PureCare Air Purifier

Jimmy MacDonald
dept. of Computer Science
Hanyang University
Seoul, South Korea
james.macdonald.1@slu.edu

Yim Soobeen
dept. of Information System
Hanyang University
Seoul, South Korea
s00been@hanyang.ac.kr

Kim Minjin
dept. of Information System
Hanyang University
Seoul, South Korea
idid02@hanyang.ac.kr

Kim Yeonwoo
dept. of Computer Science
Hanyang University
Seoul, South Korea
bbongvong@hanyang.ac.kr

***Abstract*—Air purifiers have become more and more common in people's houses, but as houses gain more technology, it seems that air purifiers haven't had any major improvements. We propose a smart air purifier which learns and adapts how it runs based on it's users. Specifically, by monitoring and recognizing when users are sick, gaining insights about air quality outside as well as seasonal allergies, and smart scheduling on when to run the air purifier to minimize electricity usage. We are going to implement 3 models-cough detection, sniff detection, machine control. Respiratory diseases are air-sensitive, which means complicated factors, such as air quality-include dust, viruses, temperature, and humidity affect users. Our expanded goal is to implement total air care control.**

***Index Terms*—Smart Air Purifier, AQI**

TABLE I
ROLE ASSIGNMENTS

Roles	**Name**	**Task description and etc.**
User	Yim Soobeen	Identifies user needs and improves product usability. Thinks of ideas, designs functions, and suggests practical ways to implement features that address real-world problems.
Customer	Jimmy MacDonald	Conducts competitive analysis and plans differentiation strategy. Examines competitor services, identifies market gaps, and designs UI elements that attract customers.
Software developer	Kim Yeonwoo	Converts requirements into functional software. Writes code, implements features, debugs programs, monitors performance, and ensures all components execute properly.
Development Manager	Kim Minjin	Manages overall development process and team coordination. Creates plans, assigns roles, monitors progress, manages schedules, and ensures smooth communication among team members.

I. INTRODUCTION

A. Motivation

As the world continues to industrialize, air quality has declined as a direct consequence. As harmful particulate matter such as $PM_{2.5}$, PM_{10}, and other volatile organic matter enter the atmosphere, residents find difficulty in breathing under these conditions. The World Health Organization (WHO), estimates that 99% of the global population breathes air that exceeds guidelines, leading to millions of premature deaths annually from various diseases [1].

Despite living in a technology-driven age, air purifiers have seemed to lack any significant advances, remaining more reactive than proactive. Modern air purifiers activate only once particles have been detected in the air, leaving users exposed to a harmful environment until the device responds. While this reactive approach provides some benefit, the delay between exposure and activation can lead to numerous audible health symptoms, such as sneezing or coughing, potentially interrupting important moments such as presentations or meeting. Furthermore, current air purifiers require human configuration and continue to run continuously regardless if anyone is home resulting in a waste of electricity.

We propose PureCare, a truly intelligent air purification system that acts as a proactive health assistant rather than a passive appliance. PureCare distinguishes itself through several key innovations: Audible Health Symptom Recognition technology that detects sneezing, coughing, and snoring to automatically adjust purification settings; geofencing capabilities that optimize operation based on occupancy; API integration with local air quality data to prepare for poor outdoor conditions; personalized care recommendations that learn individual sensitivities over time; and sleep mode optimization for nighttime respiratory issues. Unlike traditional purifiers that simply filter air, PureCare anticipates needs, provides gentle human-like notifications, and creates safer environments before problems arise.

B. Similar Products

A. LG PuriCare Objet Collection AI+ 360° Air Purifier

- Detects harmful gas and fumes from three major sick-building substances-formaldehyde ammonia, and volatile oraganinc compounds (VOCs)-and automatically purifies the air according to gas type and pollution level.
- Detects ammonia, the main source of pet waste odors.
- Detects contaminants such as cooking fumes.
- "AI Customized Operation" learns and analyzes indoor air quality every hour. Using accumulated data, it identifies times when the air is clean, stops the purifier's fan, and dims the display to save energy.
- By autonomously analyzing air quality and adjusting operating intensity, it can reduce power consumption by up to 50 percent or more compared with the conventional AI mode, helping lower electricity costs.

B. SAMSUNG BESPOKE Cube Air Infinite Line

- Provides an AI-powered integrated personalized cleaning solution.
- "Customized Clean AI+" compares and learns indoor and outdoor air quality, and pre-cleans when deterioration is predicted.
- "AI Saving Mode" automatically adjusts airflow or stops the fan when indoor air improves, cutting energy use by up to 45
- "Customized Clean AI+" is certified with "AI+ Certification" by the Korea Standards Association.

C. SK Wellness Robot

- The Wellness Robot by SK Namoothix's brand, NAMUHX, removes harmful substances that enter during ventilation to maintain air quality. It also offers performance equivalent to six air purifiers. All of its air solution functions are controlled entirely by voice without touch, and through the AI control system, device monitoring and remote support for maintenance are provided, greatly enhancing user convenience.

II. Requirement Analysis

A. App Installation and First Launch

1) FR-001: Application and First Launch:

A. Platform Availability

1) iOS Platform: Available on App Store for iOS 12.0+ (iPhone 6S and newer)
2) Android Platform: Available on Google Play Store for Android 10+ (Armv7 and ARM64 architectures)

B. First Launch Experience

1) Splash Screen: Displays application logo and version number for 1-2 seconds
2) Permission Requests: Camera (for QR scanning), Notifications (for device alerts), and optional Location (for weather recommendations)

B. Login/Sign Up

1) FR-002: User Authentication:

A. Welcome Screen
Displays after splash screen with application logo, tagline, and two buttons: "Login" and "Sign Up"

B. Login Screen
Contains email and password fields, "Remember me" checkbox, "Forgot Password?" link, and "Don't have an Account? Sign Up" link

C. Sign-up Flow

1) Page 1: Basic account information (Full Name, Email, Password, Confirm Password, Terms Agreement)
2) Page 2: Optional profile information (Phone, Profile Picture, DOB, Country)

C. Product Setup via QR Code

1) FR-003: Product Registration: Note: Only accessible after successful login/sign up

A. Access Points

1) Automatically shown to new users after login (if no devices registered)
2) "Add Device" button on Home Page
3) "+" button in navigation bar

B. Initial Screen for New Users
If user has zero registered devices, shows empty state with title "Connect your first device", description "Scan the QR code on your device to get started", "Scan QR Code" button, and "Enter serial number manually" link

C. QR Code Scanning Process

1) Camera Activation: Checks auth token validity. If invalid, redirects to login with message "Please login to register devices"
2) QR Detection and Validation: Scans QR code, validates against database, checks if device already registered. Error shown: "This device is registered to another account"
3) Device Linking: Links device to user account, stores User ID, Device Serial Number, Device Model, and Registration Timestamp

D. Product Information Confirmation
Displays product image, model name, serial number, and user's name with prompt "Register this device to {Name}'s account?" with confirm/cancel buttons

E. Registration Complete
Shows success message "Device registered successfully", sends email notification, offers "Go home" or "Add another device" options, then navigates to Home Page

D. *Home Page*

1) FR-004: Home Page: Note: Only accessible after successful login/sign up

A. Home Page Access
Displayed after successful login, device registration (first-time), or tapping "home" tab. Checks authentication on load—expired tokens redirect to Login

B. Home Page Layout
1) Header: User greeting "Welcome back {user}", current AQI conditions, notification bell, settings gear
2) Device List: Shows all registered devices or "No devices yet..." if empty. Each card displays Device Name, Model, Status, Last Active time, and Quick Action button. Includes "add device" floating action button

