

Design/Test Document

Ventilation Calculator App Project

Course: Design and Implementation of Mobile Applications

Professor: Luciano Baresi

Students:

- Felipe Azank 10919711
- Felipe Bagni 10912321

Table of contents

1. Document Overview	3
1.1 Purpose	3
1.2 Scope	3
1.2.1 Core Features	3
1.2.2 Target Audience	3
1.2.3 Platform	3
1.2.4 Boundaries and Context	4
1.2.5 Expected Outcome	4
1.3 References	4
2. System Overview	4
2.1 Project Description	4
2.1.1 Problem Statement	4
2.1.2 Unique Features	5
2.1.3 Development Context	5
2.1.4 Project Constraints	5
2.1.5 High-Level Goals	5
2.2 High-Level Architecture	5
3. Design Document	6
3.1 Design Goals	6
3.2 Functional Requirements	7
3.3 Non-Functional Requirements	8
3.4 User Interface Design	8
3.5 System Design	14
3.6 Technology Stack	15
4. Test Plan	16
4.1 Testing Goals	16
4.2 Unit Tests	16
4.3 Integration Tests	16
4.4 Acceptance Tests	17
4.5 Test Cases	17
5. Next Steps and Final Remarks	18

1. Document Overview

1.1 Purpose

The purpose of this document is to outline the design and testing framework for the Flutter application developed in partnership with the World Health Organization (WHO). The application aims to estimate indoor ventilation in a simple and accessible manner, making it easier for individuals to assess the safety of their living and working environments against respiratory infections. This aligns with the guidelines provided in the WHO's "Roadmap to improve and ensure good indoor ventilation in the context of COVID-19."

The document describes how the idea of the app was structured, how the design as well as the "look and feel" of the pages were done and how the implementation in flutter (along with the tests was made).

The github repository of the project as well as its current releases can be located here:

[AZANK7173/VentilationCalculator App](https://github.com/AZANK7173/VentilationCalculator_App)

1.2 Scope

This document covers the design and testing of a Flutter application developed in partnership with the World Health Organization (WHO). The application aims to provide a simple and accessible method for users to estimate indoor ventilation quality and assess whether it meets the United Nations (UN) guidelines for safe indoor spaces.

1.2.1 Core Features

The app enables users to input key details such as:

- Room dimensions.
- Information about windows and doors (e.g., size, type, and state).
- Meteorological conditions (e.g., temperature and wind speed).

Based on this data, the app calculates the ventilation quality of the space and determines if it adheres to the WHO's recommendations for indoor safety.

1.2.2 Target Audience

The primary audience for the app is the general public, particularly individuals who want to evaluate how safe their living or working spaces are from infectious respiratory diseases. Additionally, healthcare workers can benefit from the app's inclusion of hospital-specific guidelines for ventilation.

1.2.3 Platform

The app is built using Flutter, making it compatible with both Android and iOS platforms, ensuring a broad reach and ease of access, a important requirement from the WHO proposition.

1.2.4 Boundaries and Context

The app provides a rough estimation of ventilation quality, intended for users who might not seek professional assessments. While the app empowers individuals with valuable insights, it is not a replacement for professional evaluations and should not be used for legally binding or critical safety certifications.

1.2.5 Expected Outcome

The app aims to raise awareness about the importance of proper indoor ventilation in preventing respiratory infections. By making this information easily accessible, it encourages users to take proactive steps toward creating safer environments, thereby contributing to public health awareness and prevention efforts.

1.3 References

1. **Roadmap to Improve and Ensure Good Indoor Ventilation in the Context of COVID-19**

This document, published by the World Health Organization (WHO), served as the primary reference for defining the app's frontend design and core mechanisms for calculating ventilation quality.

Link: <https://www.who.int/publications/i/item/9789240021280>

2. **Course Materials**

The materials provided during the college course were fundamental for the development of the application, including the implementation of its features, adherence to Flutter development best practices, and the design and execution of testing methodologies.

3. **Flutter Docs**

The information on Flutter documentation provided comprehensive guides, tutorials, and API references. It helped to understand the framework's features, learn best practices, and troubleshoot issues.

