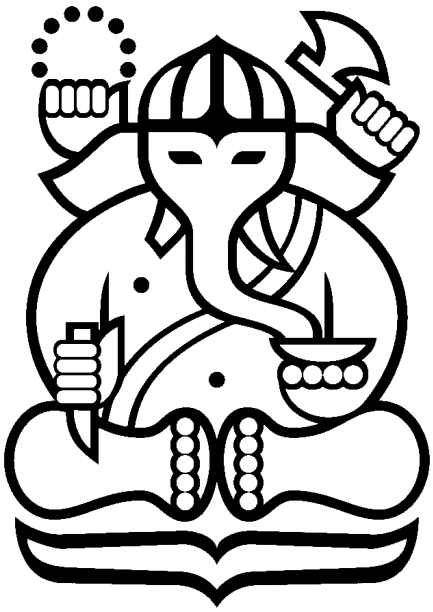
TUGAS BESAR
II3160 TENOLOGI SISTEM TERINTEGRASI
GREENGROW FINAL REPORT



Disusun oleh:

Muhammad Reffy Haykal 18222103

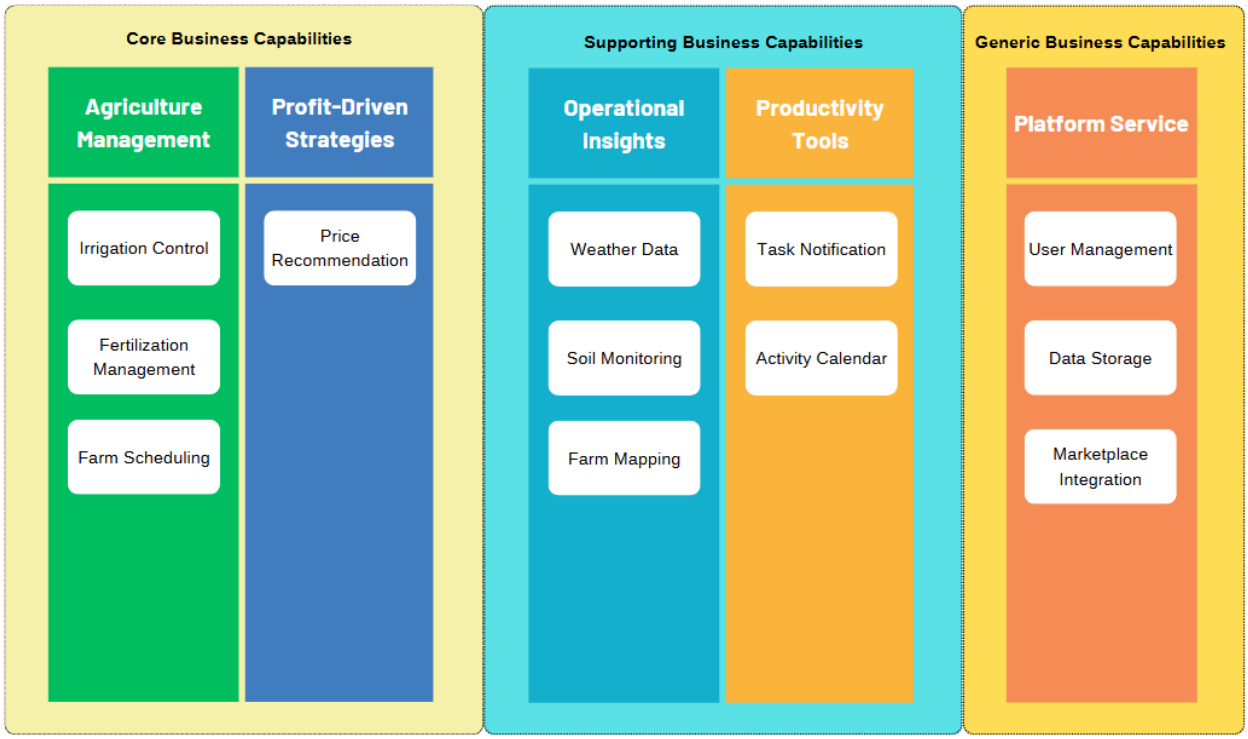
PROGRAM STUDI SISTEM DAN TEKNOLOGI INFORMASI
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2024

Daftar Isi

1. Business Capability Map

Berikut merupakan *Business Capability Map* yang telah GREENGROW.



Gambar 1.1 Business Capability Map

Core Business Capabilities mencakup kemampuan inti yang mendukung pengelolaan pertanian secara optimal dan strategi keuntungan. Dalam **Agriculture Management**, terdapat beberapa kemampuan utama yaitu *Irrigation Control* untuk memastikan pengairan yang efisien, *Fertilization Management* untuk merencanakan dan menjadwalkan pemupukan, *Farm Scheduling* untuk menentukan jadwal terbaik bagi aktivitas pertanian. Selain itu, **Profit-Driven Strategies** mencakup *Price Recommendation*, untuk memberikan rekomendasi harga jual yang menguntungkan berdasarkan analisis modal dan pasar.

