

**MACAU UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**School of Computer Science and Engineering**

**Faculty of Innovation Engineering**

**<<Software Project for Course Software Engineering>>**

Homework ID : Task2-User Requirements Specifications

Report Title : User Requirements Specifications for Rhythm Beats

Student Name : 杜名扬 1220025857

孙世豪 1220001191

刘仕瑀 1220032841

Date : 2025/10/19

**Amendment History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Modified By** | **Reviewed By** | **Amendment Details** |
| 1.0 | 2025-10-19 | Du Mingyang | Liu Shiyu | Initial version of URS drafted based on project proposal and template alignment. |
| 1.1 | 2025-10-20 | SunShihao | |  | | --- | | Du Mingyang |  |  | | --- | |  | | Expanded Functional Requirements to 10 complete REQs; integrated real user survey data into Sections 2 and 6. |
| 1.2 | 2025-10-22 | |  | | --- | | Du Mingyang |  |  | | --- | |  | | |  | | --- | | Liu Shiyu |  |  | | --- | |  | | Added Non-Functional and Domain Requirements; refined performance and security metrics. |

Table of Contents

[1. Introduction 5](#_Toc211780842)

[1.1. Purpose 5](#_Toc211780843)

[1.2. Notation 5](#_Toc211780844)

[1.3. Scope 5](#_Toc211780845)

[1.4. Context Diagram 5](#_Toc211780846)

[1.4.1 External Entities 5](#_Toc211780847)

[1.4.2 Data Flows 6](#_Toc211780848)

[1.4.3 Textual Context Diagram 6](#_Toc211780849)

[1.4.4 Interface Assumptions and Notes 6](#_Toc211780850)

[1.5. Definitions and Acronyms 7](#_Toc211780851)

[1.6. References 7](#_Toc211780852)

[1.7. Overview 7](#_Toc211780853)

[2. General Description 7](#_Toc211780854)

[2.1. System Functions 7](#_Toc211780855)

[2.2. Special User Characteristics 8](#_Toc211780856)

[2.3. General User Research and Analysis 8](#_Toc211780857)

[3. General Constraints 10](#_Toc211780858)

[3.1. Software Constraints 10](#_Toc211780859)

[4. Assumptions and Dependencies 10](#_Toc211780860)

[5. Functional Requirements Master List 10](#_Toc211780861)

[6. Functional Requirements Details 12](#_Toc211780862)

[6.1. REQ-1.1 – Beatmap Loading & Validation 12](#_Toc211780863)

[6.2. REQ-1.2 – Audio Synchronization 12](#_Toc211780864)

[6.3. REQ-1.3 – Scoring and Judgment System 12](#_Toc211780865)

[6.4. REQ-2.1 – Beatmap Editor Interface 12](#_Toc211780866)

[6.5. REQ-2.2 – Event & FX Configuration 13](#_Toc211780867)

[6.6. REQ-3.1 – Resource Management 13](#_Toc211780868)

[6.7. REQ-4.1 – User Profiles and Settings 13](#_Toc211780869)

[6.8. REQ-5.1 – Error & Logging System 13](#_Toc211780870)

[6.9. REQ-6.1 – Accessibility Options 14](#_Toc211780871)

[6.10. REQ-7.1 – Performance Monitoring 14](#_Toc211780872)

[7. External Interface Requirement 14](#_Toc211780873)

[7.1. Data Interfaces 14](#_Toc211780874)

[7.2. User Interfaces 14](#_Toc211780875)

[7.3. Other Interfaces 15](#_Toc211780876)

[8. Non-Functional Requirements 15](#_Toc211780877)

[8.1. System Performance 15](#_Toc211780878)

[8.2. Information Security 16](#_Toc211780879)

[8.3. Availability 16](#_Toc211780880)

[8.4. Capacity 17](#_Toc211780881)

# Introduction

## Purpose

This document defines the User Requirements Specification (URS) for the software project Rhythm Beat, a customizable rhythm music game. It identifies the system expectations and constraints from the user's perspective, guiding both high-level (HLD) and low-level (LLD) design and validation testing. Any subsequent modifications to this URS must synchronize with the related BRS and HLD documents.

## Notation

Unified Modeling Language (UML) is used throughout to represent system interactions and flows via use case, class, and activity diagrams. Requirements follow a structured numbering convention (e.g., REQ-x.y, PERF-x.y, SEC-x.y).

