Product Requirements Document:
Software Version Comparator Application

Version: 1.0
Date: May 11, 2025
Project: Internal Tool Development

1. Introduction & Project Goal

1.1. Overview:
This document outlines the requirements for a "Software Version Comparator" application. The application will allow users to input a software version number and view a list of software products (from a predefined internal catalog) that have a version number strictly greater than the one entered. The application must also validate user input and provide clear feedback.

1.2. Business Need:
To provide a simple, accessible tool for quickly identifying software products within the company's catalog that meet or exceed a specified version threshold.

1.3. Purpose of this Document:
This PRD serves as the definitive specification for the application's features, functionality, and quality attributes. It is intended for stakeholders, designers, developers, and testers.

2. User Interface (UI) & User Experience (UX) Requirements

2.1. General Presentation:

- **QUAL-UI-01: Professional Appearance:** The application interface should present a clean, professional, and user-friendly design. It should not appear as unstyled, plain elements.
- **QUAL-UI-02: Clarity & Intuitiveness:** The layout and interactive elements should be intuitive, allowing users to understand its purpose and operate it with minimal instruction.
- **QUAL-UI-03: Readability:** Text, labels, and messages must be easily readable, using appropriate font choices, sizes, and color contrast.

2.2. Core Interface Elements:

- **FR-UI-01: Application Title:** A clear title (e.g., "Software Version Comparator") shall be displayed to identify the application.
- **FR-UI-02: Instructional Text:** Brief instructional text shall guide the user on how to use the application (e.g., explaining the input format or purpose).
- **FR-UI-03: Version Input Field:** A dedicated input field shall allow users to type a software version string. Placeholder text (e.g., "Enter version...") is recommended.
- **FR-UI-04: Submission Mechanism:** A clear action element (e.g., a "Find Software" button) shall allow users to submit their entered version for processing. Input submission should also be possible via keyboard "Enter" key when the input field has focus.
- **FR-UI-05: Results Display Area:** A designated area shall display the list of software products matching the criteria.
- **FR-UI-06: Message Display Area:** A designated area shall display informational messages (e.g., "No results found") or error messages (e.g., "Invalid version format"). These messages must be distinct from the results area.
- **FR-UI-07: Initial State:** Upon loading, the application should be in a clean state: input field ready, results and message areas clear or showing default/welcoming information.

2.3. Feedback & Responsiveness:

- **QUAL-UX-01: Loading Indicators:** The application must provide visual feedback (e.g., a spinner or loading message) to the user during potentially time-consuming operations, such as initial data loading or when processing a search. The primary input/action elements may be temporarily disabled during such operations.
- **QUAL-UX-02: Error Message Clarity:** Error messages must be user-friendly, clearly indicating the nature of the problem (e.g., "Invalid version format: version parts must be numbers.") without exposing technical jargon where possible.
- **QUAL-UX-03: Informational Message Clarity:** Messages for scenarios like "no results found" should be clear and state the criteria used (e.g., "No software products found with a version greater than '1.2.3'.").

3. Functional Requirements

3.1. Version Format Definition (System-wide):

- **FR-VF-01: Structure:** Software versions are strings with up to 5 parts: `[major].[minor].[patch].[build].[compilation]`.
- **FR-VF-02: Part Type:** Each part must be a non-negative integer.
- **FR-VF-03: Separator:** The period (`.`) is used exclusively as a separator between parts and does not denote a decimal.
- **FR-VF-04: Normalization for Comparison:**
    - Shorter versions are treated as if padded with trailing zero parts for comparison (e.g., "2" is equivalent to "2.0.0.0.0").
    - All versions, whether from user input or the software catalog, will be normalized to a 5-part numerical representation for internal comparison.

3.2. User Input Processing & Validation:

- **FR-INPUT-01: Trimming:** Leading and trailing whitespace shall be removed from the raw user input string.
- **FR-INPUT-02: Empty Input:** If the trimmed input is empty, it is invalid.
- **FR-INPUT-03: Trailing Dot Handling:** If the trimmed input (and non-empty) ends with a single period (`.`), this trailing period shall be removed. If this removal results in an empty string (e.g., input was `"."`), it is invalid.
- **FR-INPUT-04: Leading Dot Handling & Interpretation:**
    - If the input (after steps 01-03, and non-empty) starts with a period (`.`), this leading period indicates a "right-aligned" or "suffix" version match. The leading period itself is removed for parsing the subsequent parts.
    - If this removal results in an empty string (e.g., input was `".."` after step 03), it is invalid.
- **FR-INPUT-05: Core String Validation:** The string remaining after steps 01-04 (the "core version string") must be validated:
    - It must not be empty.
    - It must only contain digits (0-9) and periods (`.`).
    - It must not contain more than four periods (implying a maximum of five version parts).
    - When split by periods, each resulting part must not be empty (e.g., `"1..2"` is invalid) and must parse to a non-negative integer.