Supporting Business Capabilities berfungsi sebagai pendukung operasi dan produktivitas. Dalam kategori **Operational Insights**, terdapat *Weather Data* untuk memberikan informasi cuaca yang relevan, *Soil Monitoring* untuk memantau kelembapan dan pH tanah, serta *Farm Mapping* untuk memetakan lahan pertanian. Pada kategori **Productivity Tools**, terdapat *Task Notifications* untuk memberikan pengingat terkait aktivitas pertanian, serta *Activity Calendar* untuk membantu memvisualisasikan jadwal kegiatan.

Generic Business Capabilities menyediakan layanan umum yang dibutuhkan oleh platform. Dalam kategori **Platform Services**, terdapat *User Management* untuk mengelola pengguna aplikasi, *Data Storage Management* untuk menyimpan dan mengelola data pertanian dengan aman, serta *Marketplace Integration* untuk mendukung transaksi, penjualan, dan sistem

pembayaran. Semua kemampuan ini bersama-sama memastikan bahwa sistem manajemen pertanian dapat berjalan secara efektif, efisien, dan mendukung keberlanjutan bisnis petani.

2. Domains: Core, Supporting, Generic

2.1 Core Domain

- Irrigation Control: Mengelola dan mengoptimalkan penyiraman lahan.
    - Watering Schedule: Menjadwalkan penyiraman baik secara otomatis ataupun manual. Dipengaruhi juga oleh kelembapan tanah dan prakiraan cuaca.
    - Water Usage Optimization: Bagi tempat yang sulit air, fungsi ini mengoptimalkan penggunaan air agar lebih efisien dan mengurangi pemborosan air
- Fertilization Management: Mengelola pemupukan secara efisien.
    - Fertilization Recommendation: Memberikan rekomendasi pupuk yang ideal berdasarkan kondisi tanah dan faktor cuaca serta pertimbangan pemakaian pilihan pupuk organik/anorganik.
    - Fertilizer Calculator: Menghitung jumlah dan jenis pupuk yang diperlukan berdasarkan input seperti jenis tanaman, luas lahan, jenis tanah, dan pupuk yang digunakan.
    - Fertilization Schedule: Menyusun jadwal secara otomatis atau manual berdasarkan data. Dipengaruhi oleh rekomendasi pupuk dan prakiraan cuaca.
- Farm Scheduling: Menentukan kapan waktu optimal untuk pembibitan dan panen berdasarkan aktivitas pertanian yang telah dilakukan (pengairan, pemupukan, kondisi tanah, dan cuaca).
- Price recommendation: Memberikan rekomendasi harga jual dari kalkulasi modal yang menjamin petani tidak rugi.

2.2 Supporting Domain

- Farm Mapping: Memetakan lahan pertanian yang berhuna untuk membantu core domain dalam melakukan perhitungan-perhitungannya. Dapat menggunakan GPS dan supaya lebih akurat disediakan manual juga.
- Weather Data: Menyediakan data cuaca *real-time* serta prakiraan cuaca (*forecast)* yang dapat membantu core domain dalam melakukan perhitungan-perhitungannya. (Real-time and Forecast)
- Notification: Memberikan notifikasi cuaca, penyiraman, pemupukan, panen, dan lain-lain.
- Monitor Soil Condition: Menggunakan sensor tanah, memantau kondisi kelembaban tanah. Melalui sensor tanah tersebut, dapat mengetahui PH tanah dan memberikan data yang berguna untuk core domain yang lain.

- Calendar: Membantu dalam visualisasi tanggal untuk pembibitan, pengairan, pemupukan, pembibitan, dan pemanenan.