C. Device Status Synchronization
Loads device list from database, queries IoT server status, updates UI with real-time data, auto-refreshes every 30 seconds

D. Navigation Bar
The Home Page must include the bottom navigation bar as specified in FR-005, with the "Home" tab highlighted/active

E. *Navigation Bar*

1) FR-005: Navigation Bar:

A. Bottom Navigation Structure
Persistent across all main screens with three tabs arranged from left to right: Automation (clock icon), Home (house icon), Settings (Gear icon). When logged out, tapping any tab redirects to login

F. *Automation Page*

1) FR-006: Automation: Note: Only accessible after successful login/sign up

A. Routine Screen Sections
1) Screen Header
   i) *FR-RTS-1.0*: The screen must display a static title "Routines"
   ii) *FR-RTS-1.1*: Below the title must display a static helper text "Automate your Air Purifier by building Routines. For each Routine, schedule events with a start time and action for your selected devices."
   iii) *FR-RTS-1.2*: The header must contain an "Add" icon button (a + symbol) in the top-right corner.
2) Routine List
   i) *FR-RTS-2.0*: The system must fetch and display the user's routines
   ii) *FR-RTS-2.1*: If the list of routines exceeds the vertical viewport, the list must be vertically scrollable.
   iii) *FR-RTS-2.2*: Each routine in the list must be displayed as a distinct "Routine Card" (see FR-RTS-3.0).
3) Routine Card
   i) *FR-RTS-3.0*: Each Routine Card must display the user-defined name of the routine (e.g., "My Routine").
   ii) *FR-RTS-3.1*: The card must contain a visual summary of the routine's logic.
   iii) *FR-RTS-3.2*: The visual summary must display a chronological list of triggers (clock icons) aligned vertically.
   iv) *FR-RTS-3.3*: The visual summary must display a corresponding list of device actions (e.g., Air Purifier, Thermostat) to the right of the triggers.
   v) *FR-RTS-3.4*: A vertical connector line must visually link the triggers and actions, with nodes indicating each time/action pair.
   vi) *FR-RTS-3.5*: Device icons in the summary must visually represent their target state:
      A) On/Active: Icon is illuminated (e.g., glowing, in color).
      B) Off/Inactive: Icon is dim (e.g., greyed out).
4) Navigation
   i) *FR-ITS-4.0*: Tapping the "Add" icon (FR-RTS-1.2) must navigate the user to the "Create Routine" screen.
   ii) *FR-ITS-4.1*: Tapping on any existing "Routine Card" (FR-RTS-3.0) must navigate the user to the "Edit Routine" screen, pre-populated with that routine's data.
   iii) *FR-ITS-4.2*: The screen must display a "New Routine" card-style button below the list of existing routines.
   iv) *FR-ITS-4.3*: The "New Routine" button must display the text "New Routine" and an "Add" icon (+).
   v) *FR-ITS-4.4*: Tapping the "New Routine" button (FR-RTS-4.2) must navigate the user to the "Create Routine" screen (same action as FR-RTS-4.0).

5) Navigation Bar
   i) *FR-RTS-5.0*: The Automation Page must include the bottom navigation bar as specified in FR-005, with the "Automation" tab highlighted/active

G. *Add Routine Screen*

1) FR-007: Add Routine Screen:

A. Screen Header
1) *FR-RTA-1.0*: The screen must display a title "Custom Routine" centered at the top
2) *FR-RTA-1.1*: The header must contain a "Back" icon button (← arrow) in the top-left corner. When clicked, if the user has unsaved changes, a dialog must warn the user about unsaved changes with options to "Discard" or "Keep Editing"
3) *FR-RTA-1.2*: The header must contain a "Delete" icon button (trash bin icon) in the top-right corner. When clicked, a confirmation dialog must appear with "Delete Routine?" message and "Cancel"/"Delete" options

B. Routine Name Section
1) *FR-RTA-2.0*: Below the header, the screen must display an editable routine name field with default text "[Time] Routine" (e.g., "10:49 AM Routine")
2) *FR-RTA-2.1*: An edit icon (pencil) must appear next to the routine name to indicate editability
3) *FR-RTA-2.2*: Tapping the routine name or edit icon must open an inline text input or modal for editing
4) *FR-RTA-2.3*: A toggle switch must appear on the right side of the routine name to enable/disable the entire routine
5) *FR-RTA-2.4*: The toggle switch must be green when enabled and gray when disabled

C. Day Selection Section
1) *FR-RTA-3.0*: A "When" label must be displayed above the day selection buttons
2) *FR-RTA-3.1*: Seven day buttons must be displayed in a horizontal row: S, M, T, W, TH, F, SA
3) *FR-RTA-3.2*: Each day button must be a rounded square with uniform size and spacing
4) *FR-RTA-3.3*: Selected days must have white background; unselected days must have dark/transparent background
5) *FR-RTA-3.4*: Multiple days can be selected simultaneously
6) *FR-RTA-3.5*: At least one day must be selected for the routine to be valid
7) *FR-RTA-3.6*: Tapping a selected day must deselect it; tapping an unselected day must select it

D. Events Section
1) *FR-RTA-4.0*: An "Events" label must be displayed as a section header
2) *FR-RTA-4.1*: A copy/duplicate icon must appear in the top-right corner of the Events section to duplicate the entire routine
3) *FR-RTA-4.2*: The section must display a scrollable list of event cards
4) *FR-RTA-4.3*: Each event must be contained in a distinct card with rounded corners and dark background

E. Event Card Structure
1) *FR-RTA-5.0*: Each event card must display a colored gradient icon on the left (unique color per event)
2) *FR-RTA-5.1*: The icon must contain a clock symbol to indicate time-based triggering
3) *FR-RTA-5.2*: Event cards must display the event name (e.g., "Untitled Event 65011")
4) *FR-RTA-5.3*: The second line must show the action state (ON/OFF), time in 24-hour format, and action type (e.g., "Set Scene")
5) *FR-RTA-5.4*: Format must be: "[STATE], [TIME], [ACTION]" (e.g., "ON, 18:00, Set Scene")
6) *FR-RTA-5.5*: A chevron (>) must appear on the right side of each card to indicate it's tappable
7) *FR-RTA-5.6*: Tapping an event card must navigate to the event detail/edit screen
8) *FR-RTA-5.7*: Event cards must support swipe gestures for quick delete actions

F. New Event Button
1) *FR-RTA-6.0*: A "New Event" button must appear at the bottom of the events list
2) *FR-RTA-6.1*: The button must have a "+" icon on the right side
3) *FR-RTA-6.2*: Tapping "New Event" must navigate to the event creation screen
4) *FR-RTA-6.3*: The button must maintain consistent styling with other UI elements (dark background, light text)

G. Event Ordering and Management

1) *FR-RTA-7.0*: Events must be automatically sorted chronologically by their trigger time
2) *FR-RTA-7.1*: Users must be able to reorder events via drag-and-drop (long-press and drag)
3) *FR-RTA-7.2*: A maximum of 20 events per routine must be enforced
4) *FR-RTA-7.3*: If a user attempts to add more than 20 events, an error message must display: "Maximum 20 events per routine"

H. Save and Validation

1) *FR-RTA-8.0*: Changes must auto-save after 2 seconds of inactivity
2) *FR-RTA-8.1*: A routine is valid only if: at least one day is selected and at least one event exists
3) *FR-RTA-8.2*: If validation fails, the back button must show the unsaved changes warning
4) *FR-RTA-8.3*: Upon successful save, a toast notification must display: "Routine saved"

H. Control Page

1) FR-008: Device Control Page:

