

# Housemates: System Verification and Validation Plan for 4G06 - Software Engineering

Team #9, Housemates  
Justin Dang - dangj15  
Harris Hamid - hamidh1  
Fady Marcos - morocof2  
Rizwan Ahsan - ahsanm7  
Sheikh Afsar - afsars

November 3, 2023

## Revision History

Date	Version	Notes
Nov 3 2023	0	Revision 0

# Contents

<b>1</b>	<b>General Information</b>	<b>1</b>
1.1	Summary . . . . .	1
1.2	Objectives . . . . .	1
1.3	Relevant Documentation . . . . .	1
<b>2</b>	<b>Plan</b>	<b>2</b>
2.1	Verification and Validation Team . . . . .	2
2.2	SRS Verification Plan . . . . .	2
2.3	Design Verification Plan . . . . .	2
2.4	Verification and Validation Plan Verification Plan . . . . .	3
2.5	Implementation Verification Plan . . . . .	3
2.6	Automated Testing and Verification Tools . . . . .	3
2.7	Software Validation Plan . . . . .	3
<b>3</b>	<b>System Test Description</b>	<b>3</b>
3.1	Tests for Functional Requirements . . . . .	3
3.1.1	Task Management System . . . . .	4
3.1.2	Bill Management System . . . . .	7
3.1.3	Scheduling System . . . . .	10
3.1.4	Account System . . . . .	11
3.2	Tests for Nonfunctional Requirements . . . . .	15
3.2.1	Look and Feel Requirements . . . . .	15
3.2.2	Usability and Humanity Requirements . . . . .	16
3.2.3	Performance Requirements . . . . .	19
3.2.4	Operational and Environmental Requirements . . . . .	20
3.2.5	Maintainability and Support Requirements . . . . .	21
3.2.6	Security Requirements . . . . .	22
3.2.7	Compliance Requirements . . . . .	23
3.3	Traceability Between Test Cases and Requirements . . . . .	24
<b>4</b>	<b>Unit Test Description</b>	<b>25</b>
4.1	Unit Testing Scope . . . . .	25
4.2	Tests for Functional Requirements . . . . .	26
4.3	Tests for Nonfunctional Requirements . . . . .	26
4.4	Traceability Between Test Cases and Modules . . . . .	26

<b>5</b>	<b>Appendix</b>	<b>27</b>
5.1	Symbolic Parameters . . . . .	27
5.2	Usability Survey Questions . . . . .	27
5.3	Reflection . . . . .	29
5.3.1	Knowledge and Skills . . . . .	29
5.3.2	Approaches . . . . .	29

# 1 General Information

## 1.1 Summary

Housemates is an application that will allow its users to communicate with their housemates more effectively. The Housemates application will have a cost management system to split costs amongst housemates, a chore management system to delegate tasks amongst housemates, and a scheduling system to allow housemates to schedule events.

## 1.2 Objectives

The objective of this verification and validation plan is to

1. Ensure that the requirements listed in the SRS for the Housemates application are correct.
2. Ensure that the design of the Housemates application satisfies the requirements listed in the SRS.
3. Ensure the usability of the Housemates application for its prospective users.

## 1.3 Relevant Documentation

- **Development Plan:** The development plan contains the tools planned to be used in the development of the application along with the overall schedule for this project. These items will be referenced to in this VnV plan.
- **SRS:** The software requirements specification contains the functional and non-functional requirements of the Housemates application. These requirements will be tested for in the VnV plan under system testing.
- **Hazard Analysis:** The hazard analysis document contains the hazards associated with the application. These hazards will need to be tested for in the VnV plan in order to be sure that they are mitigated.

## **2 Plan**

This section contains the different verification and validation plans that will be used throughout the project along with the team responsible for executing them.

