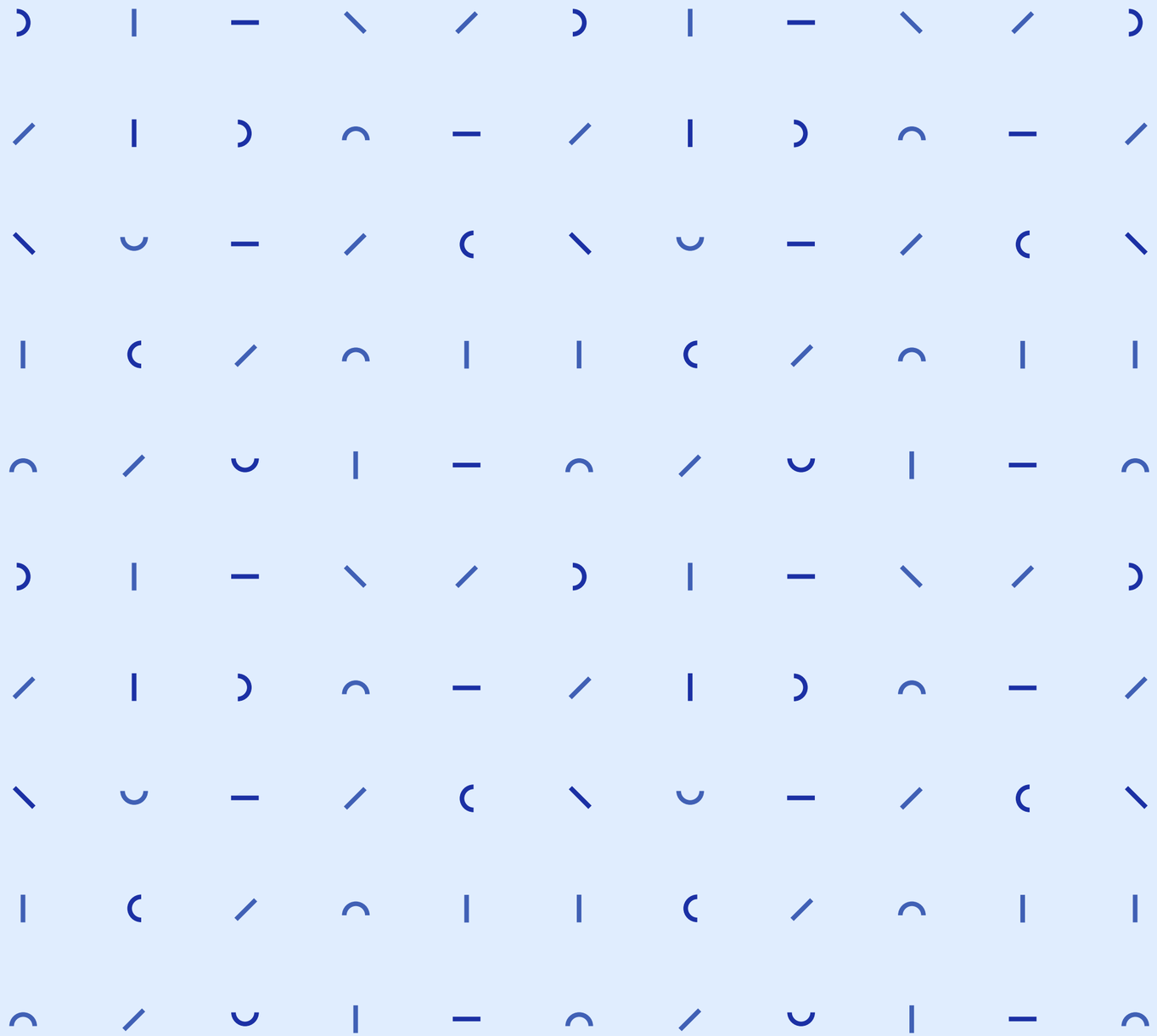
Smart Sleep System

Team 14
전영훈
Maria
Aitana
김윤성
김세령
손희관
송은기

Contents

Introduction



Smart Sleep System

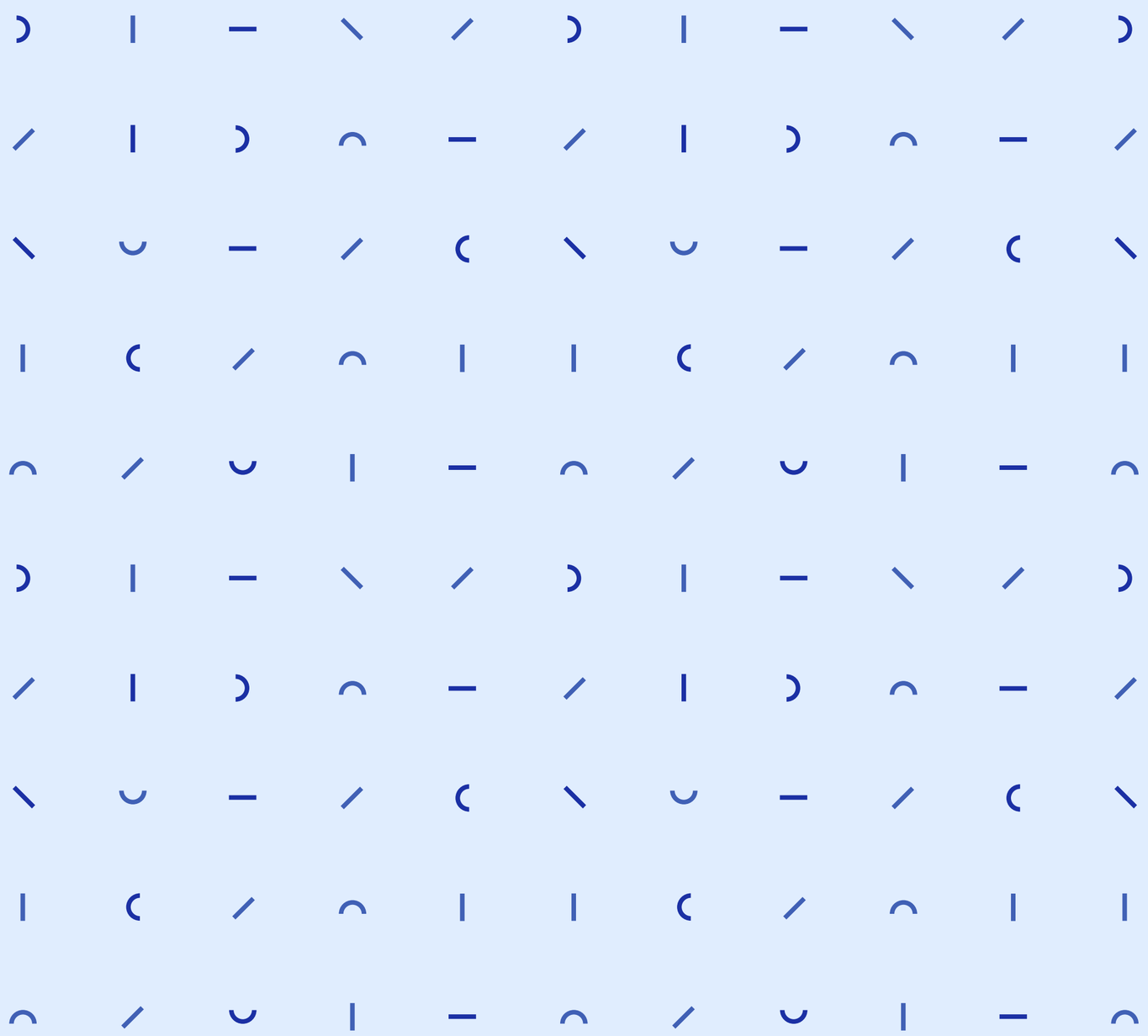
For the perfect sleep of the user

Provide an optimal sleep environment anywhere in a variety of environments

Provides a complete sleep experience for a single user

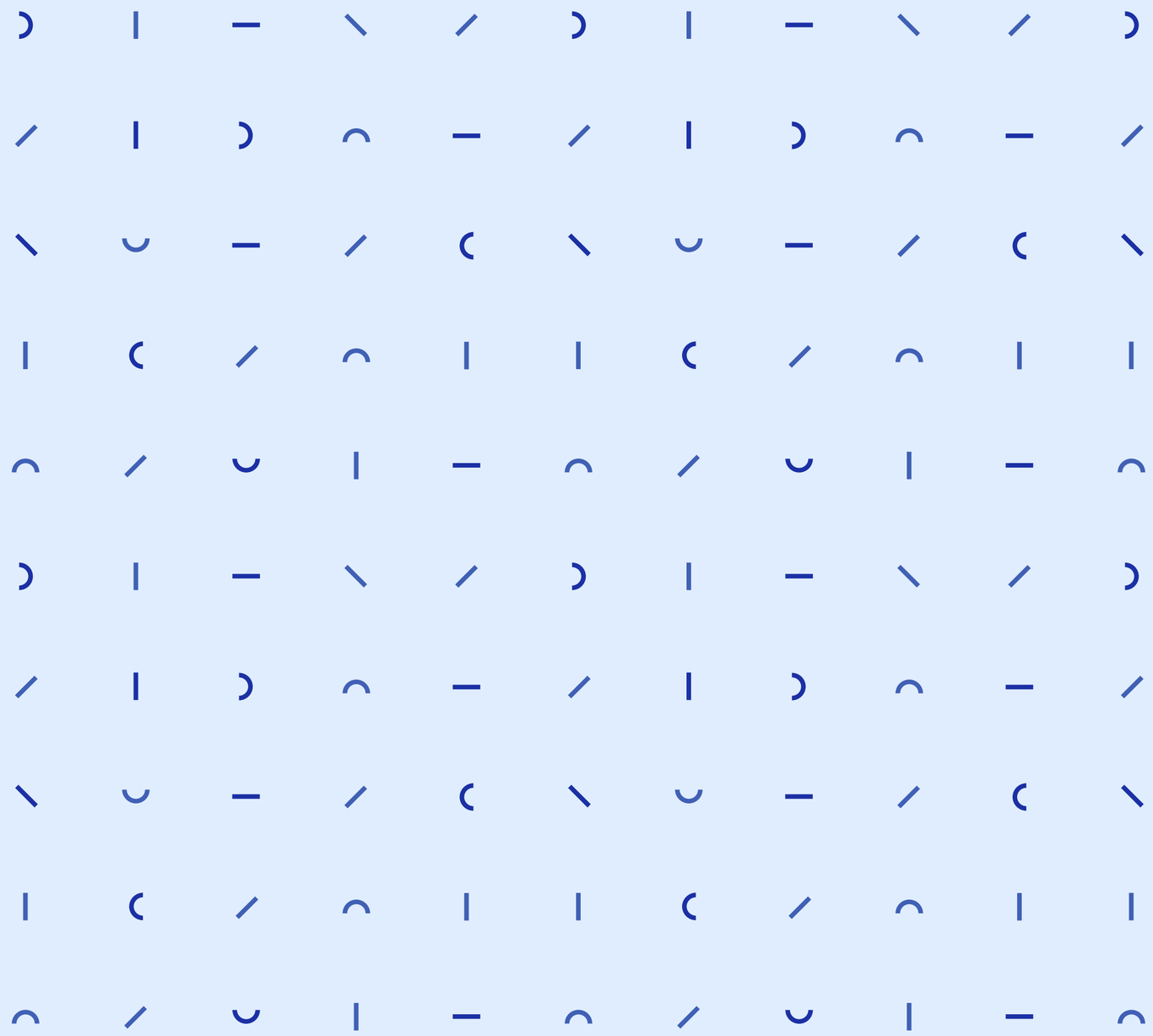
Automatically adjusts brightness, temperature, and humidity during sleep
The environment can be set by the user himself or the system can recommend optimal conditions

SRS



SRS - Overall Description

- Product Perspective
- System Interfaces
- Other Interfaces
- Operations



Product Perspective

Lightness

Lightness plays a key role in maintaining the circadian rhythm which controls sleep.

Temperature

Body temperature decreases as the body gets ready to sleep. High or Low temperature of surrounding environment could disturb people sleeping, increase the number of times people wake up.

Noise

Environmental noises and other sounds could lower the sleep quality.

Product Perspective

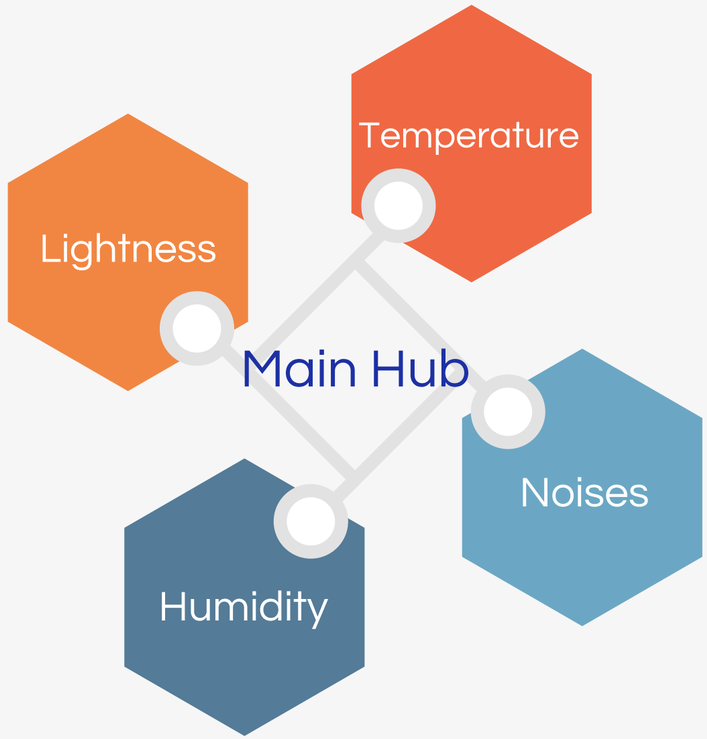
IoT devices control the light, temperature and white noises during user sleeping.