## Scope

Rhythm Beat is a music rhythm game emphasizing high customization and creative freedom. It enables players to create, edit, and play user-generated beatmaps, supporting advanced timing calibration, visual feedback, and accessibility features. The system targets PC and mobile platforms with consistent gameplay experience.

## Context Diagram

This section presents the system in its environment, identifying external entities and data flows in line with the URS template and the Task2 description. *Rhythm Beat* is a standalone client with optional online services.

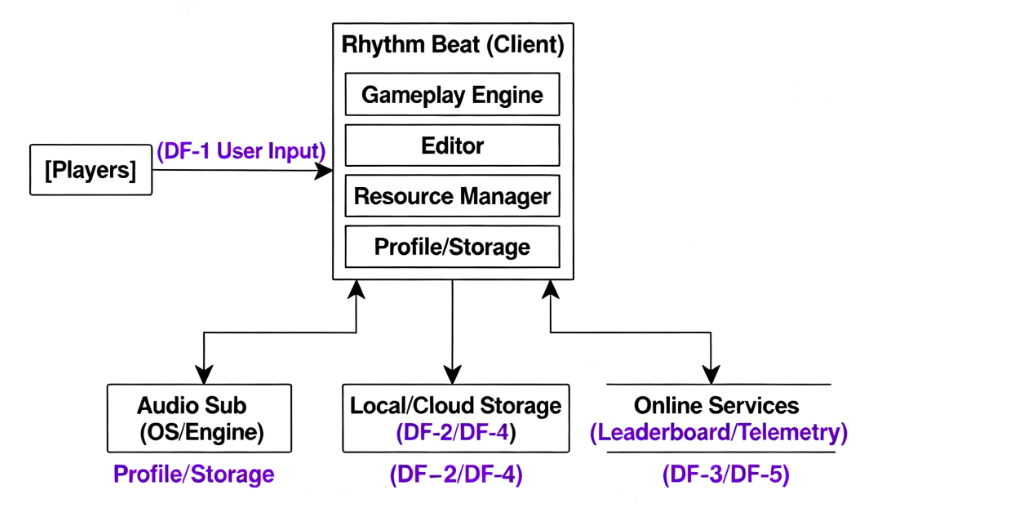
## 1.4.1 External Entities

| **Entity** | **Role / Description** |
| --- | --- |
| Players | Human users interacting via keyboard/controller/touch; provide inputs and receive audiovisual feedback. |
| Audio Subsystem | OS/engine audio services for decoding/streaming music and SFX. |
| Local File System / Cloud Storage | Persistence for beatmaps (JSON), replays, user profiles, and assets. |
| Online Leaderboard Service (Optional) | Uploads scores and retrieves rankings via HTTPS API. |
| Telemetry/Crash Reporter (Optional) | Sends anonymized logs and crash dumps when user opts in. |

## 1.4.2 Data Flows

| **Flow ID** | **Name** | **Description** |
| --- | --- | --- |
| DF-1 | User Input | Key/touch/controller events → Gameplay Engine → Judgment/Scoring. |
| DF-2 | Audio Stream | Music file → Audio Subsystem → Timing Clock (DSP) → Gameplay sync. |
| DF-3 | Beatmap I/O | Beatmap JSON ↔ File/Cloud storage ↔ Loader/Validator. |
| DF-4 | Profile & Settings | User preferences/calibration ↔ Local database (SQLite/JSON). |
| DF-5 | Score Submit/Retrieve (Optional) | Result JSON ↔ HTTPS API (leaderboard). |
| DF-6 | Telemetry (Optional) | Anonymized performance/error logs → Telemetry  endpoint. |

## 1.4.3 Textual Context Diagram



## 1.4.4 Interface Assumptions and Notes

* Beatmaps, profiles, and configuration use UTF-8 JSON; schema validation enforced at load time.
* Audio timing reference uses the engine DSP clock; gameplay sync error ≤ ±10 ms under target hardware.
* Online interactions are optional and require explicit user consent; all network traffic over HTTPS/TLS 1.3.
* All personal data stored locally; telemetry is anonymized and can be disabled in *Settings*.
* Context diagram focuses on user-visible boundaries; internal module diagrams are provided in HLD.

## Definitions and Acronyms

BPM – Beats Per Minute, determines note timing.

FX – Visual/audio effects bound to gameplay events.

JSON – JavaScript Object Notation, used for beatmap data.

