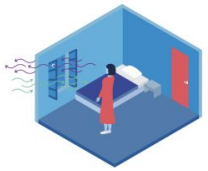
# Natural Ventilation Estimation Tool

## Final presentation

POLITECNICO
MILANO 1863



World Health
Organization

# Our Team

Felipe Azank
POLIMI

Felipe Bagni
POLIMI

POLITECNICO MILANO 1863

World Health Organization

# Project Customer

World Health Organization

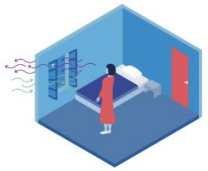
The United Nations agency working to promote health, keep the world safe and serve the vulnerable

Represented in this project by:

- Anna Silenzi
- Luca Fontana

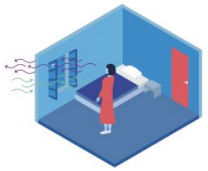
# Table of Contents





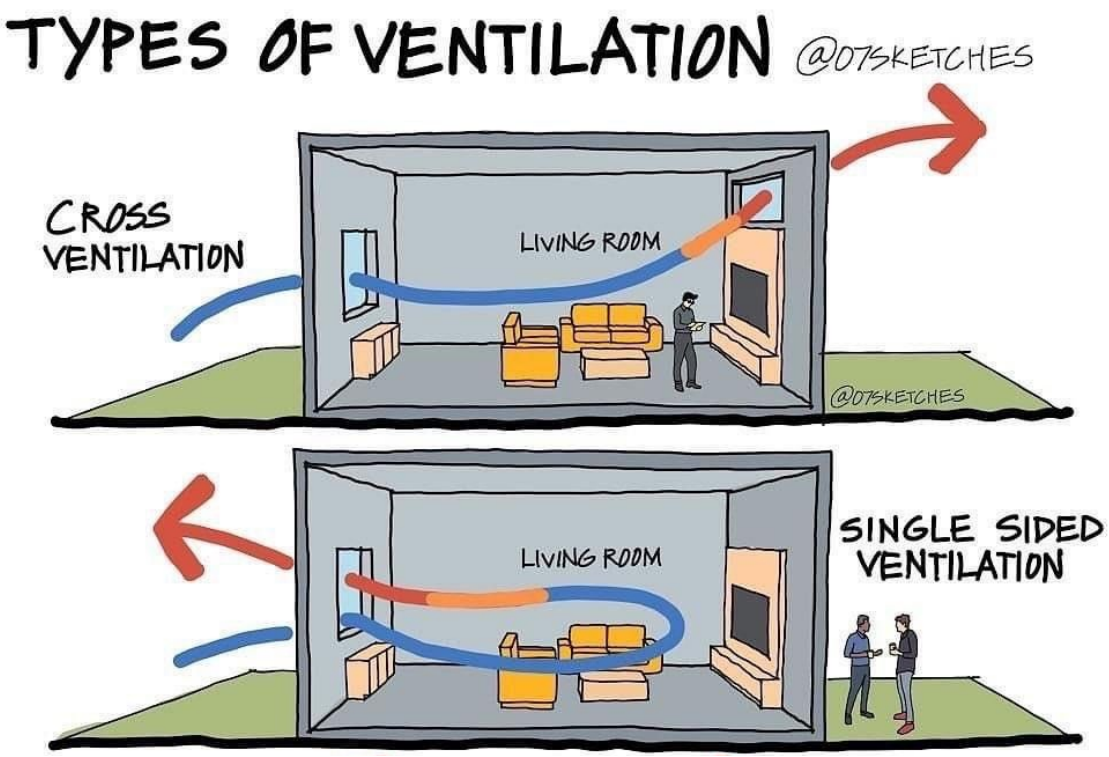
# Objectives and App Concept

Understanding the Problem, the Requirements and the what’s to be done



# Problem & Solution: Mission

- Proper natural ventilation plays a key role on controlling and preventing infections in healthcare.
- **Maximizing the amount of people able to use natural ventilation to fight infections** can play a huge role in everyday life and in possible upcoming epidemics.



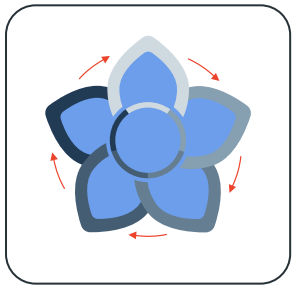

POLITECNICO MILANO 1863

World Health Organization

# Problem & Solution: Mission

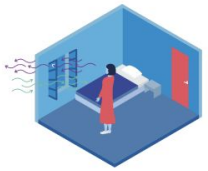
**UN guidelines - 2007:**

- Natural ventilation is considered for the first time among the effective measures to control infections in health care:
  "The guideline describes how an airborne precaution room and its adjacent areas can be designed to provide natural ventilation control of infections."



**Air Flow Estimation:**

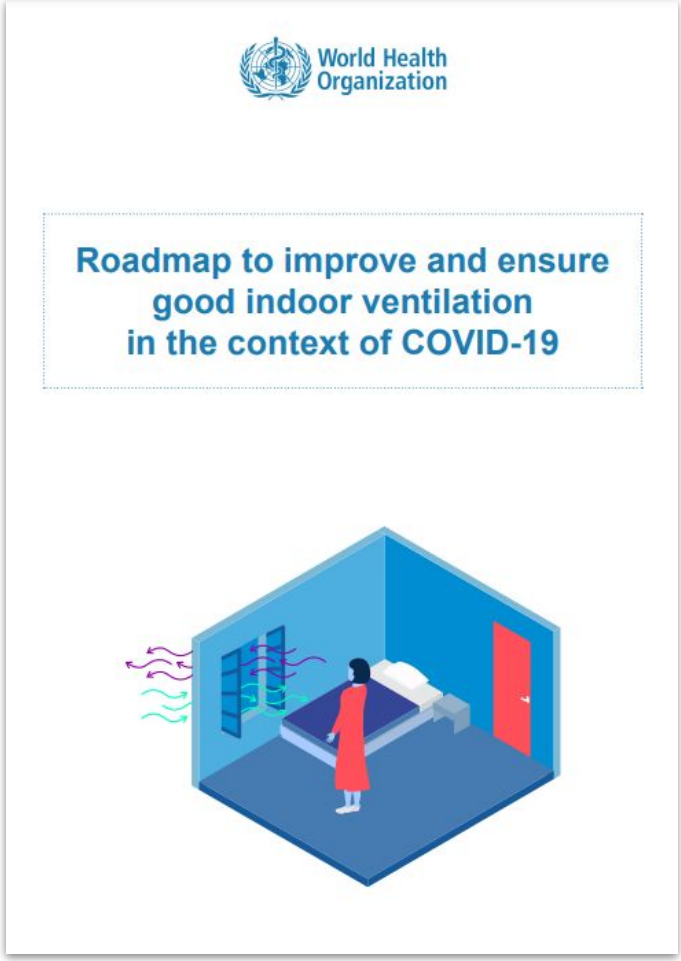
- Proper estimating the air flow and the risks on a natural ventilation system is complicated and requires specialist on the field, which is something most people don't have access to



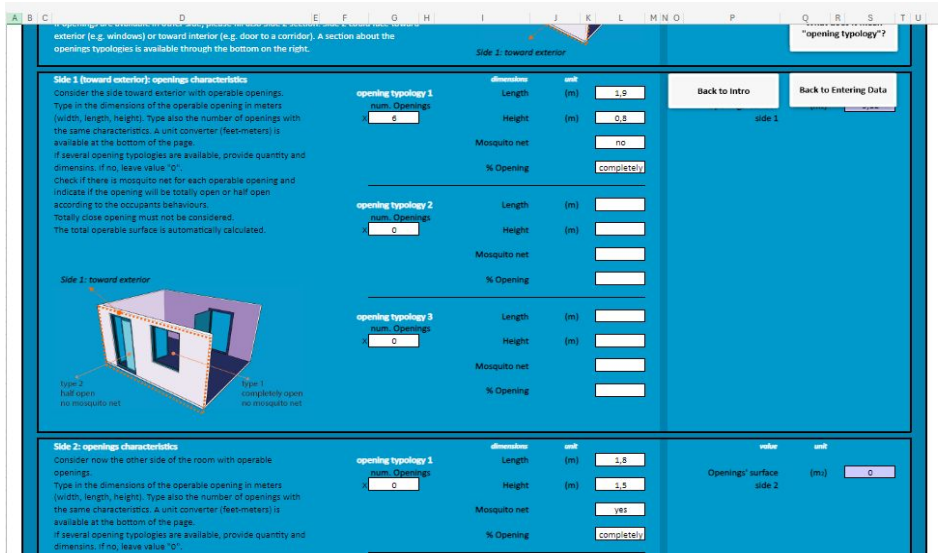
# Problem & Solution: Proposition

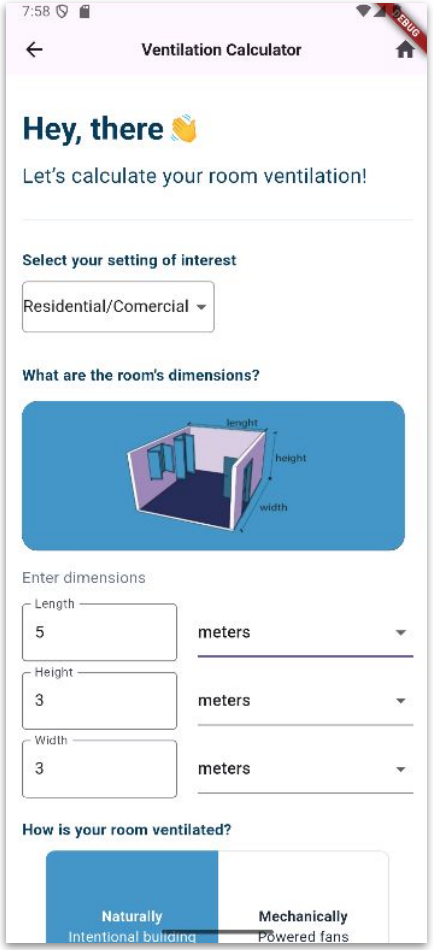
- Create a simple and democratic application that can help **non-specialized users** on correctively estimating the ventilation rate within an indoor environment.

