HOMEWISE

SMART HOME SYSTEM MOBILE APP FOR PREDICTIVE MAINTENANCE

ABSTRACT

- The aim of this project is to develop a mobile application that collects data from sensors in home appliances and transfers this data to users. Thus, users will be able to monitor and detect potential problems in their smart home devices through the mobile application. The name of the application is "**HomeWise**".

PLANNING AND REQUIREMENTS

Scope:

-laundry machine

-dishwasher

-fridge

-climate

-Robot Vacuum Cleaners

PLANNING AND REQUIREMENTS

Functional requirement:

-Microcontrollers in the device analyze error codes

-Sensors automatically calculate the maintenance period

-The data from the sensors is transferred to the cloud via Wi-Fi

Non-functional requirements:

-The system must work stably in any environment.

-Data must be well secured.

-Easy to develop.

DESIGN

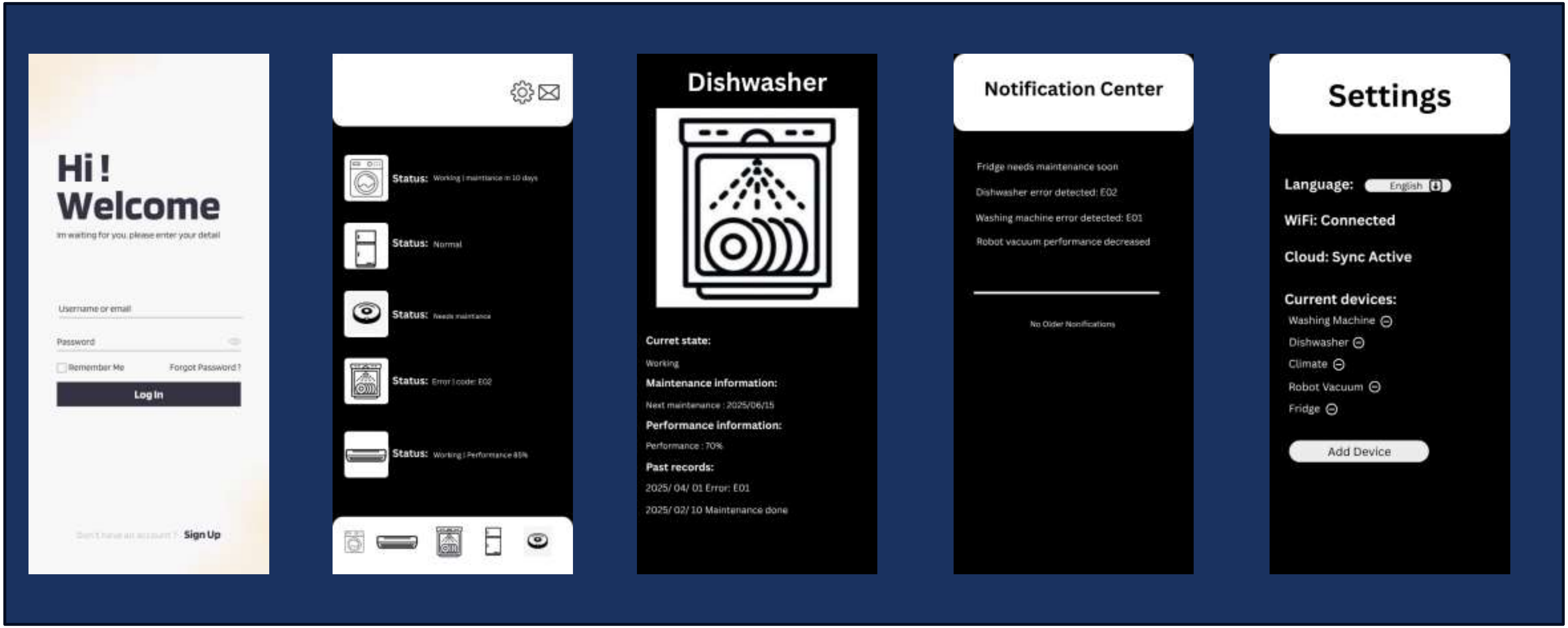
System Layers:

Smart home device layer: collects and analyzes data from devices and identifies error codes.

Cloud-based server layer: stores, analyzes and processes the transmitted data and forwards it to the mobile app.

Mobile Application layer: is the mobile application user interface, providing the user with information about fault information, error codes, device history, performance status and maintenance schedule.

UI DESIGNS



Hi !
Welcome

Im waiting for you, please enter your detail

Username or email

Password

□ Remember Me

Forgot Password ?

Log In

Don't have an account ? Sign Up

Status: Working | maintiance in 10 days

Status: Normal

Status: Needs maintiance

Status: Error | code: E02

Status: Working | Performance 85%

Dishwasher

Curret state:

Working

Maintenance information:

Next maintenance : 2025/06/15

Performance information:

Performance : 70%

Past records:

2025/ 04/ 01 Error: E01

2025/ 02/ 10 Maintenance done

Notification Center

Fridge needs maintenance soon

Dishwasher error detected: E02

Washing machine error detected: E01

Robot vacuum performance decreased

No Older Notifications

Settings

Language: English

WiFi: Connected

Cloud: Sync Active

Current devices:

Washing Machine ⊖

Dishwasher ⊖

Climate ⊖

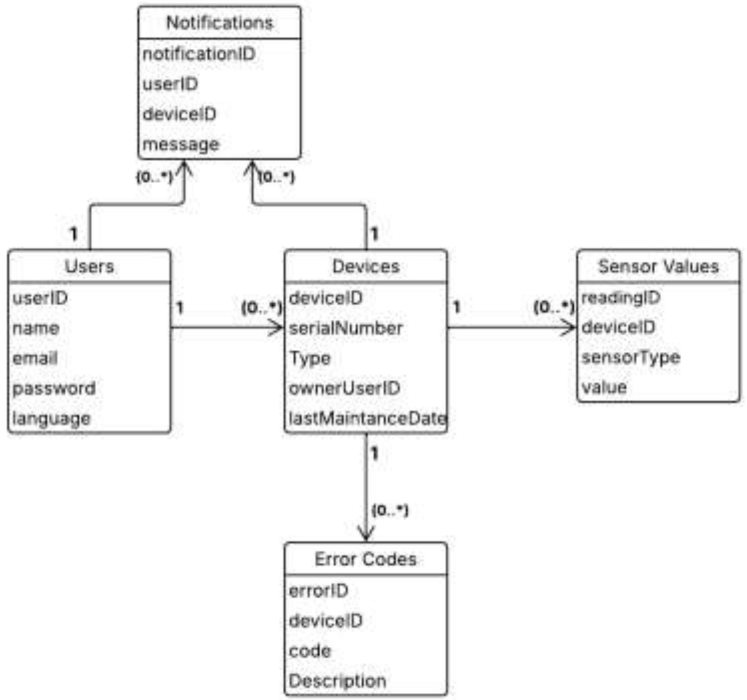
Robot Vacuum ⊖

Fridge ⊖

Add Device

DESIGN

DIAGRAMS:



Notifications
- notificationID
- userID
- deviceID
- message

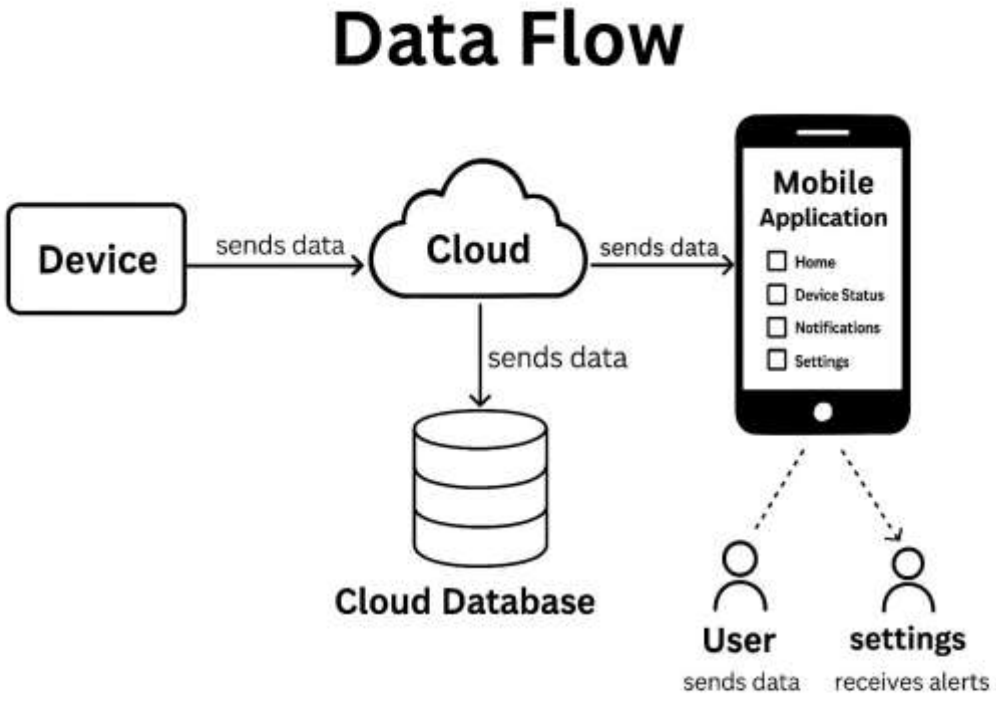
Users
- userID
- name
- email
- password
- language

