SOFTWARE REQUIREMENTS SPECIFICATION

Room8

Version 1.0.0

Prepared by:
Mohammed Abed
Maged Armanios
Jinal Kasturiarachchi
Jane Klavir
Harshil Patel

McMaster University

April 4, 2025

Revision History

Name	Date	Reason For Changes	Version
Mohammed A. Maged A. Jinal K. Jane K. Harshil P.	2024-10-11	First Draft	0.0
Mohammed A.	2024-11-04	Added FR225 for ChatBot	0.1
Harshil P.	2025-04-04	Removed and modified cleanliness manager FRs and NFRs	1.0

This document follows the requirements documentation structure presented in the Handbook of requirements and business analysis, by Bertrand Meyer.

Contents

Goals

Contents

G.1 Context and overall objective

Shared living environments often create tension between roommates. It is usually cumbersome to reach out to a roommate about messes they left behind in a shared space, concerns over bill payments, household upkeep, and more. Our project, Room8, aims to prevent these cumbersome interactions by providing an application and the necessary hardware components to help monitor the cleanliness of shared spaces, schedule tasks, track shared billing cycles, and alert roommates of any issues within the shared house.

Goal 1. Create a system that monitors the cleanliness of a shared living space, such as a kitchen or a living room.

Goal 2. Provide roommates with a centralized platform to manage, schedule, and task household tasks.

Goal 3. Provide roommates with a centralized platform to manage, schedule, and track bill payments.

Goal 4. Alert and remind roommates of issues regarding issues related to the shared living spaces, reducing tension between each other.

G.2 Current situation

Room8 is designed to reduce points of tension and reduce the need for frustrating dialogue commonplace with people in shared housing, especially students. Currently, in order to manage the many shared responsibilities, roommates have to utilize multiple techniques or tools such as physical calendars, notification reminders, and existing applications such as Splitwise which do not have any integrations between each other. Additionally, it is estimated that approximately 25% students experience conflicts with a roommate, harming academic performance and inducing stress [1]. Recognizing these factors, Room8 aims to create a centralized platform that not only simplifies communication but reduces stress and conflict to improve the lives of students.

G.3 Expected benefits

As stated in section G.2, conflicts between roommates create more than issues in the home and involve more than just misunderstandings in scheduling tasks or living standards. By creating a centralized suite that will help roommates monitor, schedule, and coordinate, Room8 will reduce conflicts between roommates over misunderstandings and miscommunications, and allow for greater flexibility in assigning responsibilities. Additionally, reduced conflicts will lead to mental health benefits for those living in the home by lowering conflict-related stress. Finally, Room8 is expected to improve academic performance in some students who are hindered academically by conflicts with roommates (approximately 17% of students [1]).

Benefit 1. An improvement in response times amongst roommates over household concerns.

Benefit 2. Increased flexibility in task scheduling and management, due to the presence of a centralized application.

Benefit 3. Improved communication and transparency over household matters between roommates.

Benefit 4. A reduction of events in which a roommate forgets or neglects their responsibilities to the home.

Benefit 5. Improved household cleanliness and upkeep.

Benefit 6. A reduction in conflicts between roommates over household matters.

Benefit 7. A reduction in stress for roommates due to reduced conflicts amongst each other.

Benefit 8. Improved mental health for students who experience conflicts with roommates.

Benefit 9. An improvement in academic performance for students experience stress caused by issues with roommates.

G.4 Functionality overview

User & House Management: The system allows students to register, create, and manage their accounts. Students are also able to create homes and invite other students.

Cleanliness Management: Students are able to set up cameras in their shared living spaces. Machine learning algorithms are used to detect messes and assign them to the students to increase accountability and to reduce communication friction.

Scheduler: The system allows users to create and manage chore and cleaning schedules. It will also send reminders to users about their assigned chores and cleaning tasks. Additionally, users are able to "book" common areas that are available within the shared space (i.e. party room, study room, etc.) to avoid conflicts

Bill Splitter: Students are able to add shared expenses to the house and keep track of who owes what. The system will automatically split the bills between the chosen students.

Chat: The system provides an exportable ChatBot to SMS group chats to aid in home management and communication. The ChatBot will be able to send reminders and notifications to the group chat about upcoming events, chores, and bills.

Compliance and Data Privacy: The system follows data privacy and compliance with Personal Information Protection and Electronic Documents Act (PIPEDA), Ontario's Freedom of Information and Protection of Privacy Act (FIPPA), and Anti-Spam Legislation (CASL) to ensure that students' data is secure and not shared with third parties.

Progressive Web App: The system is hosted as a progressive web app (PWA), so users can use it on their phones and computers.

G.5 High-level usage scenarios

Figure 1: High level use case diagram

- UC1: Account Management
  1. User accesses the Room8 app.
  2. User selects "Create Account" or "Log In".
  3. User fills in personal information (name, email, password).
  4. User submits the form.
  5. System validates the information and authenticates the account.
  6. System redirects the user to their dashboard.
- UC2: Home Management
  1. User navigates to their dashboard.

2. User selects the "Create Home" or "Join an Existing Home" option.
3. User enters home details (home name, address, etc.).
4. User sends invitations to housemates by entering their email addresses.
5. System sends email notifications to invited users.
6. Invited users accept or decline the invitation.

- UC3: Schedule Manager
  1. User navigates to the "Schedule" section.
  2. User selects "Add New Chore" or "Add New Event".
  3. User inputs chore details (name, description, time, frequency, assigned users, etc.).
  4. System saves the chore and sends reminders to the assigned users.
  5. Users are able to view the schedule and mark chores as complete.
- UC4: Bill Splitter
  1. User navigates to the "Bill Splitter" section.
  2. User selects the "Add Expense" option.
  3. User enters the total expense amount and description.
  4. User selects which housemates are responsible for the expense.
  5. System splits the expense and notifies the involved users.
  6. Users are able to view the total expenses and who owes what.
  7. Users are able to mark expenses as paid.
- UC5: Cleanliness Management
  1. User navigates to the "Cleanliness Management" section.
  2. System displays the list of objects added, moved or removed.
  3. User selects another user to view their cleanliness tasks.
  4. System displays the selected user's pending cleanliness tasks.
- UC6: SMS ChatBot
  1. User navigates to the "Chat Management" section.
  2. User selects the option to add a ChatBot to their house group chat.
  3. User inputs notification and reminder settings.
  4. System generates an SMS numbers and instructions to add the ChatBot to the group chat.
  5. The ChatBot sends a welcome message to the group.
  6. The ChatBot sends notifications and reminders to the group chat.

G.6 Limitations and exclusions

Below is a list of limitations and exclusions the system will not address:

Limitation 1. System will not track activity completed by the user in the shared environment.

Exclusion 1. System will not request or send money directly to users in the bill splitting functionality.

Exclusion 2. System will not use images taken to train machine learning model.

