Hazard Analysis
Software Engineering

Team 6, Pitch Perfect
Damien Cheung
Jad Haytaoglu
Derek Li
Temituoyo Ugborogho
Emma Wigglesworth

October 25th, 2024

Table 1: Revision History

	Developer(s)	**Change**
October 25th, 2024	Entire Team	Inital Completion
April 4th, 2025	Jad Haytaoglu	Final Revision

Contents

1 Introduction **1**

2 Scope and Purpose of Hazard Analysis **1**

3 System Boundaries and Components **1**

4 Critical Assumptions **2**

5 Failure Mode and Effect Analysis **2**

6 Safety and Security Requirements **4**
6.1 Newly Discovered Requirements 5
6.2 Rationale Design Process (Faking It) 6

7 Roadmap **7**

8 Revision 1 Traceability **10**

List of Tables

1 Revision History i
2 FMEA 3
3 Traceability of Revision 1 Changes 10

1 Introduction

A hazard analysis identifies potential risks associated with software, assessing how they might impact safety, security, and performance. This process aims to prevent accidents, reduce risks, and ensure the reliability of the platform. For the McMaster GSA Softball League Platform, the analysis will focus on potential issues like user interactions, data handling, scheduling conflicts, and access controls, aiming to create a safe and dependable user experience.

2 Scope and Purpose of Hazard Analysis

The hazard analysis for the McMaster GSA Softball League Platform encompasses the entire software system, focusing on three primary areas:

1. **User Interaction Hazards:** Risks associated with user authentication, role-based access, and data input processes.
2. **System Operation Hazards:** Risks related to scheduling algorithms, game management, and system availability.
3. **Data Management Hazards:** Risks concerning data integrity, security, and backup procedures.

The goal is to identify potential risks that could affect system safety, security, and user satisfaction. Specifically, the analysis addresses:

- **Data Loss:** Unintentional loss of user data, schedules, or game results, as detailed in the Database component analysis (Table 2).
- **Security Breaches:** Unauthorized access to sensitive information, like player details or team data, which is addressed through both User Authentication and Database security measures (Table 2).
- **Operational Disruptions:** Scheduling errors or system failures that cause conflicts or missed games, as outlined in the Game Scheduling component analysis (Table 2).
- **User Frustration:** Confusing navigation, input errors, or difficulties in tracking payment status, which are mitigated through various system components (Table 2).

3 System Boundaries and Components

- **User Authentication and Role-Based Access Control (RBAC)**: This component manages login and role assignment (commissioner, captain, player) to ensure proper access to system functionalities.

- **Game Scheduling Component**: Responsible for generating, displaying, and updating the schedule for games, including handling rescheduling requests.
- **Game Result Reporting**: Allows team captains to report game results and scores, which are then used to update standings.
- **Team and Player Management**: Manages the creation of teams, player assignments, and tracking captain roles and rosters.
- **Standings and Ranking Calculation**: Computes team standings based on game results, including win/loss records and score tracking.
- **Database**: Centralized storage for users, teams, games, schedules, and standings.
- **External Payment Tracking**: Tracks the payment status of players, though actual payment processing is handled externally (e.g., through e-transfers).
- **Communication and Announcement System**: Allows commissioners to send league-wide announcements and updates to users.

4 Critical Assumptions

- **User Authentication**: It is assumed that any authentication systems for role-based access control will function reliably and securely, preventing unauthorized access.
- **External Payment System**: It is assumed that external payment tracking will be accurate, and users will report their payment status honestly, as payment processing is outside the scope of the system.
- **Internet Connectivity**: The platform assumes that all users (commissioners, captains, players) have stable internet access to interact with the system, especially when accessing schedules and reporting game results.
- **Manual Input of Game Results**: Captains are responsible for entering accurate game scores and results. While the system assumes that captains will input scores correctly, it is possible that the score could be inputted incorrectly due to human error. To address this, the system must include a mechanism that will allow captains to review and correct submitted scores within a defined timeframe. This approach ensures data integrity and provides an extra layer of security against the impact of manual errors. Any further discrepancies will be addressed through manual oversight.