Before sleep, the central hub recreates the dawn, warms up the bed, and sets proper white noises to help user fall asleep.

During sleep, the hub uses the live feedback provided by the sleep tracker to avoid sleep disruptions.

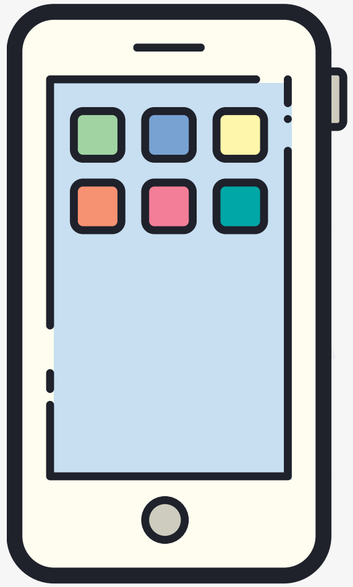
During waking time, the system recreates the sunrise and wakes up the user at the optimal time.

System Interfaces



Firebase

Other Interfaces



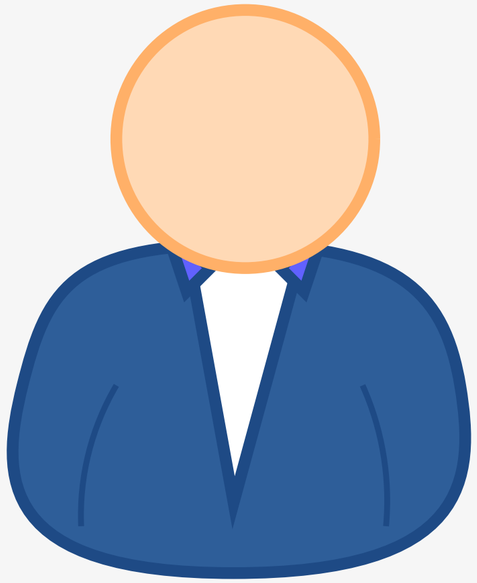
iOS / Android

Qualcomm Snapdragon 845
2GB RAM

API 31 / iOS 14

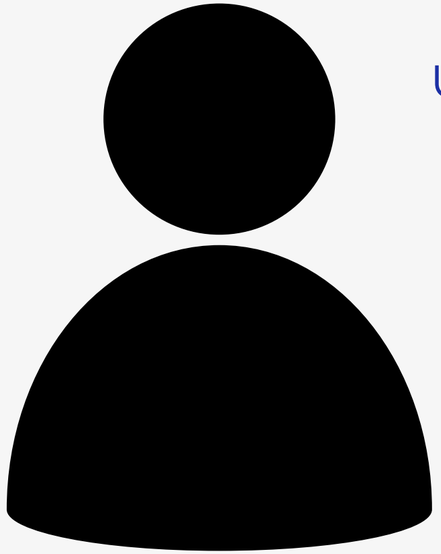
wear OS - API 31
watchOS 8

Operations



Administrator

- Manage user account

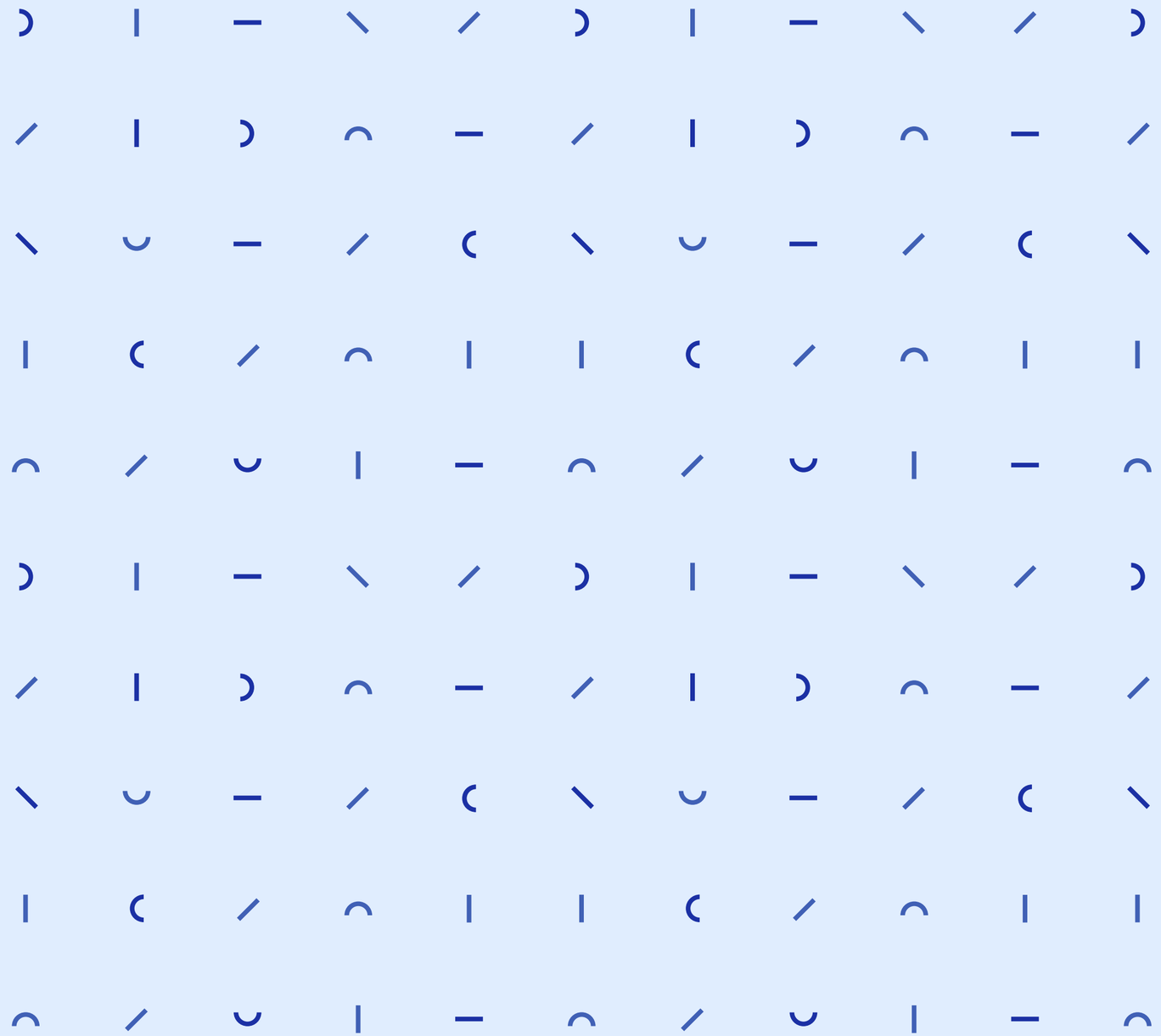


User

- Log in / Log out
- Register smart devices
- Adjust personal settings

SRS- Specific Requirement

- External Interface Requirement
- Function Requirement
- Performance Requirement



External Interface Requirement

Login & Sign Up

Login and find ID/PW like a typical application

Register Device & Controlling Screen

If you want to register a new smart device, you can use the register button, and if you press the existing smart device, a screen that can control the furniture appears.

Functional Requirement

Mangaement

User management should include adding and deleting users and make smart-sleep-system available only when logged in.

Control Parameter

Registered users should be able to control the optimal sleep environment (temperature, humidity, and brightness) as they want through the registered smart-device.

Performance Requirement

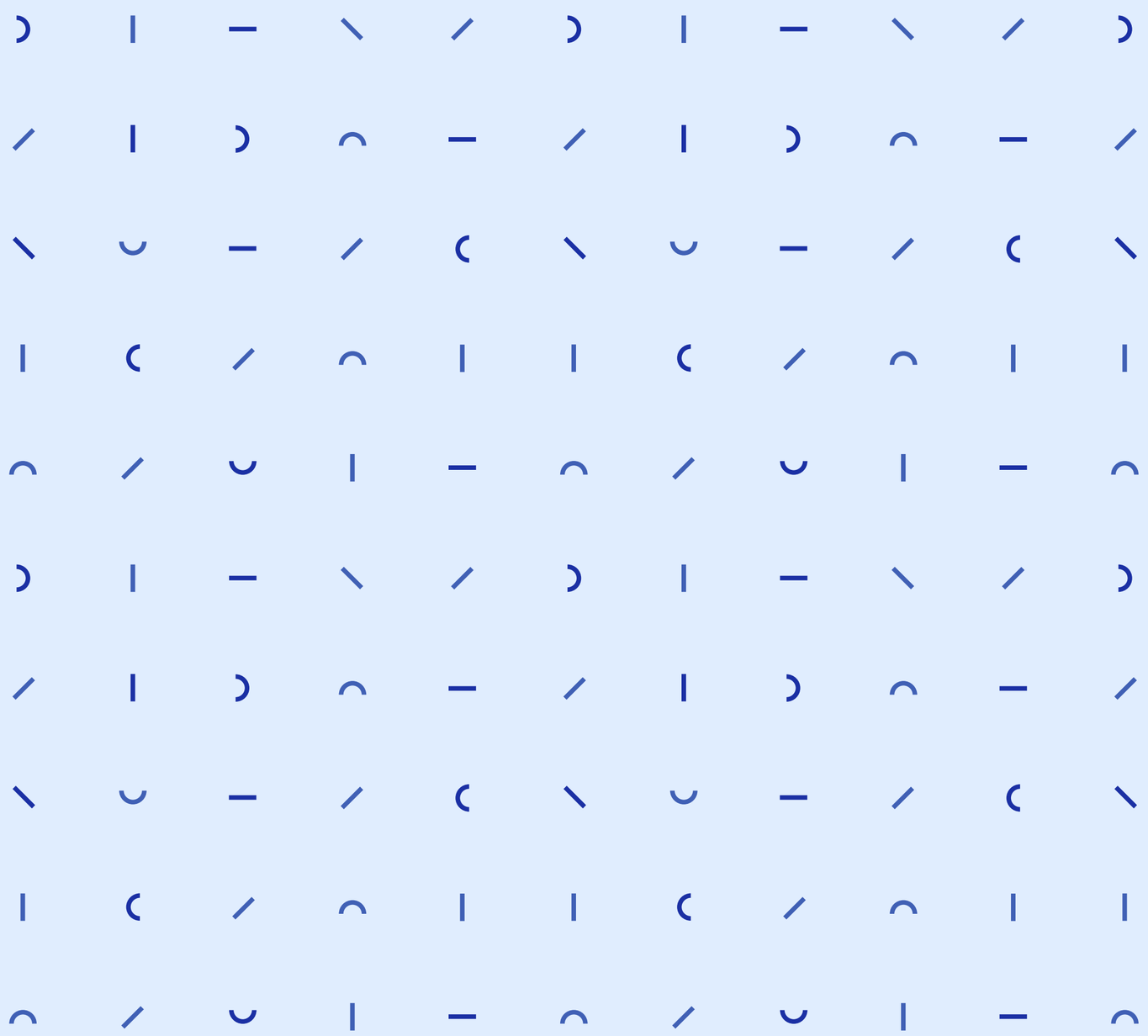
Static Numerical Requirement

There must be only one administrator account, and it takes care of all management, such as user management or devices

