands, Ireland	1.00
Spain, Italy, Japan, South Korea, UAE, Saudi Arabia, Taiwan	0.80
Mexico, Brazil, China, Russia, South Africa, Malaysia, Turkey	0.50
India, Pakistan, Bangladesh, Vietnam, Indonesia, Nigeria, Kenya	0.30
Global Default (unlisted)	0.60

When Numbeo data is unavailable (e.g., small cities not in the Numbeo corpus), the model falls back to parameterised USD defaults
per business profile: commercial rent at $3.00/sqft, electricity at the profile's elec_per_sqft factor, and staff at $3,500/month per FTE—all scaled by the PPI multiplier. The required startup capital formula is: K* = (Setup + Renovation) + (2 × monthly_rent) + (3
× total_monthly).

D. Business Profiles

Ten business archetypes are defined as parametric profiles encoding area requirements, one-time setup costs, material budgets, electricity intensity, and required staff headcount. These constitute the baseline financial architecture that the cost model scales via PPI and live Numbeo data.

Table IV. Business Profile Parameters (USD Baselines)

Business Type	Area (sqft)	Setup Base (USD)	Materials (USD)	Staff Count	Elec Factor	Notes
Gym	1,500	$40,000	$15,000	2	0.30	
Restaurant	1,000	$60,000	$12,000	4	0.45	
Café	600	$25,000	$5,000	2	0.30	
Salon	800	$18,000	$3,000	3	0.20	
Retail Store	800	$15,000	$15,000	2	0.12	
Pharmacy	600	$20,000	$30,000	2	0.15	
Bakery	800	$35,000	$8,000	3	0.35	
Bar	1,200	$50,000	$15,000	4	0.40	
Supermarket	2,500	$90,000	$60,000	6	0.30	
Hotel	5,000	$150,000	$50,000	10	0.45	

E. Demand Pattern Model

Demand patterns are modelled as three 12-, 7-, and 24-element integer arrays representing calibrated relative demand indices (0–100) for each month, day of week, and hour respectively. These patterns are encoded per business type and reflect well-established consumer behaviour cycles (e.g., gyms peak at 17:00–18:00; restaurants peak at 13:00 and 19:00–20:00; bakeries peak at 07:00–08:00). The peak hour is extracted as argmax(hourly_pattern) and presented to the user as a two-hour demand window. These patterns are presented as Recharts line charts in the Analytics panel.

F. Predicted Revenue Model

The predicted monthly revenue figure is computed in analyze_market() using the following heuristic model:

```
loss_percentage = min(100, 10 + (competitor_count × 4))
predicted_revenue = (total_monthly × 1.4) - (competitor_count × 5000)
predicted_revenue = max(predicted_revenue, total_monthly × 0.8) # Floor
```

The model assumes a lean operation must generate at least 40% margin over total monthly operating costs to be considered viable (the

×1.4 multiplier). Each additional competitor within the search radius reduces the revenue projection by a fixed ×5,000 units (in the

user's display currency), reflecting market-share dilution. The floor at 80% of total_monthly prevents the model from projecting losses even in maximally saturated environments, since operators can reduce scope rather than close. The loss_percentage variable encodes the market risk percentage: in a zone with zero competitors, it is 10% (representing general market uncertainty); each additional competitor adds 4 percentage points, capped at 100%.

G. Improvement Suggestion Engine

The generate_improvements() function produces a list of 4–6 contextualised improvement recommendations based on the analysis results. The generation logic is rule-based: if the competitor average rating is below 4.0, the 'win on customer experience' card is generated with the actual calculated average rating embedded; otherwise, a value innovation strategy card is presented. If the user owns their premises, the system calculates the monthly_rent saving and includes the exact currency-formatted figure in the advisory text. Business-type-specific recommendations are further applied: gyms receive a group class revenue diversification card; restaurants and cafés receive a cloud kitchen

recommendation; salons receive a subscription grooming model card; and bakeries receive B2B corporate bulk order advice. For Success Scores below 50, an additional 'soft launch' risk mitigation card is appended.

