StressSpec - Requirements Stress Tester

A user-friendly web application for analyzing requirement documents and detecting potential risks before development begins.

StressSpec acts as a "wind tunnel" for requirements, helping project managers, business analysts, and development teams identify hidden risks in requirement documents early in the development process.

🚀 Quick Start

What You Need

- **Python 3.8 or higher** (check with `python --version`)
- **Internet connection** (for initial setup)

Getting Started (3 Simple Steps)

1. **Install Dependencies**

   ```
   pip install -r requirements.txt
   ```

2. **Start the Web Application**

   ```
   python web_utils/run_web.py
   ```

3. **Open Your Browser**

📖 User Guide

Using the Web Interface

Step 1: Upload Your Requirements File

1. Open the web interface at http://127.0.0.1:8000
2. Click "**Choose File**" or drag and drop your requirements file
3. Supported formats:
    - **.txt files** - One requirement per line
    - **.md files** - Requirements in bullet points or numbered lists

Step 2: Analyze Your Requirements

1. After uploading, click "**Analyze Requirements**"
2. The system will automatically:
    - Parse your requirements and assign unique IDs (R001, R002, etc.)

🎯 Understanding Risk Scores

How Risk Scoring Works

Each requirement receives a **Total Risk Score** based on detected issues:

- **Low** = 1 point
- **Medium** = 2 points
- **High** = 3 points
- **Critical** = 4 points
- **Blocker** = 5 points

Example:

- Requirement R001 has 2 risks: HIGH (3) + MEDIUM (2) = **Total Score: 5**
- Requirement R002 has 1 risk: CRITICAL (4) = **Total Score: 4**
- R001 would be ranked higher (more risky) than R002

📋 Supported File Formats

Text Files (.txt)

One requirement per line:

```
The system shall allow users to login with email and password
The system shall display user dashboard after successful login
The system shall support password reset functionality
```

Markdown Files (.md)

Requirements in bullet points or numbered lists:

```
- The system shall allow users to login with email and password
- The system shall display user dashboard after successful login
- The system shall support password reset functionality
```

Note: Lines starting with `#` or `//` are automatically ignored as comments.

🔧 Setup & Installation

Automated Setup (Recommended)

Run the setup script to automatically configure everything:

```
python web_utils/setup_web.py
```

This will:

- Install all required dependencies
- Create necessary directories
- Download required libraries
- Verify the setup

Manual Setup

If you prefer manual setup:

🌐 Accessing the Application

Once the server is running, you can access:

- **Main Web Interface**: http://127.0.0.1:8000
- **API Documentation**: http://127.0.0.1:8000/api/docs (for developers)
- **Health Check**: http://127.0.0.1:8000/health

💻 Command Line Interface (Advanced)

For users who prefer command-line tools or want to integrate StressSpec into automated workflows:

Basic Usage

```
# Analyze a requirements file and generate a Markdown report
python main.py --file data/sample_requirements.txt --report-format md --verbose

# Generate different report formats
python main.py --file requirements.txt --report-format csv --output report.csv
python main.py --file requirements.txt --report-format json --output report.json
python main.py --file requirements.txt --report-format html --output report.html
```

CLI Options

- `--file` - Path to your requirements file (required)
- `--report-format` - Output format: `md`, `csv`, `json`, or `html` (default: `md`)

🛠️ Troubleshooting

Common Issues

1. Port Already in Use

- The default port 8000 is already in use
- **Solution**: Change the port in `web_utils/run_web.py` or use a different port:

```
python -m uvicorn web.main:app --host 127.0.0.1 --port 8001
```

2. Dependencies Not Installed

- Missing required Python packages
- **Solution**: Reinstall dependencies:

```
pip install -r requirements.txt --force-reinstall
```

3. File Upload Errors

📊 Features

Core Capabilities

- **Automatic ID Assignment** - Each requirement gets a unique ID (R001, R002, etc.)
- **Line Number Tracking** - Maintains traceability to original file location
- **8-Category Risk Detection** - Comprehensive risk analysis across multiple dimensions
- **Risk Scoring** - Combined risk scores per requirement
- **Top 5 Riskiest Requirements** - Automatic prioritization of critical issues
- **Multi-format Reporting** - Download reports in Markdown, CSV, JSON, or HTML
- **Real-time Analysis** - Fast processing with immediate results
- **Interactive Interface** - User-friendly web UI with responsive design

Risk Detection Categories

📚 Additional Resources

- **Project Documentation**: See `docs/StressSpec_Project_Progress.md` for detailed technical documentation, project planning, and development history
- **Sample Files**: Download sample requirement files from the web interface to test the application
- **API Documentation**: Available at `/api/docs` when the server is running (for developers)

🎓 Requirements

- **Python 3.8 or higher**
- **All dependencies** (installed via `pip install -r requirements.txt`)

📝 Example Workflow

1. **Prepare Your Requirements File**
   - Create a `.txt` or `.md` file with your requirements
   - One requirement per line (or bullet point)
2. **Start the Web Application**

   ```
   python web_utils/run_web.py
   ```

3. **Upload and Analyze**
   - Open http://127.0.0.1:8000 in your browser
   - Upload your requirements file
   - Click "Analyze Requirements"
4. **Review Results**

💡 Tips for Best Results

- **Be Specific** - Clear, specific requirements result in fewer false positives
- **Use Consistent Format** - One requirement per line makes parsing more accurate
- **Review Top 5 First** - Focus on the highest-risk requirements first
- **Use HTML Reports** - Best for stakeholder presentations
- **Export to CSV** - Use for spreadsheet analysis and sorting

Need help? Check the troubleshooting section above or review the project documentation in `docs/StressSpec_Project_Progress.md`.