A. Access Control

1) *FR-CTL-1.0*: Accessible only after successful device pairing (FR-003)
2) *FR-CTL-1.1*: Must verify device connectivity before displaying controls
3) *FR-CTL-1.2*: If device is offline, display "Device Offline" message with retry button

B. Power Control

1) *FR-CTL-2.0*: Display prominent ON/OFF toggle button at the top of control interface
2) *FR-CTL-2.1*: Toggle must show current device state with visual feedback (colored indicator)
3) *FR-CTL-2.2*: Power state changes must be reflected on device within 2 seconds
4) *FR-CTL-2.3*: When device is OFF, all other controls must be visually disabled/grayed out

C. Fan Speed Control

1) *FR-CTL-3.0*: Display horizontal percentage slider for fan speed control (0-100%)
2) *FR-CTL-3.1*: Slider must show current fan speed percentage as text above or beside slider
3) *FR-CTL-3.2*: Fan speed adjustments must be sent to device in real-time during slider movement
4) *FR-CTL-3.3*: Slider must be disabled when device is in Timer Mode or Auto Mode

D. Timer Mode

1) *FR-CTL-4.0*: Provide timer mode selection with options: OFF, 4hr, 6hr, 8hr
2) *FR-CTL-4.1*: Display timer as segmented control or dropdown menu
3) *FR-CTL-4.2*: When timer is active, display countdown showing remaining time
4) *FR-CTL-4.3*: Device must automatically turn OFF when timer expires
5) *FR-CTL-4.4*: User must receive push notification when timer completes: "Air purifier timer finished"
6) *FR-CTL-4.5*: Timer mode overrides Auto Mode when activated

E. Child Lock

1) *FR-CTL-5.0*: Display Child Lock toggle switch with lock icon
2) *FR-CTL-5.1*: When enabled, all physical controls on device must be disabled
3) *FR-CTL-5.2*: Child Lock state must persist until manually disabled via app
4) *FR-CTL-5.3*: Lock status must be visually indicated on device (LED indicator or display message)
5) *FR-CTL-5.4*: App controls remain functional regardless of Child Lock state

F. Operation Modes

1) *FR-CTL-6.0*: Provide mode selection between "Manual Mode" and "Auto Mode"
2) *FR-CTL-6.1*: Display current active mode prominently with visual distinction
3) *FR-CTL-6.2*: Mode changes must be confirmed with user before applying

G. Manual Mode Controls

1) *FR-CTL-7.0*: In Manual Mode, enable direct fan speed control via percentage slider
2) *FR-CTL-7.1*: Manual controls must override any automatic adjustments
3) *FR-CTL-7.2*: Display "Manual Mode" indicator when active
4) *FR-CTL-7.3*: User must manually adjust all settings (fan speed, timer, etc.)
5) *FR-CTL-7.4*: Manual Mode remains active until user switches to Auto Mode or device power cycle

H. Auto Mode Operation

1) *FR-CTL-8.0*: In Auto Mode, device must take sensor readings every 15 minutes
2) *FR-CTL-8.1*: Fan speed must automatically adjust based on environmental sensor data
3) *FR-CTL-8.2*: Display "Auto Mode" indicator and show "Next sensor reading in: X minutes"
4) *FR-CTL-8.3*: Auto Mode adjustments must consider: air quality, temperature, humidity, and detected health symptoms
5) *FR-CTL-8.4*: Fan speed slider must be disabled/read-only in Auto Mode, showing current automatic setting
6) *FR-CTL-8.5*: User must be able to view sensor reading history and auto-adjustment reasoning
7) *FR-CTL-8.6*: Auto Mode must respect Child Lock and Timer Mode settings when active

I. Control Interface Layout

1) *FR-CTL-9.0*: All controls must be organized in logical sections with clear visual separation
2) *FR-CTL-9.1*: Emergency/priority controls (Power, Child Lock) must be easily accessible
3) *FR-CTL-9.2*: Current device status must be displayed at top: power state, mode, timer status
4) *FR-CTL-9.3*: Interface must update in real-time to reflect device state changes
5) *FR-CTL-9.4*: Loading states must be shown during control command transmission

I. Settings Page

1) FR-009: Settings Page (Main Menu): Note: Only accessible after successful login/sign up

A. Settings Page Access

1) *FR-SET-1.0*: The Settings Page is accessed by tapping the "Settings" tab in the bottom navigation bar (FR-005)
2) *FR-SET-1.1*: The page must check authentication on load—expired tokens redirect to Login
3) *FR-SET-1.2*: The Settings Page must include the bottom navigation bar with the "Settings" tab highlighted/active

B. Screen Header

1) *FR-SET-2.0*: The screen must display "Settings" as the page title at the top-left
2) *FR-SET-2.1*: An optional icon (such as a palette or settings gear) may appear in the top-right corner

C. Settings Menu List

1) *FR-SET-3.0*: The screen must display a vertically scrollable list of settings options
2) *FR-SET-3.1*: Each menu item must have an icon on the left, title text in the center, and a chevron (>) on the right
3) *FR-SET-3.2*: The following options must be displayed in order:
   - Account (with user/profile icon)
   - My Devices (with device icon)
   - Location (with location/pin icon)
   - Privacy (with shield/lock icon)
4) *FR-SET-3.3*: Tapping any menu item must navigate to its respective detail screen
5) *FR-SET-3.4*: Menu items must have sufficient touch target size (minimum 44pt height)

2) FR-010: Account Screen:

A. Screen Structure

1) *FR-ACC-1.0*: Display "Account" or user's account name as the title
2) *FR-ACC-1.1*: Include a back button (<) in the top-left corner to return to Settings menu
3) *FR-ACC-1.2*: Display a section header "Account Settings"

B. Account Options

1) *FR-ACC-2.0*: Display "Edit Account" option with a chevron (>)
2) *FR-ACC-2.1*: Tapping "Edit Account" navigates to account editing screen
3) *FR-ACC-2.2*: Display "Log Out" option without a chevron
4) *FR-ACC-2.3*: Tapping "Log Out" must show confirmation dialog: "Are you sure you want to log out?"
5) *FR-ACC-2.4*: Upon logout confirmation, clear authentication token and redirect to Login screen

C. Edit Account Screen

1) *FR-ACC-3.0*: The Edit Account screen must allow users to edit their account name/display name
2) *FR-ACC-3.1*: Display current account name in an editable text field
3) *FR-ACC-3.2*: Include a "Save" button to save changes
4) *FR-ACC-3.3*: Include a "Cancel" button or back button to discard changes
5) *FR-ACC-3.4*: Show success message "Account updated" after successful save
6) *FR-ACC-3.5*: Validate that account name is not empty and is between 2-50 characters

3) FR-011: My Devices Screen:

A. Screen Structure

1) *FR-DEV-1.0*: Display "My Devices" as the screen title
2) *FR-DEV-1.1*: Include a back button (<) in the top-left corner
3) *FR-DEV-1.2*: Include a "+" button in the top-right corner to add new devices
4) *FR-DEV-1.3*: Tapping the "+" button must trigger the device registration flow (FR-003)

B. Device List Display

1) *FR-DEV-2.0*: Display all registered devices in a scrollable vertical list
2) *FR-DEV-2.1*: If no devices are registered, show empty state: "No devices registered" with "Add Device" button
3) *FR-DEV-2.2*: Each device must be displayed in a card with the following information:
   - Device icon (left side)
   - Device name with optional status indicator icon
   - Room/location name (below device name)
   - Device version or model number (below location)
   - Three-dot menu button (...) on the right side
4) *FR-DEV-2.3*: Device cards must have rounded corners and appropriate padding/spacing
5) *FR-DEV-2.4*: Cards should maintain consistent styling with other UI elements (dark background for dark mode)