Devices
- deviceID
- serialNumber
- Type
- ownerUserID
- lastMaintanceDate

Sensor Values
- readingID
- deviceID
- sensorType
- value

Error Codes
- errorID
- deviceID
- code
- Description

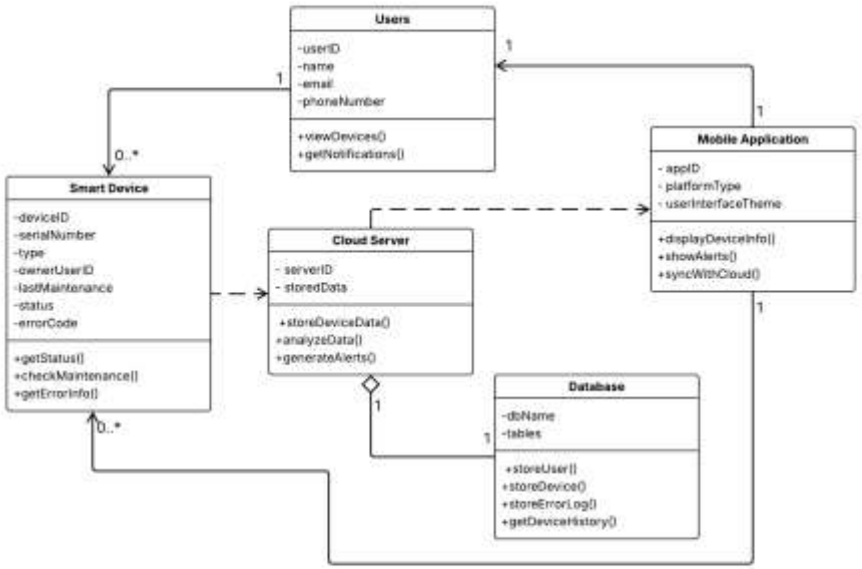
DATABASE DIAGRAM



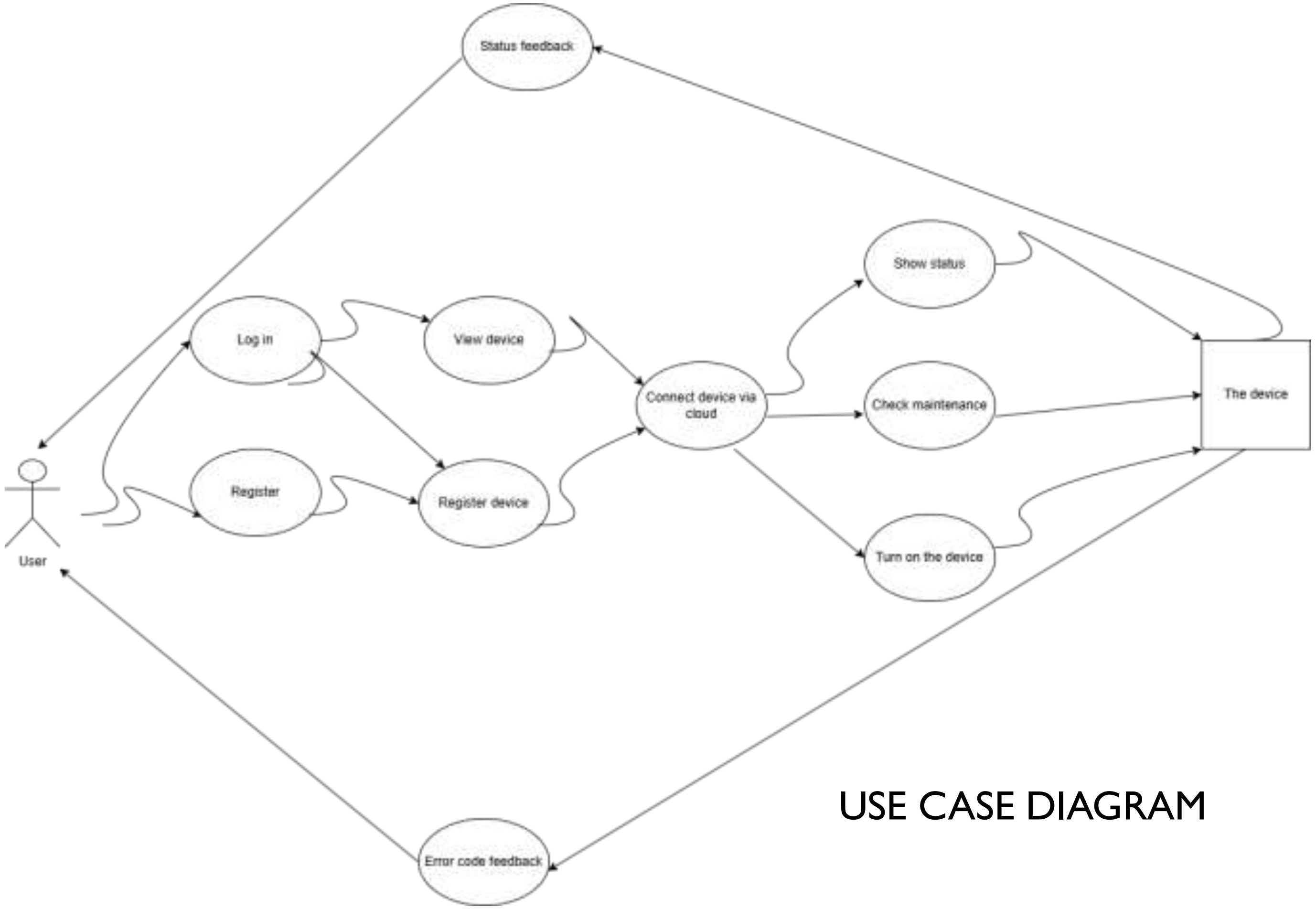
Data Flow

DATAFLOW DIAGRAM

DESIGN



UML CLASS DIAGRAM



USE CASE DIAGRAM

DESIGN

System Dependency:

-Hardware Compatibility

-Network Connections

-Cloud Servers

-Mobile Application

METHODOLOGY

Tools and Technologies Used:

Programming Language : **Python**

Database : Firebase cloud-based data storage and real-time data synchronization

IDE: Thonny IDE is preferred because its interface is simple and understandable.

METHODOLOGY

agile methodology was considered for the project developed, the project was divided into sprints and a dynamic mindset was adopted for each sprint, and it was prepared with team awareness by distributing different tasks to different people.

METHODOLOGY

Some alghorithms examples:

-Getting the error code

-Sending the error code to user

-Check the maintenance

-Showing device status

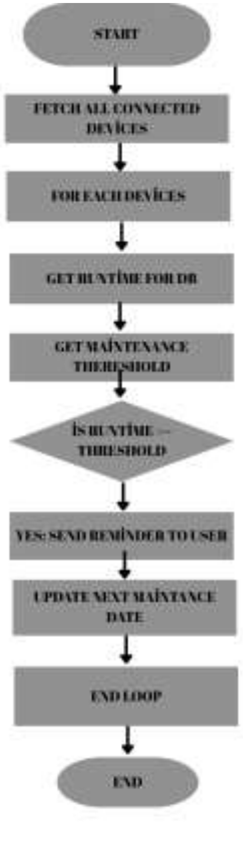
-Water heater control

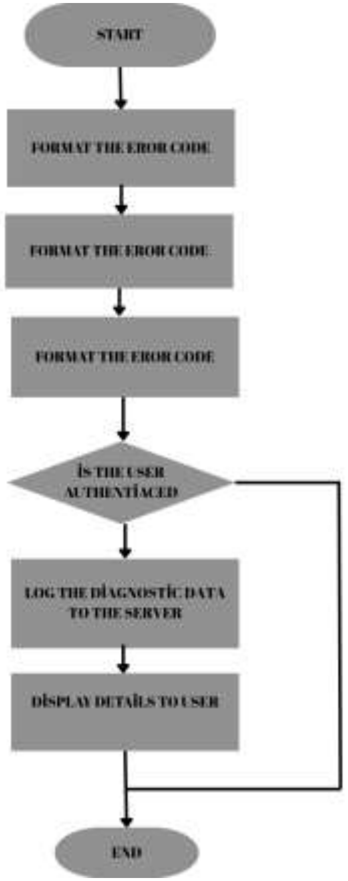
Getting the error code

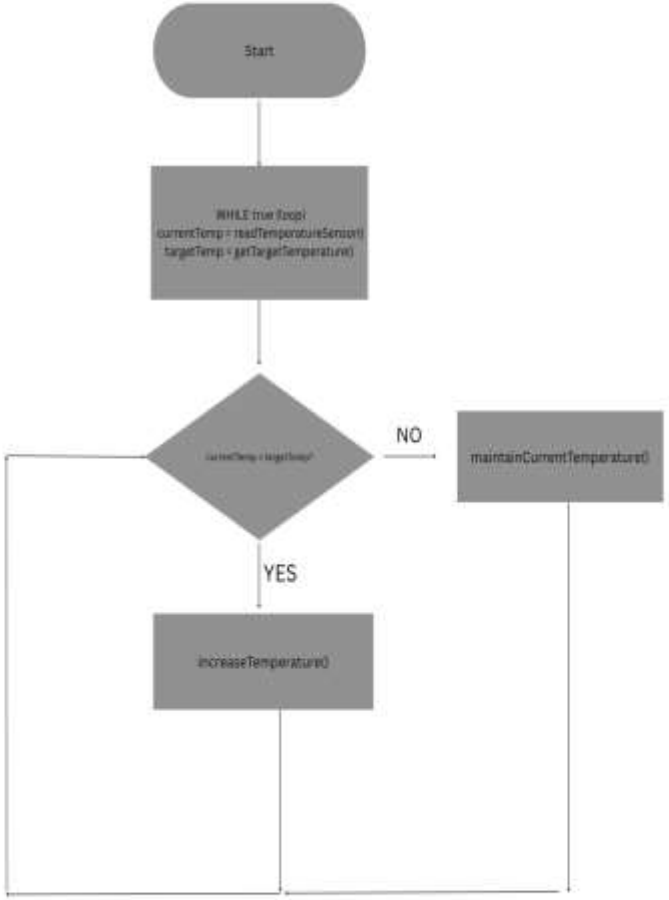
1) algorithm does error checking the moment the application is opened
2) Checks if any device has received an error
3) Send to the database if any error is found
4) If no errors are found, terminate the process

METHODOLOGY

Flowcharts:


START
FETCH ALL CONNECTED DEVİCES
FOR EACH DEVİCES
GET RUNTİME FOR DB
GET MAİNTENANCE THERESHOLD
İS RUNTİME — THRESHOLD
YES: SEND REMİNDER TO USER
UPDATE NEXT MAİNTANCE DATE
END LOOP
END




Start
WHILE true (loop)
currentTemp = readTemperatureSensor()
targetTemp = getTargetTemperature()
currentTemp < targetTemp?
NO
maintainCurrentTemperature()
YES
increaseTemperature()

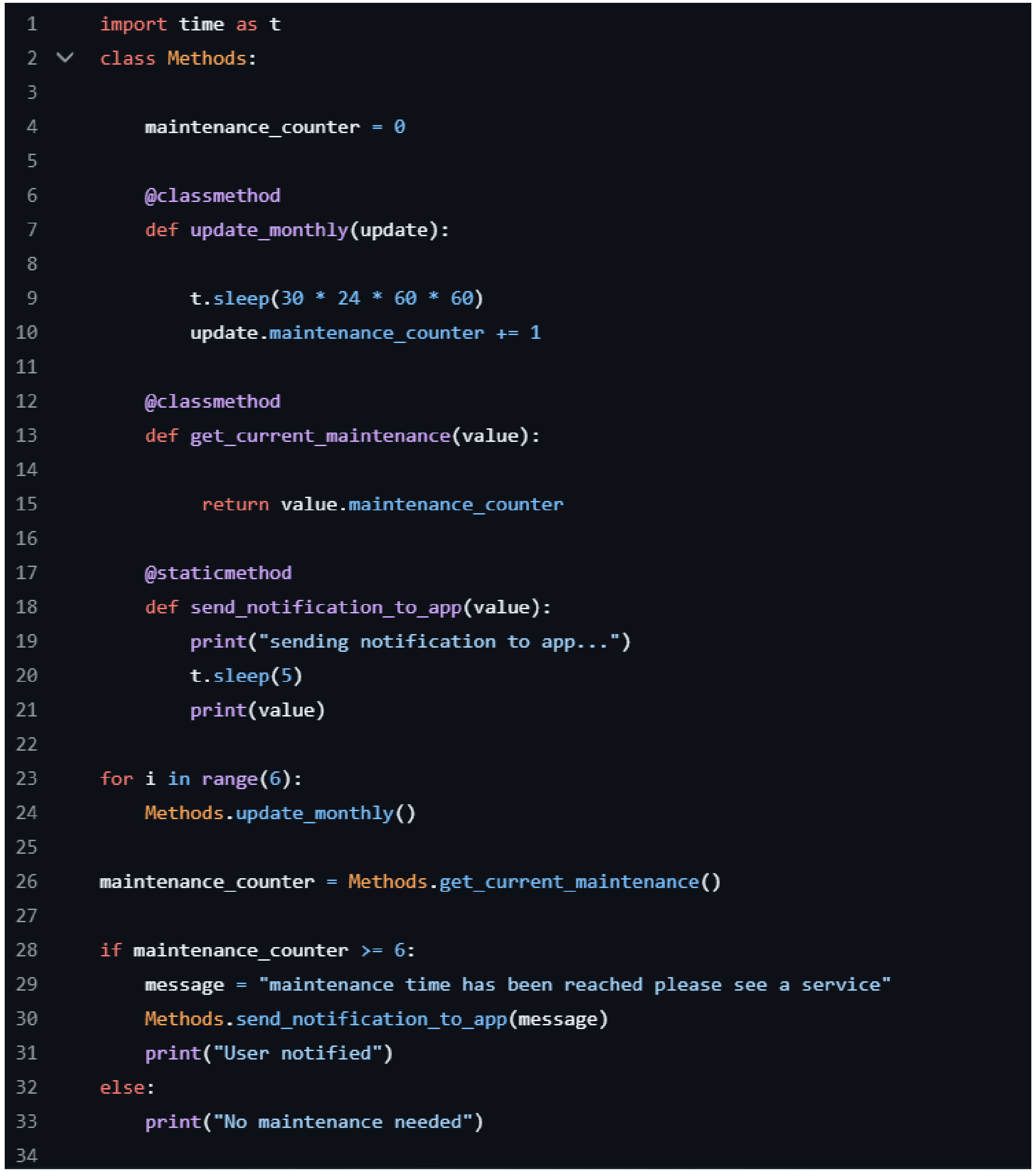
IMPLEMENTATION

First, a data processing module was developed to detect the fault codes of the devices, then a fault code matching algorithm was created to make sense of this data.

Another important part of the application was the calculation of maintenance times. An algorithm was developed that dynamically estimates the maintenance time for each device based on its usage time, fault history and model. Thanks to this algorithm, the maintenance time of the device could be notified to the user via the mobile application.

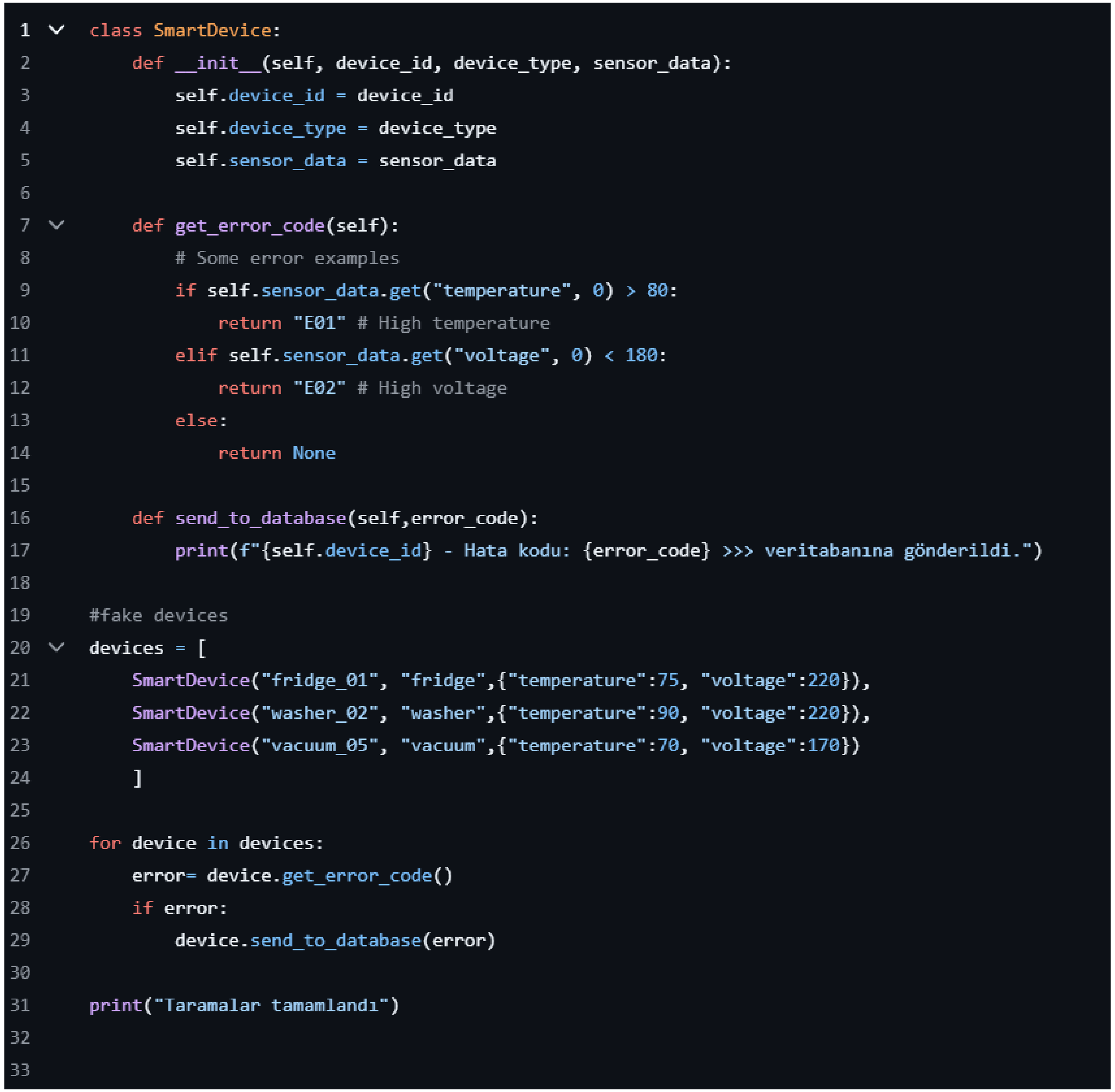
The python codes of the algorithms are given in a simplified form.

