WeCasa

Business Requirements Document

Team HAGS JP

Team Lead:

Allison Austin

Team Members:

Githel Lynn Suico

Haley Nguyen

Joshua Quibin

Judy Li

Matthew Chung

https://github.com/githelsui/WeCasa

Date Submitted: December 06, 2022

BRD Version Table

Version	Description	Date

1.0	Initial Business Requirements
	- Project Overview
	  - Document Purpose
	  - Problem
	  - Solution
	  - Target Customers
	  - Competition
	- Scope
	- Features
	- System Failures
	- Glossary
	- References
		10/5/2022

1.1	Content Improvements
	- Scope
	  - Product-Wide Requirements
	- System Failures
	  - Formatting
		12/05/2022

1.2	Content Improvements - Removed implementation details - Claims-based Authorization - Added claims checks in Appendix	12/06/2022

Table Of Contents

Project Overview

Document Purpose

The purpose of the Business Requirement Documents (BRD) is to gather all business requirements and criterias for success necessary to build WeCasa as per client's needs and requirements. This document focuses on defining the failures, successes, and specifications of WeCasa's system and features, as well as scenarios and solutions for failures. This document should be referenced in relation to the high level design, and project plan.

Problem

Roommates do not have a designated one-stop app that can manage all the necessary organization required in their shared housing situation, resulting in problems of stress and tension due to a lack of coordination and miscommunication.

Solution

The value of our app is its ability to effectively solve the mentioned pain points that arise from living with roommates.

As a solution to these problems, our app is a dependable toolbox for organizing all aspects of shared housing management to relieve the stress and tension of living with roommates.

Target Customers

WeCasa is designed to make the entire roommate experience go as seamlessly as possible. Anyone from students, to working professionals, to even family members can utilize the app to help streamline daily tasks. Whether or not one feels like their roommates work well with each other, our app will help strengthen weak points in communication. The app is designed for anyone with roommates, but it is especially helpful to those who feel like they live in a disorganized home and could use a third-party to help with that.

Pain Points

Several core problems and issues exist for those living with roommates that our app will address:

- **Disorganization:** Roommates often share lists of chores, maintenance dates, and many other items to be managed. For example, it can be difficult to manage whose turn it is to do a specific chore or when to complete it by.
- **Miscommunication**: Roommates will create assumptions that may result in future disagreements unless properly handled with transparency. Therefore, the app's features aim to encourage compromise and communication between roommates.
- **Aversion to Confrontation:** Roommates will run into disagreements and have trouble effectively mitigating the situation, due to an ability to initiate the confrontation. The app will have features that incentivize users to initiate confrontation.
- **Burden of Managing Multiple Apps:** Several task and home management apps exist on the market to help roommates. However, it is overwhelming for the user to sign up and keep track of multiple accounts.

Competition

One of the primary goals of our app is to combine features of several existing apps into one. Our main competitors consist of applications that target both organization and communication within a group. WeCasa derives its core components from a variety of these apps while also providing distinct, original features to its users.

Home Organization Apps

Include gamified chore or task management features for families living under the same roof. Examples of these apps include OurHome, Nipto, and RoomMate. The target audience for these apps is typically children, and the apps focus on incentivization through a reward system. OurCasa targets adults, and does not use a reward or point system to motivate its users.

Social Media Groups/Community Apps

Include group forum or chat room applications that allow users to share information, content, and/or events. Examples of these apps include Facebook Groups, WhatsApp, GroupMe, Slack, and Discord. While the target audience of these apps

intersect with WeCasa, these apps focus on communication rather than organization.

Task Management Apps

Include project management, organization, and note-taking software. Examples of these apps include Trello and Notion. While the target audience for these apps intersect with WeCasa, these apps aim to reach the general audience, rather than being tailored to shared living spaces. Our app is created specifically with roommates in mind, which allows our features to be more defined and useful. Additionally, these apps typically charge a premium to unlock key features and storage, and we hope to provide a solution at no cost.

Scope

Our single-page-application is targeted initially to the US market. As such, it will only support American English at launch and comply with California privacy laws. And as our app grows, we would like to expand this to other markets such as Canada and Europe. However, this will not initially be supported.

Furthermore, our product will follow the Web Content Accessibility Guidelines (*WCAG*, 2022) in order to make our application's content more accessible to our audience.

Product-Wide Requirements

Functional

- **Browser:** The primary browser that will allow users to access the full functionality of our app will be Chrome Version 104.X and above. We chose to target Chrome because this is the most popular browser (Oberlo, 2022). W3C validation will allow our app to be accessible on smaller screens such as mobile devices. (LoginRadius, 2020).

Non-Functional

- **Accessibility:** Our app will utilize the Web Content Accessibility Guidelines (WCAG) as the set of standards for digital accessibility (*WCAG*, 2022).

WeCasa will provide the following accessibility features in all versions of the application:

- Alternative text for graphics and media
- Clear form labels
- Meeting a color contrast ratio of at least 4:5:1 for normal text and 3:1 for large text (18 point or larger, or 14-18 point and bolded)
- Disabling key traps

Authorization

WeCasa will implement claims-based authorization, which will require resources to check the claim value of the user requesting the resource. Access to resources will be based on whether the user possesses a claim with the correct value required by the resource's security policy. See Appendix A for the complete list of claims checks.