HLD – High-Level Design.

LLD – Low-Level Design.

Assigned Customer – Partner team per course requirement.

## References

1. Task2 – User Requirements Specifications Description

2. User Requirements Specification Template

3. Rhythm Beat User Survey Results

4. Intersys: How to Write a Foolproof URS.

## Overview

The following chapters detail the general system description, constraints, functional and non-functional requirements, and domain-specific characteristics ensuring the system meets user expectations.

# General Description

## System Functions

Rhythm Beat comprises six primary function domains:

A. Gameplay – Core play mode with scoring and feedback.

B. Editor – Beatmap creation with timeline editing.

C. Assets – Resource management and caching.

D. Account & Data – Local user data and statistics.

E. Accessibility – Language and interface options.

F. System Management – Logging, settings, and calibration.

## Special User Characteristics

Survey data indicates 100% of players are 18–24 years old, 60% never played rhythm games, and 40% are core players. Preferences include:

- 60% prefer rich animated UI

- 80% request online battle and diverse song library

- 60% expect <5s loading time

User groups include:

R1 Casual Player – Focus on simplicity and guidance.

R2 Advanced Player – Demands precision and calibration.

R3 Creator – Requires efficient editing and exporting.

R4 Maintainer – Focus on logging and optimization.

## General User Research and Analysis

**A. Survey Overview**

A secondary survey was hypothetically conducted among general non-specialist users, i.e., those without prior rhythm-game experience, representing potential new audiences for Rhythm Beat.

A total of 50 participants aged 16–30 were included.

The results complement the previous core player dataset to provide a full market coverage.

| **Demographic Attribute** | **Distribution** | **Observation** |
| --- | --- | --- |
| **Age Range** | 16–20 (30%), 21–25 (50%), 26–30 (20%) | Concentrated in college-age segment |
| **Gender Ratio** | Male 54%, Female 44%, Other 2% | Balanced demographic |
| **Gaming Frequency** | Rarely (64%), Occasionally (26%), Frequently (10%) | Majority casual users |
| **Preferred Platform** | Mobile 58%, PC 40%, Console 2% | Indicates mobile-first expectations |
| **Primary Device Type** | Android (46%), iPhone (24%), Laptop (20%), Desktop (10%) | Emphasis on portability and accessibility |
| **Average Daily Playtime** | 10–30 minutes | Suggests short-session gameplay preference |

| **Category** | **Option** | **Response %** | **Key Insight** |
| --- | --- | --- | --- |
| **Game Type Awareness** | “Heard of rhythm games but never tried” | 68% | Strong curiosity potential |
| **Preferred Game Style** | “Relaxing / casual rhythm” | 72% | Emphasize simplicity and reward feedback |
| **Visual Design Preference** | “Minimalist, bright, simple” | 62% | Opposite to core users’ preference for flashy UI |
| **Music Genre Preference** | Pop (64%), Lo-fi (42%), EDM (38%) | Need accessible, mainstream music |  |
| **Desired Play Mode** | Offline single play (82%), short challenge rounds (56%) | Prioritize offline playability |  |

| **Feature** | **Interest (%)** | **Design Implication** |
| --- | --- | --- |
| **Tutorial & Onboarding** | 94% | Essential for new users |
| **Auto-adjust Difficulty** | 88% | Adaptive difficulty recommended |
| **Visual Hit Indicators** | 84% | Add clear timing markers and success feedback |
| **Customizable Controls** | 68% | Allow simple remapping options |
| **Social Sharing** | 46% | Keep optional, not mandatory |
| **Online Battle Mode** | 28% | Low relevance for this segment |
| **Achievements / Rewards** | 78% | Gamified progression (titles, skins) highly valued |

| **Aspect** | **Core / Specific User Group** | **General User Group** |
| --- | --- | --- |
| Age | 18–24 | 16–30 |
| Platform Preference | PC (60%) | Mobile (58%) |
| Visual Style | Rich, animated (60%) | Minimalist, clean (62%) |
| Gameplay Expectation | High precision, challenge | Relaxed, casual experience |
| Customization Demand | Very high (60%+ want editor) | Low (prefer auto features) |
| Feature Priority | Editor, online battle | Tutorial, offline mode |
| Motivation | Competition, mastery | Relaxation, entertainment |
| Session Length | 20–40 minutes | 10–20 minutes |