Link: <https://docs.flutter.dev/>

2. System Overview

2.1 Project Description

The Flutter application was developed in partnership with the World Health Organization (WHO) to address the increasing need for tools to assess indoor ventilation quality. This need became particularly significant after the COVID-19 pandemic, which underscored the health risks associated with poorly ventilated spaces.

2.1.1 Problem Statement

Proper indoor ventilation is crucial for minimizing health risks, particularly in the context of respiratory infections. However, assessing ventilation quality often requires expert evaluations or costly tools, making it inaccessible for many individuals.

2.1.2 Unique Features

The app provides a simple and intuitive solution for estimating ventilation quality. Its user-friendly design ensures that anyone can input basic information about their space and receive a reliable estimate of ventilation quality without requiring professional expertise or equipment.

2.1.3 Development Context

This app is part of a broader initiative to raise awareness about the importance of indoor ventilation in preventing health risks. It implements calculations and guidelines previously established by the United Nations (UN) and WHO, making these valuable resources accessible to the general public.

2.1.4 Project Constraints

The app faced no significant constraints during its development phase, allowing for a clear focus on usability and functionality.

2.1.5 High-Level Goals

The primary objectives of the project are to educate the public about the importance of proper ventilation, encourage awareness of related health risks, and provide a simple, accessible solution for evaluating ventilation quality in everyday spaces.

2.2 High-Level Architecture

The overall structure of the application consists in a two-tier (Client/Server) architecture. Containing a Presentation Layer and a Calculation Layer. As part of WHO's requirements, we did not store data in database systems nor save them online for future exploration or analysis.

The application contains three main components, described below:

- **Presentation Layer (Client-side):** The layer consists of a representation of the functionality offered to the user. The User can interact with it to get the information he needs. For this the Interface will process the inputs and forward them to the Calculation Layer.
- **API Layer (Server-side):** This layer represents the interaction of the app with external resources in order to retrieve 2 types of information: outside temperature and wind speed, both based on the geolocation of the user. However, this step can be skipped and the person can also input the value directly in case they do not want to allow tracking.

- **Calculation Layer (Server-side):** This layer gets all inputs obtained in the previous 2 layers and performs the computations necessary to obtain the ventilation rates and compare them with the WHO guidelines. It also computes how much ventilation improvement is necessary to match the amount of people the User wants to fit in the room.

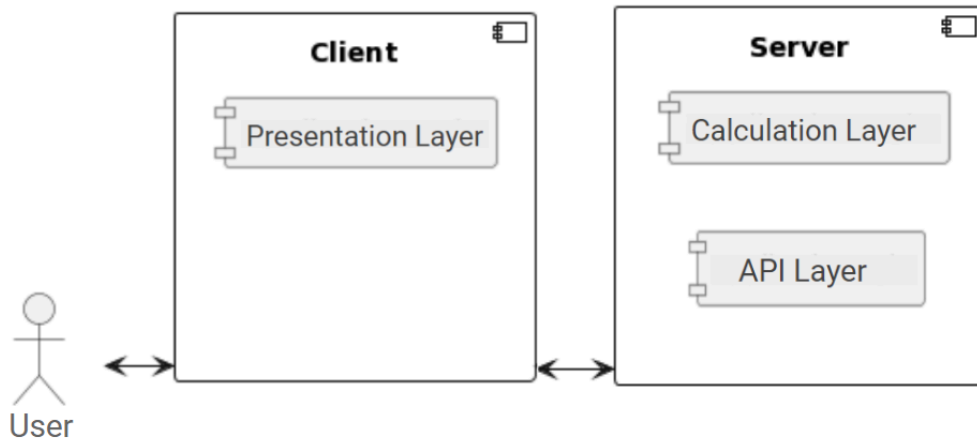


Figure 1: High level architecture of the Application

3. Design Document

3.1 Design Goals

The design goals were developed based mostly on the WHO's inputs obtained by some meetings. The main ones, as interpreted by the group were:

3.1.1 Usability

The app is designed with simplicity and user-friendliness in mind, ensuring that individuals without technical expertise can easily navigate and use its features. By providing a clean and intuitive interface, the app enables users to input essential data such as room dimensions, window details, and meteorological information with minimal effort. Clear instructions and visual elements enhance the user experience, making the app accessible to a broad audience.

