Airwise

Project Documents

Course:
Integrative Software Engineering
Afeka - semester B - 2025

Lecturer:
Eyal Eisenstein

Students:
Avital Iskhakov
Islam Saadi
Maor Mordo
Shani haker
Ariel Arviv

Table Of Content:

1.SRS

AirWise Project Requirements

Course Name:

Integrated Software Engineering

Project Name:

Airwise – Smart Air-Conditioning Management System

Team Members:

ID	Name	Roles	Avatar
	Islam Saadi	- Scrum Master - QA Engineer - Team	
	Avital Iskhakov	- Team Leader - Product Owner - Team	
	Maor Mordo	- Devops - System Architect - Team	
	Ariel Arviv	- DBA - Technical Writer - Team	
	Shani Haker	- UX Engineer - Team	

Submission Date:

18.06.2025

1. Introduction:

This project is an innovative air-conditioning management system designed to provide users with full control and insights over their environment and energy consumption through a centralized software platform.

The software enables both individual control of air conditioners and group management, while also tracking usage patterns, power consumption, scheduling operations, and issuing notifications.
The overall vision is to support smarter climate control that is energy-aware, user-friendly, and adaptable to different environments.

This system is designed following the Ambient Invisible Intelligence principles as outlined by Gartner, aiming to deliver seamless, context-aware environmental control while reducing user intervention and maximizing efficiency.

1.1. Purpose of System:

This system aims to provide a smart, centralized solution for managing air conditioning environments across various settings — from individual users in homes to administrators overseeing large-scale installations.

By enabling remote control, automation, and personalized configurations of air conditioners, the system aims to enhance energy efficiency, reduce operational costs, and improve overall user comfort.

Beyond simple control, it empowers users with meaningful insights into power consumption patterns, environmental conditions, and system behavior through advanced monitoring and scheduling tools.

The business value lies in streamlining device management, supporting scalable deployment, and offering actionable data that facilitates smarter decisions, whether for cost-saving, maintenance, or user experience. In doing so, the system addresses both consumer convenience and operational efficiency, positioning itself as a modern, adaptable platform for intelligent climate management.

1.2. Scope of System

In Scope:

- Managing user account
- Controlling air conditioners individually or as groups
- Interacting with Air Conditioner units via demoAC API
- Creating and managing Schedules for automated tasks
- Collecting power consumption data
- Sending notifications
- Customizing the system via Settings and Preferences

Out of Scope:

- Physical hardware control (actual firmware on the AC units or sensors)
- External integration with third-party smart home systems
- Billing or payment systems
- Mobile application development

2. Actors and goals:

Actor Name	Type	Description
Tenant	Primary Actor	A regular user of the system who manages their personal AC units, schedules, and preferences to optimize comfort and energy use.
Air Conditioner	Supporting Actor	A virtual representation of the AC units that interact with the system via API, reporting status and executing commands.
Time	Supporting Actor	Responsible for managing timed events and scheduling tasks within the application.

2.1. Actor #1

Actor: Tenant

Type: Primary

Description: A user of the system who operates and customizes air conditioning behavior for personal comfort and efficiency. Tenants interact with the interface to manage their own climate preferences and benefit from automation and insights.

Goals:

- Achieve optimal indoor climate control
- Benefit from automated scheduling to simplify daily routines
- Gain insights into energy usage and cost-saving opportunities
- Receive timely notifications on system events
- Personalize the AC experience based on personal preferences and routines

2.2. Actor #2

Actor: Air Conditioner

Type: Supporting Actor

Description: Represents the virtual AC units the system interacts with through the demoAC API. Though not a human user, it is involved in many interactions where the system sends or receives AC status, state changes, or alerts.

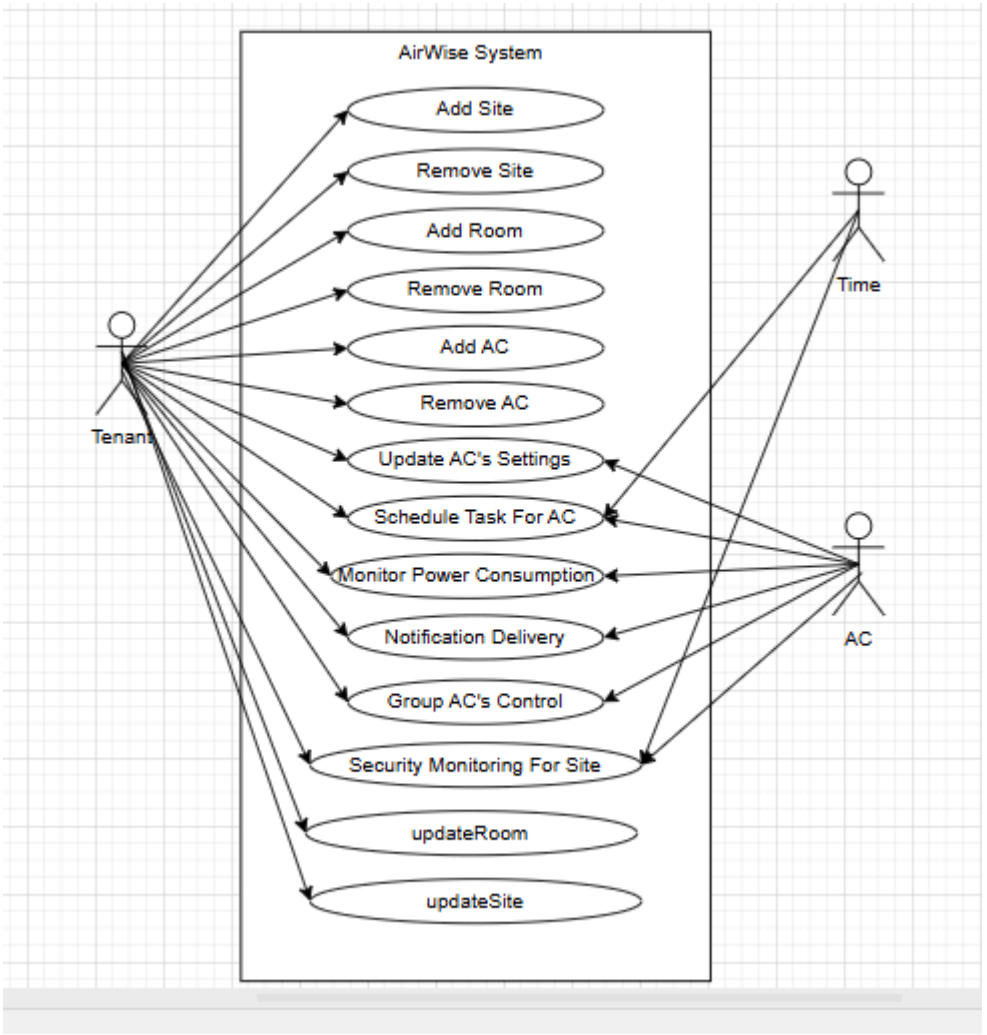
2.3. Actor #3

Actor: Time

Type: Supporting Actor

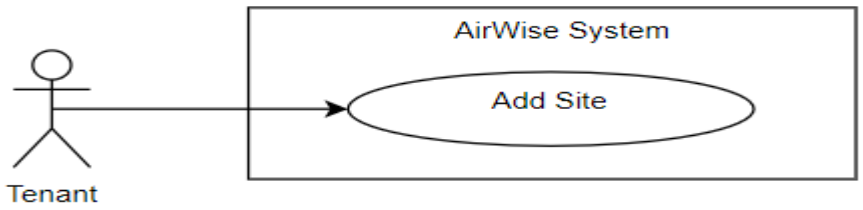
Description: Represents scheduled time triggers that cause the system to execute actions such as turning on AC units.

3. Functional Requirements



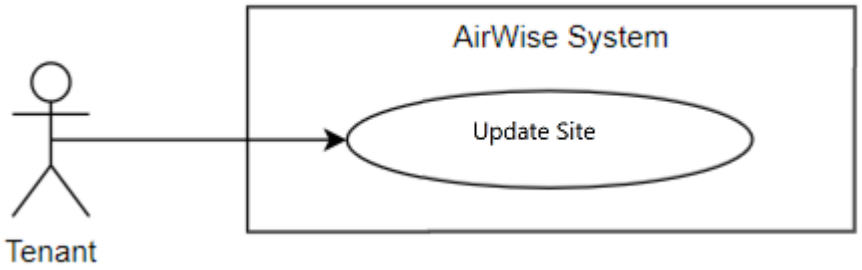
3.1 Use Cases With Diagrams

Use Case 1: Add Site



- **Goal in Context:** Allow a tenant to create a new site where rooms and air conditioners will be managed.
- **Primary Actor:** Tenant
- **Supporting Actors:** None
- **Preconditions:**
  a. Tenant is logged in.
- **Postconditions:**
  a. A new site is added to the system and linked to the tenant.
- **Main Success Scenario:**
  1. Tenant navigates to the AC Control screen.
  2. Tenant enters site details.
  3. System validates the input fields.
  4. System saves the site information and links it to the tenant's account.
  5. System displays confirmation and updates the list of available sites.
- **Alternate Flows:**
  a. **Site name is missing or invalid:** System prompts tenant to correct the input.
  b. **Save operation fails:** System displays failure message.

Use Case 2: Update Site Name



Goal in Context:
Allow a tenant to modify the name of an existing site they have created.

Primary Actor: Tenant
Supporting Actors: None

Preconditions:

- Tenant is logged in.
- At least one site exists under the tenant's account.

Postconditions:

- The selected site's name is updated in the system.

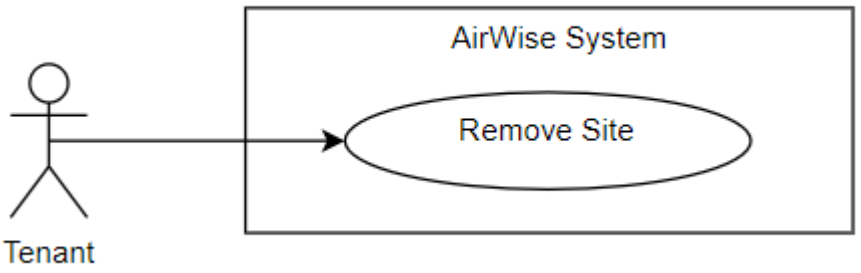
Main Success Scenario:

1. Tenant navigates to the AC Control screen.
2. Tenant selects a site to update.
3. System displays the current site name.
4. Tenant modifies the name of the site.
5. System validates the updated name.
6. System saves the new site's name.
7. System reflects the changes in the site list.

Alternate Flows:

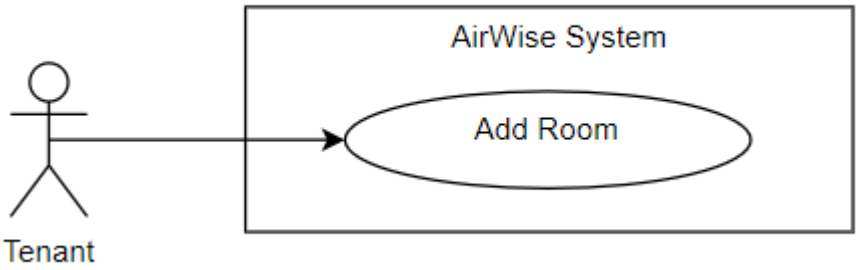
- missing field:
  System prompts the tenant to correct the input.
- Update operation fails:
  System displays an error message and does not save changes.

Use Case 3: Remove Site



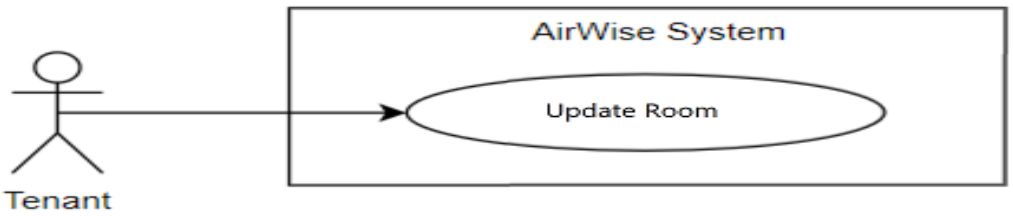
- **Goal in Context:** Allow a tenant to delete a site they previously created, along with its associated rooms and devices.
- **Primary Actor:** Tenant
- **Supporting Actors:** None
- **Preconditions:**
  a. Tenant is logged in.
  b. The site to be removed exists and is associated with the tenant.
- **Postconditions:** The selected site is deleted along with all nested rooms and devices.
- **Main Success Scenario:**
  1. Tenant navigates to the AC Control screen.
  2. System displays a list of sites associated with the tenant.
  3. Tenant selects a site to remove.
  4. System prompts for confirmation.
  5. Tenant confirms the deletion.
  6. System deletes the site and all associated rooms and AC devices.
  7. System updates the list of sites.
- **Alternate Flows:**
  a. **Deletion fails due to system error:** System shows error message.

Use Case 5: Add Room



- **Goal in Context:** Allow a tenant to add a new room within a selected site
- **Primary Actor:** Tenant
- **Supporting Actors:** None
- **Preconditions:**
  a. Tenant is logged in.
  b. At least one site exists and is selected.
- **Postconditions:** A new room is added under the selected site and is available for adding ACs.
- **Main Success Scenario:**
  1. Tenant navigates to the AC Control screen.
  2. Tenant chooses a site on the list that he wishes to add room to.
  3. System displays a list of rooms and the current site context.
  4. Tenant clicks “Add Room” and enters room details (e.g., room name).
  5. System validates the input.
  6. System saves the new room under the selected site.
  7. System updates the room list and displays confirmation.
- **Alternate Flows:**
  a. **Room name is missing:** System requests correction.
  b. **Save operation fails:** System shows failure message.

Use Case 6: Update Room



Goal in Context:
Allow a tenant to update the details of an existing room within a selected site (e.g., rename the room).

Primary Actor: Tenant
Supporting Actors: None

Preconditions:

- Tenant is logged in.
- At least one site exists and is selected.
- At least one room exists under the selected site.

Postconditions:

- The selected room's details are updated in the system.
- Any linked air conditioners remain associated with the updated room.

