**II: Input Interpretation**

The Input Interpretation layer is the sensory cortex of Atlas PALM v1.5, serving as the entry point for user queries in the cannabis-focused Atlas Insight Engine. This layer transforms raw natural language inputs into structured, actionable data, enabling downstream layers—Intent & Journey Mapping, Mentor Matrix, and beyond—to deliver personalized, mentor-guided insights. Designed for precision, scalability, and emotional intelligence, Input Interpretation builds on v1.0’s robust parsing capabilities while introducing v1.5’s JSON Schema Export for rigorous validation, reducing error rates from 5% to <1% (Appendix D). This section provides a detailed, developer-focused overview of the layer’s functions, workflows, technical implementation, and performance, ensuring clarity for the development team, including our colleague Chat, as we prepare for the May 5, 2025, delivery. References to Appendices C (JSON Schemas) and D (Simulation Tests) anchor the technical details, supporting implementation and testing.

**2.1 Role and Importance**

Input Interpretation is the first operational tier, responsible for understanding, tagging, and preparing user inputs—queries, sentiments, and contextual cues—for accurate routing to subsequent layers. It ingests diverse inputs, from simple queries like “best strain for sleep” to complex reflections like “I tried Blue Dream and felt anxious, is that normal?” By parsing language, detecting emotions, and recalling user history, this layer ensures Atlas PALM v1.5 responds with precision and empathy, casting the user as the hero of their cannabis journey. For developers, this layer is critical for ensuring data integrity, as it validates inputs against JSON schemas (Appendix C) and flags errors before processing, supporting seamless integration via APIs (Appendix E).

**2.2 Key Functions**

The Input Interpretation layer performs six core functions, enhanced in v1.5 for scalability and global compliance:

1. **Natural Language Parsing**: Tokenizes and embeds queries using OpenAI GPT-4o’s contextual embedding, structuring text into interpretable elements (e.g., keywords, phrases). This process achieves <300ms latency, supporting 10,000 queries/second (Appendix D).
2. **Sentiment & Emotion Detection**: Classifies emotional tone (e.g., curious, anxious, excited) using linguistic cues, punctuation, and stylistic markers, with 95% accuracy validated by the Simulation Framework (Appendix D).
3. **Intent Pre-Classification**: Flags broad intent types (e.g., exploration, validation, decision-making) to guide early routing to the Intent & Journey Mapping layer, reducing misrouting by 80% compared to v1.0.
4. **Memory Recall Integration**: Retrieves user history, preferences, and journey logs from a Postgres/Redis database, using lightweight JSON profiles (Appendix C) to minimize query costs while preserving context.
5. **Journey Cue Recognition**: Detects milestone keywords and patterns (e.g., “I finally tried…” or “I feel stuck”) to signal potential stage shifts, informing mentor selection (Appendix C).
6. **Keyword & Named Entity Tagging**: Extracts cannabis-specific entities (e.g., “Blue Dream” as strain, “WA” as region) and descriptors (e.g., “anxiety relief”) for anchoring in the Domain Expertise layer (Appendix F).

**2.3 Technical Implementation**

Input Interpretation leverages a combination of advanced NLP and schema-driven validation, optimized for v1.5’s global deployment:

* **NLP Backend**: GPT-4o processes raw text, tokenizing and embedding queries for semantic analysis. v1.5’s multi-model logic enhances parsing for multilingual inputs (e.g., Spanish, French), planned for Q3 2025 (Section XI).
* **Schema Validation**: JSON Schema Export validates inputs against predefined schemas (Appendix C), ensuring structural integrity and regulatory compliance. For example, a query missing a required “query” field returns a 400 Bad Request via the API (Appendix E).
* **Database Integration**: Postgres stores user profiles and journey logs, with Redis caching for 20% faster recall compared to v1.0 (Appendix D).
* **Error Handling**: Rejects malformed inputs (e.g., empty strings, invalid regions) with detailed error codes, logged for analysis (Appendix C).

**Code Snippet**: Example schema validation logic, used by the Input Interpretation layer:

const validateInput = (input) => {

const schema = require('./input\_schema.json'); // See Appendix C

const Ajv = require('ajv');

const ajv = new Ajv({ allErrors: true });

const valid = ajv.validate(schema, input);

return valid ? { valid: true } : { valid: false, errors: ajv.errors };

};

// Example: validateInput({ query: "strain for sleep", region: "US" }) → { valid: true }

See Appendix C for the full input schema, including examples and edge-case handling.