3.1.2 Accessibility

Accessibility is a key priority for the app, ensuring that it can be used by individuals with diverse needs. The app follows Flutter's best practices for accessibility, including support for screen readers, high contrast modes, and scalable widgets. This ensures that users with visual or motor impairments can access and benefit from the app's functionalities without barriers. A design choice made by the team was also to focus on simple data input formats and to limit the amount of images used.

3.1.3 Maintainability

The app's codebase is structured to be modular and well-documented, making it easy for developers to update or extend its features in the future, given that, as discussed with the WHO team, this app can be developed further with the objective of a future real deployment by the UN. Using Flutter and Dart ensures a single codebase for both Android and iOS platforms, reducing maintenance overhead and allowing for streamlined debugging, testing, and deployment. This maintainability is crucial for incorporating future updates or adhering to evolving guidelines.

3.1.4 Portability

The app leverages Flutter's cross-platform capabilities, allowing it to run seamlessly on both Android and iOS devices with a single codebase. This approach ensures consistent functionality and design across platforms, reducing development time and resource requirements. By adhering to platform-specific guidelines and optimizing performance for different devices, the app delivers a smooth user experience regardless of the operating system or device specifications, which may increase the range of users the app can obtain.

3.2 Functional Requirements

After the meetings with WHO and the outlining of the app, the functional requirements for the development of the application can be summarized by the table below

Requirement	Description
R1	The app must allow user input (room and window dimensions, and meteorological data in case the user does not want to provide geolocation)
R2	The app must process user inputs using predefined formulas based on UN and WHO guidelines to estimate ventilation quality.
R3	The app must compare the calculated ventilation against WHO standards and inform users whether their space meets the recommended guidelines.
R4	The app must present results in an easy-to-understand format, such as a numerical ventilation score and how much to improve it.
R5	The app must present an easy-to-understand input format, that is accessible and easy to interpret
R6	The app must function smoothly on both Android and iOS devices.
R7	The app must validate user inputs and display error messages if invalid data is entered (e.g., negative dimensions).
R8	The app must function smoothly on both Android and iOS devices.



R9	The app functioning should be straightforward and help the user understand which ventilation case they're located in
----	--

3.3 Non-Functional Requirements

Requirement	Description
Accessibility	The app must meet accessibility standards (e.g., WCAG 2.1 AA), including support for screen readers.
Portability	The app must deliver consistent functionality and appearance on both Android and iOS devices without platform-specific bugs or glitches.
Scalability	The app's architecture must support the addition of new features, such as language change and a notification system
Usability	The app must adhere to Flutter's Material Design guidelines to ensure a consistent user experience.
Security	The app must ensure that no user data is stored and it's not managed and not accessible by unauthorized parties.

3.4 User Interface Design

After gathering the requirements and the outline of the app. The design was built in Figma, in order to produce professional looking screens (which were then validated by the WHO team).

Given that the overall structure of the app did not change significantly, the Figma screens are almost a carbon copy of the final screens developed in Flutter. And the overall functioning and use case of the app can be understood by them:

3.4.1 Introduction Screen and Instruction

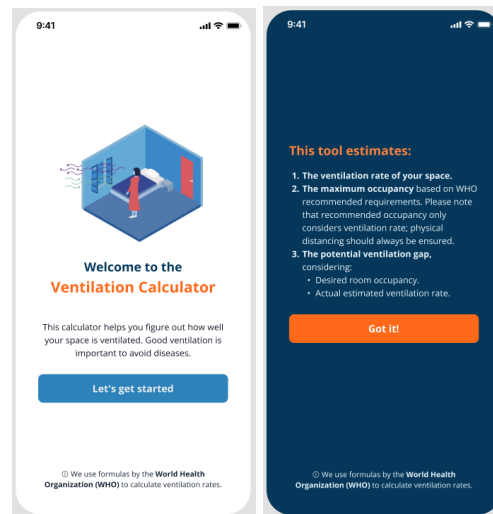
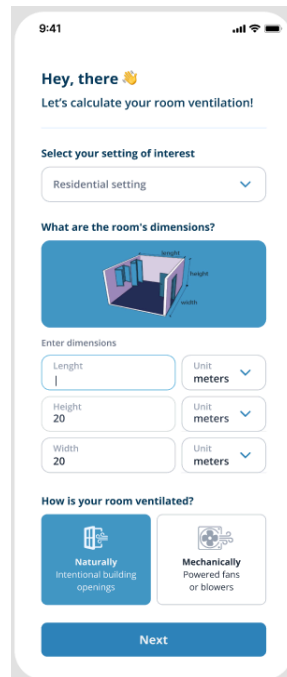


Figure 2: first and second screen showing the overall instructions

The first and second screen setup the overall color scheme and style of the app, while informing how the user should proceed on interacting with the app in order to get the desired information about ventilation.

