Hazard Analysis
Software Engineering

Team #16, The Chill Guys
Hamzah Rawasia
Sarim Zia
Aidan Froggatt
Swesan Pathmanathan
Burhanuddin Kharodawala

Table 1: Revision History

Date	**Developer(s)**	**Change**
October 7th, 2025	Burhanuddin Kharodawala	Worked on Sections 1 and 2
October 7th, 2025	Hamzah Rawasia	Worked on Safety and Security Requirements
October 7th, 2025	Sarim Zia	Worked on Sections 3, 4 and 7
October 9th, 2025	Swesan Pathmanathan	Worked on Failure Methods
October 9th, 2025	Aidan Froggatt	Worked on Effect Analysis

Contents

1 Introduction

Hitchly is an application that aims to provide a safe and reliable ride share platform to McMaster students, staff and faculty members. Anything that would become an obstacle to selling this software could be considered a hazard. Large software platforms like Hitchly handle user data and interactions between different users. These areas require attention as they are the most prone to hazards.

2 Scope and Purpose of Hazard Analysis

The purpose of this document is to highlight potential hazards within components of the application and mitigation strategies to prevent those critical hazards from taking place. This FMEA stratergy is used to identify hazards related to user safety and security and system failure. This ensures that there is no risk of user safety, loss of brand reputation and loss of confidential user data.

3 System Boundaries and Components

The system can be divided into the following components:

3.1 Authentication System

This component allows users to verify themselves as McMaster students and to log in to the system in order to access the application features.

3.2 Profile System

This component allows users to input relevant personal information such as their schedule, GPS location, and role (Driver or Passenger).

3.3 Matching System

This component uses backend logic to pair users based on their schedule, location, and preferences.

3.4 Cost Calculation System

This component calculates the total trip cost for the driver and creates a per-rider estimate that factors in gas, parking, and other related expenses.

3.5 Database

This component stores user details, ride requests, matches, historical data, and any other relevant information required by the system.

3.6 Ratings and Review System

This component allows users to rate each other after completing their trips and to add reviews to their profiles.

4 Critical Assumptions

The system follows these critical assumptions about the software and system:

CA-1: The Maps API provides up-to-date navigation for all users in the surrounding GTA region.

CA-2: The application will only target McMaster students for the initial build.

CA-3: The expected user base of the application will be within the planned demographic (students and university faculty).

CA-4: API rate limits will not be exceeded under the expected user load.

5 Failure Mode and Effect Analysis

The following FMEA worksheet identifies possible hazards associated with Hitchly, their causes, effects, detection methods, and recommended mitigation actions. Traceability to the System Requirements Specification (SRS) is included where relevant.

5.1 FMEA Worksheet

Table 2: Failure Modes and Effects Analysis for Hitchly

Ref.	**Component**	**Failure Mode**	**Causes of Failure**	**Effects of Failure**	**Detection Controls**	**Recommended Action / SRS Ref.**
5.1	Authentication & Access Control	Unauthorized user gains access	Weak verification of McMaster email; credential theft	Privacy/safety risk, personal commute data exposed	Domain restriction; failed login logging	Enforce 2FA; lockouts after failed attempts (SRS S.3.2 Auth API)
5.2	Driver Verification	Unverified driver offers rides	Failure to validate license/vehicle data	Rider safety compromised	Admin checks; incomplete profile flags	Require license approval before matching (SRS S.3.1 Driver Registration)