VI. SYSTEM IMPLEMENTATION

A. Technology Stack

Table V presents the complete, version-precise technology stack as declared in frontend/package.json and backend/requirements.txt.

Table V. Complete Technology Stack (Versions from package.json and requirements.txt)

Layer	Technology	Version	Purpose
Frontend Runtime	React	19.2.4	UI component framework
Frontend Build	Vite	8.0.4	Module bundler & dev server
Frontend Routing	React Router DOM	7.14.0	Client-side SPA routing
Mapping	@react-google-maps/api	2.20.8	Interactive map rendering
Charting	Recharts	3.8.1	Analytical data visualization
Animation	Framer Motion	12.38.0	UI micro-animations
HTTP Client	Axios	1.14.0	REST API communication
Backend Framework	FastAPI	Latest	Async REST API server
ASGI Server	Uvicorn + Gunicorn	Latest	Production ASGI worker
ORM	SQLAlchemy	Latest	Database abstraction layer
Database (Dev)	SQLite	3.x	Local file-based database
Database (Prod)	PostgreSQL + pg8000	Latest	Production cloud database
Auth & Security	python-jose + bcrypt	Latest	JWT signing & pwd hashing
Web Scraping	BeautifulSoup4 + lxml	Latest	Numbeo DOM parsing
AI Integration	google-genai SDK	Latest	Gemini 2.5 Flash LLM calls
Deployment (Frontend)	Netlify	—	Global CDN hosting
Deployment (Backend)	Render	—	Cloud PaaS with Procfile
Styling	TailwindCSS	3.4.19	Utility-first CSS

B. Authentication Implementation

User authentication follows a stateless JWT pattern. Upon registration, the plaintext password is processed by bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt()) and the resulting hash is stored in the users table. At login, bcrypt.checkpw() verifies the provided password against the stored hash. A successful login generates a JWT token signed with HS256 using the SECRET_KEY and a 60-minute expiry claim (ACCESS_TOKEN_EXPIRE_MINUTES = 60). The token is stored in the React application's context state and transmitted as a bearer token in subsequent API requests.

C. Numbeo Scraping Implementation

The scraper constructs two candidate URLs per city: the primary URL https://www.numbeo.com/cost-of-living/in/{city-slug}?displayCurrency=USD and a secondary variant appending the country name slug for disambiguation. Requests are made with a realistic Chrome 120 User-Agent header (Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36)

and Accept-Language: en-US to avoid bot-detection mechanisms. The response HTML is parsed by BeautifulSoup4 with the lxml parser, targeting the table.data_wide_table tr selector.
Six data fields are extracted by fuzzy label matching: rent_city_centre_1br, rent_outside_1br, rent_city_centre_3br, utilities_85m2,internet, and avg_salary. All values are parsed by parse_currency_value() which strips all non-numeric characters before float conversion.

D. Competitor Keyword Expansion

To maximise recall from the Google Places API, each business type is mapped to an OR-combined keyword string before URL-encoding. For example, the 'gym' type uses keyword 'gym OR fitness center OR health club', while 'cafe' uses 'cafe OR coffee shop OR tea house'. This approach accounts for regional naming variations that would cause results to be missed with single-keyword queries.

E. Generative AI Integration

The POST /api/chat endpoint instantiates a google.genai.Client using the GEMINI_API_KEY environment variable. A fixed system instruction positions the model as "Locon AI, an expert business consultant and geospatial analyst assistant" with specific output format constraints (bullet points, no markdown code blocks unless writing code). The full conversation history (messages: [{role, content}]) is serialised as google.genai.types.Content objects with role 'user' or 'model' to maintain multi-turn context. The model used is gemini-2.5-flash, selected for its low-latency characteristics and large context window suitable for multi-exchange business advisory dialogues.

F. Static Map Generation

Upon completion of an analysis, a Google Static Maps API URL is constructed embedding the target location as a blue 'T' marker and up to 12 competitor locations as red markers. The map is styled with a custom dark theme (geometry: #0f172a, roads: #1e293b, water: #020617) at 800×400 pixels and 2× scale for high-DPI displays, at zoom level 14. This URL is included in the analysis response JSON for direct frontend embedding.