### **2.1 Verification and Validation Team**

- Justin Dang: Will focus on non-functional testing and usability survey.
- Harris Hamid: Will focus on bill management system as a part of functional testing
- Fady Marcos: Will focus on Account system as a part of functional testing
- Rizwan Ahsan: Will focus on scheduling system as a part of functional testing
- Sheikh Afsar: Will focus on task management system as part of functional testing

### **2.2 SRS Verification Plan**

- Classmates will review the SRS documentation.
- A TA for SFWRENG 4G06 will review the SRS documentation.
- The SRS checklist listed in the Checklists folder will be used by the verification and validation team to review the SRS documentation.
- A focus group with potential users of the application will be shown a wire-frame version of the Housemates to see if fits user needs.

### **2.3 Design Verification Plan**

- Classmates will review the Design documentation.
- A TA for SFWRENG 4G06 will review the Design documentation.
- The Design checklists in the Checklists folder will be used by the verification and validation team to review the Design documentation.

## **2.4 Verification and Validation Plan Verification Plan**

- Classmates will review the VnV documentation.
- A TA for SFWRENG 4G06 will review the VnV documentation.
- The VnV checklist listed in the Checklists folder will be used by the verification and validation team to review the VnV documentation.
- The tests in the VnV plan will be executed and the results will be documented in the VnV report.

## **2.5 Implementation Verification Plan**

After the completion of the Housemates application a code inspection and walkthrough will be conducted on the source code of the application. This will be used to identify any potential errors in the source code that prevent the application from satisfying its requirements.

## **2.6 Automated Testing and Verification Tools**

The automated testing and verification tools used in the VnV of the Housemates application is covered the development plan under sections 4,6 and 7.

## **2.7 Software Validation Plan**

We plan to conduct user testing on a focus group of potential users of the Housemates application after the Revision 0 version is complete. The user testing will be using the usability survey questions as described in the appendix.

# **3 System Test Description**

## **3.1 Tests for Functional Requirements**

This section contains the tests for the functional requirements. The subsections in this section are based off the SRS and cover the different features of

the Housemates application. The fit criterion for each functional requirement of the SRS are used here to help create tests for those functional requirements

### **3.1.1 Task Management System**

This subsection covers the tests for the task management system of the Housemates application. The task management system of the housemates application will allow users to split and delegate common household tasks to their housemates.

#### **1. test-TM1-1**

Control: Manual

Description: Verify navigation to Task Management page via tasks icon.

Initial State: Housemates app is open.

Input: Tap on the tasks icon.

Output: User is taken to the Task Management page.

Test Case Derivation: This is to be able to navigate to the task management page of the app instead of utilizing one of the other features.

How test will be performed: Open the Housemates app. Tap on the tasks icon. Verify that the Task Management page is displayed.

#### **2. test-TM2-1**

Control: Manual

Description: Verify that the Task Management page displays a list of tasks with user's chores highlighted.

Initial State: Task Management page is open.

Input: None.

Output: User can clearly identify their assigned tasks.

Test Case Derivation: This is to be able to clearly view the tasks which the user must complete.

How test will be performed: Open the Task Management page. Verify that a list of tasks is displayed, with user's chores highlighted.



### 3. test-TM3-1

Control: Manual

Description: Verify creation of a new task.

Initial State: Task Management page is open.

Input: Task Name: "Clean the Living Room" Due Date: 2023-11-05  
Roommate Assignment: Fardeen Afsar.

Output: New task "Clean the Living Room" is added to the list of tasks.

Test Case Derivation: Users need to be able to add new chores/tasks as needed.

How test will be performed: Open the Task Management page. Click on "Create New Task" button. Fill out the task creation form with the provided input. Click "Create" button. Verify that the new task appears in the list of tasks.

### 4. test-TM4-1

Control: Manual

Description: Verify assigning a task to a roommate.

Initial State: Task Management page is open with at least one task.

Input: Task Name: Select a task from the list. Choose roommate: Fardeen.

Output: Task is assigned to Fardeen.

Test Case Derivation: This feature enables users to divide tasks among roommates.

How test will be performed: Open the Task Management page. Select a task from the list. Choose roommate "Fardeen" from the list of roommates. Verify that the task is now assigned to Fardeen.