Types of Users

System Administrator

Has full read and write permissions for all users. This user has the ability to disable/enable accounts, recover accounts, and monitor the logs and analytics dashboard.

Generic User

Any user that is not the system administrator. The 3 permission levels are in effect for these users, and this type of user is created with the Account Creation feature (see User Management).

Features

Budget Sharing Bar/Bill List

- **Description**: A budgeting system that allows roommates to keep track of shared expenses and balances.
- **Functional Requirements**
  - The budget bar functionalities:
    - The budget bar will be variable with no set budget for each month.

        - Budget values have no upper-bound limit, however, progression towards the budget may not exceed the defined budget resulting in a negative budget readout.
        - The bill list is visible when the user interacts with the bar.
        - The number of sections are determined by the number of users in the group. Each section will be color coordinated with predefined colors.
    - The Bill List will show the following information:
        - The date
        - The bill name
        - The bill reporter
        - The bill amount
        - The bill status - 'UNPAID', 'PAID'
        - A receipt icon (if the user uploaded a receipt)
    - The Bill List enables users to:
        - Add a bill by prompting the user to:
            - (Required User Input) Inputting the name of bill
                - Valid Input: a-z, A-Z, 0-9, 60 characters limit
            - (Optional User Input) Input a description
                - Valid Input: a-z, A-Z, 0-9, 2000 characters limit
            - (Required User Input) Inputting how much is the bill
                - Based on U.S. monetary system (decimal system)
                - Valid Input: Floating point values only ($X.XX)
            - (Optional User Input) Checking a box if the bill is to be repeated in the next period
                - Valid input: Boolean value only
            - (Required User Input) Select the member to split the bill with. After which an input field will appear under each member prompting the user for a percentage. If no other member is chosen, the reporter is assigned 100% of the bill. The bill will be calculated, split accordingly and updated each member's budget bar and transaction list upon refresh.
                - Valid Input: Floating point values only for percentage input
            - (Optional User Input) Upload a receipt.
                - Valid Input: .PNG, .JPEG, .HEIF

        - Update the status of payment.
        - View uploaded receipts.
        - View information related to uploaded bills. Pre-existing bills can only be edited by the user who initially reported the bill.
    - Only users with an active authentication session and access to the space can utilize this feature.
    - Changes to the budget sharing bar and adding bills to the transaction list will be logged to a database in the form of activity and/or errors (if any).
- **Non-functional Requirements**
    - When user wants to add a bill, a form appears within 3 seconds
    - When the user submits the form, the budget bar and the transaction list should update within 3 seconds.
    - The information should be recorded in the bills database within 5 seconds.
    - Deleting and editing a bill will present a confirmation prompt to the user, “Are you sure you want to edit/delete this bill”, followed by the text options of ‘YES’ and ‘NO’
- **Preconditions**
    1. The user must have an active authenticated session (logged in).
    2. The user must have access to the shared space they want to edit.
- **Successful Postcondition**
    1. All transactions reflect in the bar and the transactions list accurately and within the maximum allotted time.
    2. All submitted information should be recorded in the bills database.
- **Failure Scenarios**
    1. The form doesn’t appear when requested.
    2. The user abandons the form and exits.
    3. The budget bar doesn’t update after a bill is submitted.
    4. The transaction list doesn’t update after a bill is submitted
    5. The budget bar doesn’t refresh on the 1st of every month.
- **Failure Handling:**
    1. If a component of the budget bar is not successfully loaded, then the bar will appear gray with a message saying that there was a problem loading the budget bar.
    2. If the user exits the form, data will not be recorded in the database.

3. If the transaction list is not successfully updated, the user will not be notified but the error will be logged.
4. If the form can't be loaded, an alert will appear notifying the user of the failure.
5. All failures will be logged.

Bulletin Board

- **Description**
    - A space for roommates to pin common information (e.g., Wi-Fi name and password).
- **Functional Requirements**
    - All members of the group can edit, add or delete sticky notes on the board.
    - Users can add a sticky note on the bulletin board
    - Users can also delete sticky notes
        - The app will ask for confirmation before deleting the sticky note
    - Users will be able to undo the latest note deletion
    - The user will be able to add the following to a sticky note:
        - (Optional User Input) Text
            - Valid input: String limited to 500 characters
        - (Optional User Input) Photos
            - Valid input: Compatible image file types: .PNG, .JPEG, .HEIF
    - Only users with an active authentication session and access to the space can utilize this feature.
    - Changes to the bulletin board will be logged to a database in the form of activity and/or errors (if any).
- **Non-functional Requirements**
    - Bulletin Board view will load within 3 seconds
    - New bullet points will buffer within 3 seconds
    - Deletion of bullet points will buffer within 3 seconds
    - Reverting a deleted bullet point will buffer within 3 seconds
- **Preconditions**
    1. The user must have an active authenticated session (logged in).
    2. The user must have access to the shared space they want to edit
- **Successful Postconditions**
    1. The Bulletin Board is viewable.