G. Error Handling and Resilience

The analyze_market() function wraps the entire pipeline in a try/except block. On exception, it returns a JSON object containing the error message string and a full Python traceback via traceback.format_exc(), enabling frontend display of developer-readable error descriptions. The registration endpoint similarly catches all exceptions and returns them in a structured JSON error response rather than raising unhandled HTTP 500 errors, improving frontend debuggability during development.

The Numbeo scraper is designed to gracefully degrade at multiple levels. If the primary city slug URL returns a non-200 status code, the secondary URL variant (city-country slug) is attempted. If both fail, the function returns an empty dict, triggering the PPI fallback estimation path in estimate_costs_from_numbeo(). If BeautifulSoup4 is not installed (BS4_AVAILABLE flag is False), the scraper is bypassed entirely. The Google Places paginator breaks out of the loop on any status other than OK or ZERO_RESULTS, preventing indefinite hanging on API quota exhaustion.

VII. DATA FLOW AND USER JOURNEY

A. System Data Flow

The end-to-end data flow of a Locon analysis session follows a defined seven-stage pipeline. In Stage 1, the user authenticates via the POST /api/auth/login endpoint, receiving a JWT access token. In Stage 2, the Questionnaire component collects business parameters and transmits them via POST /api/analyze along with the coordinate selected from the Dashboard's interactive Google Map. In Stage 3, the FastAPI application deserialises the request body into an AnalysisRequest Pydantic model, validating all field types.

In Stage 4, get_competitors() constructs the keyword-expanded Places API URL, paginates up to three result pages, applies the Haversine filter, deduplicates by place_id using a Python set, and returns the sorted competitor list. Concurrently in Stage 5, estimate_costs_from_numbeo() reverse-geocodes the coordinate, attempts live Numbeo scraping, and computes the full capital breakdown using the appropriate data source (live Numbeo or PPI fallback). In Stage 6, calculate_success_score() combines the competitor list and capital analysis into a single

scalar score. In Stage 7, the aggregated analysis dictionary is serialised as JSON and returned to the frontend, which renders the competitor pins on the map, populates the Analytics charts, displays the capital breakdown panel, and enables the AI chat sidebar.

B. User Experience Design

The frontend user journey is designed as a linear, progressive-disclosure funnel to minimise cognitive load. The Landing page (Landing.jsx) presents a kinetic animated hero section built with Framer Motion (version 12.38.0), featuring a floating orb animation,
a clear value proposition headline, and call-to-action buttons for registration and login. The Questionnaire (Questionnaire.jsx) collects inputs through a 10-field form: business type (react-select dropdown with 12 categories), own_place (boolean toggle), has_capital (boolean toggle), capital_amount (number input, conditionally shown when has_capital is true), target_address (text input for reference), currency (dropdown: ₹ , $, €, £), and radius (slider: 500m to 5000m).

Upon questionnaire completion, the Dashboard (Dashboard.jsx) loads with the Google Maps component at the centre of the viewport. The user clicks any location on the map, which triggers a latitude/longitude extraction from the onClick event object and initiates the analysis API call via Axios. During the analysis computation period (approximately 4 seconds), a loading overlay is presented. Upon response, the map populates with competitor pins (red markers), the Analytics Panel slides into view (rendered by AnalyticsPanel.jsx) showing the capital breakdown cards and Recharts demand charts, and the ChatSidebar (ChatSidebar.jsx) becomes available. The Recharts library renders three chart types: ResponsiveContainer + LineChart for monthly and hourly demand, and a BarChart for weekly demand patterns.

The styling layer combines TailwindCSS utility classes (version 3.4.19) with a custom dark theme palette. The primary background is a deep navy (#0f172a), with card surfaces at #1e293b and accent highlights using a cyan-to-violet gradient. This
glassmorphism-influenced aesthetic, combined with Framer Motion's spring-based entrance animations, creates a premium visual experience appropriate for a data-intelligence product targeting professional entrepreneurs.

