

# Working Log

You are a technical product manager. Write a concise Product Requirements Document for a hackathon project.

The product is an AI car variant comparison system that helps users choose between trim levels of the same car model, for example Swift LXi vs VXi vs ZXi Plus.

## Context

Hackathon duration is 7 to 8 days.

UI is built using Streamlit.

Show more

## Product Requirements Document: AI Car Variant Comparison System

### Product Vision

An AI-assisted decision support tool that simplifies car variant selection within the same model. Users select a variant and receive intelligent recommendations for higher-tier options with clear price-to-feature justification. The system bridges the information gap between sparse dealership brochures and overwhelming specification sheets. Target audience: first-time car buyers and budget-conscious consumers in the Indian market. Core value proposition: transparent variant comparison with actionable upgrade recommendations. The LLM layer provides natural language summarization of structured data, not predictive analytics or subjective advice.

Reply...

+ ⏪

Sonnet 4.5 ▾

↑

Based on the above PRD, create a Technical Requirements Document detailing architecture, database schema, backend APIs, agent flow, and deployment assumptions.

## Technical Requirements Document: AI Car Variant Comparison System

### System Architecture

**Architecture Pattern:** Three-tier monolithic application with clear separation of concerns.

**Presentation Layer:** Streamlit frontend handling user interactions and rendering.

**Business Logic Layer:** Python service modules for variant retrieval, comparison logic, and feature differencing.

**Data Layer:** MongoDB for persistent storage with connection pooling.

**LLM Integration:** External API calls to Gemini via LangChain, treated as formatting utility not core logic.

**Communication Flow:** Streamlit → Service Layer → Database/LLM → Response formatting → UI update.

**State Management:** Streamlit session state for dropdown selections and cached query results.



**Deployment Model:** Single containerized application deployed on Streamlit Community

Reply...



Sonnet 4.5



M Manvendra Prata...  Mini Hackathon / PRD Edited 1d ago Share ⌂ ...

Search Home Meetings Notion AI Inbox

Private

- Mini Hackathon
- PRD
- TRD
- Prompt
- GPU Programming
- HTM Innovation Expo: ...
- Basic Widget in Flutter
- Flutter Mobile Developm...
- Autonomous Newsroom ...
- Emotion-Aware News: D...
- AI-tool Comparison
- Running a Swift Project i...

Teamspaces

# PRD

## AI Car Variant Comparison System

Version: 1.0  
Project Type: Hackathon MVP with Production Roadmap  
Timeline: 7-8 Days

