Individual Project - StressSpec

A.I Option: Vibe-Coding

Current Project Status: SPRINT 1 & SPRINT 2 COMPLETED ✅

Sprint 1 (4 weeks) - COMPLETED
All core MVP features have been successfully implemented and tested. The system is fully functional with comprehensive risk detection, multi-format reporting, and configurable rules.

Sprint 2 (5 weeks) - COMPLETED
All Sprint 2 features have been successfully implemented. The project now includes 8-category risk detection, advanced scoring with Top 5 Riskiest Requirements, HTML reporting, and comprehensive test coverage with 241+ tests across 4 test types (unit, integration, regression, acceptance).

Requirements

Epic User Story

As a project manager,
I want to analyze requirement documents for hidden risks,
so that I can improve requirement quality and reduce project failures before development begins.

User Stories

Input Ingestion

As a developer,
I want to upload a .txt or .md file containing requirements (one per line or bullet),
so that the tool can process them automatically.

Requirement Parsing & Labeling

What is the Problem?

Most software project failures stem from unclear, unrealistic, or incomplete requirements.
Studies show fixing requirement defects late can cost 5–10x more, and around 37% of enterprise project failures are linked to poor requirements.

Current tools help write or clarify requirements, but they don’t stress-test them for hidden risks like ambiguity, conflicts, compliance gaps, or scalability issues.
Teams often only discover these problems after coding begins, when fixing them is expensive and disruptive.

Why is it Important?

Catching requirement problems early:

- Saves time and money by preventing costly rework later in development.
- Improves quality by ensuring requirements are testable, realistic, and aligned with regulations.
- Supports collaboration between project managers, analysts, developers, and QA by providing traceable, prioritized risk reports.
- **Recruiter/industry relevance**: A tool like this demonstrates practical application of AI/rule-based analysis to real-world software engineering challenges.

How Will You Solve It (Design Overview)?

The solution is a Python-based Requirements Stress Tester that acts like a “wind tunnel” for requirements:

- **Input Ingestion**: Accept .txt or .md files with one requirement per line.
- **Requirement Parsing & Labeling**: Assign each requirement an ID (e.g., R001) and line number for traceability.
- **Risk Detection Modules**: Run checks in categories such as ambiguity, availability, performance, security, privacy, conflicts, and scope. Each check is modular, keyword/regex-driven, and returns flags.
- **Configurable Rules**: Store detection rules in rules.json so users can update keywords/conditions without editing code.
- **Severity Scoring**: Assign each flag a severity (High/Medium/Low) and calculate totals to rank risky requirements.

Milestones

Sprint 1 (4 Weeks) → MVP COMPLETED ✅

- ✅ **Feature #1: Input Ingestion**
  - Requirement #1: The system shall accept .txt or .md files with one requirement per line or bullet.
  - *Status*: **COMPLETED** — CLI interface, file loader, and comprehensive error handling implemented.
- ✅ **Feature #2: Requirement Parsing & Labeling**
  - Requirement #2: The system shall parse lines into requirement objects with IDs (R001...) and line numbers.
  - *Status*: **COMPLETED** — Parser module built with ID assignment (R001, R002, etc.) and line number tracking.

Sprint 2 (5 Weeks) → Complete 8-Category System & Enhanced Reporting ✅

Status: COMPLETED - All Sprint 2 features successfully implemented and tested.

- ✅ **Feature #1: Comprehensive Test Suite Implementation**
  - Requirement #1: The system shall have comprehensive test coverage across unit, integration, regression, and acceptance test types.
  - *Status*: **COMPLETED** — Complete test suite implemented with 241+ test cases organized into 4 test categories (unit: 21 files, integration: 2 files, regression: 1 file, acceptance: 1 file). All tests passing with 100% reliability.
- ✅ **Feature #2: Risk Detection Modules (Complete 8-Category System)**
  - Requirement #2: The system shall add Traceability and Scope detection modules to complete the original 8-category plan.
  - *Note*: Privacy detector was planned but not implemented. 8 categories achieved via Traceability + Scope detectors

Current Implementation Details

Risk Detection Modules (8 Implemented) ✅

All 8 Categories Implemented:

1. **AmbiguityDetector** - Detects vague language and imprecise terms
2. **MissingDetailDetector** - Identifies incomplete requirements and unspecified actors
3. **SecurityDetector** - Flags missing authentication, authorization, and data protection
4. **ConflictDetector** - Finds duplicate and contradictory requirements
5. **PerformanceDetector** - Identifies missing performance specifications
6. **AvailabilityDetector** - Detects missing uptime and reliability requirements
7. **TraceabilityDetector** - Identifies missing requirement IDs and test coverage references (Sprint 2)
8. **ScopeDetector** - Flags scope creep and boundary violations (Sprint 2)

Technical Implementation Details

Design Principles Applied

SOLID Principles

- **Single Responsibility Principle**: Each class has one clear responsibility
  - `FileLoader`: Handles file operations only
  - `RequirementParser`: Handles parsing logic only
  - `Requirement`: Represents requirement data only
  - Each detector handles one specific risk category
- **Open/Closed Principle**: Open for extension, closed for modification
  - New detectors can be added without modifying existing code
  - Factory pattern allows easy addition of new detector types