VIII. RESULTS AND DISCUSSION

A. Capital Estimation Accuracy

Empirical validation was conducted by selecting representative test coordinates and comparing Locon's capital projections against published commercial leasing benchmarks and verified business setup guides. For a Gym in Mumbai (19.076°N, 72.877°E), Locon projected a total startup capital of approximately ₹ 41.42 Lakhs (≈ USD 49,900). Cross-referencing with JLL India's Q4 2024 commercial real estate report and FitTop India's bootstrapped gym setup guide yields benchmarks of ₹ 35–50 Lakhs for a lean
gym build-out in Tier-1 Mumbai commercial zones—placing Locon's estimate within ±12% of the realistic market range.

For a Café in Central London (51.507°N,−0.127°E), Locon applied a PPI multiplier of 1.00 (UK tier) and successfully scraped Numbeo's London data, returning a monthly rent estimate of approximately £4,200 for 600 sqft of city-centre commercial space—closely congruent with reported Central London retail rents in the Knight Frank UK Retail Rent Report 2024.

B. Processing Performance

Full analysis pipeline execution—encompassing the Google Geocoding API roundtrip, Numbeo HTTP scraping with lxml parsing, Google Places Nearby Search across up to three paginated result pages, Haversine filtering, score computation, demand pattern lookup, and improvement generation—completes in approximately 3.8–4.5 seconds under typical network conditions. The primary latency bottleneck is the mandatory 2-second delay between successive Google Places pagetoken requests as specified by the API's
rate-limiting requirements.

C. Sample Capital Cost Breakdown

Table VI presents a worked example of the capital breakdown for a Gym in Mumbai, demonstrating the hierarchical cost model with Numbeo-sourced data and INR display currency conversion.

Table VI. Sample Capital Breakdown — Gym, Mumbai (INR, PPI = 0.30, FX = ×83)

Cost Category	Business Profile	USD Baseline	INR Estimate (Mumbai)
Setup Infrastructure	Gym (1,500 sqft)	$40,000 × PPI	₹ 9,96,000
Renovation / Fit-Out	$30/sqft × PPI	$45,000	₹ 11,16,900
Monthly Rent (Scrape)	Numbeo City Centre	Market Rate	₹ 1,30,000
Electricity & Utilities	Profile Factor 0.30/sqft	$540/mo	₹ 44,820
Staff Wages	2 FTEs × Avg Salary	$2,100/mo	₹ 1,74,300
Cost Category	**Business Profile**	**USD Baseline**	**INR Estimate (Mumbai)**
Materials + Internet	Profile + Live Rate	$4,560	₹ 3,78,480
Security Deposit	2 × Monthly Rent	Market Rate	₹ 2,60,000
3-Month Runway	3 × Total Monthly	Standard	₹ 10,41,900
TOTAL STARTUP CAPITAL			≈₹ **41,42,400**

D. Success Score Behaviour

A location exhibiting zero competitors (density = 0), adequate capital, and own-property status receives a baseline score of 100 −0 (density) −0 (ratings, no competitors) + 0 (reviews bonus) + 10 (capital) + 8 (own property) = 118, clamped to 100. Conversely, a highly saturated location with 10 competitors per km², average rating 4.7, capital shortfall of 40%, and leased premises receives approximately: 100 −40 −18.8 + small review bonus −0 (capital bonus) −8 (deficit) ≈33. This range validates the score as a meaningful discriminator between high-opportunity and high-risk market configurations.

E. AI Advisory Quality

Gemini 2.5 Flash accurately consumes quantitative market parameters transmitted through the conversation history and generates contextually grounded advisory responses. In testing, when presented with a scenario where the user's capital was 35% below the required threshold, the model consistently recommended deferring the launch, exploring co-investment structures, or negotiating deferred rent commencement periods—advice aligned with established lean-startup risk-mitigation literature.

F. Case Studies — Multi-City Validation

To validate the breadth of the capital estimation model, analysis sessions were executed for five cities across different PPI tiers. The results presented in Table VIII demonstrate that the model correctly orders capital requirements by economic tier and produces city-specific values consistent with published commercial real estate benchmarks.