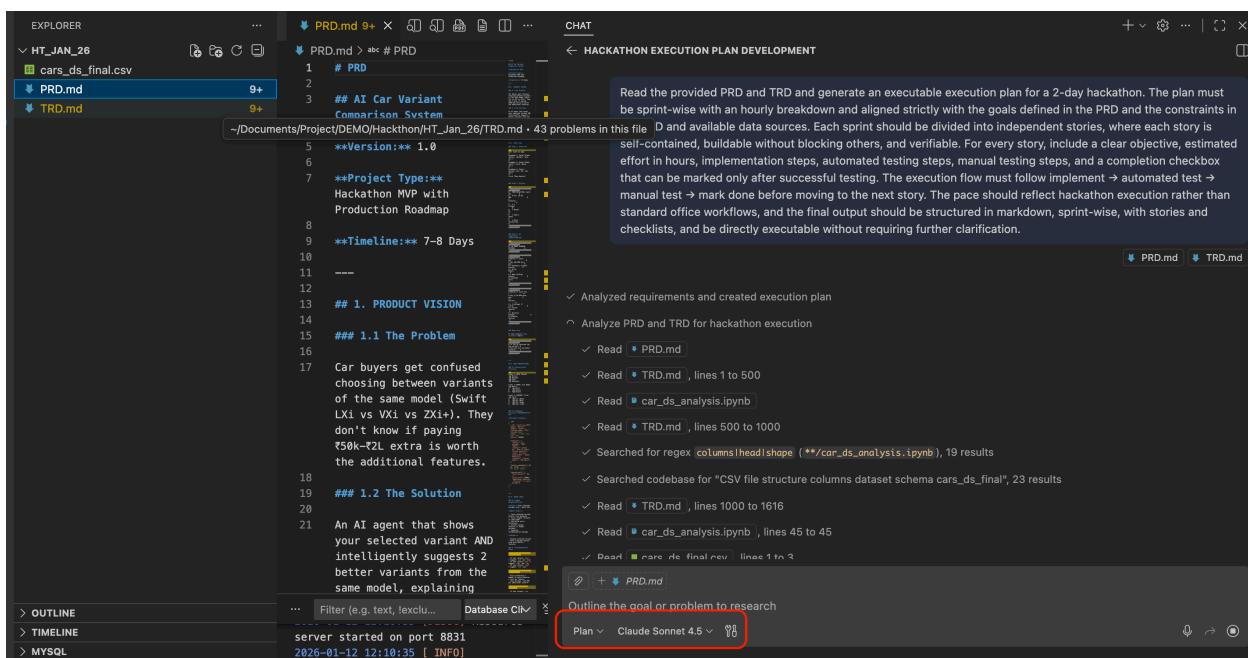
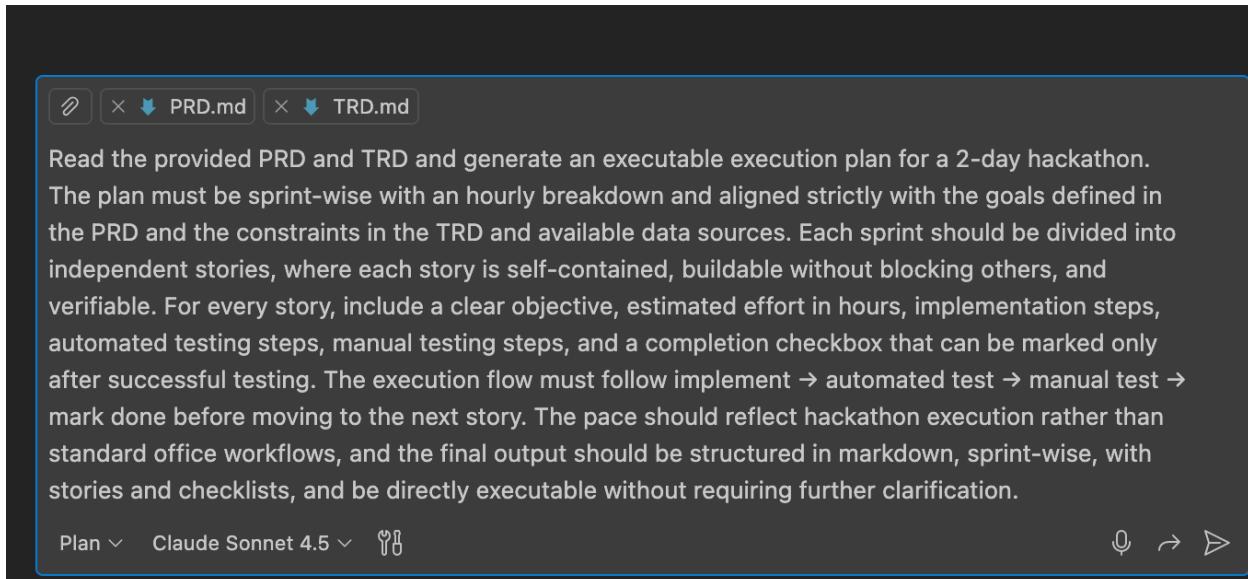
### 1. PRODUCT VISION

#### 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.



#### Sprint 6: Documentation & Submission (Hours 43-48, Day 2 Late Evening)

1. [Story 6.1] Write README.md with local setup instructions, Gemini API key configuration, run commands (1.5 hours)
2. [Story 6.2] Create requirements.txt with pinned versions, .env.example template, .gitignore (30 mins)
3. [Story 6.3] Document agent architecture and tool functions with inline comments (1.5 hours)
4. [Story 6.4] Create presentation slides showing PRD vision, TRD architecture, live demo screenshots, future roadmap (2.5 hours)

#### Further Considerations

1. **Gemini Rate Limits:** Free tier allows 60 req/min - implement exponential backoff retry logic? Or acceptable for demo scenario?
2. **Model Coverage:** 5 models (faster testing) vs 7 models (broader appeal)? Balance between polish and coverage.
3. **Agent Verbosity:** Show LangChain reasoning trace in UI expander for transparency, or hide for cleaner UX?