G.7 Stakeholders and requirements sources

Stakeholder	**Category**
Students	Direct
Landlords and Residence Assistants	Indirect
University Housing Committee	Indirect

Table 1: Stakeholders and Categories

G.7.1 Direct Stakeholders

Students Off Campus
Students off campus are the primary direct stakeholders for Room8. They are the main users of the mobile application who create houses within the app and set up the camera systems. These students seek to maintain cleanliness in their shared living spaces and establish accountability when a mess is left behind by a roommate. The application addresses common challenges faced by students in shared living arrangements, such as maintaining cleanliness, splitting expenses, and scheduling activities.

Students On Campus
Students on campus are the other direct stakeholders for Room8. They are the main mainly concerned with the chore and activity scheduling aspect of the app. In most cases students living on campus do not have a kitchen and individuals are responsible for keeping their own rooms clean so these types of students would benefit from scheduling the cleaning of public spaces (eg. pool table, game space) and seeing events that are happening.

G.7.2 Indirect Stakeholders

Landlords
Landlords are the main indirect stakeholders to the project. Landlords are renting out their homes to students and are concerned with the condition their house is kept in.

They are looking to maintain the clean condition of their home which is done by holding students accountable for messes that are made.

Residence Assistants
Resident assistants are another indirect stakeholder who would benefit from Room8 specifically the activity scheduling feature. Resident assistants are responsible for hosting events for students living on campus and Room8 provides a calendar system to make events on a shared calendar to inform students of any upcoming activities.

University Housing Committee
The University Housing Committee is another key indirect stakeholder in the project. These committees often seek to help students transition into living in shared spaces and provide guidance and support. Room8 offers a wide range of services that address common points of frustration faced by students, which are often brought up to these university committees. By facilitating better communication and organization, Room8 helps enhance the overall living experience for students.

Environment

Contents

E.1 Glossary

The glossary provides definitions for the key terms used throughout the project. It includes terminology related to the system's environment and functionality, ensuring that all parties involved have a clear understanding of the concepts and language specific to the project. This helps prevent miscommunication and fosters consistency in documentation and development.

SRS (Software Requirements Specification): A formal document (this document) that outlines the functional and non-functional requirements of a system or software application. It serves as a blueprint for both developers and stakeholders, detailing what the system is expected to do, how it should perform, as well as assumptions/ constraints for its development.

Room8: The name of the application being described by this SRS. Room8 is an app designed to help students manage shared living spaces. It includes functionalities like cleanliness monitoring, chore scheduling, bill splitting, and communication tools to streamline household tasks.

Home: In the context of Room8, a "home" refers to the digital representation of a shared living space comprised of students. Essentially, it is the user group for one living space where all members can manage, organize, and recieve alerts for household responsibilities.

Students: In the context of Room8, a "student" is a user of the app.

V&V (Verification and Validation): A process in software development aimed at ensuring that a system meets its specifications (verification) and that it fulfills its intended purpose (validation).

Splitwise: A real-world app used for tracking and splitting shared expenses between individuals.

Chatbot: Software designed to simulate conversation with users, often through text-based interaction. In Room8, the ChatBot automates tasks such as sending reminders, notifications, and facilitating communication about chores and bills in group chats.

PIPEDA (Personal Information Protection and Electronic Documents Act): A Canadian law governing how private sector organizations collect, use, and disclose personal information in the course of their business activities. Compliance ensures that user data is protected and handled with consent.

FIPPA (Freedom of Information and Protection of Privacy Act): A piece of Canadian legislation that outlines the rights of individuals to access government information and the responsibilities of public organizations to protect personal privacy. It helps ensure the secure handling of personal information.

CASL (Canadian Anti-Spam Legislation): A Canadian law aimed at regulating commercial electronic messages, including emails and texts, to prevent spam. It requires consent before sending such communications, ensuring transparency and privacy.

"Before" Picture: The image of the shared living space before a student enters and uses it.

"After" Picture: he image of the shared living space after a student uses it.

Cleanliness Tasks: A list assigned to a user after they have finished using the shared living space and have assigned themselves or been assigned by another user. It is derived from an algorithm in which the inputs are a "before" and "after" picture pair.

OAuth 2.0: An open standard protocol for authorization, allowing secure access to user data across different services without revealing login credentials. OAuth is

commonly used to enable single sign-on or to grant third-party applications limited access to a user's resources.

PPM (Pixels Per Metre): A measurement used to define the resolution of digital images, specifically indicating the number of pixels within a meter of space. It is often used in imaging, mapping, and display technologies to describe the level of detail or clarity in an image.

FoV (Field of View): In digital images, FoV refers to the extent of the observable area at any given moment. It is the angle/ scope that the camera sees.

UTC (Coordinated Universal Time): A time standard used globally to regulate clocks and time. It is the primary time standard by which the world regulates time and is not affected by time zones or daylight saving changes. UTC is crucial for coordinating activities across different time zones, especially in computing, aviation, and telecommunications.

JS (JavaScript): A popular, lightweight programming language commonly used to build dynamic content on websites. JavaScript allows developers to create interactive web pages and web applications.

TS (TypeScript): A superset of JavaScript that introduces static types, making the code more robust and easier to debug. TypeScript is particularly helpful for large-scale applications by catching potential errors at compile time.

AWS (Amazon Web Services): A comprehensive cloud computing platform provided by Amazon, offering a wide range of services such as computing power, storage, databases, and machine learning. AWS enables scalable and secure cloud solutions for various applications.

CI/CD (Continuous Integration and Continuous Deployment/Development): A set of practices in software development aimed at automating and streamlining the process of integrating code changes (CI) and deploying applications (CD). CI/CD improves code quality, speeds up development cycles, and ensures faster delivery of new features or updates.

PoC (Proof of Concept): A prototype or small-scale experiment conducted to verify that an idea or technology is feasible. In software development, a PoC helps validate whether a particular solution or approach will work as intended before full-scale implementation.

E.2 Components

This section outlines the relevant external components that the system will interact with. These include existing systems or services, particularly software, that will provide or consume APIs. These interactions play a significant role in how the system operates within its broader ecosystem, impacting design and integration.

Motion-activated camera: Room8 will be connected to a camera that is activated by motion in the shared space. This component is used for taking a "before" and "after" picture that serves as the input for the cleanliness detection model.

Object detector API: Room8 will utilize an object detector that performs on a pair of images (i.e. the "before" and "after" picture) taken by the motion-activated camera. Object detection enables the algorithm to quantify the difference in room states as a function of objects and their transformations.

Household items dataset: The ML model will be pre-trained with a dataset containing common household and kitchen items. This will allow the object detector to classify the items it detects and is important for providing valuable output to students.

Google calendar API: Room8 will interface with Google calendar to facilitate scheduling chores and events. This is critical for Room8's chore scheduling.