NOTE: These codes are draft and do not reflect the actual implementation



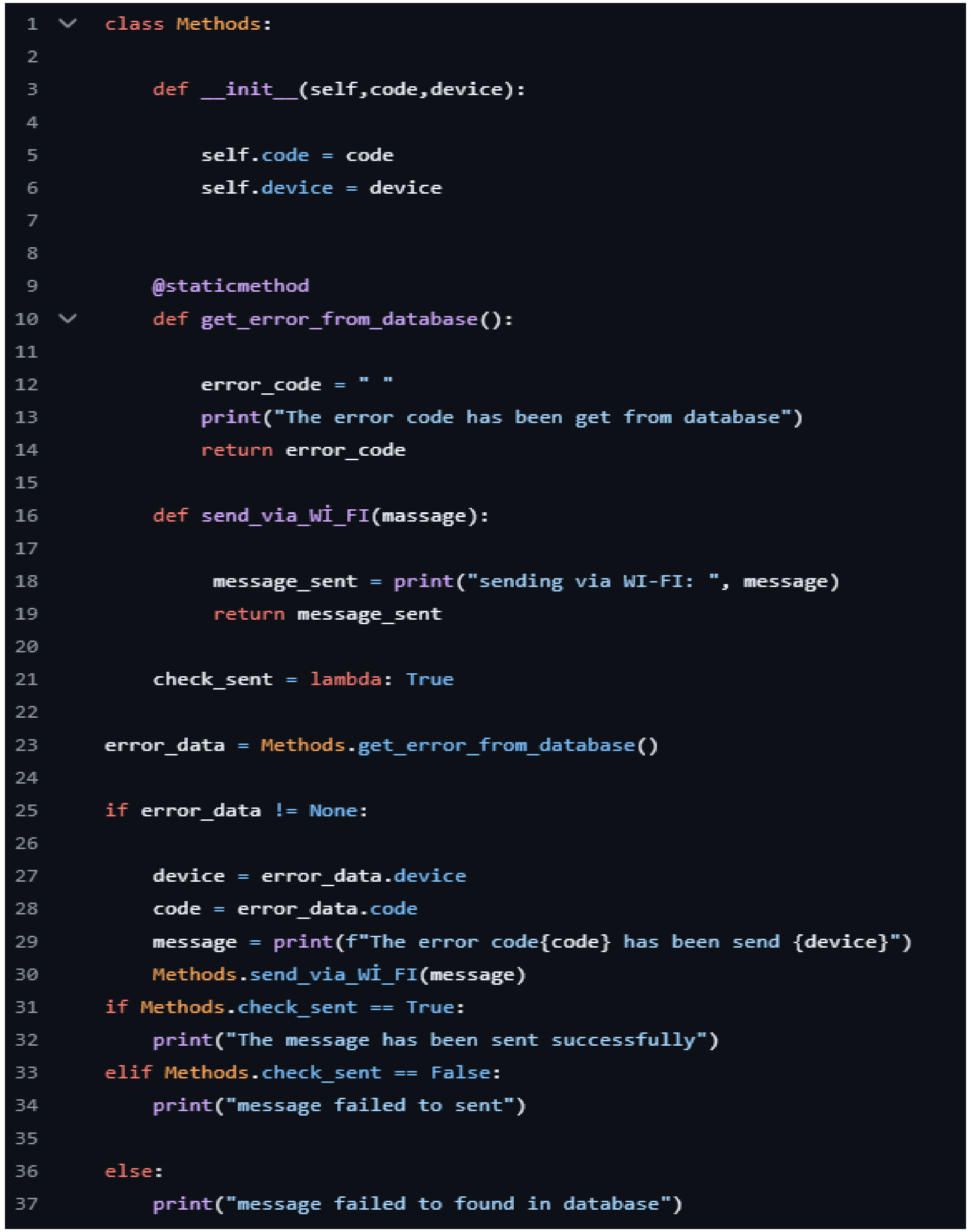
```
import time as t
class Methods:

    maintenance_counter = 0

    @classmethod
    def update_monthly(update):

        t.sleep(30 * 24 * 60 * 60)
        update.maintenance_counter += 1

    @classmethod
    def get_current_maintenance(value):

         return value.maintenance_counter

    @staticmethod
    def send_notification_to_app(value):
        print("sending notification to app...")
        t.sleep(5)
        print(value)

for i in range(6):
    Methods.update_monthly()

maintenance_counter = Methods.get_current_maintenance()

if maintenance_counter >= 6:
    message = "maintenance time has been reached please see a service"
    Methods.send_notification_to_app(message)
    print("User notified")
else:
    print("No maintenance needed")

```

This script allows to check the maintenance date monthly



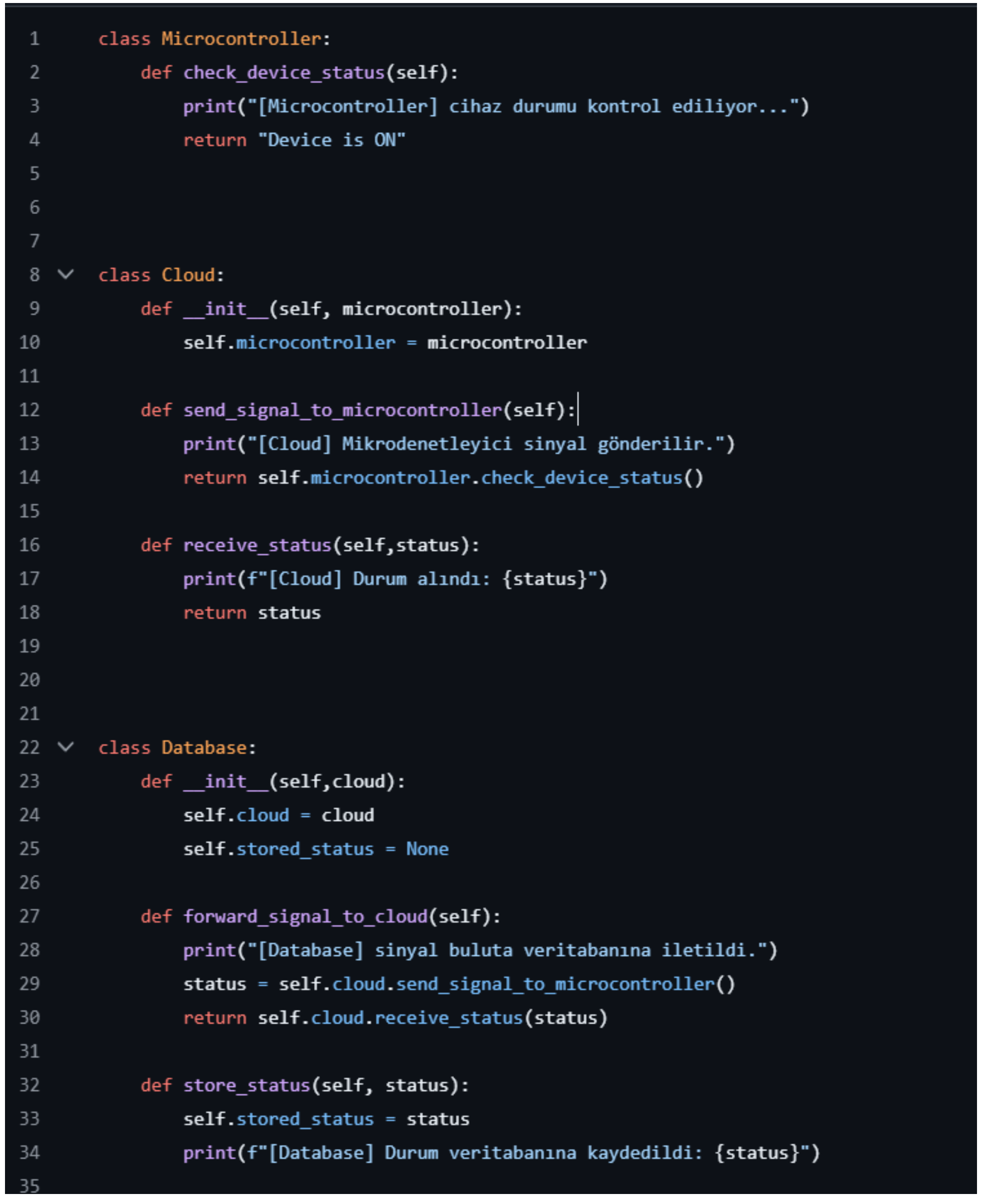
```python
class SmartDevice:
    def __init__(self, device_id, device_type, sensor_data):
        self.device_id = device_id
        self.device_type = device_type
        self.sensor_data = sensor_data

    def get_error_code(self):
        # Some error examples
        if self.sensor_data.get("temperature", 0) > 80:
            return "E01" # High temperature
        elif self.sensor_data.get("voltage", 0) < 180:
            return "E02" # High voltage
        else:
            return None

    def send_to_database(self,error_code):
        print(f"{self.device_id} - Hata kodu: {error_code} >>> veritabanına gönderildi.")

#fake devices
devices = [
    SmartDevice("fridge_01", "fridge",{"temperature":75, "voltage":220}),
    SmartDevice("washer_02", "washer",{"temperature":90, "voltage":220}),
    SmartDevice("vacuum_05", "vacuum",{"temperature":70, "voltage":170})
    ]

for device in devices:
    error= device.get_error_code()
    if error:
        device.send_to_database(error)

print("Taramalar tamamlandı")
```