### 5. test-TM5-1

Control: Manual

Description: Verify marking a task as completed.

Initial State: Task Management page is open with at least one task.

Input: Task Name: Select a task from the list. Mark task as completed.

Output: Task is marked as completed and status is updated.

Test Case Derivation: Users need to be allowed to track the progress of tasks and mark them as completed when they are done.

How test will be performed: Open the Task Management page. Select a task from the list. Mark the task as completed. Verify that the task is now marked as completed on the Task Management page.

#### 6. test-TM3-2

Control: Automatic

Description: Create New Task.

Initial State: Task management page is open.

Input: Task Name: Task Name: "Clean the Kitchen" Due Date: 2023-11-05 Roommate Assignment: Fardeen.

Output: New task "Clean the Kitchen" is added to the list of tasks.

How test will be performed: Call the function `createNewTask()` with parameters: Task Name: "Clean the Kitchen" Due Date: 2023-11-05 Roommate Assignment: Fardeen. Verify that the new task "Clean the Kitchen" is added to the list of tasks.

#### 7. test-TM4-2

Control: Automatic

Description: Verify that a task can be assigned to the roommate.

Initial State: Task Management page is open with at least one task.

Input: Task Name: Select a task from the list. Choose Roommate: Fardeen.

Output: Task is assigned to Fardeen.

How test will be performed: Call the function `'assignTaskToRoommate()'` with parameters: Task: Selected Task (from the list) Roommate: Fardeen. Verify that the task is now assigned to Jane Doe.

#### 8. test-TM5-2

Control: Automatic

Description: Verify that a task can be marked as completed.

Initial State: Task Management page is open with at least one task.

Input: Task Name: Select a task from the list. Mark task as completed.

Output: Task is marked as completed and status is updated.

How test will be performed: Call the function 'markTaskAsCompleted()' with parameter: Task: Selected Task (from the list). Verify that the task is marked as completed and status is updated.

### 3.1.2 Bill Management System

This subsection covers the tests for the bill management system of the Housemates application. The bill management system of the housemates application will allow users to split bills with their housemates who incur a specific expense.

#### 1. test-BM1-1/2-1

This test checks whether a user is able to create a new bill and assign housemates to it

Control: Manual

Initial State: The bill management system is active and ready to handle events the user provides.

Input: Total amount incurred and users who to split among.

Output: A bill is generated with provided total amount being split evenly among housemates.

Test Case Derivation: The system expects a positive value in a specified format according to the chosen currency which is provided in the input the chosen users to share the expense with thus accepted by the system.

How test will be performed: A tester will be provided instructions to follow with a valid bill amount and associated housemates. The tester will then determine by cross-checking whether the result provided by the system matches with the actual result in the instructions.

## 2. test-BM3-1

This test ensures whether a user (bill issuer) is able to modify an already existing bill

Control: Manual

Initial State: The bill management system is active and ready to handle events the user provides.

Input: User selects to edit any past expense, alters previously inputted information as deemed fit and may change the users by adding or removing users.

Output: A new bill is generated with provided total amount being split evenly among housemates Note: this alteration will not change the date on which this expense was incurred.

Test Case Derivation: The system expects a positive value in a specified format according to the chosen currency which is provided in the input and the chosen users to share the expense with thus accepted by the system.

How test will be performed: A tester will be provided instructions to follow with a valid bill amount and associated housemates. The tester will then determine by cross-checking whether the result provided by the system matches with the actual result in the instructions.

## 3. test-BM4-1

This test checks whether a user is able to categorize a bill (e.g. food, utilities etc) when created.

Control: Manual

Initial State: The bill management system is active and ready to handle events the user provides.

Input: Select from a pre-defined set of categories to assign the type of bill and other necessary inputs.

Output: A bill is generated with provided total amount being split evenly among housemates along with the nature(type) of the bill.

Test Case Derivation: The system expects the user to choose the type of expense for the bill. This will help the system categorize certain bills together for easier retrieval for user.

