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] This document will contain the Hazard Analysis for EventHub. EventHub is a platform which provides the McMaster Engineering Society with the functionality to create and manage events, surveys and perform other administrative functions. It will allow both admins and users to interact with platform and register for events and complete surveys. For this Hazard Analysis we will be defining hazard as a potential risk situation which can result in or contribute to a mishap. The purpose of this document is to identify potential hazards and develop requirements and measure to help prevent or mitigate the risk posed by these hazards.

2 Scope and Purpose of Hazard Analysis

The purpose of this Hazard Analysis is to identify and mitigate potential hazards that could negatively impact the EventHub platforms. This includes both technical hazards relating to the design of the system as well as user interaction hazards.

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

- Users will have stable Internet Access: We are assuming that the user will remain connected to the internet as it will be required for using our application.
- Users will use the system as intended: We are assuming that users will be not attempting to use the system maliciously and focus on the relevant use cases for this platform.

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	SR1.1 SR1.3	H1-1
	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	SR1.2	H1-2
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	SR2.1 SR2.2	H2-1
	Waivers not stored / lost	Legal liability exposure	Improper storage, file corruption, missing object storage link	Storage integrity checks	Redundant storage, integrity checksum	SR2.3	H2-2

(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	SR3.1	H3-1
	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	SR3.2	H3-2
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	SR4.1 SR4.2	H4-1
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	SR5.1	H5-1

(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	SR6.1 SR6.2 SR6.4	H6-1
	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	SR6.3	H6-2
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	SR7.1	H7-1
	Data breach	PII exposed	Weak encryption, insecure connections	Security scanning, breach detection tools	End-to-end encryption, key rotation, VPC isolation	SR7.2 SR7.3	H7-2
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	SR8.1 SR8.2	H8-1

(c) Fig. 3: FMEA Table Part 3

6 Safety and Security Requirements

The following safety and security requirements were derived from the Failure Mode and Effects Analysis (FMEA). They represent newly discovered requirements that should also be included in the Software Requirements Specification (SRS).

6.1 Form Builder & Renderer

- **SR1.1**: The system shall validate all form schemas before rendering to prevent failures.
- **SR1.2**: The system shall enforce schema consistency across frontend and database logic.
- **SR1.3**: The system shall provide redundant servers or fallback UI during outages.

6.2 Registration & Waiver Manager

- **SR2.1**: The system shall implement retry logic and backup queues for database writes.
- **SR2.2**: The system shall send confirmation receipts after successful registration.
- **SR2.3**: The system shall store waiver data redundantly with integrity checks.

6.3 Check-In & QR Verification

- **SR3.1**: The system shall provide retry and backup QR code generation.
- **SR3.2**: The system shall enforce synchronized clocks across all check-in endpoints.

6.4 Attendee Overview Dashboard

- **SR4.1**: The system shall continuously sync databases to prevent inconsistencies.
- **SR4.2**: The system shall implement anomaly detection to detect corrupted or missing data.

6.5 Analytics & Reporting

- **SR5.1**: The system shall support asynchronous report generation to handle load.

6.6 Authentication & Authorization

- **SR6.1**: The system shall enforce role-based permissions and periodic access reviews.
- **SR6.2**: The system shall support multi-factor authentication for administrators.
- **SR6.3**: The system shall provide a fallback login during SSO outages.
- **SR6.4**: The system shall monitor and log failed login attempts and alert admins.

6.7 Data Storage Layer

- **SR7.1**: The system shall back up databases automatically and validate restore processes.
- **SR7.2**: The system shall encrypt data at rest and rotate encryption keys regularly.
- **SR7.3**: The system shall store sensitive data in a secure isolated environment (VPC).

6.8 External Integration Adapters

- **SR8.1**: The system shall implement retry queues and failover providers for external APIs.
- **SR8.2**: The system shall continuously monitor third-party service health and alert admins.

7 Roadmap

The hazard analysis identified several safety and security requirements. Within the capstone timeline, the focus will be on core reliability and usability features, including schema validation and fallback UI (SR1.1), reliable registration storage (SR2.1), QR code generation fallback (SR3.1), dashboard synchronization (SR4.1), and authentication fallback handling (SR6.2). More advanced requirements such as multi-factor authentication (SR6.1), scalable analytics (SR5.1), stronger data protection mechanisms (SR7.1, SR7.2), and redundant integration failover (SR8.1) are deferred to future development beyond the capstone scope. At project completion, the hazard analysis will be revisited to assess which risks have been mitigated and which requirements should be prioritized for follow-up work.

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?