- **FR-INPUT-06: Invalid Input Action:** If any validation (FR-INPUT-02 to FR-INPUT-05) fails:
    - An appropriate error message shall be displayed.
    - Any previously displayed software results shall be cleared.
    - No filtering shall proceed.

3.3. Version Parsing (for Valid User Inputs):

- **FR-PARSE-01: Numerical Conversion:** For a valid core version string, its parts are converted to integers.
- **FR-PARSE-02: 5-Part Array Construction:** The parsed integer parts form a 5-element numerical array:
    - **Standard Input (No Leading Dot):** Parts `[p1, ..., pk]` become `[p1, ..., pk, 0, ..., 0]`.
    - **Leading Dot Input:** Parts `[p1, ..., pk]` from the core string (after the leading dot was removed) become `[0, ..., 0, p1, ..., pk]` (right-aligned padding with leading zeros). If `k=5`, all 5 parts of the array are filled by `[p1, ..., p5]`.

3.4. Software Catalog & Data Handling:

- **FR-DATA-01: Data Source:** The application will retrieve a list of software products from a predefined internal source. Each product has at least a `Name` (string) and a `Version` (string, adhering to FR-VF-01).
- **FR-DATA-02: Data Retrieval:** The full list of software products should be retrieved efficiently. For user sessions, this data should ideally be fetched once and cached client-side to improve performance of subsequent searches.
- **FR-DATA-03: Catalog Version Parsing:** Each `Version` string from the software catalog will be parsed into the standard 5-part numerical array (as per FR-VF-04, assuming standard left-aligned parsing).
- **FR-DATA-04: Catalog Data Integrity:** If any `Version` string in the catalog cannot be parsed correctly, an informational message or log may be generated, but the application should continue to function with the valid entries. Affected products should be excluded from comparison.

3.5. Filtering Logic:

- **FR-FILTER-01: Comparison:** A catalog software's version (`S`) is "greater than" the user's input version (`U`) if, comparing their 5-part numerical arrays part-by-part from left to right, the first differing part in `S` is numerically larger than the corresponding part in `U`.
- **FR-FILTER-02: Result Set:** The application shall identify and collect all software products from the catalog whose version is strictly greater than the user's valid, parsed input version.

3.6. Results Display:

- **FR-DISPLAY-01: Positive Results:** If matching software products are found:
    - The `Name` and original `Version` string of each product shall be displayed.
    - Any previous error/informational messages (not related to catalog data integrity) shall be cleared.
- **FR-DISPLAY-02: No Results:** If no matching software products are found:
    - A message indicating "no results found" for the specified user input version shall be displayed.
    - Any previous error messages shall be cleared.
- **FR-DISPLAY-03: Search Context:** It is desirable to remind the user of the version criteria they searched for, especially when displaying "no results" or in a footer area.

4. Quality Requirements (QUAL)

- **QUAL-PERF-01: Responsiveness:** User interface interactions (typing, button clicks) should feel responsive. Search/filter operations on the cached data should be near-instantaneous. Initial data load should be as fast as possible, with clear loading indication.
- **QUAL-USAB-01: Accessibility:** The application should follow basic web accessibility guidelines (e.g., keyboard navigability, sufficient color contrast, ARIA attributes for inputs if applicable).
- **QUAL-MAINT-01: Code Quality:** Code should be well-organized, maintainable, and include comments where necessary for clarity. Logic related to UI presentation should be separated from core business/functional logic where feasible.
- **QUAL-TEST-01: Testability:** Core functional logic (especially version parsing and comparison) should be designed in a way that is easily unit-testable, independent of the UI framework.
- **QUAL-SEC-01: Data Handling:** As this is an internal tool with predefined data, complex security measures are not required for this phase. However, input sanitization (as defined in

functional requirements) is a basic security measure against malformed data. No user-specific data is stored.

- **QUAL-SCALE-01: Data Volume Consideration:** While the initial catalog may be small, the design of filtering and comparison logic should be mindful of potential future growth to thousands of software entries, avoiding unnecessary performance bottlenecks with larger datasets.

5. Out of Scope

- User authentication or accounts.
- Persistent storage of user searches or preferences.
- Editing or managing the software catalog via this application.
- Advanced search criteria (e.g., "less than," "equal to," wildcard search beyond leading dot).
- Support for different versioning schemes or formats.