C. Device Options Menu

1) *FR-DEV-3.0*: Tapping the three-dot (...) button must display a context menu/modal overlay
2) *FR-DEV-3.1*: The menu must appear as a popup/modal centered or near the device card
3) *FR-DEV-3.2*: The menu must have a light background with rounded corners (iOS-style)
4) *FR-DEV-3.3*: The menu must include the following options in order:
   - "Rename" (with edit/pencil icon)
   - "Identify" (with location/search icon)
   - "Settings" (with gear icon)
   - "Delete" (with trash/minus icon, displayed in red/warning color)
5) *FR-DEV-3.4*: Tapping outside the menu must close it without taking action
6) *FR-DEV-3.5*: Each menu option must have an icon on the right side

D. Rename Device

1) *FR-DEV-4.0*: Tapping "Rename" must open a dialog/modal with a text input field
2) *FR-DEV-4.1*: The dialog must display current device name as default value
3) *FR-DEV-4.2*: Include "Cancel" and "Save" buttons
4) *FR-DEV-4.3*: Device name must be 1-50 characters
5) *FR-DEV-4.4*: Show success message "Device renamed" after save

E. Identify Device

1) *FR-DEV-5.0*: Tapping "Identify" must send a command to the physical device to identify itself
2) *FR-DEV-5.1*: The device should flash its LED, beep, or otherwise indicate which device it is
3) *FR-DEV-5.2*: Show message "Identifying device..." while command is being sent
4) *FR-DEV-5.3*: Show success message "Device should now be blinking/beeping"

F. Device Settings

1) *FR-DEV-6.0*: Tapping "Settings" must navigate to device-specific settings screen
2) *FR-DEV-6.1*: Device settings may include: firmware version, WiFi connection, update options, etc.

G. Delete Device

1) *FR-DEV-7.0*: Tapping "Delete" must show confirmation dialog: "Remove [Device Name]? This device will be removed from your account."
2) *FR-DEV-7.1*: Dialog must have "Cancel" and "Remove" buttons
3) *FR-DEV-7.2*: "Remove" button must be in red/warning color
4) *FR-DEV-7.3*: Upon confirmation, remove device from user account
5) *FR-DEV-7.4*: Show success message "Device removed"
6) *FR-DEV-7.5*: Update the device list to remove the deleted device

4) FR-012: Location Screen:

A. Screen Structure

1) *FR-LOC-1.0*: Display "Location" as the screen title (centered)
2) *FR-LOC-1.1*: Include a back button (<) in the top-left corner
3) *FR-LOC-1.2*: Include a "Save" button in the top-right corner

B. Location Search

1) *FR-LOC-2.0*: Display a search bar below the header with placeholder text "Search"
2) *FR-LOC-2.1*: Include a magnifying glass icon on the left side of the search bar
3) *FR-LOC-2.2*: Include a location/GPS icon button on the right side of the search bar
4) *FR-LOC-2.3*: Tapping the GPS icon must request device location and auto-populate nearest city
5) *FR-LOC-2.4*: As user types, show autocomplete suggestions for cities/locations
6) *FR-LOC-2.5*: User can select a location from search results or autocomplete suggestions

C. Location Selection

1) *FR-LOC-3.0*: Selected location must be saved to the user's profile in the database
2) *FR-LOC-3.1*: Tapping "Save" must save the location and return to Settings menu
3) *FR-LOC-3.2*: Show success message "Location saved"
4) *FR-LOC-3.3*: If user taps back button without saving, show warning: "Discard changes?"
5) *FR-LOC-3.4*: Location data should include: city name, country, latitude, longitude

D. Location Usage

1) *FR-LOC-4.0*: Saved location will be used for external AQI (Air Quality Index) API integration
2) *FR-LOC-4.1*: Location will be used for weather-based automation recommendations

5) FR-013: Privacy Screen:

A. Screen Structure

1) *FR-PRIV-1.0*: Display "Privacy" as the screen title
2) *FR-PRIV-1.1*: Include a back button (<) in the top-left corner

B. Privacy Content

1) *FR-PRIV-2.0*: Display privacy policy statement or summary
2) *FR-PRIV-2.1*: Include link to full privacy policy document: "Read Full Privacy Policy"
3) *FR-PRIV-2.2*: Optionally include toggles for:
    - Data collection consent
    - Audio processing (for cough/sneeze detection)
    - Location services
4) *FR-PRIV-2.3*: Include explanation text for each privacy setting
5) *FR-PRIV-2.4*: Include "Download My Data" option
6) *FR-PRIV-2.5*: Include "Delete My Account" option at bottom in red text

6) FR-014: Empathetic Intelligence Notifications:

A. Notification Permissions

1) *FR-NOTIF-1.0*: During app setup (FR-001), request push notification permissions with explanation: "Allow notifications to receive caring health insights and device updates"
2) *FR-NOTIF-1.1*: If permissions denied initially, provide in-app prompt in Settings to enable notifications
3) *FR-NOTIF-1.2*: User must be able to customize notification types and frequency in Settings menu
4) *FR-NOTIF-1.3*: Notifications must respect device "Do Not Disturb" and quiet hours settings

B. Notification Design Principles

1) *FR-NOTIF-2.0*: All notifications must use empathetic, human-like language that demonstrates care and understanding
2) *FR-NOTIF-2.1*: Notifications must be actionable, providing clear next steps or automatic solutions
3) *FR-NOTIF-2.2*: Tone must be supportive and non-alarming, avoiding medical terminology or urgent language
4) *FR-NOTIF-2.3*: Messages must personalize to individual users when multiple users detected
5) *FR-NOTIF-2.4*: Include gentle emoji or icons where appropriate to enhance emotional connection

C. Responsive Notifications

1) *FR-NOTIF-3.0*: Monitor cough/sneeze detection events and trigger notifications when threshold exceeded
2) *FR-NOTIF-3.1*: Threshold definition: 5+ cough events or 8+ sneeze events within 30-minute window
3) *FR-NOTIF-3.2*: Example responsive notification: "We've noticed you have been coughing quite a bit lately, I've turned on Pollen Protection Mode to help out "
4) *FR-NOTIF-3.3*: Automatic action must be taken simultaneously with notification (e.g., activate protection mode)
5) *FR-NOTIF-3.4*: Follow-up notification after 2 hours: "How are you feeling? The air should be much cleaner now"
6) *FR-NOTIF-3.5*: Avoid sending duplicate responsive notifications within 4-hour window for same symptom type

D. Predictive Notifications

1) *FR-NOTIF-4.0*: Integrate external AQI data with user health sensitivity profiles to predict issues
2) *FR-NOTIF-4.1*: Learn individual user sensitivities over time (Person A reacts to pollen, Person B to dust, etc.)
3) *FR-NOTIF-4.2*: Multi-user household example: "Pollen levels in the area are rising and [Person A] might have some struggles, we recommend moving the purifier to [Person A]'s room "
4) *FR-NOTIF-4.3*: Location-specific example: "The ultrafine dust level in [Location] is high this afternoon, and I've detected some sneezing. I've activated 'Pollution Defense' mode to create a safe zone for you at home. Please remember to keep the windows closed! "
5) *FR-NOTIF-4.4*: Predictive notifications must be sent 1-2 hours before predicted air quality deterioration
6) *FR-NOTIF-4.5*: Include specific recommendations: window closure, room changes, device repositioning
7) *FR-NOTIF-4.6*: Track prediction accuracy and adjust sensitivity thresholds based on user feedback