2. The user can edit the contents of the board.
3. The board can hold text and images.

- **Failure Scenarios**
    1. The user exceeds the character limit.
    2. The user attempts to add invalid image type.
- **Failure Handling:**
    1. The user is asked to stay within character limit.
    2. The user is asked to only upload valid image types.

Calendar

- **Description:** A visual representation of scheduled events. This is especially handy for trash pickup day, rent due dates and appliance maintenance dates.
- **Functional Requirements**
    - The Calendar will display the current month.
    - Calendar view can be changed from month, week, and day.
    - The current day will be outlined and its tasks will be displayed chronologically on a sidebar by default
    - Events on a particular date can be shown.
    - Upon creating a new event, the user will be prompted to:
        - (Required User Input) Input a name for the event
            - Valid input: a-z, A-Z, 0-9, 60 characters limit
        - (Optional User Input) to give an event a color.
            - Valid input: Blue, Orange, Yellow, Red, Green
        - (Optional User Input) Input a time of occurrence for the event
            - Valid input: Integer values only
        - (Optional User Input) Check if event will be repeated daily, weekly, annually
            - Valid input: Categorical array
        - (Optional User Input) Check if the event will be private or public (defaults to public)
            - Valid input: Boolean value only
        - (Optional User Input) Input a time before the event for reminders via email
            - Valid input: Integer values only, must be larger than current date
        - (Optional User Input) Input description for event
            - Valid input: String limited to 250 characters

    - Events can be created, edited, and deleted
    - Changes to the calendar and adding events will be logged to a database in the form of activity and/or errors (if any).
- **Non-functional Requirements**
    - The user should easily be able to navigate to the Calendar view.
    - The user should easily be able to add an event.
    - The calendar view should load within a maximum of 3 seconds.
- **Preconditions**
    1. The user must be logged in to their own account.
    2. The user must be in the Calendar view.
- **Success Postconditions**
    1. If a user creates/edit an event, an event update should be seen by allowed parties.
    2. The user receives a reminder in accordance with their reminder time selection.
- **Failure Scenarios**
    1. The calendar view does not load.
    2. The user is unable to create/update an event.
    3. The user does not receive a message notification reminder.
    4. The event is updated/created, however other users cannot view.
    5. The user abruptly leaves the system while creating/updating events.
- **Failure Handling**
    1. The user is notified of an error and is suggested to contact Support.
    2. The system aborts operation.

Chore List

- **Description:** A task board to split up chores amongst housemates.
- **Functional Requirements**
    - The chore list will display as a weekly list of chores separated by dates. This list will start on Monday.
    - The user can select between Current To-Do's and To-Do History views on the page for their viewing options.
    - The user can enter a new chore into the list with the following required inputs:
        - (Required User Input) Input chore name
            - Valid input: a-z, A-Z, 0-9, 60 character limit
        - (Required User Input)Input time to reset

                - Valid input: Integer values only, must be larger than current date
            - (Optional User Input) Input chore notes
                - Valid input: String limited to 250 characters
            - (Optional User Input) Input assignments to a specific person
                - Valid input: Array value only
            - (Optional User Input) Check if the chore is repeated
                - Valid input: Boolean value only
        - The user can mark a task as completed.
        - If a chore isn't done by the due date, it is added to the assigned user's next chore list.
        - Chores are unassigned if someone is uninvited from the group.
        - Every week, an even number of tasks are randomly split between each user.
        - Changes to the chore list will be logged to a database in the form of activity and/or errors (if any).
- **Non-functional Requirements**
    - Users should easily navigate to Chore List View.
    - Chore List View will be loaded within 3 seconds.
    - List of chores per day should be loaded after accessing the page.
- **Preconditions**
    1. The user must be logged in to their own account.
    2. The user must be at the Chore List View.
- **Successful Postconditions**
    1. Interacting with the 'TO-DO' section will display a list of current chores.
    2. Interacting with the 'History' section will display a list of completed chores.
    3. When a user checks a task as completed:
        - The task will be removed from TO-DO and added to History
        - The color bar will reflect its status and change from red to green (green for complete, red for incomplete)
- **Failure Scenarios**
    1. The user abruptly exits from saving a new chore.
    2. The user attempts to leave required fields empty.
    3. The user exceeds the maximum character limit for a new chore name.
- **Failure Handling**
    1. System aborts operation.

2. The user is notified to input all required fields and try again.
3. The user is notified to try again with limited characters.