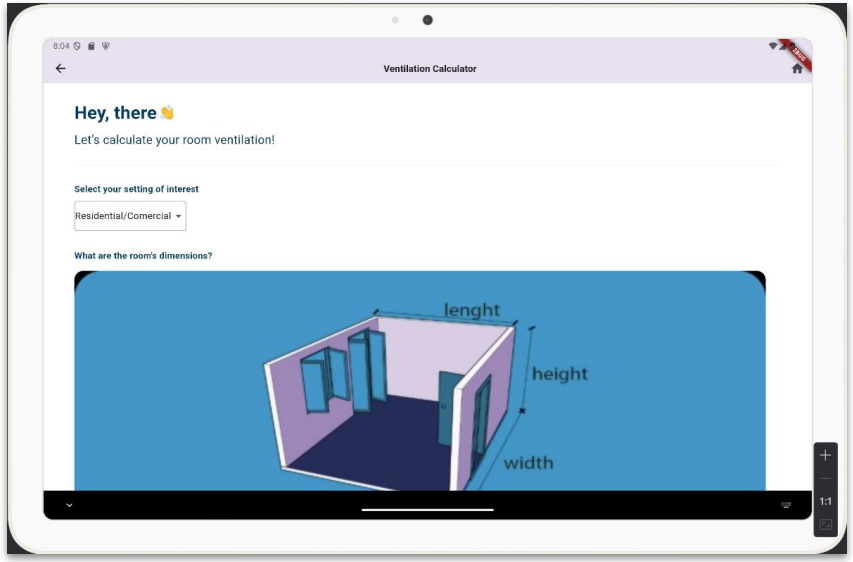
- This is a key step to consider the strategies to improve indoor ventilation as suggested in the document "**Roadmap to improve and ensure good indoor ventilation in the context of COVID-19**" from WHO.

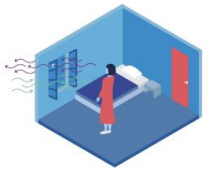


# Solution: job to be done



# Design Process

From the requirements definition, to the design setup and first screens

# App Development Plan

App **Requirements** creation and **Validation**

**Definition** of the overall **Design** of the App

**Creation** of the screens using **Figma**

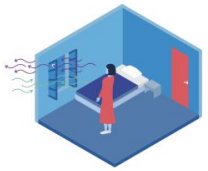
**Validation** of the **design** and **flow** of the app

**Implementation** of the screens in **Flutter**

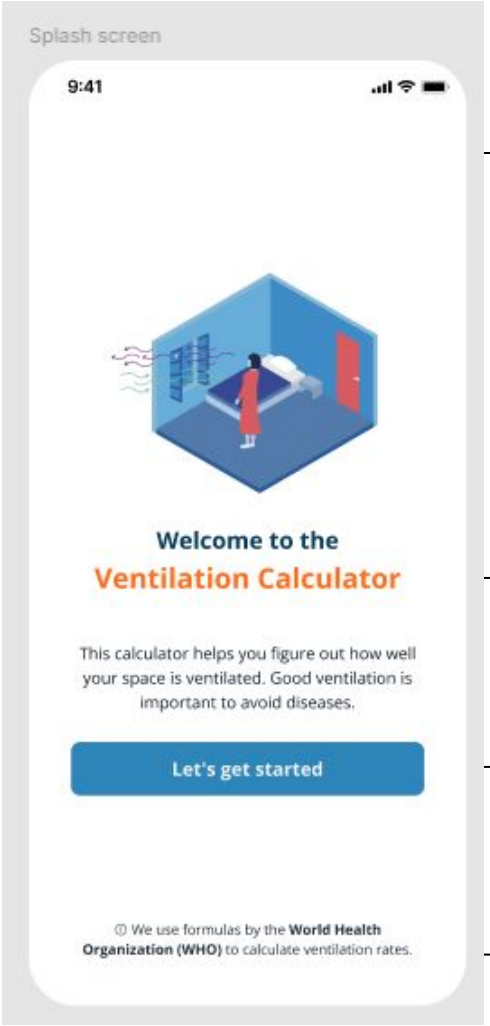
Implementation of the **API** and **Ventilation** calculations

**Documentation, testing** and **final adjustments**

**Deployment** of the **final version** of the application according to the requirements of WHO



# Overall Design of the App




Main Background color

Second Color for highlights

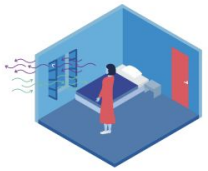
Main Color for buttons and inputs

Text Color in black

- Clear colors
- Simple layout
- Fonts that are easy to read
- Clear instructions

# Technology Stack

Main technologies used during development

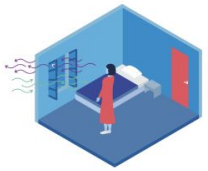


# Technology Stack

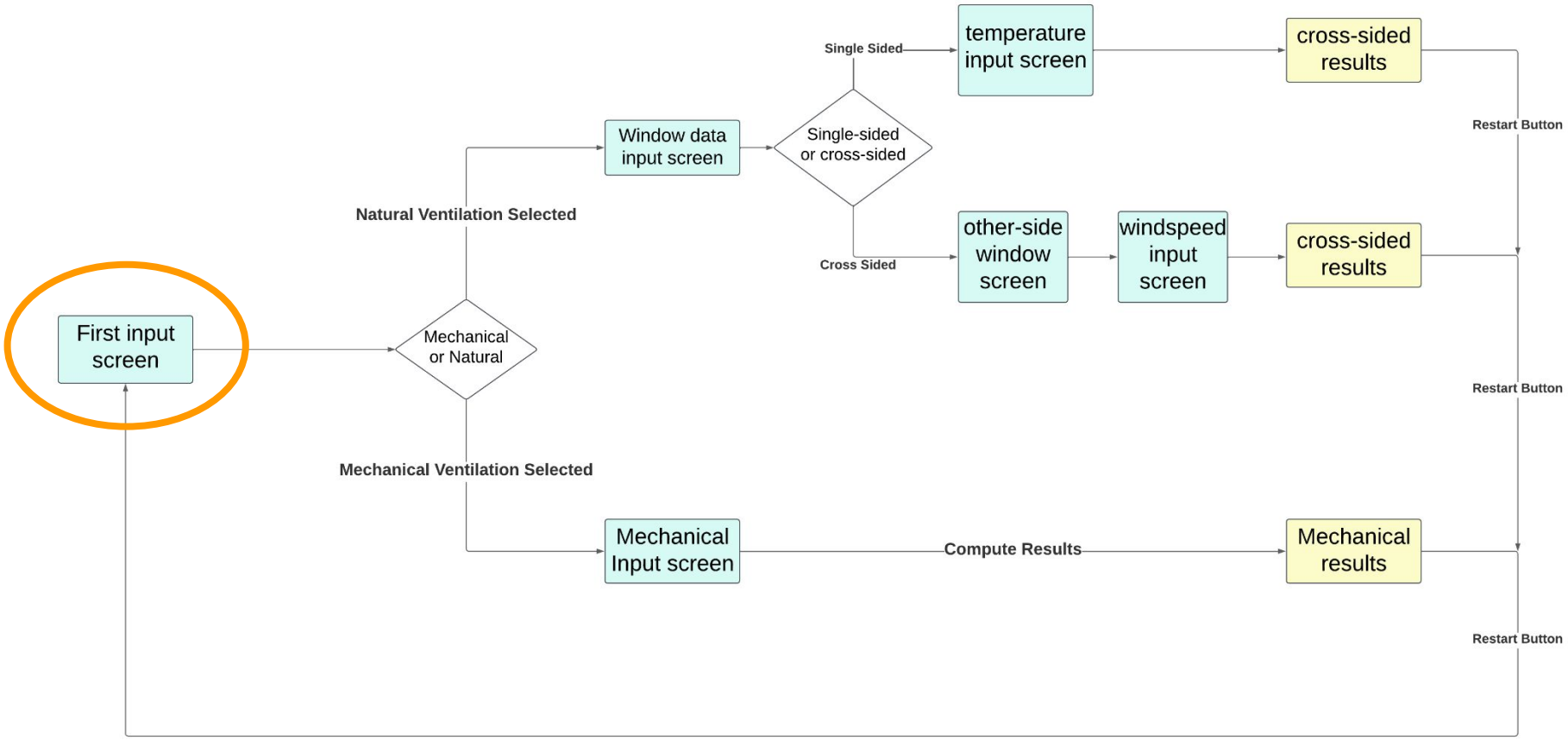
- **Version control:** GitHub
- **Design and Front End:** Figma
- **Development and Testing:** Flutter
- **API Testing:** Postman
- **Emulating:** Android Studio