Claude Sonnet 4.5 • 1x



Gemini Rate Limit no problem  
Model Coverage -> go with all data  
Agent Verbosity -> like copilot show, show while thing in small box.. hide with button to show (after process)

- ✓ Updated plan to reflect new considerations.

#### Plan: 2-Day Hackathon Execution for AI Car Variant Advisor (FINAL)

PROCEED FROM PLAN

[Start Implementation](#) | [Open in Editor](#)



PRD.md

Outline the goal or problem to research

Plan ▾ Claude Sonnet 4.5 ▾

- .gitignore (30 mins)
- 3. [Story 6.3] Add inline documentation to agent tools, orchestrator logic, and data processing scripts (1 hour)
- 4. [Story 6.4] Prepare presentation slides - PRD problem statement, TRD architecture diagram, demo screenshots with agent trace, future production roadmap (3 hours)

#### Further Considerations

- Data Quality:** If tier inference fails for some models, acceptable to exclude those variants or manually fix critical models?
- Feature Mapping Coverage:** Target 80% feature coverage from 141 columns or focus on 40-50 most common features for consistency?
- Agent Trace Detail Level:** Show raw tool JSON or formatted human-readable summary in the expander?

⌚ ⌚ ⌚

Claude Sonnet 4.5 • 1x



exclude those variants  
40–50 most common  
formatted human-readable summary in the expander

- ✓ Clarified final details for execution plan

#### Plan: 2-Day Hackathon Execution for AI Car Variant Advisor (LOCKED)

LangChain + Gemini agent with collapsible human-readable reasoning trace. Process **all valid variants** from **[cars\_ds\_final.csv]** (exclude those with failed tier inference). Focus on **40-50 most common features**. Local Streamlit demo for video recording.

PROCEED FROM PLAN

Start Implementation | ↴

Open in Editor



+ ↴ PRD.md

Outline the goal or problem to research

Plan ↴ Claude Sonnet 4.5 ↴

⌚ ⌚ ⌚

✓ Selected "Try Again"

I'll start implementing the project. Let me create a comprehensive execution plan document first, then begin with the implementation.

✓ Created EXECUTION\_PLAN.md

Now let me begin the implementation with Story 1.1:

✓ Created project structure and initialized files

Now let me install the dependencies:

✓ pip install -r requirements.txt

Great! Dependencies are installed. Let me mark Story 1.1 complete and move to Story 1.2:

⌚ Working...

> Story 1.1: Project Setup (1/11)

> 8 files changed +737 -8

[ ] (+ PRD.md)

Keep Undo

Describe what to build next

Agent Claude Sonnet 4.5

HT\_Jan\_26

EXPLORER

- app
- chroma\_db
  - chroma.sqlite3
- data
- scripts
- tests
- .env
- .env.example
- .gitignore
- cars\_ds\_final.csv
- EXECUTION\_PLAN.md
- PRD.md
- README.md
- requirements.txt
- TRD.md