Grocery List

- **Description:** A shared list designed to efficiently complete grocery shopping.
- **Functional Requirements**
    - A user has the ability to add grocery item with the following inputs:
        - (Required User Input) Input grocery item name
            - Valid input: a-z, A-Z, 0-9, 60 character limit
        - (Optional User Input) Input Optional notes
            - Valid input: String limited to 250 characters
        - (Optional User Input) Input assignments to a specific person
            - Valid input: Array value only
    - The user is able to delete an item on the list that has already been purchased
    - Clearing the list will clear all crossed and uncrossed items
    - Items can be deleted from the list
    - The list instantly syncs up changes across all user accounts.
    - Changes to the grocery list will be logged to a database in the form of activity and/or errors (if any).
- **Non-functional Requirements**
    - Users should easily navigate to the Grocery List View.
    - Grocery List View will be loaded within 3 seconds.
    - List of grocery items should be loaded after accessing the page.
- **Preconditions**
    1. The user must be logged in to their own account.
    2. The user must be at the Grocery List View.
- **Success Postconditions**
    1. The user is able to add grocery item name
    2. The user is able to delete a grocery item
        - The deleted item is visible in the list, but is crossed out
- **Failure Scenarios**
    1. The user abruptly exits from saving a new grocery item.
    2. The user attempts to leave required fields empty.
    3. The user exceeds the maximum character limit for the new grocery item name.
- **Failure Handling**

1. System aborts operation.
2. The user is notified to input all required fields and try again.
3. The user is notified to try again with limited characters.

Reminders

- **Description:** An easy way to get notified of scheduled events labeled on the calendar.
- **Functional Requirements**
    - If users want reminders about an event, they would configure that setting when creating/editing the event.
    - Reminders will be sent via email.
    - Adding or changing reminders will be logged to a database in the form of activity and/or errors (if any).
- **Non-functional Requirements**
    - Reminders must be sent at the correct time the user has scheduled it for.
- **Preconditions**
    1. To configure reminders, users must be logged into their own accounts.
    2. To receive reminders, users do not have to be logged in.
    3. The user must have access to the space to receive reminders for that space.
    4. The user must have configured reminders to true when creating/editing events.
    5. Reminders will be configured to true as default.
- **Success Postconditions**
    1. The user receives an email at the scheduled time they configured the reminder for.
- **Failure Scenarios**
    1. Reminder has not been sent via email at the scheduled time it has been configured for.
- **Failure Handling**
    1. Users can report this failure in the User Feedback feature.

Nudging

- **Description**
    - A friendly reminder sent to roommates who haven't completed their assigned tasks in time by another user.

- **Functional Requirements**
    - The nudging option will be available on all task cards other than the current user's.
    - Users can 'nudge' a roommate, sending the assignee a friendly notification to complete that task.
    - The user will have pre-set messages options to send to the assignee.
    - Assignee will be notified through email and once they login.
    - Only users with an active authentication session and access to the space can utilize this feature.
        - Users do not have to be logged in to receive a Nudge.
    - Sending a nudge will be logged to a database in the form of activity and/or errors (if any).
- **Non-functional Requirements**
    - The assignee should receive their nudge within a minute of being sent.
    - Users can only nudge one assignee per day.
- **Preconditions**
    1. To send a nudge, the user must be logged in and have access to a space.
    2. Users do not have to be logged in to receive a nudge.
    3. The user must be in the Chore List view.
    4. The task being nudged must be incomplete.
- **Success Postconditions**
    1. The user should have a confirmation that the nudge is sent.
    2. Assignee should have an email and be notified of the nudge when they log in.
- **Failure Scenarios**
    1. Assignee is not notified of a nudge via email or when they log in.
    2. Nudge option is disabled even though the task is incomplete.
- **Failure Handling**
    1. The user should be notified that nudge was not sent and of further instructions to retry or contact Support.
    2. The user should be notified of the reason and to contact Support.

Incomplete Task Summary

- **Description:** A summary of incomplete tasks is automatically generated and sent to all roommates via text/email notification.
- **Functional Requirements**

    - Comprehensive list of users with incomplete tasks from the past week (Monday - Sunday) is compiled.
    - Comprehensive list of incomplete tasks from the past week (Monday - Sunday) is compiled.
    - Only users with an active authentication session and access to the space can receive this notification and/or edit this feature.
        - Users do not have to be logged in to receive the Incomplete Task Summary.
    - All users within a space should receive the Incomplete Task Summary.
    - This feature will be logged to a database in the form of errors (if any).
- **Non-functional Requirements**
    - Notification will be sent at 12AM (UTC) every Sunday, any tasks left incomplete will appear in the summary.
- **Preconditions**
    1. The user must have access to a space.
    2. Tasks from the shared chore list must be left incomplete by the end of the week.
- **Successful Postconditions**
    1. The notification is sent successfully.
    2. All incomplete tasks appear in summary.
- **Failure Scenarios**
    1. Notification fails to send, prompting the notification to be resent.
    2. This is an automatic notification that is sent out at the end of every week.
    3. It will include a list of users that haven't finished their tasks.
    4. A list of the unfinished tasks will be included under each user's name.

Circular Progress Bar

- **Description**: This feature reflects the percentage of assigned chores a user completes onto a progress bar.
- **Functional Requirements**
    - If a user doesn't complete their tasks by the due date, it will negatively impact their scores on this bar:
      $\#\ of\ incomplete\ tasks/(\#\ of\ incomplete\ tasks\ +\ \#\ of\ complete\ tasks)$.
    - If the user completes their tasks, the bar will progress:
      $\#\ of\ complete\ tasks/(\#\ of\ incomplete\ tasks\ +\ \#\ of\ complete\ tasks)$.