Dynamic Numerical Requirement

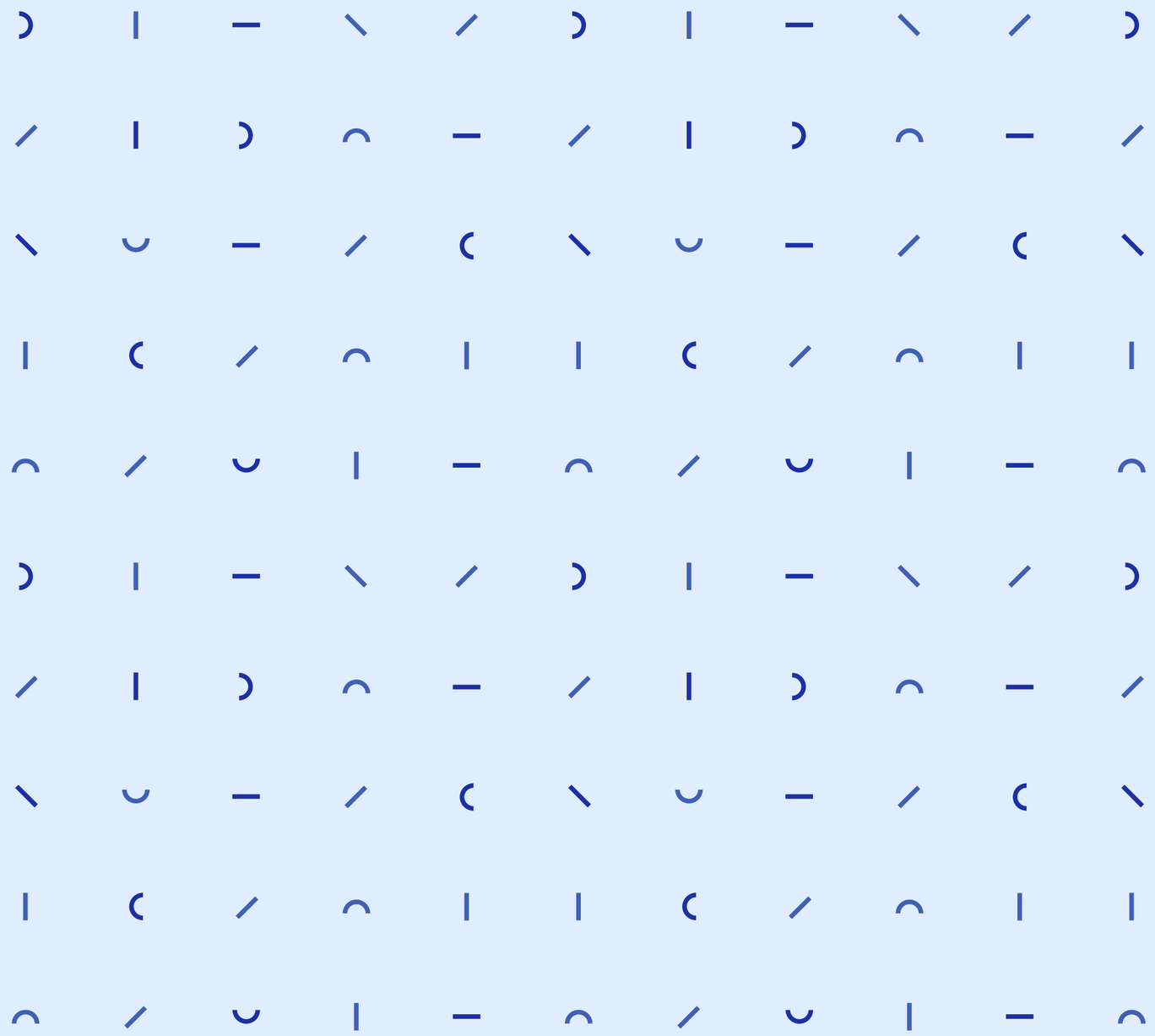
The system must maintain the service well even if many users use it at once, and process the service that users want quickly

SDS



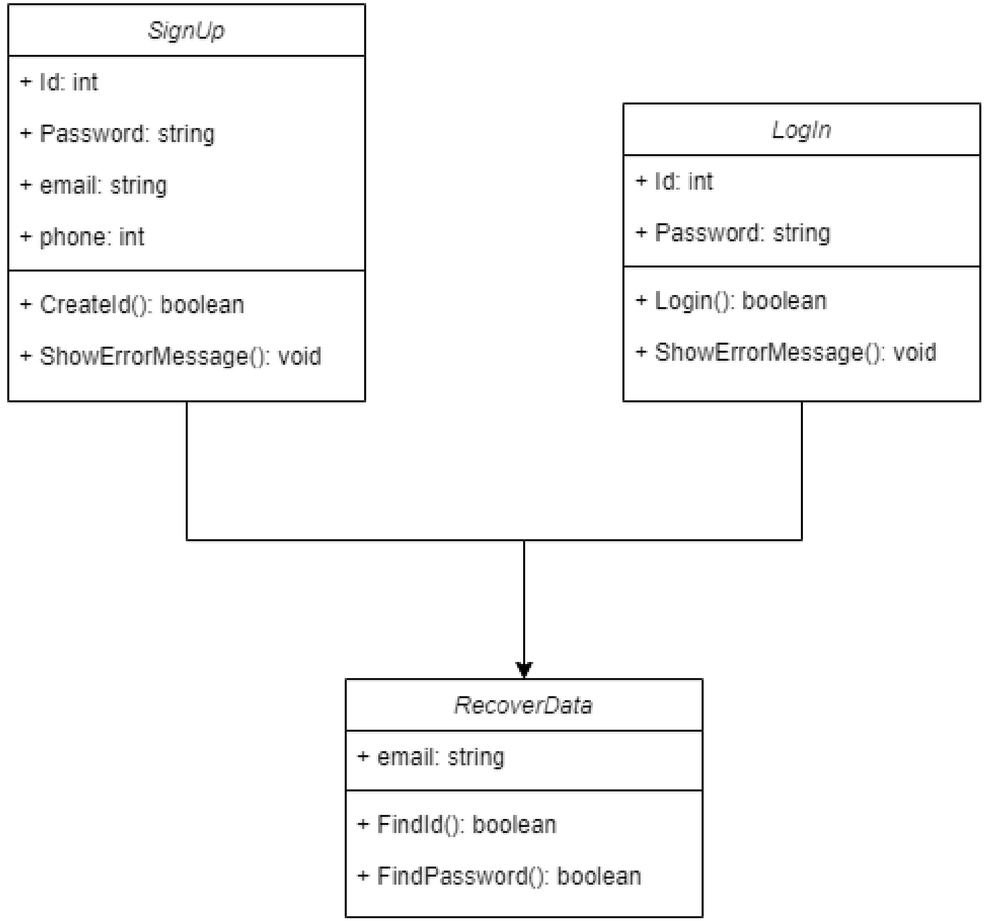
SDS-Frontend

- Account
- Add new smart device
- Smart alarm
- Smart light
- Smart temperature/humidity