SMS ChatBot: The application employs a ChatBot to send notifications to students via SMS about their responsibilities in the home (i.e. cleaning their mess from shared spaces, upcoming chore deadlines, and money owed for shared bills). This helps with bookkeeping, enforcing fairness, and keeping shared spaces clean.

OAuth 2.0: The app utilizes the OAuth 2.0 protocol to manage user authentication and authorization securely. OAuth 2.0 allows Room8 to provide a seamless login experience using third-party identity providers without requiring users to create and manage separate credentials. OAuth 2.0 handles the app's legal and privacy compliance standards.

E.3 Constraints

Constraints are non-negotiable limitations imposed on the system, stemming from external factors such as business rules or technical requirements. These restrictions must be strictly adhered to during development. Constraints can influence system design

and implementation, and their fulfillment is critical to ensuring the system meets its operational requirements.

Camera resolution: The resolution of the motion-activated camera affects the quality of the input that is used by the object detector. Depending on the PPM of the image, there is a minimum viable object size for a given object to be classified.

Platform compatibility: The application must be compatible on Android and iOS to accommodate all users in the home.

Data privacy compliance: The application adheres to all applicable data privacy regulations for the region in which it operates. User data must be handled with the utmost care and cannot be shared or used without explicit consent. Prior to collecting, storing, or processing any personal data, the appropriate permissions must always be obtained to ensure compliance with privacy laws and to respect user rights.

E.4 Assumptions

Assumptions are conditions that are presumed to be true during development to simplify system design. These are not externally enforced but are accepted for the sake of convenience or efficiency. While helpful, assumptions must be carefully evaluated, as changes in the assumed conditions could affect the system's success or performance.

Network connectivity: The application assumes that there is reliable internet connection. Operating under this assumption, the camera is able to transfer images to the detector and students have access to updated synchronized information when using the app.

Operating system support: The application is compatible with the versions of Android and iOS that are currently supported by mobile providers.

Common household items dataset: The classifier can assume that any object it comes across is a common household item that belongs to a class represented in its training data.

FoV of the camera: The camera is stationary and the FoV is identical in any pair of "before" and "after" images.

E.5 Effects

This section describes how the system's operations will influence its environment. These effects could be on processes, workflows, or other systems that interact with the system. Understanding these impacts is crucial to ensure that the system's operation aligns well with its intended environment and does not cause unintended disruptions.

Digital minimalism: Students part of a home cab conduct all interactions on one platform. All common virtual services are provided by the Room8 application, hence eliminating the need for other applications that clutter the digital space.

Accountability: Room8 promotes social responsibility and user accountability by assessing the cleanliness state of the room before/after use and quantifying the difference in the 2 states, providing a cleanliness task list. Students are given objective feedback and all students in the home are held accountable for their actions.

Ease social tension: Students living in a home can feel tense about confronting their roommates when asking them to clean up, pay them back, or do the chores they agreed to. The ChatBot sends reminders about these duties, taking the pressure off students to confront each other and risk compromising their relationships.

E.6 Invariants

Invariants are properties of the environment that the system must maintain throughout its operation. These are foundational conditions that must hold true at all times during system activities. Ensuring that these invariants remain intact is essential for the correct and safe functioning of the system within its environment.

Limited camera presence: The motion-activated camera is initially triggered by motion, and takes a "before" picture. When motion has not been detected for a prolonged and predefined amount of time, it takes an "after" picture. These 2 images aside, the camera does not take any pictures or footage and thus does not have any insight on the activities occurring in the environment.

Students' freedom to not comply: While the application sends reminders to students for them to pay their debts or clean up their messes, it cannot enforce these events. Room8 serves as a logistical coordinator but it cannot have a direct impact on the physical environment.

System

Contents

S.1 Components

1. Front-End Cluster

- **1.1 - User Authentication and House Management:** The user interface which allows the user to log in and manage their account and/or house. It depends on a corresponding back-end component and allows the user to access all other functionality within the application once authenticated.

- **1.2 - ChatBot Configuration:** The view which allows users to configure their ChatBot settings. This includes features such as opting in or out of certain personal and group notifications.
- **1.3 - Cleanliness Manager:** Allows users to view their relevant details within the cleanliness detection section of the application. Includes details such as history of events within the shared space (i.e. time of entry, time of exit, changes made, etc.).
- **1.4 - Schedule Configuration:** Users are able to configure details related to the scheduler within this view. Options to add/remove chores (including assigned users) and events (booked spaces) are found here.
- **1.5 - Bill Splitter Configuration:** The area of the application where users can input and view details regarding shared expenses. Includes configuration to add/remove/edit bills and which users to split with, as well as outstanding balances and history of balances paid. Additionally, users can manually "settle" their outstanding payments by confirming that they've been paid within this interface.

2. Back-End Cluster

- **2.1 - Auth and House Management Logic:** The logic for authentication and user/house details are housed here. Any updating of information (as per user request) will be carried out in this component. Authentication (through OAuth) will also be the responsibility of this software.
- **2.2 - SMS ChatBot:** Responsible for sending out ChatBot messages to home group chats including reminders to complete tasks, notification of cleanliness assessment, expenses due from bill splitter, or update to configuration of user/home. Has the most dependencies with other back-end components since updates in those components will drive the output of new messages from the bot.
- **2.3 - Cleanliness Calculator:** Runs the algorithm to determine which changes have been made in the shared space by the identified user. Receives input from the hardware (dependenciy) and returns changes to the GUI after each event.
- **2.4 - Schedule Generator:** Houses logic that deals with the scheduling aspect of the application. Generates events to be displayed to each users calendar based on whether or not they're included in a chore or event.
- **2.5 - Bill Split Calculator:** Calculation of charges due for shared expenses is done within this component. It will also keep track of which expenses are due by each user and who they owe, along with a history of charges paid.

3. Hardware Cluster

- **3.1 - Camera + Sensors:** All hardware used for the cleanliness calculation. Includes sensors to detect when a user enters or exits the shared space. Takes images which are sufficient quality for detecting the changes that a user makes within the space.