**2.4 Interaction with Other Layers**

Input Interpretation interfaces with downstream layers to ensure seamless data flow:

* **Intent & Journey Mapping**: Passes parsed content (query text, sentiment, intent flags) and memory context, enabling accurate intent classification and journey stage detection (Appendix C).
* **Mentor Matrix**: Signals early opportunities for role transitions (e.g., Guardian for anxious users), informing archetype selection (Appendix C).
* **Feedback & Learning**: Updates session summaries with parsing insights, logged in feedback schemas (Appendix C) for continuous improvement.

For example, a parsed query with “anxious” sentiment triggers a Guardian/Sherpa mentor recommendation, validated by simulation tests (Appendix D).

**2.5 Example Processing Workflow**

Consider a user query: “I tried Blue Dream last night and I felt super anxious. Is that normal?”

* **Input**: Received via API (Appendix E), validated against input schema (Appendix C).
* **Processing**:
  1. **Parsing**: Tokenizes query, extracts “Blue Dream” (strain), “anxious” (sentiment).
  2. **Sentiment Detection**: Classifies as “concerned/anxious” (95% accuracy, Appendix D).
  3. **Intent**: Flags as “experience validation + education.”
  4. **Memory Recall**: Notes Blue Dream as previously untried in user profile (Appendix C).
  5. **Journey Cue**: Identifies “effect mismatch” milestone.
  6. **Tagging**: Tags “Blue Dream,” “anxiety” for domain lookup (Appendix F).
* **Output**: Structured JSON object:

{

"query": "I tried Blue Dream last night and I felt super anxious. Is that normal?",

"sentiment": "anxious",

"intent": "validation\_education",

"entities": { "strain": "Blue Dream" },

"memory": { "Blue\_Dream": "untried" },

"journey\_flag": "effect\_mismatch",

"region": "US",

"context": { "cultural": "therapeutic", "timestamp": "2025-05-04T12:00:00Z" }

}

* **Routing**: Recommends Guardian/Sherpa mentor, triggers reassurance and educational response (Appendix C).

This workflow, tested in 1,000+ simulation cases (Appendix D), ensures robust handling of complex queries.

**2.6 Performance Metrics**

Input Interpretation is optimized for high performance, validated by the Simulation Framework (Appendix D):

* **Latency**: <300ms for parsing and validation, supporting real-time interaction.
* **Throughput**: 10,000 queries/second, enabled by Redis caching and Node.js gateway.
* **Error Rate**: <1%, down from v1.0’s 5%, due to JSON Schema Export (Appendix C).
* **Sentiment Accuracy**: 95%, ensuring reliable emotional detection.
* **Compliance**: 100% validation of regional inputs (e.g., EU, JP), per Cultural Context Filter schemas (Appendix C).

These metrics confirm the layer’s readiness for production, supporting developers in building scalable applications.

**2.7 Design Considerations**

To ensure robustness and extensibility, Input Interpretation incorporates:

* **Latency-Sensitive Design**: Optimized for <300ms processing, critical for user experience.
* **Multilingual Extensibility**: Future-proofed for Spanish, French, and other cannabis-legal markets (Section XI), with schema support in Appendix C.
* **Bias-Aware Parsing**: Sanitizes outputs to avoid stigma, trauma, or cultural misalignment, validated through bias checks (Appendix B).
* **Lightweight Profiles**: Uses compact JSON profiles (Appendix C) to minimize query costs while preserving memory depth.

**2.8 Alignment with v1.0 and Advancements**

Compared to v1.0, v1.5’s Input Interpretation layer significantly enhances precision and compliance:

* **v1.0**: Basic parsing with GPT-4o, no schema validation, 5% error rate.
* **v1.5**: JSON Schema Export (Appendix C) for structured validation, Cultural Context Filter integration, and multilingual readiness, reducing errors to <1% (Appendix D).

Version history is documented in Appendix A, with AI contributions and ethical protocols in Appendix B.

**2.9 Developer Notes**

For Chat and the dev team:

* **Implementation**: Use the input schema in Appendix C for query validation, ensuring compliance with regional enums (e.g., “CA,” “JP”).
* **API Integration**: Submit queries via the /query endpoint (Appendix E), handling 400 Bad Request for invalid inputs.
* **Testing**: Leverage Appendix D’s simulation tests (e.g., “invalid region” cases) to validate parsing accuracy.
* **Extensibility**: Prepare for multilingual schemas (Appendix C) to support Q3 2025 roadmap goals (Section XI).
* **Debugging**: Monitor error logs for schema validation failures, cross-referenced with Appendix C examples.

**2.10 Narrative Context**

In the hero’s journey, Input Interpretation is the moment the user’s voice is heard, their words carefully parsed to begin a tailored, empathetic response. For the user, it’s the first step in a guided exploration of cannabis; for developers like Chat, it’s a robust, validated gateway ensuring every query is understood with precision and care, setting the stage for the Atlas Insight Engine’s transformative impact.