2.3 Generic Domain

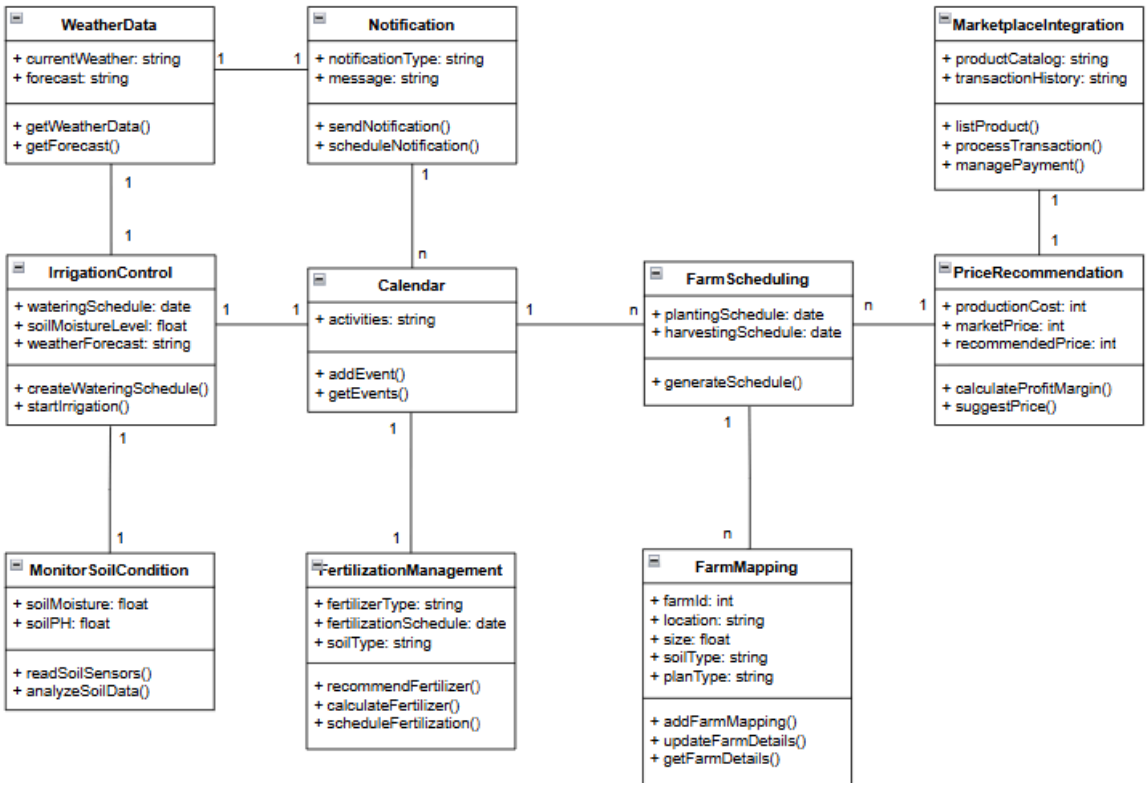
- User Management: Mengelola fitur aplikasi yang berhubungan dengan akun. Seperti registrasi, *log-in,* autentikasi, dan kontrol akun.
- Data storage: Penyimpanan dan pengelolaan data yang digunakan oleh aplikasi termasuk data lahan, tanah, cuaca, hasil pertanian, transaksi, dan lain-lain. Begitu juga untuk memastikan data dapat diakses serta melakukan *backup and recovery*.
- Marketplace Integration: Terdapat penjualan produk, harga jual, dan pembayaran.

3. Arsitektur Aplikasi dan Implementasi

3.1 Model Sistem

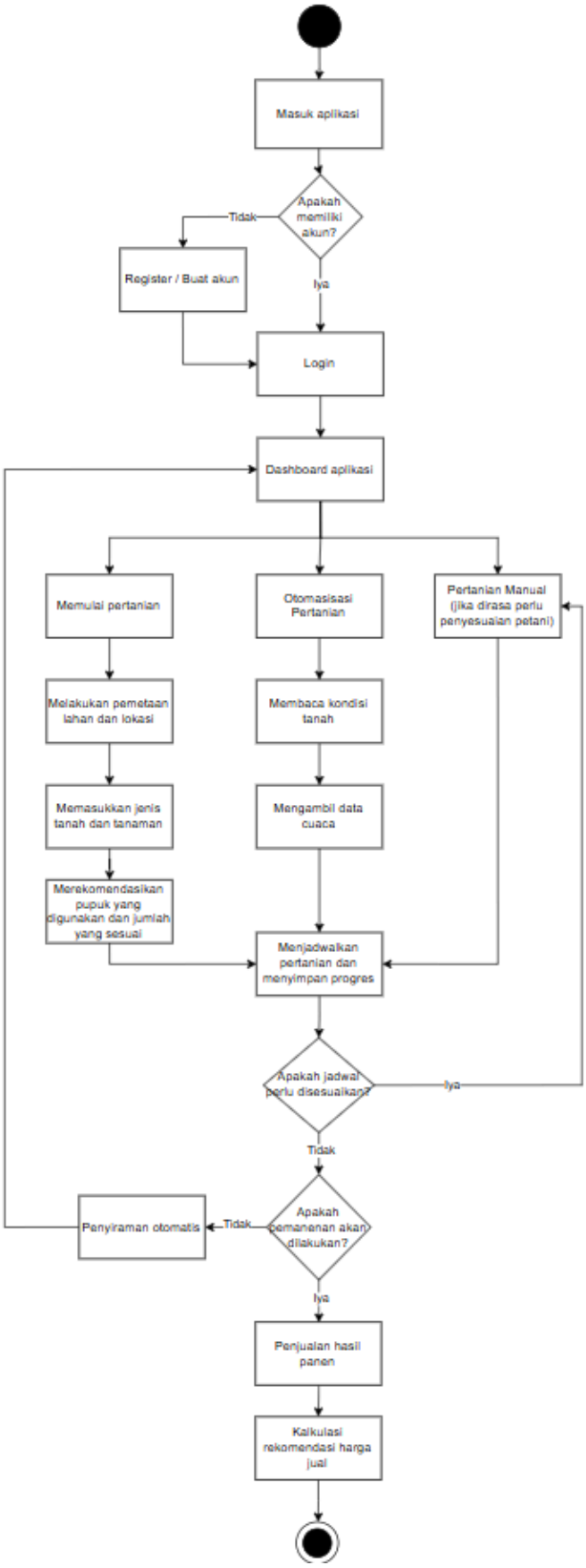
Model sistem dibagi menjadi dua, yaitu struktural dan behavioral. Untuk struktural, saya menggunakan *class diagram* sedangkan untuk behavioral saya menggunakan Berikut merupakan visualisasi dari kedua model tersebut.

