Hazard Analysis
GenreGuru

Team 8 – Rhythm Rangers

Ansel Chen
Muhammad Jawad
Mohamad-Hassan Bahsoun
Matthew Baleanu
Ahmed Al-Hayali

Table 1: Revision History

Date	Version	Notes
2024-10-25	0.0	Revision 0
2025-04-04	1.0	Revision 1, Fixed issues outlined under peer reviews Issue #111, Issue #112, Issue #113, Issue #114, Issue #115

Contents

1 Introduction

This document is dedicated as a hazard analysis of the GenreGuru music system. The GenreGuru software is designed to aid its users with their consumption and creation of music, doing so by providing song analysis, recommendation, and music generation services. As such, we define a hazard as a potential malfunction in the system, either due to internal factors (such as the training data, defects in the software) or external factors (such as user inputs).

2 Scope and Purpose of Hazard Analysis

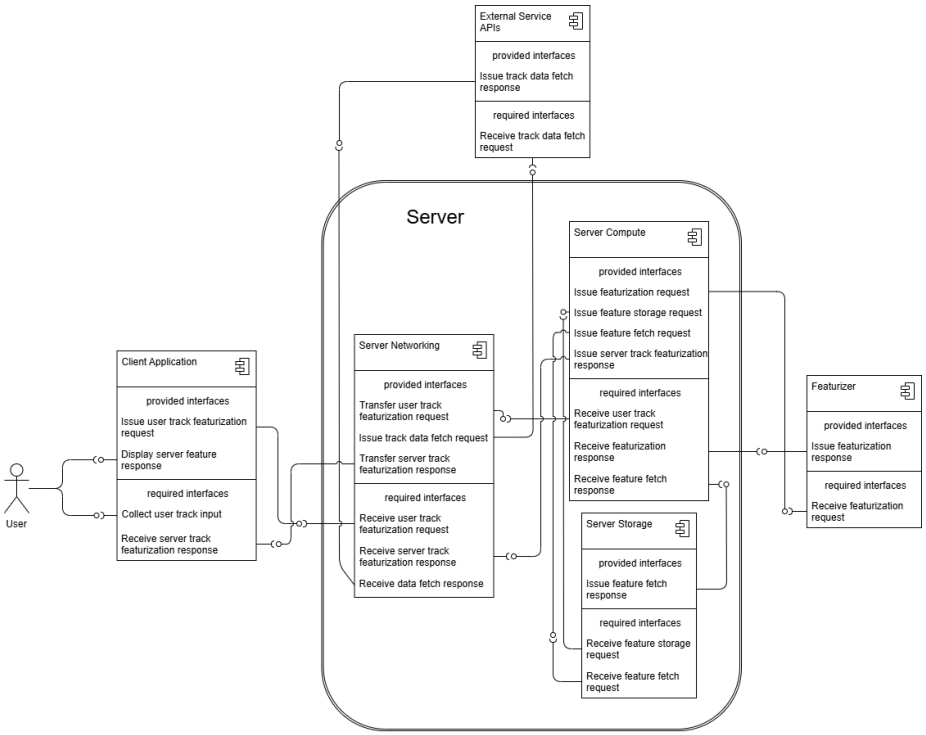
The purpose of hazard analysis for this project is to determine points and causes of failure in the system. ~~This includes their effects and considering mitigation methods.~~ This includes analyzing the effects of potential failures and devising appropriate mitigation strategies. Here are some brief types of loss we expect hazards could incur:

- Compromised Data – such as corruption of the training dataset/outputs.
- Project History – improper GitHub merging causing portions of documentation to vanish.
- User Experience Loss – generated recommendations/snippets do not satisfy user needs.

~~Note that traditional/physical hazards are not applicable in a pure software system.~~ It is important to note that traditional physical hazards (e.g., fires, explosions, mechanical failures) are not applicable in a pure software system such as this.

3 System Boundaries and Components

3.1 Diagrammatic System Characterization



3.2 System Component Descriptions

Please refer to table 3.2.