5 Failure Mode and Effect Analysis

Table 2: FMEA

Component	Failure Modes	Effects of Failure	Causes of Failure	Recommended Action	Req.	Ref.
User Authentication	Fails to authenticate/register user	Access denied to league members	Error in credentials or input, server issues, database connectivity	Allow credential recovery and auth retry	FR1, FR2	H1
Game Scheduling	• Fails to generate valid schedule • Missing games in the schedule • Teams end up with significantly different number of games compared to others • Rescheduling requests not processed or applied correctly	• Games may be scheduled at conflicting times or locations and an "infeasible schedule" error will be shown. • Some games may not be scheduled, causing dissatisfaction • Unbalanced league experience • Teams are not aware of reschedule request	• Bugs or logical errors in the scheduling algorithm • Incomplete automation in schedule generation • Scheduling algorithm fails to equally distribute games • Errors in reschedule approval or processing steps	• Implement a view to show commissioner where the schedule conflicts are and allow manual override or regeneration. • Automate schedule verification • Conduct pre-release schedule check • Implement algorithm to handle concurrent requests	FR3, FR4, FR7, FR8	H2.1-4
Game Result Reporting	Failed score reporting	League standings incorrect or not updated on time	Network issues, missing auto-save functionality	Implement auto-save and score-reporting validation checks	FR4, FR12	H3
Team/Player Management	• Failure to register player to team • Failure to register team to league	• Player participation impacted • Team participation impacted	Form validation errors, server timeouts	Ensure robust signup and error handling mechanisms, logging errors and allowing retries	FR5, FR6	H4.1-2
Database	• Data corruption • Unauthorized access to data	• Loss of team, player, or league data • Data breach, exposing sensitive information	• Server malfunction, database corruption • Weak access control measures	• Implement regular backups and database health monitoring • Implement strong access controls and database encryption	all	H5.1-2
Announcement System	Failed announcement creation or display	Players may miss important updates such as game cancellations	Server timeouts, database corruption, code bugs	Implement multiple announcement delivery methods (site announcements, email), display error message on announcement creation or display failures, allow easy creation retries	FR9, FR10	H6
System Availability	Server downtime	Users unable to access the platform	Hosting server issues, lack of redundancy	Implement server redundancy, monitor uptime, and create a disaster recovery plan	all	H8

6 Safety and Security Requirements

SAF1 User Authentication

- Requirement: Ensure authentication for higher-role users (league administrators).
- Rationale: Prevent unauthorized access to manage the league.
- Fit Criterion:
  - League administrators must authenticate using multi-factor authentication (MFA), combining a password and a time-based one-time password (TOTP).
  - The system must lock out a user after 5 consecutive failed login attempts and send a notification to the registered email address.
  - Authentication attempts must be logged, and the logs must show 100% successful tracking of both successful and failed attempts during security testing.

SEC1 Data Encryption

- Requirement: Ensure all sensitive data, such as personal emails, phone numbers, etc. is encrypted.
- Rationale: We must protect user data and maintain privacy to avoid legal issues and uphold respectable engineering ethics.
- Fit Criterion:
  - Encryption must be validated using automated security testing tools to ensure 100% compliance with the encryption standard.
  - The system must undergo a security audit every season to verify that no unencrypted sensitive data is stored or transmitted.

SEC2 Role-Based Access Control (RBAC)

- Requirement: Define access levels (players, team managers, administrators) with varying permissions.
- Rationale: These access levels ensure that users only have access to data related to their role. For example, only team managers can manage their team and invite users to their roster.
- Fit Criterion:
  - The system must restrict access to resources outside the assigned role, verified by attempting 50 test cases with an automated tool and user acceptance testing. Unauthorized access attempts must fail 100% of the time.
  - Administrators must have the ability to review and modify role permissions through a secure administrative interface.

6.1 Newly Discovered Requirements

DI1 Data Integrity and Backup

- Requirement: Schedule regular data backups and integrity checks to prevent data (game results, standings, etc.) from being lost in worst-case scenarios, like corrupted data.
- Rationale: Guarantee that the platform can recover from any potential data loss or corruption in case of system failures or web attacks.
- Fit Criterion:
  - Data backups must occur at least once daily during the season, with verification that all critical data (game results, standings, user profiles) is included in the backup.
  - Backup data must be stored in another location to ensure security and retainment of data.
  - In the event of a system failure, the system must allow data restoration from the latest backup.
  - Backup and recovery processes must be tested every season with a full system restore from backup to ensure data integrity and recovery procedures are functional.