Ref.	**Component**	**Failure Mode**	**Causes of Failure**	**Effects of Failure**	**Detection Controls**	**Recommended Action / SRS Ref.**
5.3	Matching Engine	Incorrect ride matches	Algorithm error; timetable parsing bug; corrupted location input	Riders matched incorrectly; missed/unsafe rides	Backend logs; user feedback reports	Unit/integration testing; fallback to safe “no match” state (SRS S.6 Matching Engine)
5.4	Notifications	Missed or delayed notifications	Push service outage; API failure	Users unaware of confirmations/cancellations	Health checks; delivery logs	Retry queue; allow in-app refresh (SRS S.3.1 Notifications)
5.5	Trip Data & History	Trip not recorded or lost	DB outage; sync failure	Loss of audit trail; disputes over costs or safety	DB integrity checks; monitoring	Regular backups; retries on failure; notify user (SRS S.3.2 Trip API)
5.6	Ratings & Reviews	Malicious/fake reviews	Collusion or spamming	Misleading trust scores; reduced credibility	Anomaly detection; user reporting	Rate limit submissions; only allow after confirmed trips (SRS S.3.2 Ratings API)
5.7	Location Services	Incorrect geolocation	API failure; GPS drift; spoofing	Unsafe pickups; delays; privacy risks	Compare expected vs. actual location	Allow manual override; enforce geofence on campus (SRS S.3.2 Mapping API)
5.8	Data Security	Personal data leak	Server breach; weak encryption	Severe privacy violation; reputational harm	Static analysis; penetration tests	Encrypt at rest & in transit; GDPR/PIPEDA compliance (SRS S.6.5 Code Quality)
5.9	Payment (future)	Incorrect cost calculation	Bug in cost-sharing logic	Disputes between riders and drivers	Automated tests; manual audits	Provide breakdown in summary; add dispute mechanism (SRS S.3.2 Trip Summary)
5.10	System Availability	App/API downtime	Server crash; container failure; network outage	Users stranded; loss of trust	Monitoring dashboards; downtime alerts	Deploy redundancy; user-facing error messaging (SRS S.6.3 System Testing)

5.2 Conclusion

This FMEA highlights Hitchly's most critical hazards in authentication, ride matching, driver verification, and data security. Some risks are mitigated directly through **safety requirements** in the SRS (e.g., verified drivers only), while others are addressed through **operational safeguards** such as monitoring, backups, or user documentation. By implementing these mitigations, Hitchly can meet its goals of safety, affordability, and sustainability for the McMaster community.

6 Safety and Security Requirements

SR-1: The system shall enforce multi-factor authentication using McMaster email verification.

SR-2: The system shall automatically lock an account after 5 consecutive failed login attempts within 10 minutes.

SR-3: The system shall verify each driver's status by validating credentials and license before enabling them to offer rides.

SR-4: The system shall record all failed matchmaking attempts and their causes for backend review and debugging.

SR-5: The system shall retry failed notication deliveries up to three times at 30-second intervals.

SR-6: The system shall include in-app refresh mechanism to allow users to manually retrieve notifications.

SR-7: The system shall perform periodic backups of all trip and booking data at intervals of 15 minutes or less.

SR-8: The system shall only allow users to submit ratings and reviews after a confirmed ride completion.

SR-9: The system shall automatically flag abnormal rating behavior for admin review.

SR-10: The system shall compare expected route location against live GPS data to detect abnomral activities, unless manually overrided by users.

SR-11: The system shall encrypt all personal and sensitive data both at rest and in transit.

SR-12: The system shall provide a dispute resolution interface for users to contest fare calculations.

SR-13: The system shall display user-facing error messages in case of system unavailability, including retry options.

SR-14: The system shall deploy redundancy to ensure 99.9% uptime.

SR-15: The system shall comply with GDPR and PIPEDA standards for user data protection.

7 Roadmap

The safety requirements which will be implemented as part of the capstone project include:

1. **Identity Verification** – Implemented in the initial release to ensure only verified McMaster users can register and log in. This is a foundational security feature.
2. **Ride Check-ins** – Integrated during the first testing phase to ensure user accountability and to support trip completion tracking.
3. **Data Access Control** – Implemented in parallel with backend development to ensure sensitive data (user info, trip data) is protected and only accessible by authorized roles.
4. **Rating System** – Developed during the user interaction stage to provide feedback mechanisms for improving safety and trust among users.

The following requirements are planned as **future milestones** beyond the capstone project timeline:

- **Audit Logging** – Will be added once the system architecture is stable, to record all security-related actions for monitoring and compliance.
- **Data Encryption Enhancements** – Advanced encryption for all stored and transmitted user data will be prioritized for production release.

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?

