PRD

AI Car Variant Comparison System

Version: 1.0

Project Type: Hackathon MVP with Production Roadmap

Timeline: 7-8 Days

---

1. PRODUCT VISION

1.1 The Problem

Car buyers get confused choosing between variants of the same model (Swift LXi vs VXi vs ZXi+). They don't know if paying ₹50k-₹2L extra is worth the additional features.

1.2 The Solution

An AI agent that shows your selected variant AND intelligently suggests 2 better variants from the same model, explaining exactly what extra features you get for the additional cost.

1.3 One-Line Pitch

"Smart variant upgrade advisor that tells you: pay ₹X more, get these Y features"

---

2. USER FLOW

Step 1: Selection

```
User lands on page
↓
Dropdown 1: Select Brand (Maruti, Hyundai, Tata, etc.)
↓
Dropdown 2: Select Model (Swift, Creta, Nexon, etc.)
```

```
↓
Dropdown 3: Select Variant (LXi, VXi, ZXi, ZXi+)
↓
Click "Show Details"
```

Step 2: Display

```
┌──────────────────────────────────────────┐
│  YOUR SELECTION: Swift VXi               │
│  Price: ₹6,99,000                        │
│  Features:                               │
│  ✓ 4 Airbags                             │
│  ✓ Manual AC                             │
│  ✓ Fabric Seats                          │
│  ✓ Basic Touchscreen                     │
└──────────────────────────────────────────┘
```

Step 3: AI Suggestions (Agent-Powered)

```
┌──────────────────────────────────────────┐
│  🤖 SMART UPGRADE OPTIONS                │
└──────────────────────────────────────────┘

┌──────────────────────────────────────────┐
│ Option 1: Swift ZXi                      │
│ Pay ₹60,000 more, get:                   │
│ ✓ Automatic Climate Control              │
│ ✓ Alloy Wheels                           │
│ ✓ Rear Parking Sensors                   │
│ [Consider This]                          │
└──────────────────────────────────────────┘

┌──────────────────────────────────────────┐
│ Option 2: Swift ZXi+                     │
│ Pay ₹1,50,000 more, get:                 │
```

```
│ ✓ Sunroof                    │
│ ✓ 6 Airbags (2 extra)        │
│ ✓ Cruise Control             │
│ ✓ Wireless Charger           │
│ [Premium Choice]             │
└──────────────────────────────┘
```

Edge Case

If user selects **top variant** (ZXi+):

```
┌──────────────────────────────┐
│ 🎉 You've selected the top variant! │
│ This has all available features.    │
└──────────────────────────────┘
```

---

3. DATA ARCHITECTURE

3.1 Hierarchical Structure

```
Level 1: MAKE (Brand)
├── Maruti
├── Hyundai
├── Tata
└── Mahindra

Level 2: MODEL (Car Name)
├── Maruti
│   ├── Swift
│   ├── Baleno
│   └── Brezza

Level 3: VARIANT (Trim)
├── Swift
│   ├── LXi (Base)
```

```
│   ├── VXi (Mid)
│   ├── ZXi (High)
│   └── ZXi+ (Top)
```

3.2 Database Structure (MongoDB/Vector DB)

Document Schema:

```
{
  "_id": "swift_zxi_2024",
  "make": "Maruti",
  "model": "Swift",
  "variant_name": "ZXi",
  "variant_tier": "high",  // base, mid, high, top
  "price": 799000,

  "features": {
    "safety": ["4 airbags", "ABS", "EBD"],
    "comfort": ["Auto AC", "Rear AC vents", "Cruise control"],
    "tech": ["7-inch touchscreen", "Apple CarPlay"],
    "exterior": ["Alloy wheels", "LED DRLs"]
  },

  "feature_embedding": [0.23, -0.45, ...],  // For vector search

  "upgrade_path": {
    "next_variant": "ZXi+",
    "price_diff": 50000,
    "additional_features": ["Sunroof", "2 extra airbags"]
  }
}
```

---

4. AGENT LOGIC

4.1 Agent Responsibilities

Input: User selected variant (e.g., Swift VXi)

Agent Tasks:

1. Fetch selected variant details from database
2. Find 2 higher variants in same model
3. Calculate price difference
4. Extract unique features in higher variants
5. Generate recommendation message

Output:

- Display selected variant
- Show 2 upgrade options with price delta + features

4.2 Recommendation Rules

Rule 1: Suggest Next 2 Higher Tiers

- If user selects `base` → suggest `mid` and `high`
- If user selects `mid` → suggest `high` and `top`
- If user selects `high` → suggest `top` only

Rule 2: Calculate Value Proposition

- Price difference ÷ Number of extra features = Cost per feature
- Show message: "₹25,000 per additional feature"

Rule 3: Top Variant Handling

- If user selects `top` variant → No suggestions
- Display: "You've chosen the fully loaded variant!"