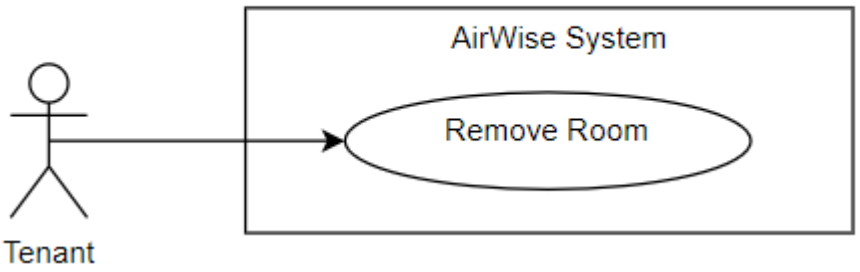
Main Success Scenario:

1. Tenant navigates to the AC Control screen.
2. System displays a list of sites
3. Tenant chooses a site
4. System displays a list of rooms and the selected site context.
5. Tenant selects a room to update.
6. System displays the current room details.
7. Tenant edits one or more fields (e.g., room name).
8. System validates the updated input.
9. System saves the updated room details.
10. System updates the room list and displays confirmation.

Alternate Flows:

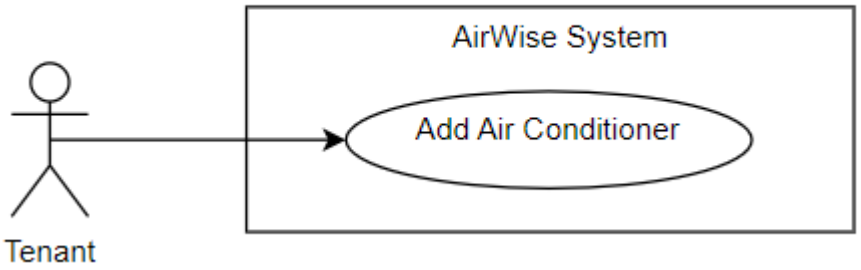
- Room name is missing: System prompts tenant to correct the input.
- Update operation fails: displays a failure message to the tenant.

Use Case 6: Remove Room



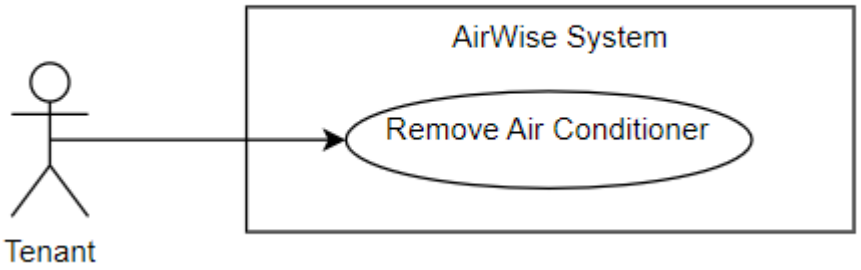
- **Goal in Context:** Allow a tenant to delete a room from a specific site, including its associated air conditioners.
- **Primary Actor:** Tenant
- **Supporting Actors:** None
- **Preconditions:**
  a. Tenant is logged in.
  b. A site with at least one room is selected.
- **Postconditions:** The selected room and its ACs are deleted from the system.
- **Main Success Scenario:**
  1. Tenant navigates to the AC Control screen.
  2. System displays a list of sites
  3. Tenant chooses a site
  4. System displays a list of rooms.
  5. Tenant selects a room to remove.
  6. System prompts for confirmation.
  7. Tenant confirms the deletion.
  8. System deletes the room and all associated air conditioners.
  9. System updates the list of rooms and displays a confirmation message.
- **Alternate Flows:**
  a. **Deletion fails due to system error:** displays a failure message to the tenant.

Use Case 7: Add Air Conditioner



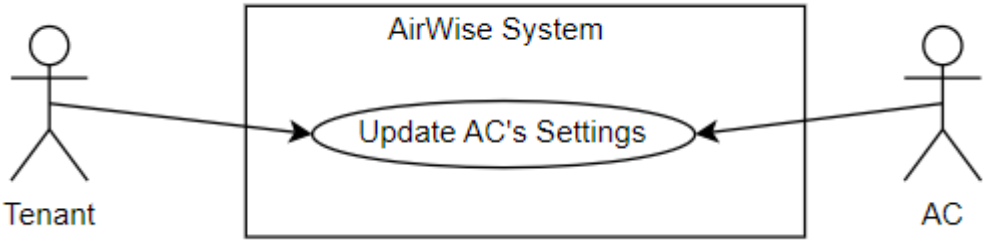
- **Goal in Context:** Allow a tenant to register a new air conditioner within a specific room.
- **Primary Actor:** Tenant
- **Supporting Actors:** None
- **Preconditions:**
  a. Tenant is logged in.
  b. At least one site and one room exist.
- **Postconditions:** The new air conditioner is stored and linked to the selected room.
- **Main Success Scenario:**
  1. Tenant navigates to the AC Control screen.
  2. System displays a list of sites.
  3. Tenant selects a site.
  4. System displays a list of rooms.
  5. Tenant selects a room
  6. Tenant enters details for the new AC.
  7. System validates the input and checks for duplicates.
  8. System registers the new AC and links it to the room.
  9. System updates the list of devices and displays confirmation.
- **Alternate Flows:**
  a. **Duplicate device serial:** System Aborts and displays a relevant message.
  b. **system error:** System shows error message.

Use Case 8: Remove Air Conditioner



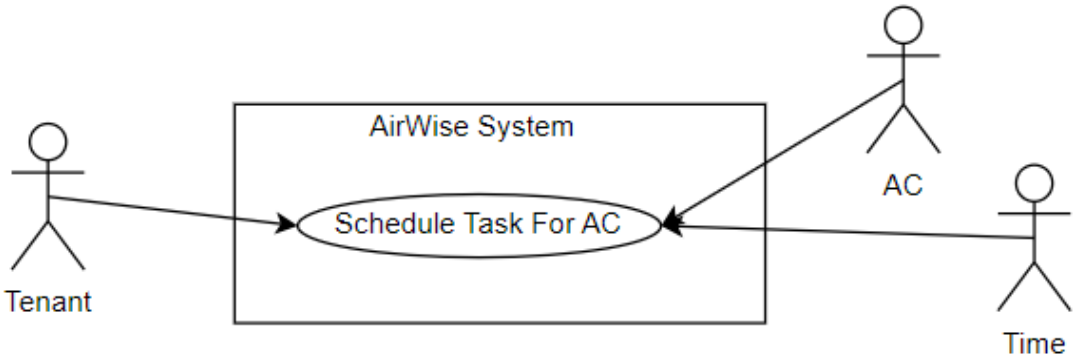
- **Goal in Context:** Allow a tenant to delete an air conditioner from a specific room.
- **Primary Actor:** Tenant
- **Supporting Actors:** None
- **Preconditions:**
  a. Tenant is logged in.
  b. At least one site and room with AC units exist.
- **Postconditions:** The selected AC unit is removed from the system and no longer appears in the room's configuration.
- **Main Success Scenario:**
  1. Tenant navigates to the AC Control screen.
  2. System loads the list of sites
  3. Tenant chooses a site
  4. System loads a list of rooms
  5. Tenant selects a room
  6. System loads a list of acs
  7. Tenant chooses a specific air conditioner.
  8. System prompts for confirmation.
  9. Tenant confirms the deletion.
  10. System deletes the AC unit from the room.
  11. System updates the device list and displays confirmation.
- **Alternate Flows:**
  a. **Deletion fails due to system error:** System shows an error message.

Use Case 9: Update Air Conditioner Settings (incl. On/Off)



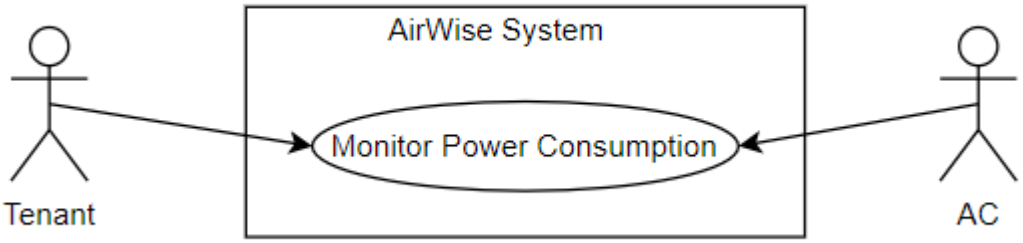
- **Goal in Context:** Allow a tenant to change the operational state and settings of an air conditioner, including turning it on/off, setting temperature, and changing mode.
- **Primary Actor:** Tenant
- **Supporting Actors:** Air Conditioner
- **Preconditions:**
  a. Tenant is logged in.
  b. At least one AC unit is available in a room associated with the tenant.
- **Postconditions:** The selected AC unit updates its settings and reflects the new status in the UI.
- **Main Success Scenario:**
  1. Tenant navigates to the AC Control screen.
  2. System loads a list of sites
  3. Tenant chooses a site
  4. System loads a list of rooms
  5. Tenant chooses a room
  6. System loads a list of ac units.
  7. Tenant selects a specific air conditioner.
  8. Tenant updates one or more settings.
  9. System sends updated configuration to the AC unit.
  10. System displays updated status and confirmation to the tenant.
- **Alternate Flows:**
  a. **AC fails to respond:** System shows error to the tenant.

Use Case 10: Schedule Task for Air Conditioner



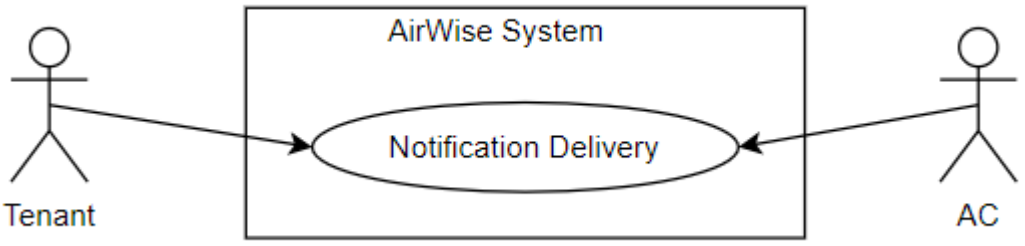
- **Goal in Context:** Allow a tenant to schedule a future task for an AC unit (e.g., turn on at a certain time, set temperature).
- **Primary Actor:** Tenant
- **Supporting Actors:** Air Conditioner, Time
- **Preconditions:**
  a. Tenant is logged in.
  b. At least one AC unit is available in a room associated with the tenant.
- **Postconditions:** A scheduled task is stored and will be triggered automatically at the defined time.
- **Main Success Scenario:**
  1. Tenant navigates to the AC Control screen.
  2. System loads a list of sites
  3. Tenant chooses a site
  4. System loads a list of rooms
  5. Tenant chooses a room
  6. System loads a list of ac units.
  7. Tenant selects an AC unit and defines task settings.
  8. System validates the task and timing.
  9. System stores the task in the database.
  10. System displays confirmation to the tenant.
  11. At the scheduled time, the Time actor triggers the task.
  12. System sends the command to the AC unit.
  13. AC unit executes the task and responds.
  14. System creates notification of the task status and sends email to the user.
- **Alternate Flows:**
  a. **AC does not respond:** System shows message to the tenant.
  b. **Task fails:** System logs failure.

Use Case 11: Monitor Power Consumption



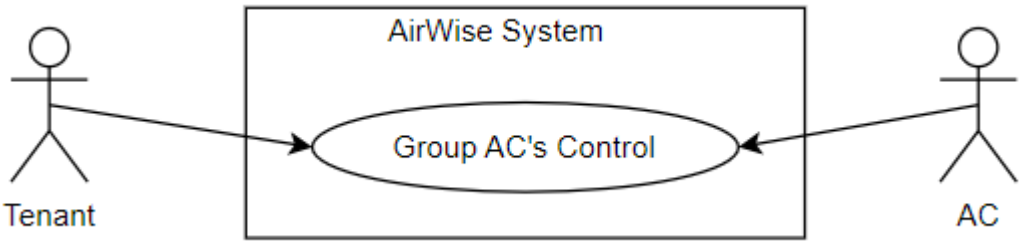
- **Goal in Context:** Allow a tenant to view and analyze power consumption statistics of their air conditioners.
- **Primary Actor:** Tenant
- **Supporting Actors:** AC
- **Preconditions:**
  a. Tenant is logged in.
  b. Power consumption data exists for at least one AC unit.
- **Postconditions:** The tenant sees detailed power consumption statistics in a chart of energy usage.
- **Main Success Scenario:**
  1. Tenant navigates to the Home screen.
  2. System loads data of energy usage per site.
  3. System presents the data as a chart.
  4. Tenant reviews the information to gain insight about the energy consumption.
- **Alternate Flows:**
  a. **No data exists:** System shows notice of missing data.

Use Case 12: Notification Delivery



- **Goal in Context:** Ensure tenants are promptly informed of important system events.
- **Primary Actor:** Tenant
- **Supporting Actors:** Air Conditioner
- **Preconditions:**
  a. Tenant registered to the system.
- **Postconditions:** A relevant notification is sent and presented to the tenant.
- **Main Success Scenario:**
  1. An event occurs.
  2. Air Conditioner reports the event to the system.
  3. System creates a notification with relevant details and then sends email to the user.
  4. The tenant logs in and views the notification in the home screen.
- **Alternate Flows:**
  a. **notification creation error:** System logs the error.
  b. **notification is not visible**: Tenant refreshes the page

Use Case 13: Group Air Conditioner Control



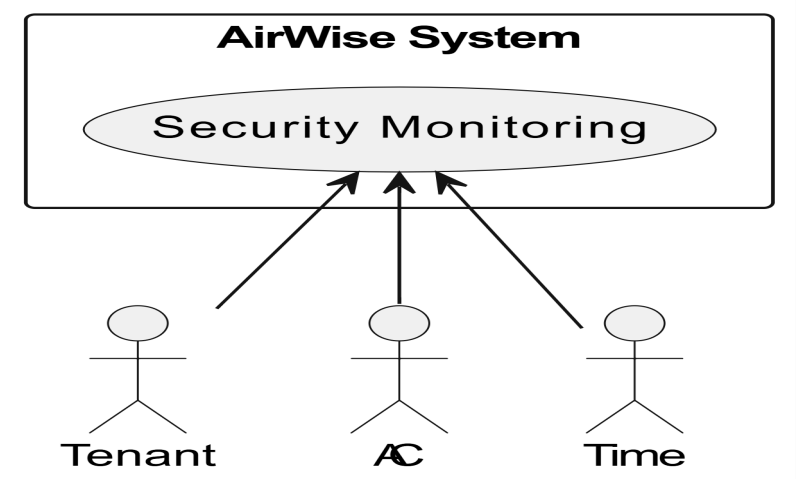
- **Goal in Context:** Enable tenants to control multiple air conditioners simultaneously within a selected room.
- **Primary Actor:** Tenant
- **Supporting Actors:**Air Conditioner
- **Preconditions:**
  a. Tenant is logged in.
  b. At least one site that has at least one room with at least one AC unit is assigned to the tenant.
- **Postconditions:** The selected action is applied to all relevant AC units, and the status of the operation is reported to the tenant.
- **Main Success Scenario:**

1. Tenant navigates to the ac control screen.
2. System loads list of sites
3. Tenant chooses a site
4. System loads a list of rooms
5. Tenant chooses a room and an operation to be executed
6. System iterates over all AC units and sends a command to each AC.
7. Each Air Conditioner responds with execution status.
8. System updates the current status of all the acs in the chosen room.