3.4.2 First Input screen

The first input screen asks the 2 most important questions: which is the room setting that is being calculated (hospital, residential or commercial). And which type of ventilation the room is subjected (mechanical, which leads to asking information about the ventilators, or natural, that leads to a screen about the windows of the room).



9:41

Hey, there 🙌
Let's calculate your room ventilation!

Select your setting of interest

Residential setting

What are the room's dimensions?

Enter dimensions

Length: 1 Unit: meters

Height: 20 Unit: meters

Width: 20 Unit: meters

How is your room ventilated?

Naturally
Intentional building openings

Mechanically
Powered fans or blowers

Next

Figure 3: first input screen

3.4.3 Natural Ventilation - Second input screen

If the natural ventilation option was selected, the next screen will require information on the windows. There are requirements in all input fields to avoid impossible measurement numbers.

9:41

Natural Ventilation

Openings Characteristics

We are now referring to the **facade toward the exterior**. Note that several typologies of opening could be available on the same wall (i.e. two different types of window or one window and one door).

What does "typologies of opening" mean? [see more](#)

Door/window number 1

Number of openings with the same size:

How much do you usually open it?

Enter dimensions

Does it have a mosquito net? ☐

Door/window number 2

Number of openings with the same size:

How much do you usually open it?

Enter dimensions

Does it have a mosquito net? ☒

[+ Add new opening type](#)

Next

Figure 4: second input screen

3.4.4 Natural Ventilation - Third screen and cross/single selection

9:41

Natural Ventilation

Openings Characteristics

Are there openings available in other side of the room? ☐

Wind speed

Enter the average wind speed at your building's location. This should be the average value at the building height, away from any obstructions.

Ⓜ If no data is available, enter "1".

See results

Figure 5: screen for single or cross section

In this screen, if the option of “is there openings on the other side?” is selected, then this will take the user to the “cross ventilation” option, that would require the wind speed and the input data of the windows from the other side. However, if the option is not selected, then the case is “single-sided” ventilation, where the data from Temperature is needed.

3.4.5 Natural Ventilation - Temperature Input data

Opening Characteristics - Temperature Data

Temperature Data

Please Enter the temperature data for the location of the building as well as the overall temperature inside the room.

① If no data is available, enter "1".

Inside Temp °C ▼

Location permission denied: enter manually the Outside Temperature.

Outside Temp °C ▼

See Result for Single Opening

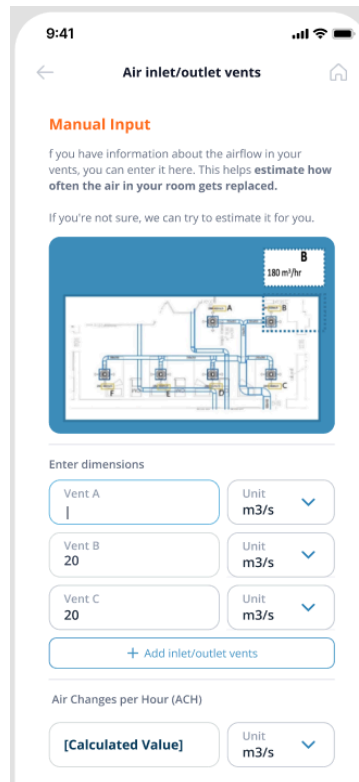
Figure 6: Temperature input screen

For single-sided ventilation, the user is then required to provide the temperature, which can either be accessed via API, or entered manually. In the case above, we see the screen when the API request is refused by the user.

After this, the result page shows up, which will be shown later on.

