rmNoto Sans

SOFTWARE REQUIREMENTS SPECIFICATION

AUTOCLAIM

AI-Based Vehicle Insurance Claims Processing System

Version 1.0

Prepared by

[Team Member Name 1]
Team Member Name 2
Team Member Name 3
Team Member Name 4

January 18, 2026

Revision History

Name	**Date**	**Reason For Changes**	**Version**
AutoClaim Team	January 18, 2026	Initial Draft based on SRC requirements	1.0

Contents

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. This SRS ensures clarity for all stakeholders in understanding the system's goals and functionality.

1.2 Document Conventions

This document follows standard formatting conventions. Functional requirements are uniquely labeled (e.g., FR-1.1) and categorized by module. All placeholders for future decisions are marked as TBD (To Be Determined).

1.3 Intended Audience and Reading Suggestions

This Software Requirements Specification (SRS) document is intended for various stakeholders:

- **Developers:** To understand the technical specifications, API requirements, and logic flows.
- **Project Managers:** To manage the scope and track progress against requirements.
- **Insurance Stakeholders:** To understand the product's capabilities in fraud detection and automation.
- **Testers:** To validate system features against the defined requirements.

Readers are advised to begin with the Introduction and Overall Description, then proceed to the specific System Features relevant to their role.

1.4 Product Scope

AutoClaim aims to streamline the vehicle insurance claim process. The system consists of:

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

1.5 References

- FastAPI Documentation: https://fastapi.tiangolo.com/
- React Documentation: https://react.dev/
- PostgreSQL Documentation: https://www.postgresql.org/docs/
- PyTorch Documentation: https://pytorch.org/docs/

2 Overall Description

2.1 Product Perspective

AutoClaim operates as a comprehensive web-based solution. The frontend handles user interactions via a responsive web application, while the backend (built on FastAPI) orchestrates data flow between the User App, the Database, and the AI Analysis Modules. It is designed to be a self-contained system that integrates with existing insurance policy databases.

2.2 Product Functions

AutoClaim performs the following major functions:

- **Claim Submission:** Secure upload of narratives, images, and documents.
- **AI Forensics:** Metadata verification, digital forgery detection (ELA), and reverse image search.
- **Damage Assessment:** Vehicle matching, license plate verification (ANPR), and damage consistency checks.
- **Automated Decisioning:** Instant approval for low-risk, low-value claims.
- **Manual Review Interface:** A dashboard for agents to review flagged claims with detailed AI reports.

2.3 User Classes and Characteristics

- **Policyholder (User):**
  - **Frequency of Use:** Low (only during accidents).
  - **Expertise:** Non-technical.
  - **Needs:** Simple submission process, quick status updates.
- **Insurance Agent (Admin):**
  - **Frequency of Use:** High (daily).
  - **Expertise:** Technical/Insurance domain expert.
  - **Needs:** Detailed forensic data, side-by-side evidence comparison, efficient workflow.

2.4 Operating Environment

- **Client Side:** Modern web browsers (Chrome, Firefox, Safari, Edge) on Desktop and Mobile.
- **Server Side:** Python environment (FastAPI), typically hosted on Linux servers.
- **Database:** PostgreSQL relational database.

2.5 Design and Implementation Constraints

- **Performance:** AI validation must complete within 15 seconds.
- **Privacy:** PII must be encrypted at rest.
- **Connectivity:** Requires internet access for submission and API calls.
- **Image Quality:** Dependent on the quality of user-uploaded photos for accurate AI analysis.

2.6 User Documentation

The following documentation will be provided:

- User Manual for the Policyholder Web App.
- Operational Guide for the Agent Dashboard.
- API Documentation for backend integration.

2.7 Assumptions and Dependencies

- Users will have access to a device with a camera and internet connection.
- The system depends on external APIs (or libraries) for Weather data and GPS mapping.
- The insurance company provides an accessible database of valid policyholders for verification.

