Hazard Analysis
Software Engineering

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

Contents

1 Introduction

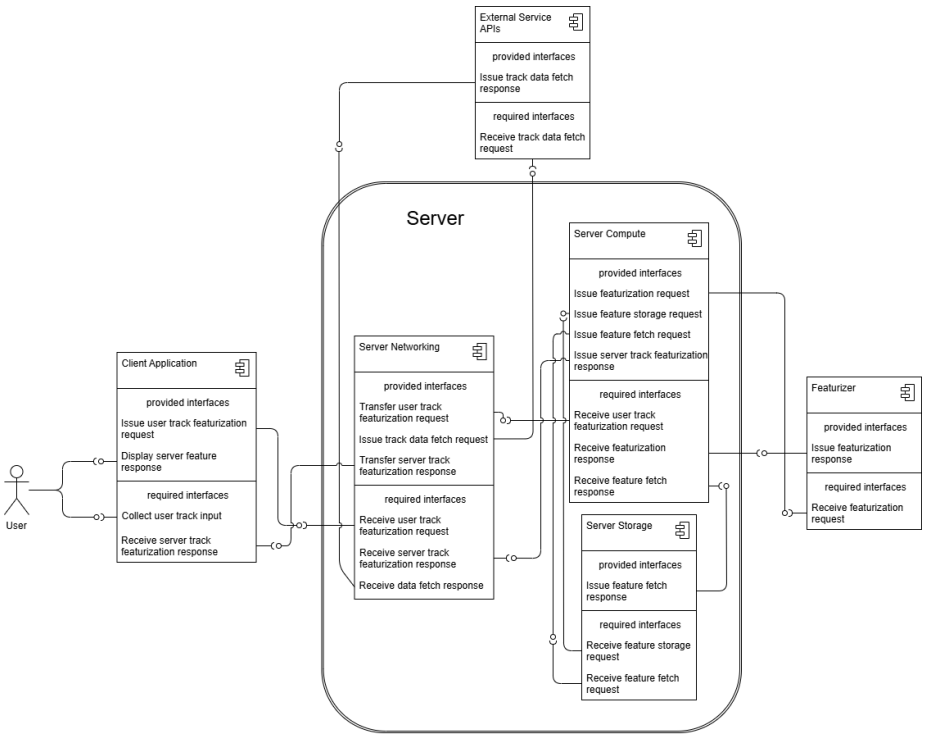
[You can include your definition of what a hazard is here. —SS]

2 Scope and Purpose of Hazard Analysis

[You should say what **loss** could be incurred because of the hazards. —SS]

3 System Boundaries and Components

3.1 Diagrammatic System Characterization



3.2 System Component Descriptions

Please refer to table 3.2.

4 Critical Assumptions

[These assumptions that are made about the software or system. You should minimize the number of assumptions that remove potential hazards. For in-

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

stance, you could assume a part will never fail, but it is generally better to include this potential failure mode. —SS]

5 Failure Mode and Effect Analysis

Component/Design Function	Failure Mode	Effects of Failure	Causes of Failure	Detection	Recommended Action(s)
User Interface App	Unresponsive UI elements	UI stops responding to user inputs User frustration	Poor backend implementation Poor frontend implementation		Display a message to the user to refresh the page
Featurizer	Poor featurization of music	Featurizer returns incorrect or incomplete data to the webapp	Poor implementation of featurizer algorithm		Ensure featurizer algorithm is robust and well tested
External Service APIs	API Request Blocked	Service that requires API stops functioning	Hit Music Streaming Provider API rate limit		- 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)
Version Control System	Merge conflict resolved improperly	Section(s) of documentation are erroneously deleted	Improper merge conflict resolution within github	- Regularly Checking Github Repo for integrity - Recompiling the PDF from latex to check for errors before deadlines - Project Supervisor(s) notice during marking	- Use Github Branchrs - Proper Merge Conflict resolution
Server Networking	Webapp can't connect to server	App gets stuck on current task	Server network module fails Server loses power Webapp device network module fails		Display message to user, attempt to reconnect to the server Regularly backup server storage
Server Compute	Server timeout	No response from the server	Server is overloaded Server power loss Poor computation logic		Display message to user to contact system administrator Regularly backup server storage Optimize processing algorithms
Server Storage	Server storage is inaccessible	Server query failure	Storage device failure Server power loss		Display message to user to contact system administrator Regularly backup server storage
	Duplicate database entries	Compute unit returns incorrect information	Lack of duplicate protection		Implement duplicate protection measures Merge duplicate entries
	Data corruption	Compute unit response contains errors or is empty Data loss	Server power loss Storage device failure		Regularly backup server storage

[The safety requirements in the table do not have to have the prefix SR. The most important thing is to show traceability to your SRS. You might trace to requirements you have already written, or you might need to add new requirements. —SS] [If no safety requirement can be devised, other mitigation strategies can be entered in the table, including strategies involving providing additional documentation, and/or test cases. —SS]

6 Safety and Security Requirements

[Newly discovered requirements. These should also be added to the SRS. (A rationale design process how and why to fake it.) —SS]

7 Roadmap

[Which safety requirements will be implemented as part of the capstone timeline? Which requirements will be implemented in the future? —SS]

Appendix — Reflection

[Not required for CAS 741 —SS]

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