Verification and Validation Report: Room8

Team 19
Mohammed Abed
Maged Armanios
Jinal Kasturiarachchi
Jane Klavir
Harshil Patel

March 10, 2025

1 Revision History

Date	**Version**	**Notes**
March 10, 2025	1	Authors on front page

2 Symbols, Abbreviations and Acronyms

Acronym	**description**
SRS	Software Requirements Specification
VnV	Verification and Validation
CI/CD	Continuous Integration and Continuous deployment
API	Application Programming Interface
Exp.	Expected
Act.	Actual
Stmts	Statements
Fncs	Functions
NFR	Non-Functional Requirement
FR	Functional Requirement
JS	JavaScript
UTC	Coordinated Universal Time

Contents

List of Tables

List of Figures

This document covers the testing results of various unit tests and test from the VnV Plan document.

3 Functional Requirements Evaluation

3.1 FR211

The system shall allow users to create an account using their Google account.

FST-UAHM-1	**Profile Creation**
Description	Tests if the system is able to handle creating user profiles in the system from frontend inputs
Input	Valid Google account sign in
Exp. Output	Room8 account created and Frontend redirects user to the dashboard
Act. Output	Room8 account created and Frontend redirects user to the dashboard
Evaluation	Acceptance testing
Result	Pass

3.2 FR212

The system shall allows users to login to their account.

FST-UAHM-2	**System Login**
Description	Tests if the system is able to recognize a user profile and to authenticate them to the system
Input	Valid Google account matching the profile in the system
Exp. Output	User profile is authenticated with a new session entry. Frontend redirects user to the dashboard
Act. Output	User profile is authenticated with a new session entry. Frontend redirects user to the dashboard
Evaluation	Acceptance testing
Result	Pass

3.3 FR213

The system shall allows users to log out of their account.

FST-UAHM-3	**System Logout**
Description	Tests if the system logs the user out, invalidating their session and returning them to the homepage
Input	User clicks logout button
Exp. Output	The system invalidates the session and redirects the user to the homepage
Act. Output	The system invalidates the session and redirects the user to the homepage
Evaluation	Acceptance testing
Result	Pass

3.4 FR214

The system shall allow users to create a home group using the home name, address, and number of roommates.

FST-UAHM-4	**Create Home Instance**
Description	Tests if the system allows users to create a new home group with specified details
Input	Home name, address, and number of roommates to their respective fields
Exp. Output	A new home group instance is created with the specified details, and the user is added as a member
Act. Output	A new home group instance is created with the specified details, and the user is added as a member
Evaluation	Acceptance testing
Result	Pass

3.5 FR215

The system shall allow users to invite other users to join their home group.

FST-UAHM-5	**Invite and Remove Users From Home Instance**
Description	Tests if the system allows users to invite others to join a home group and remove them as needed
Input	Valid email address of another user to invite; selection of a user to remove
Exp. Output	The invited user receives an invitation, joins the home group upon acceptance, and can be removed by the home admin
Act. Output	The invited user receives an invitation, joins the home group upon acceptance, and can be removed by the home admin
Evaluation	End-to-end testing
Result	Pass

3.6 FR216

The system shall allow users to view the list of users in their home group.

FST-UAHM-4	**Create Home Instance**
Description	Tests if the system allows users to create a new home group with specified details
Input	Home name, address, and number of roommates to their respective fields
Exp. Output	A new home group instance is created with the specified details, and the user is added as a member
Act. Output	A new home group instance is created with the specified details, and the user is added as a member
Evaluation	Acceptance testing
Result	Pass

FST-UAHM-6	**Leave Home Instance**
Description	Tests if the system allows users to leave a home group as needed
Input	Leave home button press
Exp. Output	User receives feedback on their action to leave the home group
Act. Output	User receives feedback on their action to leave the home group
Evaluation	Functional testing
Result	Pass

3.7 FR217

The system shall allow users to remove users from their home group.

FST-UAHM-5	**Invite and Remove Users From Home Instance**
Description	Tests if the system allows users to invite others to join a home group and remove them as needed
Input	Valid email address of another user to invite; selection of a user to remove
Exp. Output	The invited user receives an invitation, joins the home group upon acceptance, and can be removed by the home admin
Act. Output	The invited user receives an invitation, joins the home group upon acceptance, and can be removed by the home admin
Evaluation	End-to-end testing
Result	Pass

3.8 FR218

The system shall allow users to leave their home group.

FST-UAHM-5	**Invite and Remove Users From Home Instance**
Description	Tests if the system allows users to invite others to join a home group and remove them as needed
Input	Valid email address of another user to invite; selection of a user to remove
Exp. Output	The invited user receives an invitation, joins the home group upon acceptance, and can be removed by the home admin
Act. Output	The invited user receives an invitation, joins the home group upon acceptance, and can be removed by the home admin
Evaluation	End-to-end testing
Result	Pass

FST-UAHM-6	**Leave Home Instance**
Description	Tests if the system allows users to leave a home group as needed
Input	Leave home button press
Exp. Output	User receives feedback on their action to leave the home group
Act. Output	User receives feedback on their action to leave the home group
Evaluation	Functional testing
Result	Pass

3.9 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.

FST-CC-1	Update Chatbot Settings
Description	Tests if the chatbot settings can be updated successfully by an authorized user
Input	Randomized array of include and exclude inputs
Exp. Output	ChatBot settings entry in database reflect the array of inputs
Act. Output	ChatBot settings entry in database reflect the array of inputs
Evaluation	Acceptance testing
Result	Pass

3.10 FR222

The ChatBot shall send reminders to the group chat about upcoming chores and events in the schedule 2 days in advance.

