A Rule-Based Expert System for Laptop Model Recommendation

Deliverable 1 - Project Specification and Knowledge Base Design

Course:	COMP 474 / 6741 - Intelligent Systems
Instructor:	Pankaj Kamthan
Student:	Soroush Abdolmohammadpourbonab
Student ID:	#40206879
Date:	February 20, 2026
Repository:	https://github.com/soroushamdg/bonab-comp474

Department of Computer Science and Software Engineering
Concordia University

Contents

1 Introduction 3

2 Domain Description 3
2.1 Problem Domain . . . 3
2.2 Knowledge Domain . . . 3
2.3 Suitability for Expert System Modeling . . . 3

3 Project Goal 4
3.1 Goal Statement . . . 4
3.2 SMART Analysis . . . 4

4 User Description 4
4.1 Target User . . . 4
4.2 Knowledge Limitations and Decision Challenges . . . 4
4.3 Value of Expert System Assistance . . . 5

5 Knowledge Engineering Process 5
5.1 Knowledge Acquisition . . . 5
5.2 Feature Identification . . . 5
5.3 Template Design . . . 5
5.4 Rule Granularity and Conflict Handling . . . 5
5.5 Elimination Strategy . . . 5

6 Factbase Design 6
6.1 Templates . . . 6
6.2 Laptop Knowledge Base . . . 6

7 Rulebase Design 8
7.1 Requirement Inference Rules (24 rules) . . . 8
7.2 Filtering and Elimination Rules (11 rules) . . . 8
7.3 Recommendation Rules (6 rules) . . . 9

8 System Architecture 9
8.1 File Structure . . . 9
8.2 Inference Workflow . . . 10
8.3 Loading and Execution . . . 10

9 Validation and Verification 10
9.1 Test Case 1: Developer Profile . . . 10
9.2 Test Case 2: Gamer Profile . . . 11
9.3 Test Case 3: Business Professional Profile . . . 12
9.4 Summary . . . 13

10 Explainability 13

11 Design Quality Evaluation 14

12 Limitations 14

13 Conclusion 15

14 References 16

15 AI Usage Disclosure 16

1 Introduction

Choosing a laptop is rarely as straightforward as it appears. A buyer must reconcile CPU architecture, memory capacity, GPU capability, form factor, and price into a single purchasing decision - often without the technical background to evaluate any one of those dimensions confidently. The resulting decision space is large, non-trivial to navigate, and highly sensitive to individual context: a machine that is ideal for 3D rendering is poorly suited to someone who primarily needs a portable business device.

Rule-based expert systems address exactly this kind of structured, knowledge-intensive decision problem. By encoding the reasoning patterns of a domain expert - in this case, a hardware specialist - into explicit if-then rules, the system can reproduce expert-level guidance for any user input, reliably and transparently. Unlike statistical or machine learning approaches, rule-based systems make their logic inspectable: every recommendation can be traced back to the specific rules and facts that produced it.

CLIPS (C Language Integrated Production System), developed originally at NASA's Johnson Space Center, is a well-established shell for building forward-chaining rule-based systems. Its pattern-matching inference engine, support for structured templates, and separation of facts from rules make it a natural fit for academic expert system prototyping. The system described in this report was implemented entirely within CLIPS 6.x using standard declarative constructs.

2 Domain Description

2.1 Problem Domain

The problem domain is consumer laptop selection in the personal computing market. Given a user's intended use case, budget constraint, portability preference, and peripheral requirements, the system must identify which laptop models from a predefined knowledge base are appropriate for that user.

2.2 Knowledge Domain

The knowledge domain spans the mapping between user requirements and laptop hardware specifications. This includes understanding how use cases translate into hardware demands (e.g., gaming requires discrete GPU and high-tier CPU), how budget levels constrain the available model pool, and how portability requirements impose physical constraints on weight and screen size.

