## 3. The Router Specification (Skeleton)

Language: SWI-Prolog
Integration: Python pyswip wrapper

### 3.1 Term Shapes

The following Prolog predicates define the core routing logic interface.

- route(+Request, -Decision): Main entry point.
- classify_domain(+Request, -Domain): Determines domain category.
- assess_stakes(+Request, +Domain, -Stakes): Calculates stakes level.
- select_model(+Domain, +Stakes, -Model, -Confidence): Picks best worker.
- validation_policy(+Stakes, -Policy): Determines validation rigor.
- detect_tools(+Request, -ToolsList): Identifies multimodal needs.
- overall_confidence(+Domain, +Stakes, +Model, +ModelConf, -TotalConf): Aggregates confidence.

### 3.2 Router Input Contract

The route/2 predicate accepts a Request atom or string.

- Input: Request (String) - The raw user prompt (normalized by Python before passing).
- Context: (Optional) Passed as auxiliary facts if needed (e.g., user_preference(speed)).

### 3.3 Router Output Contract

The Decision variable returns a Dict or JSON-compatible structure:

prolog

Copy

```
Decision = {
  domain: Atom,          % e.g., coding_architecture
  stakes: Atom,          % e.g., high
  recommended_model: Atom, % e.g., qwen_coder_32b
  validation_policy: Atom, % e.g., block_by_block
  tools_required: List,    % e.g., [ocr, embeddings]
  confidence: Float,       % 0.0 - 1.0
  reasoning: String        % Explanation of the decision
}
```

### 3.4 Error Semantics

- Failure to Match: If classify_domain fails, the system defaults to unknown domain.

- Low Confidence: If overall_confidence < 0.75, the uncertainty flag is set to true in the output, triggering the Python orchestrator to use embedding-based fallback.
- Prolog Exception: Caught by Python wrapper; returns a "Safe Default" decision (GPT-OSS, Medium Stakes, End-Stage Validation).

### 3.5 Routing Sequence

The logical execution flow within the route/2 predicate:

1. Normalize: Tokenize and clean input text.
2. Domain Classification: Match keywords to domain_type.
3. Stakes Assessment: Calculate risk/complexity score based on domain and keywords.
4. Model Selection: Query model_capability facts for the identified domain.
5. Validation Policy: Map stakes to validation_policy.
6. Tool Detection: Check for multimodal triggers.
7. Confidence Scoring: Compute weighted average of domain, model, and stakes confidence.
8. Reasoning Generation: Construct a human-readable explanation string.