# Architecture and Data Flow

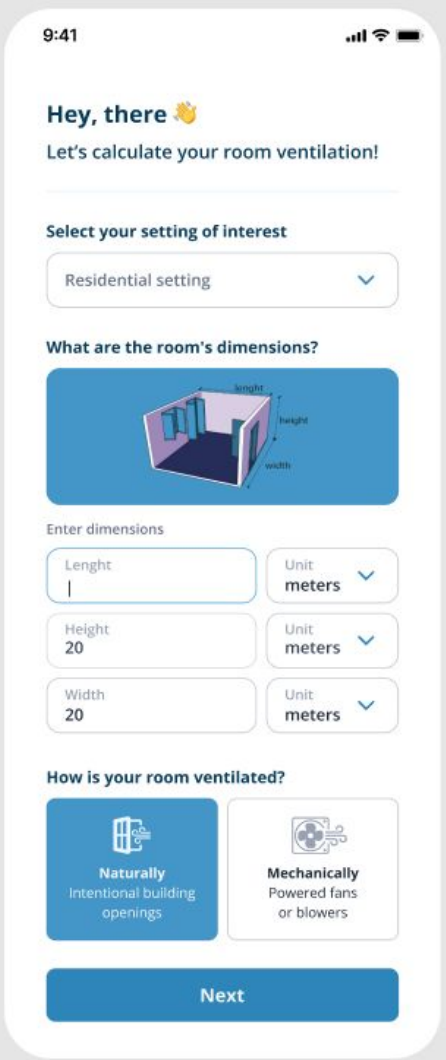
Handling inputs and choices from the user