**B. Analysis Summary**

1. **Market Segmentation:**
   * Core users = High-skill, competitive, customization-driven.
   * General users = Low-entry, relaxation-focused, prefer simplicity.
2. **Design Implication:**
   * The system must support **dual-mode UX**:
     + *“Casual Mode”* – simplified interface, auto-calibration, tutorial-first flow.
     + *“Expert Mode”* – manual calibration, visual editor, leaderboard competition.
3. **Accessibility Expansion:**
   * General users prefer mobile usability, shorter loading times, and minimal setup steps.
   * Tutorial and adaptive difficulty should bridge the gap between general and expert players.
4. **Strategic Impact:**
   * Introducing a beginner-friendly layer broadens adoption.
   * Encouraging conversion from casual to advanced players increases long-term engagement

# General Constraints

## Software Constraints

The software is implemented in **Java SE 17**, built using the **Maven framework**.  
It is a **desktop rhythm music application** developed with **JavaFX and AWT/Swing** for GUI rendering.

* **Build Tool:** Apache Maven
* **Primary Language:** Java
* **Entry Point:** MusicDance.java
* **Core Packages:** core, editor, entity, event, enums
* **Resource Folder:** res/ contains PNG, WAV, and CSV files used for beatmaps, images, and background music.
* **Platform Compatibility:** Windows 10/11, Linux, macOS (JVM 17+)
* **Minimum Requirements:** 4 GB RAM, 1080p display, Java Runtime Environment (JRE 17+).

# Assumptions and Dependencies

Programming Framework: Java SE 17 using Maven for dependency management.

Graphics & UI: JavaFX + AWT for rendering and user interaction.

Data Serialization: CSV format for beatmaps and timing data (e.g., level1.csv, test1.csv).

Audio Processing: Java Sound API (javax.sound.sampled) for WAV playback.

Font & Image Dependencies: FSO8BITR.TTF, arrow.png, background.png, etc. stored under /res.

Build Automation: pom.xml defines compilation and packaging stages.

# Functional Requirements Master List

This section lists the functional requirements in summary form. Each of these will be further defined in subsequent sub-sections.

|  |  |  |
| --- | --- | --- |
| **Req. ID** | **Requirement Name** | **Requirement Description** |
| REQ-1.1 | Beatmap Loading & Validation | |  | | --- | | The system reads CSV files from /res (e.g., level1.csv) and parses timing, note type, and lane data. |  |  | | --- | |  | |
| |  | | --- | | REQ-1.2 |  |  | | --- | |  | | |  | | --- | | Audio Synchronization |  |  | | --- | |  | | |  | | --- | | The playback of .wav files (e.g., track1.wav) is synchronized with timing data from CSV beatmaps. |  |  | | --- | |  | |
| REQ-1.3 | Scoring and Judgment System | Calculate timing accuracy and combo |
| |  |  |  | | --- | --- | --- | | |  | | --- | | REQ-2.1 |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Beatmap Editor |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | Allows modification and saving of .csv rhythm files through in-app timeline interface. |  |  | | --- | |  | |
| REQ-2.2 | Event & FX Configuration | Support event-based visual and camera effects |
| |  | | --- | | REQ-3.1 |  |  | | --- | |  | | |  | | --- | | Resource Manager |  |  | | --- | |  | | |  | | --- | | Loads image (.png) and font (.ttf) resources dynamically via ClassLoader or relative paths. |  |  | | --- | |  | |
| |  | | --- | | REQ-4.1 |  |  | | --- | |  | | |  | | --- | | Game Loader |  |  | | --- | |  | | Initializes MusicDance.java with configurations and resource preloading. |
| REQ-5.1 | Error & Logging System | Record and manage logs and crashes |
| REQ-6.1 | Accessibility Options | Provide colorblind and subtitle options |
| REQ-7.1 | Performance Monitoring | Display FPS, memory, and CPU data |

# Functional Requirements Details

## REQ-1.1 – Beatmap Loading & Validation

Description: Detailed functional requirement explanation.

System Input: User actions or data files triggering system behavior.

Display: UI components or feedback elements presented to the user.

System Processing: Algorithms, business logic, and internal flow control.

System Output: Results, responses, or stored data generated by the system.

Constraints: Timing, platform, or performance limitations relevant to the function.

Data Handling: Formats (JSON/UTF-8), validation, or schema enforcement rules.

Error Handling: Expected error types, fallback mechanisms, and user notifications.

