Exercise: Requirements Engineering – Quality Comparison
This exercise provides two System Requirements Document (SRD) examples. Students are asked to identify and analyze the **strengths** and **weaknesses** of each, focusing on clarity, organization, completeness, and adherence to standards in requirements engineering.

The properties for comparison include:
- Natural language
- Structured natural language
- Design description languages
- Graphical notations
- Mathematical specifications
- Clarity and precision
- Requirement confusion (functional vs. non-functional)
- Requirement amalgamation (several requirements in one statement)

Scenario 1 – Audio Streaming Application

This System Requirements Document (SRD) describes the functional and non-functional requirements for the 'StreamNow' application, a mobile and web platform for audio streaming. This version follows best practices of requirements engineering, including structured natural language, clear numbering, and the use of UML diagrams.

1. Introduction
The purpose of StreamNow is to allow users to stream, download, and organize digital audio content. The system will target both Android and iOS devices, as well as a web version accessible via modern browsers. StreamNow aims to deliver a seamless, low-latency streaming experience while maintaining scalability and security.

2. System Overview
The system architecture follows a client-server model. Clients interact with the backend through a RESTful API. The backend handles user authentication, content storage, and recommendation generation. The main modules include:
- **User Management Module**: Registration, authentication, and profile handling.
- **Streaming Module**: Audio playback and adaptive bitrate control.
- **Recommendation Module**: Personalized playlist suggestions based on listening history.
- **Administration Module**: Management of content and analytics dashboards.

3. Functional Requirements
FR-01: The system shall allow users to create an account using email or social media authentication.

FR-02: The system shall allow users to search for audio content by title, artist, or genre.

FR-03: The system shall stream selected audio content with a maximum delay of 2 seconds before playback starts.

FR-04: The system shall allow users to download selected audio files for offline playback.

FR-05: The system shall automatically generate daily playlists based on user preferences.

FR-06: The system shall allow users to report inappropriate content.

FR-07: The system shall log all playback activities for analytics purposes.

4. Non-Functional Requirements
NFR-01: The system shall support up to 100,000 concurrent sessions.

NFR-02: The home screen shall load within 2 seconds under standard network conditions.

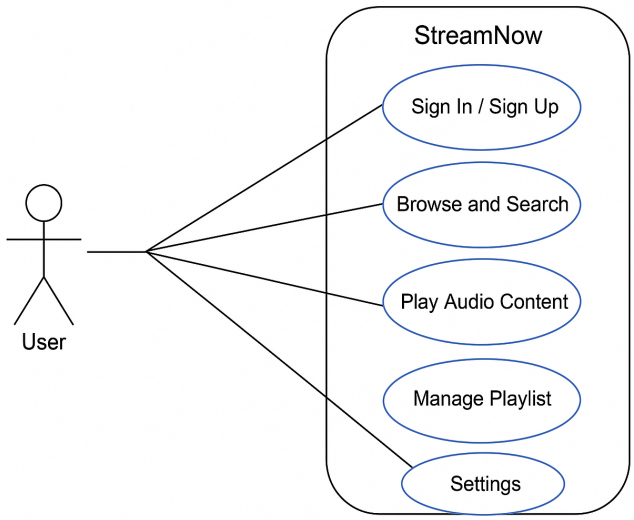
NFR-03: Audio playback shall maintain an average latency below 150 milliseconds.

NFR-04: The system shall comply with GDPR standards for user data privacy.

NFR-05: The mobile app shall have an uptime of 99.8% per month.

5. Structured Requirement Example
Requirement ID: FR-03
Title: Stream Audio Content
Description: The system shall stream selected audio content in real time.
Rationale: Real-time streaming is essential to user experience.
Dependencies: Requires internet connection and valid user authentication.
Acceptance Criteria: Playback begins within 2 seconds after user action.

6. Graphical Notations
Figure 1 shows a UML Use Case Diagram depicting interactions between the User and the StreamNow system.



7. Mathematical Specification (Optional Example)
Let S = {Playing, Paused, Stopped} represent the possible playback states.
Transition(Playing → Paused) ⇔ User triggers 'Pause' command.
Transition(Paused → Playing) ⇔ User triggers 'Play' command.
These transitions ensure deterministic behavior of the playback system.

8. Conclusion
This specification demonstrates clarity, modular organization, and precision. Each requirement is atomic, testable, and traceable, aligning with industry standards.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Data-driven Systems Engineering (ML Operations)
440MI and 305SM

Scenario 2 – Urban Ticketing Application

This SRD for the 'CityRide' ticketing app shows poor practices in requirements engineering. It lacks structure, uses vague language, and mixes functional and non-functional aspects. Students must identify all weaknesses and propose how they could be improved.

Introduction
This is about an app that helps people use buses. It will be for tickets, bus schedules, and everything related to traveling. It should be nice to use and fast and work well all the time.

Requirements
The app should allow users to buy tickets and check bus times, and it should be easy to use. It should be very fast, and it must work on all devices. It will also have good graphics and maybe some social media sharing later. The app has to support different languages, and users should be able to contact customer support when they want. It needs to be reliable and secure, and also have notifications. The app must be pretty and modern. Also, it should let drivers see user data and maybe add new schedules if needed.

Graphical Models
We might include some diagrams later on if time allows, probably something to show how the buses and users connect.

Other Stuff
The app should never crash and should be perfect. It should also have AI to suggest best routes and predict when buses arrive. It has to be easy to use and beautiful. We will add more details once the prototype is ready.