How test will be performed: A tester will be provided instructions to follow with a valid bill amount and associated housemates and the chosen bill category. The tester will then determine by cross-checking whether the category of the bill matches with the actual category generated by the bill.

#### 4. test-BM5.1/7.1

This test ensures that previously issued bills can be accessed and can verify the status of expense whether it has been paid or not.

Control: Manual

Initial State: The bill management system is active and ready to handle events the user provides.

Input: User selects to view any past expense.

Output: A bill is shown with status of each user who incurred the expense of whether they have paid or received the amount.

Test Case Derivation: The system expects to show an updated bill of all the user and what each user owes or is owed.

How test will be performed: A tester will be provided instructions to follow. It will include instructions to first view the bill and observe the current status of what user owes what, then log in as a user associated to that bill and settle the expense. Now check the previous bill again and verify whether bill has been updated.

#### 5. test-BM6.1

This test ensures that users are able to attach relevant receipts to the bill as an image and view at a later period.

Control: Manual

Initial State: The bill management system is active and ready to handle events the user provides.

Input: User selects option to add attachment and adds total amount with users who to split among.

Output: A bill is shown with the total amount split evenly amongst users and an accessible attachment in form of an image as proof of purchase.

Test Case Derivation: The system expects to show the bill with the original receipt and proof of purchase.

How test will be performed: A tester will be provided instructions to follow. It will include instructions to first create a new bill and provide required information in the fields. The tester will then add an image as an attachment and save the bill. Tester will then access the bill from previous bill section and verify whether attachment can be seen.

### **3.1.3 Scheduling System**

This subsection covers the tests for the scheduling system of the Housemates application. The scheduling system of the housemates application will allow users to schedule events to coordinate with their housemates.

#### **1. test-SS1-1**

This test is to ensure users can navigate to the scheduling system page.

Control: Manual

Initial State: The app is launched and in the home interface.

Input: Tap on the scheduling icon.

Output: The scheduling page is displayed to the user.

Test Case Derivation: The expected result in the scheduling page appearing when the scheduling icon is tapped, as described in the requirement SS1.

How test will be performed: A tester will open the app in the home interface and tap on the scheduling icon. They will verify that the scheduling page is successfully displayed.

#### **2. test-SS2-1**

This test is to ensure users can create new events and schedule them within the scheduling feature.

Control: Manual

Initial State: The user is on the scheduling page

Input: The user taps on the button to create new event and fills out the event creation form the necessary description.

Output: A new event is listed in the scheduling page.

Test Case Derivation: The system expects to populate the scheduling page with a new event and display it to the user.

How test will be performed: A tester will navigate to the scheduling page, tap on the button to create new event and input necessary details in the form. They will confirm that the new event appears correctly in the scheduling page.

### 3. test-SS3-1

This test is to ensure users can view all the events scheduled by other roommates on their calendar within the scheduling system.

Control: Manual

Initial State: The user is on the scheduling page that shows the existing events.

Input: User views the calendar within the scheduling system.

Output: The user can clearly distinguish between personal events and shared roommate events on the calendar.

Test Case Derivation: Since the system will already be populated with different user's events, the scheduling system should clearly display all events on the calendar.

How test will be performed: A tester will access the scheduling page, view the calendar, and verify that personal events and shared roommate events are clearly distinguishable.

#### 3.1.4 Account System

This subsection covers the tests for the account system of the Housemates application. The account system of the housemates application will manage and store user data.

## 1. test-AS1-1

Control: Manual

This test is to ensure a user can create an account.

Initial State: The valid user credentials to be input do not exist in the system yet.

Input: User selects create account option, then selects the username field, inputs valid username, then selects password field and inputs valid password.

Output: Interface displays that user creation was successful and the credentials are now in the system.

Test Case Derivation: The test case is derived based on the requirement of the housemates application that it must allow users to create an account having entered valid and unique credentials and give the user feedback that creation was successful.

How test will be performed: Tester will open app, will press create an account, enter valid credentials and verify that the system allows creation with no issues.