3.4.6 Mechanical Ventilation - Second input screen

Going back to the first input structure, if the mechanical ventilation is selected, then the following screen shows up, where the user needs to input the data from the ventilators present in the room.



9:41

← Air inlet/outlet vents

Manual Input

If you have information about the airflow in your vents, you can enter it here. This helps **estimate how often the air in your room gets replaced**.

If you're not sure, we can try to estimate it for you.

180 m³/hr

Enter dimensions

Vent A Unit m³/s

Vent B 20 Unit m³/s

Vent C 20 Unit m³/s

+ Add inlet/outlet vents

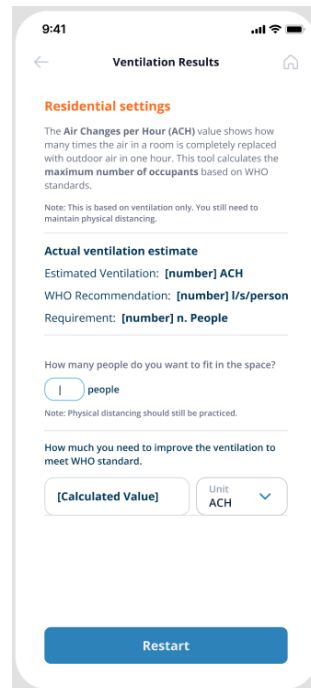
Air Changes per Hour (ACH)

[Calculated Value] Unit m³/s

Figure 7: Mechanical ventilation input screen

3.4.7 Results Screen

After obtaining the data for any case described above, the result screen will then show the calculated value of the ventilation, the WHO guidelines for the space, how many people you can fit in the room safely, and how much improvement on ventilation the user would need to make in order to fit the amount of people desired.



9:41

Ventilation Results

Residential settings

The Air Changes per Hour (ACH) value shows how many times the air in a room is completely replaced with outdoor air in one hour. This tool calculates the maximum number of occupants based on WHO standards.

Note: This is based on ventilation only. You still need to maintain physical distancing.

Actual ventilation estimate

Estimated Ventilation: [number] ACH

WHO Recommendation: [number] l/s/person

Requirement: [number] n. People

How many people do you want to fit in the space?

1 people

Note: Physical distancing should still be practiced.

How much you need to improve the ventilation to meet WHO standard.

