Hazard Analysis
Software Engineering

Team #1, Sanskrit Ciphers
Omar El Aref
Dylan Garner
Muhammad Umar Khan
Aswin Kuganesan
Yousef Shahin

Table 1: Revision History

Date	**Developer(s)**	**Change**
Date1	Name(s)	Description of changes
Date2	Name(s)	Description of changes
...	...	...

Contents

[You are free to modify this template. —SS]

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

[Dividing the system into components will help you brainstorm the hazards. You shouldn't do a full design of the components, just get a feel for the major ones. For projects that involve hardware, the components will typically include each individual piece of hardware. If your software will have a database, or an important library, these are also potential components. —SS]

This hazard analysis treats the system as four major component groups that interact to support scholarly reconstruction of manuscript fragments. (For full details, refer to the SRS, Section S.1.)

Major Component Groups

- **Front-end**: Scholar-facing web UI for secure login, batch image uploads, an interactive canvas (arrange/rotate/zoom fragments), match discovery display, and session/annotation saving. Quick list of smaller components:
  - User Authentication Interface
  - Fragment Upload Interface
  - Interactive Canvas Module
  - Match Discovery Module
  - Progress Management Module
- **Backend**: API gateway, authentication/authorization, image preprocessing pipeline (normalization/orientation/format), database access layer, and a match-orchestration service coordinating analysis results. Concise breakdown of sub-components:
  - API Gateway
  - Authentication Service
  - Image Processing Service
  - Database Access Layer
  - Match Orchestration Service

- **Data Storage**: Fragment image store (originals and derivatives with metadata), user accounts/permissions, and project records (arrangements, match history, confidence scores, session snapshots). Brief overview of minor elements:
  - Fragment Image Database
  - User Database
  - Project Database
- **AI/ML**: Services/models for edge/damage matching, handwriting/script classification, OCR text extraction (with confidence), and content similarity/embedding comparisons. Summary of key smaller parts:
  - Edge Pattern Matching Model
  - Handwriting Style Classifier
  - Damage Pattern Recognition Model
  - Text Extraction Model
  - Content Similarity Model

Component Boundaries (Exclusions)

- **UI** presents results and collects inputs; it does not run ML models or make authoritative scholarly decisions.
- **Preprocessing** only standardizes images; it does not judge correctness of reconstructions.
- **Matching** produces scored suggestions; it does not auto-merge/link records without explicit user confirmation.
- **OCR/Transcription** is machine-generated with confidence indicators; final text requires human review before being marked confirmed.
- **Datastores** are system-of-record for this application only; they do not write back to external catalogues/corpora.

See SRS Section S.1 for the detailed component descriptions and interfaces. Note that here I have separated the backend and data storage to make it a little clearer to visualize the components but data storage will be treated as backend for the project.

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 instance, 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

[Include your FMEA table here. This is the most important part of this document. —SS] [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?