2.3 Suitability for Expert System Modeling

This domain satisfies the standard criteria for expert system applicability. The reasoning involved is heuristic and experience-driven - rules like "a user doing video editing will be underserved by an integrated GPU" reflect knowledge that is not derived from a formula but from accumulated domain experience. The problem is also bounded: the number of relevant attributes is finite, the rules are non-contradictory, and outcomes are discrete (recommend or reject). Authoritative sources for the underlying technical knowledge include:

- Intel processor specifications: https://ark.intel.com

- AMD CPU documentation: `https://www.amd.com/en/products/processors/laptop`
- NVIDIA GPU product pages: `https://www.nvidia.com/en-us/geforce/laptops`
- Lenovo product specifications: `https://www.lenovo.com/us/en/laptops`
- Dell laptop lineup: `https://www.dell.com/en-us/shop/laptops`
- ASUS ROG gaming series: `https://rog.asus.com/laptops`
- Apple MacBook specifications: `https://www.apple.com/macbook-air`
- NotebookCheck benchmark database: `https://www.notebookcheck.net`

3 Project Goal

3.1 Goal Statement

The goal of this system is to assist a non-technical user in selecting an appropriate laptop model from a curated knowledge base, by reasoning over their stated usage requirements, budget level, portability needs, and feature preferences.

3.2 SMART Analysis

The goal satisfies the SMART criteria as follows. It is **Specific** in that the system targets laptop model selection, not general hardware advice. It is **Measurable** because each session produces a concrete set of recommended models, or an informative message if no model satisfies all constraints. It is **Achievable** through rule-based inference operating over a finite, well-defined knowledge base. It is **Relevant** to a real-world decision problem faced by millions of consumers. It is **Time-bound** in that each inference session completes within a single forward-chaining pass over working memory.

4 User Description

4.1 Target User

The primary intended user is a university student or early-career professional who needs to purchase a laptop but lacks the technical background to evaluate hardware specifications independently. This user understands their own needs in practical terms - “I need to write code,” “I want to play games,” “I travel frequently” - but cannot reliably translate those needs into hardware requirements.

4.2 Knowledge Limitations and Decision Challenges

This user typically does not know the performance difference between Intel Core i5 and i7 generations, cannot interpret what 16 GB DDR5 RAM implies for workload capacity, and is unsure whether an NVIDIA discrete GPU is necessary for their use case. They may also be prone to over- or under-specifying their needs when browsing product pages directly, resulting in either an overpriced purchase or an underpowered machine.

4.3 Value of Expert System Assistance

The expert system adds value by translating qualitative user inputs (use case, budget range) into concrete hardware constraints, then evaluating those constraints against a curated set of real laptop models. The user is not required to understand the inference logic; they provide five inputs and receive an explained recommendation. This mirrors the role of a knowledgeable salesperson, but with consistent, bias-free reasoning.

5 Knowledge Engineering Process

5.1 Knowledge Acquisition

The domain knowledge encoded in this system was derived from two principal sources: published hardware documentation (manufacturer specification sheets, CPU/GPU datasheets) and established best practices in the consumer laptop market (performance benchmarks, community-vetted purchasing guides). No direct elicitation from a human expert was performed, which is a limitation acknowledged in Section 12.

5.2 Feature Identification

Five user-facing attributes were identified as sufficient to drive meaningful differentiation:

1. **Primary use** - the main intended workload
2. **Budget level** - a coarse three-tier price classification
3. **Portability priority** - whether weight and form factor matter
4. **Screen size preference** - desired display size category
5. **Webcam requirement** - whether a built-in camera is necessary

These were chosen because they capture the largest share of meaningful variation in laptop selection decisions without requiring users to understand hardware directly.

5.3 Template Design

Templates were designed to separate user input, laptop knowledge, derived requirements, and system output into distinct structured fact types. Numeric slots (`ram`, `weight`, `screen-size`) use CLIPS `NUMBER` type to support arithmetic comparisons in `test` expressions, which is essential for range-based filtering.

5.4 Rule Granularity and Conflict Handling

Rules were deliberately kept narrow in scope: each rule addresses a single inference step or a single constraint check. This avoids compound rules that are difficult to debug and maintain. The system uses a salience strategy only where strict ordering is necessary (e.g., printing a user profile header before any other output). Conflict resolution otherwise follows CLIPS default strategy (recency).

