**Booking.com Extranet Listing Automation: Workflow, API Schema, and Validation — Prepared by Macari Company**

# **Executive Summary**

This proposal focuses on the system workflow, API payload schema, and constraints/validation required to automate hotel property listing creation on the Booking.com Extranet. It outlines a practical, auditable flow from Excel-based data ingestion to field mapping, validation, and browser automation, with optional AI-assisted enrichment gated by human review. A detailed, cloud-agnostic technology stack is proposed to ensure reliability, traceability, and scalability while meeting the RFP's goals to reduce manual entry time, increase field accuracy, and enable batch processing.

- Scope centers on workflow design, payload specification, and validation/controls.

- Emphasis on accuracy, auditability, and safe automation for the Booking.com Extranet.

- Detailed, vendor-neutral technology stack aligned with reliability and scale.

|  |  |
| --- | --- |
| **Objective** | **Outcome** |
| Automate listings end-to-end | Browser automation with validated payloads and audit trails |
| Reduce time and errors | Structured mapping, strict validation, and retry/fallback logic |
| Enable safe AI use | Confidence thresholds with human-in-the-loop review |

# **Company Introduction**

Macari Company prepares this focused response on workflow, payload schema, and validation controls for Booking.com Extranet listing automation. The team functions bilingually (Arabic and English) and emphasizes clear process orchestration, robust data handling, and controllable automation aligned to operational goals.

- Brand: Macari Company

- Languages: Arabic, English

- Focus in this response: workflow, payload, validation

|  |  |
| --- | --- |
| **Attribute** | **Details** |
| Brand Name | Macari Company |
| Operating Languages | Arabic, English |
| Proposal Focus | System workflow, API payload, constraints/validation |

# **Understanding of the RFP and Objectives**

The RFP seeks a robust system workflow for automating Booking.com Extranet listings, a clear API-style payload definition for property data, and concrete validation/constraint rules to control risk, accuracy, and authorization. We align our approach to: (1) define a staged, auditable workflow; (2) formalize property payload structures that map internal form fields to extranet inputs; and (3) enforce validations, human review thresholds, and access controls to minimize errors while accelerating time-to-listing.

- System Workflow: user input → parse/validate → mapping → automation → audit

- Payload: structured property JSON aligned to extranet field needs

- Constraints: authorization, mandatory fields, confidence thresholds, fallbacks

|  |  |
| --- | --- |
| **RFP Focus Area** | **Our Alignment** |
| Workflow | Modular, auditable stages with retries and human review |
| API Payload | Typed JSON with rooms, amenities, policies, images |
| Validation | Schema + business rules, RBAC, and error routing |

# **Technical Approach and Methodology**

We propose a modular service that ingests Excel data, normalizes and validates it, generates a compliant property payload, and drives browser automation to the Booking.com Extranet. The system includes a mapping layer, strict validation with business rules, robust error reporting, and audit logging. Optional AI assists with enrichment only when confidence thresholds are met; otherwise records are routed for human review.

- Separation of concerns: ingest, validate, map, automate, audit

- Deterministic validation before any automation step

- Human-in-the-loop for low-confidence enrichment

|  |  |  |
| --- | --- | --- |
| **Stage** | **Key Function** | **Outcome** |
| Ingestion | Read Excel/uploader and normalize | Structured internal model |
| Validation | Schema + business rules + referential checks | Clean, consistent payload or actionable errors |
| Automation | Browser automation with session control | Accurate extranet listing creation |

# **Framework Overview**

The framework integrates a typed data layer, a mapping engine, and an automation driver. A rules engine enforces constraints, and an audit subsystem records every submission with diffs, timestamps, and operator IDs. A queue decouples ingestion from automation, enabling retries and batch operations.

- Typed models for predictable mappings

- Rules engine for constraints and defaults

- Queue-driven automation with idempotency

|  |  |  |
| --- | --- | --- |
| **Module** | **Purpose** | **Interaction** |
| Data Models | Define property/room/policy/images | Validates and serializes payload |
| Mapping Engine | Map internal keys to extranet fields | Produces field-level instructions |
| Automation Driver | Login, navigate, input, upload | Executes mapped actions safely |

# **Phased Methodology**

Delivery is phased to reduce risk: define the workflow and payload first, then implement validation and automation, and finally harden reliability with retries, monitoring, and UAT. This mirrors the RFP timeline while concentrating on sections covering workflow, payload, and constraints.