FST-CC-3	ChatBot Sends Messages to Groupchats
Description	Tests if the ChatBot is able to send messages to a group chat as per configuration
Input	Newly created chore, cleanliness score, and a shared expense
Exp. Output	Chatbot sends the appropriate configured messages to the group chat
Act. Output	Chatbot sends the appropriate configured messages to the group chat
Evaluation	End-to-end testing
Result	Pass

3.11 FR223

The ChatBot shall send notifications to the group chat about new shared living space cleanliness scores immediately after an event is added to the cleanliness manager page.

FST-CC-3	**ChatBot Sends Messages to Groupchats**
Description	Tests if the ChatBot is able to send messages to a group chat as per configuration
Input	Newly created chore, cleanliness score, and a shared expense
Exp. Output	Chatbot sends the appropriate configured messages to the group chat
Act. Output	Chatbot sends the appropriate configured messages to the group chat
Evaluation	End-to-end testing
Result	Pass

3.12 FR224

The ChatBot shall send notifications to the group chat about new shared expenses added to the bill splitter page immediately after its addition.

FST-CC-3	**ChatBot Sends Messages to Groupchats**
Description	Tests if the ChatBot is able to send messages to a group chat as per configuration
Input	Newly created chore, cleanliness score, and a shared expense
Exp. Output	Chatbot sends the appropriate configured messages to the group chat
Act. Output	Chatbot sends the appropriate configured messages to the group chat
Evaluation	End-to-end testing
Result	Pass

3.13 FR231

The system shall list the changes in objects of the shared living space before and after a user enters and exits the space.

FST-CM-1	**System Evaluates Cleanliness**
Description	Tests if the system can evaluate and record changed objects based on captured images
Input	N/A
Exp. Output	List of objects changed in the shared space
Act. Output	List of objects changed in the shared space
Evaluation	Acceptance testing
Result	Pass

3.14 FR233

The system shall display the picture of the detected mess in the shared living space.

FST-CM-2	**Display After Use Image**
Description	Tests if the system can evaluate and record cleanliness scores based on captured images
Input	N/A
Exp. Output	Prints user's changes to environment if any were made
Act. Output	Prints user's changes to environment if any were made
Evaluation	Unit testing
Result	Pass

3.15 FR234

The system shall allow users to view the cleanliness score of other users.

FST-CM-2	**Display After Use Image**
Description	Tests if the system can evaluate and record cleanliness scores based on captured images
Input	N/A
Exp. Output	Prints user's changes to environment if any were made
Act. Output	Prints user's changes to environment if any were made
Evaluation	Unit testing
Result	Pass

3.16 FR235

The system shall allow users to view the history of cleanliness scores and detected messes.

FST-CM-2	**Display After Use Image**
Description	Tests if the system can evaluate and record cleanliness scores based on captured images
Input	N/A
Exp. Output	Prints user's changes to environment if any were made
Act. Output	Prints user's changes to environment if any were made
Evaluation	Unit testing
Result	Pass

3.17 FR241

The system shall allow users to add a new chore to the schedule.

FST-SC-1	**Adding Event to Schedule**
Description	Tests that new event appears in calendar after it is inputted.
Input	Title, date, time, and duration
Exp. Output	Calendar displaying new chore with input parameters
Act. Output	Calendar displaying new chore with input parameters
Evaluation	Functional testing
Result	Pass

3.18 FR242

The system shall allow users to add a new event to the schedule.

FST-SC-1	**Adding Event to Schedule**
Description	Tests that new event appears in calendar after it is inputted.
Input	Title, date, time, and duration
Exp. Output	Calendar displaying new event with input parameters
Act. Output	Calendar displaying new event with input parameters
Evaluation	Functional testing
Result	Pass

3.19 FR243

The system shall allow users to input chore and event details (name, description, time, frequency, assigned users, etc.).

FST-SC-1	**Adding Event to Schedule**
Description	Tests that new event appears in calendar after it is inputted.
Input	Title, date, time, and duration
Exp. Output	Calendar displaying new chore/event with input parameters
Act. Output	Calendar displaying new chore/event with input parameters
Evaluation	Functional testing
Result	Pass

3.20 FR244

The system shall allow users to edit and delete chores and events.

FST-SC-2	**Removing Event from Schedule**
Description	Tests that event disappears from calendar after existing event is removed.
Input	Chore/event from calendar represented by name, date, time, and duration.
Exp. Output	Chore/event that was removed is no longer displayed in its previous timeblock in the calendar
Act. Output	Chore/event that was removed is no longer displayed in its previous timeblock in the calendar
Evaluation	Functional testing
Result	Pass

FST-SC-3	Editing Event in Schedule
Description	Tests that changes can be made to an already-existing event.
Input	Chore/event from calendar represented by name, date, time, and duration.
Exp. Output	Chore/event displayed in calendar after edits contains updated information (i.e. name, date, time, and/or duration)
Act. Output	Chore/event displayed in calendar after edits contains updated information (i.e. name, date, time, and/or duration)
Evaluation	Functional testing
Result	Pass

3.21 FR245

The system shall allow users to view the schedule and mark chores as complete.

FST-SC-1	Adding Event to Schedule
Description	Tests that new event appears in calendar after it is inputted.
Input	Title, date, time, and duration
Exp. Output	Calendar displaying new chore/event with input parameters
Act. Output	Calendar displaying new chore/event with input parameters
Evaluation	Functional testing
Result	Pass

FST-SC-4	**Display Event Schedule**
Description	Tests that all events displayed in calendar view.
Input	User goes to calendar page
Exp. Output	Calendar displaying chores and events pertaining to the home group
Act. Output	Calendar displaying chores and events pertaining to the home group
Evaluation	Unit testing
Result	Pass

3.22 FR251

The system shall allow users to add a new expense to the bill splitter and notify the involved users.