## 2. test-AS1-2

Control: Manual

This test is to ensure that no two users can exist in the system

Initial State: The valid user already exists in the system.

Input: User selects create account option, then selects the username field, inputs the valid username that is already in the system, then selects password field and inputs valid password.

Output: The system does not create the profile and informs the user that account creation was not successful that username is already taken.

Test Case Derivation: The test case is derived from the requirements of the housemates app that there should not be two users with the same username.



How test will be performed: Tester will open app, select create an account option and input a username that is already taken and valid password then submit the form.

### 3. test-AS1-3

Control: Manual

This test is to ensure that no user profile can be created that does not meet the username and password requirements.

Initial State: The invalid user credentials do not exist in the system.

Input: User selects create account option, then selects the username field, inputs all cases of invalid passwords and all cases of invalid usernames that is already in the system, then selects password field and inputs valid password.

Output: The system does not create the profile and informs the user of the credential requirements and that creation was not successful due to invalid credentials.

Test Case Derivation: The test case is derived from the requirements of the housemates app which is that user credentials must meet certain requirements.

How test will be performed: Tester will open app, select create an account option and input an invalid username and/or password. Tester will test valid username with invalid password, invalid username with valid password and a case where they are both invalid. Tester will also test all cases of invalid username and all cases of invalid password.

### 4. test-AS2

Control: Manual

This test is to ensure that users can login.

Initial State: The user credentials do exist in the system.

Input: User selects login option, then enters user credentials.

Output: The system lets the user login.

Test Case Derivation: This is derived based on the requirement that users can have working credentials which they can log into the system with.

How test will be performed: Tester will open app, select login feature and enter valid user credentials. System should then let user login to associated account.

#### 5. test-AS3

Control: Manual

This test is to ensure that users can update profile details.

Initial State: User is logged in.

Input: User selects edit profiles option, then enters new user details and saves details.

Output: The system saves new user details.

Test Case Derivation: The test case is derived from the requirements of the housemates app which is that users can edit profile.

How test will be performed: With logged on account tester will select edit profile details option. They will then enter valid profile details and select the save option. System should now display new user profile details.

#### 6. test-AS4

Control: Manual

This test is to ensure that users can delete their account.

Initial State: User is logged in.

Input: User selects delete account option and submits the correct user credentials for that account.

Output: The system should delete that user account.

Test Case Derivation: The test case is derived from the requirements of the housemates app which is that users can delete their account.

How test will be performed: With logged on account Tester will select edit profile details option. They will then enter the correct user

credentials for that account. The account will then be deleted by the system.

#### 7. test-AS5

Control: Manual

This test is to ensure users are able to recover their account in case they forgot their login information.

Initial State: The user account already exists in the system.

Input: User selects the "forget password" option and enters username of an existing user account.

Output: The system sends an email to the email associated with the user account.

Test Case Derivation: The test case is derived from the requirements of the housemates app which is that user should be able to recover their account.

How test will be performed: Tester will open app, select login feature, then select "forget password" feature and enters a username of an account. The email of the associated account will receive an email from the system.

## 3.2 Tests for Nonfunctional Requirements

This section contains the tests for the non-functional requirements. The subsections in this section are based off the SRS. The fit criterion for each non-functional requirement of the SRS are used here to help create tests for those non-functional requirements

### 3.2.1 Look and Feel Requirements

This section contains the tests for the Look and Feel Requirements.

#### 1. test-LF-A1-1

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: The developers will observe the interface of the application.

Output/Result: The developers of the application will confirm the interface follows [Google's material design guidelines](#).

How test will be performed: Once the development of the application is complete, the developers will go through all of the various pages/interfaces of the application. They will then confirm it follows the [Google's material design guidelines](#).

## 2. test-LF-A1-2

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: Users will be shown the application and be asked if it is visually appealing and intuitively designed.

Output/Result: 90 % of users will say that the application is visually appealing and intuitively designed.

How test will be performed: A focus of group of users will be shown the application after its development is complete. They will be then given a usability survey (located in the appendix). During this usability survey they will be asked if they find the interface of the application visually appealing and intuitively designed.