Component Name	Component Description
Client Application	User-facing component through which they can input track information and view track features as system output.
Server Networking	Abstract component that represents communication between the server and external components, i.e., the client application and external service APIs.
Server Compute	Component that represents the server processing mechanism, i.e., the component that issues the featurization request.
Server Storage	Component that represents the server storage, i.e., the database.
External Service APIs	Component that represents external service APIs, e.g., the Spotify API or Deezer API.
Featurizer	Component that handles the featurization process.

Table 2: System Component Descriptions

4 Critical Assumptions

- ~~It is assumed that the user will not attempt to exploit system vulnerabilities to gain unauthorized access to the components of the system.~~
  It is assumed that users will not intentionally misuse or attempt to break the system.

- It is assumed that users will have the physical capabilities to interact with the system. (Specifically, users are expected to have adequate visual acuity, fine motor control, and familiarity with input devices such as keyboards, mice, or touchscreens. Additionally, users must have access to hardware (e.g., a computer or smartphone) that meets the minimum display, processing, and connectivity requirements specified in the user documentation.)

- It is assumed that all libraries used in development are ~~validated and tested.~~ validated and tested, where validation involves verifying that each library meets its documented functional and non-functional requirements (including API contracts and dependency compatibility), and testing involves comprehensive unit, integration, and regression tests to ensure stability and correct behavior under expected conditions.

5 Failure Mode and Effect Analysis

Component	Failure Mode	Effects of Failure	Causes of Failure	Occurence (1-10)	Severity (1-10)	Detection (1-10)
Client Application/User Interface	Unresponsive UI elements	UI stops responding to user inputs User frustration	Poor backend implementation	7	4	10
	Incorrect display of data	Confuse users and lead to incorrect decisions	Data formatting or parsing errors from backend	2	3	4
	Transfer of User Request fails	Cannot Process User Request	Invalid User Input Server Failure	1	10	4
Featurizer	Fail to extract any or all features of a particular input (similar to the one above)	Missing features degrades the quality/performance of the featurizer	Poor implementation, or bugs in the feature extraction process	3	7	10
	Inconsistent feature outputs for the same input	Leads to unreliable song recommendations and system distrust	Improper configuration of random state	3	7	10
External Service APIs	API Request Failure	Service that requires API stops functioning	Hit Music Streaming Provider API rate limit API Format/Version Changes	1	3	1
Version Control System	Portions of Repository Are of Erroneous version	Section(s) of documentation are erroneously deleted	Improper merge conflict resolution within github	7	5	5
	Lack of commit message standards	Difficulties in tracking changes and debugging	No commit message policy or enforcement	3	4	10
Server Networking	Client Application cannot connect to server	App gets stuck on current task User Input is Lost	Server network module fails Server loses power Webapp device network module fails	6	10	5
Server Compute	Server timeout	No response from the server	Server is overloaded Server power loss Poor computation logic	4	10	6
	Network Latency/ High ping times	Slow response times and degraded user experience	Congested network or bandwidth limitations	7	5	5
Server Storage	Server storage is inaccessible	Server query failure	Storage device failure Server power loss	1	10	10
	Duplicate database entries	Compute unit returns incorrect information	Lack of duplicate protection	1	6	3
	Data corruption	Compute unit response contains errors or is empty Data loss	Server power loss Storage device failure	1	10	7
	Data retrieval delay	Slow system response or timeouts during data access	Fragemented storage or high I/O usage	10	3	8