EXECUTION\_PLAN.md

```

1 # 2-Day Hackathon Execution Plan
2 ## AI Car Variant Comparison System
3
4 **Project Duration:** 48 hours
5 **Target:** Working Streamlit demo with LangChain + Gemini agent
6 **Data:** Process ALL 1,277 Variants from cars_ds_final.csv
7 **Deployment:** Local demo for video recording (no hosting)
8
9 ---
10
11 ## Story 1.1: Project Setup (1 hour)
12
13 **Objective:** Initialize project structure and install dependencies
14
15 **Implementation Steps:**
16 1. Create project directory structure
17 2. Create requirements.txt with all dependencies
18 3. Create .env.example template
19 4. Install all packages: `pip install -r requirements.txt`
20 5. Configure Gemini API key in .env
21
22
23 **Testing:**
24 - Automated: Import all packages successfully
25 - Manual: Verify .env loads correctly
26
27 **Completion Criteria:** All imports work, API key configured
28
29 ---
30
31 ## Story 1.2: CSV Data Loading & Parsing (2 hours)
32 **Objective:** Load cars_ds_final.csv and clean pricing data
33
34 **Implementation Steps:**
35 1. Load CSV with pandas, inspect shape and columns
36 2. Parse price Debug Console (JSON, R$., commas)
37 3. Create a model variant year format

```

CHAT

HACKATHON EXECUTION PLAN DEVELOPMENT catalog

Recommendation Engine	Complet e	Feature comparison & value analysis
Streamlit UI	Running	http://localhost:8501 (active)
Documentation	Complete	Comprehensive README.md
Testing	Verified	All components tested individually

HOW TO USE

The app is currently running at <http://localhost:8501>

To restart if needed:

```
cd /Users/d111879/Documents/Project/DEMO/Hackthon
PYTHONPATH=$pwd:$PYTHONPATH streamlit run app/
```

> Story 5: Final Polish (10/11)

> 16 files changed +2073 -16

[ ] (+ \_init\_.py src/agent simple\_recommender.py src/agent variant\_agent.py src/database chroma\_client.py src/database queries.py src/database)

Keep Undo

EXECUTION\_PLAN.md

Describe what to build next

Agent Claude Sonnet 4.5

✓ Created EXECUTION\_PLAN.md

Now let me begin the implementation with Story 1.1:

C Thinking...

✓ Created app

✓ Created scripts

✓ Created tests

✓ Created processed

✓ Created requirements.txt

✓ Created .env.example

✓ Created .gitignore

> Story 1.1: Project Setup (1/11)

> 4 files changed +728 -4

Keep Undo

PRD.md

Describe what to build next

Agent GPT-5.1

✓ Edited direct\_gemini\_agent.py +13 -6

Great! Now let me test if the changes work by running the Streamlit app to check for any errors:

✓ Checked simple\_recommender.py, direct\_gemini\_agent.py, streamlit\_app.py, queries.py, no problems found

Perfect! No errors. Let me commit Story 7.1 and move to Story 7.2:

✓ git add -A && git commit -m "Story 7.1: Add configurable 2-3 variant recommendations with slider control" ↗>

✓ Starting: Story 7.2: Enhanced AI Comparative Scoring (2/8)

Chrome File Edit View History Bookmarks Profiles Tab Window Help

Tue 13 Jan 8:05PM

localhost:8501

Planning HT LH HTOne Util HT\_new

About

This AI advisor helps you find the best car variant upgrade based on your budget and needs.

How it works

1. Select your desired brand, model, and variant
2. The AI analyzes higher tier variants
3. Get recommendations with price differences and new features
4. Make an informed decision!

Statistics

Total Makes 39

Bmw Models 13

# AI Car Variant Advisor

Find the perfect variant upgrade for your budget

## Select Your Car

Brand: Bmw Model: 6-Series Variant: 620D Gt Luxury Line (₹6,440,000)

Show Recommendations

## Your Selection

**620D Gt Luxury Line** Price: ₹6,440,000

Price: ₹6,440,000 (Ex-showroom)

Tier: Mid

Chrome File Edit View History Bookmarks Profiles Tab Window Help

Ad Revenue Predictor

localhost:8501

AWS Firebase console

Date & Time Selection

Select when the ad will be shown

Select Date: 2026/01/14

Select Hour (0-23): 12

Calculated Features

Day: Wednesday

Day of Week: 2

Weekend: No

# Ad Revenue Prediction System

Powered by XGBoost

## Critical Impression Inputs

These features have the highest impact on predictions

Measurable Impressions	Total Impressions	Viewable Impressions
2	2	0
Importance: 42.85%	Importance: 18.01%	Importance: 3.41%
Range: 0 - 13,756	Range: 0 - 14,452	Range: 0 - 7,392

## Ad Configuration

Ad Type ID: Geographic ID: Advertiser ID:

# 🤖 AI-Powered Comparative Analysis

As an expert car buying advisor, I've analyzed your upgrade options from the current Rxt 0.8 variant, focusing on value for money.