5.5 Elimination Strategy

The system uses `rejected` fact assertion rather than fact retraction. This preserves the complete audit trail of the inference session: every rejected model and its reason remain visible in working memory and are printed to the console. The final recommendation rule then uses `not (rejected (model ?m))` as its eligibility condition. This approach is both transparent and safe - it avoids the risk of prematurely removing facts that may be needed by other rules.

6 Factbase Design

6.1 Templates

The system defines five deftemplates:

user captures the five input attributes described in Section 5.2. All slots are symbolic except for screen preference, which uses symbolic categories (small, medium, large, any) to reduce the input burden on the user.

laptop encodes a single model's hardware profile. It contains eleven slots: `model` (string), `year` (integer), `cpu-tier`, `storage-type`, `gpu-type`, `webcam`, `price-tier`, and `category` (all symbolic with controlled allowed-values), and `ram`, `screen-size`, and `weight` (numeric, enabling test-based comparisons).

requirement represents a derived constraint produced by the inference engine. It has three slots: `attribute` (the requirement type), `value` (symbolic qualifier), and `numeric-value` (threshold, defaulting to 0). This design allows both symbolic requirements (e.g., gpu-type = discrete) and numeric thresholds (e.g., min-ram = 16) to coexist in a single template.

rejected records a model that has been eliminated, with its `model` name and a `reason` string. Using a string reason rather than a symbolic code makes the output immediately readable to a user inspecting working memory.

recommended records a model that has passed all filters, with a brief `reason` string confirming it satisfied all derived requirements.

6.2 Laptop Knowledge Base

Sixteen laptop models were selected to span multiple categories, price tiers, and hardware configurations. The dataset was designed to ensure that each test case produces a non-trivial filtering outcome — models are drawn from business, gaming, ultrabook, and creator categories, and price tiers are distributed across low, medium, and high to exercise budget constraint rules meaningfully.

Table 1: Complete Laptop Knowledge Base – All 16 Models (2025)

Model	**Year**	**CPU Tier**	**RAM (GB)**	**Storage**	**GPU**	**Weight (kg)**	**Price Tier**	**Category**
Lenovo ThinkPad E14 Gen 6	2025	mid	16	SSD	integrated	1.42	medium	business
ASUS ROG Strix G16 2025	2025	high	32	SSD	discrete	2.34	high	gaming

Table 1: (continued)

Model	**Year**	**CPU Tier**	**RAM (GB)**	**Storage**	**GPU**	**Weight (kg)**	**Price Tier**	**Category**
Apple MacBook Air M4 13"	2025	high	16	SSD	integrated	1.24	high	ultrabook
Acer Aspire 3 2025	2025	mid	16	SSD	integrated	1.79	low	business
Dell XPS 15 9530	2025	high	32	SSD	discrete	1.86	high	creator
HP Pavilion Plus 16 2025	2025	mid	16	SSD	integrated	1.85	medium	business
MSI Thin GF63 12th Gen	2025	mid	16	SSD	discrete	1.86	medium	gaming
Lenovo IdeaPad Flex 5 Gen 9	2025	mid	16	SSD	integrated	1.49	medium	ultrabook
Microsoft Surface Laptop 7	2025	high	16	SSD	integrated	1.34	high	ultrabook
Samsung Galaxy Book5 Pro 16	2025	mid	16	SSD	integrated	1.56	high	creator
Razer Blade 16 2025	2025	high	32	SSD	discrete	2.10	high	gaming
Lenovo Legion Pro 5i Gen 10	2025	high	32	SSD	discrete	2.40	high	gaming
LG Gram 16 2025	2025	mid	16	SSD	integrated	1.40	high	ultrabook
HP Spectre x360 14 2025	2025	mid	16	SSD	integrated	1.45	high	creator
ASUS Zenbook 14 OLED 2025	2025	high	32	SSD	integrated	1.39	medium	ultrabook
Dell Inspiron 15 3000 2025	2025	low	16	SSD	integrated	1.83	low	business

The dataset intentionally includes models that appear superficially attractive but fail specific constraint combinations. The ASUS Zenbook 14 OLED, for instance, carries a high CPU tier and 32 GB RAM but has only integrated graphics, making it unsuitable for gaming or editing profiles. The Dell XPS 15 9530 carries a discrete GPU and high CPU but belongs to the creator category and survives gaming profiles precisely because the filtering logic targets hardware attributes rather than category labels.