SEC3 Session Management and Timeouts

- Requirement: Implement session timeouts when users are inactive for some time to prevent malicious unauthorized access.
- Rationale: Reduce the risk of someone accessing the account if a user leaves their session open on a shared device.
- Fit Criterion:
  - The system must automatically log out users after 10 minutes of inactivity, with a visible countdown and warning message starting 1 minute before logout.
  - After a session timeout, users must be required to re-authenticate with their credentials, ensuring that no session remains active without proper verification.

OP1 Audit Logging

- Requirement: Implement an audit to log and track actions to the system, such as login attempts, changes to the schedule, team roster updates, etc.
- Rationale: Assist in troubleshooting, and detecting suspicious activity, and encourage transparency in system actions.
- Fit Criterion:
  - The system must log all critical actions, including user logins, schedule changes, team roster updates, and permission changes, with the following details: user ID, action timestamp, and action description.

  - Logs must be stored securely in an encrypted format and accessible only to authorized users, such as administrators.
  - Audit logs must be retained for a minimum of 90 days and be automatically archived after this period, ensuring compliance with data retention policies.
  - The system must allow administrators to search and filter logs by user, action type, and date range for efficient analysis.
  - The logging system must generate alerts for suspicious activities (e.g., multiple failed login attempts, unauthorized changes to schedules) and notify administrators in real-time.
  - The audit logging mechanism must be tested regularly for coverage and completeness, with 100% of critical actions being logged during system testing.

SEC4 End-to-End Testing for Security

- Requirement: Perform testing to identify potential security vulnerabilities.
- Rationale: Ensures that the platform performs securely as expected under various scenarios, especially when exposed to potential threats.
- Fit Criterion:
  - Conduct comprehensive end-to-end security testing to identify potential security issues such as SQL injection.
  - The testing process must be automated to ensure that it is consistently applied after every major update or system change.

OP2 Automated Verification checks

- Requirement: Perform automated verficaion of schedule completeness after generation to ensure all games are correctly scheduled.
- Rationale: Improve data accuracy and prevent user dissatisfaction caused by missing games due to incomplete scheduling logic.

OP3 Retry Mechanisms

- Requirement: Implement easy retry mechanisms to prevent conflicts from race conditions or simultaneous requests.
- Rationale: Reduce errors caused by failed or incomplete request processing, ensuring seamless updates.

6.2 Rationale Design Process (Faking It)

- Mock Multi-Factor Authentication (MFA): Implement a placeholder where a user would receive a fake code (without actually sending one) to simulate a multi-factor authentication scenario.

- Simulated Encryption: Show that data is being encrypted by visualizing a password to jumbled text.
- Audit Logs: Keep simple text-based logs of actions that are done by our mock system.

7 Roadmap

1. Capstone Timeline Requirements
    - **User Authentication:** This will include the username and password authentication for the league administrators and team captains.
        - Authentication system setup (login, logout, etc.).
        - Store sensitive information (password) in the database.
        - Unique authentication methods for various roles (team members, team captains, admin, etc.)
    - **Role-Based Access Control (RBAC):** Implement basic roles to differentiate roles (team members, team captains, admin, etc.)
        - Create and assign roles with their corresponding permissions.
        - Limit features based on user roles.
        - Create test for user access restrictions.
    - **Session Management and Timeouts:** Implement basic timeout functionalities for inactive user to provide improved security.
        - Track user activity with session handling system.
        - Set time for session expiration when users are inactive.
        - Inform users before timeout occurs.
    - **Data Encryption:** Ensure secure communication by using SSL/TLS to encrypt data during communication.
        - Enable HTTPS to secure HTTP communications.
        - Encrypt sensitive data during transit to provide better security.
        - Ensure an SSL/TLS certificate is obtained for the platform.
2. Future Requirements
    - **Multi-Factor Authentication:** This will include a full implementation of an MFA system for critical user accounts.
        - Develop a system for further authentication via verification codes sent over text, email, etc.
        - Implement support for app-based authentication systems (Microsoft authenicator, duo, etc.).
        - Include the option for users to enable/disable MFA.