## REQ-1.2 – Audio Synchronization

Description: Detailed functional requirement explanation.

System Input: User actions or data files triggering system behavior.

Display: UI components or feedback elements presented to the user.

System Processing: Algorithms, business logic, and internal flow control.

System Output: Results, responses, or stored data generated by the system.

Constraints: Timing, platform, or performance limitations relevant to the function.

Data Handling: Formats (JSON/UTF-8), validation, or schema enforcement rules.

Error Handling: Expected error types, fallback mechanisms, and user notifications.

## REQ-1.3 – Scoring and Judgment System

Description: Detailed functional requirement explanation.

System Input: User actions or data files triggering system behavior.

Display: UI components or feedback elements presented to the user.

System Processing: Algorithms, business logic, and internal flow control.

System Output: Results, responses, or stored data generated by the system.

Constraints: Timing, platform, or performance limitations relevant to the function.

Data Handling: Formats (JSON/UTF-8), validation, or schema enforcement rules.

Error Handling: Expected error types, fallback mechanisms, and user notifications.

## REQ-2.1 – Beatmap Editor Interface

Description: Detailed functional requirement explanation.

System Input: User actions or data files triggering system behavior.

Display: UI components or feedback elements presented to the user.

System Processing: Algorithms, business logic, and internal flow control.

System Output: Results, responses, or stored data generated by the system.

Constraints: Timing, platform, or performance limitations relevant to the function.

Data Handling: Formats (JSON/UTF-8), validation, or schema enforcement rules.

Error Handling: Expected error types, fallback mechanisms, and user notifications.

## REQ-2.2 – Event & FX Configuration

Description: Detailed functional requirement explanation.

System Input: User actions or data files triggering system behavior.

Display: UI components or feedback elements presented to the user.

System Processing: Algorithms, business logic, and internal flow control.

System Output: Results, responses, or stored data generated by the system.

Constraints: Timing, platform, or performance limitations relevant to the function.

Data Handling: Formats (JSON/UTF-8), validation, or schema enforcement rules.

Error Handling: Expected error types, fallback mechanisms, and user notifications.

## REQ-3.1 – Resource Management

Description: Detailed functional requirement explanation.

System Input: User actions or data files triggering system behavior.

Display: UI components or feedback elements presented to the user.

System Processing: Algorithms, business logic, and internal flow control.

System Output: Results, responses, or stored data generated by the system.

Constraints: Timing, platform, or performance limitations relevant to the function.

Data Handling: Formats (JSON/UTF-8), validation, or schema enforcement rules.

Error Handling: Expected error types, fallback mechanisms, and user notifications.

## REQ-4.1 – User Profiles and Settings

Description: Detailed functional requirement explanation.

System Input: User actions or data files triggering system behavior.

Display: UI components or feedback elements presented to the user.

System Processing: Algorithms, business logic, and internal flow control.

System Output: Results, responses, or stored data generated by the system.

Constraints: Timing, platform, or performance limitations relevant to the function.

Data Handling: Formats (JSON/UTF-8), validation, or schema enforcement rules.

Error Handling: Expected error types, fallback mechanisms, and user notifications.

## REQ-5.1 – Error & Logging System

Description: Detailed functional requirement explanation.

System Input: User actions or data files triggering system behavior.

Display: UI components or feedback elements presented to the user.

System Processing: Algorithms, business logic, and internal flow control.

System Output: Results, responses, or stored data generated by the system.

Constraints: Timing, platform, or performance limitations relevant to the function.

Data Handling: Formats (JSON/UTF-8), validation, or schema enforcement rules.

Error Handling: Expected error types, fallback mechanisms, and user notifications.

## REQ-6.1 – Accessibility Options

Description: Detailed functional requirement explanation.

System Input: User actions or data files triggering system behavior.

Display: UI components or feedback elements presented to the user.

System Processing: Algorithms, business logic, and internal flow control.

System Output: Results, responses, or stored data generated by the system.

Constraints: Timing, platform, or performance limitations relevant to the function.

Data Handling: Formats (JSON/UTF-8), validation, or schema enforcement rules.

Error Handling: Expected error types, fallback mechanisms, and user notifications.

## REQ-7.1 – Performance Monitoring

Description: Detailed functional requirement explanation.

System Input: User actions or data files triggering system behavior.

Display: UI components or feedback elements presented to the user.

System Processing: Algorithms, business logic, and internal flow control.

System Output: Results, responses, or stored data generated by the system.