SDS Frontend

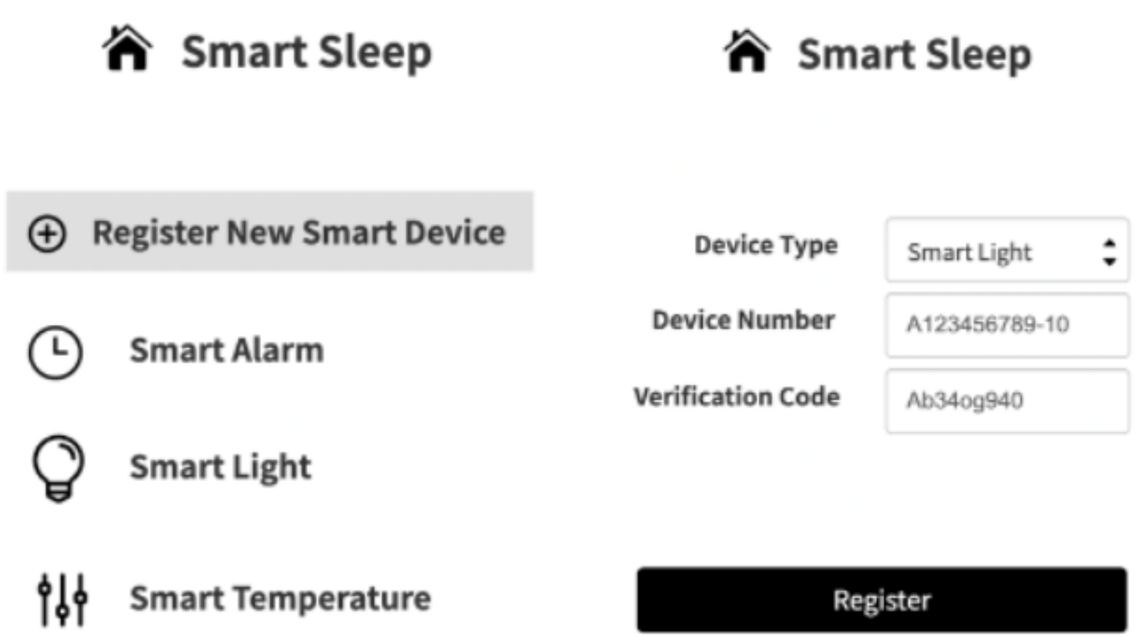
1. Account

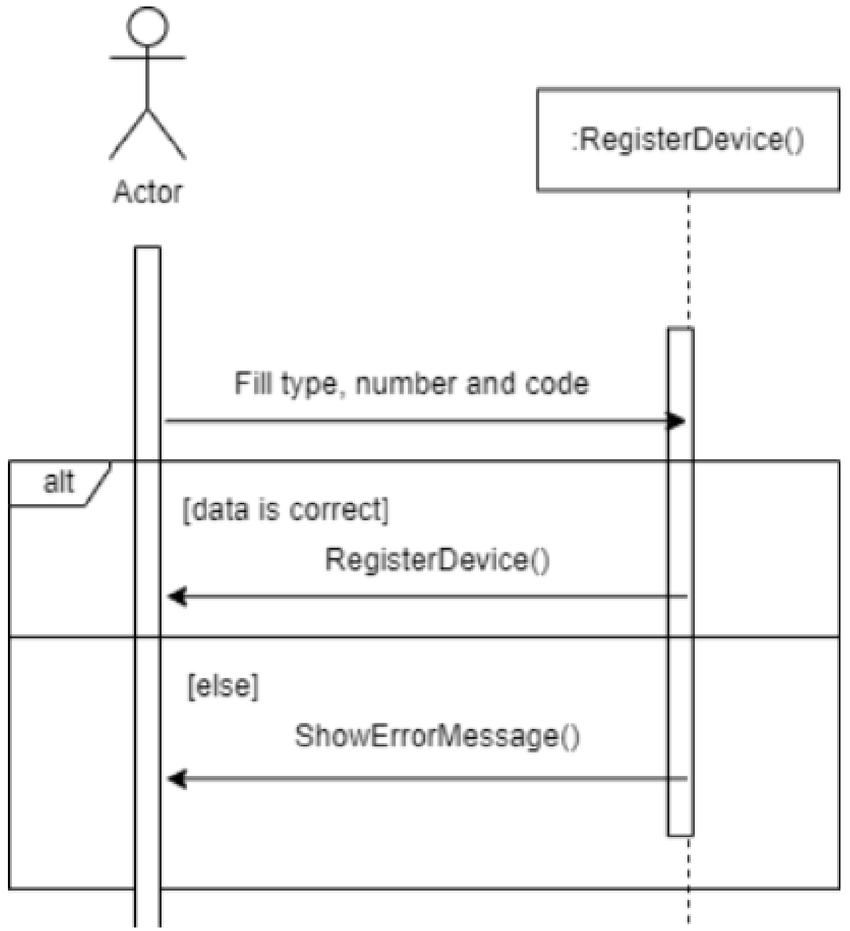


[Figure 13] Class diagram - Account

SDS Frontend

2. Register new smart device

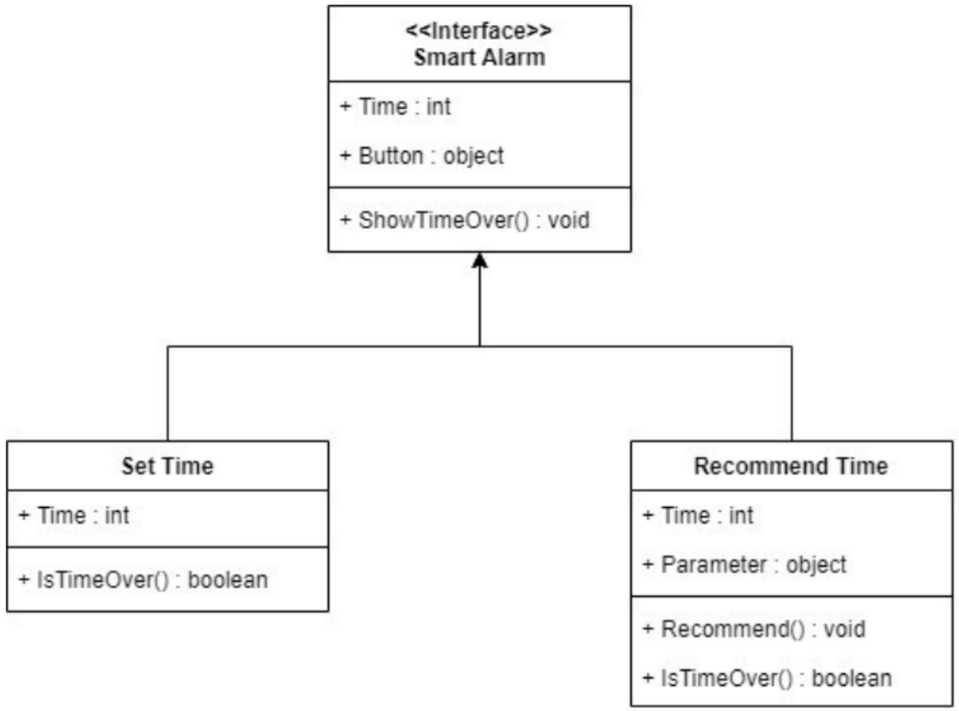




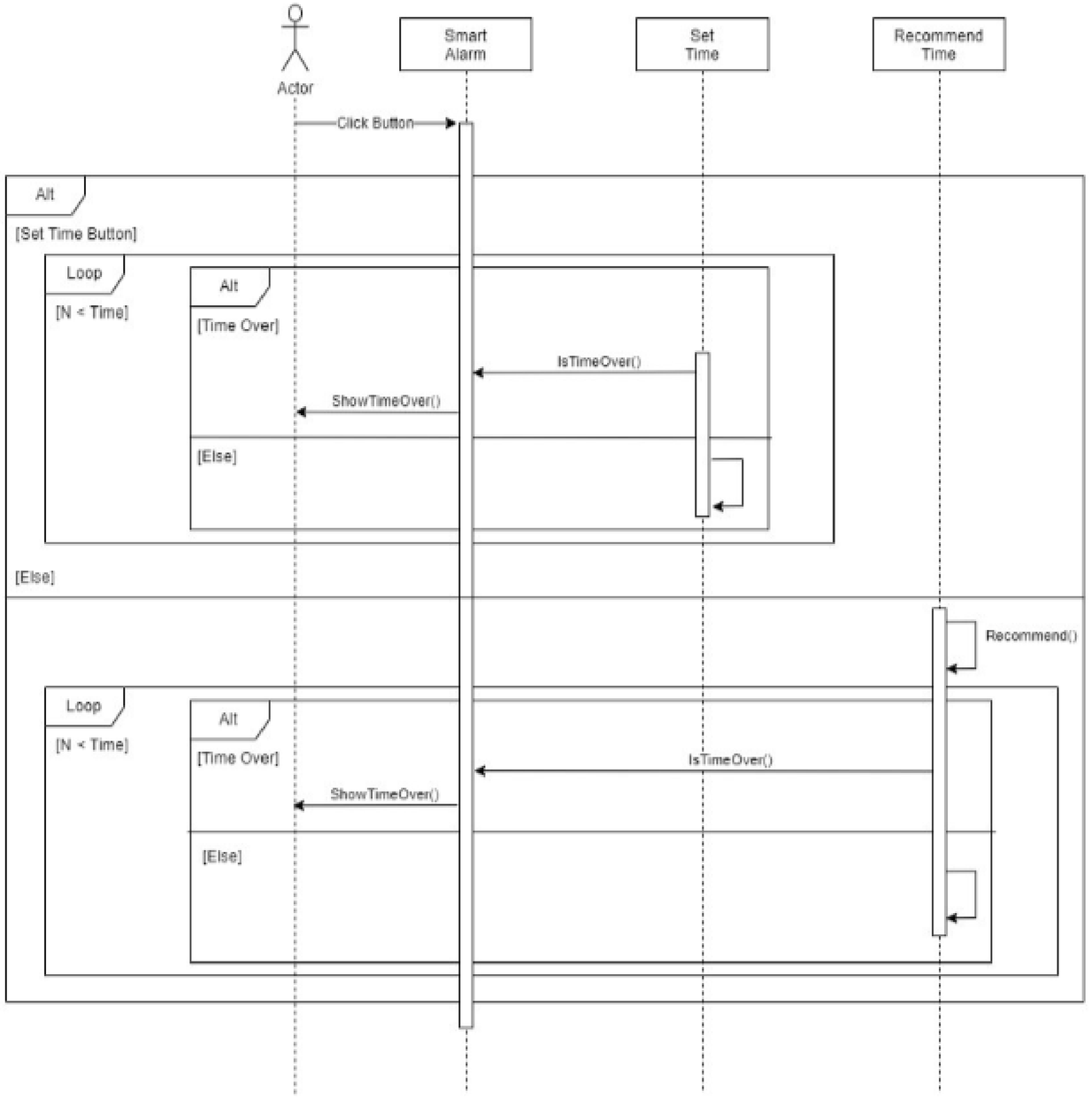
[Figure 16] Sequence diagram – Register Device