3.1.1 Class Diagram



Gambar 3.1 Class Diagram

3.1.2 Activity Diagram



Gambar 3.2 Activity Diagram

3.2 Struktur Aplikasi

Berikut adalah struktur aplikasi dari GREENGROW.

```
main.py
database/
    __init__.py
    user_db.py
    farm_db.py
    schedule_db.py
routes/
    __init__.py
    user_routes.py
    farm_routes.py
    weather_routes.py
    schedule_routes.py
models/
    __init__.py
    user_model.py
    farm_model.py
    weather_model.py
    schedule_model.py
services/
    __init__.py
    user_service.py
    farm_service.py
    weather_service.py
    schedule_service.py
utils/
    __init__.py
    notification_utils.py
    calendar_utils.py
    gps_utils.py
```

Berikut ini adalah penjelasan untuk tiap folder dan file.

1. **main.py**
   File utama untuk menjalankan aplikasi, memuat semua konfigurasi awal (server, middleware, dll.).
2. **database/**
   Folder ini berisi koneksi database dan operasi CRUD.
   a. user_db.py: Operasi CRUD untuk data pengguna.

b. farm_db.py: Operasi CRUD untuk data lahan pertanian.
c. weather_db.py: Penyimpanan data cuaca (dari API).
d. schedule_db.py: Penyimpanan jadwal pertanian

3. **routes/**

   Folder ini mengatur endpoint API.

   a. user_routes.py: Endpoint untuk login, registrasi pengguna.
   b. farm_routes.py: Endpoint untuk memetakan lahan, input data pertanian (Maps).
   c. weather_routes.py: Endpoint untuk mendapatkan data cuaca.
   d. schedule_db.py: Endpoint untuk jadwal pertanian otomatis.

4. **models/**

   Berisi model data (representasi tabel database dalam kode).

   a. user_model.py: Model pengguna (user_id, nama, role).
   b. farm_model.py: Model lahan pertanian (farm_id, lokasi, ukuran, dsb.).
   c. weather_model.py: Model data cuaca.
   d. schedule_model.py: Model jadwal pertanian.

5. **services/**

   Berisi logika bisnis utamaa.

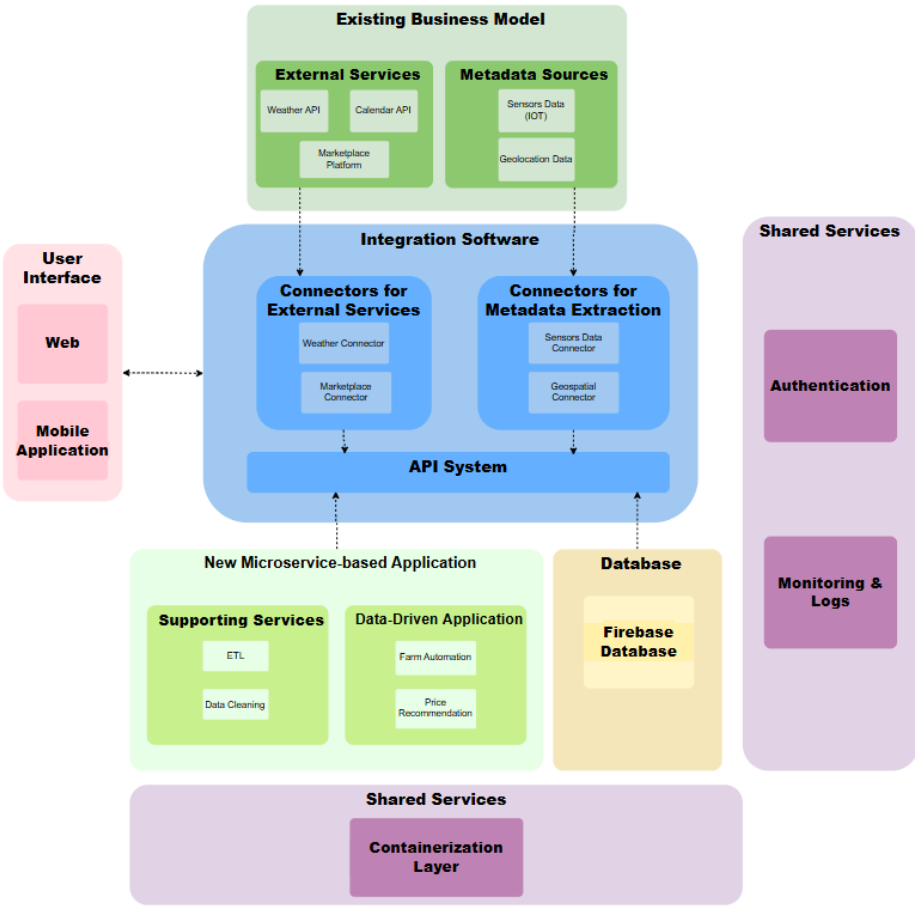
   a. user_service.py: Validasi login, hashing password.
   b. farm_service.py: Kalkulasi pemupukan atau irigasi otomatis.
   c. weather_service.py: Menghubungkan API IoT atau cuaca.
   d. schedule_service.py: Algoritma untuk jadwal pertanian.

