Risk Management Document

Introduction

The purpose of this document is to identify and manage risks related to developing and deploying an AI-powered system that generates **EXPOZY-compatible page templates from natural language prompts** (Telegram → EXPOZY → AI Generator → validated template → preview). Because the output is **render-or-fail** (must be schema-valid and safe), the project uses layered controls (structured outputs, validation, and sandboxed preview rendering). All project risks are ranked and assigned an owner, with defined mitigations and measurable evidence.

Risk scoring method

- **Probability**: 1 (Low), 2 (Medium), 3 (High)
- **Impact**: 1 (Low), 2 (Medium), 3 (High)
- **Risk level** = Probability × Impact (1–9)

Risk ownership roles

- **Project team (Student)**: implementation, testing, documentation, monitoring configuration
- **Company supervisor / EXPOZY technical stakeholder**: confirms platform conventions (endpoints, schema patterns, security requirements)
- **CEO / Product owner**: business constraints (cost, launch readiness), acceptance criteria

Risk Register

This chapter describes each risk along with its risk level, owner, countermeasure, and evidence.

Risk No. 1

Description: Hallucinated endpoints, invalid Tailwind classes, or invented EXPOZY attributes cause broken templates (semantic failure).
Ownership: Project team and EXPOZY technical stakeholder.
Probability: 3 (High).
Impact: 3 (High).
Risk Level: 9.
Mitigation: The system enforces schema-constrained output (structured outputs) so generations cannot fail structurally. Endpoint allowlist validation is applied so only EXPOZY dot-notation formats are accepted, specifically verb.resource and Module.method. A semantic validator is added to ensure references resolve correctly

and that required elements for a given page type are present. A bounded repair loop re-prompts using validation errors to correct failures without causing infinite retries. Where needed, RAG/lookup is introduced so the model selects endpoints and components only from an approved catalog rather than inventing them.
Evidence: Automated evaluation suite metrics are collected, especially endpoint_validity rate and crossref_validity rate. Validation logs record and classify rejection reasons for invalid endpoints, classes, and invented attributes. A fixed regression set ("golden prompts") is maintained to show measurable improvement after adding schema guidance, allowlists, and semantic validation.

Risk No. 2
Description: XSS or injection payloads in text fields (content, title, href, className) lead to unsafe preview or published pages.
Ownership: Project team and EXPOZY technical stakeholder.
Probability: 2 (Medium).
Impact: 3 (High).
Risk Level: 6.
Mitigation: All prompts and model outputs are treated as untrusted input. Any rich text or HTML-like content is sanitized before rendering, and the preview sandbox renders only allowlisted components while escaping user-controlled strings. A strict Content Security Policy and security headers are deployed in the preview environment to reduce the impact of any missed injections. The test suite includes adversarial security probes that attempt common payloads such as script tags, javascript: URLs, and inline event handlers.
Evidence: The evaluation suite reports a security clean rate per run. CSP headers are verified automatically as part of deployment checks. Sanitization is covered by unit tests using known malicious payloads, demonstrating that unsafe strings are removed, escaped, or blocked from rendering.

Risk No. 3
Description: Cost blow-up due to long generations, repeated retries/repairs, or using expensive models too often.
Ownership: CEO/Product owner and Project team.
Probability: 2 (Medium).
Impact: 3 (High).
Risk Level: 6.
Mitigation: Output length is controlled by capping maxOutputTokens and limiting the number of repair attempts. Model routing is implemented so a cheaper model is used by default and escalation to a more expensive model happens only after repeated validation failure. Common contexts such as component catalogs, schema text, and few-shot examples are cached to avoid repeated token overhead. Usage metrics are collected, including tokens per request, attempts per request, and estimated cost per

generated page.
Evidence: A cost dashboard tracks average tokens, average attempts, and estimated cost per page. Logs demonstrate that token and attempt limits are enforced in production. An escalation rate metric shows what percentage of requests required expensive fallback models.

Risk No. 4

Description: Latency too high for Telegram UX (slow responses, timeouts, poor experience).

Ownership: Project team.

Probability: 2 (Medium).

Impact: 2 (Medium).

Risk Level: 4.

Mitigation: The design uses an asynchronous job queue and worker, ensuring the Telegram webhook path remains fast and does not block on AI calls. The bot provides status updates to the user while generation runs, including queued, running, and preview-ready states. Provider calls have timeouts and retry/backoff handling for transient failures. If repeated failures occur, the bot provides a fallback response such as simplifying the prompt or retrying later, rather than leaving the user waiting without feedback.

Evidence: End-to-end latency is measured from prompt submission to preview link delivery. Job queue metrics track average wait time and worker processing time. Telegram bot logs confirm immediate acknowledgement of messages and non-blocking generation behavior.

Risk No. 5

Description: Vendor lock-in (hard dependency on one cloud provider/model) slows future migration or increases long-term cost.

Ownership: Project team and CEO/Product owner.

Probability: 2 (Medium).

Impact: 2 (Medium).

Risk Level: 4.

Mitigation: AI integration is isolated inside a dedicated AI Generator Service, keeping cloud credentials and provider-specific logic outside the core backend. A model abstraction layer defines a stable interface for template generation so the system can switch models or providers with minimal changes. Templates are stored in a provider-neutral JSON Template Schema format. The evaluation harness remains provider-independent by using the same prompts and metrics across models.

Evidence: The architecture documentation shows the AI provider behind a service boundary. The system can switch model IDs/configuration without changes to core EXPOZY backend logic. Test runs demonstrate at least two candidate models generating outputs that conform to the same schema shape and validation pipeline.