- **Alternate Flows:**

a. **No AC units available in selected groups:** System displays a relevant message.

Use Case 14: Security monitor for site



- **Goal in Context:** Enabling tenants to get security alert notifications.
- **Primary Actor:** Tenant
- **Supporting Actors:**Air Conditioner, Time
- **Preconditions:**
    a. Tenant is registered
    b. At least one site that has at least one room with at least one AC unit is assigned to the tenant
- **Postconditions:** The Tenant will receive a security alert notification.
- **Main Success Scenario:**
    1. Tenant logging on into the system, and navigates to the AC Control screen.
    2. The System loads a list of sites
    3. The Tenant clicks on edit button at a particular site
    4. The System opens form for editing sites properties
    5. The Tenant chooses that he is not present in the site
    6. The System checks every minute if there is a motion at the ”no tenant present” site
    7. If there is a motion in the site, then the tenant will receive a security alert notification by Email (one security alert every half hour).
- **Alternate Flows:**

a. **No AC units available in site:** The System won’t detect motion at the site.

4. Non-Functional Requirements

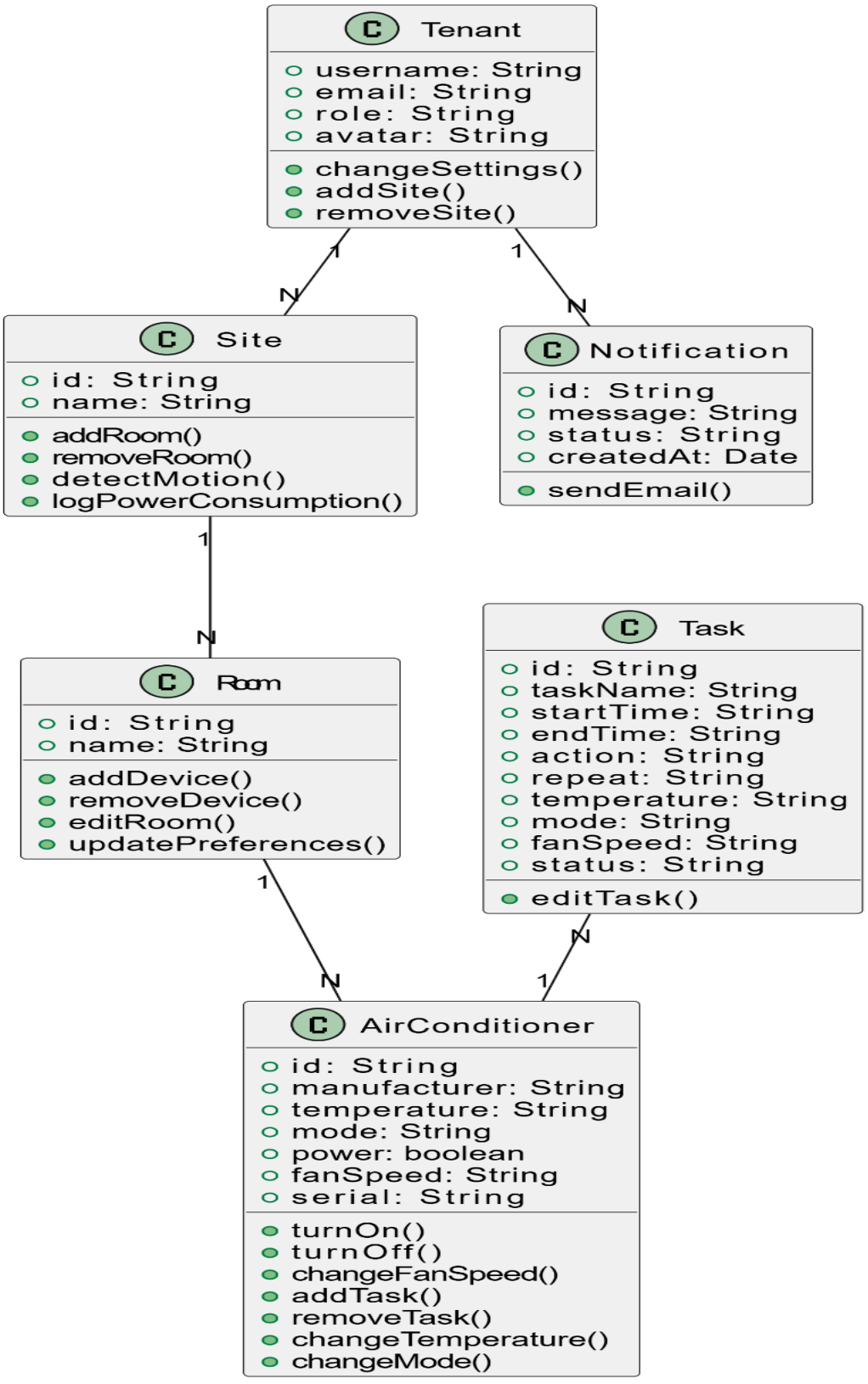
Requirement Type	Requirement Description
U - Usability	The system must support Dark and Light modes for accessibility and comfort.

5. APPENDENCIES

5.1 Objects and Commands in our project:

Object	Purpose	Attributes	Commands
Tenant	Managing tenant profiles and associated devices	email, username, avatar, role	changeSettings, addSite, removeSite
AirConditioner	Managing individual AC units	id, serial, manufacturer, temperature, power, mode, fanSpeed	turnOn, turnOff, changeTemperature, changeMode, changeFanSpeed, addTask, removeTask
Site	Managing rooms, and tracking the total of energy consumption	id, name	addRoom, removeRoom, detectMotion, logPowerConsumption
Room	Managing ACs	id, name, temperature, mode, fanSpeed	addDevice (AC), removeDevice (AC), editRoom, updatePreferences
Notification	User notifications	id, message, status, createdAt	sendEmail
Task	Represents task operation for ac unit	id, taskName, action, startTime, endTime, repeat, temperature, mode, fanSpeed, status	editTask

5.2 Class Diagram:



2. Technologies

Below is a list of the main technologies and tools used throughout the development of our project:

Backend (Main server):

- **Spring Boot (Java)** - Core backend framework, responsible for handling API logic, validation, and integration with external services.
- **MongoDB** - NoSQL database used for data persistence.

Node.js Service (Secondary Server):

- **Node.js + Express** - A lightweight service acting as an AC demo API provider, which the Spring Boot server communicates with.
- **MongoDB** - NoSQL database used for data persistence.

Frontend (Client Side):

- **ReactJS** - For building a dynamic and responsive user interface.
- **Axios** - For handling HTTP requests to the backend server.
- **JavaScript** - Main programming language for frontend logic.
- **Material UI** - For UI components and layout.
- **Recharts** - A charting library used for displaying visual data in the UI.

Testing:

- **JUnit** - Used for writing and executing unit tests in the Spring Boot backend.

Version Control & Collaboration:

- **Git** - For version control and team collaboration.
- **SourceTree** - GUI tool for managing Git workflows.
- **Bitbucket** - Repository hosting for source code and pull request management.
- **Microsoft Teams** – For team communication, coordination, and sharing updates throughout the development process.

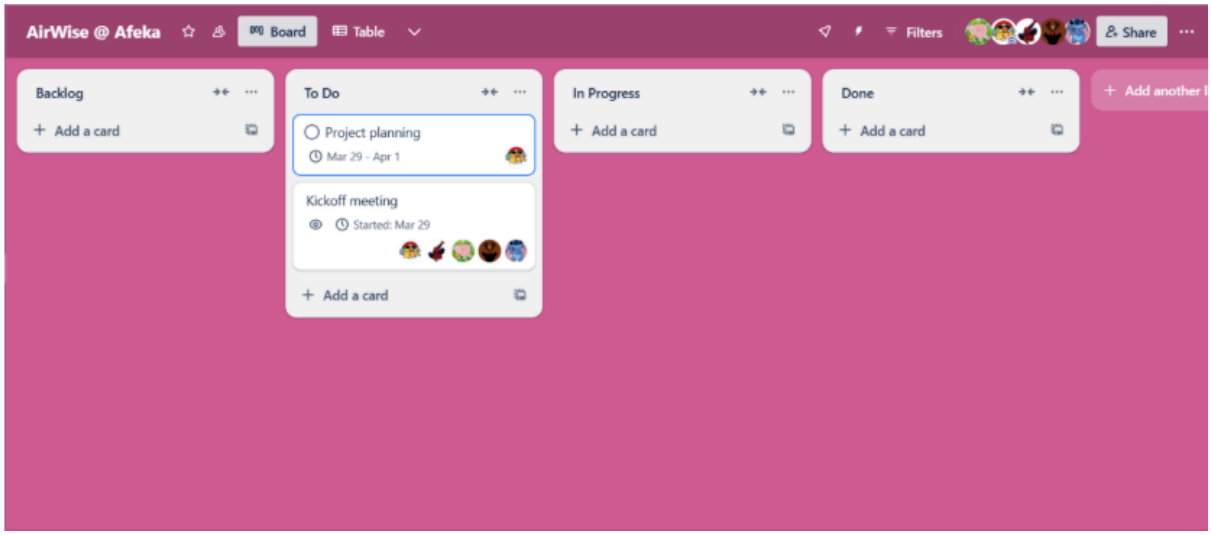
Development Tools:

- **Visual Studio Code** - Main editor used for frontend and Node.js development.
- **SprintToolSuite4** - used for backend (Spring Boot) development.
- **Swagger** - Tool for testing and debugging HTTP APIs.
- **Docker CLI & Docker Desktop** - Tools for building, managing, and running containers locally.
- **Docker Compose** - Used to manage and run multiple services like the Spring Boot server and MongoDB together in a unified environment.