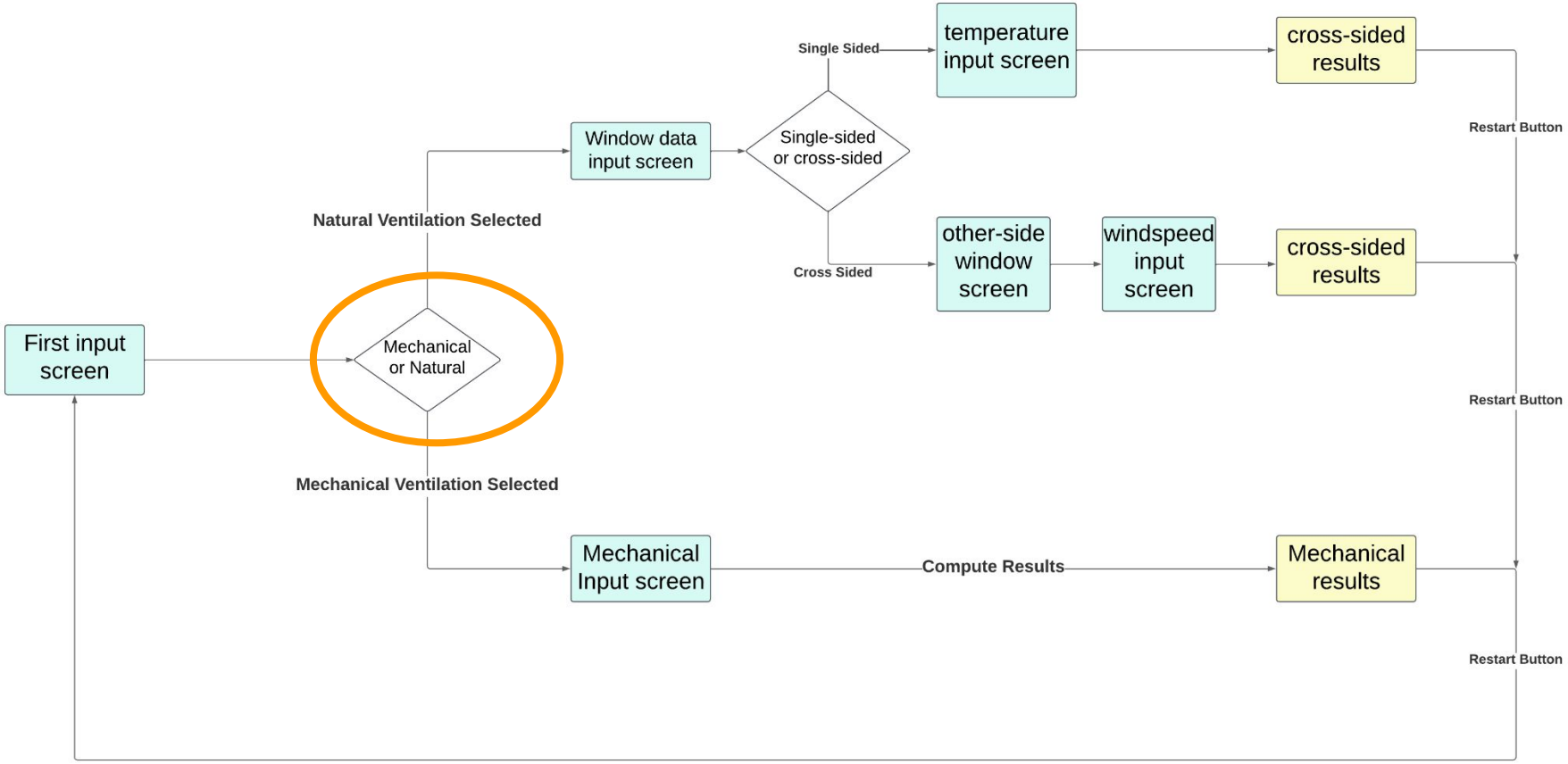


# Flow of the application

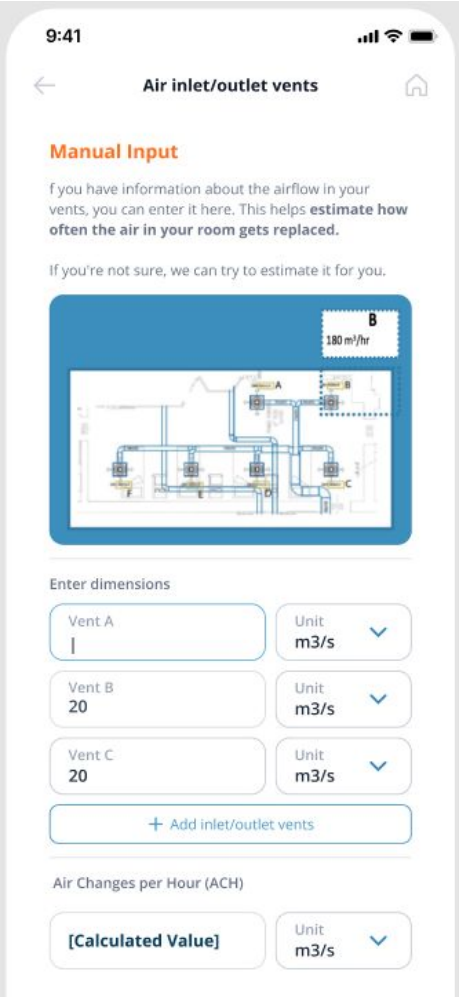



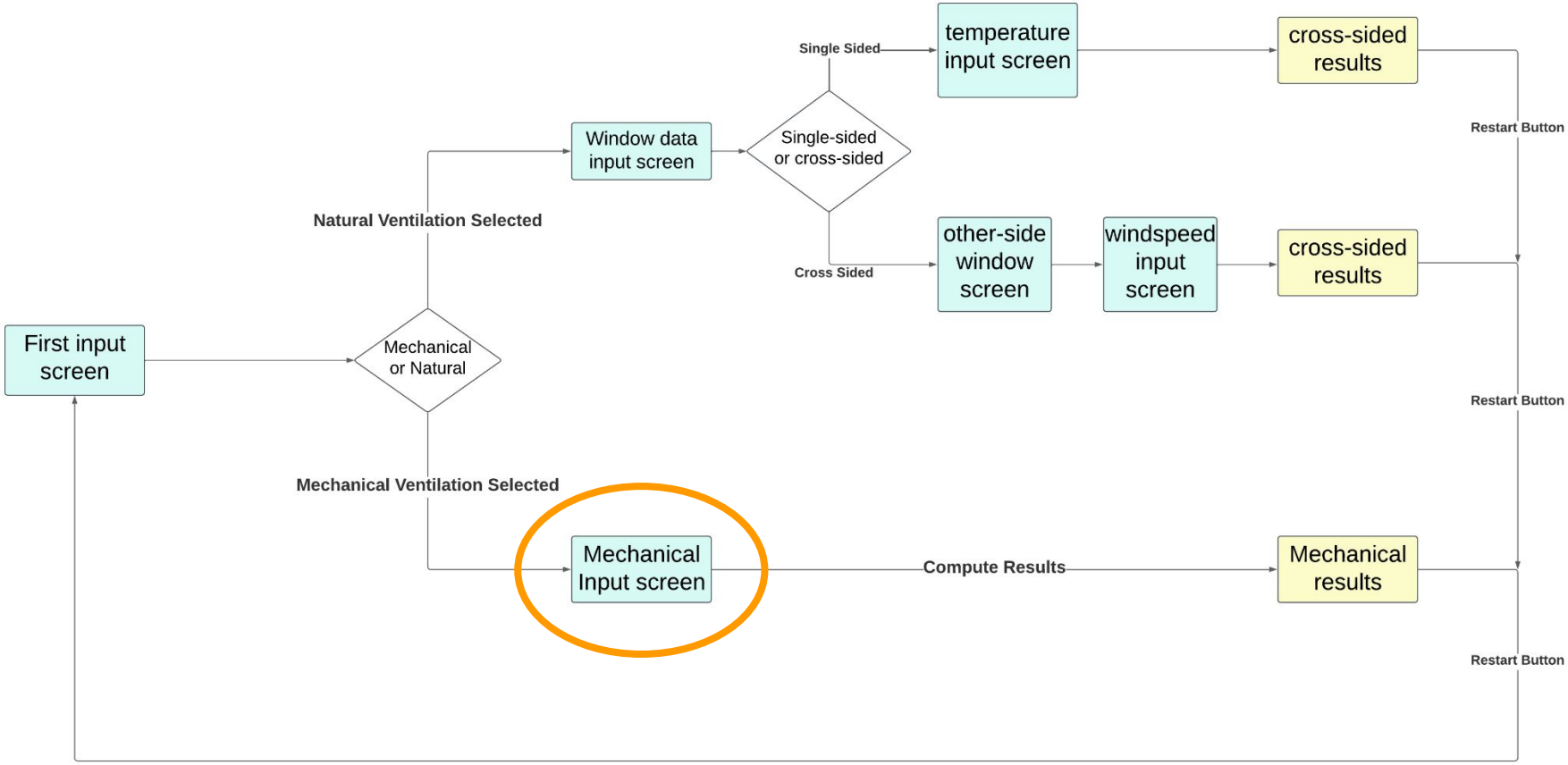
# Flow of the application



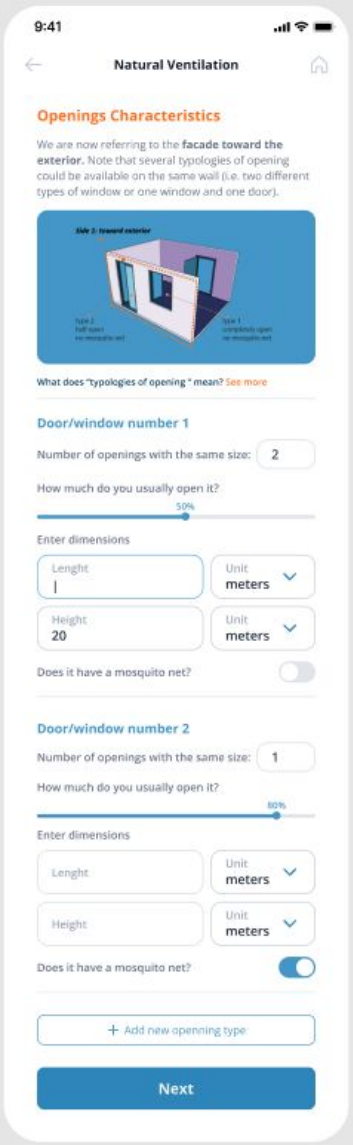



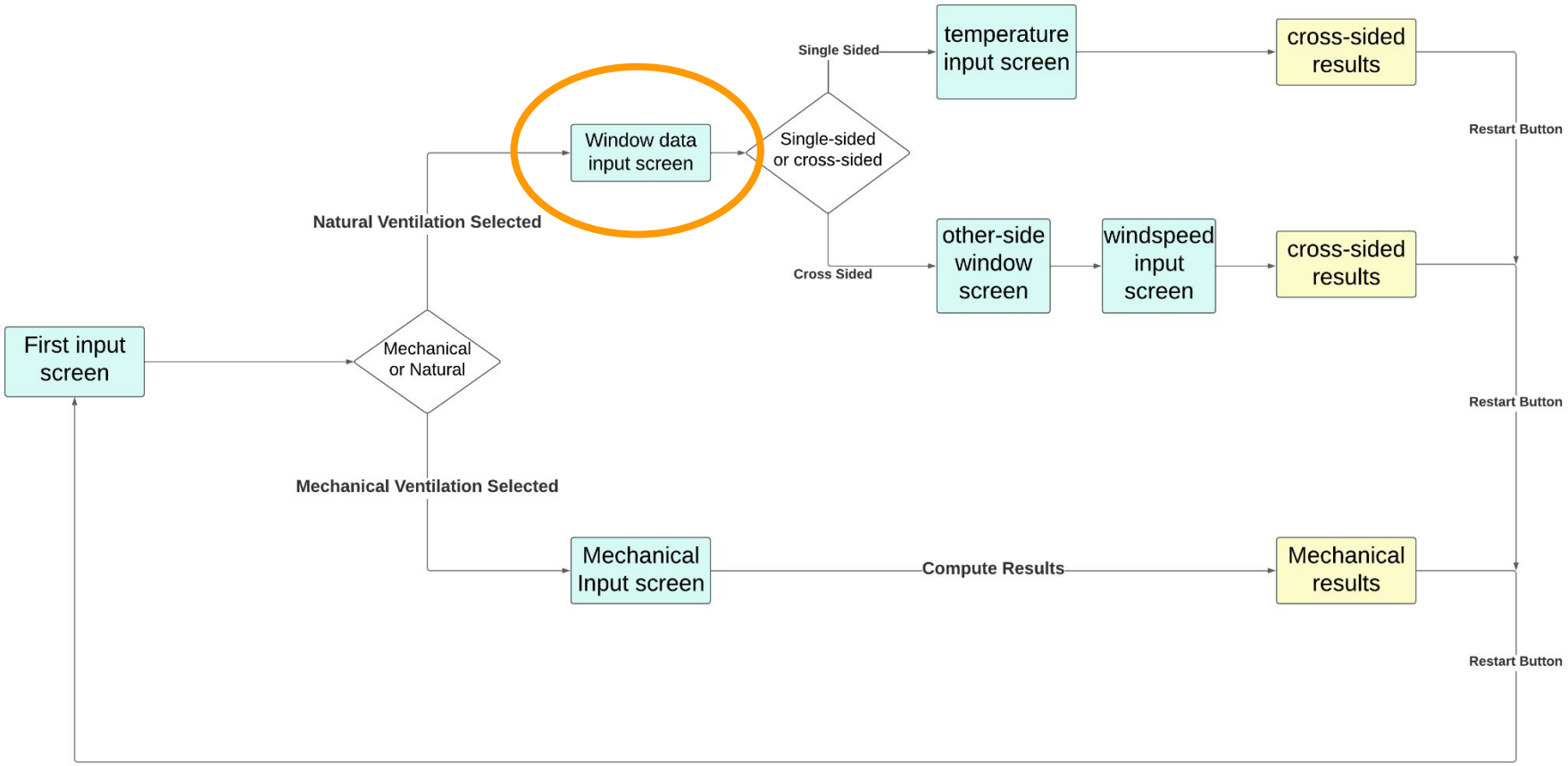
# Flow of the application



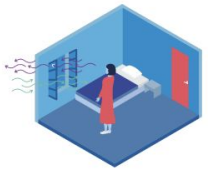