FST-BSC-1	**Add and Split Expense**
Description	Tests if the system allows users to add a new expense and splits it among group members
Input	New expense with an amount and shared expense roommates
Exp. Output	Expense is split equally among group members and recorded
Act. Output	Expense is split equally among group members and recorded
Evaluation	Acceptance testing
Result	Pass

3.23 FR252

The system shall allow users to view what they owe other housemates.

FST-BSC-2	**Display User Expenses**
Description	Tests if the system displays all expenses associated with a user
Input	User navigates to their expenses page
Exp. Output	System displays all expenses associated with the user
Act. Output	System displays all expenses associated with the user
Evaluation	Unit testing
Result	Pass

3.24 FR253

The system shall allow users to view what they owe others.

FST-BSC-2	**Display User Expenses**
Description	Tests if the system displays all expenses associated with a user
Input	User navigates to their expenses page
Exp. Output	System displays all expenses associated with the user
Act. Output	System displays all expenses associated with the user
Evaluation	Unit testing
Result	Pass

FST-BSC-3	Mark Expense as Paid
Description	Tests if the system allows users to mark an expense as paid
Input	User navigated to an expenses page and marks an expense as paid
Exp. Output	System updates the UI state and displays all expenses associated with the user
Act. Output	System updates the UI state and displays all expenses associated with the user
Evaluation	Unit testing
Result	Pass

3.25 FR254

The system shall allow users to mark expenses as paid.

FST-BSC-3	Mark Expense as Paid
Description	Tests if the system allows users to mark an expense as paid
Input	User navigated to an expenses page and marks an expense as paid
Exp. Output	System updates the UI state and displays all expenses associated with the user
Act. Output	System updates the UI state and displays all expenses associated with the user
Evaluation	Unit testing
Result	Pass

3.26 FR255

The system shall calculate debts in order to minimize the amount of transactions required between housemates.

FST-BSC-1	**Add and Split Expense**
Description	Tests if the system allows users to add a new expense and splits it among group members
Input	New expense with an amount and shared expense roommates
Exp. Output	Expense is split equally among group members and recorded
Act. Output	Expense is split equally among group members and recorded
Evaluation	Acceptance testing
Result	Pass

4 Nonfunctional Requirements Evaluation

4.1 Usability

4.1.1 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.

NFST-CM-6	**End of Use**
Description	Test to check if system detects that the user is no longer using the space.
Input	User uses shared space and leaves after use
Exp. Output	After picture
Act. Output	After picture
Evaluation	Functional testing
Result	Pass

4.1.2 NFR242

The calendar system shall display all calendar events to users in their time zone.

NFIT-SC-1	Calendar Event Time Storage and Display
Description	Tests calendar to ensure event time in database is in UTC but on user's application is displayed in user's local time
Input	Event scheduled with specific time and date
Exp. Output	Event in database in UTC and on user's calendar in their local time
Act. Output	Event in database in UTC and on user's calendar in their local time
Evaluation	Functional/Unit testing
Result	Pass

4.2 Performance

4.2.1 NFR214

The system should be able to authenticate a user with a median response time of under 1 second.

Using Jest (A front-end JavaScript testing library) we are able to rapidly execute sign-in attempts as if we were a user. Executing the sign-in attempt approximately 50 times (some failed due to timeouts) and averaging the result gives us the following execution times.

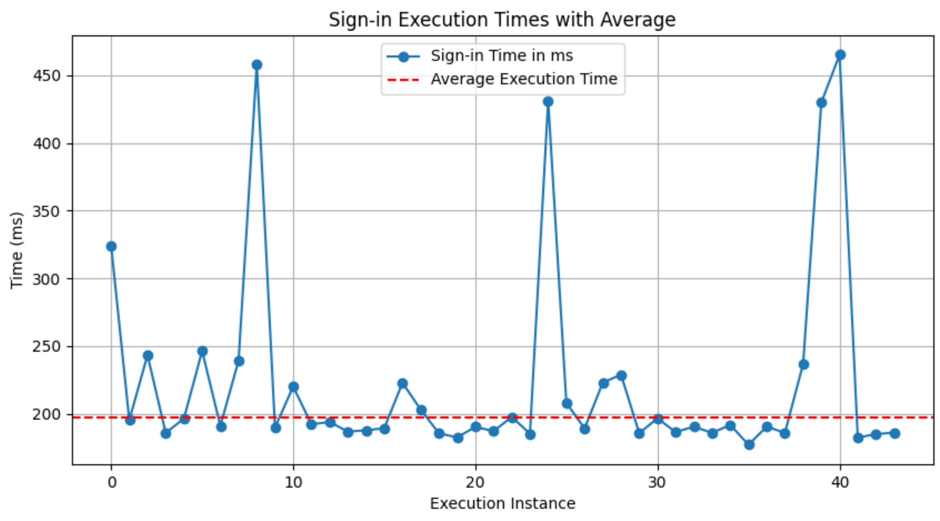

Figure 1: Sign In Times

As you can see it's clear that we satisfy the non-functional requirement.

4.2.2 NFR234

Photos captured with the system will be in a quality high enough to differentiate objects within frame.

NFST-CM-3	**Image Quality Validation**
Description	Test to check if captured image have sufficient quality
Input	Captured image of shared space
Exp. Output	Image of shared space
Act. Output	Image of shared space
Evaluation	Acceptance testing
Result	Pass

4.2.3 NFR221

The chatbot shall not disclose any sensitive information in its messages such as addresses or full names.

NFST-UAHM-4	**Authentication Response Time**
Description	Tests if system authenticates a user within a median response time of under 1 second
Input	Correct user login credentials filled in login elements
Exp. Output	Authentication process completed within one second
Act. Output	Authentication process completed within one second
Evaluation	Acceptance testing
Result	Pass

4.2.4 NFR235

The system shall process image data in under 30 minutes.