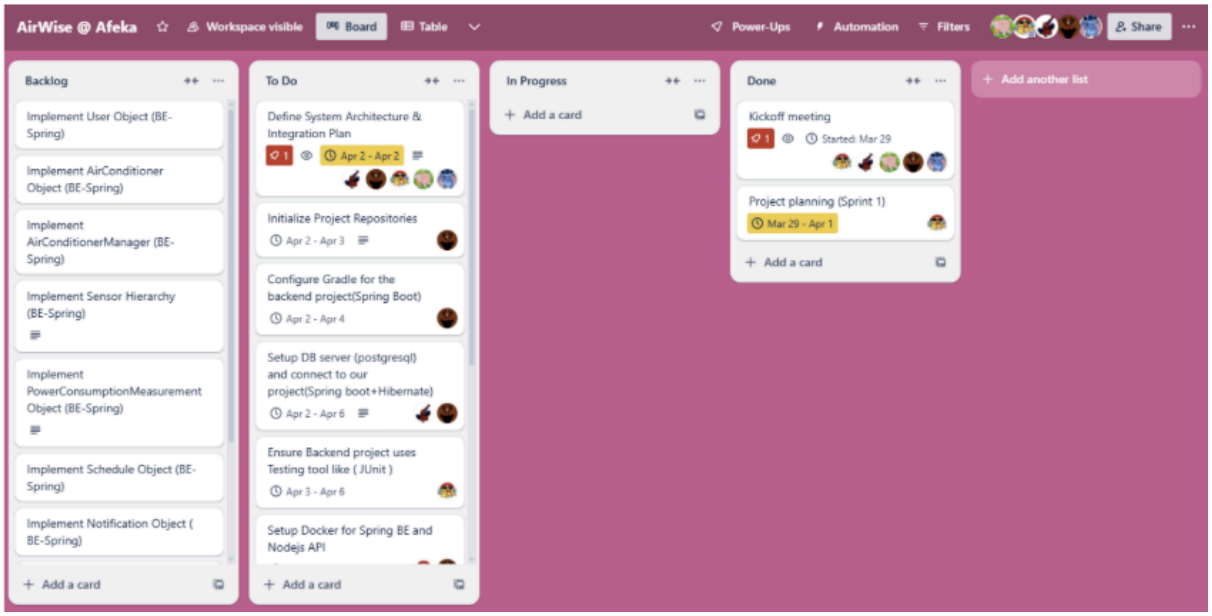
3. Project General Summary

3.1. Kanban samples (ascending):

sprint 1: project initiation
sample date: 19.3.2025

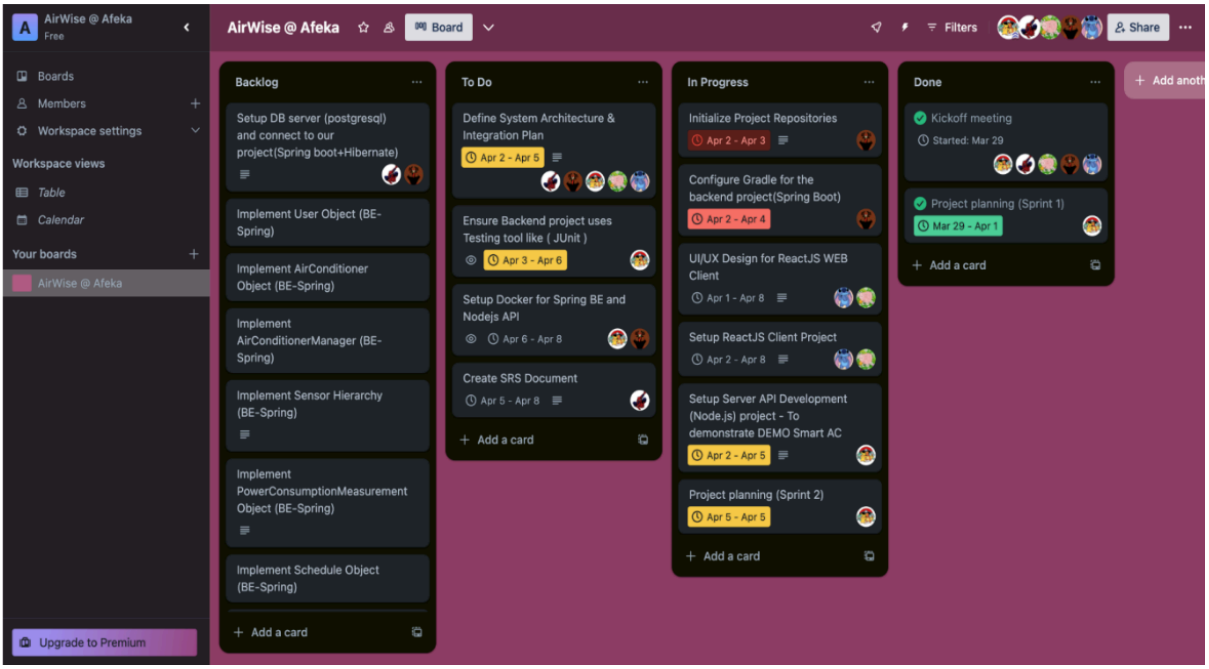


sample date: 1.4.2025

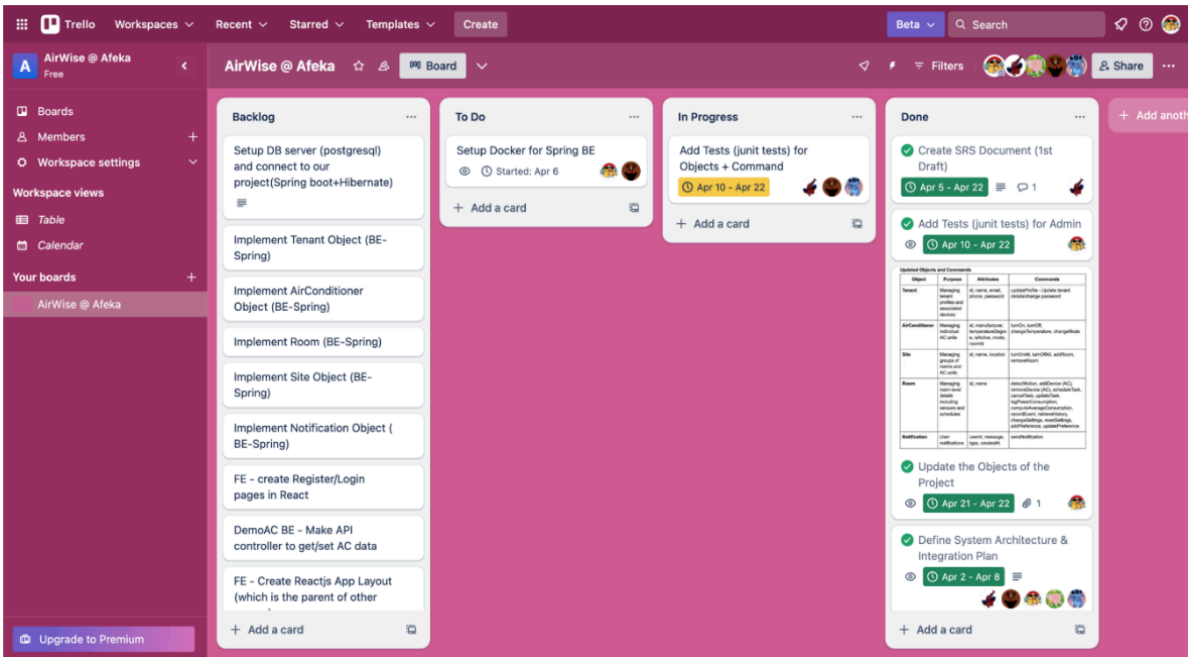


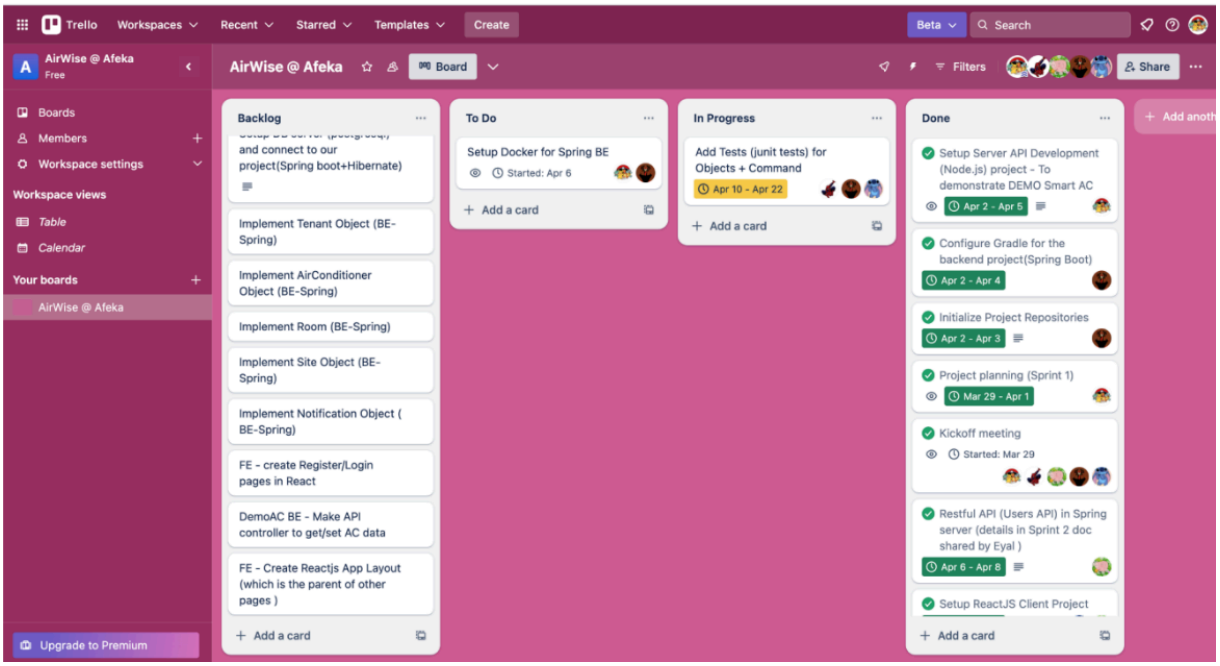
sprint 2:

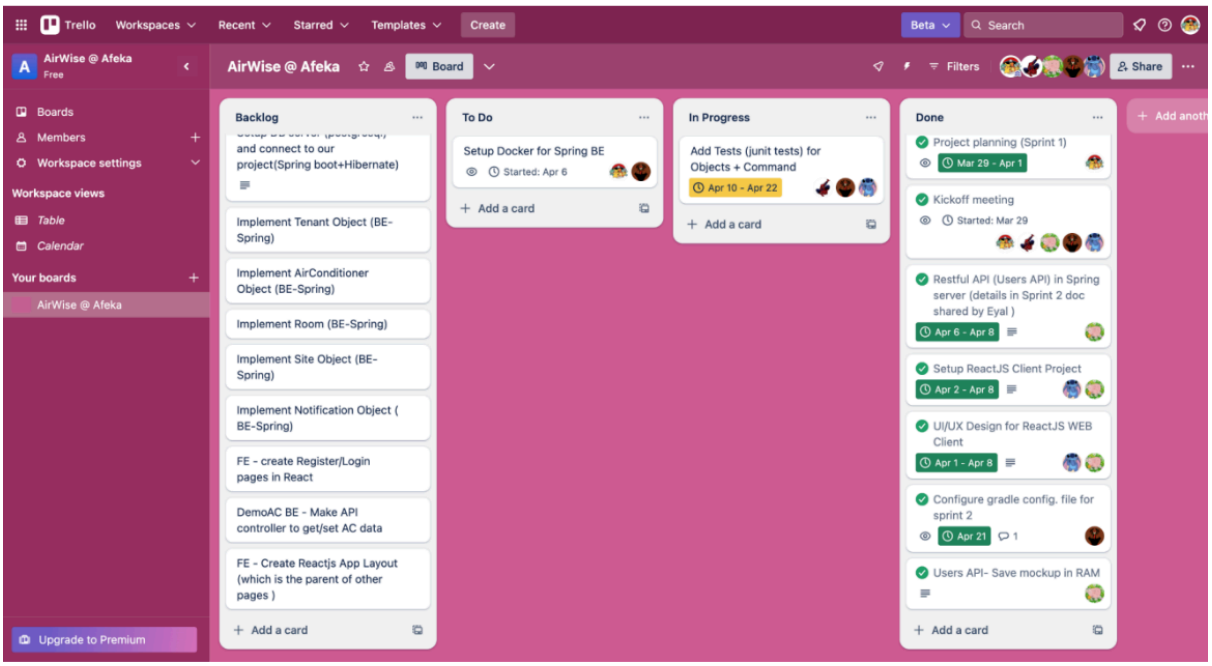
sample date: 5.4.2025



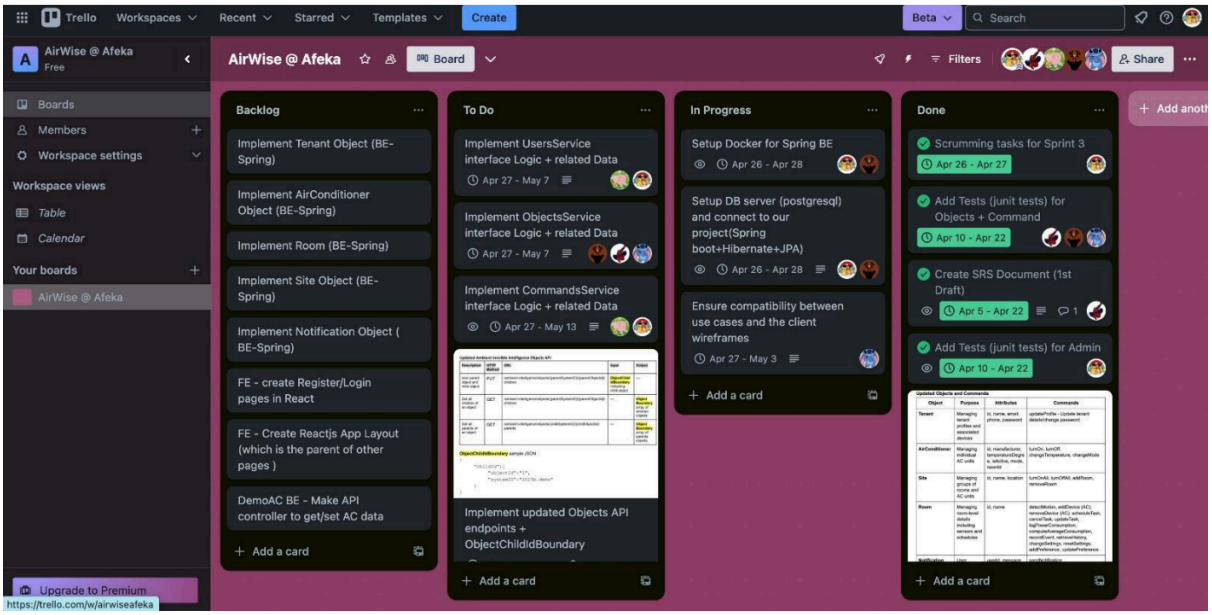
sample date: 22.4.2025

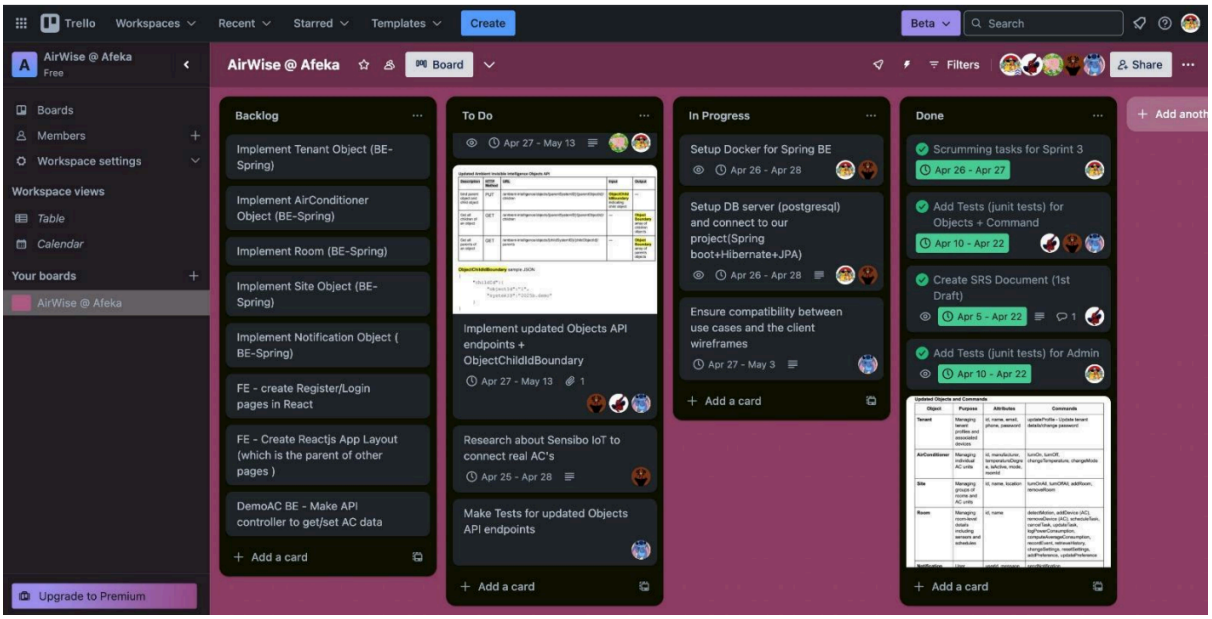



Trello
Workspaces
Recent
Starred
Templates
Create
Beta
Search
AirWise @ Afeka
Free
AirWise @ Afeka
Board
Filters
Share
Boards
Members
Workspace settings
Workspace views
Table
Calendar
Your boards
AirWise @ Afeka
Upgrade to Premium
Backlog
and connect to our project(Spring boot+Hibernate)
Implement Tenant Object (BE-Spring)
Implement AirConditioner Object (BE-Spring)
Implement Room (BE-Spring)
Implement Site Object (BE-Spring)
Implement Notification Object (BE-Spring)
FE - create Register/Login pages in React
DemoAC BE - Make API controller to get/set AC data
FE - Create Reactjs App Layout (which is the parent of other pages)
Add a card
To Do
Setup Docker for Spring BE
Started: Apr 6
Add a card
In Progress
Add Tests (junit tests) for Objects + Command
Apr 10 - Apr 22
Add a card
Done
Setup Server API Development (Node.js) project - To demonstrate DEMO Smart AC
Apr 2 - Apr 5
Configure Gradle for the backend project(Spring Boot)
Apr 2 - Apr 4
Initialize Project Repositories
Apr 2 - Apr 3
Project planning (Sprint 1)
Mar 29 - Apr 1
Kickoff meeting
Started: Mar 29
Restful API (Users API) in Spring server (details in Sprint 2 doc shared by Eyal)
Apr 6 - Apr 8
Setup ReactJS Client Project
Add a card