# Flow of the application



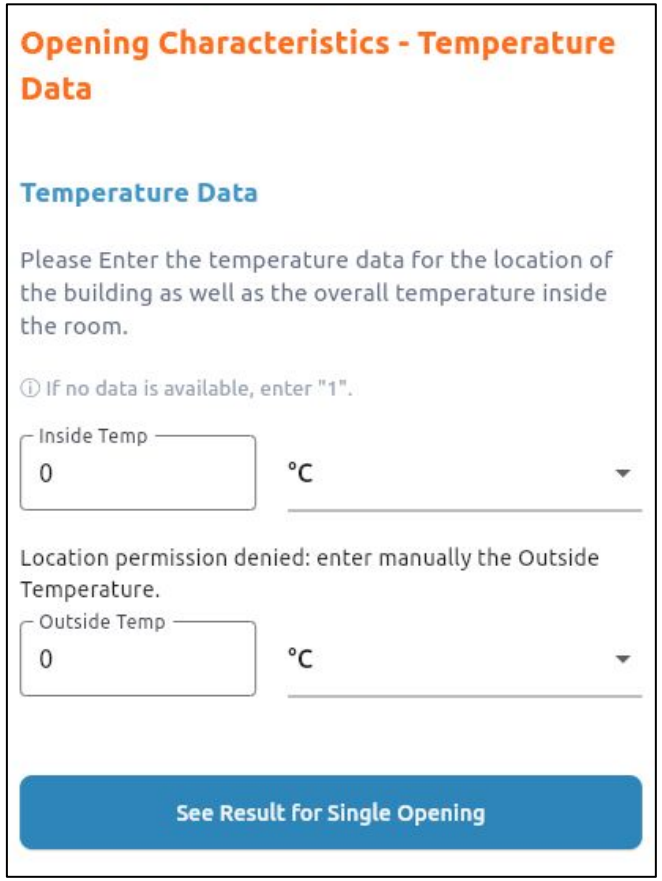


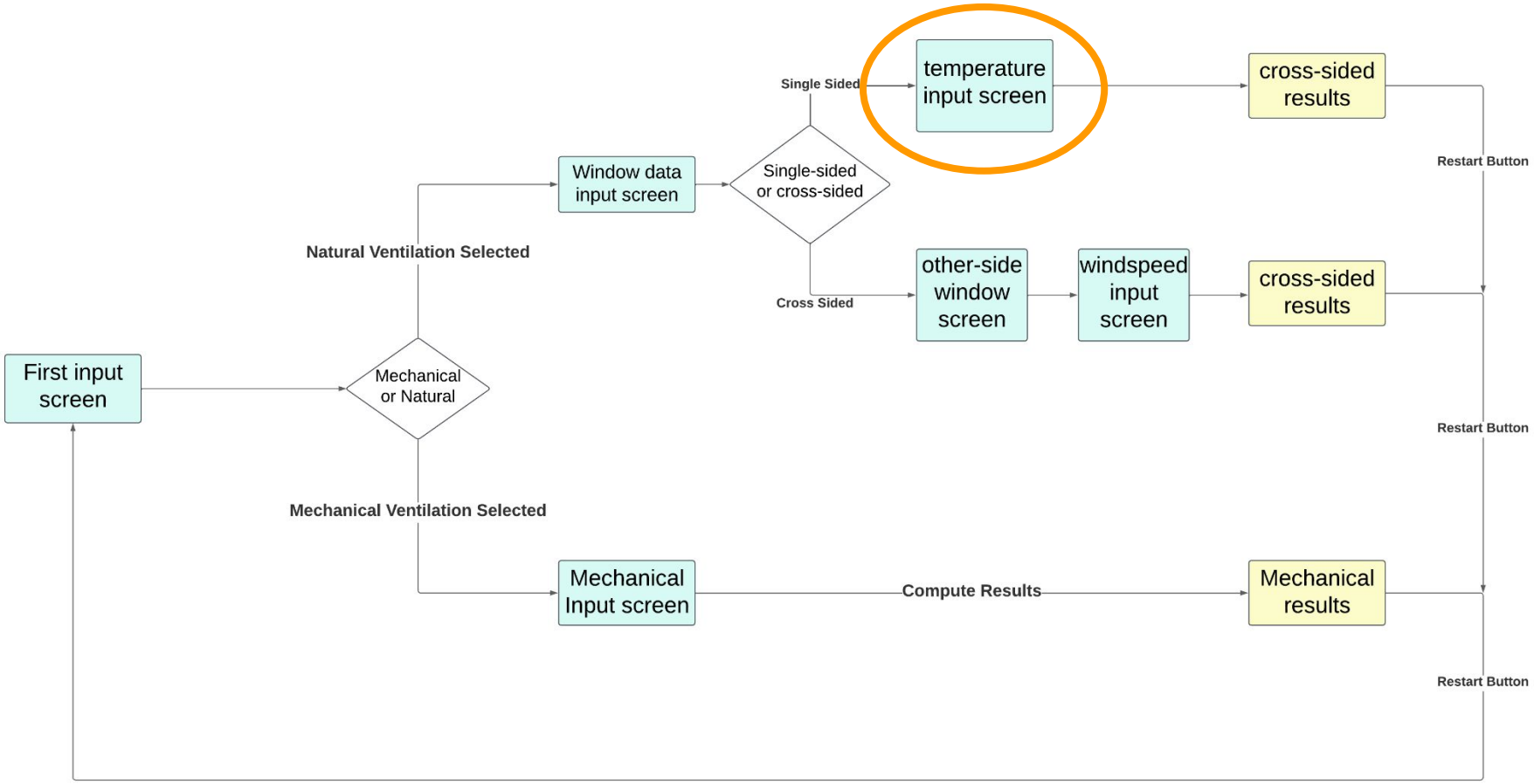


POLITECNICO MILANO 1863

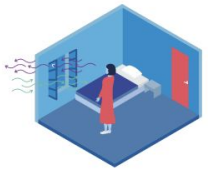
World Health Organization

# Flow of the application



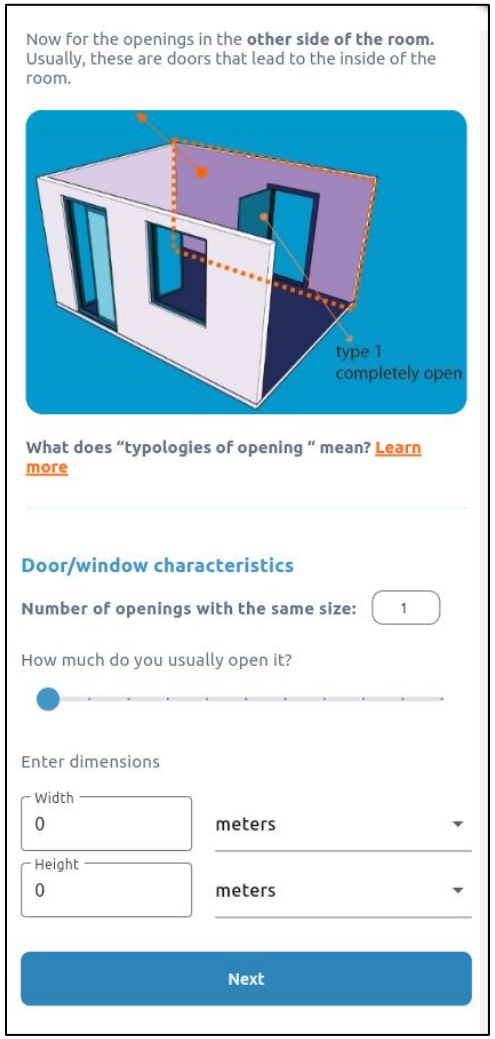

Are there openings available in other side of the room?

