Software Design Document for Project

Shreyansh Karamtot
Software Engineering Intern

February 12, 2026

1 Introduction

1.1 Purpose

This document speaks about the design and implementation of a workflow-driven insurance claim processing system built using Camunda BPM. The platform orchestrates backend services to automate end-to-end claim lifecycle while ensuring strong state control, auditability and decision automation.

1.2 Overview

System enables automated processing of insurance claims, beginning from claim submission through validation, document review, decisioning and final payout or rejection.

SUBMITTED → VALIDATED → APPROVED/REJECTED

2 Workflow Overview

Camunda BPM serves as central orchestration engine, coordinating service interactions through BPMN workflows. Gateways enforce decision paths, while service tasks invoke backend APIs responsible for business logic and persistence.

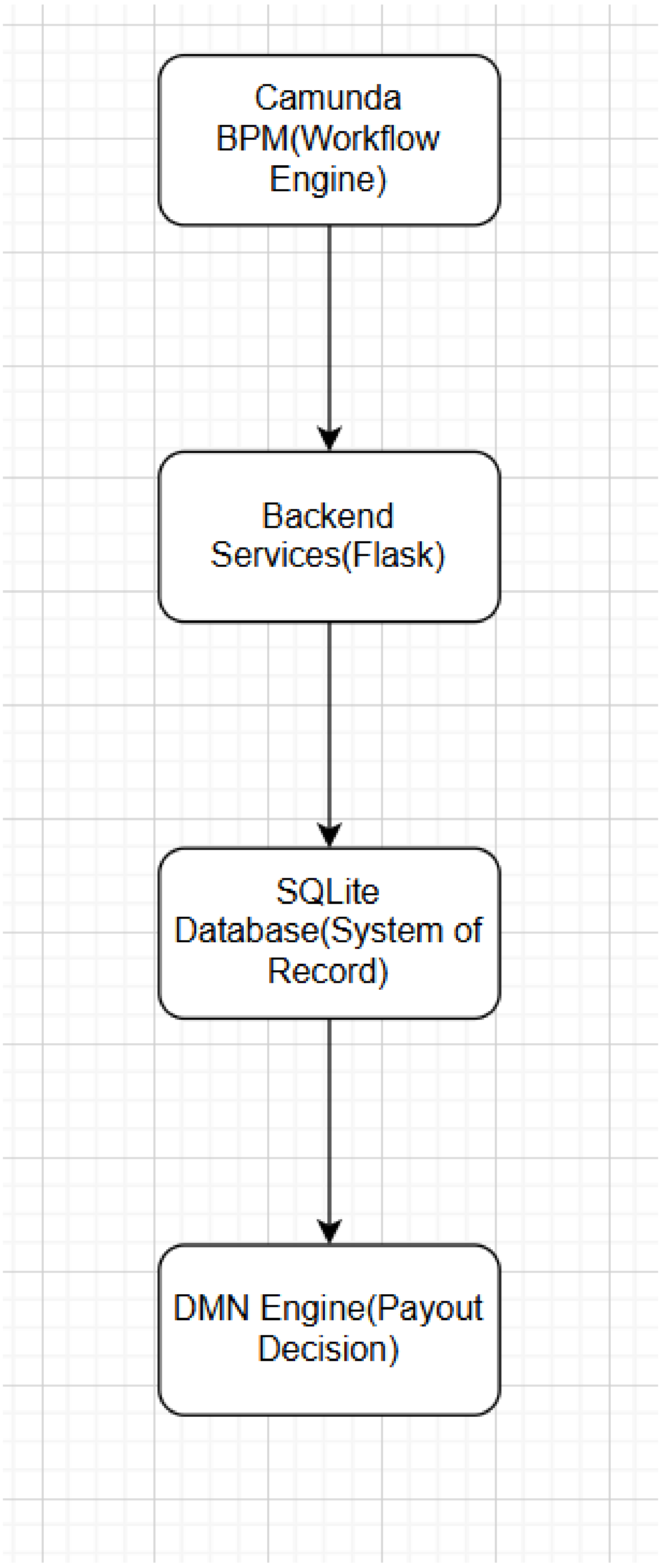
Camunda BPM(Workflow Engine)
Backend Services(Flask)
SQLite Database(System of Record)
DMN Engine(Payout Decision)

Figure 1: Workflow Overview Diagram

3 System Architecture

3.1 Architectural Design

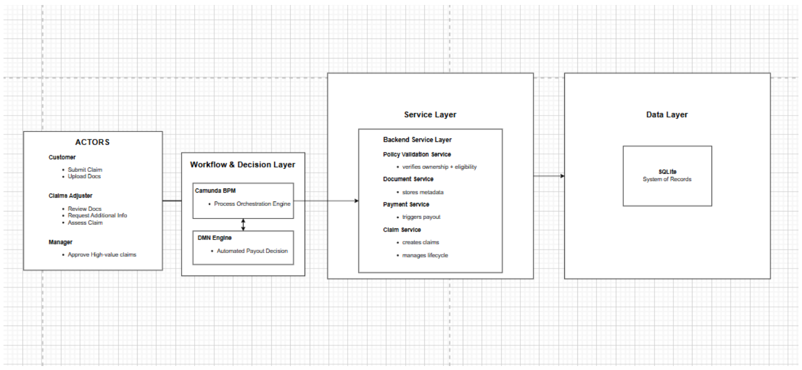

Figure 2: System Architecture Design

3.2 Decomposition Description

Provide a decomposition of the subsystems in the architectural design. Supplement with text as needed. You may choose to give a functional description or an object oriented description.

3.3 Design Principles

- Separation of orchestration and persistence.
- Backend-controlled financial state transitions.
- Single source of truth for claim data.
- Audit-first system design.
- Minimal Service payloads.
- Scalable Service architecture.

4 Key System Components

4.1 Camunda BPM

- Orchestrates the claim lifecycle.
- Controls routing via gateways.
- Maintains process variables.

4.2 DMN Engine

- Automates Payout calculation.
- Reduces System complexity.
- Ensures consistent managerial decisions.

4.3 Backend Services

- Policy Validation.
- Claim creation and persistence.
- Claim state transitions.
- Document Metadata storage.

4.4 Database (SQLite)

- Claims and claims history.
- Policies and documents.

5 State Management and Auditability

System follows state driven architecture where each claim progresses through controlled transitions. Every state change is recorded in a claim history store, providing a complete audit trail.

6 Engineering Highlights

- Implemented a state-machine like claim lifecycle.
- Prevented illegal state transitions.
- Centralized claim history tracking.
- Designed loosely coupled service interactions.

7 Future Enhancements

- Policy administration APIs.
- Role-based access control.
- Notification Integration.