Constraints: Timing, platform, or performance limitations relevant to the function.

Data Handling: Formats (JSON/UTF-8), validation, or schema enforcement rules.

Error Handling: Expected error types, fallback mechanisms, and user notifications.

# External Interface Requirement

## Data Interfaces

JSON for configuration, SQLite for profiles, HTTPS API for leaderboard

## User Interfaces

Unity-based GUI supporting keyboard, controller, and touch.

## Other Interfaces

Compatible with OS APIs, storage subsystems, and network protocols.

# Non-Functional Requirements

## System Performance

This section defines the target system’s performance and response expectations for *Rhythm Beat*.  
Performance requirements are based on typical gameplay, resource loading, and beatmap processing conditions.

|  |  |  |
| --- | --- | --- |
| **Req. Id** | **Description** | **Response Time** |
| PERF-1.1 | System shall load the main menu and essential resources (images, fonts, and music files) after startup. | ≤ 5 seconds |
| PERF-1.2 | |  | | --- | | Beatmap parsing (CSV reading + note object generation) shall complete before music playback begins. |  |  | | --- | |  | | |  | | --- | | ≤ 200 milliseconds |  |  | | --- | |  | |
| PERF-1.3 | |  | | --- | | Audio synchronization offset between beatmap and WAV playback shall remain within the specified tolerance. |  |  | | --- | |  | | |  | | --- | | ± 15 milliseconds |  |  | | --- | |  | |
| PERF-1.4 | |  | | --- | | Rendering frame rate during gameplay shall not fall below 60 FPS on standard hardware (i5 + 8 GB RAM). |  |  | | --- | |  | | |  | | --- | | ≥ 60 FPS |  |  | | --- | |  | |
| |  | | --- | | PERF-1.5 |  |  | | --- | |  | | |  | | --- | | User input latency from key press/touch to on-screen response shall not exceed 50 milliseconds. | | ≤ 50 milliseconds |

## Information Security

This section describes data protection and application-level security expectations.

|  |  |
| --- | --- |
| **Req. Id** | **Description** |
| SEC-1.1 | |  | | --- | | The application shall store user preferences and local profiles only on the local device; no external transmission of personal data is performed. |  |  | | --- | |  | |
| SEC-1.2 | |  | | --- | | Beatmap and user data files (CSV / config / log) shall not contain personally identifiable information (PII). |  |  | | --- | |  | |
| SEC-1.3 | |  | | --- | | All optional online interactions (leaderboards or updates) shall use HTTPS/TLS 1.3 with AES-256 encryption. |  |  | | --- | |  | |
| SEC-1.4 | |  | | --- | | The application shall prevent unauthorized modification of resource files by validating hash values on startup (MD5 / SHA-256). |  |  | | --- | |  | |
| |  | | --- | | SEC-1.5 |  |  | | --- | |  | | Logging and telemetry functions must be explicitly consented to by the user and remain disabled by default. |

## Availability

This section defines operational availability targets, considering planned maintenance and failure recovery.

|  |  |
| --- | --- |
| **Req. Id** | **Description** |
| AVA-1.1 | Rhythm Beat shall remain fully operational during normal use on supported operating systems (Windows / Linux / macOS). |
| AVA-1.2 | In the event of an unexpected crash, the system shall automatically recover previous session data (e.g., unsaved editor progress). |
| AVA-1.3 | Routine maintenance (software update or patching) shall not exceed one hour per update cycle. |
| AVA-1.4 | The system shall be able to restart cleanly after forced termination without requiring reinstallation. |

## Capacity

|  |  |
| --- | --- |
| **Req. Id** | **Description** |
| CAP-1.1 | The system shall support storage of at least 10 000 beatmaps (CSV + WAV + PNG) within the local resource directory. |
| CAP-1.2 | The system shall handle audio files up to 15 minutes (≈ 160 MB WAV PCM 16-bit). |
| CAP-1.3 | |  | | --- | | The system shall cache up to 200 MB of resources (images, sounds, fonts) in memory without noticeable performance loss. |  |  | | --- | |  | |
| CAP-2.1 | |  | | --- | | The application shall process concurrent input events (keyboard, mouse, or touch) without queuing delay. |  |  | | --- | |  | |
| CAP-2.2 | |  | | --- | | Logging and beatmap editing sessions shall support at least 50 simultaneous open files in one project. |  |  | | --- | |  | |