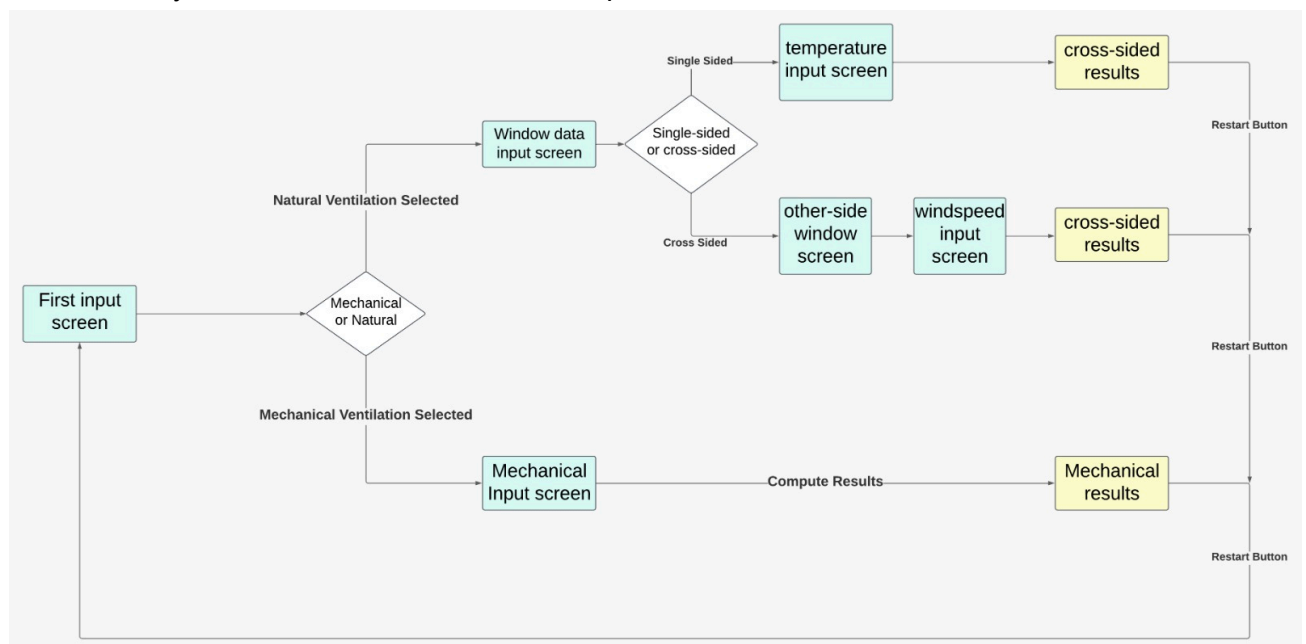
[Calculated Value] Unit: ACH

Restart

Figure 8: Results screen

3.5 System Design

The overall system is based on the a decision process held below:



As described in the 2.2 section, the components interact a limited number of times:

1. The user uses the Presentation Layer in order to upload the necessary data and “decisions” in the system.
2. Each of these inputs are saved as variables in the `state_manager` and accessed as global variables around the *provider* library.
3. By accessing these variables, the system is able to provide the following screens based on the decision flow from
4. At the result screens, the data is accessed and the necessary computations are runned at the time of compilation.

3.6 Technology Stack

3.6.1 Development Framework and Language

- **Flutter:** The primary SDK used for building the cross-platform mobile application. Flutter allows for a single codebase that runs on both Android and iOS devices.
- **Dart:** The programming language used for writing the application code, as Flutter is built with Dart.

3.6.2 Design Tool

- **Figma:** Used for UI/UX design, Figma facilitated the creation of high-fidelity interactive prototypes and the design of application screens. It enabled seamless collaboration between designers and developers throughout the design process.

3.6.3 Libraries and Dependencies

The application leverages several third-party libraries to implement various features and functionalities, such as:

- **flutter_dotenv (v5.0.2):** Used for managing environment variables and configuration.
- **cupertino_icons (v1.0.6):** Provides iOS-style icons to be used within the application.
- **url_launcher (v6.2.6):** Enables the app to launch URLs in the mobile platform's web browser.
- **carousel_slider (v4.2.1):** Implements image carousel sliders for a user-friendly interface.
- **provider (v6.0.3):** A state management library that helps with managing app state and notifying widgets of changes.
- **geolocator (v12.0.0):** Provides geolocation capabilities for accessing the current location of the device.
- **http (v1.2.2):** Facilitates making HTTP requests to communicate with external services or APIs.

3.6.4 Development Dependencies

For development, testing, and code analysis, the following dependencies are used:

- **flutter_test:** Comes with the Flutter SDK and provides testing capabilities to ensure code quality.

- **flutter_lints (v3.0.0):** Offers a recommended set of lints to promote best coding practices and maintain code quality.

3.6.5 Version Control

- **Git/GitHub:** Utilized for source code management, version control, and collaborative features to track the development progress of the application.

3.6.6 Emulators

- **Android Studio Emulators:** Utilized for testing and simulating the application on various Android devices without the need for physical hardware.
-

4. Test Plan

4.1 Testing Goals

For the testing process, a lot of trials were designed in order to assess the quality of the app. As described in the requirements section, the usability was one of the most important requirements from the WHO team, meaning that this was the most tested feature.

In addition, many tests on the Widgets developed were designed as well in order to make sure that the coding of elements follow the flutter guidelines as well as the overall structure presented in class.

4.2 Unit Tests

In order to assess the correctness of the objects, we designed Unit Tests for every screen in order to make sure the Widgets developed were done correctly. The special library “flutter_test” was used in order to perform all the Widget tests conducted.

4.3 Integration Tests

Given that the app does not contain a proper database system, the integration tests were simplified and the main integration assessment was made by checking if the file system of the application would work. This was done by changing the file path of the app images on tests and checking if it worked. In addition the API used to obtain wind speed and temperature based on the location was also assessed

4.4 Acceptance Tests

The most important requirements were evaluated in the acceptance test. Given that the main objective of the project was to develop an interface that should be easy to use for most people, the acceptance test was conducted by emulating the app and inviting people to test it.