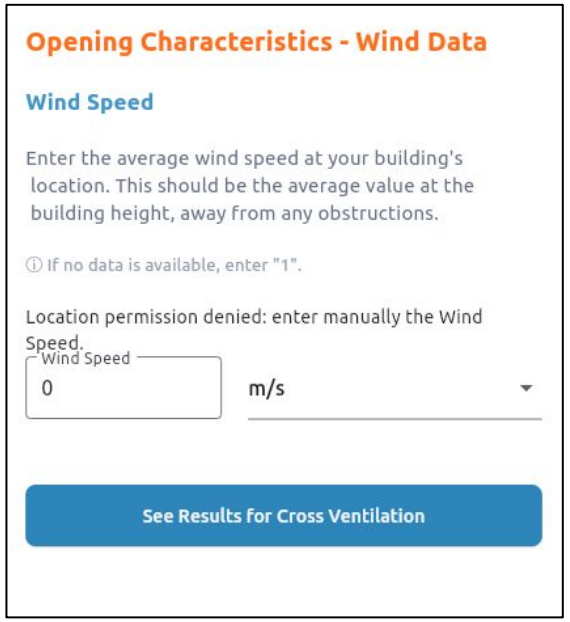


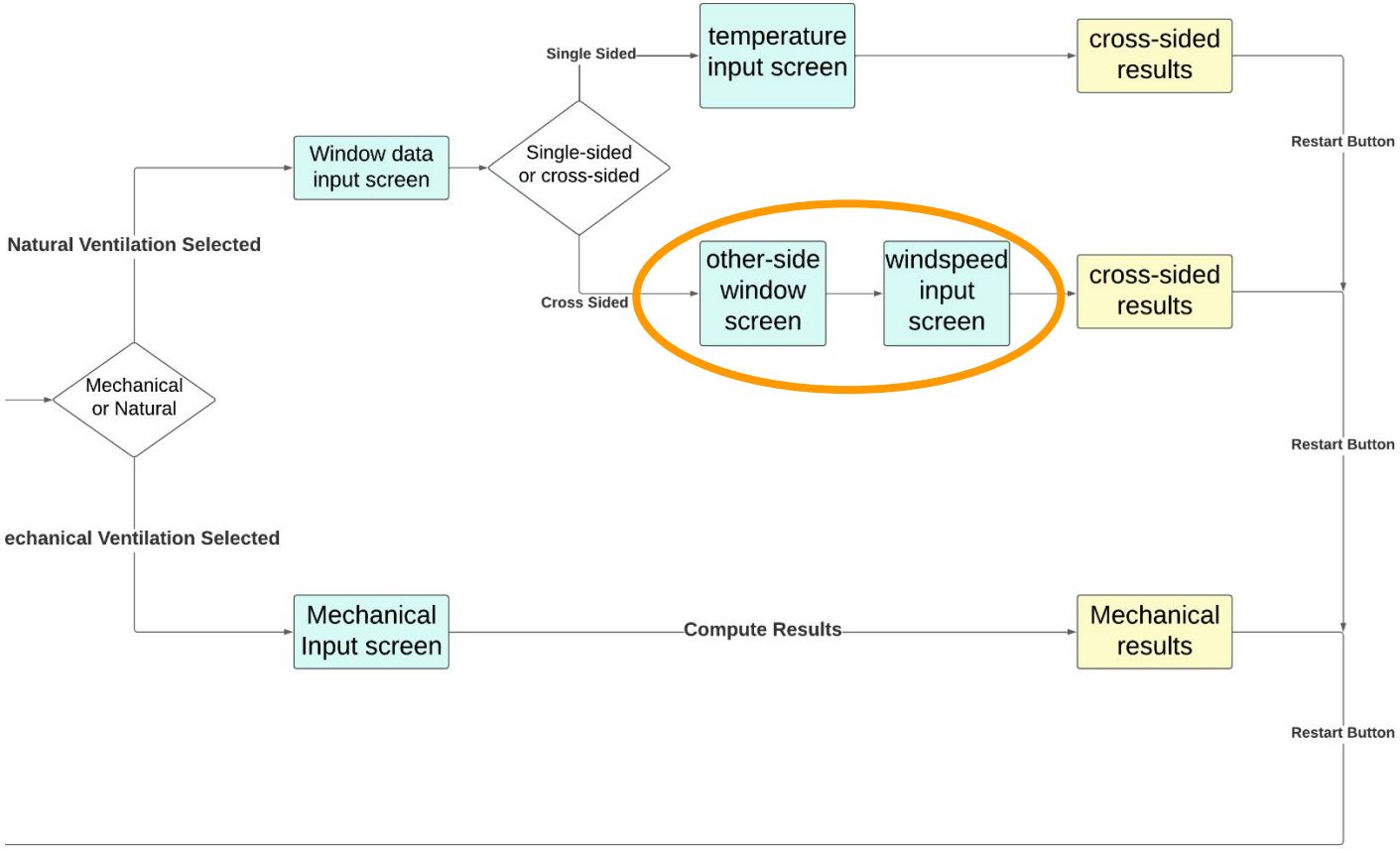
POLITECNICO MILANO 1863

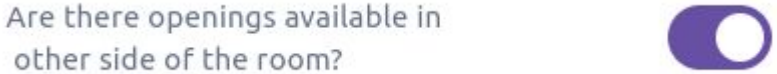
World Health Organization

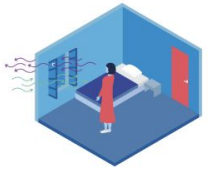
# Flow of the application



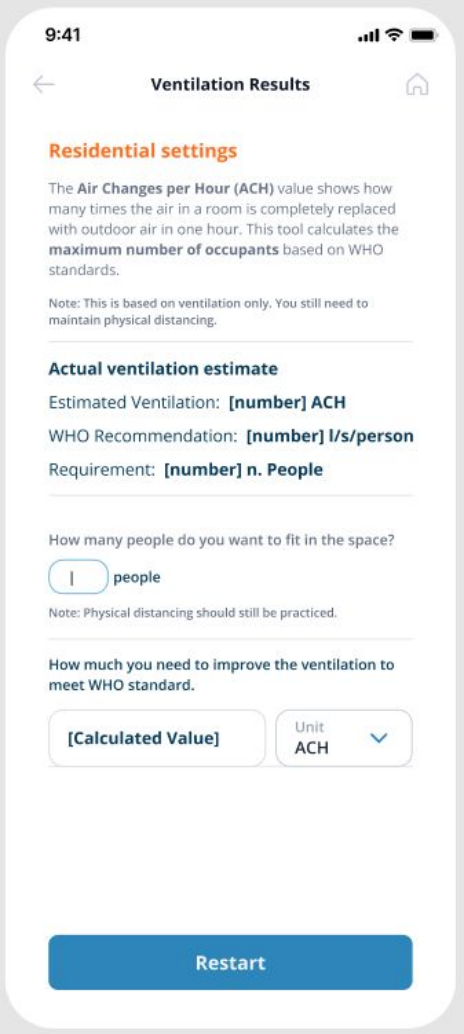



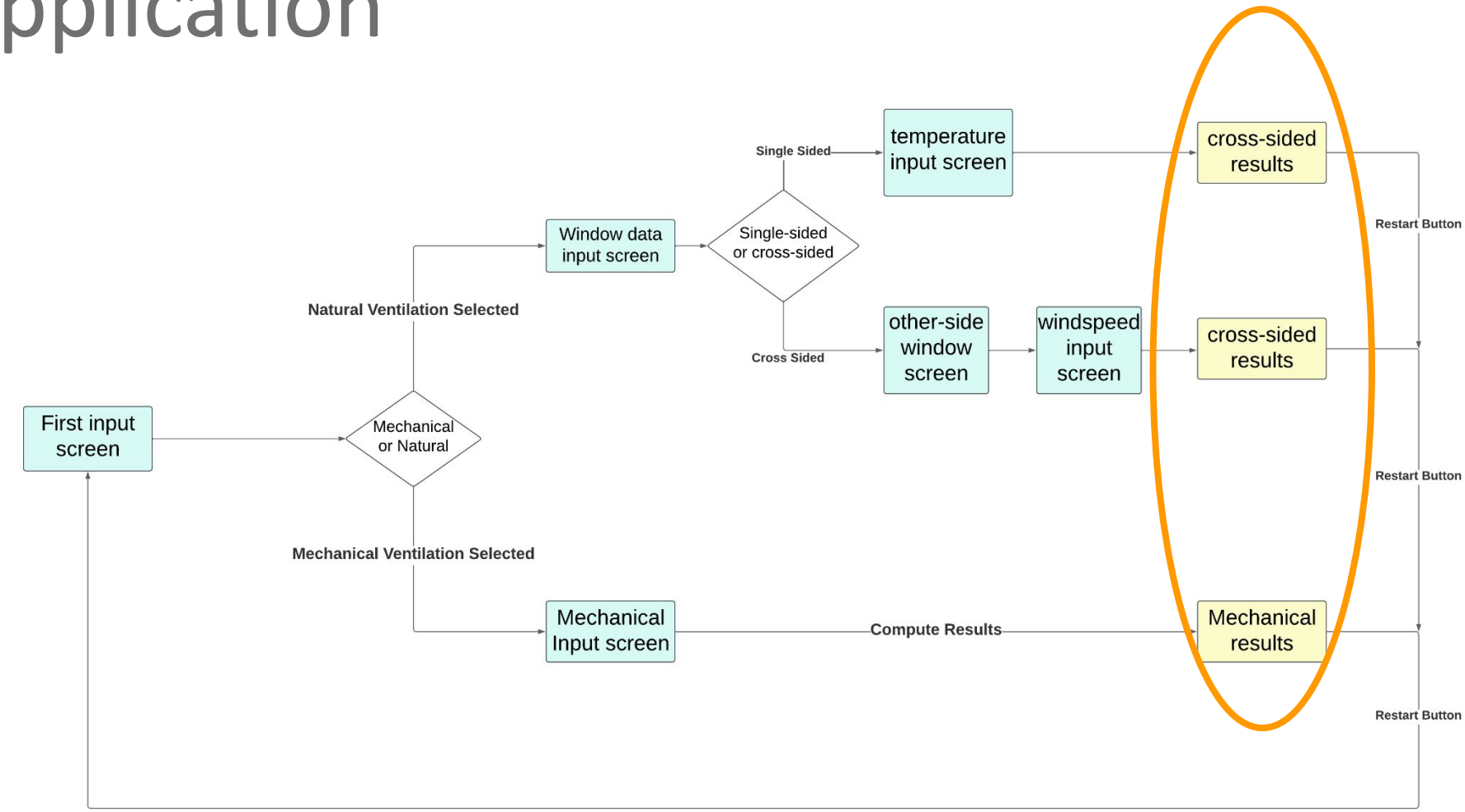



POLITECNICO MILANO 1863

World Health Organization

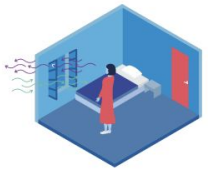
# Flow of the application



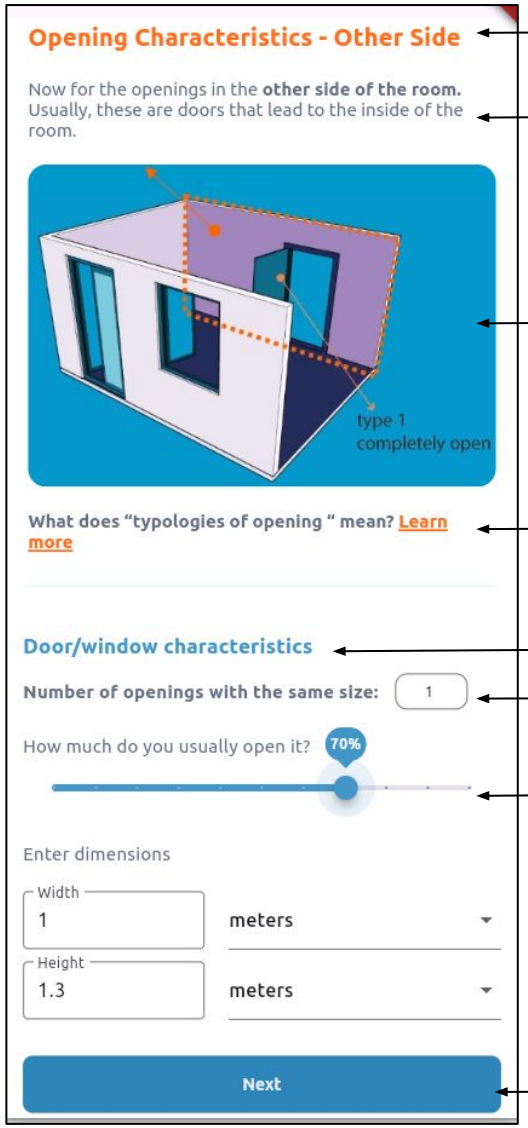



# Development and Functionalities

Key pieces of the development of the application



# Common Usage Widgets




TextEntry( )

_buildRichTextContent( )

OpeningImage( )

_buildHyperLinkText( )

TextEntry( )

CustomInputWidget( )

Slider0To100( )

DimensionInputRow( )

NextButton( )

# Routes

- Overview
  - Handles routing.
  - Uses a switch statement to return the correct MaterialPageRoute for each screen.
  - Handles undefined routes with _errorRoute().
- Key Features
  - Dynamic Routing: Efficiently maps URLs to screens.
  - Scalability: Easily add new routes.
  - Error Handling: Ensures undefined routes don’t crash the app.

```
class RouteGenerator {
  static Route<dynamic> generateRoute(RouteSettings settings) {
    // Getting arguments passed in while calling Navigator.pushNamed
    // final args = settings.arguments;

    switch (settings.name) {
      case '/':
        return MaterialPageRoute(builder: (_) => SplashScreen());
      case '/second':
        // Validation of correct data type
        return MaterialPageRoute(builder: (_) => const Instructions());
      case '/input_1':
        return MaterialPageRoute(builder: (_) => Input1());
      case '/input_nat_2':
        return MaterialPageRoute(builder: (_) => InputNat2());
      case '/input_nat_3':
        return MaterialPageRoute(builder: (_) => InputNat3());
      case '/nat_wind_speed':
        return MaterialPageRoute(builder: (_) => NatWindSpeed());
      case '/nat_temperature':
        return MaterialPageRoute(builder: (_) => NatTemperature());
      case '/nat_results_single':
        return MaterialPageRoute(builder: (_) => NatResultsSingle());
      case '/nat_results_cross':
        return MaterialPageRoute(builder: (_) => NatResultsCross());
      case '/input_mech_2':
        return MaterialPageRoute(builder: (_) => InputMec2());
      case '/mec_results':
        return MaterialPageRoute(builder: (_) => MecResults());
      case '/input_nat_more_opens':
        return MaterialPageRoute(builder: (_) => InputNatMoreOpens());
      // If args is not of the correct type, return an error page.
      // You can also throw an exception while in development.
      default:
        // If there is no such named route in the switch statement, e.g. /third
        return _errorRoute();
    }
  }
```