Risk No. 6

Description: Model deprecation or availability changes (API changes, shutdown dates, quota issues) break production behavior.

Ownership: Project team and CEO/Product owner.

Probability: 2 (Medium).

Impact: 3 (High).

Risk Level: 6.

Mitigation: The system maintains an approved list of fallback models and can reroute generation when a preferred model is unavailable. Provider health checks classify errors such as 429 rate limits, timeouts, and 5xx failures. Lifecycle notices are monitored and reviewed monthly to detect deprecations early. A circuit breaker and graceful degradation strategy prevents job storms during provider outages, including pausing queue consumption and notifying users appropriately.

Evidence: A configurable routing table includes at least one fallback model path. Error metrics track 429 rates, timeout rates, and provider error rates over time. A short runbook describes the expected behavior if a primary model becomes unavailable.

Risk No. 7

Description: Schema drift occurs when EXPOZY storefront conventions evolve (new directives, endpoints, or section patterns), causing incompatibility.

Ownership: EXPOZY technical stakeholder and Project team.

Probability: 2 (Medium).

Impact: 3 (High).

Risk Level: 6.

Mitigation: The Template Schema is versioned so changes can be introduced safely through v1, v1.1, and later versions. Compatibility tests are executed against current storefront templates using static analysis regression to confirm core patterns are still representable. A changelog and prompt/example updates accompany schema evolution. Unsupported features are handled explicitly with fail-fast errors so the system does not silently produce wrong templates.

Evidence: The schema includes a version field and migration notes between versions. A compatibility test report maps current templates to schema constructs. Changelog entries are tied to repository changes and validation updates.

Risk No. 8

Description: Queue backlog or worker failure leads to stuck jobs, long delays, or duplicate processing.

Ownership: Project team and EXPOZY technical stakeholder.

Probability: 2 (Medium).

Impact: 2 (Medium).

Risk Level: 4.

Mitigation: Job claiming uses transactional locking or equivalent mechanisms to ensure a job is processed once even when multiple workers run. Retries use backoff and a maximum attempt limit, and jobs transition to a dead-letter FAILED state after exhaustion. Monitoring and alerting cover queue depth, job age, and failure rate. An admin view allows inspection and safe retry of failed jobs when needed.

Evidence: The queue table stores status, attempts, last_error, and updated timestamps. Monitoring graphs show queue depth and the oldest job age. A demonstration test proves the worker can recover from crashes without producing duplicate template writes.

Risk No. 9

Description: GDPR and data residency issues arise if prompts contain personal data, logs store PII, or inference occurs outside EU regions.

Ownership: CEO/Product owner and Project team.

Probability: 2 (Medium).

Impact: 3 (High).

Risk Level: 6.

Mitigation: Inference is configured for EU regions where possible to reduce cross-border processing risk. Data minimization is applied so full prompts are not logged by default and known identifiers are redacted. Retention policies define how long logs and generated drafts are kept. Data flows are documented and a DPA is maintained with relevant providers.

Evidence: A data flow diagram shows where prompt data travels and where it is stored. Logging configuration demonstrates redaction and short retention settings. Documentation confirms region selection and includes a compliance checklist for the deployment.

Risk No. 10

Description: Prompt injection or jailbreak attempts cause the model to ignore platform rules and generate unsafe or off-schema outputs.

Ownership: Project team.

Probability: 2 (Medium).

Impact: 3 (High).

Risk Level: 6.

Mitigation: Hard controls remain outside the model so schema constraints and validators determine acceptance, regardless of what the model “claims.” Endpoint and component allowlists prevent capability expansion beyond approved operations. The evaluation suite includes instruction-override adversarial prompts to test resilience. Tool/function access is capped so the model cannot call arbitrary external systems or bypass policy.

Evidence: Evaluation results on injection-style prompts show reject/repair behavior. Validator logs explain rejection reasons and demonstrate enforcement independent of the model’s instructions. A controlled demonstration shows outputs cannot exceed the allowlisted vocabulary of components and endpoints.

Risk No. 11

Description: Preview link leakage occurs if signed preview URLs are shared beyond intended recipients, exposing drafts or sensitive content.

Ownership: Project team and CEO/Product owner.

Probability: 2 (Medium).

Impact: 2 (Medium).

Risk Level: 4.

Mitigation: Preview URLs are signed and short-lived with explicit expiration. Tokens are scoped to a single draft/version to limit blast radius. Where feasible, tokens can be bound to tenant or Telegram identity context. Preview endpoints are rate limited and preview access is audited to detect unusual sharing patterns.

Evidence: Tokens include an expiry and draft scope, and the preview endpoint rejects expired or invalid tokens. Access logs show preview requests, including rate limiting events, and can be reviewed during incidents.

Risk No. 12

Description: Templates are schema-valid but low quality, meaning the output is not client-ready due to weak layout, missing sections, or incorrect data binding.

Ownership: Project team and CEO/Product owner.

Probability: 2 (Medium).

Impact: 2 (Medium).

Risk Level: 4.

Mitigation: Completeness rules are added, for example requiring products/posts sections to reference a dataSource. The validator emits quality warnings separately from hard failures and the bot suggests prompt improvements when needed. Few-shot examples are curated to match EXPOZY conventions so the model learns the desired structure. Simple UX heuristics are introduced, such as requiring at least one clear CTA and reasonable spacing conventions, while still allowing flexibility for custom pages.

Evidence: The evaluation reports warning rate versus hard failure rate to track “valid but unusable” outputs. Before/after examples demonstrate quality improvements from better schema guidance and examples. A small human review sample provides qualitative confirmation that outputs are closer to client-ready pages.