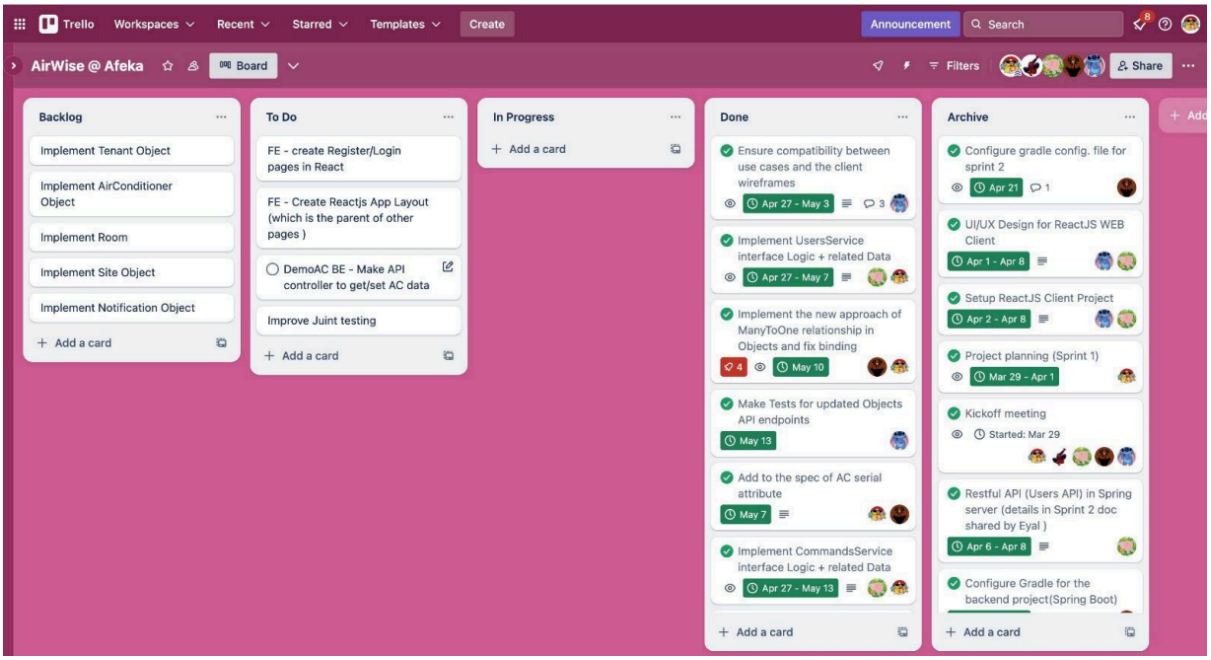

Trello
Workspaces
Recent
Starred
Templates
Create
Beta
Search
AirWise @ Afeka
Free
AirWise @ Afeka
Board
Filters
Share
Boards
Members
Workspace settings
Workspace views
Table
Calendar
Your boards
AirWise @ Afeka
Upgrade to Premium
Backlog
and connect to our project(Spring boot+Hibernate)
Implement Tenant Object (BE-Spring)
Implement AirConditioner Object (BE-Spring)
Implement Room (BE-Spring)
Implement Site Object (BE-Spring)
Implement Notification Object (BE-Spring)
FE - create Register/Login pages in React
DemoAC BE - Make API controller to get/set AC data
FE - Create Reactjs App Layout (which is the parent of other pages)
Add a card
To Do
Setup Docker for Spring BE
Started: Apr 6
Add a card
In Progress
Add Tests (junit tests) for Objects + Command
Apr 10 - Apr 22
Add a card
Done
Project planning (Sprint 1)
Mar 29 - Apr 1
Kickoff meeting
Started: Mar 29
Restful API (Users API) in Spring server (details in Sprint 2 doc shared by Eyal)
Apr 6 - Apr 8
Setup ReactJS Client Project
Apr 2 - Apr 8
UI/UX Design for ReactJS WEB Client
Apr 1 - Apr 8
Configure gradle config. file for sprint 2
Apr 21
1
Users API- Save mockup in RAM
Add a card