This script allows to getting error codes

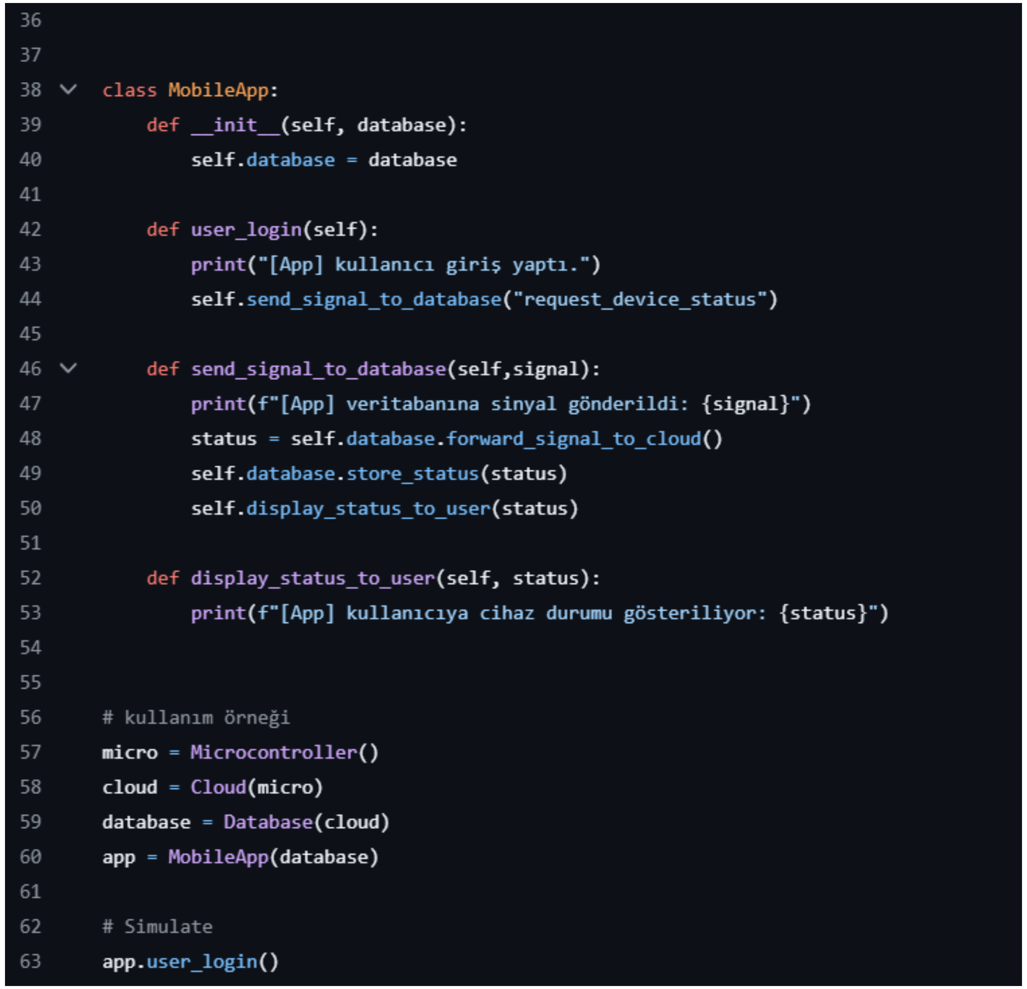


```python
class Methods:

    def __init__(self,code,device):

        self.code = code
        self.device = device


    @staticmethod
    def get_error_from_database():

        error_code = " "
        print("The error code has been get from database")
        return error_code

    def send_via_Wİ_FI(massage):

         message_sent = print("sending via WI-FI: ", message)
         return message_sent

    check_sent = lambda: True

error_data = Methods.get_error_from_database()

if error_data != None:

    device = error_data.device
    code = error_data.code
    message = print(f"The error code{code} has been send {device}")
    Methods.send_via_Wİ_FI(message)
if Methods.check_sent == True:
    print("The message has been sent successfully")
elif Methods.check_sent == False:
    print("message failed to sent")

else:
    print("message failed to found in database")
```

This script allows to sending error codes to user



```python
class Microcontroller:
    def check_device_status(self):
        print("[Microcontroller] cihaz durumu kontrol ediliyor...")
        return "Device is ON"



class Cloud:
    def __init__(self, microcontroller):
        self.microcontroller = microcontroller

    def send_signal_to_microcontroller(self):
        print("[Cloud] Mikrodenetleyici sinyal gönderilir.")
        return self.microcontroller.check_device_status()

    def receive_status(self,status):
        print(f"[Cloud] Durum alındı: {status}")
        return status



class Database:
    def __init__(self,cloud):
        self.cloud = cloud
        self.stored_status = None

    def forward_signal_to_cloud(self):
        print("[Database] sinyal buluta veritabanına iletildi.")
        status = self.cloud.send_signal_to_microcontroller()
        return self.cloud.receive_status(status)

    def store_status(self, status):
        self.stored_status = status
        print(f"[Database] Durum veritabanına kaydedildi: {status}")

```



```python


class MobileApp:
    def __init__(self, database):
        self.database = database

    def user_login(self):
        print("[App] kullanıcı giriş yaptı.")
        self.send_signal_to_database("request_device_status")

    def send_signal_to_database(self,signal):
        print(f"[App] veritabanına sinyal gönderildi: {signal}")
        status = self.database.forward_signal_to_cloud()
        self.database.store_status(status)
        self.display_status_to_user(status)

    def display_status_to_user(self, status):
        print(f"[App] kullanıcıya cihaz durumu gösteriliyor: {status}")


# kullanım örneği
micro = Microcontroller()
cloud = Cloud(micro)
database = Database(cloud)
app = MobileApp(database)

# Simulate
app.user_login()
```

This script allows to show device status

TEST PROCESS

The mobile application developed in the project was tested for basic functions such as real-time data retrieval, error code processing and maintenance time calculation. This testing process was carried out to increase the stability, reliability and user experience of the software and to eliminate potential errors.

-Unit testing:

Unit tests were performed to see if each module worked correctly individually.

-System Testing:

All components of the system were assembled and tested end-to-end.

-Security Testing:

Login, user authentication and database access permissions were checked on Firebase for data security.