E. Maintenance Notifications

1) *FR-NOTIF-5.0*: Monitor filter life and send proactive replacement notifications
2) *FR-NOTIF-5.1*: Filter warning at 90% capacity: "Your filter is running low! Replace it soon to avoid air quality issues "
3) *FR-NOTIF-5.2*: Critical filter notification at 100% capacity: "Time for a fresh filter! I've reduced fan speed to protect the motor until you can replace it"
4) *FR-NOTIF-5.3*: Include direct link to purchase replacement filters or schedule maintenance
5) *FR-NOTIF-5.4*: Send reminder notifications every 3 days after initial filter warning
6) *FR-NOTIF-5.5*: Other maintenance notifications: cleaning reminders, sensor calibration, software updates

F. Notification Timing and Frequency

1) *FR-NOTIF-6.0*: Respect user's local time zone and avoid notifications between 10 PM - 7 AM unless critical
2) *FR-NOTIF-6.1*: Maximum 3 notifications per day per device to avoid notification fatigue
3) *FR-NOTIF-6.2*: Priority system: Critical (filter/safety) > Predictive > Responsive > General updates
4) *FR-NOTIF-6.3*: User must be able to snooze non-critical notifications for 1hr, 4hr, or until tomorrow
5) *FR-NOTIF-6.4*: Smart batching: combine multiple low-priority notifications into daily summary

G. Notification Customization

1) *FR-NOTIF-7.0*: Settings menu must include notification preferences with toggles for each notification type
2) *FR-NOTIF-7.1*: Allow users to set "empathy level": Minimal, Standard, Caring, Very Caring
3) *FR-NOTIF-7.2*: Empathy level affects message tone, frequency, and personalization depth
4) *FR-NOTIF-7.3*: Quick notification response options: "Thanks!", "Not helpful", "Remind me later"
5) *FR-NOTIF-7.4*: Learn from user responses to improve future notification relevance and timing

H. Emergency and Safety Notifications

1) *FR-NOTIF-8.0*: Override quiet hours for safety-critical notifications (device malfunction, air quality emergency)
2) *FR-NOTIF-8.1*: Emergency example: "Air purifier has stopped working! Please check the device and ensure proper ventilation"
3) *FR-NOTIF-8.2*: Severe air quality example: "Air quality in your area is hazardous. Please stay indoors and run the purifier on maximum"
4) *FR-NOTIF-8.3*: Safety notifications must be persistent until acknowledged by user
5) *FR-NOTIF-8.4*: Include emergency contact information or support links when appropriate

III. Development environment

A. *Choice of Software Development Platform*

- **Host OS**: Windows 11 Build 26200.6899/7019
- **Language**: Python 3.11.9/3.12.5 for rapid prototyping, strong audio and ML ecosystem
- **Target Edge**: Ideal LG Purifier Hardware (for convenience).

B. *Software in Use*

- **ML**: TensorFlow, Scipy, librosa, scikit-learn.
- **Backend**: Firebase, Heroku
- **Database**: Firestore Database, PostgreSQL
- **DevOps**: VS Code, Git/GitHub, Git Projects, Docker

TABLE II
Task Distribution

Jobs	Name	Description
Frontend	Yim Soobeen, Kim Yeonwoo	In charge of the planning, development and rollout of the PWA onto mobile devices.
Backend	Kim Yeonwoo	In charge of database and API creation, management, and maintainment
ML/AI	Jimmy MacDonald, Kim Minjin, et Al	In charge of collecting data, planning and creating an AI model to achieve the task at hand
Project Manager	Kim Minjin	Keeps track of documentation, assigns tasks, and make sure deadlines are met on time

IV. Specification

A. *User Authentication*

A. Technology Stack

- Frontend: React with Google OAuth 2.0 integration
- Backend: Firebase Authentication via Admin SDK
- Token Management: Google ID Tokens with sessionStorage persistence
- Session Handling: Token-based authentication with automatic refresh

B. Authentication Flow

1) User clicks "Login with Google"
2) User grants permissions
3) Frontend receives Google ID token
4) Token stored in sessionStorage as JSON: { "idToken": "...", "refreshToken": "...", "expiresIn": 3600 }
5) Token included in Authorization header for all API requests
6) Backend validates token via Firebase Admin SDK
7) User authenticated and redirected to Home Page

C. Token Verification Process

1) Frontend retrieves token from localStorage: purecare_auth
2) All API requests include header: Authorization: Bearer {idToken}
3) Backend validates using Firebase Admin SDK

D. Security Measures

- HTTPS-only communication in production
- Token expiration: 1 hour (auto-refresh via Firebase SDK)
- Secure sessionStorage (inaccessible to external scripts)
- CORS configuration limits allowed origins to frontend URL only

B. *Real-Time Device Control System*

A. Communication Architecture

- Protocol: WebSocket (Socket.io v4)
- Transport: Bidirectional event-based communication
- Fallback: HTTP long-polling if WebSocket unavailable
- Connection: Persistent connection maintained per client

B. WebSocket Communication Flow

1) Frontend
2) Backend
3) Device

C. State Synchronization

- Optimistic UI Updates: Frontend immediately reflects user changes
- Confirmation: Backend acknowledges command receipt within 200ms
- Device Execution: Hardware simulator applies changes and confirms
- Broadcast: Backend emits state update to all clients monitoring device
- Timeout Handling: 3-second timeout with 2 automatic retries
- Error Recovery: Rollback UI state if command fails

D. Controls
- Power
  - type: "power"
  - value: true|false
  - deviceId
  - timestamp
- Fan Speed
  - type: "fan_speed"
  - value: 0-10
  - deviceId
- Mode Selection
  - type: "auto_mode"
  - value: true|false
  - deviceId
- Sensitivity
  - type: "sensitivity"
  - value: "low" | "medium" | "high"

C. Sensor Data Collection & Time-Series Database

A. Database Architecture
- Primary: PostgreSQL with TimescaleDB extension (Heroku Postgres Mini)
- Schema: Hypertable optimized for time-series data
- Aggregation: Continuous aggregates for hourly/daily summaries
- Retention: 7 days raw data, 30 days hourly, 1 year daily

D. Hardware Simulator (Python-based IoT Mock)

A. Purpose
- In order to simulate our air purifier without having to purchase hardware, we made a python simulator to act as a mock device.

B. Simulation Logic
- To initialize our variables, we take data from our cached database of the nearest station and set those as our active measurements.
- Connect to our websocket so that our information shows up in the database and furthermore in our front end.
- Send our current readings every `UPDATE_INTERVAL` (default 15 seconds) through REST API.
- If auto mode is on and if $PM_{2.5}$ is greater than 35, set the fan speed to the nearest integer corresponding to the total $PM_{2.5}$ / 10, with a maximum speed of 10
- Reduction rate of particulate matter is determined on fan speed, calculated by 0.05 * fan speed. This is updated on each update interval.
- Randomly events will appear based on the update interval.

C. Realistic Scenarios
- To simulate a window being open, for each of these outside variables that aren't 0 ($PM_{2.5}$, PM_{10}, NO_2), multiply the difference between the outdoor readings and the indoor readings by 0.15 and add it to the current sensor data
- To simulate cooking, to each of these variables ($PM_{2.5}$, PM_{10}, NO_2, TVOC, CO) we add a random uniform variable within norms of their respective scales.

D. Integration with Device Control
- While the settings of the device can be controlled through the python script, updating its configuration in the front end will also update the simulator's values.