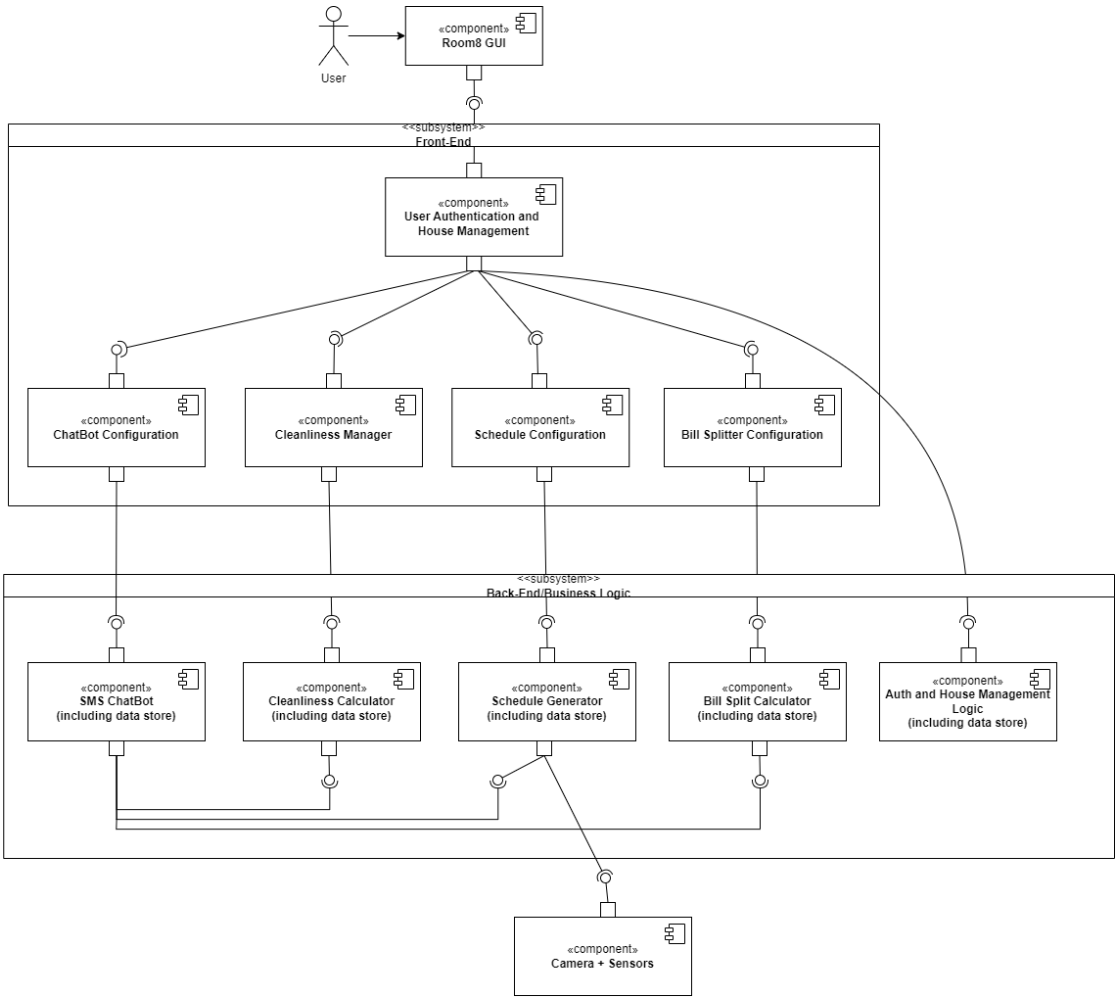

Figure 2: Component Diagram

S.2 Functionality

S.2.1 - User Authentication and House Management

- **Functional Requirements:**

- **FR211:** The system shall allow users to create an account using their accounts.
- **FR212:** The system shall allows users to log in to their account.
- **FR213:** The system shall allows users to log out of their account.
- **FR214:** The system shall allow users to create a home group using the home name, address, and number of roommates.
- **FR215:** The system shall allow users to invite other users to join their home group.
- **FR216:** The system shall allow users to view the list of users in their home group.
- **FR217:** The system shall allow users to remove users from their home group.
- **FR218:** The system shall allow users to leave their home group.

- **Non-Functional Requirements:**
  - **NFR211:** All data related to authentication must be encrypted in both transit and in storage.
  - **NFR212:** Error messages related to authentication should not disclose sensitive details such as "Incorrect Password".
  - **NFR213:** All data related to houses such as addresses and residents should be encrypted in transit and in rest.
  - **NFR214:** The system should be able to authenticate a user with a median response time of under 1 second.

S.2.2 - ChatBot Configuration

- **Functional Requirements:**
  - **FR221:** The system shall allow users to configure the ChatBot settings to include or exclude messages corresponding to chore schedule, cleanliness manager, and bill splitter.
  - **FR222:** The ChatBot shall send reminders to the group chat about upcoming chores and events in the schedule 2 days in advance.
  - **FR223:** The ChatBot shall send notifications to the group chat about new shared living space cleanliness tasks immediately after an event is added to the cleanliness manager page.
  - **FR224:** The ChatBot shall send notifications to the group chat about new shared expenses added to the bill splitter page immediately after its addition.
  - **FR225:** The system shall allow users to activate the ChatBot feature which will create combined groupchat including all the residents as well the bot itself.

- **Non-Functional Requirements:**
  - **NFR221**: The chatbot shall not disclose any sensitive information in its messages such as addresses or full names.
  - **NFR222**: The chatbot shall not send users too frequently to prevent annoying users.

S.2.3 - Cleanliness Manager

- **Functional Requirements:**
  - **FR231:** The system shall evaluate the cleanliness of the shared living space before and after a user enters and exits the space.
  - **FR233:** The system shall display the detected messes in the shared living space.
  - **FR234:** The system shall allow users to view the history of cleanliness tasks completed or pending and detected messes.
- **Non-Functional Requirements:**
  - **NFR231:** The system shall not record users.
  - **NFR232:** The system shall not capture images of users.
  - **NFR233:** The system shall encrypt and securely store all images of homes.
  - **NFR234:** Photos captured with the system will be in a quality high enough to differentiate objects within frame.
  - **NFR235:** The system shall process image data in under 30 minutes.
  - **NFR236:** The system shall not report false events.
  - **NFR237:** The system shall declare an instances of someone altering a room finished one there has been no activity in the room for a designated period of time.

S.2.4 - Schedule Configuration

- **Functional Requirements:**
  - **FR241:** The system shall allow users to add a new chore to the schedule.
  - **FR242:** The system shall allow users to add a new event to the schedule.
  - **FR243:** The system shall allow users to input chore and event details (name, description, time, frequency, assigned users, etc.).
  - **FR244:** The system shall allow users to edit and delete chores and events.
  - **FR245:** The system shall allow users to view the schedule and mark chores as complete.
- **Non-Functional Requirements:**

- **NFR241:** The calendar system shall store all calendar events in UTC
- **NFR242:** The calendar system shall display all calendar events to users in their time zone.
- **NFR243:** The calendar system shall have a granularity of 5 minutes.
- **NFR244:** The calendar system shall encrypt all events stored.

S.2.5 - Bill Splitter Configuration

- **Functional Requirements:**
  - **FR251:** The system shall allow users to add a new expense to the bill splitter and notify the involved users.
  - **FR252:** The system shall allow users to view what they owe other housemates.
  - **FR253:** The system shall allow users to view what they owe others.
  - **FR254:** The system shall allow users to mark expenses as paid.
  - **FR255:** The system shall calculate debts in order to minimize the amount of transactions required between housemates.
- **Non-Functional Requirements:**
  - **NFR251:** The Bill Splitter system shall encrypt all events stored.
  - **NFR252:** The Bill Splitter shall allow users to record numerical values of prices with a granularity of two decimal places.