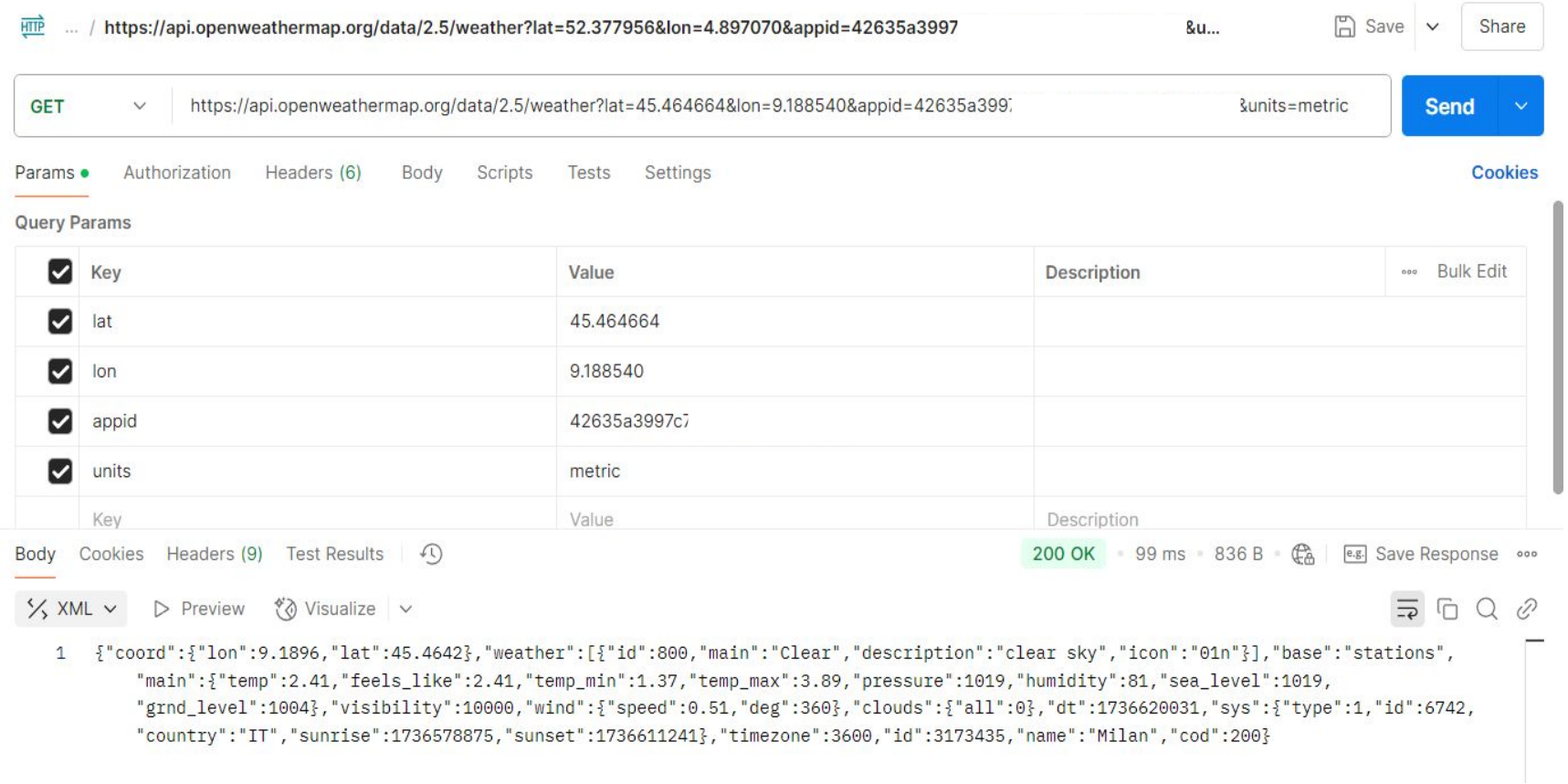
# State Manager: CalculationState

- **Overview**:
  - Manages room settings and ventilation parameters using ChangeNotifier.
- **Key Features:**
  - Real-time updates with ChangeNotifier.
  - Uses maps for unit selection and boolean toggles for options.
  - Modular methods for dynamic updates to dimensions, rates, and environmental conditions.

```
class CalculationState with ChangeNotifier {
  //3-input_1
  Map<String, bool> _settingOfInterest = {
    'Residential/Comercial': true,
    'Hospital Setting': false
  };

  String _lenght = '0';
  String _height = '0';
  String _width = '0';

  Map<String, bool> _unitLeght = {'meters': true, 'inches': false};

  Map<String, bool> _unitHeight = {'meters': true, 'inches': false};

  Map<String, bool> _unitWidth = {'meters': true, 'inches': false};

  Map<String, bool> _ventType = {'nat': true, 'mec': false};

  Map<String, bool> get settingOfInterest => _settingOfInterest;
  String get lenght => _lenght;
  String get height => _height;
  String get width => _width;
  Map<String, bool> get unitLeght => _unitLeght;
  Map<String, bool> get unitHeight => _unitHeight;
  Map<String, bool> get unitWidth => _unitWidth;
  Map<String, bool> get ventType => _ventType;
```

# OpenWeather API (1/3)

OpenWeather




POLITECNICO MILANO 1863

World Health Organization

# OpenWeather API (2/3):

## geolocation.dart

- Overview
  - The *geolocation.dart* file provides the *LocationService* class to retrieve the user's current location.
- Key Features
  - Service & Permission Checks: ensures location services are active and handles permission requests or denials.
  - Accurate Location Retrieval: uses high-accuracy positioning via *Geolocator.getCurrentPosition()*.
  - Error Handling: handles disabled services and permission issues with clear exceptions.

```dart
import 'package:geolocator/geolocator.dart';

class LocationService {
  Future<Position> getCurrentLocation() async {
    bool serviceEnabled = await Geolocator.isLocationServiceEnabled();
    if (!serviceEnabled) {
      throw Exception('Location services are disabled.');
    }

    LocationPermission permission = await Geolocator.checkPermission();
    if (permission == LocationPermission.denied) {
      permission = await Geolocator.requestPermission();
      if (permission == LocationPermission.deniedForever) {
        throw Exception('Location permissions are permanently denied.');
      } else if (permission == LocationPermission.denied) {
        throw Exception('Location permissions are denied.');
      }
    }

    return await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.high,
    );
  }
}
```

# OpenWeather API (3/3)

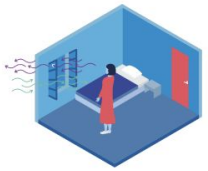
weather.dart

- Overview
  - The *weather.dart* file defines the *WeatherService* class to fetch real-time weather data using OpenWeather API based on the user's location.
- Key Features
  - API Integration: Constructs API calls with latitude, longitude, and API key parameters.
  - HTTP Requests: Sends asynchronous GET requests to the OpenWeather API.
  - Error Handling: Throws exceptions for non-200 HTTP responses.
  - JSON Parsing: Decodes and returns the weather data in a structured format.

```dart
import 'package:geolocator/geolocator.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'package:ventilation_app/api_key.dart';

class WeatherService {
  // final String apiKey = apiKey; // API key

  Future<Map<String, dynamic>> getWeatherData(Position position) async {
    String apiUrl = 'https://api.openweathermap.org/data/2.5/weather'
        '?lat=${position.latitude}&lon=${position.longitude}&appid=$apiKey&units=metric';

    final response = await http.get(Uri.parse(apiUrl));

    if (response.statusCode == 200) {
      return jsonDecode(response.body);
    } else {
      throw Exception('Failed to load weather data');
    }
  }
}
```