7 Rulebase Design

The system contains **41 rules** distributed across three files. Rules are organized into six functional groups, described below with rule counts and representative code.

7.1 Requirement Inference Rules (24 rules)

Twenty-four rules populate working memory with `requirement` facts derived from the `user` fact. These are organized into five sub-groups:

Usage rules (8): Map each primary-use value to hardware requirements. Gaming asserts discrete GPU and high CPU. Development asserts minimum 16 GB RAM and SSD. Editing asserts 32 GB RAM, high CPU, discrete GPU, and SSD. Browsing asserts minimum 8 GB RAM. Business asserts webcam.

Budget rules (4): Map the three budget levels to a `max-price-tier` requirement. A low budget permits only low-tier models; medium permits low and medium; high places no restriction. A fourth advisory rule fires for the degenerate gaming-on-low-budget combination.

Portability rules (4): When `portability = yes`, assert a weight limit of 1.6 kg and a screen limit of 14.1 inches as separate rules. When `portability = no`, an advisory printout fires and screen size is delegated to the screen-preference group.

Screen preference rules (4): Active only when `portability = no`. Map `screen-pref` values to numeric `max-screen-size` requirements: `small` to 13.5 inches, `medium` to 15.6 inches, and `large` or `any` to an unrestricted sentinel value of 99.

Webcam rule (1): Fires for any user with `needs-webcam = yes`, regardless of use case. A `not` guard prevents a duplicate fact if the business rule already asserted this requirement.

GPU suitability rules (3): Browsing and business users assert a `no-discrete-gpu` requirement, directing the filtering rules to eliminate machines whose GPU adds cost and weight without benefit. Editing users assert a discrete GPU requirement for GPU-accelerated rendering.

Listing 1: Webcam requirement rule – fires for any use case

```
(defrule req-user-needs-webcam
   (user (needs-webcam yes))
   (not (requirement (attribute webcam) (value yes)))
   =>
   (assert (requirement (attribute webcam) (value yes)))
   (printout t "REQUIREMENT: User requires a built-in webcam."
       crlf))
```

7.2 Filtering and Elimination Rules (11 rules)

Eleven rules cross-reference laptop attributes against derived requirements and assert `rejected` facts for violations. The original nine rules handle RAM, GPU type, CPU tier, weight, screen size, price tier (two rules for low and medium budget separately), webcam, and storage type. Two additional rules were introduced to tighten recommendations when the knowledge base contains many high-specification models:

F10 — Discrete GPU elimination: When the `no-discrete-gpu` requirement is active, any laptop with a discrete GPU is rejected. This prevents browsing and business profiles from receiving gaming-class hardware recommendations.

F11 — Gaming category elimination: For any non-gaming user, laptops in the `gaming` category are rejected. Gaming machines sacrifice battery life, fan noise profile, and portability for sustained GPU throughput — characteristics that are liabilities rather than assets for productivity use cases.

Listing 2: Gaming category filter for non-gaming users

```
(defrule filter-gaming-category-for-non-gaming-user
   (user (primary-use ?u))
   (test (not (eq ?u gaming)))
   (laptop (model ?m) (category gaming))
   (not (rejected (model ?m) (reason "Gaming category
      unsuitable for this use case")))
   =>
   (assert (rejected (model ?m) (reason "Gaming category
      unsuitable for this use case")))
   (printout t "REJECTED: " ?m " -- gaming-category laptop is
      unsuitable for "
             ?u " use." crlf))
```

7.3 Recommendation Rules (6 rules)

The six recommendation rules operate after filtering is complete. The core rule asserts a `recommended` fact for every laptop with no matching `rejected` fact. A companion explanation rule prints a full hardware profile for each recommendation. A high-salience header rule controls output ordering. A fallback rule fires if no `recommended` fact is produced, advising the user to relax constraints. A user profile display rule and a rejection summary advisory complete the group.

