Analysis Document – Quote Processing Module

Author: Shouryan Sood

Date: 13 June 2025

# 1. Introduction

This document presents the analysis for the QuoteProcessingService module in the Insurance Application. It includes the code flow, database interactions, and relevant table structures.

# 2. Code Flow

1. QuoteController

- Entry point via /api/quote/process  
 - Invokes QuoteProcessingService

2. QuoteProcessingService

- Validates input DTO  
 - Calls QuoteCalculator to compute values  
 - Saves the quote using QuoteRepository

3. QuoteCalculator

- Applies business logic (base premium + rider cost)  
 - Returns computed values

4. QuoteRepository

- JPA Repository saving to quotes table

@PostMapping("/process")  
public ResponseEntity<QuoteResponse> processQuote(@RequestBody QuoteRequest req) {  
 return ResponseEntity.ok(quoteProcessingService.process(req));  
}

# 3. Technologies & Frameworks

- Java: 17  
- Spring Boot: 3.1.0  
- Database: PostgreSQL 13  
- JPA: Spring Data JPA

# 4. Database Design

Table: quotes

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| id | UUID | Primary Key |
| customer\_id | VARCHAR | FK to customers table |
| quote\_amount | DECIMAL | Total premium |
| created\_date | TIMESTAMP | Creation time |
| status | VARCHAR | Quote status (ACTIVE/DRAFT) |

Table: customers

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| id | UUID | Primary Key |
| name | VARCHAR | Customer Name |
| dob | DATE | Date of Birth |

# 5. Query Sample

SELECT \* FROM quotes WHERE customer\_id = '1234-abcd' AND status = 'ACTIVE';

# 6. Observations / Recommendations

- Ensure customer\_id exists before saving quote.  
- Consider indexing status for faster lookups.  
- Business rules are hardcoded — move to rules engine if complex logic increases.