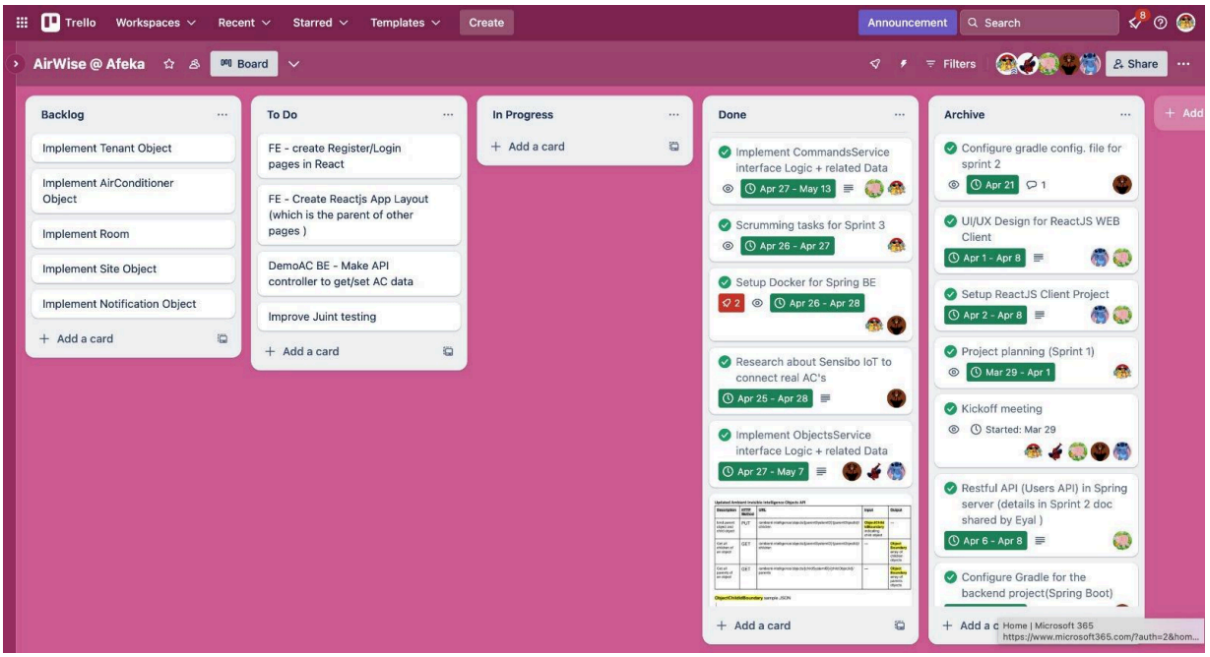
sprint 3:
sample date: 26.4.2025

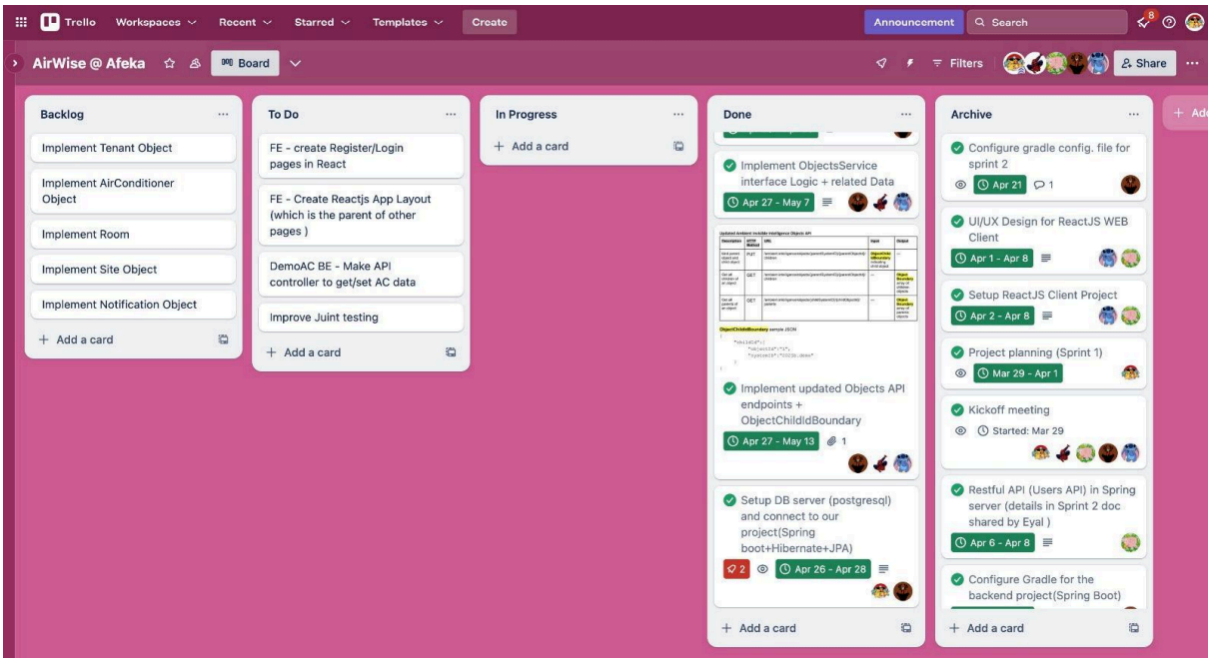




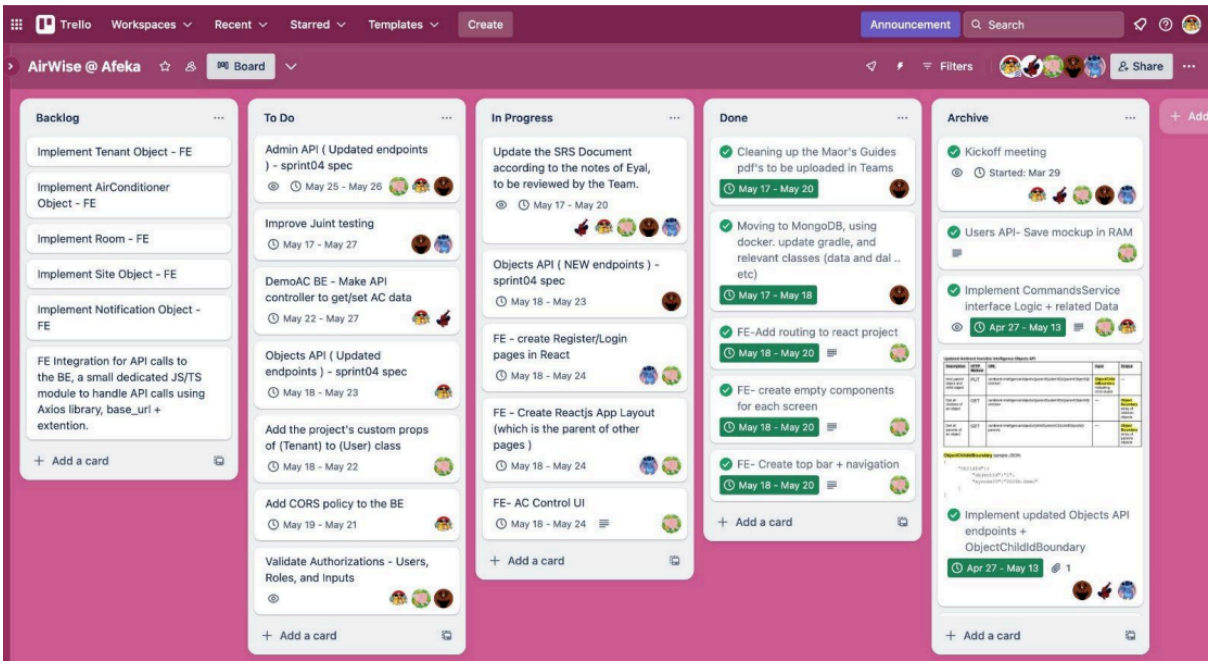
sample date: 13.5.2025



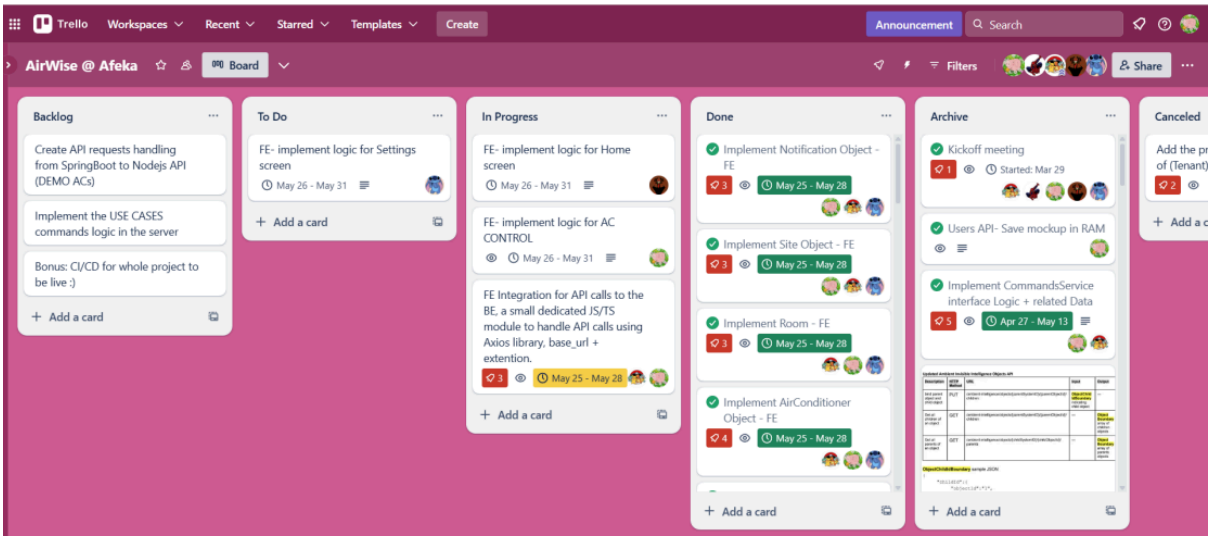


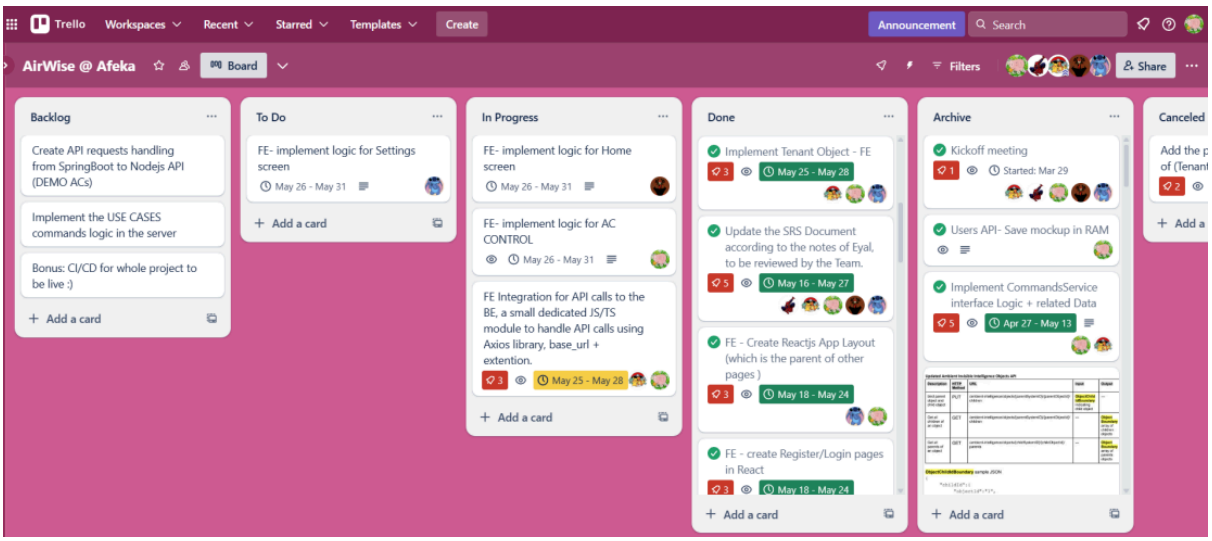

Trello
Workspaces
Recent
Starred
Templates
Create
Announcement
Search
AirWise @ Afeka
Board
Filters
Share
Backlog
Implement Tenant Object
Implement AirConditioner Object
Implement Room
Implement Site Object
Implement Notification Object
Add a card
To Do
FE - create Register/Login pages in React
FE - Create Reactjs App Layout (which is the parent of other pages)
DemoAC BE - Make API controller to get/set AC data
Improve Juint testing
Add a card
In Progress
Add a card
Done
Implement ObjectsService interface Logic + related Data
Apr 27 - May 7
Implement updated Objects API endpoints + ObjectChildIdBoundary
Apr 27 - May 13
Setup DB server (postgresql) and connect to our project(Spring boot+Hibernate+JPA)
Apr 26 - Apr 28
Add a card
Archive
Configure gradle config. file for sprint 2
Apr 21
UI/UX Design for ReactJS WEB Client
Apr 1 - Apr 8
Setup ReactJS Client Project
Apr 2 - Apr 8
Project planning (Sprint 1)
Mar 29 - Apr 1
Kickoff meeting
Started: Mar 29
Restful API (Users API) in Spring server (details in Sprint 2 doc shared by Eyal)
Apr 6 - Apr 8
Configure Gradle for the backend project(Spring Boot)
Add a card