- Discovery: confirm fields, mappings, and rules

- Build: payload schema, validation, automation

- UAT: accuracy checks and operator feedback

|  |  |  |
| --- | --- | --- |
| **Phase** | **Focus** | **Primary Deliverable** |
| Discovery | Field catalog, mappings, workflow | Workflow spec and mapping dictionary |
| Implementation | Schema, validation, automation driver | Working prototype with audit logs |
| UAT & Hardening | Edge-cases, retries, observability | UAT sign-off and runbooks |

# **Methodological Pillars**

Our methodology prioritizes determinism, transparency, and safety. Deterministic validations prevent malformed submissions. Transparency is ensured via full audit logging. Safety comes from role-based controls, confidence thresholds, and structured fallbacks.

- Deterministic validation before automation

- Full auditability and traceability

- Guardrails for AI and permissions

|  |  |
| --- | --- |
| **Pillar** | **Practices** |
| Reliability | Schema validation, retries, idempotency |
| Traceability | Audit logs, mapping references, diffs |
| Safety | RBAC, confidence thresholds, human review |

# **Project Architecture**

A service-oriented architecture decouples ingestion/validation from automation. Data enters via an uploader or Excel import, passes through schema and business-rule validators, is transformed to a property payload, and is executed by an automation driver against the extranet. Observability and audit services provide end-to-end visibility.

- Service boundaries for resilience and scale

- Clear data contracts between components

- Observability first for rapid diagnosis

|  |  |  |
| --- | --- | --- |
| **Tier** | **Components** | **Responsibility** |
| Interface | Uploader/UI, API gateway | Intake and operator control |
| Core | Mapper, validators, rules engine | Correctness and business logic |
| Automation | Browser automation worker | Extranet interactions |

# **System Components**

Core components include an ingestion service for Excel/uploads, a mapping engine correlating internal form keys to extranet fields, a validation layer combining schema and business rules, an automation worker handling login, navigation, inputs, and image uploads, and an audit/logging subsystem. Optional AI enrichment is isolated and gated by confidence thresholds and human review.

- Ingestion: Excel/uploader to normalized models

- Mapping: dictionary-driven field alignment

- Automation: session management, inputs, uploads

|  |  |  |
| --- | --- | --- |
| **Component** | **Key Functions** | **Notes** |
| Ingestion Service | Parse Excel, normalize, dedupe | Supports batch processing |
| Validation Layer | Schema + business rules + referential checks | Stops on mandatory field failures |
| Automation Worker | Login, form fill, image upload, submit | Retry on transient errors |
| Audit & Monitoring | Event logs, metrics, alerts | Compliance and support |
| AI Enrichment (Optional) | Fill missing data under guardrails | Human-in-the-loop if low confidence |

# **Data Flow & Integration**

The workflow proceeds as follows: (1) User uploads or selects an internal Excel content form; (2) The system parses and normalizes data; (3) Validation enforces mandatory fields (e.g., name, location), types, and business rules; (4) A mapping engine translates internal keys to Booking.com Extranet fields; (5) A property payload is generated; (6) The automation worker logs in, navigates the extranet, inputs values, and uploads images; (7) Audit logs capture all actions and outcomes. Optional AI-assisted enrichment occurs before validation and requires human approval if below confidence thresholds. Example property payload structure is shown below.

- Strict validation before automation

- Mapping dictionary governs field alignment

- Full audit, with per-field status and timestamps

|  |  |  |
| --- | --- | --- |
| **Step** | **Description** | **Output** |
| User Input | Excel upload or form-based entry | Raw property dataset |
| Parsing | Normalize and type-cast | Internal typed model |
| Validation | Mandatory fields, formats, rules | Validated model or errors |
| Mapping | Form keys → extranet fields | Submission-ready payload |
| Automation | Login, fill, upload, submit | Created/updated listing |

# **Technology Stack**

The proposed stack is cloud-agnostic and emphasizes reliability, observability, and maintainability. It supports Excel ingestion, strict validation, resilient automation, and auditable operations. Alternatives are provided to fit the client's environment and preferences.