8 System Architecture

8.1 File Structure

The system is organized into six CLIPS source files and three test files:

Listing 3: Repository file structure

```
Laptop-Expert-System/
  clips/
    templates.clp              -- 5 deftemplate definitions
    laptop-facts.clp           -- 16 laptop model instances (
       deffacts)
    requirement-rules.clp      -- 24 requirement inference
       rules
    filtering-rules.clp        -- 11 elimination rules
    recommendation-rules.clp  -- 6 recommendation and output
       rules
    main.clp                  -- Batch loader entry point
    test-case-1.clp           -- Developer, medium budget,
       portability yes
    test-case-2.clp           -- Gamer, high budget,
       portability no
    test-case-3.clp           -- Business, low budget,
       portability no
  docs/
    Deliverable1_Report.pdf
```

8.2 Inference Workflow

The system follows a three-phase forward-chaining pipeline. In the first phase, requirement inference rules translate the single `user` fact into a set of `requirement` facts covering hardware thresholds, budget constraints, and portability limits. In the second phase, filtering rules cross-reference all 16 `laptop` facts against the active requirements and assert `rejected` facts for violations. In the third phase, recommendation rules collect all laptops with no `rejected` facts and produce `recommended` facts with printed hardware summaries.

Phase separation is enforced by data dependency: filtering rules cannot fire until requirements exist, and recommendation rules cannot fire until filtering has had the opportunity to run. This structure eliminates the need for explicit salience manipulation in most rules.

8.3 Loading and Execution

The system is loaded via the CLIPS `batch` command. Each test file calls `load` on all five source files, then `reset` to instantiate the 16 `laptop` facts from the `deffacts` block, then asserts a fresh `user` fact before calling `run`.

9 Validation and Verification

Three test cases were executed against the 16-model knowledge base using the updated 41-rule system. Results below reflect actual CLIPS console output.

9.1 Test Case 1: Developer Profile

Input: primary-use = development, budget-level = medium, portability = yes, needs-webcam = yes, screen-pref = medium

Inferred Requirements:

- min-ram = 16 GB (development workload)
- storage-type = SSD (development workload)
- webcam = yes (user requirement)
- max-price-tier = medium (budget)

- max-weight = 1.6 kg (portability)
- max-screen-size = 14.1 inches (portability)

Rejection summary (13 models eliminated):

- *Lenovo Legion Pro 5i Gen 10, Razer Blade 16, MSI Thin GF63, ASUS ROG Strix G16* – gaming category unsuitable for development use (F11)
- *HP Spectre x360, LG Gram 16, Legion Pro, Razer Blade, Samsung Galaxy Book5, Surface Laptop 7, Dell XPS 15, Apple MacBook Air M4, ASUS ROG Strix G16* – price tier high exceeds medium budget (F7)
- *Dell Inspiron 15, Legion Pro, Razer Blade, MSI GF63, HP Pavilion Plus, Dell XPS 15, Acer Aspire 3, ASUS ROG Strix G16* – weight exceeds 1.6 kg limit (F4)
- *Dell Inspiron 15, LG Gram 16, Legion Pro, Razer Blade, Samsung Galaxy Book5, MSI GF63, HP Pavilion Plus, Dell XPS 15, Acer Aspire 3, ASUS ROG Strix G16* – screen exceeds 14.1 inch limit (F5)

Recommended (3 models):

- **ASUS Zenbook 14 OLED 2025** – high CPU, 32 GB RAM, SSD, integrated GPU, 14;, 1.39 kg, medium
- **Lenovo IdeaPad Flex 5 Gen 9** – mid CPU, 16 GB RAM, SSD, integrated GPU, 14;, 1.49 kg, medium
- **Lenovo ThinkPad E14 Gen 6** – mid CPU, 16 GB RAM, SSD, integrated GPU, 14;, 1.42 kg, medium

Analysis: Portability is the dominant constraint, eliminating all machines heavier than 1.6 kg or wider than 14.1 inches. The F11 gaming-category rule eliminates four additional machines before budget filtering even fires. The three surviving models are all 14-inch designs at or below the weight threshold with medium price-tier pricing. The ASUS Zenbook 14 OLED 2025 stands out within the recommendation set for its 32 GB RAM and high-tier CPU, making it the strongest option for demanding development workloads.