To test this, we ran the image processing algorithm 100 times on both a high-spec and a low-spec machine and averaged the execution matrix.

The specifications for the low-spec machine are:

- **GPU**: Integrated Graphics with CPU (shared GPU memory of 8GB)
- **RAM**: 16 GB
- **CPU cores**: 6 with 12 logical processors
- **CPU speed**: 2.5 GHz

The specifications for the high-spec machine are:

- **GPU**: NVIDIA Geforce GTX 1650 (4 GB dedicated GPU memory, 8 GB shared, 12 GB total)
- **RAM**: 16 GB
- **CPU cores**: 6 with 12 logical processors
- **CPU speed**: 4 GHz

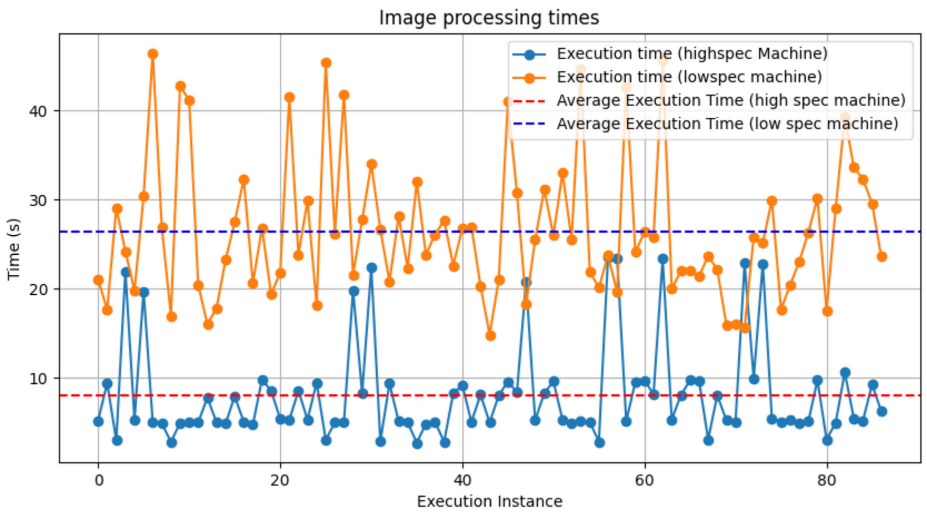

Figure 2: Cleanliness Algorithm Processing Times

As you can see it's clear that we satisfy the non-functional requirement.

4.3 Privacy and Security

4.3.1 NFR211

All data related to authentication must be encrypted in both transit and in storage.

NFST-UAHM-1	**Encryption of Authentication Data**
Description	Tests if system encrypts data in transit and when stored
Input	User inputted data in login elements
Exp. Output	Stored and data in transit are encrypted
Act. Output	Stored and data in transit are encrypted
Evaluation	Acceptance testing
Result	Pass

4.3.2 NFR212

Error messages related to authentication should not disclose sensitive details such as "Incorrect Password".

NFST-UAHM-2	**Sensitive Information Error Management**
Description	Tests if the system does not reveal sensitive user information during login attempts
Input	Valid email and incorrect password
Exp. Output	Error message that avoids revealing sensitive information
Act. Output	Error message that avoids revealing sensitive information
Evaluation	Acceptance testing
Result	Pass

4.3.3 NFR213

All data related to houses such as addresses and residents should be encrypted in transit and in rest.

NFST-UAHM-3	**Encryption of House Data**
Description	Tests if system encrypts all data related to houses in transit and at rest
Input	View or edit house data
Exp. Output	Verification that data in transit and stored data are encrypted
Act. Output	Verification that data in transit and stored data are encrypted
Evaluation	Acceptance testing
Result	Pass

4.3.4 NFR233

The system shall encrypt and securely store all images of homes.

NFST-CM-2	Encryption of Images and Videos
Description	Tests system to ensure encryption and secure storage of images and videos.
Input	Captured image and recording of shared space
Exp. Output	Image recording in storage are encrypted
Act. Output	Image recording in storage are encrypted
Evaluation	Acceptance testing
Result	Pass

4.3.5 NFR244

The calendar system shall encrypt all events stored.

NFST-SC-3	Events Are Encrypted
Description	Tests if system encrypts stored events
Input	Creating event
Exp. Output	Event is encrypted in transit and storage
Act. Output	Event is encrypted in transit and storage
Evaluation	Acceptance testing
Result	Pass

4.3.6 NFR251

The Bill Splitter system shall encrypt all events stored.

NFST-BSC-1	Events Are Encrypted
Description	Tests if system encrypts bill splitter events
Input	Bill splitting is used
Exp. Output	Event is encrypted in transit and storage
Act. Output	Event is encrypted in transit and storage
Evaluation	Acceptance testing
Result	Pass

4.4 Other

4.4.1 NFR222

The chatbot shall not send users too frequently to prevent annoying users.

NFST-CC-2	User Messaging Frequency
Description	Tests ChatBot to ensure it reasonably messages user and not too frequently.
Input	Trigger multiple notifications.
Exp. Output	Verify notifications are not excessively frequent
Act. Output	Verify notifications are not excessively frequent
Evaluation	Acceptance testing
Result	Pass

4.4.2 NFR236

The system shall not report false events which accuse someone of reducing the cleanliness score of an environment.

NFCT-CM-5	**False Event Prevention**
Description	Test to check if system avoids false
Input	User uses shared space but does not make a mess.
Exp. Output	Objects that were not changed are not listed
Act. Output	Objects that were not changed are not listed
Evaluation	Acceptance testing
Result	Pass

4.4.3 NFR241

The calendar system shall store all calendar events in UTC.