6. **utils/**

   Folder untuk fungsi pendukung.

   a. notification_utils.py: Mengatur pengiriman notifikasi.
   b. calendar_utils.py: Sinkronisasi dengan kalender.
   c. gps_utils.py: Mendapatkan lokasi GPS atau data geolokasi.

3.3 Arsitektur Sistem



Gambar 4.1 Struktur Aplikasi

Berikut merupakan penjelasan dari struktur aplikasi GREENGROW.

1. **User Interface**
    a. **Web:** Akses bagi petani untuk mengelola lahan, melihat jadwal, dan menerima notifikasi.
    b. **Mobile:** Aplikasi petani dengan fitur untuk memantau kondisi tanah, cuaca, dan otomatisasi aktivitas.
2. **Existing Business Application**
    a. **External Services**
        - Calendar API: Menampilkan jadwal pertanian.
        - Weather API: Mendapatkan data prakiraan cuaca.
        - Marketplace Platform: Untuk integrasi harga jual hasil panen.
    b. **Metadata Sources**
        - Sensor Data: Data dari perangkat IoT (kondisi tanah).
        - Geolocation Data: Data pemetaan lahan pertanian.
3. **Integration Software**

a. **Connectors for External Services**
    - Weather Connection: Integrasi ke API cuaca.
    - Marketplace Connector: Integrasi ke platform e-commerce.
b. **Connectors for Metadata Extraction**
    - Sensor Data Connector: Mendapatkan data *real-time* dari perangkat IOT.
    - Geospatial Connector: Mengintegrasikan data lokasi untuk peta lahan.
c. **System API**
    API terpusat yang menghubungkan User Interface dengan Core System.

4. **New Microservice-Based Application**
    a. **Supporting Services**
        - ETL: Mengelola ekstraksi, transformasi, dan loading data dari berbagai sumber.
        - Data Cleaning: Validasi data cuaca, tanah, dan sensor.
    b. **Data-Driven Application**
        - Farm Automation: Menjadwalkan aktivitas pertanian secara otomatis.
        - Price Recommendation: Menghitung rekomendasi harga jual hasil panen.
5. **Shared Service**
    a. **Authentication:** Layanan untuk login dan manajemen akun pengguna.
    b. **Monitoring & Logs:** Layanan analitik dan log sistem untuk memantau performa.
6. **Database**
    a. **Firebase:** Database utama untuk menyimpan data petani, lahan, jadwal, dan hasil analisis.
7. **Infrastructure Services**
    a. **Containerization Layer:** Infrastruktur untuk menjalankan aplikasi berbasis microservices.

3.4 Teknologi Dipilih

Berikut adalah beberapa teknologi yang dipilih.

1. **Hardware**
    - NodeMCU ESP8266
    - Sensor Temperatur dan Kelembapan Tanah
    - Kabel female to female secukupnya
2. **Communication Protocol**
    - MQTT (Message Queuing Telemetry Transport)

- HTTP/HTTPS
- Bluetooth

3. **Software Tools**
   - Arduino IDE
   - Visual Studio Code
   - Adafruit IO
   - Firebase

4. Rencana Pengembangan

4.1 Langkah Pengembangan

Berikut merupakan langkah-langkah pengembangan GREENGROW.