---

5. TECHNOLOGY STACK

5.1 Hackathon MVP

Component	Technology	Justification
Frontend	Streamlit	Quick UI, no React needed
Agent	LangChain + Gemini	Free tier, smart reasoning
Database	MongoDB Atlas / Local Vector DB	Free tier, flexible schema
Embedding	SentenceTransformer	Runs locally, no API cost
Data Source	Kaggle Indian Cars Dataset	Ready-to-use variant data

5.2 No Machine Learning Required

- **Why:** All prices are fixed (not predicted)
- **What we use instead:** Simple business logic + agent reasoning

---

6. FEATURE SCOPE

6.1 Hackathon MVP (Week 1)

Must Have:

- ✅ 3-level dropdown (Make → Model → Variant)
- ✅ Display selected variant (price + features)
- ✅ Agent suggests 2 upgrade options
- ✅ Price difference + feature list
- ✅ Works for 5 models (Swift, Creta, Nexon, Venue, Seltos)

Nice to Have:

- 🟡 Feature comparison table (side-by-side)
- 🟡 Agent reasoning trace (show tool calls)

Out of Scope:

- ❌ Cross-model comparison (Swift vs i20)
- ❌ Used car resale prediction
- ❌ Multi-turn chat conversation
- ❌ Image upload for variant detection

6.2 Future Enhancements (Post-Hackathon)

Phase 2: Contextual Recommendations (Month 1-2)

- Add user context input: "I have 2 kids, drive in city"
- Agent weighs features based on needs (safety > luxury for families)
- Personalized recommendations

Phase 3: Comparison Mode (Month 3)

- Compare any 2 variants side-by-side
- Highlight differences in table format
- Show real-world owner reviews

Phase 4: Financial Tools (Month 4-6)

- EMI calculator integration
- Insurance cost estimates
- Total cost of ownership (5-year projection)
- Resale value prediction (add ML model here)

Phase 5: Production Features (Month 6-12)

- Multi-brand database (50+ models)
- Real-time price updates (API integration with CarDekho)
- User accounts + saved comparisons
- Test drive booking integration
- WhatsApp/Email sharing of recommendations

---

7. SUCCESS METRICS

7.1 Hackathon Demo Goals

- ✅ System responds in < 10 seconds
- ✅ Agent makes correct suggestions 95% of time

- ✅ UI is clean and intuitive (no confusion)
- ✅ Works for all 5 demo models without errors

7.2 User Value Metrics (Future)

- **Decision Clarity:** Users understand upgrade cost-benefit
- **Time Saved:** 5 minutes vs 30 minutes manual research
- **Confidence:** 80%+ users feel informed to decide

7.3 Technical Metrics (Future)

- Database query latency < 200ms
- Agent reasoning depth: 2-3 tool calls per query
- Uptime: 99% (production)

---

8. USER STORIES

Story 1: Budget-Conscious Buyer

As a first-time car buyer with limited budget

I want to see if upgrading to next variant is worth ₹50k

So that I don't overspend on features I won't use

Acceptance:

- Agent shows price difference clearly
- Lists exact features I'm paying for
- I can decide if features justify cost

Story 2: Family Safety Priority

As a parent with 2 kids

I want to know which variant has best safety features

So that I can prioritize airbags over luxury features

Acceptance (Future):

- I can tell agent "I prioritize safety"
- Agent highlights safety upgrades (airbags, ESP)
- Recommendation ranks safety variants higher

Story 3: Feature Explorer

As a tech enthusiast

I want to see all features in top variant

So that I know what maximum I can get in this model

Acceptance:

- Selecting top variant shows full feature list
- No upgrade suggestions (already maxed out)
- Clear message: "This is fully loaded"

---

9. HACKATHON EXECUTION PLAN

Day 1-2: Data Preparation

- Download Kaggle dataset
- Clean and structure data (Make → Model → Variant hierarchy)
- Upload to MongoDB Atlas / Local Vector DB
- Generate embeddings for variants

Day 3-4: Agent Development

- Set up LangChain + Gemini API
- Create agent tools (fetch variant, find upgrades, calculate diff)
- Test agent logic with sample queries

Day 5-6: UI Development

- Build Streamlit interface (dropdowns, display cards)
- Connect UI to agent backend

- Test end-to-end flow

Day 7: Polish & Demo

- Fix bugs, improve UI styling
- Prepare demo script with 3 scenarios
- Record demo video (3 minutes)

---

10. RISKS & MITIGATIONS

Risk	Impact	Mitigation
Dataset incomplete (missing variants)	Can't demo all models	Manually add 20-30 key variants
Agent suggests wrong variant	Demo fails	Hardcode upgrade paths as fallback
Gemini API rate limit	System breaks during demo	Cache responses for demo queries
UI looks unprofessional	Low judge scores	Use Streamlit themes, focus on clarity

---

11. OPEN QUESTIONS

1. **Data Storage:** MongoDB Atlas (cloud, free tier) or Local ChromaDB (faster, no internet dependency)?
2. **Number of Models:** Start with 5 or cover 10+ for broader appeal?
3. **Agent Verbosity:** Show full reasoning trace or just final recommendation?

---

12. GO/NO-GO CRITERIA

Proceed to TRD if:

- ✅ Clear understanding of 3-level data hierarchy
- ✅ Agent logic is simple (no complex ML)
- ✅ Scope is achievable in 7 days

Red Flags:

- ❌ Dataset doesn't have variant-level granularity
- ❌ Agent logic becomes too complex
- ❌ MongoDB setup takes > 1 day

Status: Ready for Technical Requirements Document (TRD)

Next Step: Define exact database schema, agent tool functions, and API contracts

---