- **Audit logging:** Implement a logging system to track user activity and review system performance.
  - Implement system that creates a detailed log of user actions across the system.
  - Create a dashboard to review logs and user activities statistics.
  - Set up alert for errors and suspicious activity across the system.
- **Data Ecryption (At Rest):** Full encryption of all sensitive data stored in the system's database.
  - Implement an encryption system to properly secure sensitive information that is stored.
  - Ensure security of stored keys.
  - Test the encryption system to confirm security of the data that is stored (Ensure data cannot be accessed without valid decryption).
- **Data Breach Response Plan:** Develop a formal response for data breaches , including proper notification mechanisms.
  - Create a standard for the detection, response, and mitigation of data breaches.
  - Implement a proper system for informing users in the case of a data breach.
  - Ensure data protection laws and policies are followed.
- **Retry Mechanism for Scheduling Conlficts:** Implement a retry mechanism to ensure scheduling requests are processed smoothly, even in cases of failure.
  - Design logic for detecting and handling conflicts in scheduling requests.
  - Enable automatic retries for failed requests, such as due to unavailable time slots.
  - Provide clear error messages and an option for users to retry manually if the automatic retry fails.
- **Automated Verification of Scheduling** Ensure the accuracy and fairness of generated schedules automatically.
  - Develop a verification system that checks generated schedules for missing games or conflicts.
  - Automate adjustments to resolve detected issues, such as scheduling overlaps or unassigned games.
  - Test the verification system thoroughly with real-world data.

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 process of writing this deliverable allowed the team to take a deeper look at the potential impact of our software on users. We were able to generate new insights by analyzing the components from different perspectives, which helped uncover critical details for our implementation plan. The FMEA table, in particular, was a valuable tool for identifying and documenting potential risks, leading to a clearer understanding of each component's potential failures. This step has proven beneficial, as it not only clarified our next steps but also helped us identify new requirements that need to be added to the SRS.

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

   One of the main challenges was aligning our understanding of hazards with the software components and translating these into a detailed FMEA table. Initially, it was difficult to distinguish between critical issues and minor potential problems, which led to some confusion. We resolved this by holding focused discussions among team members, drawing from our combined experiences, and revisiting key sections of the SRS to ensure proper traceability.

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?

   Before working on this deliverable, the team had already considered general risks like data breaches and scheduling conflicts. However, during the hazard analysis, we identified additional risks, such as incomplete game result reporting, missing notifications, and specific user interface issues that could affect user satisfaction. These new insights came about by breaking down each component and examining potential failure modes.

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?

    - **Data Integrity Risks**: Data integrity issues can arise from incorrect data handling, database failures, or corruption. These risks are critical to consider because they can lead to incorrect information being presented to users or complete loss of essential data, undermining trust and reliability.
    - **Security Risks**: Security risks include unauthorized access, weak password policies, and vulnerabilities to hacking attempts. These are important to consider because they can lead to breaches that compromise user privacy, resulting in legal and ethical consequences, as well as damage to the platform's reputation. The platform is to be used by all members each annual season, so solid security and a good reputation is important to maintain.

8 Revision 1 Traceability

The following table tracks all significant changes made to the document based on feedback, requirements changes, and improvements:

Table 3: Traceability of Revision 1 Changes

Change ID	Description of Change	Affected Section(s)	GitHub Issue
REV1.1	Reformatted Table 2 to be more readable	5 - FMEA Table	Issue 510
REV1.2	Added scope to hazards	2 - Introduction	Issue 510
REV1.3	Added abbreviations to new requirements	6.1	Issue 510
REV1.4	Changed some unclear things with schedule feasibility and announcement failures	5 - FMEA Table	Issue 510
REV1.5	Removed an incorrect hazard from the FMEA table (was payment tracking which is out of scope) and updated requirement traceability	5 - FMEA Table	Issue 510