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
...	...	...

Contents

List of Tables

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 covers the entire software system, with a focus on user interactions, scheduling algorithms, data management, and access controls. The goal is to identify potential risks that could affect system safety, security, and user satisfaction. Specifically, the analysis aims to address:

- **Data Loss:** Unintentional loss of user data, schedules, or game results.
- **Security Breaches:** Unauthorized access to sensitive information, like player details or team data.
- **Operational Disruptions:** Scheduling errors or system failures that cause conflicts or missed games.
- **User Frustration:** Confusing navigation, input errors, or difficulties in tracking payment status.

By addressing these risks early, we aim to make the platform safer, more reliable, and user-friendly.

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 the authentication system using JSON Web Tokens (JWT) for role-based access control will function reliably and securely, preventing unauthorized access.
- **External Payment System**: It is assumed that external payment tracking (via e-transfer) 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. The system assumes that no errors will occur during this process, and that any discrepancies will be addressed through manual oversight.
- **Data Integrity**: It is assumed that the database will maintain data integrity during read/write operations, especially when updating schedules, standings, and user details.
- **System Availability**: The platform assumes that the hosting server will remain consistently available, with minimal downtime, to ensure uninterrupted access for users.

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	1, 2, 3	H1
Game Scheduling	Fails to generate valid schedule	Games may be scheduled at conflicting times or locations, affecting the season	Bugs in the scheduling/ team preferences algorithms	Implement schedule conflict resolution checks, allow reschedule requests	4	H2.1
Game Scheduling	Missing games in the schedule	Some games may not be scheduled, causing dissatisfaction	Incomplete automation in schedule generation	Automate schedule verification to ensure coverage for all teams	4	H2.2
Game Scheduling	Teams end up with either different games compared to others	Unbalanced league experience, as some teams may not get enough playtime while others may feel fatigued	Scheduling algorithm fails to equally distribute games across teams	Conduct a pre-release schedule check to ensure fair game distribution among all teams, and introduce byes for unavoidable discrepancies	4	H2.3
Game Scheduling	Rescheduling requests not processed or applied correctly	Teams are not aware of reschedule request which prevents schedule conflict avoidance which can lead to game no-shows	Errors in reschedule approval or processing steps, or race conditions caused by simultaneous reschedule requests	Implement algorithm to handly many concurrent requests, log error messages, allow easy retries	4	H2.4
Game Result Reporting	Failed score reporting	League standings incorrect or not updated on time	Network issues, missing autosave functionality	Implement auto-save and score-reporting validation checks	4, 6	H3
Team/Player Management	Failure to register player to team	Player participation impacted	Form validation errors, server timeouts	Ensure robust signup and error handling mechanisms, logging errors and allowing retries	4, 6	H4.1
Team/Player Management	Failure to register team to league	Team participation impacted	Form validation errors, server timeouts	Ensure robust signup and error handling mechanisms, logging errors and allowing retries	4, 6	H4.2
Database	Data corruption	Loss of team, player, or league data, disrupting league operations	Server malfunction, database corruption	Implement regular backups and database health monitoring	4	H5.1
Database	Unauthorized access to data	Data breach, exposing sensitive player or league information	Weak access control measures or database vulnerabilities	Implement strong access controls and potentially database encryption for confidential information (if required)	4	H5.2
Announcement System	Failed announcement delivery	Players may miss important updates such as game cancellations creating risks	Form validation errors, server timeouts	Implement multiple announcement delivery methods (site announcements, email), display error message on announcement failures, allow easy retries	6	H6
External Payment Tracking	Incorrect payment status recorded	Players marked as unpaid despite paying (or vice versa), leading to disputes	Payment record synchronization issues	Ensure payment status synchronization and require validation to be accepted	5, 6, 7	H7

6 Safety and Security Requirements

1. User Authentication
    - Requirement: Ensure authentication for higher-role users (league administrators).
    - Rationale: Prevent unauthorized access to manage the league.
2. 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.
3. 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.

6.1 Newly Discovered Requirements

4. 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.
5. 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.
6. 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.
7. 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.

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.

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.