S.3 Interfaces

At the time of writing, the application has not yet been implemented and no interfaces are being used. Thus, S.3 Interfaces will be available in a future revision of this document.

S.4 Detailed usage scenarios

S.4.1 User Creating an Account

- **Use Case:** UC1
- **Primary Actor:** New User
- **Precondition:** New User has a compatible device with internet access and opened the Room8 app.
- **Trigger:** New user wants to start using the app to manage their student home.
- **Main Success Scenario:**
  1. User accesses the Room8 app.

2. User selects the "Create Account" option.
3. User fills in personal information (name, email, phone number, password).
4. User submits the form.
5. System validates the information and creates the account.
6. System authenticates the user.
7. System redirects the user to their dashboard.

- **Secondary Scenarios:**

  5.1 System is unable to create new user's account and informs the user of this error.

  5.1.1 System informs the user of the server error.

- **Success Postcondition:** New user is successfully registered and authenticated in the system. User is redirected to their dashboard with the option to create a home or join an existing one.

S.4.2 User Logging Into an Account

- **Use Case:** UC1
- **Primary Actor:** User
- **Precondition:** User has an existing account and is not already logged in.
- **Trigger:** User wants to log into their account to use application features.
- **Main Success Scenario:**
  1. User accesses the Room8 app.
  2. User selects the "Log In" option.
  3. User enters email and password.
  4. User submits the form.
  5. System authenticates the credentials and logs in the user.
  6. System redirects the user to their dashboard.
- **Secondary Scenarios:**

  5.1 User enters an invalid email and password combination.

  5.1.1 System informs the user of the invalid credentials.

  5.2 System is unable to log into user's account and informs the user of this error.

  5.2.1 System informs the user of the server error.

- **Success Postcondition:** User is successfully logged into the system.

S.4.3 User Logging Out of the System

- **Use Case:** UC1
- **Primary Actor:** User
- **Precondition:** User is already logged into their account.
- **Trigger:** User wants to log out of the system.
- **Main Success Scenario:**
  1. User accesses the main menu of the Room8 app.
  2. User selects the "Log Out" option.
  3. System prompts for confirmation.
  4. User confirms the action.
  5. System logs the user out and redirects to the login screen.
- **Secondary Scenarios:**

  4.1 User cancels the log out action.
- **Success Postcondition:** User is successfully logged out of the system.

S.4.4 User Creates a Home

- **Use Case:** UC2
- **Primary Actor:** User
- **Secondary Actor:** User's roommates
- **Precondition:** User is logged in and has no existing home.
- **Trigger:** User wants to create a home to manage their living situation with their roommates.
- **Main Success Scenario:**
  1. User navigates to their dashboard.
  2. User selects the "Create Home" option.
  3. User enters home details (home name, address, etc.).
  4. User sends invitations to housemates by entering their email addresses.
  5. System sends email notifications to invited users.
  6. Invited users accept or decline the invitation.
- **Secondary Scenarios:**

  4.1 User does not invite any housemates and chooses to invite them later.

- **Success Postcondition:** User has successfully created a home and invited housemates to join.

S.4.5 User Joins a Home

- **Use Case:** UC2
- **Primary Actor:** User
- **Secondary Actor:** User's roommates
- **Precondition:** User is logged in and has no existing home.
- **Trigger:** User wants to join their home's existing group.
- **Main Success Scenario:**
  1. User receives an email invitation to join a home.
  2. User clicks on the provided link in the invitation.
  3. System redirects user to the Room8 app.
  4. User logs in.
  5. User confirms their decision to join the home.
  6. System adds the user to the home group.
  7. System notifies the housemates of the new addition.
- **Secondary Scenarios:**
  - 4.1 User does not have an account and is prompted to create one.
  - 5.1 User is unable to join the home and is informed of this error.
  - 5.2 User declines the prompt and is not added to the home.
- **Success Postcondition:** User has join home alongside their housemates.

S.4.6 User Leaves a Home

- **Use Case:** UC2
- **Primary Actor:** User
- **Secondary Actor:** User's roommates
- **Precondition:** User is logged in and has an existing home.
- **Trigger:** User wants to leave the home attached to their account.
- **Main Success Scenario:**
  1. User navigates to their Home settings.

2. User chooses the "Leave Home" option.
3. System prompts for confirmation.
4. User confirms the action.
5. System removes the user from the home group.
6. System redirects the user to their dashboard.
7. System notifies the housemates of the user's departure.

- **Secondary Scenarios:**

  4.1 User aborts leaving the home and remains in the group.

- **Success Postcondition:** User has left the home and is no longer associated with the group. User is in the dashboard page with the option to create a home or join an existing one.

S.4.7 User Schedules a Chore

- **Use Case:** UC3
- **Primary Actor:** User
- **Secondary Actor:** User's roommates
- **Precondition:** User is logged in and has an existing home.
- **Trigger:** User wants to schedule a chore amongst housemates to assign responsibilities.
- **Main Success Scenario:**
  1. User navigates to the "Schedule" section.
  2. User selects "Add New Chore".
  3. User inputs chore details (name, description, time, frequency, etc.).
  4. User assigns housemates to the chore.
  5. User confirms chore creation.
  6. System saves the chore and adds it to the schedule.
  7. System notifies housemates of the new chore.
- **Secondary Scenarios:**

  3.1 User does input all required chore details correctly.

  3.1.1 System prompts the user to fill in all required fields.

  5.1 User cancels the chore creation.

- **Success Postcondition:** New chore is added to the schedule and housemates are notified. Housemates are reminded of the chore when it is approaching.

S.4.8 User Edits a Chore

- **Use Case:** UC3
- **Primary Actor:** User
- **Secondary Actor:** User's roommates
- **Precondition:** User is logged in and has an existing home. User has an existing chore.
- **Trigger:** User wants to update chore details.
- **Main Success Scenario:**
  1. User navigates to the "Schedule" section.
  2. User selects a scheduled chore.
  3. User chooses the "Edit" option.
  4. User modifies the chore details (name, description, time, frequency, assigned users, etc.).
  5. User confirms the changes.
  6. System updates the chore and notifies relevant housemates.
- **Secondary Scenarios:**

  5.1 User cancels the chore creation.
- **Success Postcondition:** Chore is updated and housemates are notified of the updated chore.

S.4.9 User Completes a Chore

- **Use Case:** UC3
- **Primary Actor:** User
- **Precondition:** User is logged in and has an existing home. User has an existing chore.
- **Trigger:** User has completed their chore duties and wants to update the schedule to reflect their completed duties.
- **Main Success Scenario:**
  1. User navigates to the "Schedule" section.
  2. User selects a chore assigned to them.
  3. User is able to see chore details, description, and due date.
  4. User chooses the "Complete" option.