3 External Interface Requirements

3.1 User Interfaces

The system will provide two primary interfaces:

- **Policyholder App:** A clean, step-by-step wizard for claim submission. It will guide users to take photos from specific angles and upload documents.
- **Agent Dashboard:** A data-rich interface displaying the claim queue. Clicking a claim opens a detailed view with the "Claim Summary Report," highlighting AI flags (e.g., red boxes around rust, metadata alerts).

3.2 Hardware Interfaces

- **Mobile/Camera:** The system interfaces with user device cameras (via file upload) to capture evidence.
- **Server Infrastructure:** Runs on standard server hardware capable of supporting GPU acceleration for AI processing (PyTorch/TensorFlow).

3.3 Software Interfaces

- **FastAPI:** The backend framework handling API requests.
- **AI/ML Libraries:** OpenCV, EasyOCR, Scikit-learn, PyTorch used for image processing and forensics.
- **PostgreSQL:** The primary database for storing user data, claims, and audit logs.

3.4 Communications Interfaces

- **HTTP/HTTPS:** All client-server communication occurs over secure HTTP channels.
- **JSON:** The primary data format for API responses and requests.

4 System Features

4.1 Module 1: Claim Submission (Frontend)

4.1.1 Description and Priority

This module handles the intake of data from the policyholder. Priority: **High**.

4.1.2 Functional Requirements

- **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.

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

4.2.1 Description and Priority

The core engine responsible for verifying authenticity and detecting fraud. Priority: **High**.

4.2.2 Functional Requirements

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 digital alterations.

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 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 to identify non-accident wear.

Phase C: Contextual Consistency

- **FR-2.7 Cross-Verification:** The system shall compare the user's text narrative with visual evidence (e.g., checking for rear damage if "rear-ended" is claimed).
- **FR-2.8 Environmental Consistency:** The system shall compare image weather/lighting against historical weather data for the claimed timestamp.

4.3 Module 3: Decision Logic

4.3.1 Description and Priority

A rule-based engine routing claims based on risk scores. Priority: **High**.

4.3.2 Functional Requirements

- **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 Claim Amount < Threshold AND Confidence Score > 90%, set Status = Approved.
- **FR-3.4 Escalation:** If Claim Amount > Threshold OR Confidence Score < 90%, set Status = Flagged for Review.

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

4.4.1 Description and Priority

Interface for manual review of escalated claims. Priority: **Medium**.

4.4.2 Functional Requirements

- **FR-4.1 Claim Summary Report:** For flagged claims, the system shall generate a PDF report highlighting specifically why the claim failed.
- **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 Other Non-Functional Requirements

5.1 Performance Requirements

NFR-1: The AI validation process (all 8 checks) should complete within 15 seconds of submission to ensure a smooth user experience.

5.2 Security Requirements

NFR-2: All PII (Personally Identifiable Information) and uploaded documents must be encrypted at rest in the PostgreSQL database.

5.3 Scalability

NFR-3: The FastAPI backend must be capable of handling asynchronous requests to prevent blocking during heavy image processing loads.

5.4 Reliability

NFR-4: The ANPR (License Plate) module must handle angled or partially obscured plates with a fail-safe to ”Human Review” rather than false rejection.

6 Other Requirements

6.1 Database Design (Preliminary)

The system utilizes a relational database (PostgreSQL) to store claim and evidence data.

- **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` (String)
  - `metadata_gps` (JSON)
  - `is_forged` (Boolean)
  - `rust_detected` (Boolean)
  - `plate_number_detected` (String)

A Glossary

- **ANPR:** Automatic Number Plate Recognition.
- **ELA:** Error Level Analysis (used for detecting digital image manipulation).
- **EXIF:** Exchangeable Image File Format (metadata stored in images).
- **OCR:** Optical Character Recognition.
- **PII:** Personally Identifiable Information.
- **SRS:** Software Requirements Specification.

B Analysis Models

- **Architecture Diagram:** (Refer to System Architecture in design documents).
- **Flowchart:** (Refer to the Decision Logic flowchart).

C To Be Determined List

- **Hosting Provider:** Selection of cloud provider (AWS/GCP/Azure) for GPU instances.
- **Exact Thresholds:** Finalization of the monetary threshold (currently estimated at 20,000).
- **Localization:** Support for languages other than English in the Narrative Input.