- Cloud-agnostic components to support hybrid or cloud deployment

- Typed models and validation-first design

- Resilient, observable automation with safe fallbacks

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer** | **Primary Option** | **Alternatives** | **Purpose** |
| Backend Framework | Python (FastAPI) | Node.js (NestJS), Java (Spring Boot) | REST APIs for ingestion, validation, orchestration |
| Data Parsing | Pandas (Excel/CSV) | Apache Arrow, openpyxl | Robust Excel ingestion and normalization |
| Validation | Pydantic (typed models) | Marshmallow, Cerberus | Schema and type validation; business rules layering |
| Rules Engine | Custom rule layer (Python) | Drools (Java), json-rules-engine | Business constraints, defaults, and gating |
| Automation | Playwright (headless, Python) | Selenium, Puppeteer | Booking.com Extranet login, navigation, inputs, uploads |
| Queue/Workers | Celery + Redis | RabbitMQ, Kafka | Async jobs, retries, rate control, idempotency |
| Storage (Relational) | PostgreSQL | MySQL | Mappings, job states, audit references |
| Object Storage | S3-compatible storage | MinIO (on-prem) | Image files, logs, artifacts |
| Logging | Structured JSON + OpenSearch | ELK Stack | Centralized, searchable logs and audit trails |
| Monitoring | Prometheus + Grafana | OpenTelemetry, Cloud vendor tools | Metrics, dashboards, alerting |
| Error Tracking | Sentry | Open-source equivalents | Exception visibility and triage |
| CI/CD | GitHub Actions | GitLab CI, Jenkins | Build, test, security scans, deploy |
| Containerization | Docker | Podman | Packaging services and workers |
| Orchestration | Kubernetes | Docker Swarm | Scalable deployment of services |
| Secrets | Vault or cloud secrets manager | Doppler | Manage credentials and session tokens |
| AuthN/AuthZ | RBAC within backend | OIDC/OAuth2 (SSO) if available | Operator access control and audit linkage |
| AI Enrichment (Optional) | Configurable provider via API | Local LLM with retrieval | Suggest values; gated by confidence thresholds |

# **Relevant Experience and Case Evidence**

This response is scoped to workflow, payload, and validation. While specific case details are not provided in the digest, Macari Company applies industry-standard engineering patterns and bilingual delivery to ensure clarity, correctness, and operator usability for internal automation projects.

- Bilingual delivery (Arabic and English)

- Focus on deterministic, testable automation

- Emphasis on operator-centric workflows

|  |  |
| --- | --- |
| **Area** | **Relevance** |
| Automation Patterns | Decoupled workers and idempotent jobs |
| Data Discipline | Typed models, schema-first validation |
| Usability | Human-in-the-loop for safety and accuracy |

# **Project Team and Roles**

A lean team structure ensures rapid delivery of the workflow, payload schema, and validation rules. Roles are defined to maintain separation of concerns across data, automation, and QA.

- Technical Lead: overall architecture, quality gates

- Backend Engineer: APIs, validation, rules engine

- Automation Engineer: Playwright flows, resilience

|  |  |
| --- | --- |
| **Role** | **Responsibilities** |
| Technical Lead | Architecture, security, review, delivery oversight |
| Backend Engineer | Ingestion, schema, validation, mapping |
| Automation Engineer | Extranet automation scripts and hardening |
| QA Engineer | Test plans, regression, UAT support |
| Project Coordinator | Schedules, stakeholder communication |

# **Work Plan, Timeline, and Milestones**

Aligned to the RFP's milestones, with emphasis on delivering workflow, payload, and constraint artifacts early for validation and sign-off. Timelines can be refined during discovery.

- Front-load workflow/payload definitions for clarity

- Parallelize validation and automation after mapping sign-off

- UAT-driven refinements before pilot

|  |  |  |
| --- | --- | --- |
| **Milestone** | **Description** | **Duration** |
| Requirements Sign-off | Finalize workflow, payload schema, constraints | 1 week |
| Prototype Development | Core ingestion, mapping, validation | 2 weeks |
| Integration & Automation Engine | Extranet automation and session management | 3 weeks |
| Testing & UAT | End-to-end tests, operator feedback | 2 weeks |
| Pilot Launch | Live run with 5 properties | 1 week |
| Full Deployment | Rollout to all properties | 1 week |

# **Quality Assurance and Risk Management**

Quality is enforced via schema validation, business-rule checks, and automated tests. Key risks include Booking.com UI changes and data quality issues. Mitigations include resilient selectors, health checks, retries, and human-in-the-loop routing when validation confidence is low.