5. System marks the chore as complete and updates the chore history.

- **Success Postcondition:** Chore is completed and system displays the next upcoming undone chore. Chore history reflects list of completed chores in the home.

S.4.10 User Schedules an Event

- **Use Case:** UC3
- **Primary Actor:** User
- **Secondary Actor:** User's roommates
- **Precondition:** User is logged in and has an existing home.
- **Trigger:** User wants to schedule the use of a shared living space and notify housemates.
- **Main Success Scenario:**
  1. User navigates to the "Schedule" section.
  2. User selects the "Add Event" option.
  3. User inputs event details (date, time, description).
  4. User sets reminders or notifications.
  5. User confirms the event creation.
  6. System adds the event to the users' schedule.
- **Secondary Scenarios:**

  3.1 User does not input all required event details correctly.

  3.1.1 System prompts the user to fill in all required fields.

  5.1 User cancels the event creation.
- **Success Postcondition:** New event is added to the schedule and housemates are notified. Housemates are able to view event in the schedule. Housemates are reminded of the event when it is approaching.

S.4.11 User Views Chores Or Events

- **Use Case:** UC3
- **Primary Actor:** User
- **Precondition:** User is logged in and has an existing home.
- **Trigger:** User wants to view the history of completed and pending chores.
- **Main Success Scenario:**

1. User navigates to the "Schedule" section.
2. User selects the "Chore History" option.
3. System displays a list of completed and pending chores.
4. User can filter or search through the history.

- **Secondary Scenarios:**

  3.1 User does not have any completed or pending chores.

  3.1.1 System informs the user that there are no chores to display.

- **Success Postcondition:** User is able to view the history of completed and pending chores. User is able to filter and search through the history.

S.4.12 User Adds an Expense

- **Use Case:** UC4
- **Primary Actor:** User
- **Secondary Actor:** User's roommates
- **Precondition:** User is logged in and has an existing home.
- **Trigger:** User wants to add a shared expense to the house and split the bill between housemates to keep track of who owes what.
- **Main Success Scenario:**
  1. User navigates to the "Bill Splitter" section.
  2. User selects the "Add Expense" option.
  3. User enters the total expense amount and description.
  4. User selects which housemates are responsible for the expense.
  5. User confirms the expense.
  6. System splits the expense and notifies the involved users.
- **Secondary Scenarios:**

  3.1 User does not input all required expense details correctly.

  3.1.1 System prompts the user to fill in all required fields.

  5.1 User cancels the expense creation.

- **Success Postcondition:** New expense is added. Expense is split between housemates and housemates are notified. Housemates are able to view the expense in the bill splitter section.

S.4.13 User Views Expenses

- **Use Case:** UC4
- **Primary Actor:** User
- **Secondary Actor:** User's roommates
- **Precondition:** User is logged in and has an existing home. Home has existing expenses.
- **Trigger:** User wants to view the history of expenses and bills in the house. User wants to see who owes them money. User wants to see who they owe money to.
- **Main Success Scenario:**
  1. User navigates to the "Bill Splitter" section.
  2. User selects the "Expense History" option.
  3. System displays a list of past payments and outstanding bills.
  4. User can filter the history by type or amount.
- **Success Postcondition:** User is able to see current list of housemates and their outstanding bills between themselves and the other housemates. User is able to filter and search through the history of shared expenses.

S.4.14 User Pays an Expense

- **Use Case:** UC4
- **Primary Actor:** User
- **Secondary Actor:** User's roommates
- **Precondition:** User is logged in and has an existing home. User has an existing unpaid bill with one of the housemates.
- **Trigger:** User wants to mark an expense between themselves and another housemate as paid and update the payment status.
- **Main Success Scenario:**
  1. User navigates to the "Bill Splitter" section.
  2. User selects their profile's bills.
  3. User selects an unpaid bill.
  4. User marks the bill as paid.
  5. System updates the payment status and recalculates debts.
  6. System notifies other housemates about the payment status.

- **Success Postcondition:** Expenses are updated and expense log reflects expense transactions. Debts are recalculated to facilitate the least amount of transactions between housemates.

S.4.15 User Views Cleanliness Tasks

- **Use Case:** UC5
- **Primary Actor:** User
- **Precondition:** User is logged in and has an existing home. Home has cleanliness detector setup.
- **Trigger:** User wants to view current cleanliness tasks of the shared living space and latest usage.
- **Main Success Scenario:**
  1. User navigates to the "Cleanliness Management" section.
  2. System displays the current cleanliness tasks and detected messes.
  3. User selects another user to view their cleanliness tasks.
  4. System displays the selected user's current and pending cleanliness tasks associated with their shared living space usage.
- **Secondary Scenarios:**

  3.1 User does not have any detected messes.

  3.1.1 System informs the user that there are no detected messes.
- **Success Postcondition:** User is able to view the current and pending cleanliness tasks of the shared living space. User is able to view detected messes and cleanliness tasks of other housemates.

S.4.16 Add ChatBot to Group Chat

- **Use Case:** UC6
- **Primary Actor:** User
- **Secondary Actor:** User's roommates, ChatBot
- **Precondition:** User is logged in and has an existing home.
- **Trigger:** User wants to recieve SMS messages regarding cleanliness, chores, expenses, and events in the shared living space in a group chat outside of the Room8 app.
- **Main Success Scenario:**

1. User navigates to the “Chat Management” section.
2. User selects the option to add a ChatBot to their house group chat.
3. User inputs notification and reminder settings.
4. System generates an SMS numbers and instructions to add the ChatBot to the group chat.
5. User adds the ChatBot to the group chat.
6. The ChatBot sends a welcome message to the group.

- **Secondary Scenarios:**

  5.1 Users fail to add the ChatBot to the group chat.

  5.1.1 System informs the user that the ChatBot has not been added.

- **Success Postcondition:** User is able to recieve SMS messages regarding cleanliness, chores, expenses, and events in the shared living space in a group chat outside of the Room8 app.

S.5 Prioritization

Requirement ID	Description	MoSCoW Classification	Reasoning
F211, FR212, FR213	Account Authentication	Must have	App features require users to be authenticated to use.
F214, FR215, FR216, FR217, FR218	House Management	Must have	App features all require users to be part of a home.
FR221	ChatBot Configuration	Could have	Allows users to customize notifications to their needs.
FR222, FR223, FR224	ChatBot Notification Messages	Should have	Allows users to conveniently recieve notifications in a shared setting.
FR231	Cleanliness Evaluation After Usage	Should have	Removes frequent point of friction in living spaces for users.
FR232	Cleanliness Tasks	Could have	Summarizes detected messes into a tasks.
FR233	Cleanliness Detected Messes	Should have	Very important to assigning responsibilities to users.
FR234, FR235	User Cleanliness and History	Could have	Provides history for user's performance.

