Hazard Analysis
Room8

Mohammed Abed
Maged Armanios
Jinal Kasturiarachchi
Jane Klavir
Harshil Patel

2024-10-25

Table 1: Revision History

Date	**Developer(s)**	**Change**
Date1	Name(s)	Description of changes
Date2	Name(s)	Description of changes
...	...	...

Contents

[You are free to modify this template. —SS]

1 Introduction

[You can include your definition of what a hazard is here. —SS]
Room8 is a suite of tools aimed at reducing the occurrence of frustrating situations between roommates. Room8 is expected to be implemented as a mobile application and interact with the physical world using a camera and as a result, is expected to handle sensitive user data such as addresses, names, birthdays, images, and financial details. As a result, the application can pose potential hazards to users with a hazard defined as any property or state of the system that could cause our users harm. This document aims to outline the scope, critical assumptions, potential failures, and mitigation strategies for Room8 in order to allow the development to design mitigations for these hazards.

2 Scope and Purpose of Hazard Analysis

[You should say what **loss** could be incurred because of the hazards. —SS]
The purpose of this document is to identify and mitigate losses that can be incurred from system hazards. There are multiple ways to mitigate the losses, such as following appropriate regulations, implementing thorough testing, and informing users how to properly use the system. By examining as many scenarios as possible where the system can cause harm and recording it in this document, the development team aims to minimize the harm exposed to users, stakeholders, and the development team. Possible losses that can occur from hazards include financial loss, loss of reputation, and service disruptions.

3 System Boundaries and Components

This section goes over the components that the system can be divided into.

3.1 User Device

Smart phone the user is using with the supported version of Android or iOS.

3.2 Camera

Responsible for taking picture for cleanliness detection analysis when sensor sends information of user.

3.3 Motion Sensor

Detects movement in the shared space to determine if user has entered or exited shared space signaling the camera for a picture.

3.4 User-Facing Application

A mobile application installed on smart phones which have versions of Android and iOS that is currently being supported mobile providers. This includes front-end of the system where users can see details and change settings of various back-end components listed below.

3.5 Authentication

Authentication using OAuth of user credentials and house details are processed in this component including the update of information mentioned previously.

3.6 SMS ChatBot

ChatBot responsible for sending messages to group chat of home members for notifying them of cleanliness assessment, expenses from bill splitter, or reminders to complete tasks.

3.7 Calendar Tool

Allows users to add events to calendar and display to other housemates, if involved in event, in their respective calendars. Also houses logic for generating chore/cleaning schedule and adding in calendars of users.

3.8 Cleanliness Manager

Runs algorithm for detecting change in environment through input received from hardware and stores user's information for the user to view on application along with history of cleanliness.

3.9 Bill Splitter

Calculate charges due from a shared expense and keeps track of which expenses are due from each user and who they owe using the SMS ChatBot to notify users. Also stores history of expenses and charges paid for user to view.

3.10 Database

Used to securely store user and house information, calendar events, expense history, and pictures for cleanliness calculator.

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]

- **CA1**: Homes will have a consistent and uninterrupted supply of electricity available.
- **CA2**: Homes will have internet speeds capable of streaming video.
- **CA3**: Every resident of a shared home will have their own personal electronic device.
- **CA4**: Users have used other applications before and are familiar with common signifiers, mappings, and UI metaphores (ex. Heart implies like).
- **CA5**: External services, such as location services, map integrations, and calendar APIs will be available and reliable.
- **CA6**: Users' devices will have additional free storage beyond the what's required for the applications install.
- **CA7**: Camera setup in shared environment will not be moved or blocked to ensure clear pictures of space.

5 Failure Mode and Effect Analysis

[Include your FMEA table here. This is the most important part of this document. —SS] [The safety requirements in the table do not have to have the prefix SR. The most important thing is to show traceability to your SRS. You might trace to requirements you have already written, or you might need to add new requirements. —SS] [If no safety requirement can be devised, other mitigation strategies can be entered in the table, including strategies involving providing additional documentation, and/or test cases. —SS]