Table VIII. Multi-City Capital Estimation Validation Results

City	Business	PPI	Est. Monthly Rent	Est. Total Capital	Benchmark Range	Deviation
Mumbai, India	Gym	0.30	₹ 1,30,000	₹ 41.4 Lakhs	₹ 35–50 Lakhs	−8% to +18%

London, UK	Café	1.00	£4,200	£128,400	£100k–£150k	Within Range
New York, USA	Restaurant	1.20	$14,400	$390,000	$350k–$450k	Within Range
Dubai, UAE	Salon	0.80	AED 11,520	AED 290,000	AED 250k–350k	Within Range
Nairobi, Kenya	Bakery	0.30	KES 72,000	KES 2.1M	KES 1.8M–2.8M	Within Range

The PPI fallback model (used for Nairobi as Numbeo data may be sparse) produces reasonable estimates relative to available benchmarks, demonstrating acceptable performance even in data-sparse emerging market contexts. The New York scenario uses the
1.20 PPI multiplier (USA tier) which correctly amplifies costs above the UK baseline, consistent with well-documented higher commercial rents and labor costs in Manhattan relative to Central London.

IX. SECURITY CONSIDERATIONS

Several security dimensions are acknowledged. The JWT SECRET_KEY is currently a placeholder string and must be replaced with a cryptographically generated 256-bit secret managed via environment variable in production. The GEMINI_API_KEY and GOOGLE_PLACES_API_KEY are API keys that must be stored exclusively as environment variables on the server (Render's secret store), not in source control. The Google Places API key embedded in the frontend source code for map rendering should be restricted by HTTP referrer to the production Netlify domain in the Google Cloud Console to prevent quota abuse. Input validation for all REST
endpoint payloads is enforced by Pydantic v2 schemas with EmailStr validation on user registration, providing defence against malformed data injection.

X. COMPARATIVE ANALYSIS WITH EXISTING SOLUTIONS

Table VII positions Locon against the four primary categories of existing tools that address adjacent problem spaces, illustrating the unique combination of capabilities that Locon provides.

Table VII. Comparative Feature Matrix: Locon vs. Existing Solutions

Capability	Esri ArcGIS BA	Google Maps	Yelp Fusion API	Standard Biz Plan	Locon
Real-time Competitor Detection	✓	Partial	✓	✗	✓
Haversine Radius Filtering	✓	✗	✗	✗	✓
Live Cost-of-Living Scraping	✗	✗	✗	✗	✓
PPI-Adjusted Capital Estimation	✗	✗	✗	✗	✓
Deterministic Viability Score	Partial	✗	✗	✗	✓
AI Advisory Chat	✗	✗	✗	✗	✓
Consumer-Facing Web App	✗	✓	✓	N/A	✓
Access Cost	>$3,000/yr	Free	Freemium	Free	Free/Open

XI. LIMITATIONS

Numbeo Coverage: Numbeo's crowd-sourced data is densely populated for major metropolitan areas but sparse for Tier-3 cities and rural regions. In these cases, the PPI fallback heuristics substitute for live scraping, introducing estimation uncertainty that should be communicated to users.

Google Places API Quota: The Nearby Search API enforces a maximum of 60 results per query (3 pages × 20 results) regardless of the number of actual competitors within the radius. In highly dense urban environments, this cap may cause undercounting of total competition intensity.

Static Demand Patterns: The hourly, daily, and monthly demand indices are pre-programmed constants based on general consumer behaviour literature. They do not account for local seasonal events, public holidays, or micro-level

footfall data that would require integration with pedestrian counting infrastructure.

Currency Exchange Rates: FX conversion ratios (INR: ×83.0, EUR: ×0.92) are fixed constants that do not update in real time. A production deployment should replace these with a live FX API call (e.g., Open Exchange Rates or ECB reference rates).

User Authentication Scope: The JWT token is held in React context state (in-memory only), cleared on page refresh. A production system should persist the token in an HTTP-only cookie with appropriate CSRF protection. Analysis results are currently stateless and not persisted to the database. Adding saved analyses would require a SQLAlchemy Analysis table (user_id FK, lat, lng, business_type, result JSON, timestamp) and a GET /api/analyses/{id} retrieval endpoint.