E. Automation Routine System

A. Routine Data Structure (Firebase Firestore)
- Routines use a fan-out architecture. The master copy lives under the users collection. This is where the front end creates and modifies routines. The local copy lives under the devices collection and is what the hardware reads.
- Using a cloud function/trigger, it listens to the master copy of the routines and then updates the local copy upon change
- Each routine has these fields: name, enabled, a schedule object containing time, days, and timezone_offset, an action object with a type, mode, and duration, and an array of target devices. The local hardware device for each routine also keeps track of whether the routine has ran today.

B. Execution Logic
1) Every second we get the current day, minute, and hour and iterate through all the routines for the device.
2) If all of the conditions of a routine are met, the action is started.

F. Notification System with Empathetic Intelligence

1) Notification Types:

2) Personalization Engine:

3) Delivery System:

4) Rate Limiting:

G. Audible Health Symptom Detection

A. System Architecture: The PureCare Air Purifier implements a two-stage neural network architecture for intelligent cough detection and classification, replacing traditional threshold-based approaches with sophisticated machine learning inference.

1) Cough Detection Model
    - Architecture: Lightweight 1D Convolutional Neural Network (CNN)
    - Task: Binary classification (cough vs. non-cough)
    - Input: 39-dimensional MFCC features over 1.5-second audio windows
    - Output: Sigmoid activation producing probability score [0,1]
    - Decision threshold: 0.7 confidence for positive detection
    - Design priority: Real-time inference speed and low computational overhead
2) Cough Classification Model
    - Architecture: Hybrid CNN-LSTM network
    - Task: Multi-class classification of cough types
    - Purpose: Identifies specific cough characteristics for health insights
    - Output: Softmax activation over N cough categories
    - Temporal modeling: LSTM layer captures sequential dependencies in cough patterns
    - Activation: Only processes audio segments flagged by Stage 1 detection

B. Feature Extraction

- MFCC (Mel-Frequency Cepstral Coefficients): The system employs MFCC feature extraction, a standard technique in audio signal processing that mimics human auditory perception
    - Sample Rate: 8kHz (optimized for low-frequency respiratory sounds)
    - Window Size: 1.5 seconds (captures complete cough events)
    - Base MFCCs: 13 coefficients
    - Delta Features: 13 first-order derivatives (velocity of spectral change)
    - Delta-Delta Features: 13 second-order derivatives (acceleration of spectral change)
    - Total Feature Dimensions: 39 per time frame
    - FFT Window: 512 samples with 256-sample hop length
    - Mel Filterbank: 40 triangular filters spanning 50Hz to 4000Hz
- Feature Normalization
    - All extracted features undergo zero-mean unit-variance normalization to improve neural network training stability and convergence:

$$\text{normalized} = \frac{\text{features} - \mu}{\sigma}$$

      where μ and σ are computed per-feature across the time dimension

C. Live Detection Pipeline

- Audio Acquisition and Buffering
    1) Continuous audio stream captured at 8kHz, mono channel
    2) Circular buffer maintains 10 seconds of recent audio (80,000 samples)
    3) Thread-safe deque structure with mutex-protected access
    4) Block size: 2048 samples for efficient callback processing
- Sliding Window Detection
    1) Detection Window: 1.5 seconds (12,000 samples)
    2) Hop Length: 0.5 seconds (4,000 samples) — 66% overlap between windows
    3) Processing Rate: Approximately 2 windows per second
    4) Overlap strategy: Ensures no cough events are missed between windows
- Energy Pre-filtering (Optional Optimization): Before neural network inference, a fast RMS energy calculation filters silent segments:

$$\text{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

    - Energy Threshold: 0.01 (configurable)
    - Purpose: Reduces unnecessary NN inference on silence, improving efficiency
    - Computational cost: O(N) vs. O(N log N) for MFCC extraction
- Neural Network Inference
    1) Extract MFCC features from 1.5s window: Shape (39, 47 frames)
    2) Stage 1: Pass features through detection CNN
    3) If detection confidence >= 0.7: Flag as cough event
    4) Stage 2: Pass same features through classification CNN-LSTM
    5) Extract cough type/category and confidence score
    6) Log detection with timestamp, confidence, and classification

D. Training and Data Augmentation

- Dataset Preparation
  - Train/Validation/Test Split: 70%/15%/15%
  - Fixed random seed for reproducible splits
  - Stratified sampling maintains class balance across splits
- Data Augmentation Techniques To improve model generalization and robustness to real-world variations:
  1) Time Shifting: Circular shift of up to ±20% of signal length
  2) Speed Perturbation: Time-stretching with rate factor in [0.9, 1.1]
  3) Additive Noise: White noise or environmental noise at SNR between -5dB and 15dB
  4) Application: Random combination of augmentations during training only
- Training Configuration **Detection Model:**
  - Optimizer: Adam with learning rate 0.001
  - Loss Function: Binary cross-entropy
  - Batch Size: 32
  - Epochs: 50 (with early stopping, patience=10)
  - Learning Rate Schedule: ReduceLROnPlateau (factor=0.5, patience=5)
  - Metrics: Accuracy, Precision, Recall, AUC

  Classification Model:
  - Optimizer: Adam with learning rate 0.0001
  - Loss Function: Categorical cross-entropy
  - Batch Size: 32
  - Epochs: 100 (with early stopping, patience=15)
  - Learning Rate Schedule: ReduceLROnPlateau (factor=0.5, patience=7)
  - Metrics: Categorical accuracy, Top-2 accuracy

E. Model Architecture Details

- Detection Model (1D CNN)
  1) Input: (39 features, 47 time frames)
  2) Permutation: Transpose to (47 time frames, 39 features) for 1D convolution
  3) Conv1D Block 1: 32 filters, kernel=3, ReLU $\rightarrow$ BatchNorm $\rightarrow$ MaxPool(2) $\rightarrow$ Dropout(0.3)
  4) Conv1D Block 2: 64 filters, kernel=3, ReLU $\rightarrow$ BatchNorm $\rightarrow$ MaxPool(2) $\rightarrow$ Dropout(0.3)
  5) Conv1D Block 3: 128 filters, kernel=3, ReLU $\rightarrow$ BatchNorm $\rightarrow$ MaxPool(2) $\rightarrow$ Dropout(0.3)
  6) Global Average Pooling: Reduces temporal dimension
  7) Dense: 64 units, ReLU $\rightarrow$ Dropout(0.3)
  8) Output: 1 unit, Sigmoid activation
  9) Total Parameters: 50K (lightweight for edge deployment)
- Classification Model (CNN-LSTM)
  1) Input: (39 features, 47 time frames)
  2) Permutation: Transpose to (47 time frames, 39 features)
  3) Conv1D Block 1: 64 filters, kernel=3, ReLU $\rightarrow$ BatchNorm $\rightarrow$ MaxPool(2) $\rightarrow$ Dropout(0.4)
  4) Conv1D Block 2: 128 filters, kernel=3, ReLU $\rightarrow$ BatchNorm $\rightarrow$ MaxPool(2) $\rightarrow$ Dropout(0.4)
  5) Conv1D Block 3: 256 filters, kernel=3, ReLU $\rightarrow$ BatchNorm $\rightarrow$ MaxPool(2) $\rightarrow$ Dropout(0.4)
  6) LSTM Layer: 128 units (captures temporal dependencies in cough dynamics)
  7) Dropout(0.4)
  8) Dense: 128 units, ReLU $\rightarrow$ Dropout(0.4)
  9) Output: N units (number of cough classes), Softmax activation
  10) Total Parameters: 400K

H. *Cost Analysis & Infrastructure*

TABLE III
HEROKU DEPLOYMENT COSTS

Service	Plan	Monthly Cost	Features
Web Dyno	Hobby	$7	Always-on, 512MB RAM
Postgres	Essential	$5	10GB storage, TimescaleDB
Total		**$12/month**	26 months with $312 credit