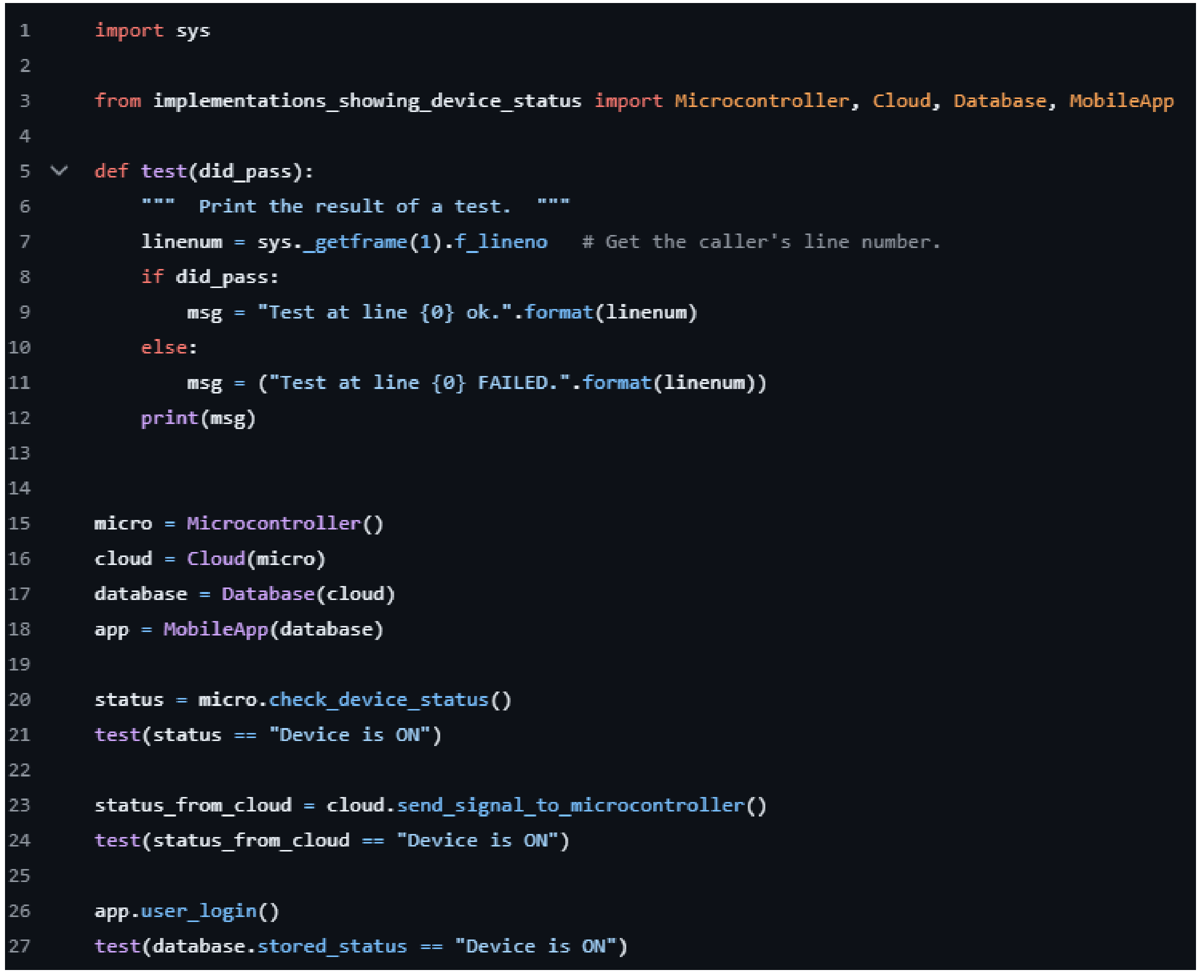
-Acceptance Test:

User scenarios were created to test whether the application is functional for the end user.

TEST PROCESS

Example of unittes script

this code does unit testing and checks that the software works correctly.



```python
import sys

from implementations_showing_device_status import Microcontroller, Cloud, Database, MobileApp

def test(did_pass):
    """  Print the result of a test.  """
    linenum = sys._getframe(1).f_lineno   # Get the caller's line number.
    if did_pass:
        msg = "Test at line {0} ok.".format(linenum)
    else:
        msg = ("Test at line {0} FAILED.".format(linenum))
    print(msg)


micro = Microcontroller()
cloud = Cloud(micro)
database = Database(cloud)
app = MobileApp(database)

status = micro.check_device_status()
test(status == "Device is ON")

status_from_cloud = cloud.send_signal_to_microcontroller()
test(status_from_cloud == "Device is ON")

app.user_login()
test(database.stored_status == "Device is ON")
```

DEPLOYMENT AND MAINTENANCE

Deployment:

After the completion of this project, the software is not only intended to be published, but also to be easy, efficient and functional for the users. Several steps were taken to achieve this goal:

-User Guides

-Education Support

-On-Site Support

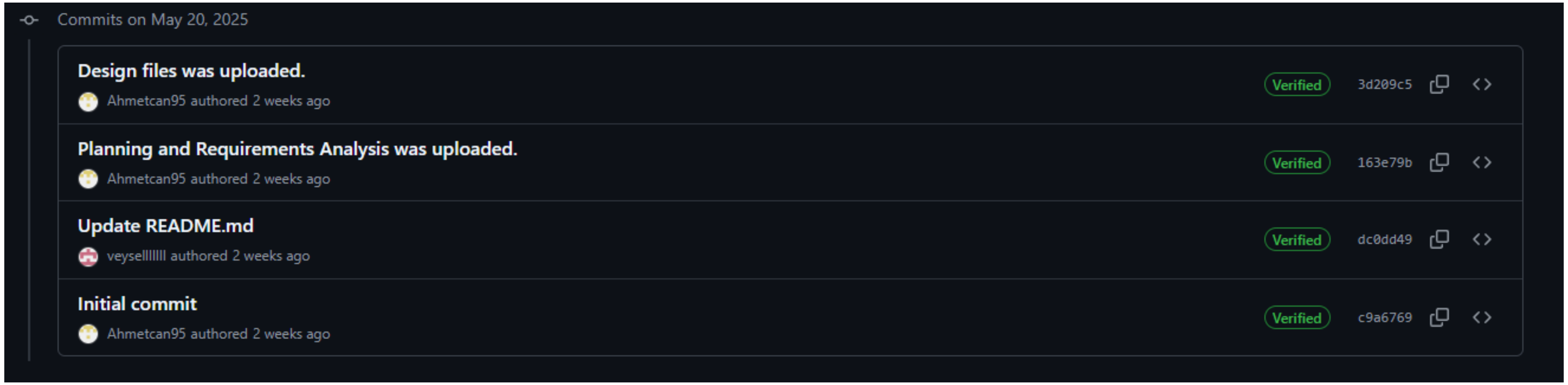
Maintenance:

It is designed to ensure that the system can be easily adapted to new software during the maintenance process without interruption in the long term. Maintenance has several critical points:

-Updates and Bug Fixes

-User Online Support

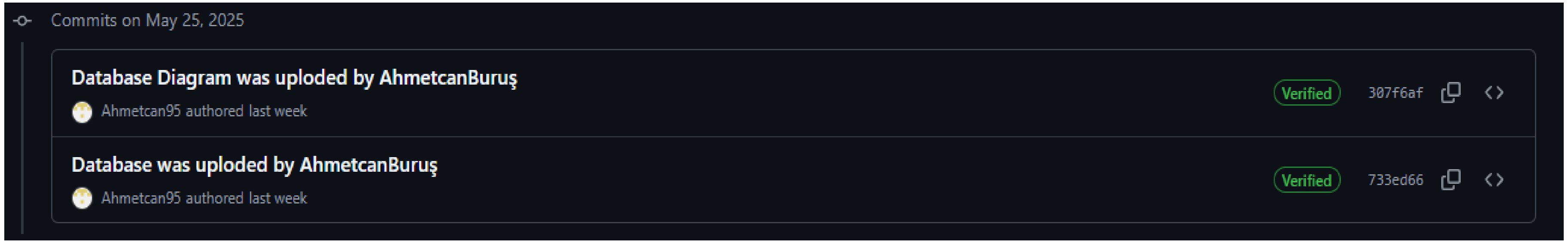
GITHUB LOGS



Commits on May 20, 2025

- **Design files was uploaded.**
  Ahmetcan95 authored 2 weeks ago — Verified — 3d209c5
- **Planning and Requirements Analysis was uploaded.**
  Ahmetcan95 authored 2 weeks ago — Verified — 163e79b
- **Update README.md**
  veysellllllll authored 2 weeks ago — Verified — dc0dd49
- **Initial commit**
  Ahmetcan95 authored 2 weeks ago — Verified — c9a6769