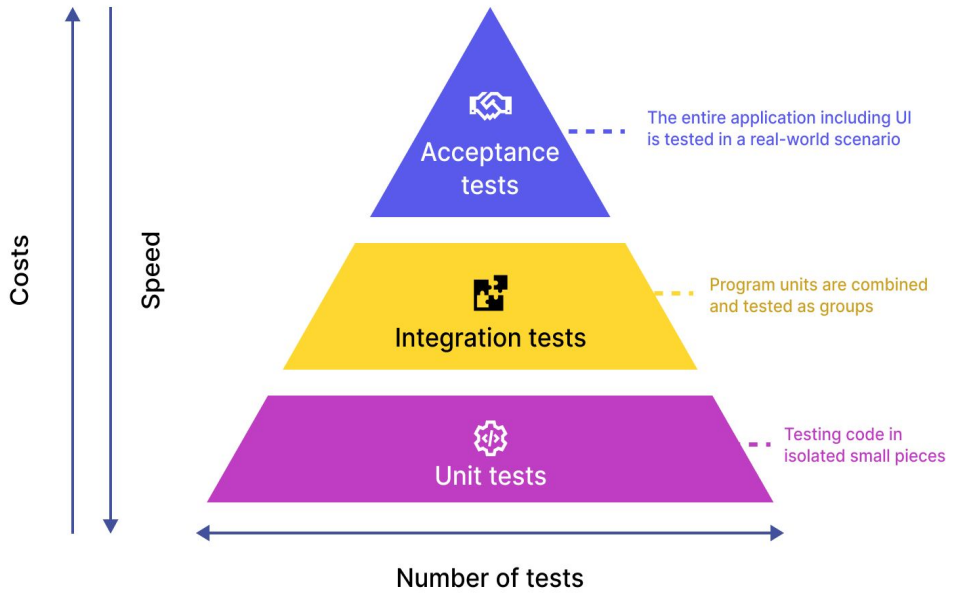
# Testing

The assessments made to make sure the requirements were satisfied



# Testing: Unit Tests

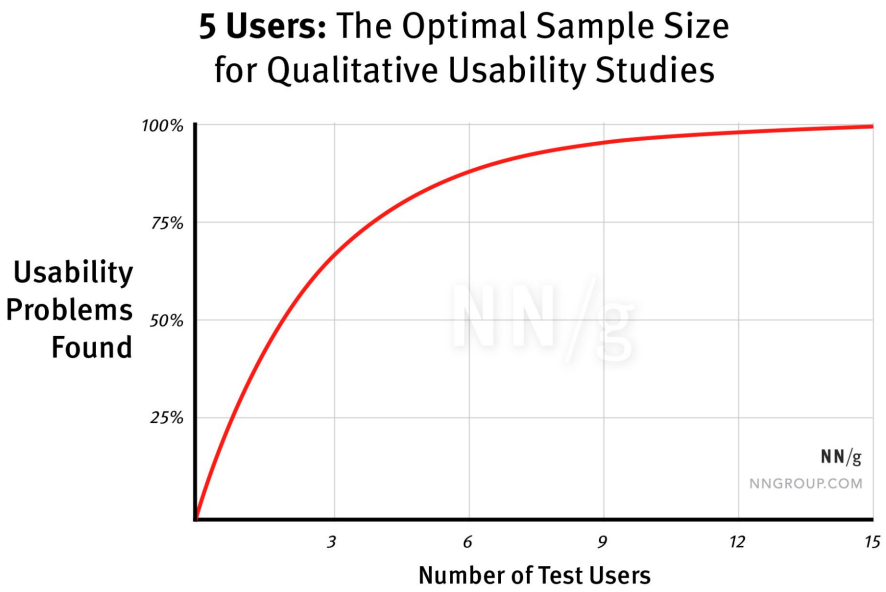
- Tests developed using the *flutter_test* library.
- Coverage of 57 Unit Tests, mainly to test each Widget individually
- 2 Integration tests:
  - One to assert that the file path interacts correctly with the system
  - One to test the API procedure




```
00:16 +59: All tests passed!
→ ventilation_app git:(shadow) flutte
```

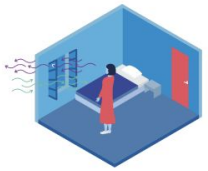
# Testing: Acceptance Test

- Based on this work: Why You Only Need to Test with 5 Users by Jakob Nielsen
- We invited 5 friends to test the app by running the emulator.
- We collected some feedback after each test.
- All 5 managed to get a result and found no major problems.
- Minor problems noted added to “future work”.




# Demonstration

Use Cases



# Demonstrations (1/3)

- Scenario 01
  - Giacomo wants to have a dinner reception in his new house, but he is not sure on how many people he can invite to his living room in order to respect WHO recommendations to avoid respiratory diseases.
  - Room dimensions (l,h,w):
    - 12.5m x 3.1m x 7.2m
  - Side 01:
    - 1 windows of 0.8m x 1.2m
      - No mosquito nets
      - Opened 20%
  - Side 02:
    - 2 windows of 2.1m x 1.5m
      - No mosquito nets
      - Opened 60%
  - He has enabled the location permissions on his smartphone.

# Demonstrations (2/3)

- Scenario 02
  - Maria is responsible for a refugee hospital and she wants to check the natural ventilation of a room to see how many patients fit it in order to control and prevent infections.
  - Room dimensions (l,h,w):
    - 8.5m x 4.2m x 5m
  - Single Side:
    - 2 windows of 1.5m x 1.5m
      - With mosquito nets
      - Opened 50%
    - 1 window of 2m x 1.5m
      - With mosquito nets
      - Opened 40%
  - The temperature inside: 25 °C
  - The temperature outside: 30 °C
  - She disabled the location service of the app on her tablet.

POLITECNICO MILANO 1863

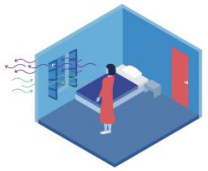
World Health Organization

# Demonstrations (3/3)

- Scenario 03
  - Paolo has just been promoted in his company and his first task as the new Head of Ventilation is to check if the meeting room for 20 people is in agreement with the WHO recommendations to control and prevent respiratory infections.
  - Room dimensions (l,h,w):
    - 10.2m x 4.3m x 6.2m
  - Mechanical Ventilation, with 2 the specifications:
    - 30 l/s
    - 2  ACH (Air Change per Hour)
  - He is using his tablet.

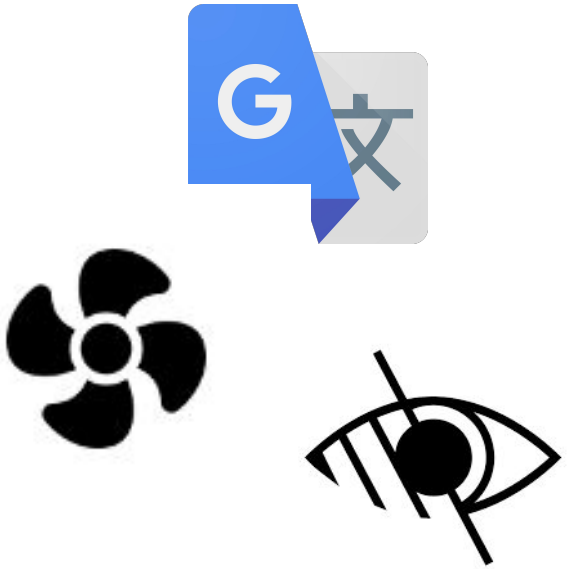
# Future Improvements

Improvements based on the interviews taken



# Future Improvements

- Implement a tool to add more languages to the app, making it more universal for non-english speakers
- Develop a better system of visually impaired people
- Tools that help in the measurement process of the window such as guides for estimating
- Add a window and ventilator databases so that people can also chose to select their data by brand



# Thanks for your attention!

Questions?

POLITECNICO
MILANO 1863

World Health Organization

# Widget Example:

```
46  class DimensionInputRow extends StatefulWidget {
47    final String labelText; // Label for the text field
48    final List<String> dropdownItems; // List of dropdown items
49    final String? initialNumber; // Initial value for the text field
50    final String? initialDropdownValue; // Initial value for the dropdown
51    final Function(String?, String?)? onChanged; // Callback for value changes
52
53    const DimensionInputRow({
54      Key? key, // Key for the widget to interact with the statemanager
55      required this.labelText,
56      required this.dropdownItems,
57      this.initialNumber,
58      this.initialDropdownValue,
59      this.onChanged,
60    }) : super(key: key);
61
62    @override
63    DimensionInputRowState createState() => DimensionInputRowState();
64  }
```

```
66  class DimensionInputRowState extends State<DimensionInputRow> {
67    late TextEditingController
68        _textController; // Persistent controller for the text field
69    String? selectedDimensionValue;
70
71    @override
72    void initState() {
73      super.initState();
74      // Initialize the text controller and dropdown value
75      _textController = TextEditingController(text: widget.initialNumber);
76      selectedDimensionValue = widget.initialDropdownValue;
77    }
78
79    @override
80    void dispose() {
81      _textController.dispose(); // Clean up the controller
82      super.dispose();
83    }
```

# Widget Example:

```
85   @override
86   Widget build(BuildContext context) {
87     return Row(
88       mainAxisAlignment: MainAxisAlignment.start,
89       children: [
90         SizedBox(
91           width: 150.0,
92           child: TextField(
93             decoration: InputDecoration(
94               border: const OutlineInputBorder(),
95               labelText: widget.labelText,
96             ), // InputDecoration
97             keyboardType: TextInputType.number,
98             controller: _textController, // Use the persistent controller
99             onChanged: (value) {
100              // Notify parent whenever the text changes
101              if (widget.onChanged != null) {
102                widget.onChanged!(value, selectedDimensionValue);
103              }
104            },
105          ), // TextField
106        ), // SizedBox
107        const SizedBox(width: 20.0),
108        Flexible(
```

```
108        Flexible(
109          child: DropdownButtonFormField<String>(
110            value: selectedDimensionValue,
111            items: widget.dropdownItems.map((item) {
112              return DropdownMenuItem(
113                value: item,
114                child: Text(item),
115              ); // DropdownMenuItem
116            }).toList(),
117            onChanged: (value) {
118              setState(() {
119                selectedDimensionValue = value;
120              });
121              // Notify parent whenever the dropdown value changes
122              if (widget.onChanged != null) {
123                widget.onChanged!(_textController.text, selectedDimensionValue);
124              }
125            },
126          ), // DropdownButtonFormField
127        ), // Flexible
128      ],
129    ); // Row
```

# State Manager: CalculationState (2/2)

- Highlights:
  - Real-time updates with ChangeNotifier.
  - Uses maps for unit selection and boolean toggles for options.
  - Modular methods for dynamic updates to dimensions, rates, and environmental conditions.

```
void updateSetOfInterest(bool res) {
  _settingOfInterest['Residential/Comercial'] = res;
  _settingOfInterest['Hospital Setting'] = !res;
  notifyListeners();
}

void updateRoomDimensions(
    String h, String l, String w, Map unitH, Map unitL, Map unitW) {
  _lenght = l;
  _height = h;
  _width = w;
  _unitLeght = unitL.cast<String, bool>();
  _unitHeight = unitH.cast<String, bool>();
  _unitWidth = unitW.cast<String, bool>();
  notifyListeners();
}

void updateVentType(bool nat) {
  _ventType['nat'] = nat;
  _ventType['mec'] = !nat;
  notifyListeners();
}
```