### 3.2.2 Usability and Humanity Requirements

This section contains the tests for the Usability and Humanity Requirements.

## 1. test-UH-E1-1

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: The user will be asked to navigate to one of the main features of the application.

Output/Result: The user will be able to navigate to the function within 5 taps.

How test will be performed: A focus group of users will be given a phone running the application. They will be asked to navigate to one of the main features of the application. The number of taps to navigate to the asked feature will be recorded.

2. test-UH-E1-2

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: Users will be shown the application and be asked if it they find the application easy to navigate.

Output/Result: 90 % of users will say that the application is easy to navigate.

How test will be performed: A focus of group of users will be shown the application after its development is complete. They will be then given a usability survey (located in the appendix). During this usability survey they will be asked if they find the application easy to navigate.

3. test-UH-P1

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input: The developers will observe the interface of the application.

Output: The developers of the application will observe that the interface is in Canadian English.

How test will be performed: Once the development of the application is complete, the developers will go through all of the various pages/interfaces of the application. They will then confirm that they are in Canadian English.

4. test-UH-L1-1

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input: The user will be given an overview of the application and a walkthrough of a specific feature of the application in 30 minutes. The user will be then asked to use that feature without assistance.

Output: The user will be able to use that feature effectively without external assistance.

How test will be performed: A focus group of users will be given a phone running the application. They will be given an overview of the application and a walkthrough of a specific feature of the application in 30 minutes. They will be then asked to use that feature without assistance.

#### 5. test-UH-L1-2

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: Users will be shown the application in a walkthrough and be asked to use it. Afterwards they will be asked if it they found the application easy to learn.

Output/Result: 90 % of users will say that the application is easy to learn.

How test will be performed: A focus of group of users will be shown the application after its development is complete. They will be then given a usability survey (located in the appendix). During this usability survey they will be asked if they find the application easy to learn.

#### 6. test-UH-A1

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: The developers will observe the interface of the application.

Output/Result: The developers of the application will confirm the interface uses [android accessibility features](#).

How test will be performed: Once the development of the application is complete, the developers will go through all of the various

pages/interfaces of the application. They will then confirm the interface uses [android accessibility features](#).

### 3.2.3 Performance Requirements

This section contains the tests for the Performance Requirements.

1. test-P-SL1

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: The developers will observe the interface of the application.

Output/Result: The developers of the application will go through all of the various pages/interfaces of the application. They will confirm the response time to user input is within 0.5 s for all interactions.

How test will be performed: Once the development of the application is complete, the developers will go through all of the various pages/interfaces of the application. They will record the response time for all possible user interactions.

2. test-P-PA1

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: The developers will use the bill-splitting feature of the application.

Output/Result: The output of the bill-splitting feature will be accurate to two decimal places.

How test will be performed: Once the development of the application is complete, the developers will use the bill-splitting feature of the application with multiple monetary amounts. The output of bill-splitting feature will be accurate to two decimal places.

### 3. test-P-RFT1

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: The developers will disconnect the phone from the server. They will then attempt to use the features of the application.

Output/Result: The features of the application will continue to work for at least 10 minutes after losing connection.

How test will be performed: Once the development of the application is complete, the developers will disconnect the phone from the server. They will then attempt to use the features of the application. They will observe how the application performs without connection, and how it performs when connection is resumed.

### 4. test-P-C1

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: 100 users will be simulated trying to login to application servers.

Output/Result: The application servers will be able to process those login requests.

How test will be performed: Once the development of the application is complete, the developers will simulate 100 people trying to use the application at the same time. The application servers should show no slow down.

## 3.2.4 Operational and Environmental Requirements

This section contains the tests for the Operational and Environmental Requirements.

### 1. test-OE-PE1

Type: Non-functional, Dynamic, Manual



Initial State: The application is run on a phone with Android OS.

Input/Condition: The developers will observe the application on multiple different phones running Android OS.

Output/Result: The application will be able to run.