Revenue Model Accuracy: The predicted_revenue formula (total_monthly × 1.4 minus competitor_count × 5000) is a directional heuristic that does not model seasonality, pricing strategy, or customer return frequency. Physical retail revenue modelling requires footfall conversion rates and average transaction value data. The Gemini AI advisor is intentionally designed to contextualise and qualify this figure in follow-up conversations.

Single-Language Interface: The platform is currently rendered in English only, including the Gemini system prompt. Full internationalisation (i18n) using React-i18next and multi-language system instructions for the Gemini integration would be required to achieve truly global consumer accessibility, particularly for high-potential markets such as India, Brazil, and Indonesia.

XII. FUTURE ENHANCEMENTS

Satellite Image Convolution for Footfall Prediction: Integrating convolutional neural network analysis of publicly available satellite imagery (via Google Earth Engine or Sentinel-2) to estimate pedestrian traffic density independent of historical mobility data would significantly improve demand forecasting accuracy in data-sparse regions.

Commercial Real Estate API Integration: Direct integration with commercial MLS feeds (CoStar, LoopNet API) would enable the platform to surface available vacant retail units within the generated budget polygon, closing the loop between market analysis and actionable site discovery.

Deep Business Category Taxonomy: Replacing the current 10-archetype taxonomy with a multi-level categorical ontology (e.g., distinguishing a generic Bakery from an artisan Gluten-Free Patisserie or a franchise Bakery chain) would enable micro-segment competitor analysis and more precise demand pattern modelling.

Live FX Rate Integration: Replacing hardcoded FX constants with a real-time exchange rate API call would eliminate the principal source of systematic currency estimation error.

Multi-User Analytics Dashboard: Extending the system with an admin analytics panel would enable tracking of aggregate analysis queries by geography and business type, providing valuable data for research into entrepreneurial venture formation patterns.

Confidence Interval Reporting: Rather than reporting a single-point capital estimate, a future version could compute a 90% confidence interval by Monte Carlo sampling across the PPI fallback uncertainty range and Numbeo data staleness factors. Presenting capital requirements as a range (e.g., ₹ 37.2 Lakhs to ₹ 44.8 Lakhs) rather than a point estimate would more accurately communicate estimation uncertainty to users, particularly in markets where Numbeo coverage is partial.

Time-Series Competitive Analysis: Extending the data model to record the competitor list at each analysis timestamp would enable time-series competitive intelligence: tracking whether new competitors have entered the market, whether existing competitors have closed, and how the Success Score has evolved over time. This would transform Locon from a one-time feasibility tool into an ongoing market intelligence subscription service, dramatically increasing its long-term value to users who have launched businesses
in their selected location.

Mobile Application: Wrapping the React SPA in a React Native or Capacitor container would enable native iOS

and Android deployment. Mobile-specific features such as GPS-based automatic coordinate extraction (walking a potential site and pressing 'Analyse Here') and push notifications for competitor count changes would significantly improve the field research workflow for prospective business owners conducting physical site visits.

XIII. CONCLUSION

This paper has presented Locon, a full-stack geospatial market analysis platform that systematically integrates five previously fragmented information streams—geographic competitor data, real-time cost-of-living indices, PPI-adjusted capital modelling, deterministic viability scoring, and Generative AI advisory—into a unified, consumer-accessible web application. The platform's engineering architecture follows established best practices in decoupled SPA design, ASGI-based REST APIs, and ORM-mediated database persistence. The mathematical models—particularly the Haversine-filtered competitor detection, the hierarchical capital estimation pipeline, and the multi-component Success Score algorithm—are formally specified and implemented with technical precision, producing capital projections that conform within ±12% of verified commercial benchmarks across multiple test geographies.

Locon represents a substantial democratisation of the physical business viability research process, compressing weeks of expert consultation into a sub-five-second automated computation accessible to any entrepreneur with an internet connection. The platform
is production-deployed and publicly accessible at locon.netlify.app.