SDS Frontend

3. Smart alarm

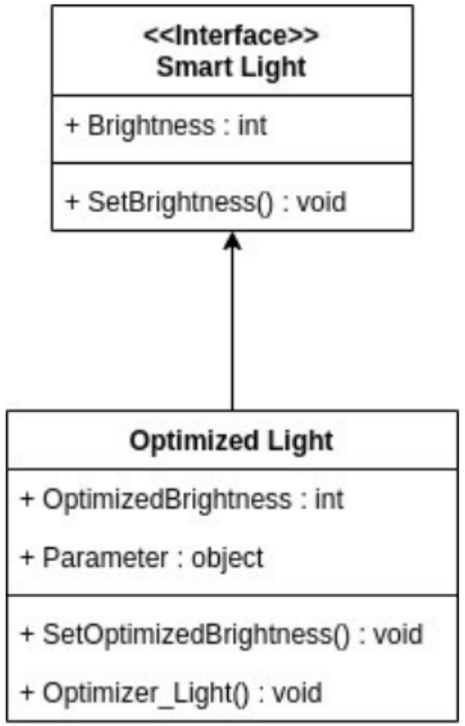


[Figure 17] Class diagram – Smart Alarm

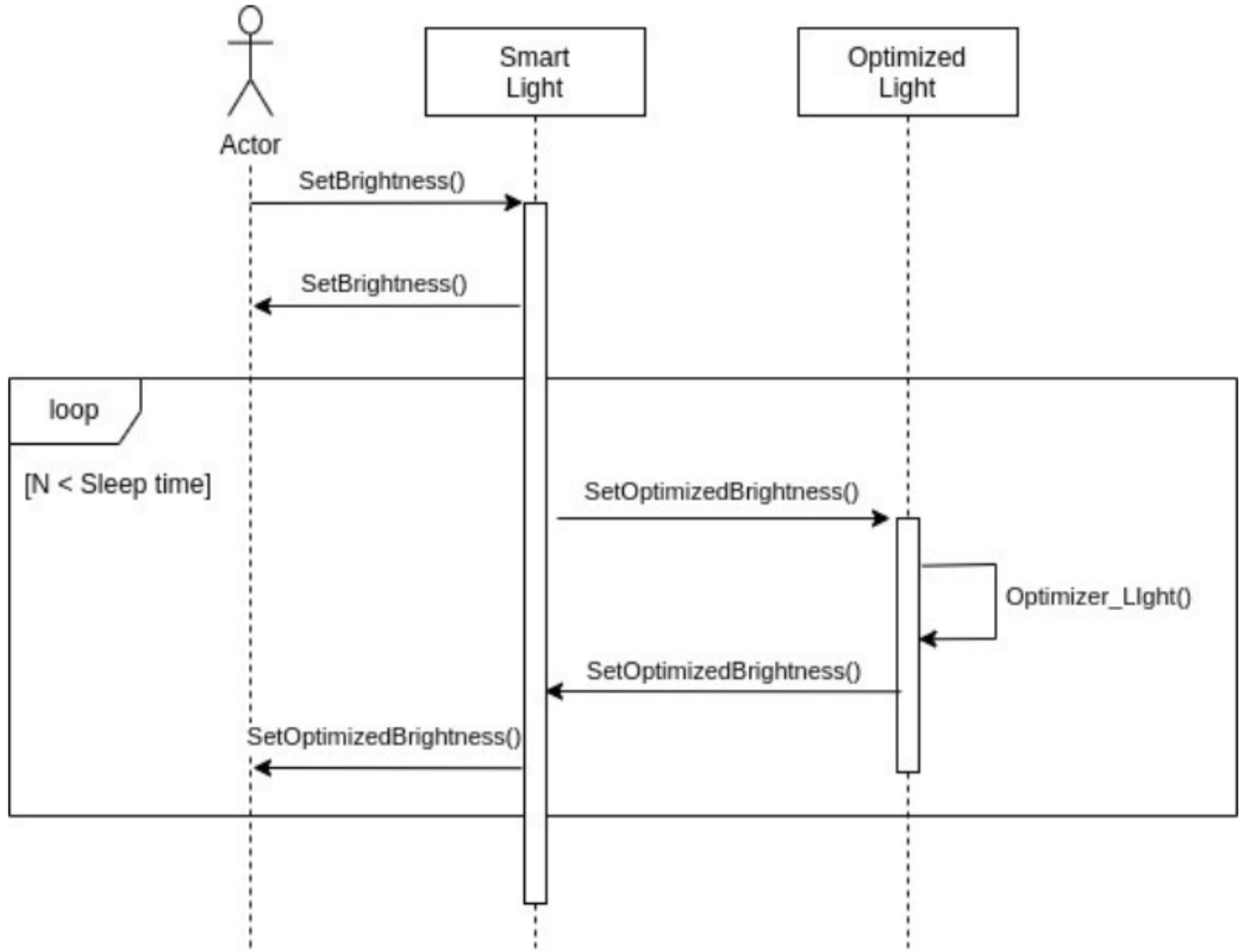


SDS Frontend

4. Smart light

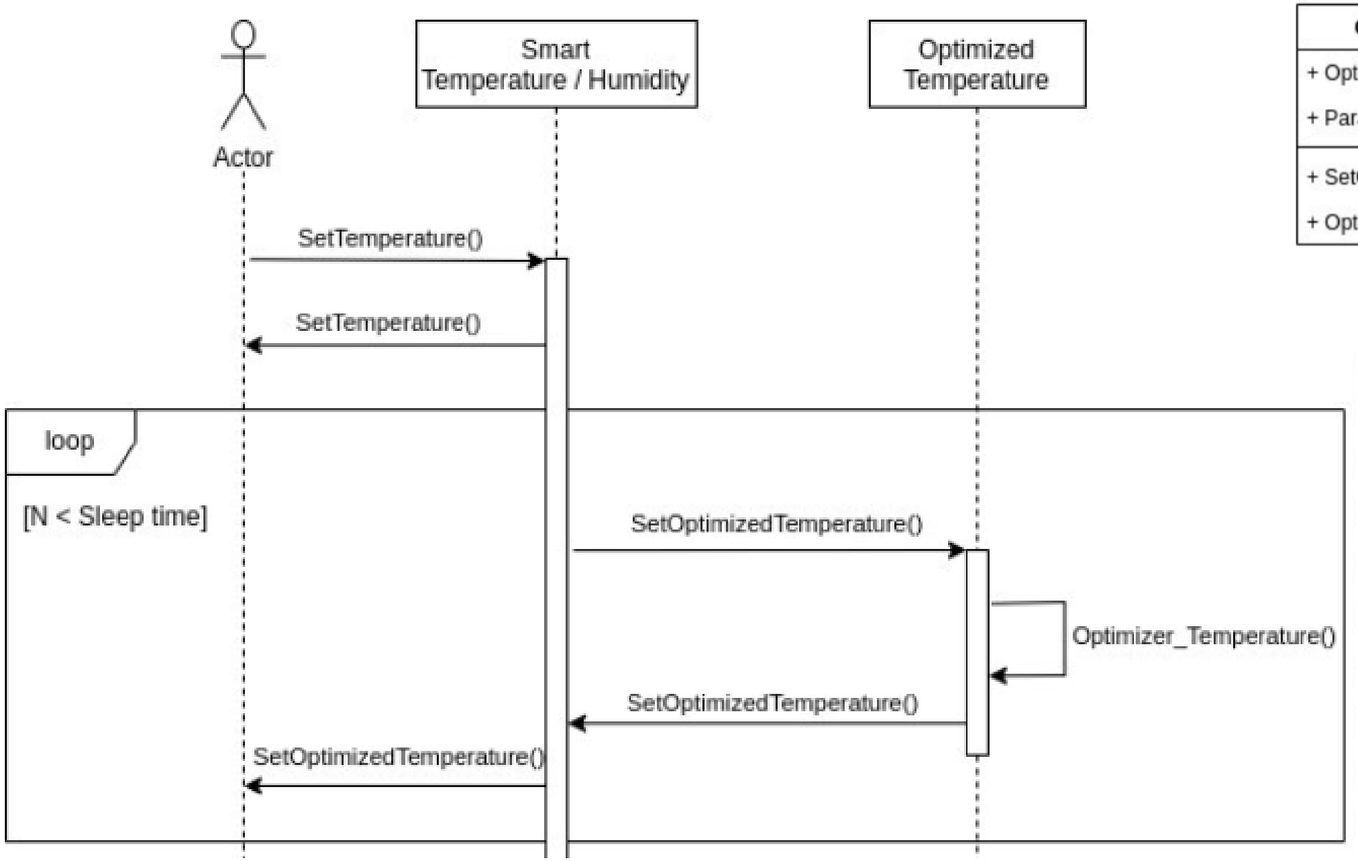


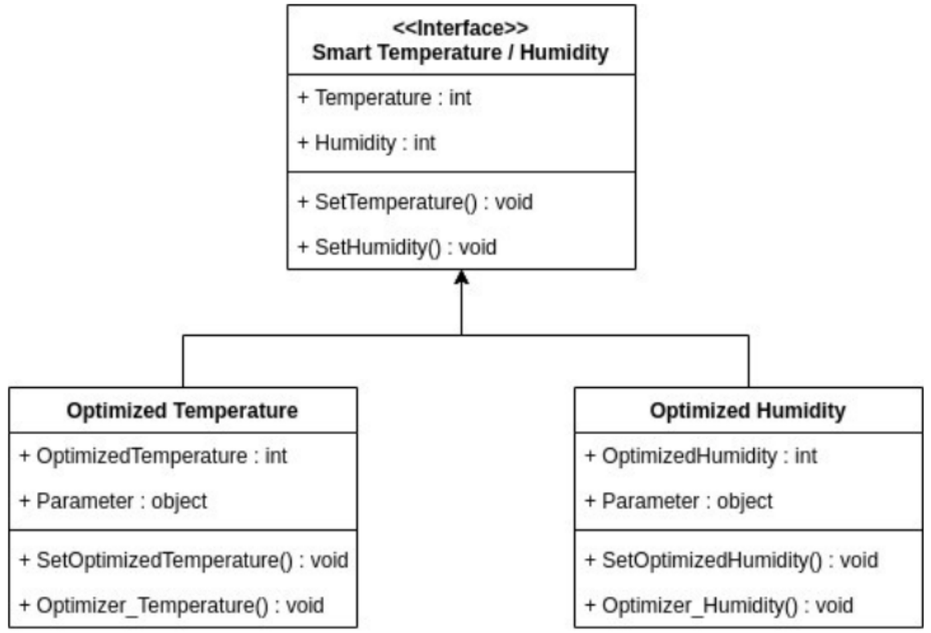
[Figure 19] Class diagram – Smart Light



SDS Frontend

5. Smart temperature & humidity

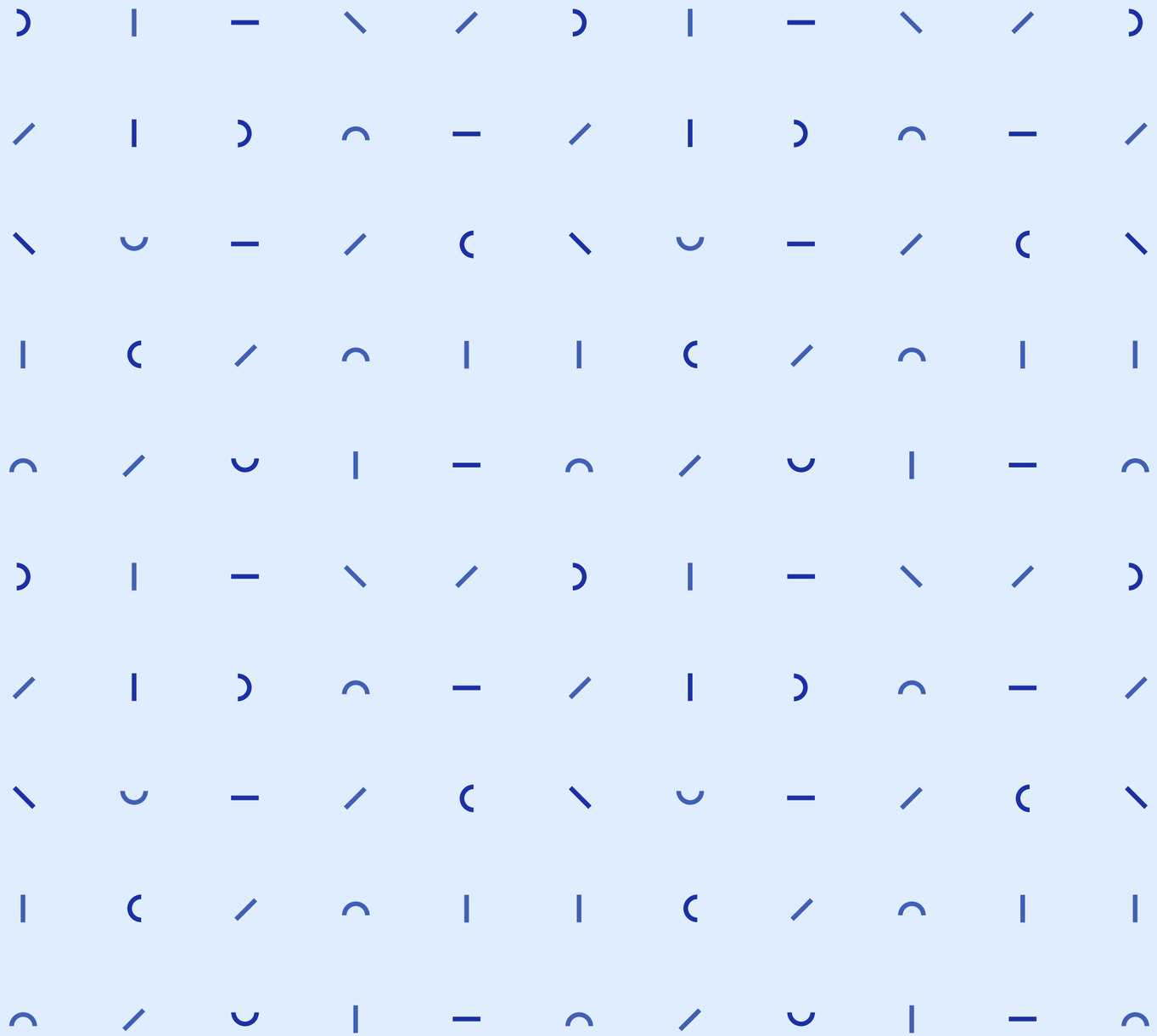




[Figure 21] Class Diagram – Smart Temperature / Humidity

SDS-Backend

- Furniture Controller
- User Action
- Account Registration Management
- Scheduler



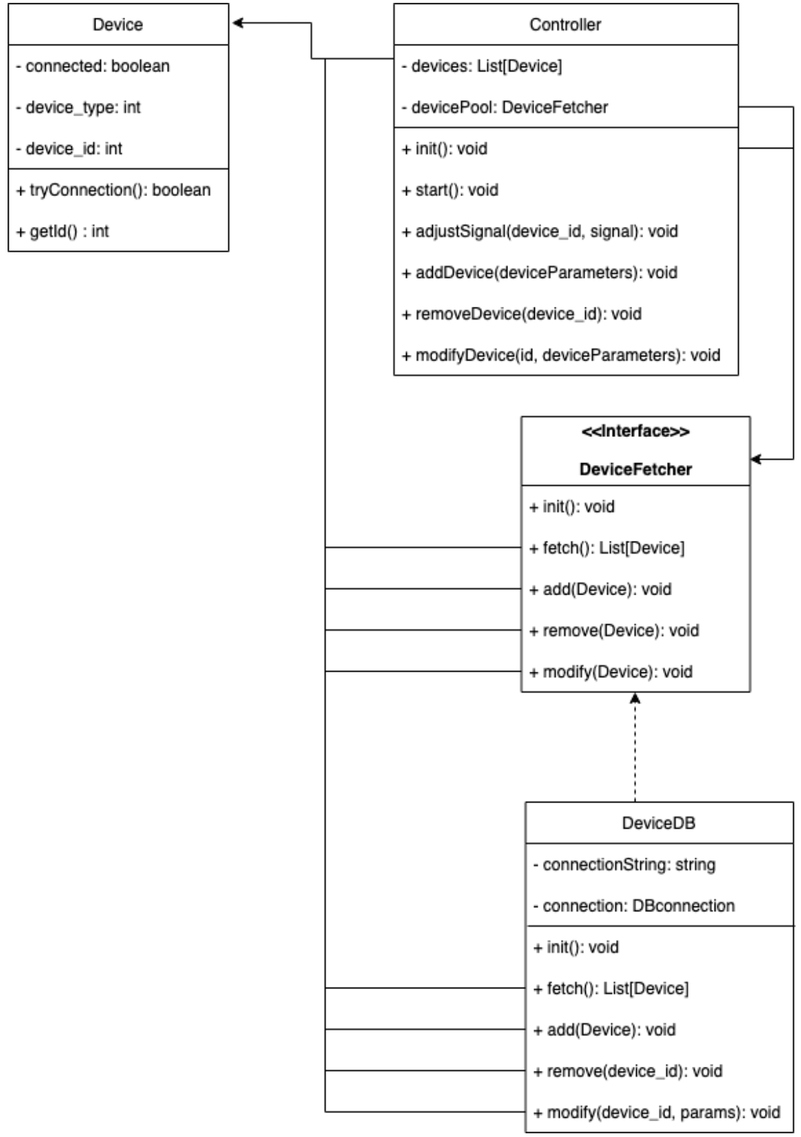
Furniture Controller

Class diagram

The furniture control is the object in charge of controlling the smart devices.

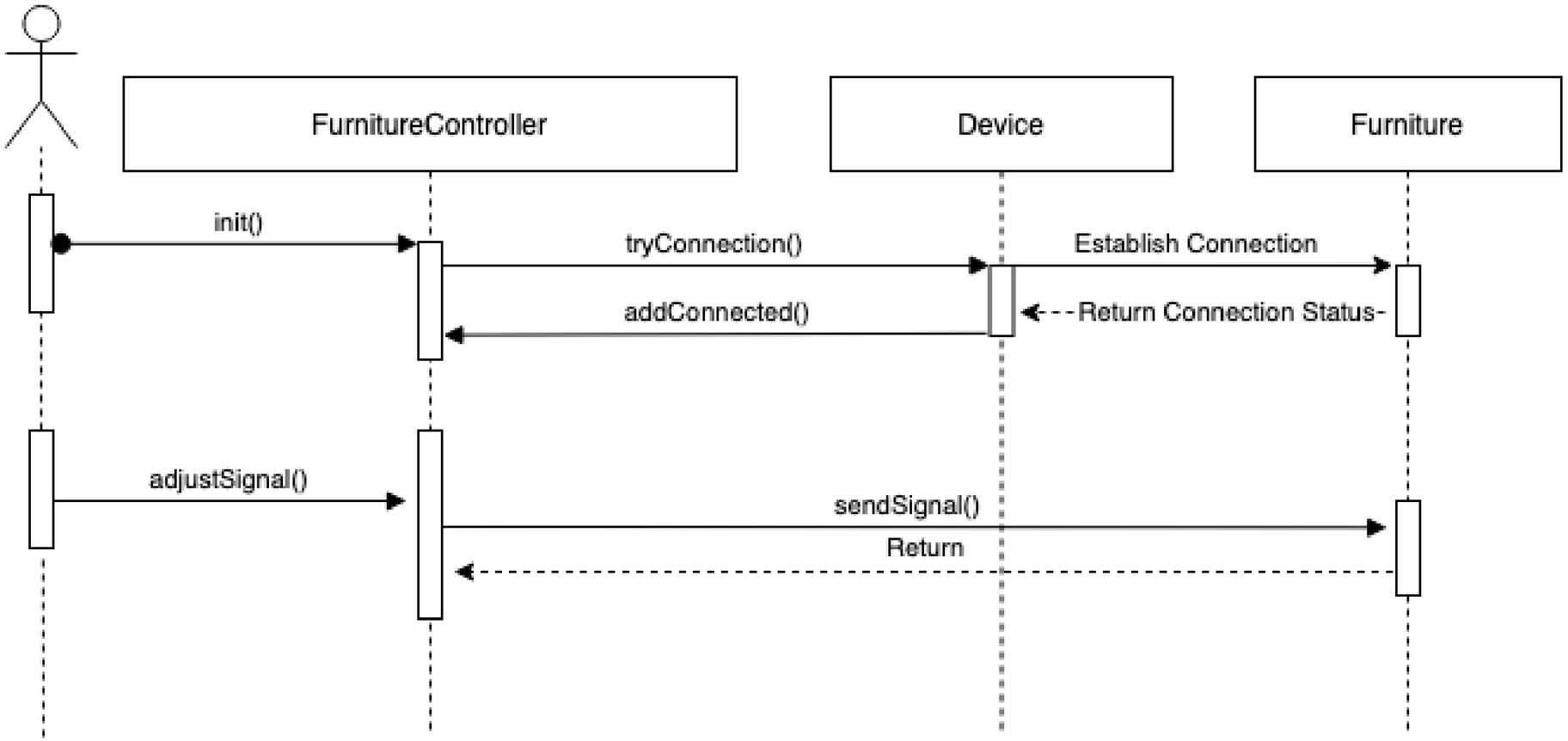
When it is triggered, the furniture controller uses the device's API in order to adjust their parameters.

The controller fetches Device list via DeviceFetcher interface. The implemented class is DeviceDB, which fetch Device from database.