9.2 Test Case 2: Gamer Profile

Input: primary-use = gaming, budget-level = high, portability = no, screen-pref = large, needs-webcam = no

Inferred Requirements:

- gpu-type = discrete (gaming workload)
- cpu-tier = high (gaming workload)
- storage-type = SSD (gaming workload)
- max-price-tier = any (high budget – no restriction)
- max-screen-size = 99 (screen-pref large – no restriction)
- max-weight = no constraint (portability = no)

Rejection summary (12 models eliminated):

- *Dell Inspiron 15, ASUS Zenbook 14 OLED, HP Spectre x360, LG Gram 16, Sam-*

sung Galaxy Book5, Surface Laptop 7, Lenovo IdeaPad Flex 5, HP Pavilion Plus, Acer Aspire 3, Apple MacBook Air M4, Lenovo ThinkPad E14 Gen 6 – integrated GPU (F2)

- *Dell Inspiron 15, HP Spectre x360, LG Gram 16, Samsung Galaxy Book5, Lenovo IdeaPad Flex 5, MSI GF63, HP Pavilion Plus, Acer Aspire 3, Lenovo ThinkPad E14 Gen 6* – CPU tier low or mid (F3)

Recommended (4 models):

- **Lenovo Legion Pro 5i Gen 10** – high CPU, 32 GB RAM, SSD, discrete GPU, 16; 2.4 kg, high
- **Razer Blade 16 2025** – high CPU, 32 GB RAM, SSD, discrete GPU, 16; 2.1 kg, high
- **Dell XPS 15 9530** – high CPU, 32 GB RAM, SSD, discrete GPU, 15.6; 1.86 kg, high
- **ASUS ROG Strix G16 2025** – high CPU, 32 GB RAM, SSD, discrete GPU, 16; 2.34 kg, high

Analysis: With a high budget and no portability constraint, price and weight filters place no restriction on the recommendation set. The only active hardware requirements are discrete GPU and high CPU tier. All four surviving models satisfy both simultaneously, and all carry 32 GB RAM and SSD storage. The Dell XPS 15 9530, though a creator-category machine rather than a pure gaming laptop, passes all filters because the filtering logic targets hardware attributes, not category labels. This is a deliberate design choice: a developer or editor who also games would benefit from the XPS 15's more balanced thermal profile.

9.3 Test Case 3: Business Professional Profile

Input: primary-use = business, budget-level = low, portability = no, screen-pref = any, needs-webcam = yes

Inferred Requirements:

- webcam = yes (business workload + explicit user requirement)
- no-discrete-gpu = yes (business workload)
- max-price-tier = low (low budget)
- max-screen-size = 99 (screen-pref any – no restriction)
- max-weight = no constraint (portability = no)

Rejection summary (14 models eliminated):

- *Lenovo Legion Pro 5i Gen 10, Razer Blade 16, MSI Thin GF63, Dell XPS 15, ASUS ROG Strix G16* – discrete GPU unnecessary for business use (F10)
- *Lenovo Legion Pro 5i Gen 10, Razer Blade 16, MSI Thin GF63, ASUS ROG Strix G16* – gaming category unsuitable for business use (F11)
- *ASUS Zenbook 14 OLED, HP Spectre x360, LG Gram 16, Legion Pro, Razer Blade, Samsung Galaxy Book5, Surface Laptop 7, Lenovo IdeaPad Flex 5, MSI GF63, HP*

Pavilion Plus, Dell XPS 15, Apple MacBook Air M4, ASUS ROG Strix G16, Lenovo ThinkPad E14 Gen 6 – price tier medium or high exceeds low budget (F6, F7)

Recommended (2 models):

- **Dell Inspiron 15 3000 2025** – low CPU, 16 GB RAM, SSD, integrated GPU, 15.6·, 1.83 kg, low
- **Acer Aspire 3 2025** – mid CPU, 16 GB RAM, SSD, integrated GPU, 15.6·, 1.79 kg, low

Analysis: Budget is the dominant constraint, accounting for the elimination of 14 out of 16 models. The GPU and gaming-category filters fire first and remove the discrete GPU machines, but budget filtering would have caught most of those anyway. Both surviving models are the only low-price-tier entries in the knowledge base, both carry webcams, and both use integrated graphics appropriate for business workloads. Between the two, the Acer Aspire 3 has a mid-tier CPU compared to the Inspiron's low-tier CPU, making it the stronger recommendation for users whose budget is the primary constraint.