Table 2: **FMEA Table**

Design Functions	**Failure Modes**	**Effects of Failure**	**Causes of Failure**	**Recommended Action**	**SR**	**Ref**	**Severity**
Camera Uplaods Image to Cleanliness Detection System	Image gets compressed.	Same as HR2.1.	Network speeds are slow and compresses image to save space and and send image faster.	Take another picture and upload to cleanliness detection system again.	IR1	H1.1	BLANK
Camera Takes Picture	Something irregular occuring in-frame	Cleanliness detection algorithm does not produce good results due to bad input	a. Camera field of view being blocked b. Problem with lighting	a. Take hourly pictures (i.e. as long as motion not detected), use last picture taken b. Same as H2.1a	IR2	H2.1	BLANK
	Delay in camera shot timing	User in-frame, user privacy is breached	a. Motion sensor not properly connected to camera b. Motion sensor not working	a. Remove picture from database, follow H2.1a, do troubleshooting for motion sensor b. Same as H2.2a	SR1	H2.2	BLANK
Camera Uploads Image Pair To Cleanliness Management System	Camera fails to upload the images	Cleanliness management system does not detect or notify users of the changes to the space	a. Internet connectivity issues	a. Retry to upload images multiple times b. Notify users if there has been no activity detected from the camera for an extended period of time	SR?	H3.1	BLANK

Design Functions	**Failure Modes**	**Effects of Failure**	**Causes of Failure**	**Recommended Action**	**SR**	**Ref**	**Severity**
Motion Sensor Detects Motion	Motion should not be detected (false positive)	Unnecessary computation and notifications	a. Motion sensor is overly sensitive b. Some brief/light movement that should not be classified as motion occurred (e.g., insect flew by) c. Continuous movement that should not be classified as motion is occurring (e.g., air conditioner causing curtain to move)	a. Include regular motion-calibration testing in system design b. Have a clause in the cleanliness detection algorithm to check for false positives (i.e., if the before/after pictures do not meet a threshold to be considered dissimilar) c. ????	SR?, SR?	H4.1	Medium
	Motion is not detected although it is occurring (false negative)	Cleanliness detection does not occur	a. Motion sensor is underly sensitive b. Motion sensor is broken	a. Same as H3.1a b. Same as H4.1a	???	H4.2	BLANK

Design Functions	**Failure Modes**	**Effects of Failure**	**Causes of Failure**	**Recommended Action**	**SR**	**Ref**	**Severity**
System Authenticates User	Bad actor is able to log into user's account	Sensitive user information is revealed, such as full name and address. User privacy is violated.	a. Data breach b. Weak account password c. Lack of multi-factor authentication d. Brute force attacks e. Man-int-the-middle attacks f. Out-dated/insecure methods of storing user credentials	a. Check for unusual login patterns, such as different geolocations, IP addresses, and repeated failed attempts on same login b. Require all account passwords to satisfy a minimum password strength criterion c. Reccommend multi-factor authentication for logging in d. Impose rate limits on failed authentication attempts e. IDK HELP f. IDK but is this even an issue if we're using OAuth?	???	H5.1	Medium
	Bad actor alters account credentials	User cannot access their account	a. Lack of multi-factor authentication for changing account credentials	a. Require multi-factor authentication for changing account credentials	???	H5.2	BLANK
	Bad actor impersonates user	Violation of user privacy and distruption of services for other house members	a. Log-in procedures not secure	a. Same as 4.1a - 4.1f	???	H5.3	BLANK

Design Functions	**Failure Modes**	**Effects of Failure**	**Causes of Failure**	**Recommended Action**	**SR**	**Ref**	**Severity**
	User forgot their password	User cannot log-in	a. No way to recover account	a. System has a ”forgot password” clause where multi-factor authentication is used to create new password	???	H5.4	BLANK
ChatBot Sends Notifaction	User is notified of deleted event	False information sent to user and user gets annoyed	a. ChatBot SMS did not get update from calendar that event was deleted and user does not need to be informed	a. Have test cases covering testing if ChatBot SMS is updated when calendar events update	NFR222	H6.1	Medium
	User is not notified of event	Same as HR7.1	a. ChatBot SMS failed to get event from calendar	a. Make sure ChatBot SMS synchronizes with calendar.	NFR222	H6.2	BLANK
Scheduling an Event in the Calendar	Event cannot be scheduled	User is frustrated, and important information is not being sent to roommates	a. Conflict occurred due to multiple users scheduling events simultaneously	a. Put a lock on the calendar resource	SR?, SR?	H7.1	Medium