1. Setup Lingkungan
   a. Menyipakan Repository.
   b. Konfigurasi sever.
   c. Instal framework.
   d. Desain API.
   e. Desain UI/UX.
   f. Setup IOT.
      i. Instalasi perangkat lunak pendukung untuk IoT.
      ii. Uji koneksi perangkat IoT dengan server menggunakan protokol.
2. Pengembangan Backend
   a. Membuat struktur aplikasi (routes, controller, service, repositories, database).
   b. Mengimplementasikan microservice dan autentikasi.
   c. Integrasi IoT backend:
      i. Menyambungkan IoT gateway dengan data server.
      ii. Membuat endpoint API untuk komunikasi IoT.
      iii. Menyimpan data dari sensor ke database.
      iv. Mengembangkan sistem notifikasi berbasis data IoT.
3. Pengembangan Frontend
   a. Mengimplementasikan halaman utama, antrian, dan notifikasi.
   b. Mengintegrasikan API backend.
   c. Menampilkan data real-time dari perangkat IoT pada dashboard.
4. Integrasi Sistem
   a. Menghubungkan semua komponen melalui API.
   b. Integrasi penuh perangkat IOT dengan sistem.
      i. Sinkronisasi data IoT dengan fitur-fitur seperti pemetaan lahan, pengelolaan jadwal, dan otomatisasi.

5. Pengujian
    a. Uji sistem secara menyeluruh, termasuk data dari perangkat IoT.
    b. Pengujian komunikasi IoT (responsivitas, akurasi data sensor, perintah ke aktuator).
6. Deployment
    a. Deploy backend, frontend, dan server IoT.

4.2 Timeline

Berikut merupakan timeline pengembangan GREENGROW.

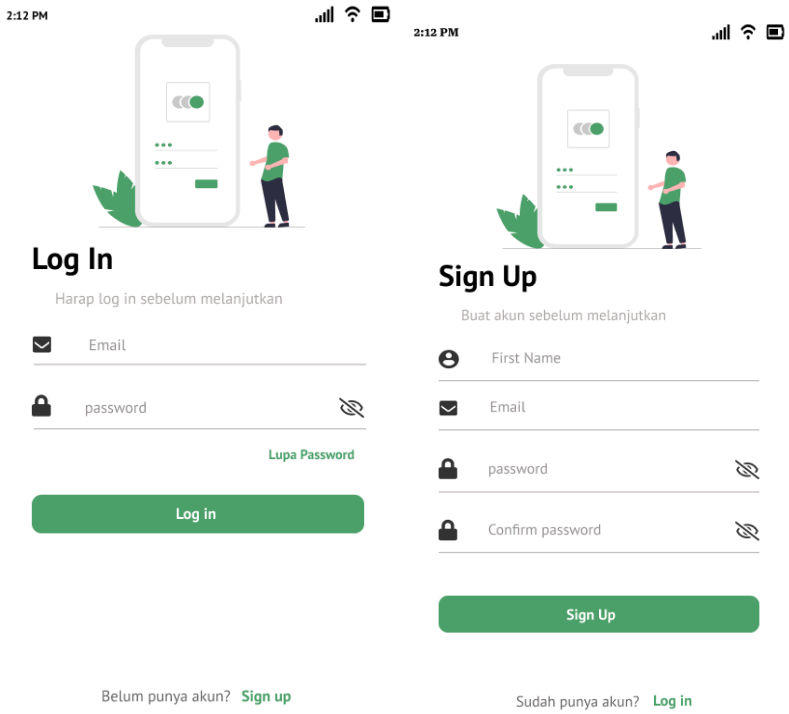
Kegiatan	Bulan ke-1				Bulan ke-2			
	M 1	M 2	M 3	M 4	M 1	M 2	M 3	M 4
Perencanaan dan desain sistem								
Setup lingkungan								
Setup IoT								
Pengembangan backend								
Pengembangan frontend								
Integrasi sistem								
Pengujian								
Deployment								