NFIT-SC-1	**Calendar Event Time Storage and Display**
Description	Tests calendar to ensure event time in database is in UTC but on user's application is displayed in user's local time.
Input	Event scheduled with specific time and date.
Exp. Output	Event in database in UTC and on user's calendar in their local time
Act. Output	Event in database in UTC and on user's calendar in their local time
Evaluation	Acceptance testing
Result	Pass

4.4.4 NFR243

The calendar system shall have a granularity of 5 minutes.

NFIT-SC-1	Calendar Event Time Storage and Display
Description	Tests calendar to ensure event time in database is in UTC but on user's application is displayed in user's local time.
Input	Event scheduled with specific time and date.
Exp. Output	Event in database in UTC and on user's calendar in their local time
Act. Output	Event in database in UTC and on user's calendar in their local time
Evaluation	Acceptance testing
Result	Pass

4.4.5 NFR252

The Bill Splitter shall allow users to record numerical values of prices with a granularity of two decimal places.

NFCT-BSC-2	Precision of Numerical Values
Description	Tests bill splitter's ability to calculate and display numerical values with preicion of two decimal places.
Input	Prices entered to split between two users.
Exp. Output	Price each user owes is display
Act. Output	Price each user owes is display
Evaluation	Acceptance testing
Result	Pass

5 Unit Testing

Unit testing for this project is defined as a way of testing the smallest piece of code that can be logically isolated. Logical isolation means to isolate a piece of code based on a particular function. In this project, unit testing was performed on the user-facing application of the project while the other codebases of the project will be tested with other methods.

For context on the this section of the report, the client-facing application was built using the following tools:

- **Next.js** as the JavaScript framework for both front-end and back-end
- **ReactQuery** for data synchronization and caching

and is broken up into different pages based on application features such as:

- Cleanliness management system
- Chore scheduling system
- Bill splitting system

While it is recommended to test as much of your code as possible, due to time constraints and generally better use of our time, the team developed unit tests only for the critical components (Referring to React components) of our application, while other components that exist for purposes such as reusable styling or wrappers were ignored. In general, unit tests were written with the objective of:

- Ensure key UI components were rendering
- Ensuring the UI displayed information fetched from the back-end
- Ensuring inputs and buttons on the UI are able to be interacted with

In the front-end, 115 different tests were written for over 20+ components resulting in the following code-coverage report.

Table 1: **Front-end Code Coverage Report**

File	**% Stmts**	**% Branch**	**% Funcs**	**% Lines**	**Uncovered Line #s**
All files	83.2	91.05	58.33	83.2	
app	100	100	100	100	
page.tsx	100	100	100	100	
app/(main)/bill-splitter/components	83.37	100	60	83.37	
debtsTable.tsx	83.72	100	25	83.72	26–32, 34–38, 76–77
historyTable.tsx	100	100	100	100	
loansTable.tsx	63.82	100	57.14	63.82	22–67, 91–93, 95–96
summaryCard.tsx	100	100	100	100	
summaryCardStub.tsx	100	100	100	100	
app/(main)/bill-splitter/hooks	33.33	100	12.5	33.33	
patchOwe.ts	55.26	100	50	55.26	5–21
useBillHistory.ts	19.44	100	0	19.44	6–29, 32–36
useBills.ts	26.08	100	0	26.08	5–16, 19–23
useOwes.ts	26.08	100	0	26.08	5–16, 19–23
app/(main)/chatbot	95.62	80	50	95.62	
page.tsx	95.62	80	50	95.62	81–85, 101
app/(main)/chatbot/components	95.74	33.33	50	95.74	
chatbot-setting-stub.tsx	95.74	33.33	50	95.74	21–22
app/(main)/chatbot/hooks	24.39	100	0	24.39	

File	**% Stmts**	**% Branch**	**% Funcs**	**% Lines**	**Uncovered Line #s**
useActivateChatbot.ts	24.39	100	0	24.39	9–24, 27–41
app/(main)/cleanliness-manager/components	88	77.77	62.5	88	
cleanliness-details-modal.tsx	86.45	16.66	100	86.45	39–50, 62
cleanliness-image.tsx	30.76	100	0	30.76	11–37
cleanliness-logs.tsx	100	100	100	100	
cleanliness-past.tsx	100	100	100	100	
task-card.tsx	91.8	75	33.33	91.8	50, 79–83, 90–93
task-list.tsx	100	100	100	100	
app/(main)/cleanliness-manager/hooks	54.1	100	20	54.1	
useCleanlinessLogs.tsx	46.34	100	33.33	46.34	6–27
useGetCleanlinessTasks.tsx	67.85	100	0	67.85	37–49, 52–56
useUpdateCleanlinessTask.tsx	44.89	100	20	44.89	13–26, 33–34, 36–41, 43–47
app/(main)/dashboard/components	100	100	100	100	
dashboard-cards.tsx	100	100	100	100	
app/(main)/house-settings/components	100	100	100	100	
create-note-modal.tsx	100	100	100	100	
edit-house-modal.tsx	100	100	100	100	
house-invites.tsx	100	100	100	100	
house-notes.tsx	100	100	100	100	