Design Functions	Failure Modes	Effects of Failure	Causes of Failure	Recommended Action	SR	Ref	Severity
Cleanliness Management System Detects Changes in the Cleanliness of a Room	System falsely concludes that a room has become more dirty. System falsely concludes that a room has become more clean. System concludes that there are no changes despite there being changes. The system concludes that there are changes when there are no changes.	Conflict amongst roommates. Trust in the system declines.	Obstructions in the images captured by the camera. Improper calibration and timing of the motion sensor. Object detection algorithm has errors and classifies items incorrectly in an image.	Create base case tests for the cleanliness management system, including no change, increase, decrease, and no room state change cases. Alert and instruct users to clear camera obstructions before setting up the system.	SR?, SR?	H8.1	Low
Inputting into Bill Splitter	Amount owing not accurate	Users receiving false information	a. User made an error inputting bill b. The bill changed (i.e. amount or people owing) c. Bad actor creating false bills	a. Provide users with a mechanism to edit outstanding bills b. Same as 8.1a c. System has a way for roommates to delete bills and report misuse	SR?, SR?	H9.1	Medium
Database something	NEW	NEW	NEW	NEW	SR?, SR?	H??	Medium

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]

The following lists new requirements which will be added to the Software Requirements Specification Document.

6.1 Safety and Security Requirements

SR1. The system will not show images of other users in the frame without proper sensoring.
Rationale: To maintain user privacy and comfort.

6.2 Access Requirements

AR1. This is a sample.
Rationale: This is a sample.

6.3 Integrity Requirements

IR1. The system will be able to send high quality images over the network without compression.
Rationale: It is important that the image does not lose quality so the cleanliness detection system does not have inaccurate results.

IR2. The system will send clear images of the shared space.
Rationale: It is important the images are not hindered by objects covering majority of the share space.

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

1. While writing this deliverable, the team had a relatively difficult time constructing the FEMA table. Referring back to previous documents, such as the SRS and the Development Plan, made it easy to extract the core design functions of our application and start evaluating the associated hazards. Another thing that had gone well when writing this deliverable was the discovery of new requirements/tests for the sake of minimizing requirements. An example of this is the cleanliness management system, we determine that the recommended action for ensuring no false positives and negatives is to create a set of test cases. Finally, we think a good thing that came out of this deliverable was that we realized we had missed a lot of requirements from our SRS, and we're expecting to use what we learned from this document to revise it.

2. The most prominent pain point when writing this deliverable was that we had difficulty dividing the work and working in parallel. This document is a lot shorter than the SRS, and in this deliverable, each section was a lot more dependent on its previous sections. When it was time to work on the deliverable, the team got caught up with other obligations, leading to less time to work on this than we would have liked. Combining this fact with the fact that some people had to wait for others in order to finalize their sections led to a time crunch.
   Another pain point was the flexibility in formatting our document compared to the SRS. The SRS was more structured than this document, and we knew exactly how to format our deliverable, as there wasn't freedom to alter it. This deliverable gave the group a lot more freedom in formatting some sections, which led to inconsistencies when working on the deliverable. These inconsistencies required group discussions as we needed to agree on a style, and then we had to revise our document to ensure it was cohesive and consistent.

3. The listed risk related to the design function "Camera Takes Picture" with the effect "User in-frame, user privacy is breached", had been discussed before. This risk was discovered by our supervisor Dr. Ratnasingham (Thamas) Tharmarasa, who pointed out that many users would feel uncomfortable being photographed. The discussion resulted in us revising our idea to capture the state of the room before and after a user interacts with a shared living space and declaring that photographing a user was a failure and a breach of their privacy. Some other risks, particularly involving bad actors accessing user accounts, had been discussed before this deliverable, but only in passing when some implementation ideas were discussed as a group in an informal setting. The other risks had all come about naturally by simply trying to ask ourselves, "What could happen?" and "If this did happen, what could go wrong?".

4. Other risks in software products include data privacy risks and security risks. Data privacy risks refer to risks that arise when a software product handles users' personal data poorly, leading to potential data leaks. Data privacy is arguably the most important requirement when designing a software system that accesses sensitive user data. If you mishandle a user's private information, the developer of the software is liable for the damage caused, which could be very severe depending on the type of data stolen. Not only will mishandling user data lead to a severe loss in user trust, but failing to comply with regulations can lead to legal consequences.
   Security risks involve vulnerabilities that could allow malicious actors to gain unauthorized access, manipulate data, or disrupt services. Security breaches are important as well because they can cause severe harm to users and lead to financial loss due to operational losses. Without the proper security measures, malicious actors could disrupt your service, gain access to large amounts of user data, and use that to produce harm outside the scope of your software service.