- Unit/integration tests for mappings and validators

- Mocked automation flows for stability

- Monitoring with alerts on failures and drifts

|  |  |
| --- | --- |
| **Risk** | **Mitigation** |
| Booking.com UI changes | Selector abstraction, monitoring, quick-fix playbooks |
| Data inconsistencies | Pre-submit validations, enrichment gates, operator review |
| Session/auth failures | Robust login flows, token refresh, backoff/retry |

# **KPIs and Service Levels**

KPIs measure time-to-listing, error rates, and throughput. SLAs cover job success rates, maximum retries, and response times for incident triage.

- Reduce manual time by 80% per listing

- Achieve >95% field accuracy for critical attributes

- Track listings per week per operator

|  |  |  |
| --- | --- | --- |
| **KPI** | **Baseline** | **Target** |
| Average time per listing | 2.5 hours | ≤ 0.5 hours |
| Listing error rate | 12% correction requests | < 5% |
| Listings per operator/week | Current baseline | Significant increase (to be confirmed in pilot) |

# **Data Privacy, Security, and IP**

Access is role-based; only authorized users can trigger listing creation. Secrets are stored securely, and all actions are audited. Data flows are minimized to the least necessary scope, with sensitive data masked in logs. IP for configurations, mappings, and code is to be defined in the final agreement.

- RBAC and least-privilege access

- Encrypted secrets and transport (TLS)

- Redacted logs and structured audits

|  |  |
| --- | --- |
| **Control** | **Description** |
| Authorization | RBAC for workflow triggering and approvals |
| Secrets | Managed via secure vaulting |
| Audit | Per-field, per-action logging with timestamps |

# **Compliance with RFP Requirements**

Our solution delivers a concrete workflow, a typed property payload schema, and robust constraints aligned with the RFP's stated needs. It also supports human review when AI confidence is low and prevents unauthorized or incomplete submissions.

- Covers System Workflow with auditable stages

- Provides example property JSON payload

- Implements constraints and validation rules

|  |  |
| --- | --- |
| **RFP Area** | **Coverage** |
| System Workflow | User input → parse → validate → map → automate → audit |
| API Payload | Property, rooms, amenities, policies, images |
| Validation Rules | Mandatory checks, defaults, confidence thresholds, RBAC |

# **Deliverables Summary**

Deliverables emphasize clarity and operational readiness for Sections 6–8: workflows, payload schema, and validation rules, along with a prototype automation that demonstrates end-to-end feasibility.

- Workflow specification and swimlane diagram (textual/visual)

- Property payload JSON schema and field mapping dictionary

- Validation/constraints matrix and error-handling strategy

|  |  |
| --- | --- |
| **Deliverable** | **Description** |
| Workflow & Runbooks | End-to-end stages, retries, and operator actions |
| Payload Schema | Typed property JSON with examples |
| Validation Rules | Mandatory/optional, thresholds, fallbacks |
| Prototype | Automation against staging with audit logs |

# **Assumptions**

Assumptions clarify boundaries and responsibilities while focusing on Sections 6–8 outcomes.

- Access to Booking.com staging/sandbox for safe testing

- Internal Excel format is stable or versioned

- Image assets are available via URLs or uploader

|  |  |
| --- | --- |
| **Assumption** | **Impact** |
| Staging access | Enables safe automation testing |
| Stable form schema | Reliable mappings and validations |
| Image availability | Ensures end-to-end listing creation |

# **Pricing Approach (Summary)**

Pricing for this scope is typically structured as a fixed fee for discovery and definition (workflow, payload, validation), followed by time-and-materials for implementation and UAT. Final pricing depends on confirmed requirements and environment constraints.

- Fixed-fee discovery for clarity and alignment

- T&M for build with capped sprints

- Pilot-based adjustment before full rollout

|  |  |
| --- | --- |
| **Phase** | **Pricing Model** |
| Discovery & Design | Fixed Price |
| Implementation | Time & Materials |
| UAT & Pilot | Time & Materials (with cap) |

# **Why Macari Company**

Macari Company aligns the automation workflow, payload design, and safeguards to your operational goals. Our bilingual capability supports broader operator adoption, and our validation-first approach enhances accuracy while reducing manual workload.

- Workflow clarity and validation-first engineering

- Auditable, deterministic automation with guardrails

- Bilingual (Arabic/English) collaboration

|  |  |
| --- | --- |
| **Strength** | **Benefit** |
| Validation-first design | Higher accuracy and fewer corrections |
| Auditability | Traceable submissions and faster support |
| Cloud-agnostic stack | Flexible deployment and scaling |