Tabel 4.1 Timeline Implementasi GREENGROW

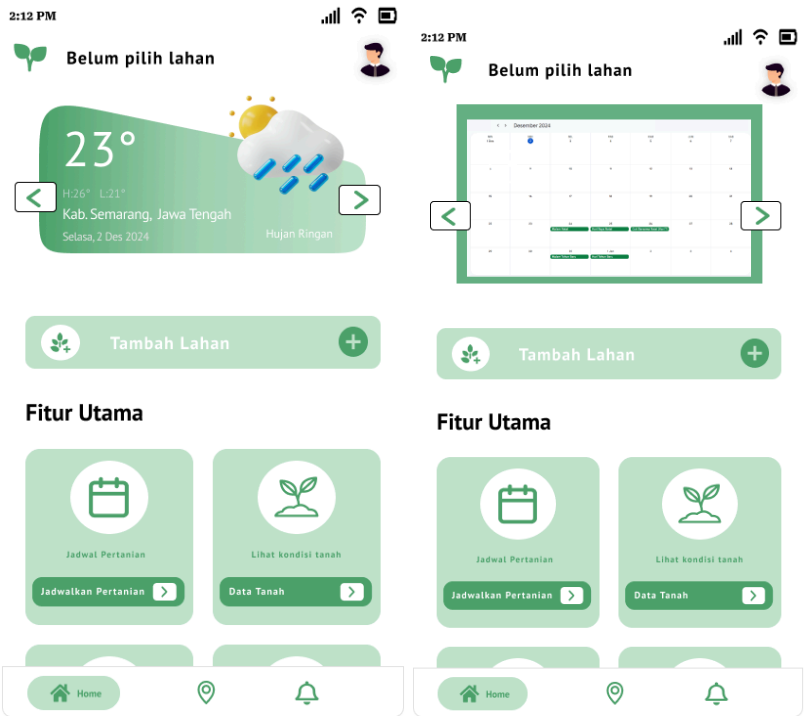
5. Implementasi

5.1 Desain UI/UX

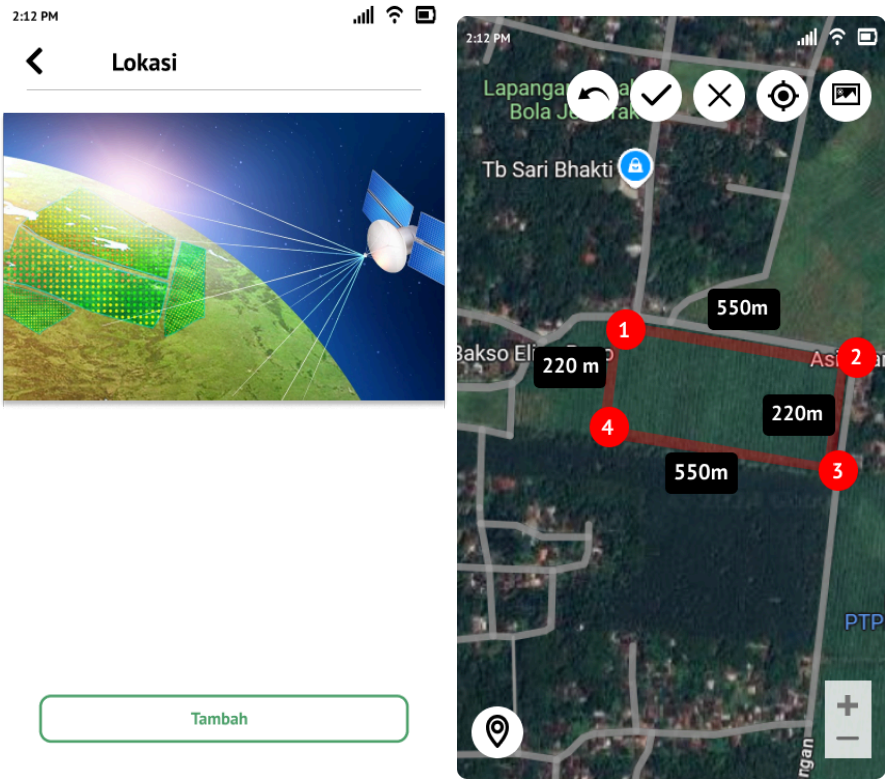
- **Laman Login dan Register**

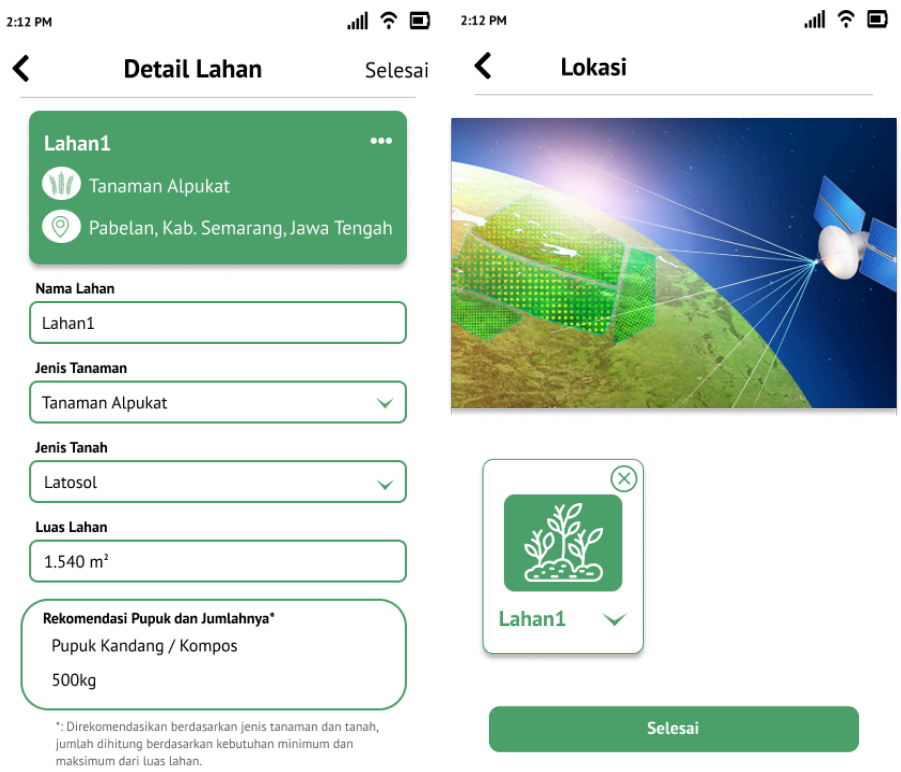