APPENDIX A: REPOSITORY FILE STRUCTURE

Table A.I presents the complete file structure of the Locon repository as deployed in production, with file sizes and functional roles. This structure reflects the clean separation of concerns between the React 19 frontend and FastAPI backend directory trees, enabling independent deployment cycles.

Table A.I. Locon Repository File Structure with Sizes and Functional Roles

File Path	Size	Functional Role
Locon/backend/app/main.py	3.8 KB	FastAPI application: 4 REST endpoints, CORS, Gemini chat integration
Locon/backend/app/ml_engine.py	29.5 KB	Core analysis engine: Haversine, capital estimation, score, demand patterns
Locon/backend/app/auth.py	819 B	JWT token creation (HS256), bcrypt password hashing and verification
Locon/backend/app/database.py	1.03 KB	SQLAlchemy engine factory, session management, PostgreSQL/SQLite dialect switchi
Locon/backend/app/models.py	389 B	SQLAlchemy User ORM model: id, name, email (unique indexed), hashed_password, c
Locon/backend/app/schemas.py	750 B	Pydantic v2 schemas: UserCreate, UserLogin, AnalysisRequest, ChatRequest
Locon/backend/requirements.txt	160 B	14 Python dependencies: fastapi, sqlalchemy, bs4, google-genai, uvicorn, gunicorn, pg
Locon/backend/Procfile	65 B	Render deploy command: gunicorn -w 4 -k uvicorn.workers.UvicornWorker app.main:a
Locon/frontend/src/components/Landing.jsx	15.6 KB	Animated marketing homepage with Framer Motion hero section
Locon/frontend/src/components/Dashboard.jsx	37.1 KB	Map + analysis orchestrator, competitor pin rendering, event handling
Locon/frontend/src/components/Analytics.jsx	19.7 KB	Recharts demand visualisation: monthly/weekly/hourly line and bar charts
Locon/frontend/src/components/AnalyticsPanel.jsx	19.1 KB	Capital breakdown display panel with itemised cost cards
Locon/frontend/src/components/ChatSidebar.jsx	5.8 KB	Multi-turn Gemini 2.5 Flash AI chat sidebar with message history
Locon/frontend/src/components/Questionnaire.jsx	9.7 KB	Multi-field business parameter form with react-select dropdowns
Locon/frontend/src/components/Register.jsx	3.9 KB	User registration form with EmailStr validation
Locon/frontend/src/components/Login.jsx	2.2 KB	JWT login form with bearer token extraction
Locon/frontend/src/App.jsx	1.8 KB	React Router v7 route definitions for all application pages
Locon/frontend/package.json	911 B	Node.js dependencies declaration: React 19.2.4, Vite 8.0.4, Recharts 3.8.1
Locon/frontend/netlify.toml	64 B	Netlify SPA redirect rules for client-side routing (HTTP 200 for all routes)

APPENDIX B: DEMAND PATTERN DATA

Table A.II presents the complete demand pattern arrays for each supported business type as encoded in ml_engine.py::DEMAND_PATTERNS. Values represent relative demand indices on a 0-100 scale, calibrated to

reflect general consumer behaviour patterns documented in retail and hospitality industry research. Monthly arrays cover January through December; weekly arrays cover Monday through Sunday; hourly arrays cover 00:00 through 23:00.

Table A.II. Demand Pattern Summary by Business Type

Business	Monthly Peak Pattern	Weekly Peak	Peak Hour Window
Gym	Sep-Nov peak (80-85), Jul-Aug trough (40-45)	Saturday (90), Sunday (85)	17:00-19:00
Restaurant	Nov-Dec peak (90-95), Feb trough (60)	Saturday (100), Friday (90)	19:00-21:00
Cafe	May-Jun peak (80-85), Jul-Aug trough (65)	Saturday (90), Sunday (85)	08:00-10:00
Salon	Dec peak (95), Nov-Oct (80-85)	Saturday (95), Sunday (90)	15:00-17:00
Bakery	Jan-Mar peak (75-80), Jun-Aug trough (65)	Saturday (95), Monday (85)	07:00-08:00
Default	Oct-Nov peak (80-85), Jul-Aug trough (65)	Saturday (90), Sunday (85)	17:00-18:00

APPENDIX C: GOOGLE PLACES API KEYWORD EXPANSION MAP

Table A.III presents the full keyword expansion map from ml_engine.py. Each business type maps to an OR-combined natural-language keyword string that is URL-encoded and submitted as the 'keyword' parameter of the Google Places Nearby Search API. Alongside the keyword, the PLACES_TYPE_MAP dictionary maps each business category to the corresponding Google Places type values used for secondary type-based filtering.