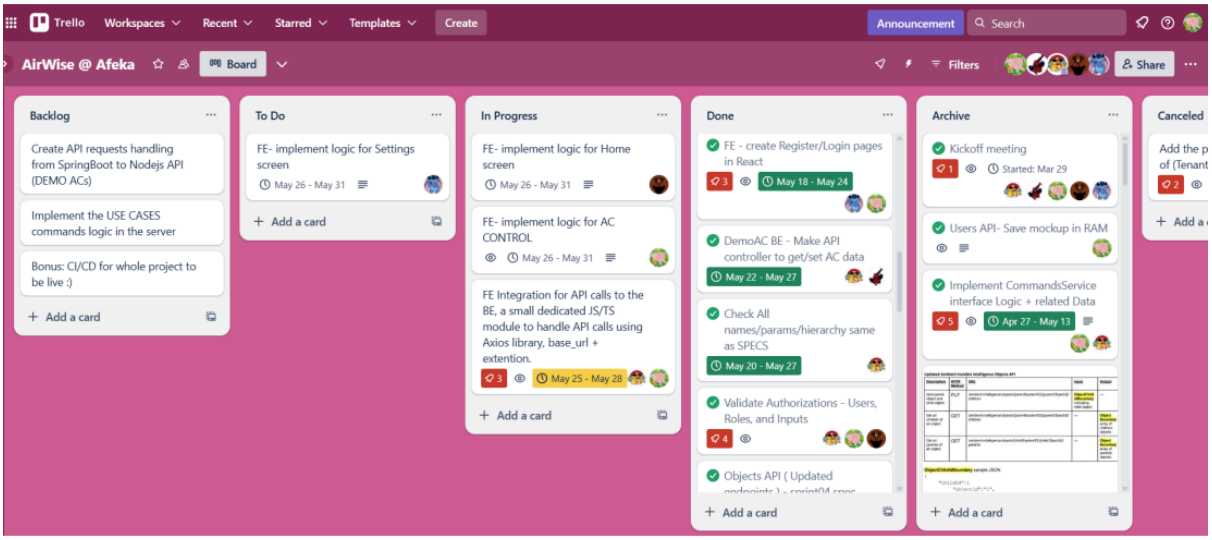
sprint 4:
sample date: 19.5.2025

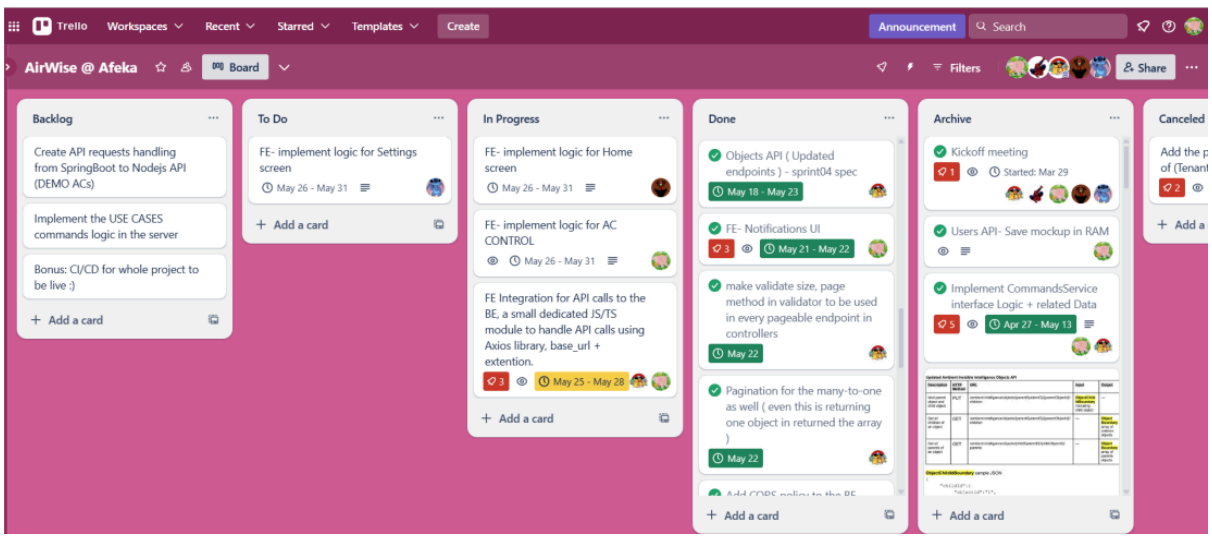


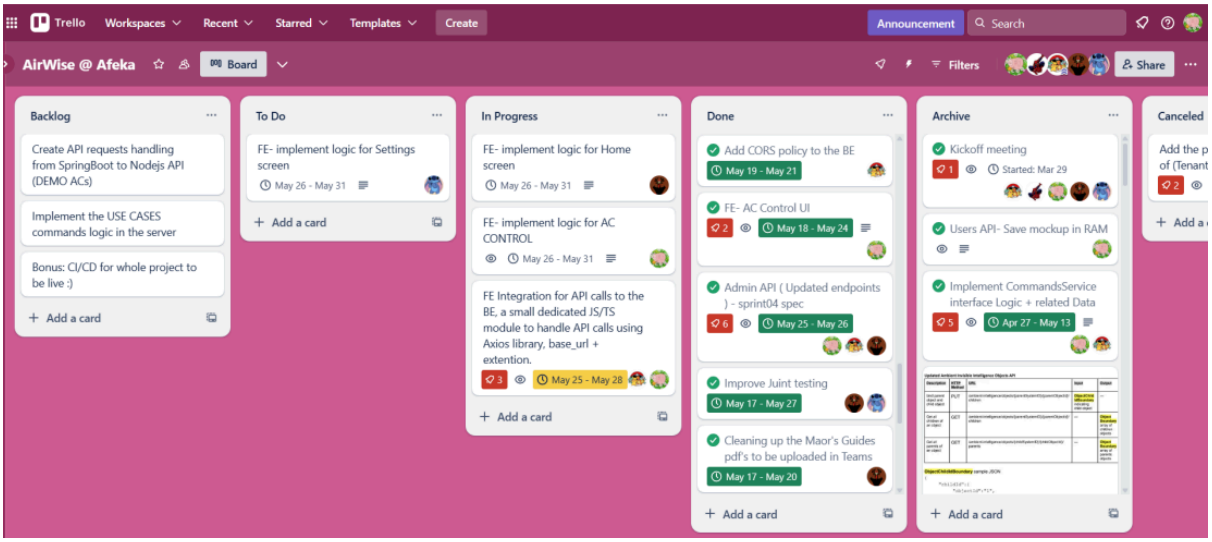
27.5.2025

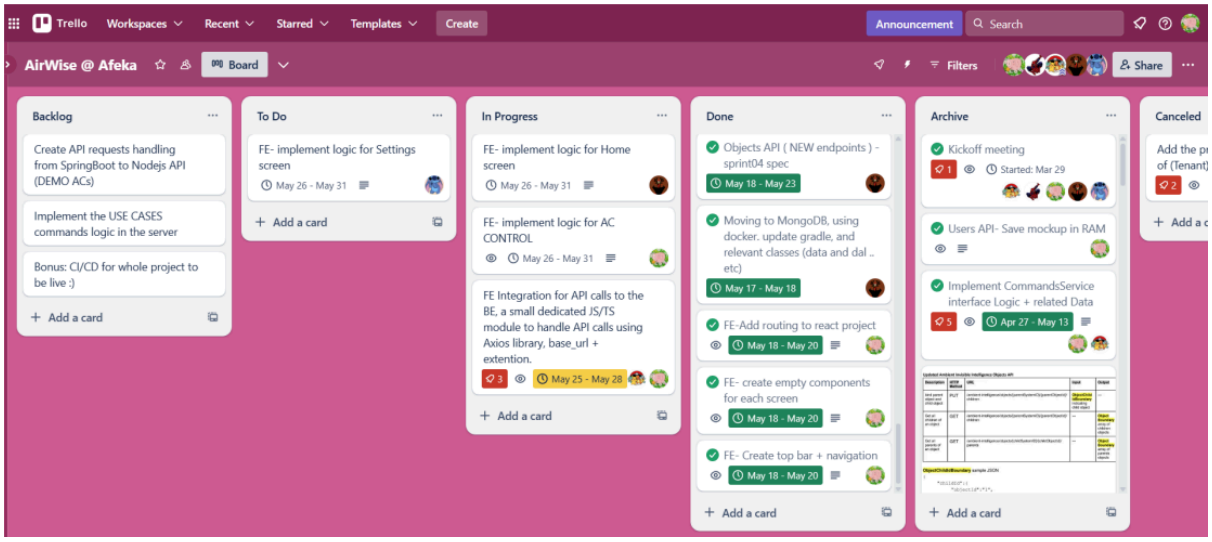


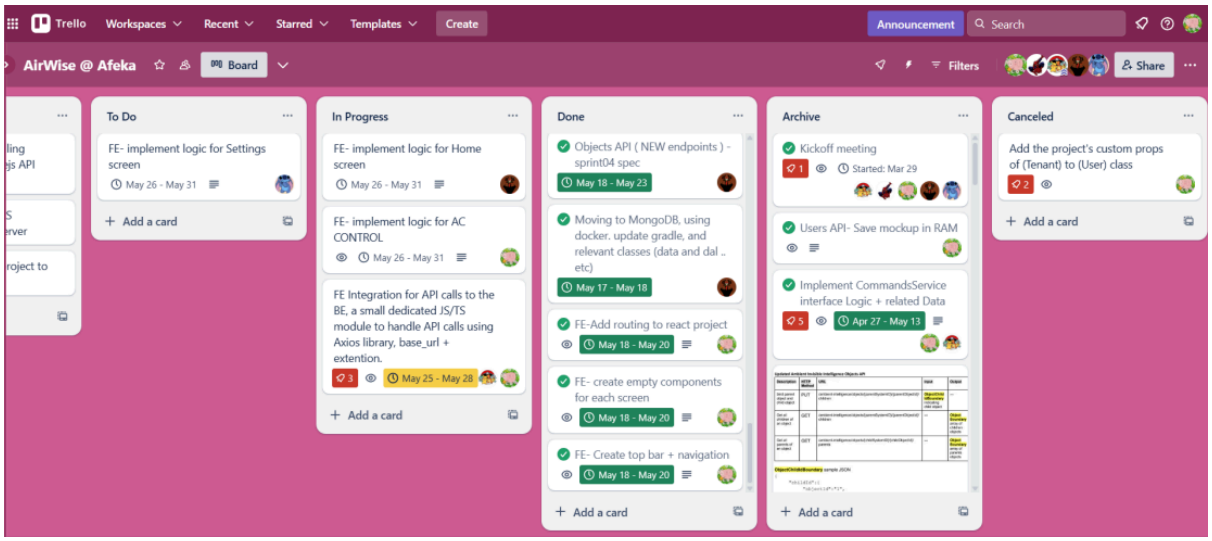

Trello
Workspaces
Recent
Starred
Templates
Create
Announcement
Search
AirWise @ Afeka
Board
Filters
Share
Backlog
Create API requests handling from SpringBoot to Nodejs API (DEMO ACs)
Implement the USE CASES commands logic in the server
Bonus: CI/CD for whole project to be live :)
Add a card
To Do
FE- implement logic for Settings screen
May 26 - May 31
Add a card
In Progress
FE- implement logic for Home screen
May 26 - May 31
FE- implement logic for AC CONTROL
May 26 - May 31
FE Integration for API calls to the BE, a small dedicated JS/TS module to handle API calls using Axios library, base_url + extention.
May 25 - May 28
Add a card
Done
Implement Tenant Object - FE
May 25 - May 28
Update the SRS Document according to the notes of Eyal, to be reviewed by the Team.
May 16 - May 27
FE - Create Reactjs App Layout (which is the parent of other pages)
May 18 - May 24
FE - create Register/Login pages in React
May 18 - May 24
Add a card
Archive
Kickoff meeting
Started: Mar 29
Users API- Save mockup in RAM
Implement CommandsService interface Logic + related Data
Apr 27 - May 13
Add a card
Canceled
Add a card


Trello
Workspaces
Recent
Starred
Templates
Create
Announcement
Search
AirWise @ Afeka
Board
Filters
Share
Backlog
Create API requests handling from SpringBoot to Nodejs API (DEMO ACs)
Implement the USE CASES commands logic in the server
Bonus: CI/CD for whole project to be live :)
Add a card
To Do
FE- implement logic for Settings screen
May 26 - May 31
Add a card
In Progress
FE- implement logic for Home screen
May 26 - May 31
FE- implement logic for AC CONTROL
May 26 - May 31
FE Integration for API calls to the BE, a small dedicated JS/TS module to handle API calls using Axios library, base_url + extention.
May 25 - May 28
Add a card
Done
FE - create Register/Login pages in React
May 18 - May 24
DemoAC BE - Make API controller to get/set AC data
May 22 - May 27
Check All names/params/hierarchy same as SPECS
May 20 - May 27
Validate Authorizations - Users, Roles, and Inputs
Objects API (Updated
Add a card
Archive
Kickoff meeting
Started: Mar 29
Users API- Save mockup in RAM
Implement CommandsService interface Logic + related Data
Apr 27 - May 13
Add a card
Canceled
Add a card

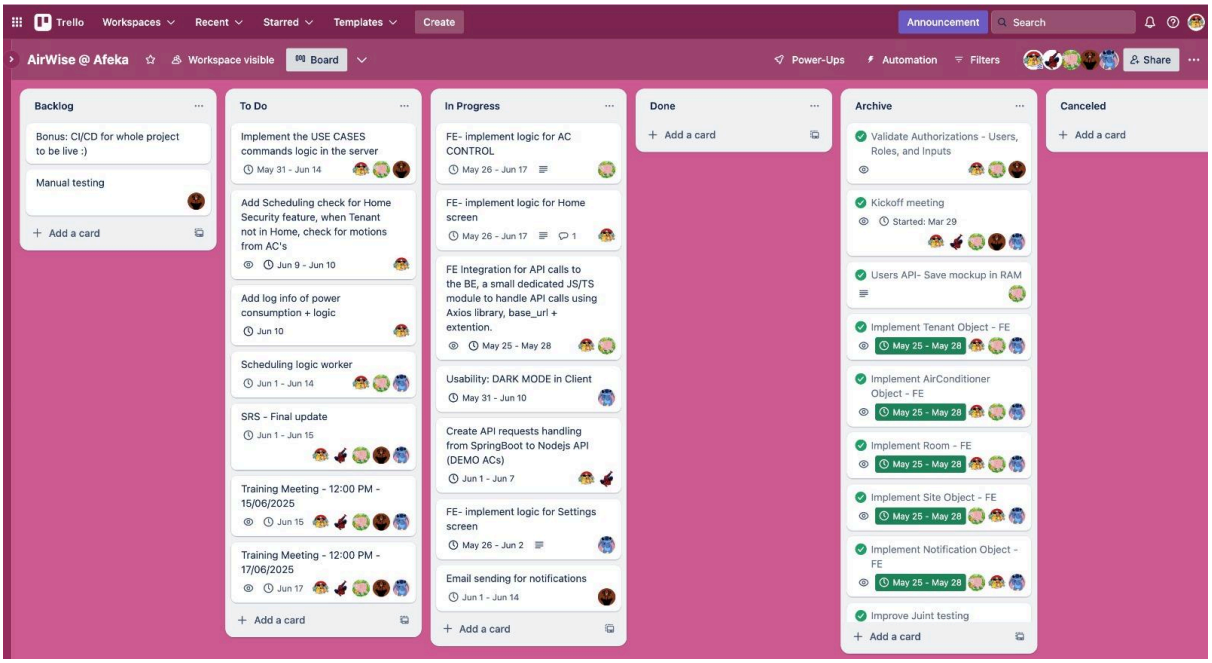

Trello
Workspaces
Recent
Starred
Templates
Create
Announcement
Search
AirWise @ Afeka
Board
Filters
Share
Backlog
Create API requests handling from SpringBoot to Nodejs API (DEMO ACs)
Implement the USE CASES commands logic in the server
Bonus: CI/CD for whole project to be live :)
Add a card
To Do
FE- implement logic for Settings screen
May 26 - May 31
Add a card
In Progress
FE- implement logic for Home screen
May 26 - May 31
FE- implement logic for AC CONTROL
May 26 - May 31
FE Integration for API calls to the BE, a small dedicated JS/TS module to handle API calls using Axios library, base_url + extention.
May 25 - May 28
Add a card
Done
Objects API (Updated endpoints) - sprint04 spec
May 18 - May 23
FE- Notifications UI
May 21 - May 22
make validate size, page method in validator to be used in every pageable endpoint in controllers
May 22
Pagination for the many-to-one as well (even this is returning one object in returned the array)
May 22
Add a card
Archive
Kickoff meeting
Started: Mar 29
Users API- Save mockup in RAM
Implement CommandsService interface Logic + related Data
Apr 27 - May 13
Add a card
Canceled
Add a card


Trello
Workspaces
Recent
Starred
Templates
Create
Announcement
Search
AirWise @ Afeka
Board
Filters
Share
Backlog
Create API requests handling from SpringBoot to Nodejs API (DEMO ACs)
Implement the USE CASES commands logic in the server
Bonus: CI/CD for whole project to be live :)
Add a card
To Do
FE- implement logic for Settings screen
May 26 - May 31
Add a card
In Progress
FE- implement logic for Home screen
May 26 - May 31
FE- implement logic for AC CONTROL
May 26 - May 31
FE Integration for API calls to the BE, a small dedicated JS/TS module to handle API calls using Axios library, base_url + extention.
May 25 - May 28
Add a card
Done
Add CORS policy to the BE
May 19 - May 21
FE- AC Control UI
May 18 - May 24
Admin API (Updated endpoints) - sprint04 spec
May 25 - May 26
Improve Juint testing
May 17 - May 27
Cleaning up the Maor's Guides pdf's to be uploaded in Teams
May 17 - May 20
Add a card
Archive
Kickoff meeting
Started: Mar 29
Users API- Save mockup in RAM
Implement CommandsService interface Logic + related Data
Apr 27 - May 13
Add a card
Canceled
Add the p
of (Tenant
Add a

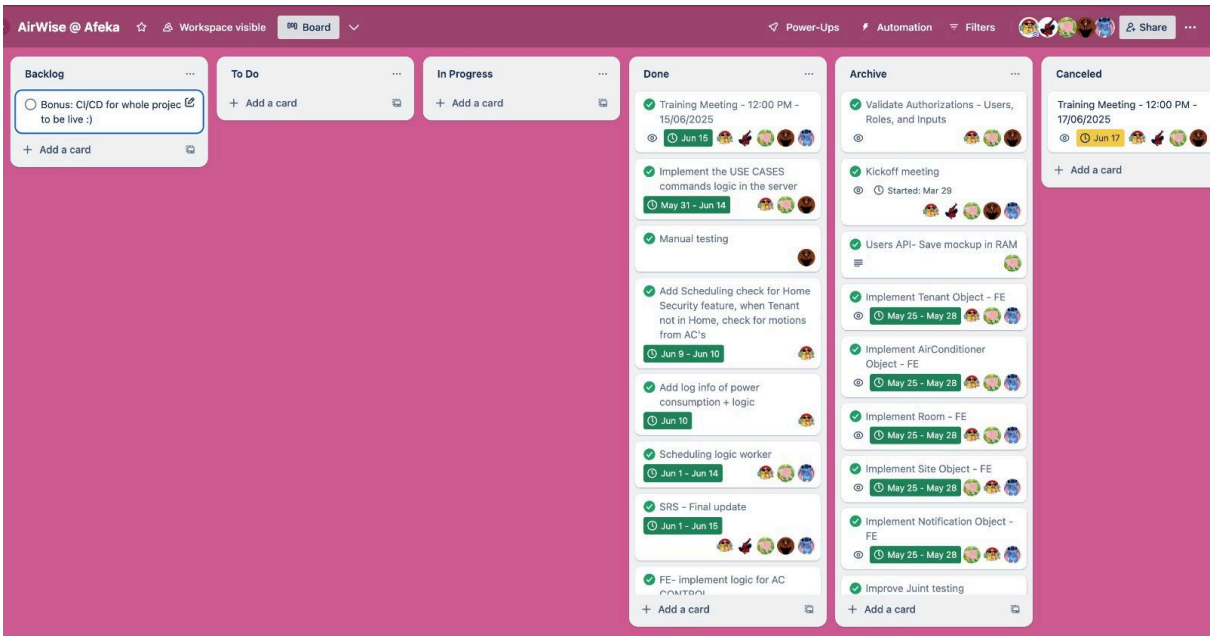

Trello
Workspaces
Recent
Starred
Templates
Create
Announcement
Search
AirWise @ Afeka
Board
Filters
Share
Backlog
Create API requests handling from SpringBoot to Nodejs API (DEMO ACs)
Implement the USE CASES commands logic in the server
Bonus: CI/CD for whole project to be live :)
Add a card
To Do
FE- implement logic for Settings screen
May 26 - May 31
Add a card
In Progress
FE- implement logic for Home screen
May 26 - May 31
FE- implement logic for AC CONTROL
May 26 - May 31
FE Integration for API calls to the BE, a small dedicated JS/TS module to handle API calls using Axios library, base_url + extention.
May 25 - May 28
Add a card
Done
Objects API (NEW endpoints) - sprint04 spec
May 18 - May 23
Moving to MongoDB, using docker. update gradle, and relevant classes (data and dal .. etc)
May 17 - May 18
FE-Add routing to react project
May 18 - May 20
FE- create empty components for each screen
May 18 - May 20
FE- Create top bar + navigation
May 18 - May 20
Add a card
Archive
Kickoff meeting
Started: Mar 29
Users API- Save mockup in RAM
Implement CommandsService interface Logic + related Data
Apr 27 - May 13
Add a card
Canceled
Add the pr
of (Tenant)
Add a

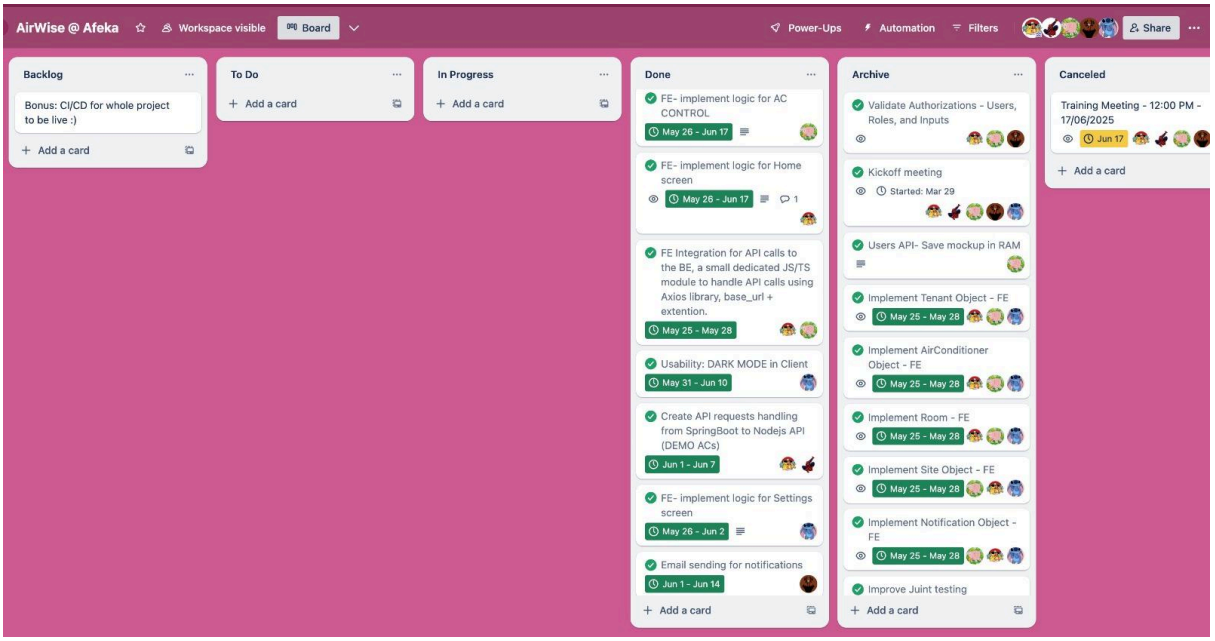

Trello
Workspaces
Recent
Starred
Templates
Create
Announcement
Search
AirWise @ Afeka
Board
Filters
Share
To Do
FE- implement logic for Settings screen
May 26 - May 31
Add a card
In Progress
FE- implement logic for Home screen
May 26 - May 31
FE- implement logic for AC CONTROL
May 26 - May 31
FE Integration for API calls to the BE, a small dedicated JS/TS module to handle API calls using Axios library, base_url + extention.
May 25 - May 28
Add a card
Done
Objects API (NEW endpoints) - sprint04 spec
May 18 - May 23
Moving to MongoDB, using docker. update gradle, and relevant classes (data and dal .. etc)
May 17 - May 18
FE-Add routing to react project
May 18 - May 20
FE- create empty components for each screen
May 18 - May 20
FE- Create top bar + navigation
May 18 - May 20
Add a card
Archive
Kickoff meeting
Started: Mar 29
Users API- Save mockup in RAM
Implement CommandsService interface Logic + related Data
Apr 27 - May 13
Add a card
Canceled
Add the project's custom props of (Tenant) to (User) class
Add a card