    - The scores will be updated whenever a task is complete or a task is overdue.
    - The bar will refresh on the first of every month.
    - Changes to the tracking bar will be logged to a database in the form of activity and/or errors (if any).
- **Non-functional Requirements**
    - The bar updates within 3 seconds when a user marks a chore as complete or if a chore becomes overdue.
- **Preconditions**
    1. The user must be logged in to their own account.
    2. The user must be in the Circular Progress Bar view.
- **Successful Postconditions**
    1. The bar recalculates and updates when a user marks a chore as complete or if a chore becomes overdue.
    2. The bar refreshes visually every month.
    3. The status of the bar and any changes made to the bar are successfully logged into the appropriate database.
- **Failure Scenarios**
    1. The bar doesn’t update visually when a chore is marked complete, or a chore becomes overdue.
    2. The bar doesn't refresh visually every month.
    3. The bar status and changes to the bar are not logged to the appropriate database.
- **Failure Handling**
    1. Errors will be logged and the system administrators will be notified.

Photo/Documentation Uploads

- **Description:** Users can upload files such as lease documentation, rental agreements, property photos, receipts, etc.
- **Functional Requirements**
    - Users have the ability to upload a new file from their device
    - Users are able to configure valid file types:
        - Photo File Type: PNG, JPEG, and GIF
        - Document File Type: PDF, DOC, DOCX, HTML, TXT
    - Users are able to configure valid file size limit
        - 10 MB File Size Limit

    - Users are able to delete a file uploaded to the storage only if they uploaded the file.
    - Files can be retrieved and previewed by all users.
    - Files can be downloaded from storage to a user's device.
    - File storage syncs with any changes across all user accounts.
    - Only the user who uploaded the file has permission to delete it
    - Adding/removing a photo/document will be logged to a database in the form of activity and/or errors (if any).
- **Non-functional Requirements**
    - Storage will load within 10 seconds.
    - File previews will load within 10 seconds.
- **Preconditions**
    1. The user must be logged in to their account.
    2. The user must have access to a space.
    3. The user must be in the file upload view.
    4. Uploaded file does not exceed storage space limit/file upload limit
        a. Storage Limit: 15 GB
        b. File Upload Limit: 10 MB
    5. Uploaded file is of valid file type
        a. Photo File Type: PNG, JPEG, and GIF
        b. Document File Type: PDF, DOC, DOCX, HTML, TXT
- **Successful Postconditions**
    1. File is uploaded to storage.
    2. Files are accessible by any user within the group.
    3. When a user uploads a file, the activity is logged into a database.
- **Failure Scenarios**
    1. File cannot be uploaded due to invalid file type.
    2. File cannot be uploaded due to invalid file size.
    3. File cannot be uploaded due to insufficient storage space.
    4. File cannot be retrieved from database due to connection error.
    5. User does not have permission to access the file.
- **Failure Handling**
    1. File upload is aborted .
    2. User is prompted with an error message based on failure scenario:
        a. Error: Invalid File Type
        b. Error: Invalide File Size
        c. Error: Insufficient Storage Space

d. Error: Connection lost! Please try again later
e. Error: Permission denied

Login

- **Description:** A mechanism for identifying a valid user of the system.
- **Functional Requirements**
    - System shall validate the user.
    - A maximum of 5 failed authentication attemptions within 24 hours is allowed for the same account before the account is disabled. temporarily for 24 hours or until the user recovers the password.
    - After 24 hours, fail count resets to 0.
    - Account recovery must be performed by the Systems Administrator.
    - Every failed authentication attempt will be logged into a secure database.
- **Non-functional Requirements**
    - Authentication must take a maximum of 3 seconds.
    - After successfully logging in, the user must be directed to the home page in a maximum of 3 seconds.
    - Data is secure and will not be leaked during the authentication process.
    - In case of any failure to authenticate the user, the user must be notified of the reason
- **Required Input:**
    - The user must provide a valid security credential whenever attempting to authenticate with the system
        - Valid usernames must meet the character requirements
            - Minimum of 8 characters
            - a-z
            - 0-9
            - (Optional) The following special characters: . - @
        - Valid passwords must meet the character requirements
            - Minimum of 8 characters
            - One of each of the following characters:
                - a-z
                - A-Z
                - 0-9
                - .,@!-

- **Preconditions:**
    1. The user must be in the Login view.
    2. The user must have an existing account.
- **Success Postconditions**
    1. The user successfully logs in
    2. The user is redirected to the user's home page.
    3. The user is notified of invalid email/password with every failed login attempt.
    4. The user is notified of the last attempt before the account is disabled and locked.
    5. If the user's account is disabled, the user is notified to contact the Support for account recovery.
- **Failure Scenarios**
    1. The user abruptly exits from the authentication process.
    2. The user leaves any of the required fields empty.
    3. The user inputs invalid username.
    4. The user input invalid password.
    5. The user is not notified of invalid login attempt.
    6. The user is not notified of disabled account and to contact Support.
    7. Failed attempts are not logged by the system.
    8. The user successfully inputs valid credentials but is still unable to log in.
- **Failure Handling**
    1. The user aborts authentication.
    2. The user is notified to input all required fields.
    3. The user is notified of invalid email/username/password and to follow the proper guidelines.
    4. The user is notified to contact Support.