How test will be performed: Once the development of the application is complete, the developers will confirm the application runs on multiple different phones running Android OS.

## 2. test-OE-PR1

Type: Non-functional, Dynamic, Manual

Initial State: The development of the application is complete.

Input/Condition: The developers will submit the app for the Google Play Store.

Output/Result: The application will be available on the Google Play Store.

How test will be performed: Once the development of the application is complete, the developers will submit the application to the Google Play Store.

### 3.2.5 Maintainability and Support Requirements

This section contains the tests for the Maintainability and Support Requirements.

## 1. test-M-M1

Type: Non-functional, Dynamic, Manual

Initial State: The development of the application is complete.

Input/Condition: The developers will check the [GitHub](#) for the Housemates application.

Output/Result: The documentation for the application will be available under the docs folder.

How test will be performed: Once the development of the application is complete, the developers will confirm that the documentation for the application is in the [GitHub](#) for the Housemates application.

### 3.2.6 Security Requirements

This section contains the tests for the Security Requirements.

#### 1. test-S-A1

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: The developers will try to access one account's user data from another user account.

Output/Result: The user data will not be accessible from other user accounts.

How test will be performed: Once the development of the application is complete, the developers will try to access one account's user data from another user account. The user data should not be accessible.

#### 2. test-S-IN1

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: The developers will attempt to introduce incorrect data into the input fields of the application.

Output/Result: The application will prevent the incorrect data from being entered into the database.

How test will be performed: Once the development of the application is complete, the developers will go through all the possible input fields of the application and try to input incorrect data. The application should prevent this from occurring.

#### 3. test-S-P1

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: The developers will launch the application.

Output/Result: The application will show the information collection policy of the application.

How test will be performed: Once the development of the application is complete, the developers will launch the app and confirm it shows the information collection policy of the application on first launch.

### 3.2.7 Compliance Requirements

This section contains the tests for the Compliance Requirements.

#### 1. test-C-SC1

Type: Non-functional, Dynamic, Manual

Initial State: The application is run on a phone.

Input/Condition: The developers will observe the application.

Output/Result: The application will follow the [Google Play development standards](#).

How test will be performed: Once the development of the application is complete, the developers will run the app and confirm it follows the [Google Play development standards](#).

### 3.3 Traceability Between Test Cases and Requirements

FR ID	TM1	TM2	TM3	TM4	TM5	BM1	BM2	BM3	BM4	BM5
test-TM1-1	×									
test-TM2-1		×								
test-TM3-1			×							
test-TM3-2			×							
test-TM4-1				×						
test-TM4-2				×						
test-TM5-1					×					
test-TM5-2					×					
test-BM1-1/2-1						×	×			
test-BM3-1								×		
test-BM4-1									×	
test-BM5-1										×

Table 1: **Functional Requirements Traceability Part 1**

FR ID	BM6	BM7	SS1	SS2	SS3	AS1	AS2	AS3	AS4	AS5
test-BM6-1	×									
test-BM7-1		×								
test-SS1-1			×							
test-SS2-1				×						
test-SS3-1					×					
test-AS-1-1						×				
test-AS-1-2						×				
test-AS-1-3						×				
test-AS-2							×			
test-AS-3								×		
test-AS-4									×	
test-AS-5										×

Table 2: **Functional Requirements Traceability Part 2**

NFR ID	LF-A1	UH-E1	UH-P1	UH-L1	UH-A1	P-SL1	P-PA1	P-RFT1
test-LF-A1-1	×							
test-LF-A1-2	×							
test-UH-E1-1		×						
test-UH-E1-2		×						
test-UH-P1			×					
test-UH-L1-1				×				
test-UH-L1-2				×				
test-UH-A1					×			
test-P-SL1						×		
test-P-PA1							×	
test-P-RFT1								×

Table 3: **Non-Functional Requirements Traceability Part 1**

NFR ID	P-C1	OE-PE1	OE-PR1	M-M1	S-A1	S-IN1	S-P1	C-SC1
test-P-C1	×							
test-OE-PE1		×						
test-OE-PR1			×					
test-M-M1				×				
test-S-A1					×			
test-S-IN1						×		
test-S-P1							×	
test-C-SC1								×

Table 4: **Non-Functional Requirements Traceability Part 2**

## 4 Unit Test Description

This section will be completed after the creation of the design document.