   Burhan:
   This document was very helpful as it provided a template to identify potential risks associated with the system and the users. The brainstorming session allowed for a diverse set of ideas to be shared. This was very helpful in setting the basis for this document. We were able to not only come up with hazards that directly affect the users, but we identified hazards related to the system that would indirectly affect the users too. This helped us understand the importance of system failures and the effects of them on users.

   Sarim:
   We were able to identify genuine potential risks that would arise in our project timeline rather than blindly writing a document that would not be of use to us. This is important for us since early identification of risks and hazards is important and ensures we are able to avoid them.

   Hamzah:
   The instructions and template were clear and easy to follow. It was also nice to split up the work in a way that related to the SRS document as well, since I was able to do the requirements in both documents. The hazard analysis helps us think about aspects that we otherwise wouldn't, since this is something that I have not done before.

   Swesan:
   What went well while writing this deliverable was that I am currently taking 3RA3 (Requirements), so a lot of the concepts and techniques overlapped directly with what we needed to produce. This made the process feel more straightforward, since I was already familiar with the structure,

terminology, and expectations for requirements deliverables. It allowed me to apply what I was learning in class right away, which helped speed up the writing process and gave me more confidence that the content was accurate and aligned with best practices.

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

   Burhan:
   Initially, we weren't able to decide how deeply we wanted to discuss each hazard. We had discussed a lot of hazards initially, but it became difficult to decide which one to keep and which one to remove. We came up with the strategy to prioritize the hazards and then check for similarities to avoid redundancy. This helped us ensure that all of the critical hazards were covered and that there were no redundancies.

   Sarim:
   Understanding some of the terminology for my assigned sections was a hurdle I faced for this deliverable. When it came to writing the critical assumptions, I struggled to understand what it was referring to. However, since I started the document early, I was able to identify this issue early, and ask our TA during our meeting.

   Hamzah:
   It was initially a bit difficult to understand how the safety and security requirements in this document are different to what is in the SRS document, but I was able to clarify by asking the TA about this. It relied on what was done in the previous section, so I had to wait for other team members to complete that section before I could finalize mine.

   Swesan:
   One of the main pain points I encountered during this deliverable was creating the diagrams and ensuring they were both accurate and visually clear. This process took more time than expected because I had to revise the structure multiple times to properly capture the relationships between components. Another challenge was making sure everyone's individual sections aligned in tone, terminology, and formatting, since inconsistencies made the overall document feel disjointed at first. To resolve these issues, I set aside extra time to refine the diagrams and cross-check them against the written content, while also coordinating with my teammates to review each other's sections and apply a consistent style guide across the deliverable. This not only improved the clarity of the final submission but also strengthened our collaboration.

Aidan:
The main challenge I faced was distinguishing between what belonged in the SRS versus the Hazard Analysis. Since both documents discuss risks and requirements, it took some coordination to avoid overlap or repetition. Another pain point was keeping the FMEA table concise while still providing enough detail to make each hazard meaningful — at first, it was easy to get lost in too much explanation. I resolved these issues by reviewing past examples and our TA's guidance to clarify expectations, and by working closely with teammates to check for consistency across both documents. Once we streamlined our structure, the writing process became much more efficient.

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?

   One of the obvious ones that arises with platforms like these includes user safety and security. These ideas came from the past experiences of my teammates. The risk of having an unverified driver, breach of data, incorrect ride matches, and fake ratings and reviews, were listed before this deliverable. The rest came up while working on this deliverable. We did some research on similar applications and brainstormed ideas focusing on Hazards that could indirectly affect the user. This deliverable sets the basis for considering all of the components of the application. That is when we came up with some of the system failure hazards like notification delays, loss of trip history data, location API failure, and system unavailability.

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?

   (a) **Risk of Data Breach:** Software platforms generally hold a lot of user data. In some cases, that can be very confidential. This harms a brand's reputation and can cause heavy penalties in some cases.

   (b) **Server Crashes and Unusual Downtime:** This is another critical hazard associated with software platforms which can leave a user stranded for hours and harm the brand's image. This can cause users to move to the platforms and cause the platform to lose users permanently.