Logout

- **Description:** This feature allows users to securely log out of their accounts on one device.
- **Functional Requirements:**
    - System shall log the user out of the system.
    - The current logged in user session will end after 5 seconds of invoking the logout feature.
    - The user will be redirected to the system's home page after logging out.
    - All logouts will be logged to a secure database.

    - System failure from invoking the logout feature will not cause the system to go offline.
- **Non-functional Requirements:**
    - Logout process must take a maximum of 5 seconds.
    - Redirection to the system home page must take a maximum of 3 seconds.
- **Required Input:**
    - The user selects the Logout option.
- **Preconditions:**
    1. The user must have an existing account.
    2. The user must be logged in.
- **Successful Postcondition:**
    1. A message is displayed to the user to confirm whether they are sure to logout.
    2. The user is logged out and the active session ends.
    3. The user is redirected to the system's home page.
    4. A confirmation message is displayed stating that they have successfully logged out.
- **Failure Scenarios:**
    1. The user abruptly exits from the logout request.
    2. The user is not logged out and the session remains active.
    3. Confirmation message is not displayed to the user.
    4. The user is not redirected to the system's home page.
    5. Logout is not logged by system.
- **Failure Handling:**
    1. The System aborts the request.
    2. The user will be notified of error and further instructions on how to proceed.

User Access Control

- **Description:** This feature allows members to control the read and edit permissions for each space. This feature will implement 3 different levels of permission for read and edit access to certain features.
- **Functional Requirements**
    - The system successfully allows users to access/edit data within their permission level.

    - The system successfully bars users from accessing/editing data above their permission level.
    - To add users to a space, a member of the shared space can generate a link to join the space. This link can be shared with members that want to join. This link expires after 24 hours or after it is used (whichever happens first).
    - To remove users from a group, members of a space can select the user they want to remove from the member list.
        - This user will no longer have read or edit access to the space, and they will no longer receive notifications from that space.
        - Members that are removed will not receive a notification that they have been removed from the space.
    - All changes to user permissions and members of a space will be logged to a database in the form of user activity or errors (if any). Any errors in generating links will also be logged.
- **Non-functional Requirements**
    - Since this feature deals with sensitive user information, this feature must be resilient against security attacks.
- **Preconditions:**
    1. The user must have an active authenticated session (logged in).
    2. The user must have access to the shared space they want to edit.
- **Successful Postcondition:**
    1. All elements of the user's permission level are visible and editable
    2. Elements not included in the user's permission level or not accessible in the application, and therefore, are not editable.
- **Failure Scenarios:**
    1. System fails to withhold content/external user information from user
    2. Unauthorized user has access to a space
- **Failure Handling:**
    1. If any successful postconditions fail, the affected users will be notified via email of the failure scenario.

User Management

- **Description:** This feature allows users to manage their account information. customize their group(s), and configure the application settings for WeCasa.
- **Functional Requirements**
    - *Account Creation*

        - For registering an account with WeCasa, users can provide an email address and a password to use when logging into their account.
            - This email address is verified with an imported library.
            - A one-time password will be used.
                - This string of characters will be displayed to the user to use once when they first log in.
                - This one-time password will be associated with the user until it is used. After it is used, the user will be prompted to enter their own password for subsequent use. A user is not “enabled” until they log in with their one-time password.
    - *Account Deletion*
        - The user can selects an option to delete their account in Settings
        - All data (excluding logs) of this user will be removed from the system database.
    - *Account Recovery*
        - To reset a password, users can choose a “forgot password” option from the Login window to receive an email with a secure link.
        - This link will navigate the user to a window to reset their password.
        - After resetting their password, WeCasa will log the user out on all devices.
    - *Updating Profile*
        - When logged in, users can change their public or private information (see glossary for more details).
    - *App Settings*
        - To configure app settings, users can navigate to the settings option.
        - Within this window, users can choose how they want to receive notifications and/or nudges from WeCasa (email/SMS).
    - *Creating Groups*
        - Any user logged in can create a group from the home page.
        - Once created using default settings, a link will be generated that will allow the user to share the space with others (see UAC).
    - *Customizing Groups*

