Software Requirements Specification (SRS)

Project Name: AutoClaim – AI-Based Vehicle Insurance Claims Processing System

---

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for **AutoClaim**, an automated insurance claims processing system. The system utilizes a "Machine-First" approach to automate initial claim assessments, detect fraud through advanced AI forensics, and route claims for either instant approval or human review[1].

1.2 Scope

The system will consist of:

1. **User Web Application:** For policyholders to submit accident narratives, photos, and documents.
2. **AI Validation Engine:** A Python-based service performing deep forensic analysis on uploaded evidence.
3. **Decision Logic:** A rule-based engine that routes claims based on risk scores and monetary thresholds.
4. **Agent Dashboard:** A web interface for insurance officers to review flagged cases with AI-generated insights.

---

2. Overall Description

2.1 Product Perspective

AutoClaim operates as a web-based solution. The frontend handles user interaction, while the backend (FastAPI) orchestrates data flow between the User App, the Database, and the AI Analysis Modules.

2.2 User Classes and Characteristics

- **Policyholder (User):** Non-technical users submitting claims via mobile or desktop.
- **Insurance Agent (Admin):** Technical experts reviewing complex or suspicious claims flagged by the AI.

---

3. System Architecture

3.1 Technology Stack

- **Frontend:** React.js (User App & Agent Dashboard).
- **Backend API:** Python (FastAPI).
- **AI/ML Services:** Python (PyTorch/TensorFlow, OpenCV, EasyOCR, Scikit-learn).
- **Database:** PostgreSQL (Relational data, JSON metadata, Audit logs).

3.2 Architecture Diagram

---

4. Functional Requirements

Module 1: Claim Submission (Frontend)

- **FR-1.1:** The system shall allow users to input a text narrative describing the accident.
- **FR-1.2:** The system shall allow users to upload visual evidence (images of vehicle/scene) and supporting documents (bills/police reports).
- **FR-1.3:** The system must validate file formats (JPG, PNG, PDF) before submission.

Module 2: AI-Driven Validation & Forensics (Backend)

The core engine responsible for verifying authenticity and detecting fraud.

Phase A: integrity & Source Checks

- **FR-2.1 Metadata Verification:** The system shall extract EXIF data (GPS, Timestamp) to verify the photo was taken at the claimed location and time.
- **FR-2.2 Reverse Image Search:** The system shall perform a reverse search to ensure photos are original and not sourced from the internet.
- **FR-2.3 Digital Forgery Detection:** The system shall perform Error Level Analysis (ELA) or Noise Analysis to detect if the image has been digitally altered (e.g., Photoshop detection).

Phase B: Vehicle & Damage Verification

- **FR-2.4 Vehicle Match:** The system shall use object recognition to confirm the make/model in the photo matches the insured vehicle.
- **FR-2.5 License Plate Verification (ANPR):** The system shall extract the license plate number via OCR and strictly match it against the policy database.

- **FR-2.6 Pre-Existing Damage Detection:** The system shall detect signs of rust, paint fading, or accumulated dirt in the damaged area to identify non-accident wear.

Phase C: Contextual Consistency

- **FR-2.7 Cross-Verification:** The system shall compare the user's text narrative with the visual evidence to ensure consistency (e.g., user says "rear-ended," AI checks for rear-bumper damage).
- **FR-2.8 Environmental Consistency:** The system shall compare the weather/lighting in the image (e.g., sunny/shadows) against historical weather data for the claimed timestamp and GPS location.

Module 3: Decision Logic

- **FR-3.1 Threshold Evaluation:** The system shall compare the estimated claim amount against a pre-defined limit (e.g., ₹20,000).
- **FR-3.2 Scoring System:** The AI shall generate a "Confidence Score" (0-100%).
- **FR-3.3 Auto-Approval:** If the Claim Amount < Threshold **AND** Confidence Score > 90% (all checks pass) → Status = **Approved**.
- **FR-3.4 Escalation:** If Claim Amount > Threshold **OR** Confidence Score < 90% (any check fails) → Status = **Flagged for Review**.

Module 4: Agent Dashboard (Human-in-the-Loop)

- **FR-4.1 Claim Summary Report:** For flagged claims, the system shall generate a PDF report highlighting specifically *why* the claim failed (e.g., "Metadata Mismatch" or "Rust Detected").
- **FR-4.2 Manual Override:** The interface shall allow agents to view side-by-side evidence and select Approve, Reject, or Request Info.
- **FR-4.3 Audit Logging:** All AI decisions and Agent actions must be logged in the database for compliance.

---

5. Non-Functional Requirements

- **NFR-1 Performance:** The AI validation process (all 8 checks) should complete within 15 seconds of submission.
- **NFR-2 Security:** All PII (Personally Identifiable Information) and uploaded documents must be encrypted at rest in PostgreSQL.

- **NFR-3 Scalability:** The FastAPI backend must be capable of handling asynchronous requests to prevent blocking during heavy image processing.
- **NFR-4 Reliability:** The ANPR (License Plate) module must handle angled or partially obscured plates with a fail-safe to "Human Review" rather than a false rejection.

---

6. Database Design (Preliminary)

- **Table: Claims**
    - claim_id (PK)
    - user_id (FK)
    - status (Enum: Pending, Approved, Rejected, Escalated)
    - ai_confidence_score (Float)
    - flagged_reason (Text)
- **Table: Evidence**
    - evidence_id (PK)
    - claim_id (FK)
    - file_path
    - metadata_gps (JSON)
    - is_forged (Boolean)
    - rust_detected (Boolean)
    - plate_number_detected (String)