1) Data Volume Estimates:

- Per Device
  - 8 sensors × 288 readings/day (every 5 min) = 2,304 records/day
  - 7 days raw data = 16,000 records = $\tilde{1}$MB

2) Performance Metrics:

- WebSocket latency <100ms
- Sensor data processing: <50ms
- Database query time: <200ms (with TimescaleDB indexes)
- Frontend load time: <2s

V. Architecture Design & Implementation

A. Overall Architecture

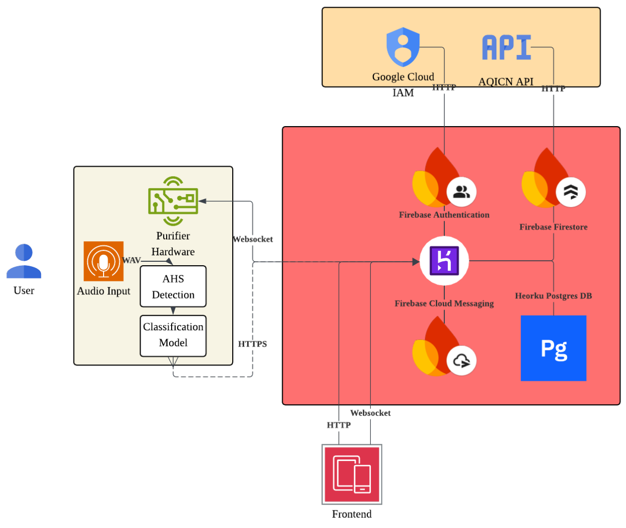

Fig. 1. Network Architecture

B. Directory Organization

TABLE IV
Front end directory

Directory	Key Files	Module Components	Description
`/client/src/app/[locale]/(auth)/`	`login/page.tsx signup/page.tsx`	User Authentication UI	Google OAuth 2.0 login/signup screens (FR-002)
`/client/src/app/[locale]/(core)/`	`home/page.tsx`	Home Page	Main dashboard with device list, AQI display, user greeting (FR-004)
`/client/src/app/[locale]/automation/`	`page.tsx`	Automation/Routine System UI	Routine management screen with routine cards (FR-006)
`/client/src/app/[locale]/room/[id]/`	`page.tsx`	Device Control Page	Real-time device control interface with power, fan speed, timer, child lock controls (FR-008)
`/client/src/app/[locale]/settings/`	`page.tsx account/page.tsx devices/page.tsx location/page.tsx privacy/page.tsx`	Settings Module	Account settings, device management, location selection, privacy controls (FR-009 to FR-013)

TABLE V
Front end directory contd.

Directory	Key Files	Module Components	Description
`/client/src/app/[locale]/devices/add/`	`page.tsx qr/page.tsx qr/confirm/page.tsx serial/page.tsx serial/success/page.tsx`	Device Registration	QR code scanning and serial number entry for device pairing (FR-003)
`/client/src/app/[locale]/report/`	`page.tsx`	Data Visualization	Sensor data reports and analytics dashboard
`/client/src/app/[locale]/weather/`	`page.tsx`	Weather Integration	External AQI and weather data display interface
`/client/src/app/api/`	`forecast/route.ts geocode/route.ts weather/route.ts`	API Routes	Next.js API endpoints for weather and geocoding services
`/client/src/components/features/`	aqi-trend-chart.tsx device-carousel.tsx kakao-map.tsx welcome-modal.tsx	Feature Components	Reusable UI components for charts, device displays, maps
`/client/src/components/layout/`	bottom-nav.tsx client-layout-wrapper.tsx splash.tsx	Layout Components	Navigation bar (FR-005), splash screen, layout wrappers
`/client/src/components/ui/`	demo-mode-banner.tsx segmented-control.tsx	UI Components	Shared UI elements like segmented controls, banners
`/client/src/lib/`	`api.ts auth.tsx firebase.ts fcm.ts weather.tsx`	Client Libraries	API client, authentication logic, Firebase config, FCM push notifications (FR-014), weather API integration
`/client/src/i18n/`	`request.ts routing.ts`	Internationalization	Multi-language support (i18n) routing and request handling

TABLE VI
BACKEND DIRECTORY

Directory	Key Files	Module Components	Description
/server/src/api/	index.js	Express Server Entry	Main server initialization with Socket.io WebSocket integration
/server/src/api/config/	firebase.js	Firebase Configuration	Firebase Admin SDK initialization for authentication
/server/src/api/database/	init.js	Database Initialization	PostgreSQL / TimescaleDB connection setup
/server/src/api/middleware/	auth.js rateLimiter.js	Authentication & Security	JWT token verification middleware, API rate limiting
/server/src/api/routes/	auth.js controlRoutes.js deviceRoutes.js roomRoutes.js sensorRoutes.js userRoutes.js notificationRoutes.js	REST API Endpoints	Authentication, device control commands, device registration, room management, sensor data queries, user profile, push notifications
/server/src/api/services/	deviceService.js sensorDataService.js notificationService.js timezoneService.js	Business Logic Services	Device state management, time-series data aggregation, empathetic notification generation (FR-014), timezone handling
/server/src/api/scripts/	aqiScripts.js midnightRoutine.js processNotifications.js	Background Jobs & Scripts	AQI data fetching, daily reset routines, notification batch processing

TABLE VII
HARDWARE SIMULATOR DIRECTORY

Directory	Key Files	Module Components	Description
/hardware/simulator/	__main__.py run_simulator.py	Simulator Entry Point	Main Python script to launch air purifier simulator
/hardware/simulator/core/	device.py audio_detector_wrapper.py	Device Simulation Logic	Air purifier behavior simulation: PM reduction rate ($0.05 \times$ fan_speed), auto mode logic ($PM_{2.5} > 35$), sensor value updates every 15s
/hardware/simulator/models/	commands.py responses.py sensor_data.py	Data Models	Command structures (power, fan_speed, auto_mode, sensitivity), response formats, sensor reading schemas
/hardware/simulator/communication/	websocket_client.py http_client.py event_sender.py	Real-Time Device Control	WebSocket client for bidirectional Socket.io communication, REST API client for sensor data upload, event emission for cough/sneeze detection
/hardware/simulator/config/	settings.py constants.py	Configuration	Update intervals (default 15s), AQI station data, device serial numbers, server URLs
/hardware/shared/events/	bus.py	Event System	Event bus for inter-process communication between simulator and audio detector
/hardware/	cli_controller.py send_command.py	CLI Control	Command-line interface for manual device control

TABLE VIII
ML/AI AUDIO DETECTION DIRECTORY

Directory	Key Files	Module Components	Description
/hardware/audio_analyzer/	train.py evaluate_model.py live_detection.py	Model Training & Inference	Train detection/classification CNNs, evaluate model performance, real-time audio detection pipeline
/hardware/audio_analyzer/	models.py	Neural Network Architecture	Stage 1: 1D CNN for binary cough detection (39 MFCCs, 0.7 confidence threshold, ∼50K params). Stage 2: CNN-LSTM for cough classification (128 LSTM units, ∼400K params)
/hardware/audio_analyzer/	feature *extraction.py*	Audio Feature Engineering	MFCC extraction (8kHz, 13 base + 13 delta + 13 delta-delta = 39 features), 1.5s windows with 0.5s hop length, zero-mean unit-variance normalization
/hardware/audio_analyzer/	data_loader.py prepare_dataset.py download_audeering_dataset.py	Dataset Management	70/15/15 train/val/test split, stratified sampling, data augmentation (time shifting, speed perturbation, additive noise)
/hardware/audio_analyzer/ hardware/AI/models/	detection_model_best.h5 classification_model_best.h5 *.json	Trained Models & Metadata	Best detection model checkpoint, best classification model checkpoint, training history logs, fixed data split configuration
/hardware/audio_analyzer/ hardware/AI/cough_dataset/	cough/ non_cough/ sneeze/ speech/	Training Dataset	Labeled audio samples for model training (multi-class categories)
/hardware/tools/	audio_capture.py playback.py segmentation.py	Audio Utilities	Audio recording tools, playback utilities, audio segmentation for dataset preparation

C. Next.js Frontend

For the PureCare Air Purifier to have empathetic intelligence, we needed a way to deliver this sentiment to the user. To do this, we made a Next.js based Progressive Web App. The app allows us to communicate and receive inputs from users, making for a truly responsive design. Within the app, users are able to register, control, and receive real time data from the air purifier. The source code for the front end can be found in the `client` folder.