        - Users with access to a shared space can modify the group name, group members, and/or the functionalities or features of a shared space.
        - A user must be logged in and viewing the shared space in the application to make edits.
        - Only users with access to the space can customize groups.
    - *Deleting Groups*
        - The user who created the space can delete a group/space.
        - To delete a group, the user who created the space can navigate to the members list and select “delete group” from the list of options.
        - This will prompt a message saying that all files, settings, and notifications for this space will be deleted, and asks if the user would like to proceed.
        - If the user chooses to proceed, all data (except for logs) related to this space will be deleted from the database, and the space (along with all members) will no longer be visible to the users.
        - Users within a space will not be notified if the creator of the space deletes the group.
    - With the exception of creating or recovering an account, users must be logged in to utilize the above features.
- **Non-functional Requirements**
    - When the user goes to edit a property, a form appears within 3 seconds.
    - Password recovery link should appear in the user’s email inbox within 1 minute.
    - When the user submits the form, the budget bar and the transaction list should update within 3 seconds.
    - The information should be recorded in the logging database within 5 seconds. Logs for deleted groups will be maintained.
- **Preconditions:**
    1. The user must have a version of the app installed on their device.
    2. For managing an existing account and/or configuring app settings, the user must have an active authenticated session (logged in).
    3. For customizing a group, the user must have access to the shared space they want to edit.
- **Successful Postcondition**

1. All modifications are reflected accurately and consistently within the application for every user in a shared space.
2. Changes to personal user information are reflected accurately and consistently.
3. Changes to notification settings should be carried out and sustained.
4. All submitted information should be recorded in the appropriate database.

- **Failure Scenarios**
    1. The user creates an account, but the email used is not associated with a WeCasa account.
    2. A user's information does not get updated.
    3. The popup form to edit a property does not appear.
    4. The user abandons the form and exits.
    5. Group configurations do not update for every user.
    6. The password recovery email does not send.
- **Failure Handling:**
    1. If the user exits any of the edit windows, no updates will be made to the database.
    2. If any of the mentioned functionalities do not execute successfully, a message will be shown to the user that there was an issue and suggest the user submit feedback.
    3. If the email does not appear in the user's inbox and a message does not appear saying there was an error, the user can submit feedback to the team through the user feedback feature.

Logging

- **Description:** This feature allows the system to log normal user operations, system failures and exceptions into a secure database that only systems administrators can access.
- **Functional Requirements**
    - The system successfully writes logs to the database.
    - The system administrator will be notified after 150 error logs.
    - Only the system administrator has permission to edit (CRUD) logs.
    - Logging occurs:
        - After all requests to the server.
        - When users login and logout unsuccessfully (wrong login information).

        - During events that can potentially become an error (exceed time limits, databases nearing capacity).
        - After every error condition (failed API calls, internal errors).
    - Log Archive will consist of:
        - User information (username, device).
        - Group information (group name).
        - Activity (features that the user interacts with or requests data from).
        - Timestamps (UTC timestamp that the user interacted with that activity).
        - Inactivity frame (time since last activity).
        - Source of error (line of code).
        - Any execution errors (feature, time of error).
- **Non-functional Requirements**
    - Logs should write to the database within 3 seconds. If a log fails to show up in the database, it will try again in an hour. This will be attempted 2 more times.
    - Database queries should return a result within 3 seconds.
    - Logs will be automatically deleted after 2 years.
    - The system administrator will be notified after 150 error logs within 5 seconds.
    - Edits (CRUD) to the log archive should register within 3 seconds.
    - Logs will not include PII (Personal identifiable Information).
- **Preconditions:**
    1. A secure database should exist for logging purposes only.
    2. The database must be persistent and active.
- **Successful Postcondition**
    1. All Logging entries must be present. No logs should be missing.
    2. Logging entries being saved to the data store must be accurate.
    3. All queries to the database should return all logs within the specified fields.
- **Failure Scenarios:**
    1. Logs are missing from the database.
    2. Logs contain missing or incorrect information.
    3. Logs aren't deleted after 2 years.
    4. The log cannot be queried or returned results are wrong.
- **Failure Handling:**

1. System Administrators should be notified in case of any logging failures.

User Feedback

- **Description:** This feature allows users to communicate issues or positive feedback with the WeCasa team.
- **Functional Requirements**
    - An option to submit feedback is visible to the user.
    - When prompted, a form will appear where the user can type a message.
    - Responses from users are sent to the WeCasa team's email.
    - All responses from the team will be followed up within one week asking users how their customer service experience was.
    - Feedback submission will be logged to a database in the form of activity and/or errors (if any).
    - Satisfaction scores obtained through follow up emails are used for monitoring and assessing WeCasa customer service.
- **Non-functional Requirements**
    - The feedback option must be easily discoverable.
    - The popup window appears within 2 seconds.
    - The message written by the user is no more than 200 characters in length.
    - Once the user selects "submit", the message should indicate a successful/unsuccessful submission within 3 seconds.
    - Response to user's feedback appears in the user's inbox within 1 minute.
- **Required Input:**
    - The user must navigate and select the "feedback" option
    - The user must provide a typed message
        - Valid messages will consist of the following:
            - Must be at minimum 8 characters
            - A-z, A-Z
            - 0-9
            - !@#$%^&*()
- **Successful Postcondition**

    - Once the feedback is submitted, a note will appear indicating that the feedback was submitted successfully or unsuccessfully and the feedback window closes.
    - The feedback is sent to the WeCasa team email and the message appears in full.
    - Replies to the user's feedback message are sent to the email associated with the user's WeCasa account.
- **Failure Scenarios**
    1) The user abandons the form and exits.
    2) User feedback does not appear in the WeCasa team email.
    3) Response to user feedback does not appear in the user's inbox.
    4) The feedback window does not restrict the user from typing a message greater than 200 characters.
    5) The user's feedback is not logged into the database.
- **Failure Handling**
    - If the user exits the message window without hitting submit, no message will be sent to the team.
    - If the Logging feature indicates that a user has submitted feedback, but there is no email in the WeCasa inbox from that user at the time the submission was logged, the WeCasa team will reach out to the user and have them message the team directly.