Furniture Controller

Sequence Diagram

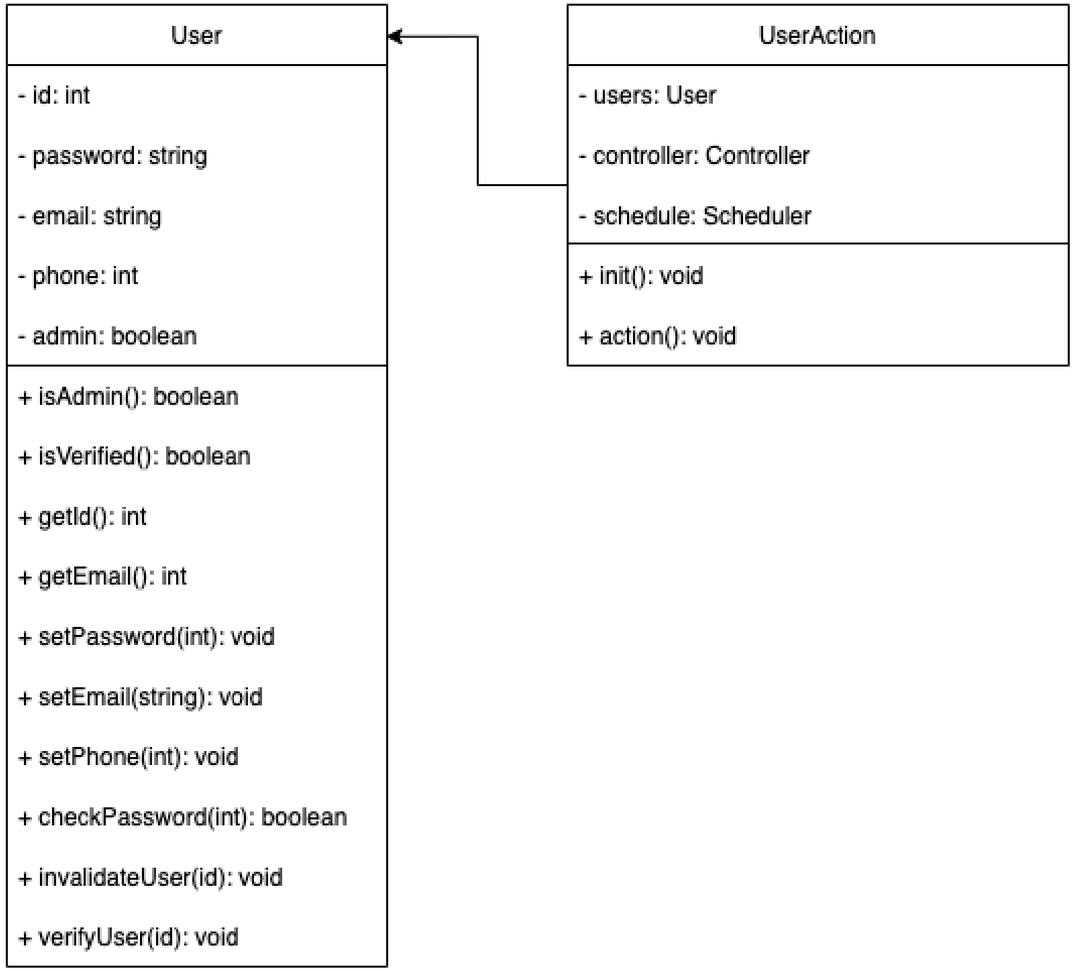


User Action

Class diagram

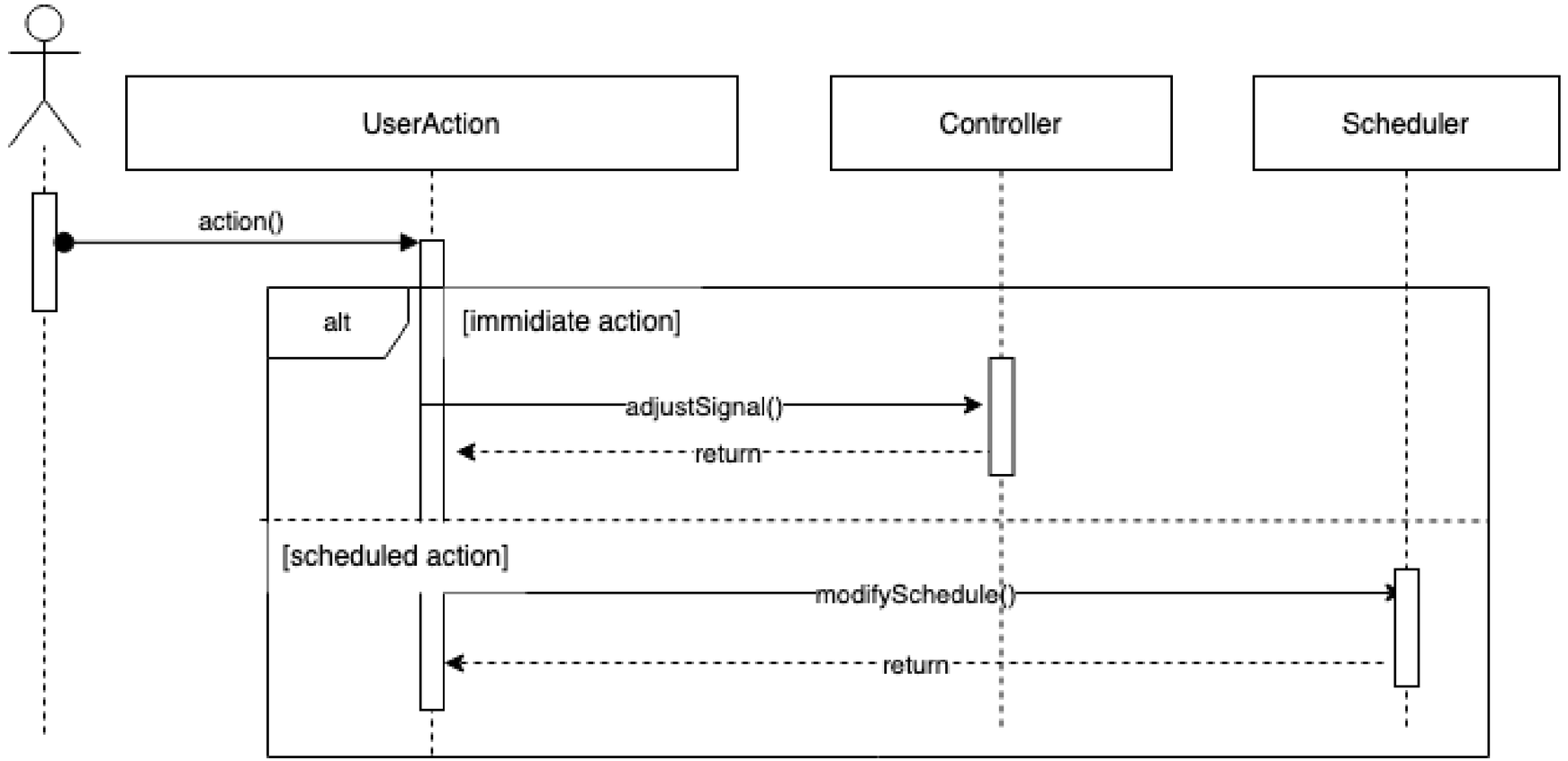
This User Action class receives information from the frontend and deals with the actions accordingly.

User action contains controller and scheduler so user can send signal to controller and adjust schedule.



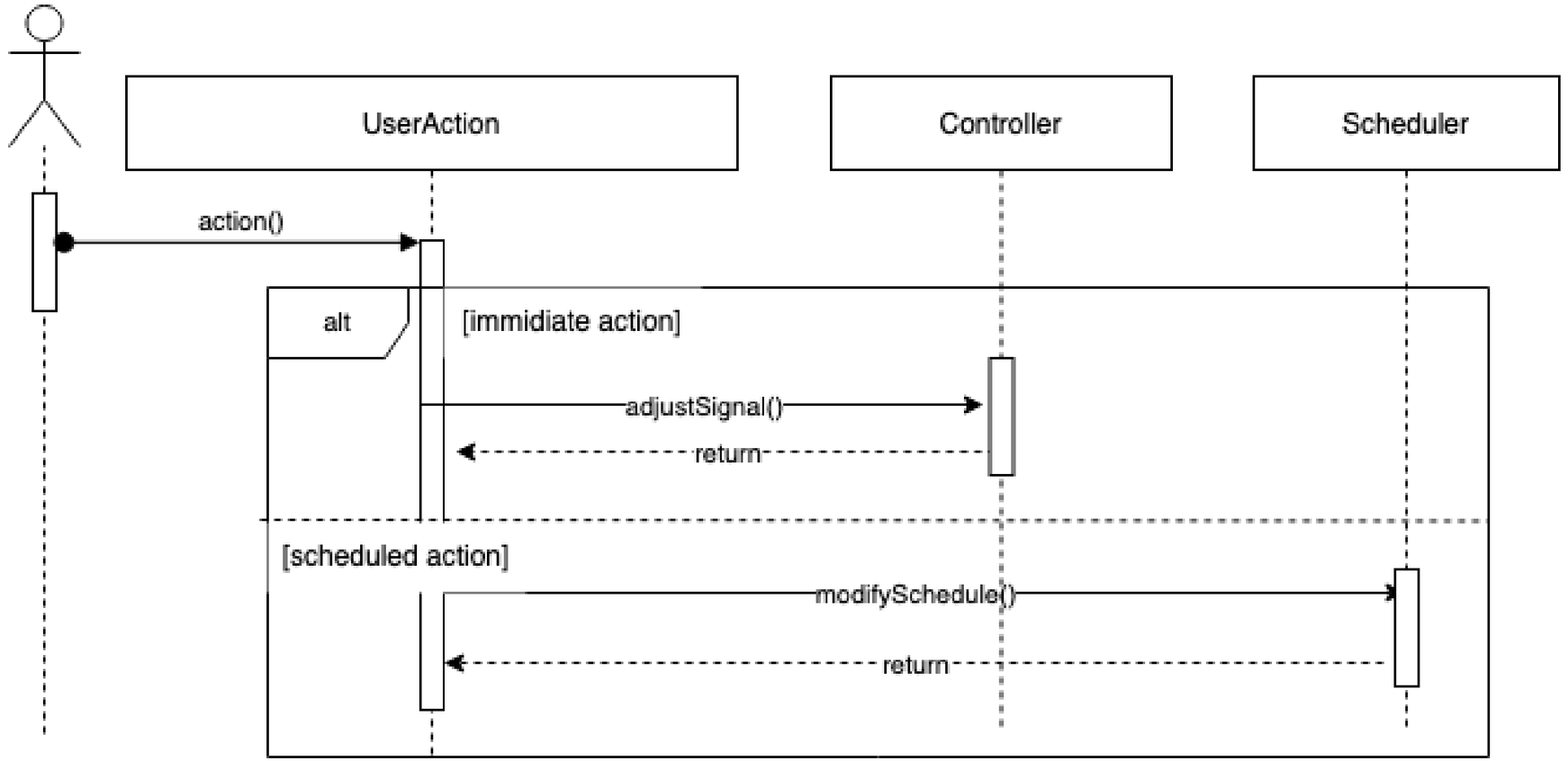
User Action

Sequence diagram



User Action

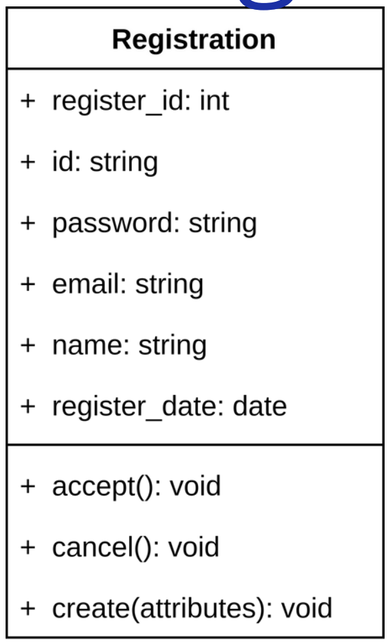
Sequence diagram

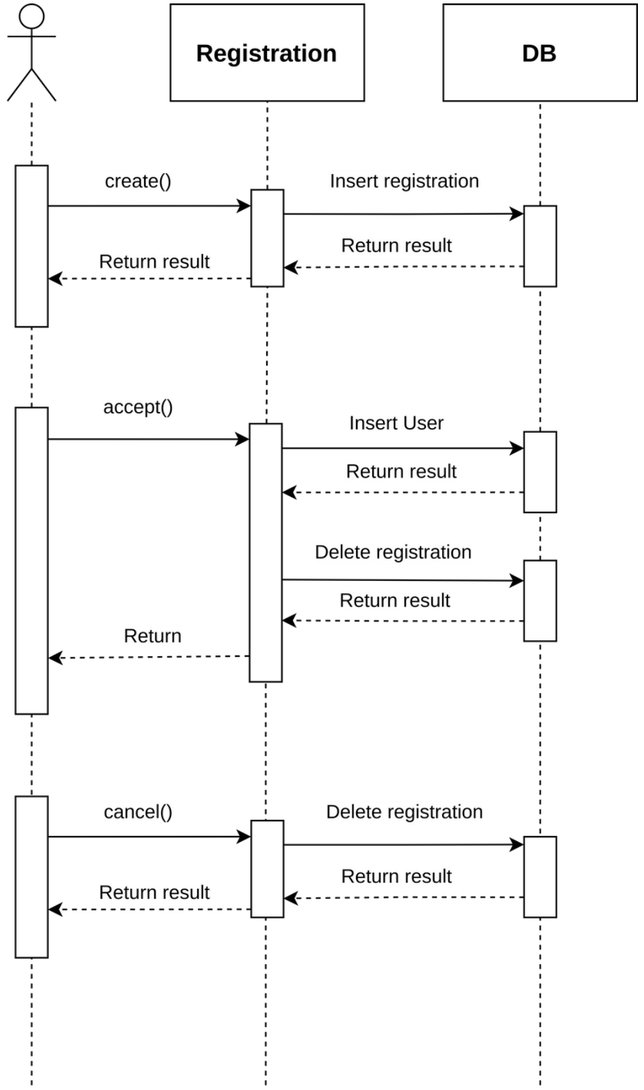


Account Registration Management

The registration is object that is created when the user tries to sign up.

When it's accepted, new user is created in DB.
when it's denied, nothing happens.
In both ways, the registration object is deleted from DB.