9.4 Summary

Table 2 summarises the outcome of all three test cases against the 16-model knowledge base.

Test	Profile	Dominant Filter	Rejected	Recommended
TC-1	Developer, medium, portability yes	Portability + budget	13 of 16	Zenbook 14 OLED, Flex 5 Gen 9, ThinkPad E14 Gen 6
TC-2	Gamer, high budget, portability no	GPU + CPU tier	12 of 16	ROG Strix G16, Razer Blade 16, Dell XPS 15, Legion Pro 5i
TC-3	Business, low budget, portability no	Budget	14 of 16	Acer Aspire 3, Dell Inspiron 15

Table 2: Test case summary – verified against actual CLIPS output with 16-model fact base

10 Explainability

Explainability is a core design objective of this system. Three mechanisms work together to make the reasoning transparent.

First, every requirement inference rule includes a `printout` statement that describes, in plain English, why a requirement was asserted. This gives the user visibility into the first phase of inference without needing to inspect working memory directly.

Second, every rejection rule prints the model name, the specific attribute value that caused rejection, and the threshold it failed to meet. This makes each elimination auditable: a user or developer can read the console output sequentially and reconstruct the full chain of reasoning.

Third, the recommendation explanation rule prints a complete hardware profile for every recommended model, allowing the user to confirm that the recommendation aligns with what they asked for. This is particularly important in cases like Test Case 1, where two models are recommended and the user may need to choose between them.

This level of transparency is not incidental - it is architecturally enforced. Because the system uses `rejected` fact assertion rather than fact deletion, every intermediate conclusion persists in working memory for the duration of the session and is available for inspection via CLIPS's `facts` command.

11 Design Quality Evaluation

Modularity: Each CLIPS file has a single, well-defined responsibility. Templates define structure. Facts encode knowledge. Rules are split by function across three files. This separation means that adding a new laptop model requires editing only `laptop-facts.clp`, and adding a new use-case rule requires editing only `requirement-rules.clp`. Neither change affects any other file.

Maintainability: Rule names are descriptive and follow a consistent naming convention (`category-condition-consequence`). Each rule group is preceded by a block comment. This makes the codebase navigable even to a reader unfamiliar with CLIPS.

Portability: The system uses only standard CLIPS 6.x constructs and no external dependencies. It compiles and runs on any platform with a standard CLIPS interpreter. The test files use `batch` rather than hard-coded initialization, making them self-contained and reproducible.

Reusability: The template definitions in `templates.clp` are generic enough to accommodate a significantly expanded laptop dataset without modification. The `requirement` template's dual slot design (symbolic value + numeric threshold) generalizes to any constraint that can be expressed as either a category or a range.

Balanced Rule Granularity: Rules deliberately avoid compound conditions. A gaming user triggers two separate requirement rules - one for GPU and one for CPU - rather than a single rule with both conclusions. This granularity makes the rule set easier to extend (adding a third gaming requirement does not require modifying existing rules) and easier to debug (each rule can be traced individually in the agenda).

12 Limitations

The system has several limitations that are important to acknowledge honestly.

The laptop dataset is static and contains only sixteen models. Real-world purchasing decisions involve hundreds of configurations, and the system cannot generalize to models outside its knowledge base. Adding new models requires manual authorship of `laptop` facts.

Price tiers are coarse categorical abstractions (low, medium, high) rather than actual prices. Two laptops in the same tier may differ by hundreds of dollars in practice, but the system treats them identically. A more precise system would use actual price values and allow the user to specify a numeric budget.

The system contains no learning component. Its rules encode fixed heuristics and do not improve with use. If a user disagrees with a recommendation, the system has no mechanism to update its reasoning accordingly.

There is no handling of uncertainty or partial preference. If a user is indifferent between two equally valid models, the system recommends both without any mechanism to rank or prioritize them further.