File	**% Stmts**	**% Branch**	**% Funcs**	**% Lines**	**Uncovered Line #s**
invite-user-modal.tsx	100	100	100	100	
app/(main)/house-settings/hooks	35	100	20	35	
useRemoveRoomate.ts	35	100	20	35	5–15, 22–23, 25–32, 34–38
app/(main)/profile	87.22	80	60	87.22	
page.tsx	87.22	80	60	87.22	25–31, 39–43, 94–101, 159–161
app/(main)/schedule	23.07	100	0	23.07	
adapters.ts	23.07	100	0	23.07	4–13
app/(main)/schedule/components	98.76	95.74	100	98.76	
chore-history.tsx	100	100	100	100	
create-chore-modal.tsx	99.41	85.71	100	99.41	143
pending-chores.tsx	100	100	100	100	
pending-item.tsx	100	100	100	100	
schedule-item.tsx	95.58	92	100	95.58	64–69, 76–77
schedule.tsx	100	100	100	100	
app/(main)/schedule/hooks	33.48	100	0	33.48	
useCreateChore.ts	14.28	100	0	14.28	5–17, 20–42
useDeleteChore.ts	14.63	100	0	14.63	5–16, 19–41
useGetAllActivities.ts	36.36	100	0	36.36	7–15, 18–22
useGetAllCompletedChores.ts	69.76	100	0	69.76	29–36, 39–43

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
useGetCompletedChores.ts	50	100	0	50	12–19, 22–26
useUpdateCompletedChore.ts	24	100	0	24	11–23, 26–50
app/auth/hooks	47.36	100	0	47.36	
useUser.tsx	47.36	100	0	47.36	19–38
components	100	100	85.71	100	
loading.tsx	100	100	100	100	
modal.tsx	100	100	100	100	
mutate-loading.tsx	100	100	100	100	
query-provider.tsx	100	100	100	100	
roommates-table.tsx	100	100	50	100	
components/ui	98.1	88.88	100	98.1	
badge.tsx	100	100	100	100	
button.tsx	100	50	100	100	42
card.tsx	100	100	100	100	
checkbox.tsx	100	100	100	100	
dialog.tsx	100	100	100	100	
form.tsx	92.26	92.85	100	92.26	50–51, 123–133
input.tsx	100	100	100	100	
label.tsx	100	100	100	100	
popover.tsx	100	100	100	100	
switch.tsx	100	100	100	100	

File	**% Stmts**	**% Branch**	**% Funcs**	**% Lines**	**Uncovered Line #s**
table.tsx	100	100	100	100	
tabs.tsx	100	100	100	100	
hooks	50	69.23	53.84	50	
useGetHouse.ts	65.51	60	100	65.51	10–19
useRoommates.ts	63.33	60	100	63.33	9–11, 13–20
useToast.ts	45.5	100	14.28	45.5	27–30, 59–72, 75–125, 131–136, 140–167
lib	61.53	100	50	61.53	
utils.ts	61.53	100	50	61.53	9–13
lib/constants	100	100	100	100	
index.ts	100	100	100	100	
lib/supabase	44.44	100	0	44.44	
browser.ts	44.44	100	0	44.44	5–9

6 Functional Testing

Functional testing is described as ensuring that the system behaves as expected given specific inputs. In the case of Room8, the functional testing is implemented in the form of test cases generated with specific samples which should match an expected output once passed through the cleanliness detection algorithm. It helped us identify which areas the software performs well in, and which ones needed improvement.

For example, the cleanliness detection software accurately captures the events of zero objects added, removed, and moved, in a scene where there are no changes. Similarly, in other test cases where single items were either added, removed, or moved, the system was able to correctly label the changes to the living spaces.

One area that the functional tests highlighted as an area of improvement was the sensitivity of our algorithm. Although it does a great job of recognizing all of the changes that are made, the heightened sensitivity results in false positives being produced.

In total, 8 test cases were produced, they are as follows:

- no changes
- 1 object added
- 1 object removed
- 1 object moved
- 1 object added, 1 object removed
- 1 object added, 1 object moved
- 1 object removed, 1 object moved
- 1 object added, 1 object removed, 1 object moved

In the end, all test cases passed, with the following table containing the results of the coverage report.

Table 2: **Detector Code Coverage Report**

File	**% Stmts**	**% Miss**	**% Cover**	**Missing**
main.py	300	129	57	63, 96, 138-140, 193-207, 238, 248, 289, 339-396, 401-469, 474-519, 522-545
testCleanlinessDetector.py	29	0	100	
TOTAL	329	129	61	

7 Changes Due to Testing

The following are requirements that were modified during the process of creating tests and/or writing the VnV Report:

- **FR211** - Change description to use Google account instead of using name, email and password.
- **FST-UAHM-1** - See above description.
- **FST-UAHM-2** - Change input field of test case to using Google account instead of email and password.
- **FST-CM-1** - Changed description to remove cleanliness score and add listing objects chaged.
- **FR232** - Removing because of it is no longer part of the implementation scope.
- **FST-CM-2** - Editing test case to remove cleanliness score mentions and update to showing after images.
- **NFR236** - Removing mention of cleanliness score.
- **NFCT-CM-5** - See above description.

7.1 Changes Due to Unit Testing

Unit testing discovered several bugs and anti-patterns in the front-end codebase that were resolved. Some examples of the discovered flaws include:

- Invalid references for form inputs and form labels. Leading to input labels not focusing their corresponding form input.
- Unneccessary props on React components.
- Missing semantic HTML attributes such as "role" on various items. Making the app less accessible and more difficult for the testing library to find components.
- Checking for lists to be empty by calling the list in a boolean statement, and not explicitly checking its length. This resulted in empty lists still being evaluated to "True" because in JS empty lists are truthy and resulted in code logic that was supposed to trigger when the list was empty to not work as expected.

- Displaying times without the time zone. This was detected because the testing suite failed on various environments due to the time zone printing differently on machines with different time zones.

The code coverage reports shows some trends, mainly that all most if not all of our hooks have poor code coverage rates. This is expected and acceptable because the purpose of the hooks is to fetch data from the back-end and display it on the front-end and since we were mocking the back-end data, most of the hooks' functionality were not used.

7.2 Changes Due to Functional Testing

As mentioned in the functional testing section above, it was discovered through functional testing that the cleanliness detection algorithm was more sensitive to slight changes in the scene that were not expected in the output. As a result, a "blacklist" was created to filter out objects that, in the context of our problem, are unreasonable to detect as changes.