Table 2: Requirement Specifications Part 1

Requirement ID	Description	MoSCoW Classification	Reasoning
FR241, FR242, FR243	Chore & Event Management	Must have	Scheduling chores is an essential feature in student housing.
FR244	Edit Chore & Event	Could have	Allows users update their schedule with ease.
FR245	Marking Chores Complete	Should have	Provides users with a way to track completed chores.
FR251, FR252, FR253, FR254	Add & View Split Bills	Must have	Essential feature in managing shared resources in student housing.
FR255	Calculate Least Transactions	Could have	Convenient feature which makes users lives easier.

Table 3: Requirement Specifications Part 2

S.6 Verification and acceptance criteria

Functional Requirements
In order to validate functional requirements, a systematic approach will be implemented. Features will be validated against criteria made by developers, testers and stakeholders. Using the criteria unit and intergration tests will be developed for consistency.

Non-Functional Requirements
Tests for User Account Mangement

- Account Creation Flow
  - A new user trying to create an account, when sign-up process is started they then should be able to enter details and create account with details being stored in secured manner. [NFR211].
- Error Handling For Incorrect Information
  - A user trying to login to an account, when login process is started and they make mistake, (eg. enter wrong password), no information pertaining to sensitive detail should be displayed. [NFR212].
- Account Authentication

- A user trying to login and authenticating their account should be able to have account authenticated must be done in less than 1 second. [NFR214].

Tests for Calendar Interactions

- Calendar Event Creation
  - A user who has logged on to the application will create an event then change the time 1 increment above the original time on the calendar and the system will store the event in UTC while the event will show up on the user's device in the user's respective timezone. [NFR241, NFR242, NFR243].

Project

Contents

P.1 Roles and personnel

Note: Roles that cannot be assigned to multiple people are denoted with a *

Role Switching: Roles can be switched and reassigned with an informal verbal agreement with the exception of Developer Liaison, which requires the previous liaison to notify all external contacts of the change in point of contact.

Project Manager*: Responsible for making sure Room8 project is operating smoothly. Ensures resources being allocated efficiently, issues on Github are being solved, project tasks are being tracked and chairs team meetings.

Application Developer: Responsible for developing the Room8 application which will be the main point of interaction between users and projects. Tasks this role is responsible for also include API design and UI/UX design.

DevOps Engineer: Responsible for creating and maintaining CI/CD pipelines to ensure the latests builds are deployed to the production environment.

Cloud Engineer: Responsible for creating designing, implementing, and managing any required cloud-based solutions needed for the project. These include cloud based storage containers and application deployments.

Hardware Specialist: Responsible for researching and implementing any required hardware solutions as well integrating them with the application and other software

components.

AI Engineer: Responsible for researching, implementing, and tuning any required AI models needed for implementation.

Developer Liaison*: The point of contact between project developers and stakeholders. The Liaison will be responsible for scheduling non-developer meetings and responding to questions from stakeholders.

P.2 Imposed technical choices

Due to time constraints, project requirements, and the skill sets of the team's current developers, some imposed technical choices are:

- **JS/TS Application Framework**. This is due to JS/TS being known by a majority of the developers and it having a plethora of existing, well-documented frameworks.
- **AWS Cloud Services**. AWS is the leading cloud platform and one which our developers have experience developing in.
- **Google Calendar and Google OAuth**. Google provides very generous free tiers for their OAuth and cloud solutions.

P.3 Schedule and milestones

- **Milestone 1 (September 23rd) - Initial concept and development plan**
  - Deliverable: Problem Statement and Goals
  - Deliverable: Development Plan
- **Milestone 2 (October 11) - Requirements**
  - Deliverable: SRS
- **Milestone 3 (October 23) - Hazard Analysis**
  - Deliverable: Hazard Analysis Report
- **Milestone 4 (November 1st) - V&V Plan**
  - Deliverable: V&V Plan
- **Milestone 5 (Within November 11-22) - PoC demonstration**
  - Event: Informal Project Demonstration

  - Goal: Core features functional

- **Milestone 6 (January 15) - Design Documents**
  - Deliverable: Software Architecture Document
  - Deliverable: Detailed Design Document
- **Milestone 7 (within February 3-14) - Revision 0 & Project completion**
  - Goal: Project is Complete and Functional
- **Milestone 8 (March 7th) - Verify & Validate**
  - Deliverable: V&V Report
  - Goal: Project meets all requirements outlined in the SRS and V&V Plan
- **Milestone 9 (within March 24-30) - Final demonstration**
  - Event: Final Project Demonstration
  - Goal: Unmet requirements from Milestone 8 are resolved
- **Milestone 10 (April) - Project EXPO demonstration**
  - Deliverable: Project Poster
  - Event: Project Expo
- **Milestone 11 (April 2nd) - Final documentation**
  - Deliverable: Final Problem Statement
  - Deliverable: Development Plan
  - Deliverable: PoC Plan
  - Deliverable: Updated Requirements document
  - Deliverable: V&V Plan
  - Deliverable: V&V Report
  - Deliverable: User Guide
  - Deliverable: Project Source Code

P.4 Tasks and deliverables

Milestone 1
Details:

- Abstract the problem to be solved and characterize it in terms of inputs and outputs.
- Research the importance of the problem and possible stakeholders.

- Establish measurable goals and selling points of the product.
- Discuss team availabilities, roles, and communication strategies.
- Discuss potential workflows, technologies, and coding styles.
- Establish plan and timeline for a PoC.

Expected Outcomes:

- Problem Statement and Goals.
- Development Plan.
- Team Charter.
- Git Repository and Online Project Board.

Milestone 2
Details:

- Revisit problem statement.
- Revisit project goals.
- Revisit stakeholders.
- Gather user requirements.
- Establish functional and non-functional requirements.
- Establish project use cases.
- Establish system constraints.
- Establish expected system environment and assumptions.

Expected Outcomes:

- SRS Document.

Milestone 3
Details:

- Explore possible data privacy issues in the project.
- Explore scenarios where user's provide incorrect input.
- Explore scenarios where the system isn't running in it's ideal environment.
- Explore hazard mitigation strategies for the above scenarios.

Expected Outcomes:

- Hazard Analysis Document.
- Improvement of project security, data privacy, and error handling.

Milestone 4
Details:

- Analyze features and determine appropriate test plan.
- Establish test plan for non-functional requirements.
- Establish metrics and success criteria for validation.
- Establish a timeline for V&V.
- Determine if manual testing is required.
- Trace tests to functional requirements.

Expected Outcomes:

- V&V Plan.
- Automated testing suite using CI/CD tools.

Milestone 5
Details:

- Establish scope of the PoC demonstration.
- Prepare presentation materials such as documents or slides. TBD based on demonstration format.
- Establish core use cases for the PoC demonstration.
- Rehearse PoC demonstration.