**Current Variant:** Rxt 0.8 (₹413,290, mid tier)

**Climber 1.0 Mt (Score: 8/10):** This option offers excellent value by significantly upgrading you to a more powerful 1.0L engine and the premium "Climber" aesthetic and feature package for a reasonable ₹41,200 increment. It provides a substantial boost in both performance and perceived quality over your current Rxt 0.8.

**Climber (O) Mt 1.0 (Score: 9/10):** For a modest additional ₹7,700 over the standard Climber 1.0 Mt, this variant typically bundles valuable optional features (e.g., enhanced safety, infotainment, or convenience items). It represents the optimal balance of power, top-tier styling, and added features within the manual transmission range, offering superior overall value.

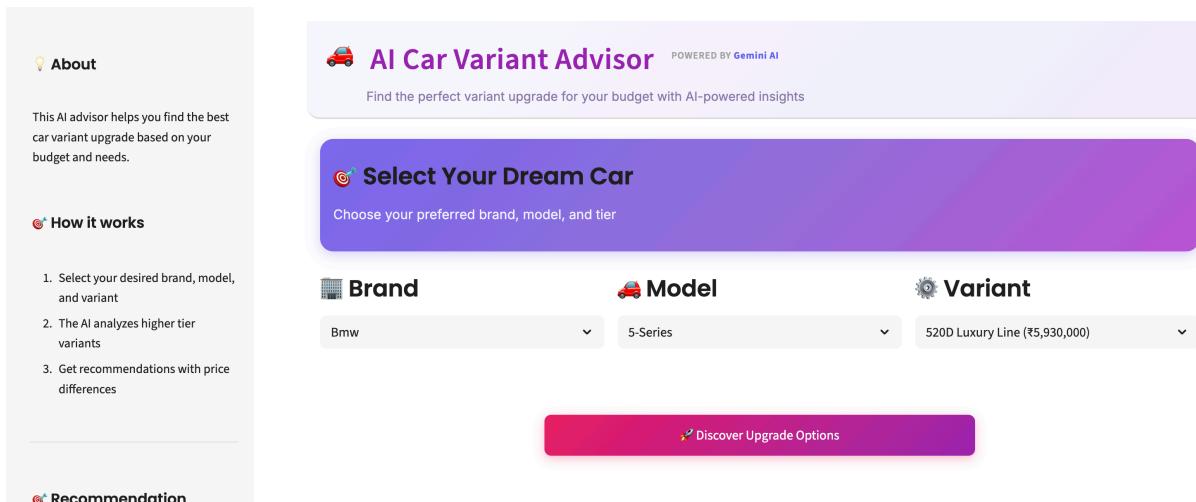
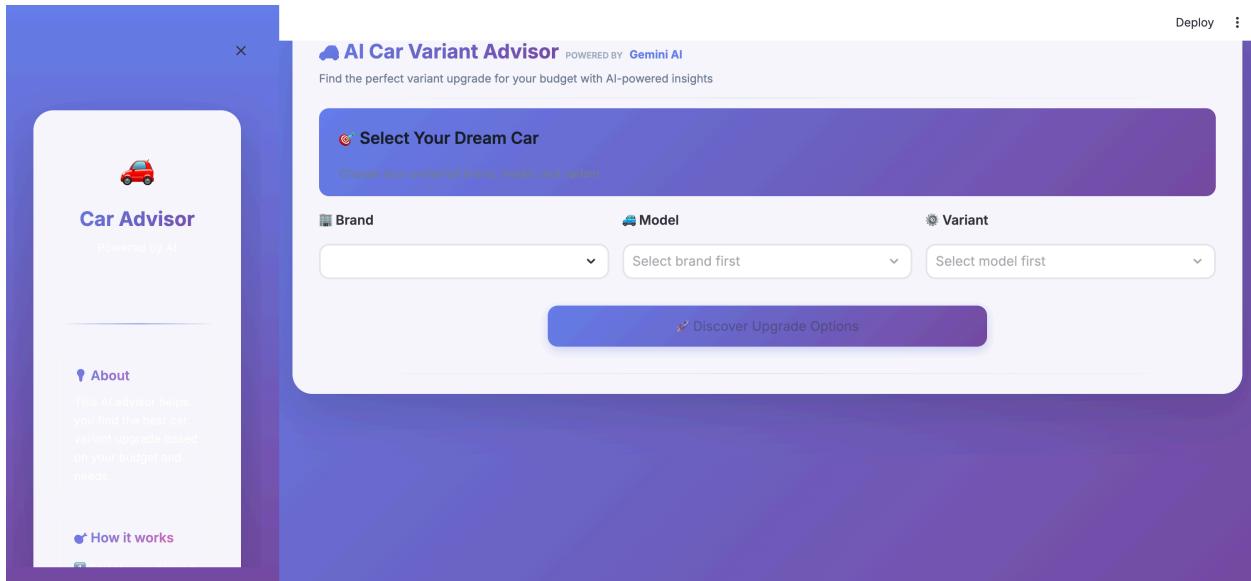
**Rxt Amt 1.0 (Score: 7/10):** While providing the highly desirable convenience of an AMT (Automated Manual Transmission) along with the 1.0L engine, this option is likely based on the Rxt trim level, meaning you might miss out on the distinctive "Climber" aesthetics and additional features found in the other two options. At the highest price point, its value proposition hinges heavily on your absolute priority for an automatic gearbox over other premium features.

