# Hazard Analysis
# Software Engineering

Team 4, EventHub
Virochaan Ravichandran Gowri
Omar Al-Asfar
Rayyan Suhail
Ibrahim Quraishi
Mohammad Mahdi Mahboob

Table 1: Revision History

| **Date** | **Developer(s)** | **Change** |
|---|---|---|
| 09-25-2025 | Rayyan Suhail | Added Component Overview and Boundaries |
| 09-29-2025 | Rayyan Suhail | FMEA Added |
| ... | ... | ... |

# 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

The proposed platform will be utilized by the McMaster Engineering Society (MES) as a centralized event management system and survey feedback collection system. The system boundary includes all the functionality required to build and render custom forms, manage event registrations and waivers, check-ins, deliver notifications, and build analytics dashboards.

## 3.1 System Boundaries

- **In Scope**: Administrator and student web and mobile interfaces, form builder and renderer, registration and waiver processing, check-in and QR verification, attendee overview dashboard, analytics and reporting, notifications, authentication and authorization, and internal data storage.
- **Not Covered**: Third-party services like university single sign-on (SSO), third-party email/SMS portals, and payment sites. These are treated as outside dependencies, with interactions regulated through integration adapters.
- **Environment**: The system is web-based, compatible with major browsers, and supports portable devices (iOS and Android). Its operation requires steady internet connectivity, although it offers minimal offline capability for the check-in process.

## 3.2 Component Overview

The system can be divided into the following major components for hazard analysis:

### 3.2.1 Admin Web Application

Provides administrators with tools to build forms, configure events, manage attendees, and view analytics.

### 3.2.2 End-User Application (Web/Mobile)

Interface for students to register for events and submit feedback surveys.

### 3.2.3 Custom Form Builder and Renderer

Schema-based system to define and display registration and feedback forms, including branching and conditional logic.

### 3.2.4 Registration Form Manager

Handles the creation and processing of registration forms, ensuring that event-specific data is collected accurately.

### 3.2.5 Attendee Overview Dashboard

Provides administrators with a consolidated view of registrations and attendee details for an event.

### 3.2.6 Analytics and Reporting

Generates data visualizations and exportable reports based on registration and survey responses, enabling event planning and improvement.

### 3.2.7 Data Storage Layer

Stores form schemas, form responses, and attendee information using structured databases and JSON-based storage.

### 3.2.8 External Integration Adapters

Interfaces with required external systems such as university single sign-on (SSO) and email services to support core functionality.

# 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 Effects Analysis (FMEA)

The following section provides a breakdown of the Failure Modes and Effects Analysis (FMEA) for the system. Each component is evaluated for potential failures, their impact on system operation, possible causes, methods of detection, and recommended actions. This analysis ensures that hazards affecting event registration, form processing, check-ins, and analytics are identified early and mitigated through targeted safety and reliability requirements.

| Design Function | Failure Modes | Effects of Failure | Causes of Failure | Detection | Recommended Action | SR | Ref |
|---|---|---|---|---|---|---|---|
| Form Builder and Renderer | Form fails to load / render | Users unable to build or complete forms → event sign-ups blocked | Frontend rendering bug, schema parsing error, server downtime | Automated UI tests, monitoring logs | Add robust schema validation, fallback UI, redundant servers | SR-1.1 | C-1 |
| | Incorrect form logic execution (branching fails) | Wrong fields shown, incorrect data captured | Logic misconfiguration, database mismatch | Unit tests on conditional logic, integration tests | Enforce schema consistency, admin preview mode | SR-1.2 | C-2 |
| Registration & Waiver Manager | Registrations not recorded | Attendees think they registered but system has no record | Database write failure, API timeout | Monitoring DB writes, user confirmation emails | Retry logic, backup DB write queue | SR-2.1 | C-3 |
| | Waivers not stored / lost | Legal liability exposure | Improper storage, file corruption, missing object storage link | Storage integrity checks | Redundant storage, integrity checksum | SR-2.2 | C-4 |

(a) Fig. 1: FMEA Table Part 1

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Check-in & QR Verification | QR code not generated | Attendees cannot check in | QR generation library bug, server failure | Test ticket generation, error alerts | Add retry system, use backup QR library | SR-3.1 | C-5 |
| | QR invalid at check-in | Attendees denied entry despite valid registration | Clock desync, mismatched record, DB corruption | Cross-check logs at check-in, time sync monitoring | Time sync service, redundant DB | SR-3.2 | C-6 |
| Attendee Overview Dashboard | Data incomplete / inaccurate | Admins make poor decisions (wrong headcount, etc.) | Analytics DB out of sync, caching issue, delayed pipeline | Scheduled DB consistency checks, anomaly detection | Real-time sync pipeline, alerting for mismatches | SR-4.1 | C-7 |
| Analytics & Reporting | Reports fail to generate | Event managers lose insights, reduced trust | Query timeout, malformed queries, overloaded analytics service | Load testing, error logging | Optimize queries, asynchronous report generation | SR-5.1 | C-8 |

(b) Fig. 2: FMEA Table Part 2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Authentication & Authorization | Unauthorized access | Data leaks, privacy breach | Role misconfiguration, SSO bypass, weak session handling | Security audits, penetration testing | Stronger role enforcement, periodic access reviews, force MFA | SR-6.1 | C-9 |
| | User unable to login | Prevents registration or event management | SSO outage, session misconfig | Monitor auth failures | Graceful fallback login (guest mode for limited ops), error messages | SR-6.2 | C-10 |
| Data Storage Layer | Data corruption | Permanent loss of registration and waiver data | Hardware failure, unhandled exceptions, misconfigured DB replication | Backups with restore tests, DB monitoring | Automatic DB failover, validated backup-restore process | SR-7.1 | C-11 |
| | Data breach | PII exposed | Weak encryption, insecure connections | Security scanning, breach detection tools | End-to-end encryption, key rotation, VPC isolation | SR-7.2 | C-12 |
| External Integration Adapters | Failure in SSO or email/SMS delivery | Users can't authenticate / receive confirmations | External API downtime, invalid API keys | Health checks, external service monitoring | Retry queue, failover provider, admin alerts | SR-8.1 | C-13 |

(c) Fig. 3: FMEA Table Part 3

## 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?