sprint 5 (final sprint):
sample date: 29.5.2025



sample date: 16.6.2025




AirWise @ Afeka
Workspace visible
Board
Power-Ups
Automation
Filters
Share
Backlog
Bonus: CI/CD for whole project to be live :)
Add a card
To Do
Add a card
In Progress
Add a card
Done
FE- implement logic for AC CONTROL
May 26 - Jun 17
FE- implement logic for Home screen
May 26 - Jun 17
FE Integration for API calls to the BE, a small dedicated JS/TS module to handle API calls using Axios library, base_url + extention.
May 25 - May 28
Usability: DARK MODE in Client
May 31 - Jun 10
Create API requests handling from SpringBoot to Nodejs API (DEMO ACs)
Jun 1 - Jun 7
FE- implement logic for Settings screen
May 26 - Jun 2
Email sending for notifications
Jun 1 - Jun 14
Add a card
Archive
Validate Authorizations - Users, Roles, and Inputs
Kickoff meeting
Started: Mar 29
Users API- Save mockup in RAM
Implement Tenant Object - FE
May 25 - May 28
Implement AirConditioner Object - FE
May 25 - May 28
Implement Room - FE
May 25 - May 28
Implement Site Object - FE
May 25 - May 28
Implement Notification Object - FE
May 25 - May 28
Improve Juint testing
Add a card
Canceled
Training Meeting - 12:00 PM - 17/06/2025
Jun 17
Add a card

3.2. Project summary:

- **What worked well for our team throughout the semester and should be maintained in future industry projects:**
  We maintained a consistent workflow using Git and pull requests, with regular code reviews that improved code quality and team-wide knowledge. Using SourceTree also helped reduce Git-related friction and ensured a smoother development process.
  In addition, open and effective communication between team members played a key role in keeping everyone aligned and resolving issues quickly. Team members also demonstrated strong self-learning abilities, often taking initiative to explore unfamiliar technologies and solve problems independently - skills that are crucial in any real-world development environment.

- **How we can improve our work in future projects:**
  In future projects, we would benefit from adopting testing earlier in the development cycle. This would improve reliability and allow us to catch issues before integration, especially in larger or multi-service systems. We also aim to improve shared documentation so that every team member can understand and modify any part of the system.

- **What we enjoyed the most while working on the project:**
  We enjoyed seeing how the different parts of the system came together - from building an interactive interface with React on the client side, to developing a backend with Spring Boot and MongoDB. Working on the server side was especially satisfying as we designed a flexible REST API that can contribute for future projects. Watching the full system operate smoothly as one was one of the most rewarding parts of the process.
  Additionally, it was fulfilling to gain practical, real-world knowledge and experience beyond theoretical concepts - bridging the gap between academic learning and real development work.

- **What we would do differently if starting the project now, after gaining knowledge and experience this semester:**
  We would focus more on early testing infrastructure and interface design. Setting up testing environments and writing reusable components earlier would have saved us time during the final integration phases.

- **Was teamwork also done remotely, in addition to face-to-face meetings?**
  Yes, our collaboration included both face-to-face meetings and remote work.

- **What tools did we use to collaborate remotely?**
  We used WhatsApp for daily communication, Teams for group calls, Trello and Google Docs for shared documentation and planning.

- **How did remote work affect the execution of our project?**
  Remote work allowed us to stay productive and it was especially useful when coordinating tasks asynchronously, although live collaboration (e.g., debugging together) was sometimes more effective in person.

4. Project Progress Report- final sprint (5)

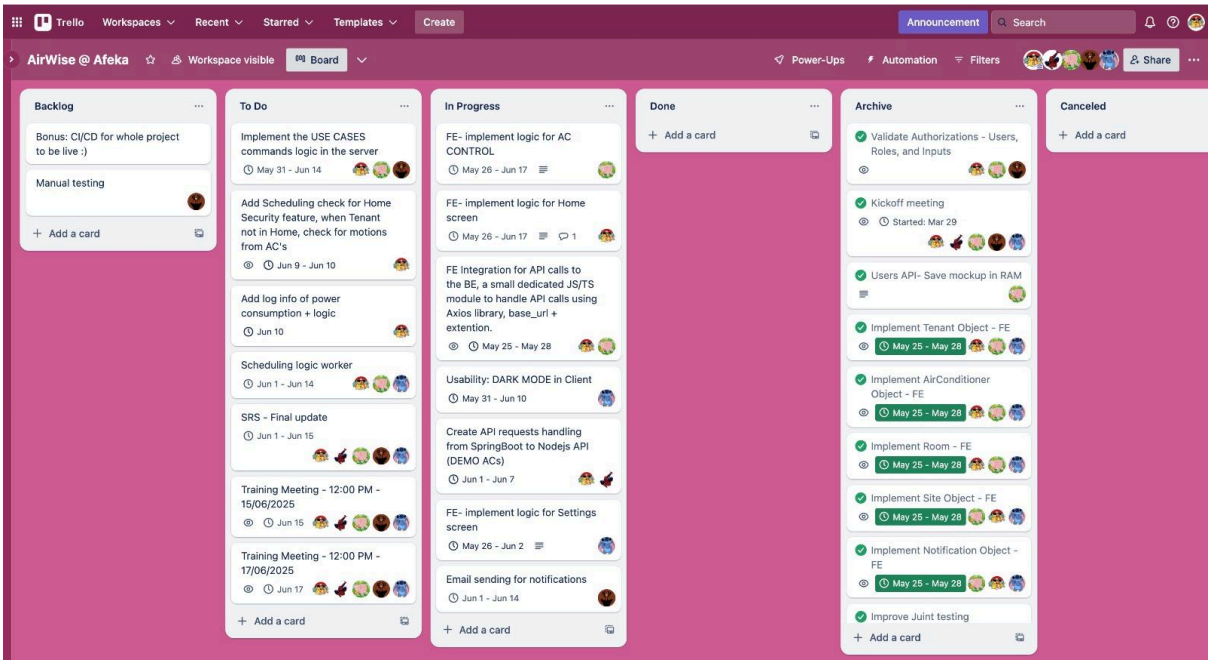
Course name: Integrative Engineering
Project name: Airwise
Document due date: 18.6.20254

4.1. Team members

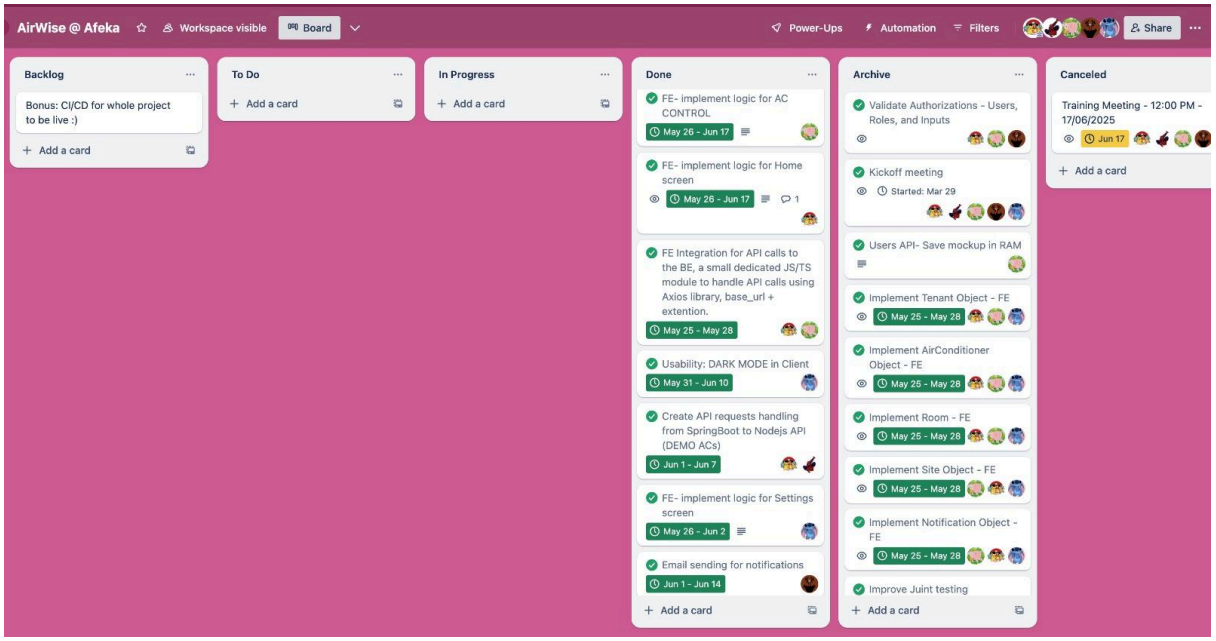
Student name	Student Id	Roles in team	Student Avatar in Kanban boards
Avital Iskhakov	207387481	Team Leader, Product Owner, Team	
Islam Saadi	203005772	Scrum Master, QA Engineer, Team	
Maor Mordo	203005772	Devops, System Architect, Team	
Ariel Arviv	318393519	DBA, Technical Writer, Team	
Shani Haker	316376748	UX Engineer, Team	

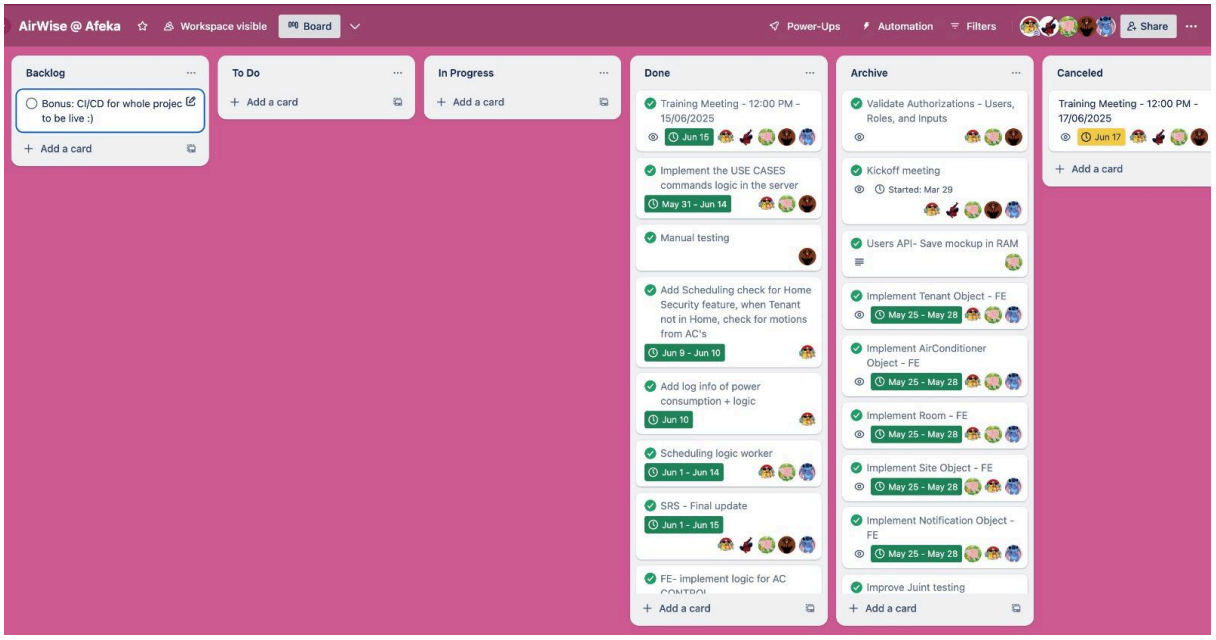
4.2. Kanban Boards

Kanban Board at the start of sprint 5 (29.05.2025):



Kanban Board at the end of sprint 5 (16.06.2025):





4.3. General summary of work in Sprint 5

- **What went well for the team and should be continued the on next phases of work:**
  1. We successfully built the client-side using **ReactJS**, laying a solid foundation for the UI and user interactions.
  2. We also created a working **Node.js + Express** server to serve API endpoints for our AC demo implementation, enabling us to test features end-to-end.
  3. The adoption of **SourceTree** as our Git client provided a consistent workflow for all team members, making version control more intuitive.
  4. All feature branches were merged via **pull requests**, with at least one reviewer involved in every case - this improved code quality and encouraged team-wide visibility

- **What should be improved in team work:**
  1. Improve time estimation during sprint planning by accounting for technical complexity more realistically.
  2. Running tests more frequently can help catch issues sooner, making it easier to resolve them before they escalate.

- **What problems did the team encountered through this phase of work:**
  1. Some adjustments to requirements required small modifications in both the client and server code.
  2. The team had to prioritize features under time constraints, which led to difficult decisions about what to leave out for the final submission.
  3. Some minor inconsistencies between the frontend and backend interfaces were discovered late in the process and required quick adjustments.
  4. We didn't have time to update the automated tests to reflect the latest server changes—the existing tests align with Sprint 4. As a result, we relied on manual testing to validate the current functionality.

- **Why did we not complete all planned work:**
  We did complete all the work related to this phase.