### 4.1 Unit Testing Scope

N/A

## **4.2 Tests for Functional Requirements**

N/A

## **4.3 Tests for Nonfunctional Requirements**

N/A

## **4.4 Traceability Between Test Cases and Modules**

N/A

## 5 Appendix

This is where you can place additional information.

### 5.1 Symbolic Parameters

N/A

### 5.2 Usability Survey Questions

The following usability survey would be distributed by Google Forms to potential stakeholders after a showcase of a preliminary version of the Housemates application. The information in this usability survey will be used in tests in the VnV plan as well as collecting information on potential stakeholders.

1. Do you have housemates?
  - (a) Yes
  - (b) No
2. What operating system is run on your phone?
  - (a) IOS
  - (b) Android
  - (c) Others
3. What do you find frustrating about having housemates?
4. Which feature of the Housemates application do you find the most interesting/would use the most often?
  - (a) Split bills
  - (b) Delegate chores
  - (c) Schedule events
5. Did you find the Housemates application visually appealing and intuitive?

(a) Yes

(b) No

6. Did you find the Housemates application easy to navigate?

(a) Yes

(b) No

7. Did you find the features of the Housemate application easy to learn?

(a) Yes

(b) No

8. Would you use the Housemates application in your regular life?

(a) Yes

(b) No



## 5.3 Reflection

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

1. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage etc. You should look to identify at least one item for each team member.
2. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

### 5.3.1 Knowledge and Skills

- The team will need to learn how to use GitHub actions in order to be able to run automatic tests during CI/CD.
- The team needs to acquire dynamic testing knowledge in order to perform the tests in this VnV plan.
- The team needs to acquire usability testing knowledge in order to perform the tests in this VnV plan.
- The team needs to acquire static testing knowledge in order to perform the tests in this VnV plan.
- The team needs to acquire domain specific knowledge on React testing.

### 5.3.2 Approaches

- GitHub Actions
  - Watch/read online tutorials on GitHub Actions.
  - Re-watch GitHub Actions tutorial for SFWRENG 4G06.

Fady Morcos: The approach chosen to acquire the GitHub skills/action I need to be successful here are to research, read and watch online tutorials on GitHub actions to reinforce my GitHub knowledge. Another thing I will do is re-watch the GitHub actions tutorial posted for the SFWRENG 4G06 course as this should have reliable information summarized for all of my GitHub needs to be successful.

- Dynamic testing
  - Research dynamic testing methods online.
  - Look at SFWRENG 3S03: Software Testing notes on dynamic testing.

Rizwan Ahsan: The approach I will choose for learning dynamic testing is researching online to find good dynamic testing techniques. This will give me in-depth knowledge about dynamic testing techniques and allow me to find the best method to perform dynamic testing for our project.

- Static testing
  - Research static testing methods online.
  - Look at SFWRENG 3S03: Software Testing notes on static testing.

Fardeen Afsar: For static testing, I'll apply the techniques and methods introduced in SFWRENG 3S03, while focusing on usability in code and documentation review. This ensures early identification of defects and adherence to standards. For static testing, I will also opt for a method focused on acquiring knowledge about static testing through online research.

- Usability testing
  - Look at testing/research methods from SFWRENG 4HC3: Human Computer Interfaces.
  - Research online methods for usability testing.

Justin Dang: The approach I chose for learning usability testing is taking methods used in SFWRENG 4HC3. This is because for 4HC3 we have had a focus on design for usability.

- Domain knowledge on React testing
  - Watch React testing tutorials.
  - Look at [React testing website](#).

Harris Hamid: The approach I'm choosing to actually create test cases for our React web app will be to use jest, a JavaScript testing framework. I'll look at examples at how it is being used in actual production level code.