For instance, an oven, refrigerator, or table are very rarely (if at all) going to be added, removed, or moved from a kitchen. As such, even if the detector incorrectly labels them as changes between the before and after images, these can be disregarded and removed from the outputs.

To be clear, the blacklist was only applied to the functional tests. This was a point that has been brought up by our project supervisor in the past, and we plan to migrate it to the overall cleanliness detector as a result. This will help fine-tune our outputs to exactly what we expect.

8 Automated Testing

Automated tests include the unit tests and the cleanliness detection system's test samples. The front-end tests were additionally automated to run on push and on merge requests to main and dev. These automated tests on GitHub in conjunction with branch rules that prevent merging branches which fail tests ensure that no changes the fail tests will make it to the production environment.

9 Trace to Requirements

- **FR211:** FST-UAHM-1

- **FR212:** FST-UAHM-2
- **FR213:** FST-UAHM-3
- **FR214:** FST-UAHM-4
- **FR215:** FST-UAHM-5
- **FR216:** FST-UAHM-4
- **FR217:** FST-UAHM-5
- **FR218:** FST-UAHM-5, FST-UAHM-6
- **FR221:** FST-CC-1
- **FR222:** FST-CC-3
- **FR223:** FST-CC-3
- **FR224:** FST-CC-3
- **FR231:** FST-CM-1
- **FR233:** FST-CM-2
- **FR234:** FST-CM-2
- **FR235:** FST-CM-2
- **FR241:** FST-SC-1
- **FR242:** FST-SC-1
- **FR243:** FST-SC-1
- **FR244:** FST-SC-2, FST-SC-3
- **FR245:** FST-SC-1, FST-SC-4
- **FR251:** FST-BSC-1
- **FR252:** FST-BSC-2
- **FR253:** FST-BSC-2, FST-BSC-3
- **FR254:** FST-BSC-3
- **FR255:** FST-BSC-1

10 Trace to Modules

Module	**Tests**
Account Management	FST-UAHM-1, FST-UAHM-2, FST-UAHM-3, FST-UAHM-4, NFST-UAHM-1, NFST-UAHM-2
Home Management	FST-UAHM-4, FST-UAHM-5, FST-UAHM-6, NFST-UAHM-3
Schedule Manager	FST-SC-1, FST-SC-2, FST-SC-3, FST-SC-1, FST-SC-4, NFIT-SC-1, NFST-SC-3, NFIT-SC-1
Bill Splitter	FST-BSC-1, FST-BSC-2, FST-BSC-3, NFST-BSC-1, NFCT-BSC-2
Cleanliness Manager	FST-CM-1, FST-CM-2, NFST-CM-6, NFST-CM-3, NFST-CM-2, NFCT-CM-5
SMS ChatBot	FST-CC-1, FST-CC-3, NFST-CC-2

11 Code Coverage Metrics

Table 1: **Front-end Code Coverage Report**

File	**% Stmts**	**% Branch**	**% Funcs**	**% Lines**	**Uncovered Line #s**
All files	83.2	91.05	58.33	83.2	
app	100	100	100	100	
page.tsx	100	100	100	100	
app/(main)/bill-splitter/components	83.37	100	60	83.37	
debtsTable.tsx	83.72	100	25	83.72	26–32, 34–38, 76–77
historyTable.tsx	100	100	100	100	
loansTable.tsx	63.82	100	57.14	63.82	22–67, 91–93, 95–96
summaryCard.tsx	100	100	100	100	
summaryCardStub.tsx	100	100	100	100	
app/(main)/bill-splitter/hooks	33.33	100	12.5	33.33	
patchOwe.ts	55.26	100	50	55.26	5–21
useBillHistory.ts	19.44	100	0	19.44	6–29, 32–36
useBills.ts	26.08	100	0	26.08	5–16, 19–23
useOwes.ts	26.08	100	0	26.08	5–16, 19–23
app/(main)/chatbot	95.62	80	50	95.62	
page.tsx	95.62	80	50	95.62	81–85, 101
app/(main)/chatbot/components	95.74	33.33	50	95.74	
chatbot-setting-stub.tsx	95.74	33.33	50	95.74	21–22
app/(main)/chatbot/hooks	24.39	100	0	24.39	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
useActivateChatbot.ts	24.39	100	0	24.39	9–24, 27–41
app/(main)/cleanliness-manager/components	88	77.77	62.5	88	
cleanliness-details-modal.tsx	86.45	16.66	100	86.45	39–50, 62
cleanliness-image.tsx	30.76	100	0	30.76	11–37
cleanliness-logs.tsx	100	100	100	100	
cleanliness-past.tsx	100	100	100	100	
task-card.tsx	91.8	75	33.33	91.8	50, 79–83, 90–93
task-list.tsx	100	100	100	100	
app/(main)/cleanliness-manager/hooks	54.1	100	20	54.1	
useCleanlinessLogs.tsx	46.34	100	33.33	46.34	6–27
useGetCleanlinessTasks.tsx	67.85	100	0	67.85	37–49, 52–56
useUpdateCleanlinessTask.tsx	44.89	100	20	44.89	13–26, 33–34, 36–41, 43–47
app/(main)/dashboard/components	100	100	100	100	
dashboard-cards.tsx	100	100	100	100	
app/(main)/house-settings/components	100	100	100	100	
create-note-modal.tsx	100	100	100	100	
edit-house-modal.tsx	100	100	100	100	
house-invites.tsx	100	100	100	100	
house-notes.tsx	100	100	100	100	

