GoRules Custom â Project Brief

Problem Statement
Hair-care brands need a lightweight â rules-as-a-serviceâ backend to evaluate treatment eligibility an

Solution Description
- **Rules API**: CRUD endpoints let admins manage JSON Logic rules in MongoDB, then evaluate st
- **Questionnaire Engine**: ‘/api/questions‘ endpoints store questions, options, and conditional routing
- **Platform Plumbing**: ‘server.js‘ boots Express, applies CORS, JSON parsing, logging, health chec

Simple DB Design
The application uses two MongoDB collections with concise schemas:

rules

Field	Type	Purpose
‘name‘	String	Human-readable rule label.
‘description‘	String	Optional editor notes.
‘rule‘	Object	JSON Logic payload evaluated at runtime.
‘createdAt‘ / ‘updatedAt‘	Date	Auditing timestamps.

Indexes: ‘{ name: 1 }‘, ‘{ createdAt: -1 }‘ to accelerate lookups and recent activity views.

questions

Field	Type	Purpose
‘questionText‘	String	Prompt shown to users.
‘questionType‘	String enum	single-choice, multiple-choice, text, number.
‘options‘	Array\<Option\>	Choices with ‘id‘, ‘text‘, ‘value‘, optional ‘nextQuestionId‘, ‘tags‘.
‘category‘	String enum	hair-type, scalp-condition, hair-problem, hair-goal, lifestyle, other.
‘order‘	Number	Primary display ordering.
‘isActive‘ / ‘isFirstQuestion‘	Boolean	Publish flag and questionnaire entry point.
‘conditionalLogic‘	Object	JSON Logic evaluated against accumulated answers.
‘metadata‘	Object	Extra UI hints (e.g., helper text).

Indexes: ‘{ category: 1, order: 1 }‘, ‘{ isActive: 1, isFirstQuestion: 1 }‘, ‘{ tags: 1 }‘ to keep traversal pred

This pared-down schema keeps the learning curve low while covering the projectâ s core use cases: r