Expected Outcomes:

- Feedback from stakeholders.
- Revised project goals and priorities.
- Feasibility Validation.
- Identification of Risks.

Milestone 6
Details:

- Discuss and establish design to satisfy functional and non-functional requirements.

- Design system modules and architecture. Associate system concerns to system modules.
- Create test cases.

Expected Outcomes:

- Software Architecture Document.
- Detailed Design Document.

Milestone 7
Details:

- Identify tasks (issues) to be finished alongside their deadlines.
- Identify task ownership and distribute tasks.
- Write system code and build the software module.
- Create any custom hardware components required.
- Connect software module to the hardware (if required).
- Perform User Acceptance Tests.

Expected Outcomes:

- Functional Application.
- Any Necessary Hardware Components.

Milestone 8
Details:

- Apply the procedure outlined in the V&V plan.
- Create additional tests and testing code needed to verify unverified requirements.
- Document unexpected results.
- Document what requirements haven't been met and develop an actual plan to meet them.

Expected Outcomes:

- V&V Report.
- Resolution of any unmet requirements.

Milestone 9 & Milestone 10 - Project Demonstrations
Details:

- Finalize list of project selling points.
- Rehearse project demo and establish primary, secondary, and tertiary use cases.

Expected Outcomes:

- Project Poster.
- Project Showcase Video.

Milestone 11
Details:

- Revisit all previously created documentation and make any necessary corrections.
- Record steps needed to install and setup the system from a user perspective.

Expected Outcomes:

- Revised Project Documentation.
- User Guide.

P.5 Required technology elements

Camera
A camera is used for the systems cleanliness detection aspect. A camera takes pictures for comparing the state of the shared space.

Motion Sensor
A sensor is also used for the systems cleanliness detection aspect. The sensor detects movement which will work in unison with the camera to take pictures of the shared space.

Third Party Calendar API
Room8's scheduler requires a calendar component where user's can book and view events and chores in a calendar. Implementing a calendar API from scratch can prove difficult and there are already many existing solutions.

Secure Database
Users create accounts for Room8 where user and home information is stored on a secure database where sensitive information such as credentials are stored and individuals are able to change and delete accounts.

Third Party OAuth Service
Users will sign in securely without risk of data leaks and malicious connections.

SMS Message Service

Room8's ChatBot SMS will send messages to users using a third party messaging system that has connection to the Room8 application to send warnings and information of shared space.

P.6 Risks and mitigation analysis

Machine Learning Model Training
Finding enough data to train the machine learning algorithm to detect when a mess has been made and what detect what is altered or added in the space. Sensitivity to slight changes such as lighting is another risk with the machine learning algorithm approach. An alternative to adapt this scenario would be to use image differencing to detect changes in environment.

User Identification
Being able to detect which user is using the shared space through the camera is a major risk. The program may not be supplied with enough information to identity which user is currently using the space. In order to work around this the program could get individuals to submit to the application that they are now using the space.

Scope Creep
Unclear requirements and adding too many elements to the scope of the project will result in not delivering on features that are listed to be part of the application and lower quality of core functionality due to time being strain. A revisal of requirements by making requirements and functionality of system clear and concise and developing a prioritization list would mitigate the risk of scope creep.

P.7 Requirements process and report

Requirements Elicitation
For the elicitation of our requirements, we were able to discuss with our peers, who live in shared living situations, and took inspiration from our own experience as students who also have the need for an application like Room8.

Requirements Analysis
The analysis of the requirements was completed as part of the problem statement and goals. As a group, we took the goals that we created in that part of the project and extracted them as functional requirements. Additionally, there were non-functional requirements that were mistakenly added as goals in revision 0 of that document, which gave us a basis for the specification.

Requirements Specification
Evidently, the requirements specification are described in our Software Requirements Specification (this document itself). They will remain as the project continues and will be updated accordingly.

Requirements Validation
The requirements validation will take place at two different stages of the project. First, they will be validated at the point of the PoC demo, where we will determine if the scope of the project is appropriate and we can pivot (add/reduce requirements) at that point. Furthermore, the final requirements will be assessed when the project is complete and demonstrated at the end of the Winter semester.

Requirements Management
As previously mentioned, additional requirements will be made as necessary in this document and appended iteratively.

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning.

1. **What went well while writing this deliverable?**
   When writing this deliverable the formatting of the document went smoothly. With the example texts and functions made intially on the example .tex document we were able to format the document to the requirements given to ensure a readable document. Also, the sectioning and table of content makes the document easy to navigate and the sections are ordered in a way that makes sense. This way when looking for a section in the requirement it is easy to find. Seeing as this is a cruicial document which will be referred to multiple times throughout the project, the setup, and readability of the document is integral.

2. **What pain points did you experience during this deliverable, and how did you resolve them?**
   A major pain point of this deliverable was we did not know which SRS format to use as there were three different ones given. Initially we went with the SRS with no label (scientific computation) but then realized after checking lecture slides this was the wrong one. We then had to start over using the proper template. This caused us to fall behind schedule.

3. **How many of your requirements were inspired by speaking to your client(s) or their proxies (e.g. your peers, stakeholders, potential users)?**
   Almost half of our requirements were inspired by talking to peers and just personal experience from each of us. We fall under the group of stakeholders (students living off campus) and we choose to make this project from scratch because we have experienced the pain points covered in the requirements much like how our peers have as well. We did this through casual conversation with our classmates and friends alike.

4. **Which of the courses you have taken, or are currently taking, will help your team to be successful with your capstone project.**
   A course all of us have taken is SFWRENG 3RA3 Software Requirements course which is meant exactly for this deliverable. It went over key software requirement document components and helped us form this document. Another course that will help later down the line is COMPSCI 3ML3 which 4 of the 5 group members are taking. This will help in developing and learning about the machine learning algorithm.

5. **What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be re-**

lated to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

There are about 4 key concepts the team will collectively need to acquire in order to successfully complete the project. The first is knowledge of machine learning. This will help us train the model and learn how the algorithm will work. Another is software testing because this will help us test the system and see if it's working correctly and properly under edge cases. Data privacy is also important for our project because without protecting user information the project will not be viable.

6. **For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?**

   For all knowledge areas listed above, there are courses available at McMaster to learn these skills. However, because we can not take all these courses because of elective restrictions, there are textbooks and online texts and videos available covering these topics. This will be the go to for most group members however some group memebrs already have experience and knowledge in these domains whether its from previous work experiences or courses taken. We will teach each other and learn together and take advantage of each others strengths to ensure progress in each field.

Appendix — Citations

[1] "How to get along with your roommate," Centre for Innovation in Campus Mental Health, https://campusmentalhealth.ca/infosheets/how-to-get-along-with-your-roommate/ (accessed Oct. 11, 2024).