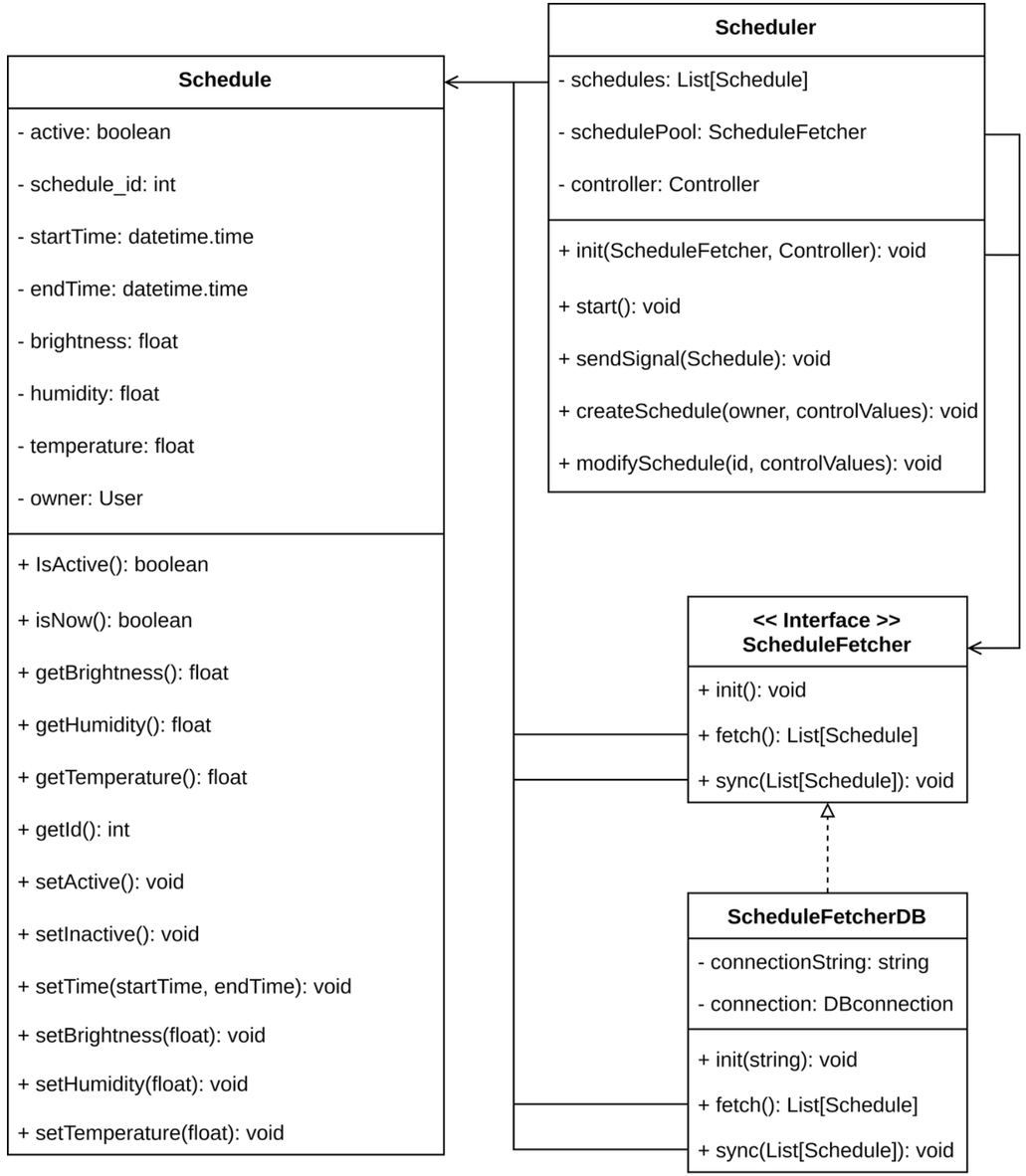


Scheduler
Class diagram

The scheduler is object that is responsible for scheduling action in Smart Sleep.

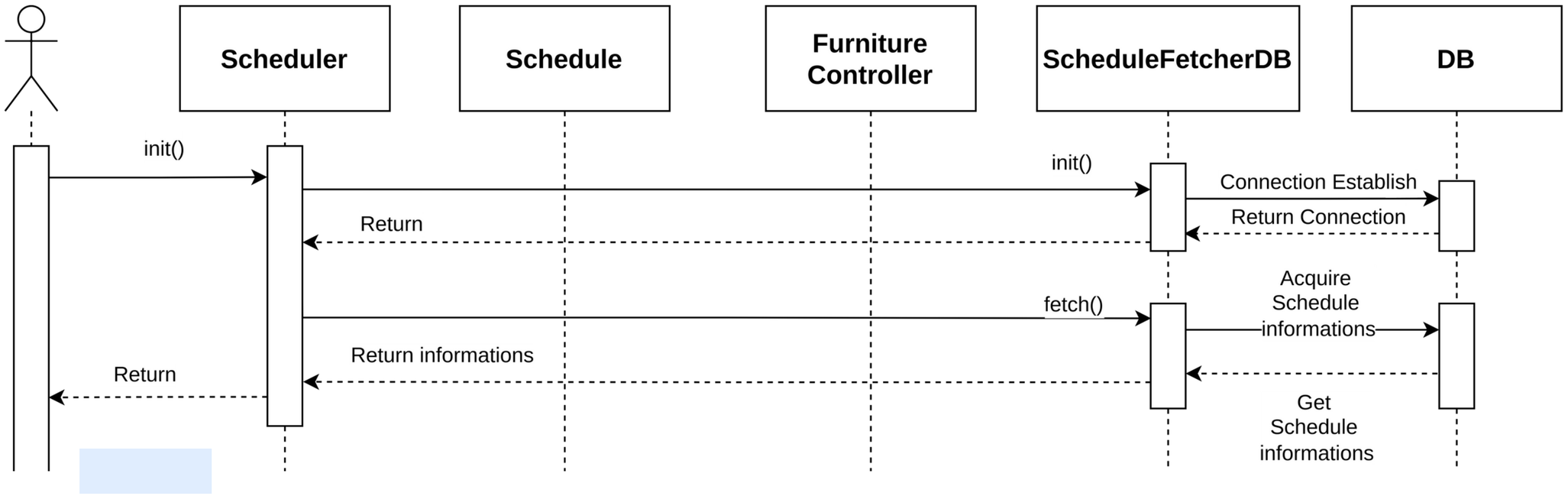
ScheduleFetcherDB is the implemented classes of ScheduleFetcher interface, which stores every schedule in database.

Schedule class describes what schedule object is. Scheduler can check time and acquire control values of schedule by Schedule class methods.