May 20, 2025

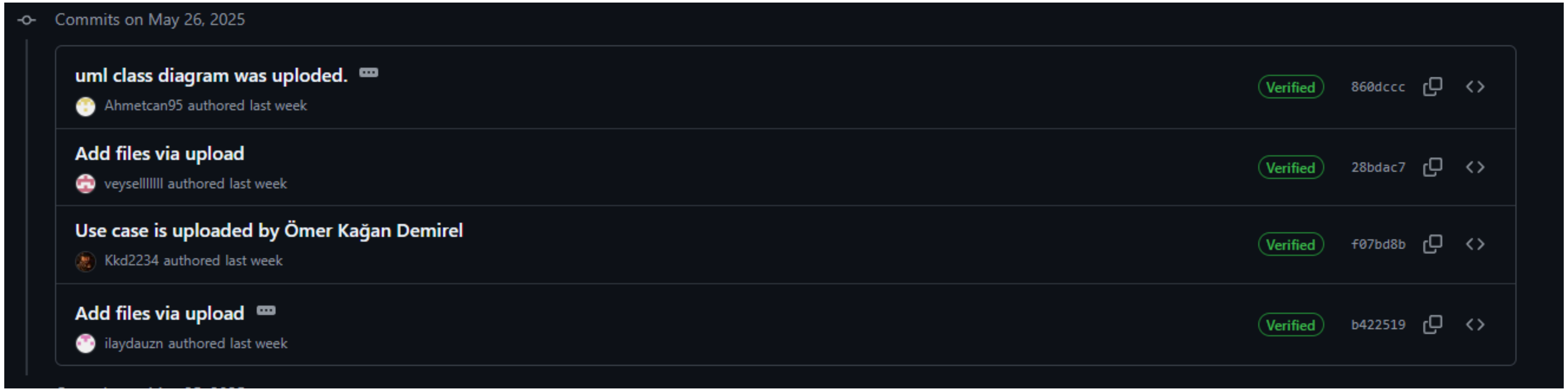


Commits on May 25, 2025

- **Database Diagram was uploded by AhmetcanBuruş**
  Ahmetcan95 authored last week — Verified — 307f6af
- **Database was uploded by AhmetcanBuruş**
  Ahmetcan95 authored last week — Verified — 733ed66

May 25, 2025

May 26, 2025



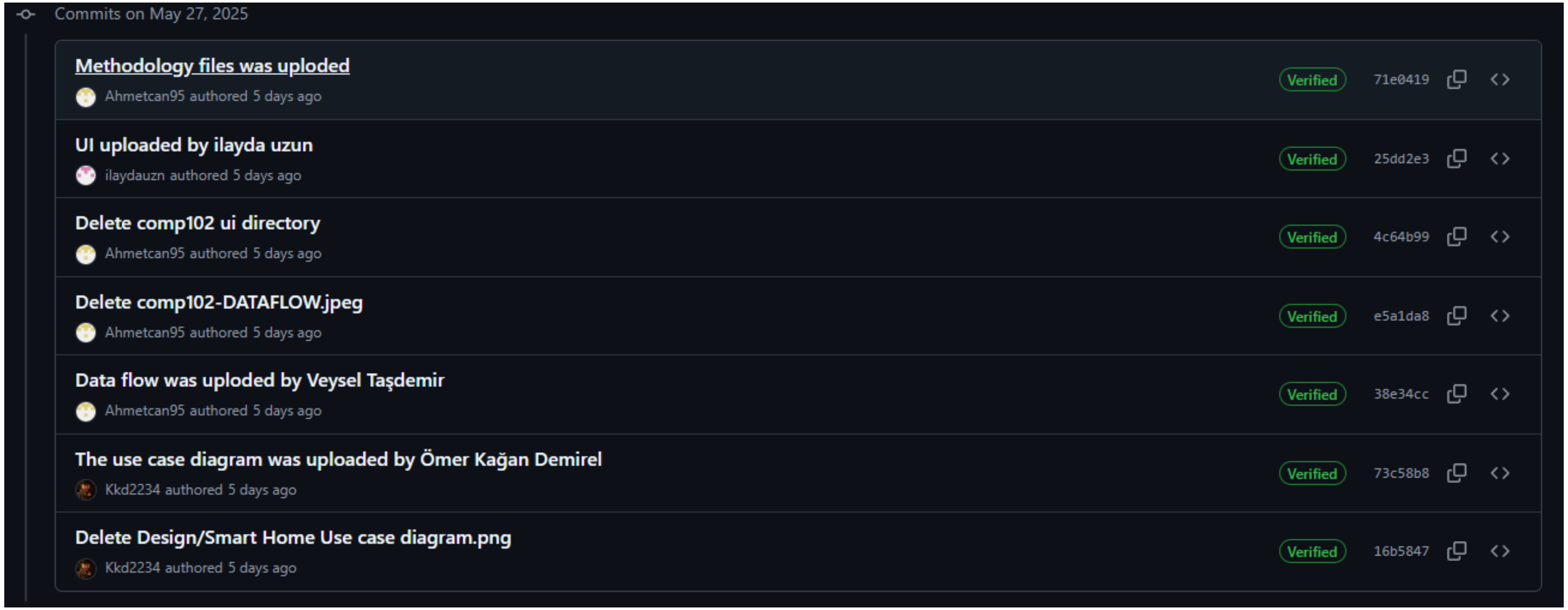
Commits on May 26, 2025

uml class diagram was uploded.
Ahmetcan95 authored last week
Verified 860dccc

Add files via upload
veysellllll authored last week
Verified 28bdac7

Use case is uploaded by Ömer Kağan Demirel
Kkd2234 authored last week
Verified f07bd8b

Add files via upload
ilaydauzn authored last week
Verified b422519



Commits on May 27, 2025

Methodology files was uploded
Ahmetcan95 authored 5 days ago
Verified 71e0419

UI uploaded by ilayda uzun
ilaydauzn authored 5 days ago
Verified 25dd2e3

Delete comp102 ui directory
Ahmetcan95 authored 5 days ago
Verified 4c64b99

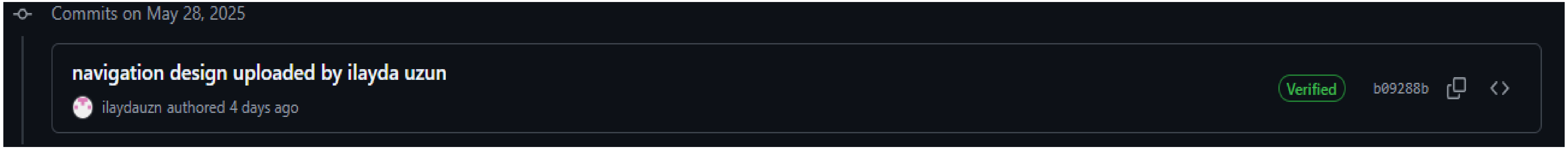
Delete comp102-DATAFLOW.jpeg
Ahmetcan95 authored 5 days ago
Verified e5a1da8

Data flow was uploded by Veysel Taşdemir
Ahmetcan95 authored 5 days ago
Verified 38e34cc

The use case diagram was uploaded by Ömer Kağan Demirel
Kkd2234 authored 5 days ago
Verified 73c58b8

Delete Design/Smart Home Use case diagram.png
Kkd2234 authored 5 days ago
Verified 16b5847