Finally, the knowledge base was derived from published specifications and general domain knowledge rather than from a structured elicitation session with a human laptop specialist. Some rules may oversimplify domain nuances - for example, treating integrated GPU as categorically insufficient for editing, when modern integrated graphics (such as Apple M-series) may be adequate for light creative work.

13 Conclusion

This report has described the design, implementation, and validation of a rule-based expert system for laptop model recommendation, implemented in CLIPS 6.x. The submitted system consists of 41 rules across three functional files, operating over a curated knowledge base of 16 laptop models drawn from 2025 market offerings across business, gaming, ultrabook, and creator categories.

The knowledge engineering process revealed a recurring challenge in expert system design: as the fact base grows, rules that were sufficient for a smaller dataset become insufficiently discriminating. The original system was designed around eight models, and several rules — particularly the RAM minimum threshold and the absence of GPU-suitability constraints — produced near-universal recommendations when the fact base expanded to 16 models all carrying 16 GB or more of RAM. Addressing this required two additional requirement inference rules (GPU suitability for browsing and business profiles) and two additional filter rules (discrete GPU elimination and gaming-category elimination), bringing the total from the initial 31 to 41. This refinement cycle is representative of real knowledge engineering practice: knowledge bases require iteration as their coverage grows.

The three test cases demonstrate that the system produces logically consistent, traceable recommendations across diverse user profiles. Portability is the dominant discriminator in Test Case 1, reducing 16 candidates to 3. Hardware requirements (discrete GPU, high CPU tier) are the dominant discriminators in Test Case 2, reducing 16 candidates to 4. Budget is the dominant discriminator in Test Case 3, reducing 16 candidates to 2. In each case, every elimination can be traced to a specific fired rule and a specific fact in working memory — a transparency property that statistical models do not provide without additional interpretability tooling.

The system remains limited in scope: it operates over a static, manually authored fact base, uses coarse three-tier classifications for price and CPU, and contains no mechanism for ranking recommendations when multiple models survive filtering. These are appropriate limitations for an academic prototype and are documented in Section 12. The core contribution of the project — demonstrating that a small, well-structured rule base can perform explainable, multi-attribute filtering over a realistic product dataset — is validated by the test results.

14 References

1. Intel Corporation. *Intel ARK - Product Specifications.* https://ark.intel.com
2. Advanced Micro Devices. *AMD Laptop Processors.* https://www.amd.com/en/products/processors/laptop
3. NVIDIA Corporation. *GeForce Laptop GPUs.* https://www.nvidia.com/en-us/geforce/laptops
4. Lenovo Group. *Lenovo ThinkPad E14 Gen 4 Product Page.* https://www.lenovo.com/us/en/laptops
5. Dell Technologies. *Dell XPS 15 Specifications.* https://www.dell.com/en-us/shop/laptops
6. ASUSTeK Computer Inc. *ASUS ROG Strix G15.* https://rog.asus.com/laptops
7. Apple Inc. *MacBook Air with M2 chip - Technical Specifications.* https://www.apple.com/macbook-air
8. NotebookCheck. *Laptop Reviews and Benchmarks.* https://www.notebookcheck.net
9. Riley, G. (2015). *CLIPS Reference Manual, Version 6.3.* NASA Johnson Space Center.
10. Giarratano, J., & Riley, G. (2004). *Expert Systems: Principles and Programming*, 4th ed. Course Technology.

15 AI Usage Disclosure

In accordance with course policy, this section documents the use of AI-assisted tools during the preparation of this deliverable.

Tool used: Claude (Anthropic), accessed via https://claude.ai.

Purpose: The AI assistant was used to support the initial structuring of CLIPS rule templates, to check rule syntax against CLIPS 6.x conventions, and to assist in drafting sections of this report. All design decisions - including the selection of laptop models, the choice of attributes, rule categorization, and test case construction - were made by the student author and reviewed for correctness and consistency.

Scope: The AI was not used to perform knowledge elicitation, evaluate recommendations, or substitute for understanding of the domain or the CLIPS inference model. All generated content was reviewed, revised, and validated by the author prior to submission.

Verification: All CLIPS code was compiled and executed in the CLIPS 6.x IDE. Test results reported in Section 9 reflect actual system output observed during live execution sessions.