D. Express Backend

In order to fulfill key essential functions for the device, we needed to design some sort of server to support our devices and front end. To resolve this issue, we designed a fully custom express.js backend to power our services. The backend, which is hosted on Heroku, allows us to be a middleman between the physical hardware and the user software, while also giving crucial information for the hardware so that it can act proactively rather than reactively.

E. Hardware

As the main product for our service, the hardware serves as the very heart of our project. Our hardware serves to build off pre existing *LG PuriCare Air Purifiers* and enhance them to provide a better experience for users.

VI. USE CASES

A. Login with Google Account

TABLE IX
UC-1. LOGIN WITH GOOGLE ACCOUNT

User	System Response
The user launches the app and views the login screen.	
The user selects the 'Google Account' button.	The system initiates the Google OAuth authentication process.
The user selects their account and logs in.	The system validates the idToken and stores it in the sessionStorage.
	Upon successful login, the system navigates the user to the Home screen.
	[Alternative 3a] If login fails, the system displays a failure message.

B. Demo Mode

TABLE X
UC-2. USE APP IN DEMO MODE

User	System Response
The user selects the 'Try the Demo' button on the login screen.	The system activates the isDemoMode() flag to switch context to demo mode.
	The system simulates network latency using mockDelay to mimic real API behavior.
The user views the Home screen.	The system loads MOCK-DEVICES and MOCK-ALERTS data instead of querying the live server.

C. Check Air Quality & Outdoor Weather

TABLE XI
UC-3. CHECK AIR QUALITY SUMMARY OUTDOOR WEATHER

User	System Response
The user views the Home screen.	The system fetches the device list via getDevices().
	The system calculates real-time AQI from $PM_{2.5}$ data using the simplified EPA formula and displays the status label.
The user checks the outdoor information section.	The system fetches outdoor AQI data (getOutdoorAQI) based on the device's station index.
The user selects the outdoor information section.	The system navigates to the detailed weather and map screen.

D. Add a device

TABLE XII
UC-4. ADD AIR PURIFIER DEVICE

User	System Response
The user selects the 'Add Device' button.	The system displays options: 'QR Code Scan' or 'Enter Serial Number'.
The user selects a method and enters/scans the info.	The system sends a POST request to /api/devices/register with the deviceId, name, and location.
The user completes the registration.	The system verifies the response (success: true) and registers the device to the user account. The system updates the device list on the Home screen.

E. Check Energy Report

TABLE XIII
UC-5. CHECK ENERGY USAGE REPORT

User	System Response
The user selects the 'Report' tab.	The system initializes the view with the default range 'Today'.
The user switches tabs (Today / This Week / This Month).	The system fetches usage, cost, and saving data corresponding to the selected range. The system renders a TinyBarChart component to visualize usage patterns.
The user checks the 'Energy Savings Progress' section.	The system displays a progress bar comparing the current savings rate against the monthly goal.

F. Manage Room Structure & Connections via Room Graph

TABLE XIV
UC-6. MANAGE ROOM STRUCTURE & CONNECTIONS VIA ROOM GRAPH

User	System Response
The user selects 'Room Graph' on the Home screen.	The system fetches room and connection data via getRooms and getRoomEdges APIs. The system renders nodes and edges using the React Flow library.
The user clicks '+ New Room' and enters a name.	The system calls the createRoom API and generates a new node at a random position.
The user drags a room node to a new position.	The system updates the node's coordinates on the screen in real-time.
The user releases the node (Drop).	The system calls the updateRoom API to persist the new (x, y) coordinates.
The user connects two room nodes.	The system calls the createRoomEdge API and renders a default 'Door' edge.
The user taps a connection edge.	The system calls updateRoomEdgeType to toggle the type to 'Airflow' and enables the flow animation.

G. Manage Automation Settings

TABLE XV
UC-7. MANAGE AUTOMATION SETTINGS

User	System Response
The user selects ‘Automation’ from the menu.	The system loads the list of configured Routines and Schedules.
The user toggles an automation to ’Active’.	The system updates the automation state to ’Active’ and sends the request to the server.
(Background) The system receives sensor data.	The system evaluates if the condition (e.g., $PM_{2.5} > 35$) is met. If triggered, the system automatically executes the defined action.

H. Check AQI Trend Report

TABLE XVI
UC-8. CHECK AQI TREND REPORT

User	System Response
The user navigates to the AQI Trend section.	The system initiates a call to the getHistoricalSensorData API.
The user selects a time range.	The system queries historical data using startTime, endTime, and limit parameters. The system renders a line chart showing trends for $PM_{2.5}$, CO_2, and Temperature.

I. Check and Manage User Account Information

TABLE XVII
UC-9. CHECK AND MANAGE USER ACCOUNT INFORMATION

User	System Response
The user selects ’Account’ in Settings.	The system displays the user profile information stored in the session.
The user selects Logout.	The system removes the purecare-auth token from sessionStorage. The system redirects the user to the login page (/login).

J. Manage Privacy and Location-Based Settings

TABLE XVIII
UC-10. MANAGE PRIVACY AND LOCATION-BASED SETTINGS

User	System Response
The user selects ’Location’ or ’Privacy’.	The system displays the current data collection consent status.
The user modifies the settings.	The system updates the user preferences on the server.

K. Device Addition Method (Detail)

TABLE XIX
UC-11. DEVICE ADDITION METHOD (DETAIL)

User	System Response
The user selects ’QR Code Scan’.	The system activates the camera module for scanning.
The user scans the QR code.	The system decodes the QR code to extract the deviceId.
The user selects ’Serial Number’.	The system provides an input form for manual entry.

L. Manage Registered Devices

TABLE XX
UC-12. MANAGE REGISTERED DEVICES

User	System Response
The user selects a device from the list.	The system displays the control panel.
The user adjusts settings.	The system calls control APIs such as setFanSpeed or toggleAutoMode.
The user selects ’Delete Device’.	The system calls the deleteDevice API and removes the device from the list.

VII. DISCUSSION

One of the biggest issues in this project was the language barrier which lead to many problems. Clear communication was a struggle and keeping track of what each other had done, despite the use of the github projects, was a real issue.

REFERENCES

[1] World Health Organization, "Billions of people still breathe unhealthy air: new WHO data," WHO News, 04-Apr-2022. [Online]. Available: https://www.who.int/news/item/04-04-2022-billions-of-people-still-breathe-unhealthy-air-new-who-data (Accessed: 01-Oct-2025).

[2] IEEE S. Amiriparian et al., "CAST a database: Rapid targeted large-scale big data acquisition via small-world modelling of social media platforms," in 2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII), 2017, pp. 340–345, doi: 10.1109/ACII.2017.8273622.