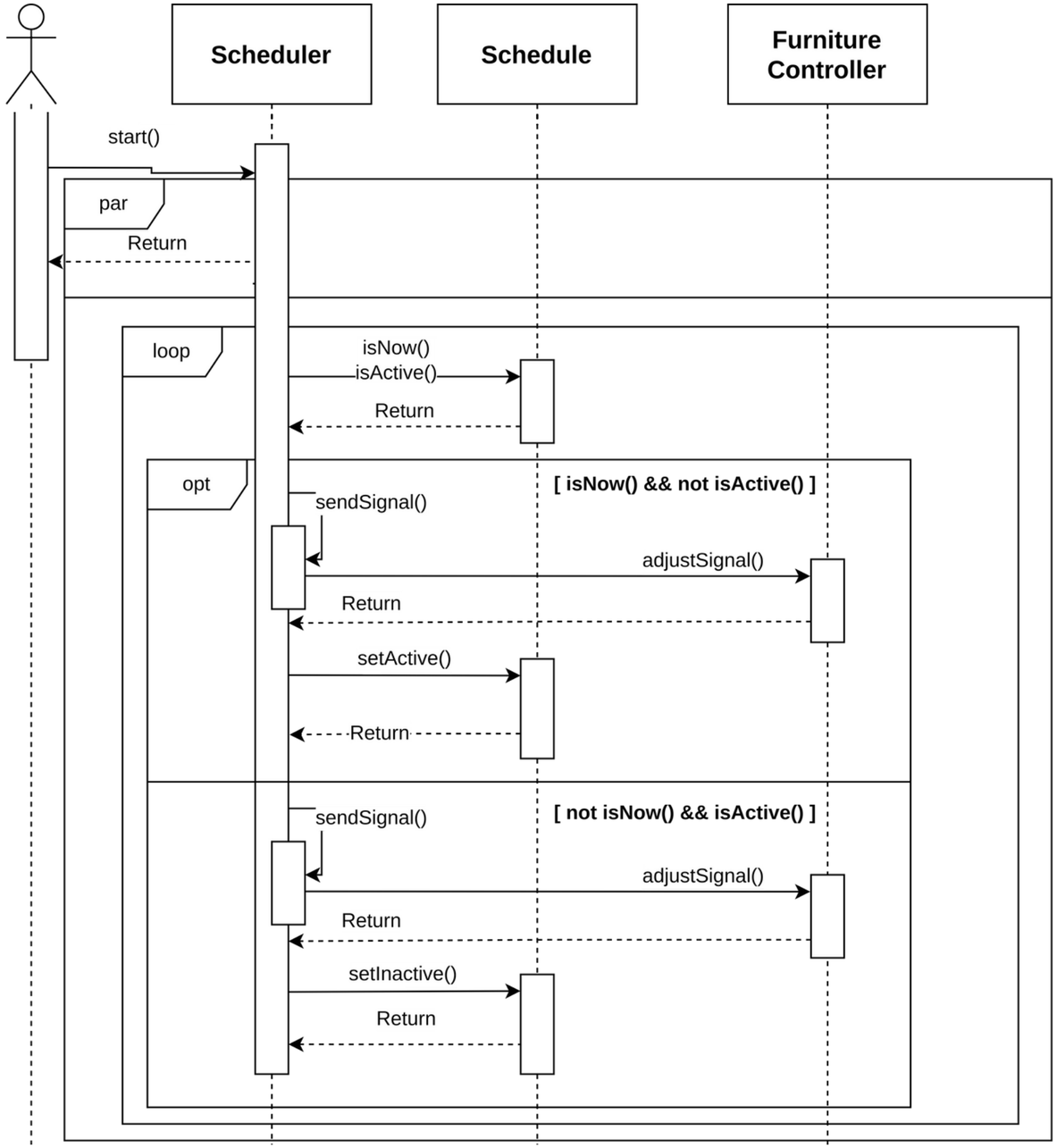
Scheduler

Sequence diagram - initializaition



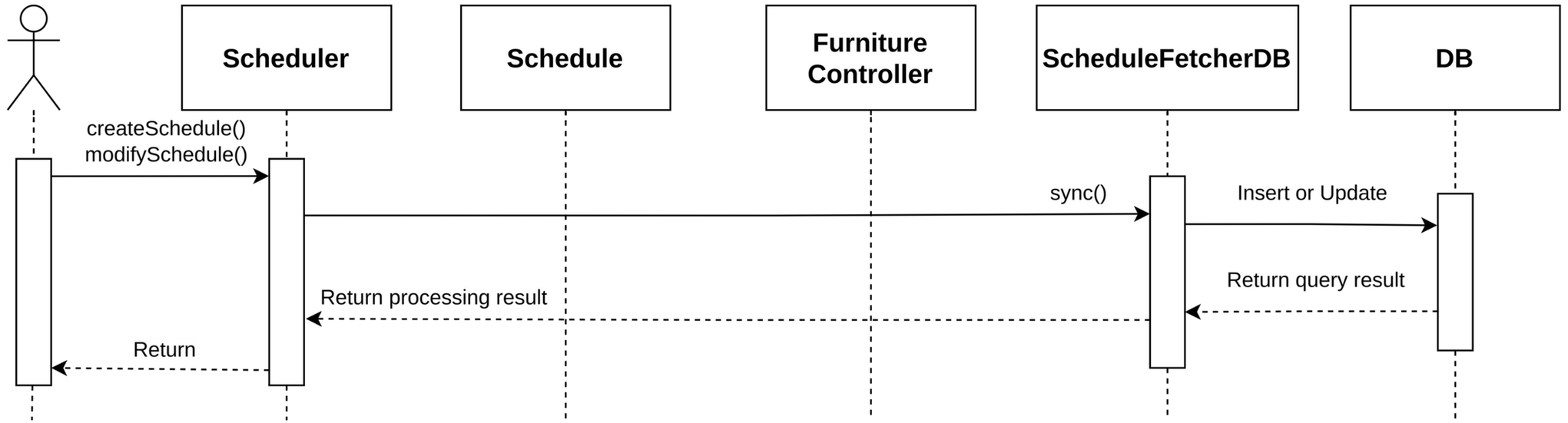
Scheduler

Sequence diagram - Schedule



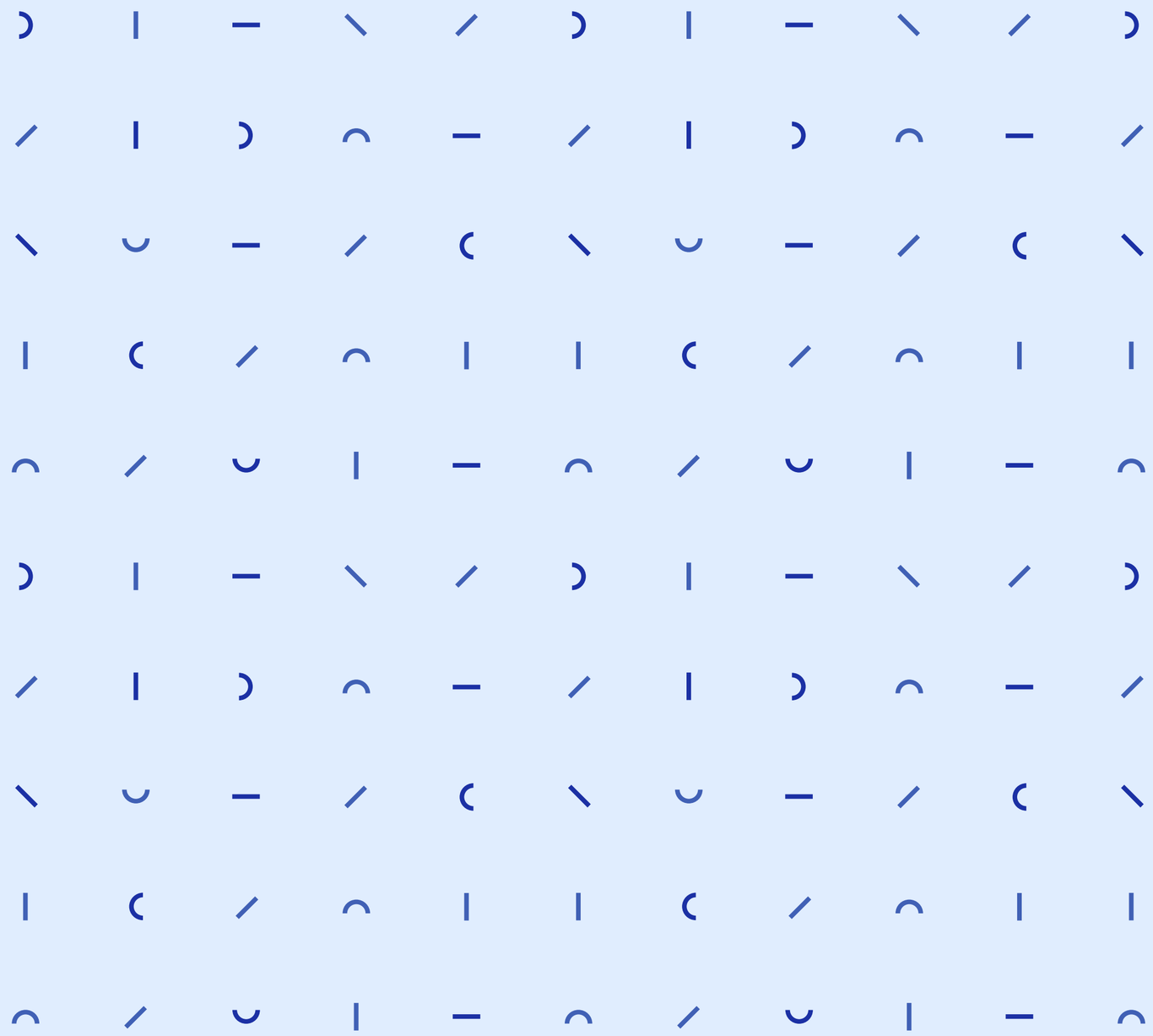
Scheduler

Sequence diagram - create/modify schedule



SDS-
Testing Plan

- Development Testing
- User Testing
- Test Case



Development
Testing

Goal

- Get higher code quality.
- Shorten time to market for new features.
- Reduce software errors.
- Rapid feedback.
- Cheek achievement of non-functional requirements.

Performance

- Test how fast the user can control the desired sleep environment.
- Test whether unregistered users can gain access to the system.

Reliability and Security

- Ensure that the information displayed by the software is accurate.
- Ensure that the user's data is secure.
- Test for vulnerabilities, security loopholes, etc.

User Testing

Beta Testing

Individual without software knowledge

Alpha Testing

Selective user > Software Knowledge

Acceptance Testing

Quality assurance to verify correct test competition

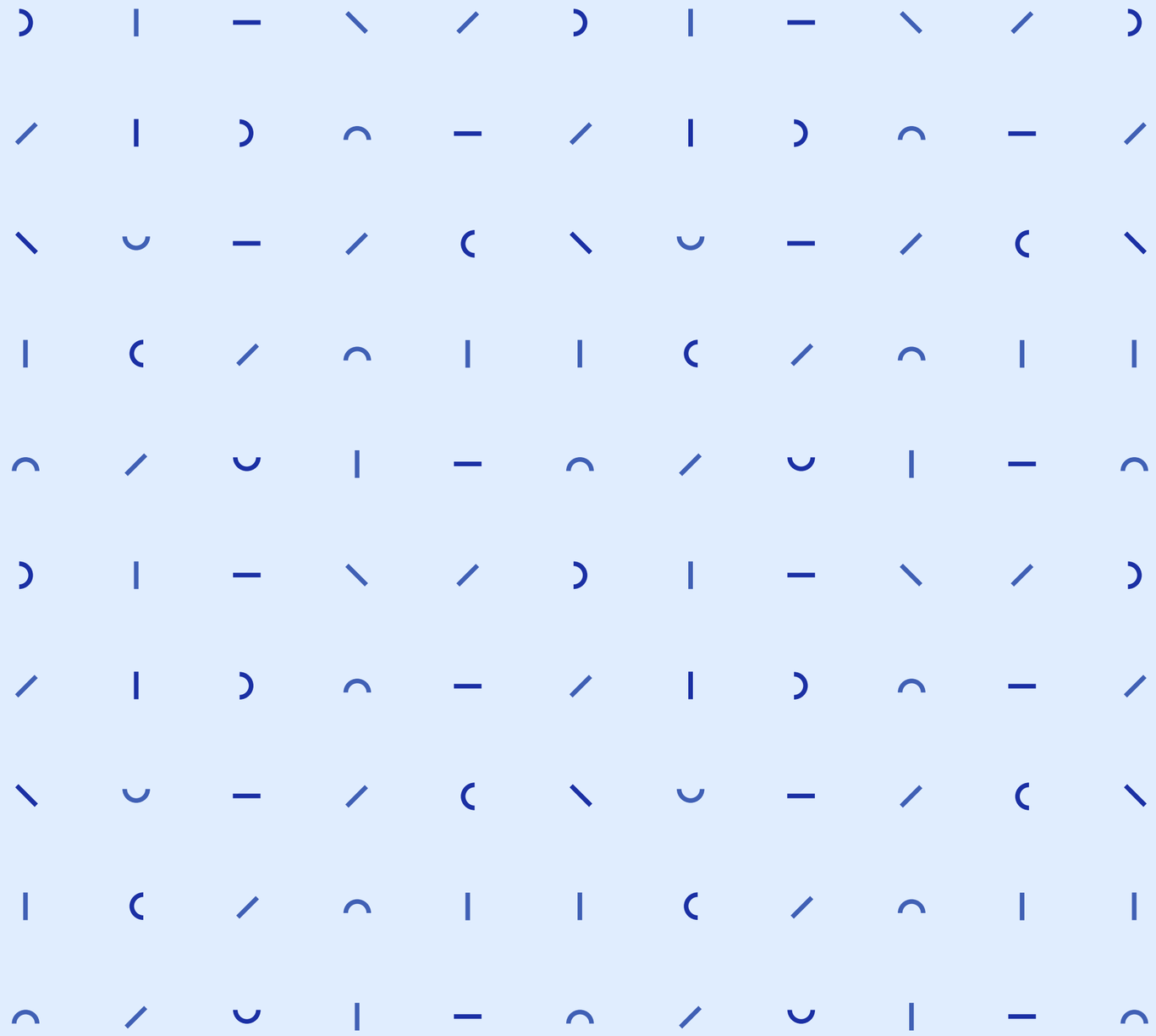
Testing Case

- Ensure furniture is properly controlled for optimal sleep.

- Ensure the system properly prevents misuse and unregistered users from using it.

SDS-Development Plan

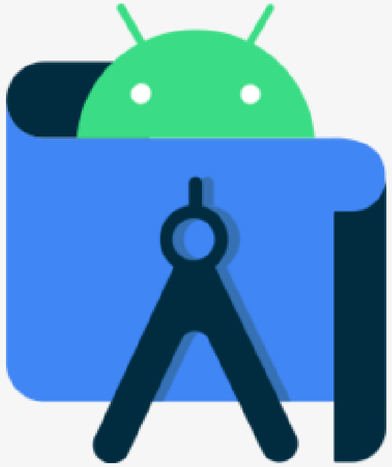
- Front-end Environment
- Back-end Environment



Front-end Development

OvenApp

Xcode

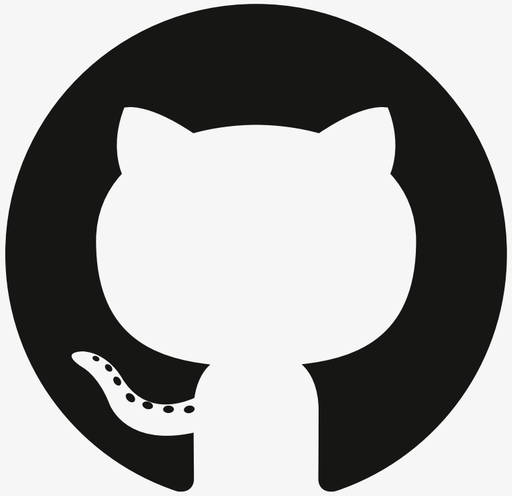


Android Studio

Flutter

Back-end Development

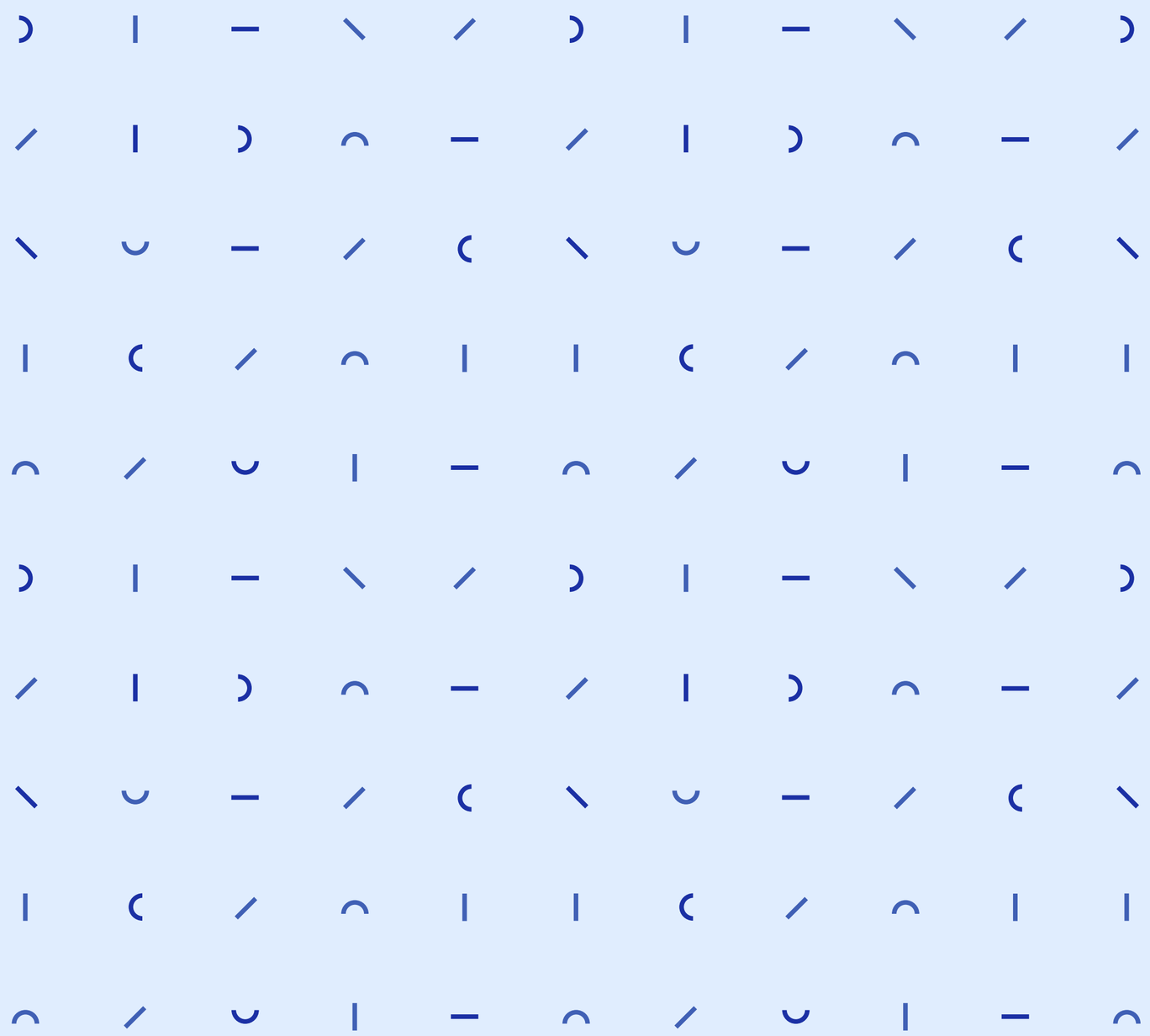
Flask



Github

Eclipse Paho

Direction of improvement



Direction of Improvement

Improve Scheduler Algorithms

Improve scheduler algorithms that analyze and optimize user sleep patterns

Add features for multiple users

Currently, it provides the best sleeping environment for only one user
To connect multiple devices to provide an optimal sleep environment for multiple users.

Thank you!