Taking into consideration the [article](#) written by Jakob Nielsen, we tested the usability with 5 people that used the app for their own particular cases. After that, we conducted a quick conversation to understand the biggest problems faced. The results are better detailed in Section 4.5.

4.5 Test Cases

4.4.1 Unit Tests

ID: 1 to 10

Description: Unit tests developed to assess the widgets of the app. All of them were made using the “flutter_test” library and are located in the “test” folder of the app.

```
void main() {
  group('DropdownButtonExample Widget Tests', () {
    testWidgets('should display initial value if provided', (WidgetTester tester) async {
      await tester.pumpWidget(
        MaterialApp(
          home: Scaffold(
            body: DropdownButtonExample(
              items: ['Item 1', 'Item 2', 'Item 3'],
              initialValue: 'Item 2',
            ),
          ),
        ),
      );

      // Find the DropdownButton
      final dropdownfinder = find.byType(DropdownButton<String>);
      expect(dropdownfinder, findsOneWidget);

      // Verify the selected value is 'Item 2'
      final dropdownButton = tester.widget<DropdownButton<String>>(dropdownfinder);
      expect(dropdownButton.value, equals('Item 2'));
    });

    testWidgets('should allow changing dropdown value', (WidgetTester tester) async {
      await tester.pumpWidget(
        MaterialApp(
          home: Scaffold(
            body: DropdownButtonExample(
              items: ['Item 1', 'Item 2', 'Item 3'],
            ),
          ),
        ),
      );

      // Find the DropdownButton
      final dropdownfinder = find.byType(DropdownButton<String>);
      expect(dropdownfinder, findsOneWidget);

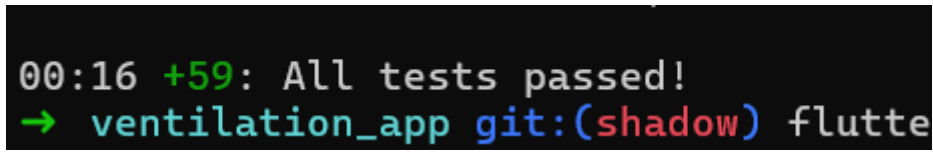
      // Tap the dropdown to open
      await tester.tap(dropdownfinder);
      await tester.pumpAndSettle();

      // Select a new value
      final newValuefinder = find.text('Item 3');
      await tester.tap(newValuefinder);
      await tester.pumpAndSettle();

      // Verify the new selected value
      final dropdownButton = tester.widget<DropdownButton<String>>(dropdownfinder);
      expect(dropdownButton.value, equals('Item 3'));
    });
  });
}
```

Figure 9: Unit Test example

Status: all of them passed, proving that the Widgets were developed correctly



```
00:16 +59: All tests passed!  
-> ventilation_app git:(shadow) flutter
```

Figure 10: Unit Test final result

4.4.2 Integration Test

ID: 11

Description: Test conducted in order to assert if the image would be correctly displayed if the image path changes.

Status: passed

ID: 12

Description: Test conducted to assert that the widget that is runned by the API works, was done using the flutter_test library as well.

Status: passed

4.4.2 Acceptance Test

ID: 13-17

Description:

Status: Passed, all potential users managed to get a reasonable result from their ventilation estimation.

Interview Observations and Notes:

- The users that selected the “mechanical ventilation” had a difficult time finding the ventilation rate of the machine, although it is printed in the machine, sometimes it's not visible.
- Estimating the room and window dimensions sometimes is a challenge for some people if the person has no measurement tool or technique for estimating it.

5. Next Steps and Final Remarks

In conclusion, it is possible to see that the application developed in fact matches the requirements expected by the WHO team, as well as the good practices of software development and app development using Flutter.

It is also possible to notice that this project has considerable room to have more features for the final users. Some of them were idealized after the interviews from the acceptance tests, such as:

1. Implementing a tool to add more languages to the app, making it more universal for non-english speakers.
2. Built-in features that help in the measurement process of the window, such as guides and tools that can help the user to better estimate the size of the room and windows.



3. Built-in features that help the user to identify ways to increase ventilation, and keep people safer from infectious diseases.

In the end, it is also possible to conclude that the project indeed helped in depth the developers to better understand the development of mobile applications, project management and software development in general.