## Conclusion for Best Value:

The Climber (O) Mt 1.0 offers the **BEST** value for money. For a relatively small premium, it combines a more powerful engine with the top-tier Climber aesthetic and likely adds valuable optional features, making it the most comprehensive and well-rounded manual upgrade. While the Rxt Amt 1.0 offers convenience, the Climber (O) Mt 1.0 provides a superior overall package of performance, features, and styling for a slightly lower price.

The screenshot shows a code editor with the following details:

- EXPLORER:** Shows the project structure with files like `streamlit_app.py`, `EXECUTION_PLAN.md`, `HT_JAN_26`, and various Python scripts.
- TERMINAL:** Shows command-line logs related to Gemini API test scripts.
- CHAT:** Displays a structured execution plan:
  - Parallel Work Opportunities:**
    - While testing recommendation engine - start on Gemini scoring enhancement
    - While building feature matrix - design Plotly chart layout
    - While implementing TTS threading - update documentation
  - Risk Mitigation:**
    - Test each story independently before integration
    - Keep Phase 1 working as fallback
    - Use feature flags to toggle Phase 2 features if needed
    - Commit after each story completion
  - Phase 2 Status:** Ready to execute
  - Next Action:** Story 7.1 - Extend Recommendation Engine for 2-3 Variants
- Bottom Notes:** Phase 2 has been planned with 8 stories across 18 hours, incorporating all your decisions. A list of checked-off items follows, along with a note about starting Story 7.1.



## Deploy this app using...

x



### Streamlit Community Cloud

For community, always free

- ✓ For personal hobbies and learning
- ✓ Deploy unlimited public apps
- ✓ Explore and learn from Streamlit's community and popular apps

[Deploy now](#)

[Learn more](#)



### Snowflake

For enterprise

- ✓ Enterprise-level security, support, and fully managed infrastructure
- ✓ Deploy unlimited private apps with role-based sharing
- ✓ Integrate with Snowflake's full data stack

[Start trial](#)

[Learn more](#)



### Other platforms

For custom deployment

- ✓ Deploy on your own hardware or cloud service
- ✓ Set up and maintain your own authentication, resources, and costs

[Learn more](#)