Analytics

- **Description:** This feature will be used by the Business/Development team to lead future work on WeCasa. This summary includes various metrics related to the usage of the app, and provides insight on what features to expand/condense, overall engagement, retention, customer service, and overall app growth.
- **Functional Requirements**
    - All metrics are visible in the summary
        - DAU (Daily Active Users)
        - SPU (Standard Product Unit)
        - Most Used Features
        - Error Rate
        - Referral Rate
        - Retention Rate

        - Customer service ratings
        - User ratings and reviews
    - Dashboard is view only and is meant for staff to take insight on app usage.
    - Dashboard gathers aggregated data from the logging database to display metrics to the user.
        - Metrics are displayed either graphically or with charts.
        - Certain data can be shown with specific time frames.
- **Non-functional Requirements**
    - The data will be displayed in an easily viewable manner.
        - Color coded
        - Easily readable
    - Data will be displayed within 3 seconds.
- **Preconditions:**
    1. Users must have valid credentials to access WeCasa staff analytics dashboard.
- **Successful Postcondition**
    1. Dashboard is viewable by system administrator.
- **Failure Scenarios**
    1. Analytics are not being saved into the database.
    2. Dashboard metrics do not load within 3 seconds.
    3. Dashboard feature is not visible to system administrators and/or it is visible to generic users.
- **Failure Handling**
    1. Error notifications will ping if any analytics are not properly documented.
    2. All errors will be properly documented.

System Failure

- **Description:** The following section details how system failure scenarios will be handled.
- **Functional Requirements:**
    - The user must be shown an error page.
    - The user must be notified of the reason behind the system failure, and a follow up action should be recommended through the error page.
- **Non-Functional Requirements:**

    - The error given to the user must be given in an user friendly manner.
    - The error page must be displayed within 3 seconds.
- **Preconditions:**
    1. User is logged in and is on any page of the website.
- **Successful Postconditions:**
    - The error page is displayed.
- **Failure Scenarios:**
    1. Database goes offline.
    2. Internet connection fails.
- **Failure Handling**
    1. User is notified through an error page that servers unexpectedly crashed and is asked to login later.
    2. User is notified through an error page that the internet connection is lost and is asked to reconnect to the internet.
    3. Users are notified through an error page that a connection error occurred and all entries are automatically aborted.

Appendix A

Claims Checks

Claim	**Description**
The user is disabled.	This user has not activated their one-time password and does not have a profile. This user does not have access to any spaces or features.
The user has an active authentication session.	This user has created a profile and successfully logged in. They can view their home page and potentially any space that they have access to. This user can view and change their account settings.
The user has access to a space	This user can view and interact with the features of the space they have access to with an authenticated session. This user can add/remove members of the space they have access to. This user can receive notifications from this space (even if they are logged out).
The user is logged as a systems administrator.	This user has read and write access to all information of all generic user accounts. This user can view the analytics dashboard and has the ability to disable users. This user does not have write access to log directories (*Security Tips - Apache HTTP Server*, 2022).

Glossary

Term	Definition
Daily Active Users (DAU)	Metric for number of active users per day
Private information	Any user information that is only accessible and editable from that user. This includes the user's email and phone number.
Public information	Any user information that members in a shared space can view. This includes the user's profile icon and the user's name.
Referral Rate	Metric for how many invites get sent out for new users
Retention Rate	Percentage of users who continue engaging with the app over time (time intervals: 30 days, 7 days)
Sessions Per User (SPU)	Metric for how long a user session stays persisted
UAC	User Access Control

References

Jack. (2021, November 24). *Does the ADA Require Mobile Websites and Apps to be Accessible?* Boia.org; Bureau of Internet Accessibility. https://www.boia.org/blog/does-the-ada-require-mobile-websites-and-apps-to-be-accessible

Michael. (2018, October). *The Basics and Importance of Color Contrast for Web Accessibility*. Boia.org; Bureau of Internet Accessibility. https://www.boia.org/blog/the-basics-and-importance-of-color-contrast-for-web-accessibility

Most Popular Web Browsers in 2022 [Sep '22 Update] | Oberlo. (2022). Oberlo.com. https://www.oberlo.com/statistics/browser-market-share.

Security Tips - Apache HTTP Server Version 2.4. (2022). Apache.org. https://httpd.apache.org/docs/2.4/misc/security_tips.html

W3C Validation: What is it and why to use it? | LoginRadius Blog. (2020). Loginradius.com. https://www.loginradius.com/blog/engineering/w3c-validation/

Web Content Accessibility Guidelines (WCAG) 2.1. (2018, June 5). W3.org. https://www.w3.org/TR/WCAG21/