File	**% Stmts**	**% Branch**	**% Funcs**	**% Lines**	**Uncovered Line #s**
invite-user-modal.tsx	100	100	100	100	
app/(main)/house-settings/hooks	35	100	20	35	
useRemoveRoomate.ts	35	100	20	35	5–15, 22–23, 25–32, 34–38
app/(main)/profile	87.22	80	60	87.22	
page.tsx	87.22	80	60	87.22	25–31, 39–43, 94–101, 159–161
app/(main)/schedule	23.07	100	0	23.07	
adapters.ts	23.07	100	0	23.07	4–13
app/(main)/schedule/components	98.76	95.74	100	98.76	
chore-history.tsx	100	100	100	100	
create-chore-modal.tsx	99.41	85.71	100	99.41	143
pending-chores.tsx	100	100	100	100	
pending-item.tsx	100	100	100	100	
schedule-item.tsx	95.58	92	100	95.58	64–69, 76–77
schedule.tsx	100	100	100	100	
app/(main)/schedule/hooks	33.48	100	0	33.48	
useCreateChore.ts	14.28	100	0	14.28	5–17, 20–42
useDeleteChore.ts	14.63	100	0	14.63	5–16, 19–41
useGetAllActivities.ts	36.36	100	0	36.36	7–15, 18–22
useGetAllCompletedChores.ts	69.76	100	0	69.76	29–36, 39–43

File	**% Stmts**	**% Branch**	**% Funcs**	**% Lines**	**Uncovered Line #s**
useGetCompletedChores.ts	50	100	0	50	12–19, 22–26
useUpdateCompletedChore.ts	24	100	0	24	11–23, 26–50
app/auth/hooks	47.36	100	0	47.36	
useUser.tsx	47.36	100	0	47.36	19–38
components	100	100	85.71	100	
loading.tsx	100	100	100	100	
modal.tsx	100	100	100	100	
mutate-loading.tsx	100	100	100	100	
query-provider.tsx	100	100	100	100	
roommates-table.tsx	100	100	50	100	
components/ui	98.1	88.88	100	98.1	
badge.tsx	100	100	100	100	
button.tsx	100	50	100	100	42
card.tsx	100	100	100	100	
checkbox.tsx	100	100	100	100	
dialog.tsx	100	100	100	100	
form.tsx	92.26	92.85	100	92.26	50–51, 123–133
input.tsx	100	100	100	100	
label.tsx	100	100	100	100	
popover.tsx	100	100	100	100	
switch.tsx	100	100	100	100	

File	**% Stmts**	**% Branch**	**% Funcs**	**% Lines**	**Uncovered Line #s**
table.tsx	100	100	100	100	
tabs.tsx	100	100	100	100	
hooks	50	69.23	53.84	50	
useGetHouse.ts	65.51	60	100	65.51	10–19
useRoommates.ts	63.33	60	100	63.33	9–11, 13–20
useToast.ts	45.5	100	14.28	45.5	27–30, 59–72, 75–125, 131–136, 140–167
lib	61.53	100	50	61.53	
utils.ts	61.53	100	50	61.53	9–13
lib/constants	100	100	100	100	
index.ts	100	100	100	100	
lib/supabase	44.44	100	0	44.44	
browser.ts	44.44	100	0	44.44	5–9

Table 2: **Detector Code Coverage Report**

File	**% Stmts**	**% Miss**	**% Cover**	**Missing**
main.py	300	129	57	63, 96, 138-140, 193-207, 238, 248, 289, 339-396, 401-469, 474-519, 522-545
testCleanlinessDetector.py	29	0	100	
TOTAL	329	129	61	

Appendix — Reflection

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

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?
2. What pain points did you experience during this deliverable, and how did you resolve them?
3. Which parts of this document stemmed from speaking to your client(s) or a proxy (e.g. your peers)? Which ones were not, and why?
4. In what ways was the Verification and Validation (VnV) Plan different from the activities that were actually conducted for VnV? If there were differences, what changes required the modification in the plan? Why did these changes occur? Would you be able to anticipate these changes in future projects? If there weren't any differences, how was your team able to clearly predict a feasible amount of effort and the right tasks needed to build the evidence that demonstrates the required quality? (It is expected that most teams will have had to deviate from their original VnV Plan.)

Answers:

1. Compiling results from tests such as code coverage metrics and satisfaction criteria was simple and easy to port over to the document. Additionally, due to how advanced the average testing library is, writing test cases to check various things such as running time, output results, etc. is very easy and that made working on the tests for this deliverable very easy. Finally, division of labour in the deliverable was a lot easier because multiple people can contribute to a section independently because our project is divided into

separate mini-projects that can each have their own automated tests, unit tests, etc.

2. Creating appropriate unit tests was difficult at first because the team was inexperienced with writing quality unit tests. After that, another pain point that we encountered while writing this document was finding appropriate sections to place our tests for the AI cleanliness detection system. Those tests are neither automated nor are they unit tests but they are extremely relevant. In the end, we ended up creating a section for those tests.

3. Tests for the cleanliness detection system (the AI system) were inspired from conversations with our supervisor Dr. Tharmarasa, who advised us to curate sample cases to prove our system's functionality. Unit tests and generally all front-end tests were implemented without discussions from users or peers. The reason clients and proxies were not consulted for the development of tests was because our requirements were not built from user feedback.

4. Our original VnV plan differed from the actual verification and validation activities in several ways. First, the testing tools we used provided unexpected insights into code improvements, such as missing accessibility labels and code smells we hadn't anticipated detecting. Second, some non-functional requirements weren't defined precisely enough to create clear test cases for them, requiring us to develop more specific interpretations. Finally, after completing this VnV report, we gained a much better understanding of what makes a requirement feasibly testable, knowledge we will apply when writing requirements for future projects. These deviations were primarily due to our team's inexperience with comprehensive testing methodologies at the project's outset.