Component	Failure Mode	RPN	Recommended Action(s)	SR	Ref.
Client Application/User Interface	Unresponsive UI elements	280	Display a message to the user to refresh the page		H1-1
	Incorrect display of data	24	Validate data before rendering and implement data consistency checks		H1-2
	Transfer of User Request fails	40	Implement Flexible Methods For User Inputs Local Storage of User Requests for recovery		H1-3
Featurizer	Fail to extract any or all features of a particular input (similar to the one above)	210	Ensure featurizer algorithm is robust and well tested		H2-1
	Inconsistent feature outputs for the same input	210	Ensure featurizer algorithm is robust and thoroughly tested with diverse datasets		H2-2
External Service APIs	API Request Failure	3	Throttle API requests if there are a large amount of them Minimize the amount of API calls the service makes Fallbacks if API request denied (request post-poned instead of cancelled) Monitor API Updates Build flexbile code, rely on classes such that change occur in one location rather than all over the place		H3-1
Version Control System	Portions of Repository Are of Erroneous version	175	Use Github Branches Proper Merge Conflict resolution		H4-1
	Lack of commit message standards	120	Enforce commit message templates		H4-2
Server Networking	Client Application cannot connect to server	300	Display message to user, attempt to reconnect to the server Regularly backup server storage		H5-1
Server Compute	Server timeout	240	Display message to user to contact system administrator Regularly backup server storage Optimize processing algorithms		H6-1
	Network Latency/ High ping times	175	Implement Quality of Service (QoS) configurations and monitor bandwidth usage		H6-2
Server Storage	Server storage is inaccessible	100	Display message to user to contact system administrator Regularly backup server storage	IR1	H7-1
	Duplicate database entries	18	Implement duplicate protection measures Merge duplicate entries	IR2	H7-2
	Data corruption	70	Regularly backup server storage	IR1	H7-3
	Data retrieval delay	240	Implement database indexing and optimize storage allocation		H7-4

6 Safety and Security Requirements

Newly-added requirements are *italicized*.

6.1 Access Requirements

ACCR1. The user shall only be able to access song features for songs they upload or ones that are licensed for their use.

6.2 Integrity Requirements

IR1. *The server shall perform a weekly backup of its database to prevent data loss in the event of a catastrophic failure.*
IR2. *The server shall implement deduplication measures to guarantee no data redundancy.*

6.3 Privacy Requirements

PR1. The system shall only expose query requests to the user that made them.

6.4 Audit Requirements

AUR1. The system shall maintain a history of user query requests.

6.5 Immunity Requirements

N/A.

7 Roadmap

As part of the capstone project timeline, we plan on implementing all safety requirements except for *PR1* and *AUR1*, which may be followed up on after May 2025.

Appendix — Reflection

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

   The majority of the team was involved in collaborating to complete all sections of the deliverable together, as opposed to appointing different sections to different people. As a result, we engaged in a deep discussion regarding project scope and system components, prompting us to restrict our scope to just the featurization component, making the music recommendation and generation components stretch goals.

2. What pain points did you experience during this deliverable, and how did you resolve them?

   Because the team was heavily engaged in collaborative completion of the document, there repeatedly were extended conversations during which we refined project scope, but also lost focus – the loss of focus resulted in the deliverable taking a longer time to complete than would have otherwise been necessary.

3. Which of your listed risks had your team thought of before this deliverable, and which did you think of while doing this deliverable? For the latter ones (ones you thought of while doing the Hazard Analysis), how did they come about?

   We fortunately were thorough in our consideration of hazards and risks while drafting prior deliverables. As a result, section 15 of our SRS document had considerable coverage except for two integrity requirements concerning data accuracy and persistence. When considering the server storage component of the system, we noted that there was no guarantee of data correctness or persistence should a server failure occur. As a result, we included IR1 and IR2 to capture these risks.

4. Other than the risk of physical harm (some projects may not have any appreciable risks of this form), list at least 2 other types of risk in software products. Why are they important to consider?

Two types of risks in software products that aren't related to physical are: the risk of compromising (user or software) data, and scope creep, where the software uncontrollably grows and developers commit to adding features beyond what was intially agreed to.

(a) Data compromise is crucial to consider because user and application data can be very sensitive, e.g., users often reuse their credentials, thus said data being compromised could have far-reaching effects beyond just the user's ability to interact with the service. Company Data can also be very valuable, if it is exposed it could be used by competitors or used to exploit other vulnerabilities within the software. Compromising unreleased songs can also result in legal action being carried out against the devlopment team.

(b) Scope creep is important to consider because it arises when the project is not properly defined during early planning stages. This often leads to extra developement work and accrual of costs for features that might not work properly or are unecessary for project success.