May 27, 2025



Commits on May 28, 2025

navigation design uploaded by ilayda uzun

ilaydauzn authored 4 days ago

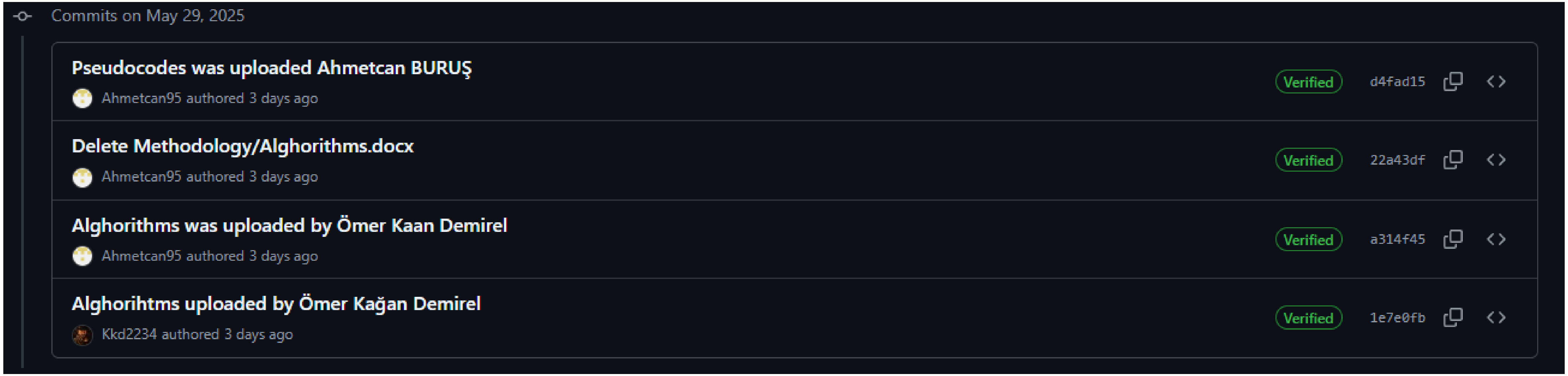
Verified b09288b

▲

May 28, 2025

May 29, 2025

▼



Commits on May 29, 2025

Pseudocodes was uploaded Ahmetcan BURUŞ

Ahmetcan95 authored 3 days ago

Verified d4fad15

Delete Methodology/Alghorithms.docx

Ahmetcan95 authored 3 days ago

Verified 22a43df

Alghorithms was uploaded by Ömer Kaan Demirel

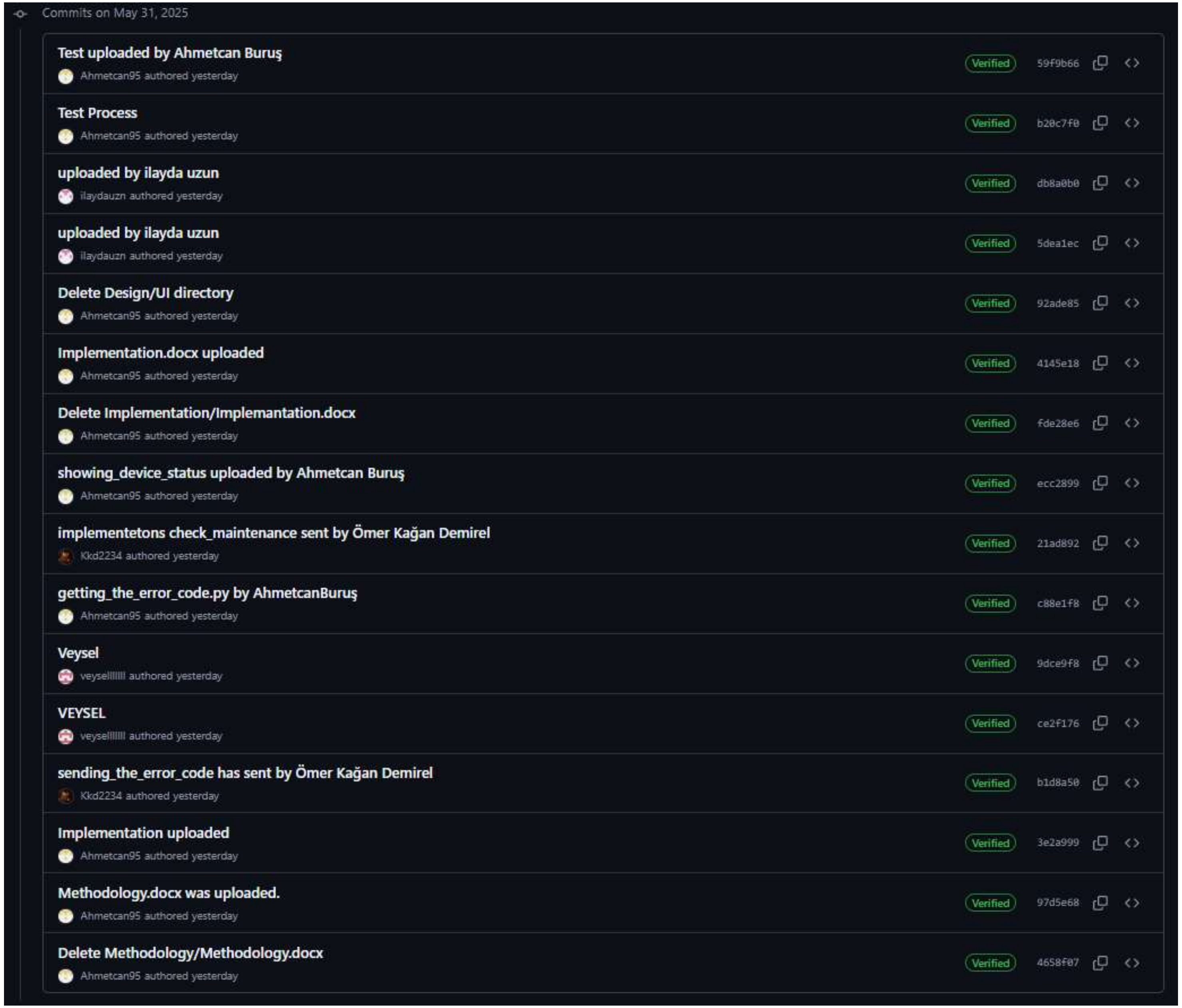
Ahmetcan95 authored 3 days ago

Verified a314f45

Alghorihtms uploaded by Ömer Kağan Demirel

Kkd2234 authored 3 days ago

Verified 1e7e0fb



Commits on May 31, 2025

Test uploaded by Ahmetcan Buruş
Ahmetcan95 authored yesterday
Verified 59f9b66

Test Process
Ahmetcan95 authored yesterday
Verified b20c7f0

uploaded by ilayda uzun
ilaydauzn authored yesterday
Verified db8a0b0

uploaded by ilayda uzun
ilaydauzn authored yesterday
Verified 5dea1ec

Delete Design/UI directory
Ahmetcan95 authored yesterday
Verified 92ade85

Implementation.docx uploaded
Ahmetcan95 authored yesterday
Verified 4145e18

Delete Implementation/Implemantation.docx
Ahmetcan95 authored yesterday
Verified fde28e6

showing_device_status uploaded by Ahmetcan Buruş
Ahmetcan95 authored yesterday
Verified ecc2899

implementetons check_maintenance sent by Ömer Kağan Demirel
Kkd2234 authored yesterday
Verified 21ad892

getting_the_error_code.py by AhmetcanBuruş
Ahmetcan95 authored yesterday
Verified c88e1f8

Veysel
veysellllll authored yesterday
Verified 9dce9f8

VEYSEL
veysellllll authored yesterday
Verified ce2f176

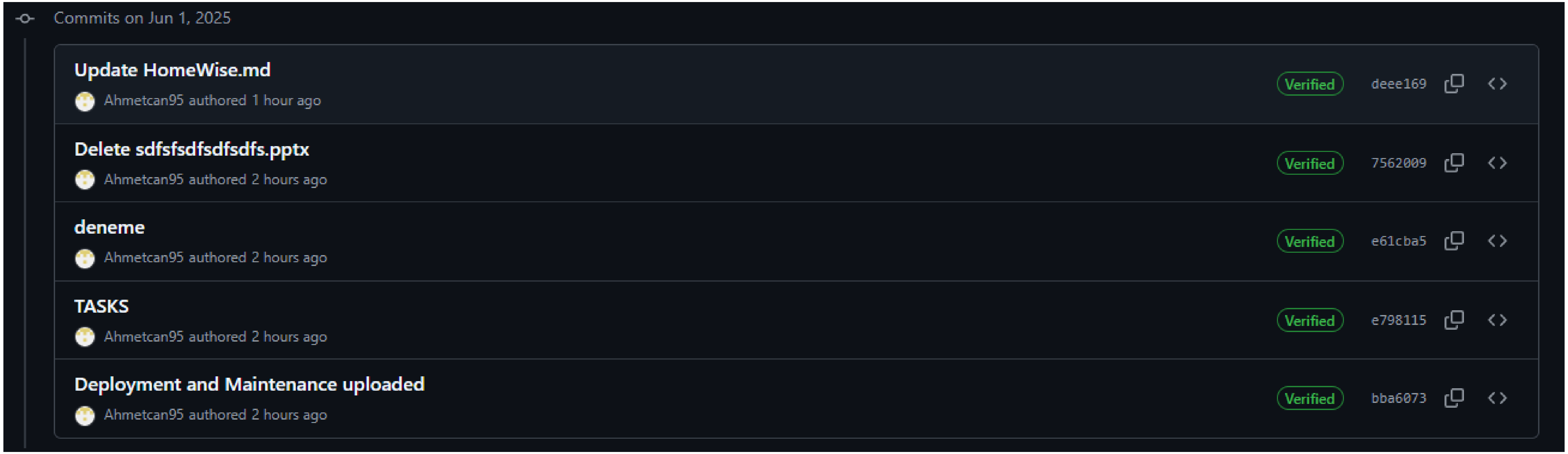
sending_the_error_code has sent by Ömer Kağan Demirel
Kkd2234 authored yesterday
Verified b1d8a50

Implementation uploaded
Ahmetcan95 authored yesterday
Verified 3e2a999

Methodology.docx was uploaded.
Ahmetcan95 authored yesterday
Verified 97d5e68

Delete Methodology/Methodology.docx
Ahmetcan95 authored yesterday
Verified 4658f07

May 31, 2025



Commits on Jun 1, 2025

Update HomeWise.md
Ahmetcan95 authored 1 hour ago
Verified deee169

Delete sdfsfsdfsdfsdfs.pptx
Ahmetcan95 authored 2 hours ago
Verified 7562009

deneme
Ahmetcan95 authored 2 hours ago
Verified e61cba5

TASKS
Ahmetcan95 authored 2 hours ago
Verified e798115

Deployment and Maintenance uploaded
Ahmetcan95 authored 2 hours ago
Verified bba6073

Jun 1, 2025

THANK YOU FOR LISTENING

Prepared by

-Ahmetcan Buruş
-Ömer Kaan Demirel
-İlayda Uzun
-Veysel Taşdemir