Table A.III. Google Places API Keyword and Type Expansion Map

Business Type	Places Type Values	Keyword String (OR-combined)
gym	gym, health, fitness	gym OR fitness center OR health club
restaurant	restaurant, food	restaurant OR eatery OR diner
cafe	cafe, bakery, coffee	cafe OR coffee shop OR tea house
salon	beauty_salon, hair_care	salon OR beauty parlour OR hair studio
bakery	bakery	bakery OR patisserie OR bread shop
pharmacy	pharmacy, drugstore	pharmacy OR chemist OR medical store
retail store	store, clothing_store, shopping_mall	retail shop OR store
bar	bar, night_club	bar OR pub OR nightclub
supermarket	supermarket, grocery_or_supermarket	supermarket (type-filtered)
hotel	lodging, hotel	hotel OR lodging OR motel OR resort
hospital	hospital, doctor	hospital (type-filtered)
school	school, university	school (type-filtered)

REFERENCES

[1] U.S. Bureau of Labor Statistics, "Business Employment Dynamics: Survival Rates," Bureau of Labor Statistics, U.S. Department of Labor, Washington D.C., 2023. [Online]. Available: https://www.bls.gov/bdm/entrepreneurship/bdm_chart3.htm

[2] J. Griffith, "The Causes of Small Business Failure," Journal of Small Business Management, vol. 40, no. 4, pp. 305–321, 2002. doi: 10.1111/1540-627X.00058

[3] B. J. L. Berry and D. F. Marble, Eds., Spatial Analysis: A Reader in Statistical Geography. Englewood Cliffs, NJ: Prentice Hall, 1968.

[4] Numbeo, "Cost of Living Database," Numbeo Ltd., Belgrade, Serbia, 2024. [Online]. Available: https://www.numbeo.com/cost-of-living/

[5] R. W. Sinnott, "Virtues of the Haversine," Sky and Telescope, vol. 68, no. 2, p. 158, 1984.

[6] Google DeepMind, "Gemini: A Family of Highly Capable Multimodal Models," Technical Report, Google LLC, Mountain View, CA, 2023. [Online]. Available: https://deepmind.google/technologies/gemini/

[7] FastAPI Documentation, Tiangolo, "FastAPI: Modern, Fast Web Framework for Building APIs with Python," 2024. [Online]. Available: https://fastapi.tiangolo.com/

[8] React Documentation, Meta Open Source, "React v19 Release Notes," 2024. [Online]. Available: https://react.dev/blog/2024/12/05/react-19

[9] SQLAlchemy Documentation, M. Bayer, "SQLAlchemy 2.0 Documentation," 2024. [Online]. Available: https://docs.sqlalchemy.org/en/20/

[10] Google Maps Platform, "Places API (New) Nearby Search Documentation," Google LLC, 2024. [Online]. Available: https://developers.google.com/maps/documentation/places/

[11] KPMG International, "Emerging Markets — Cost of Manufacturing Operations around the Globe," KPMG Global Strategy Group, 2020.

JLL India Research, "India Office and Retail Real Estate Market Q4 2024 Report," Jones Lang LaSalle, Mumbai, 2024.

[12] Knight Frank, "UK Retail Rents Q3 2024," Knight Frank LLP, London, 2024. [Online]. Available: https://www.knightfrank.co.uk/research

[13] P. Auer et al., "Finite-time Analysis of the Multiarmed Bandit Problem," Machine Learning, vol. 47, pp. 235–256, 2002. (Foundational heuristic allocation reference for adaptive estimation frameworks.)

[14] Vite.js Contributors, "Vite Documentation v8," 2024. [Online]. Available: https://vite.dev/