- **Laman Utama**

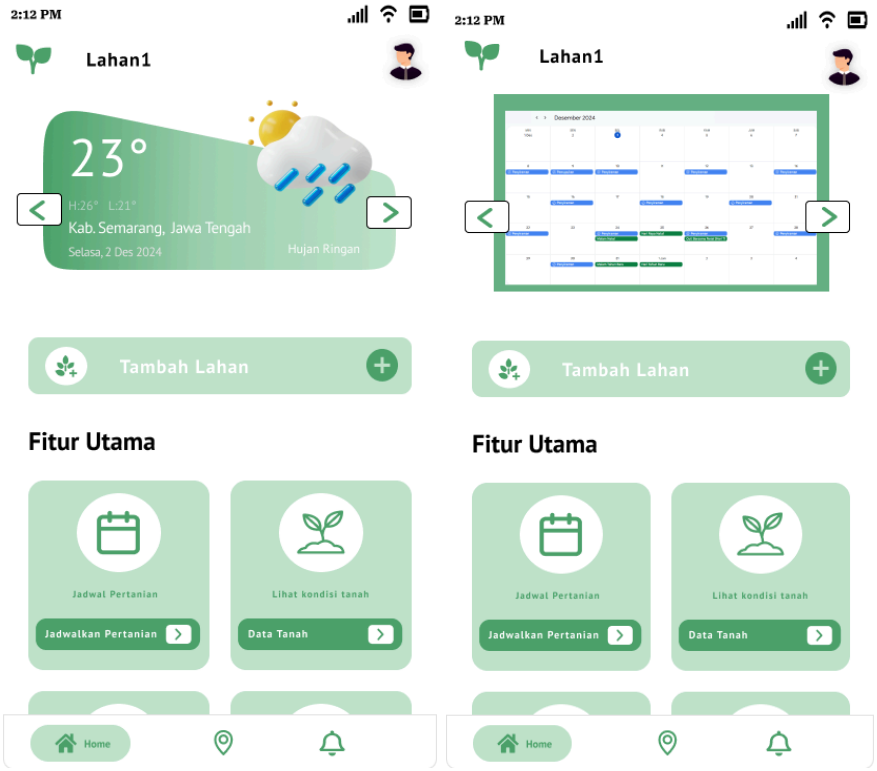


- **Laman Tambah Lahan**

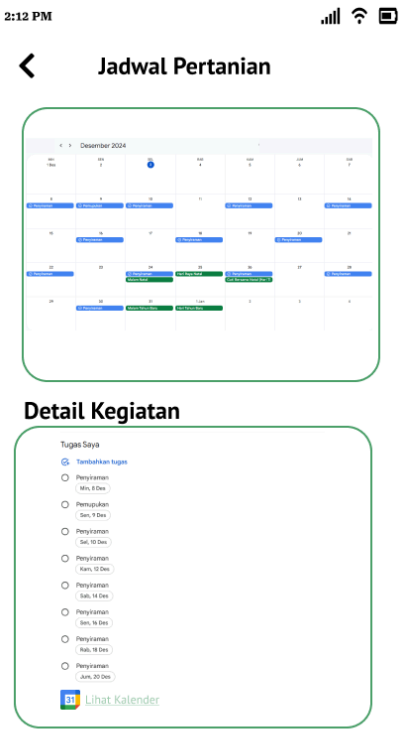




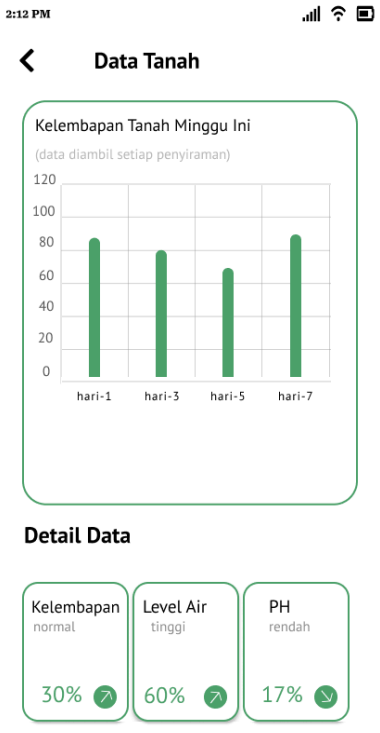
- **Laman Utama Setelah Pilih Lahan**



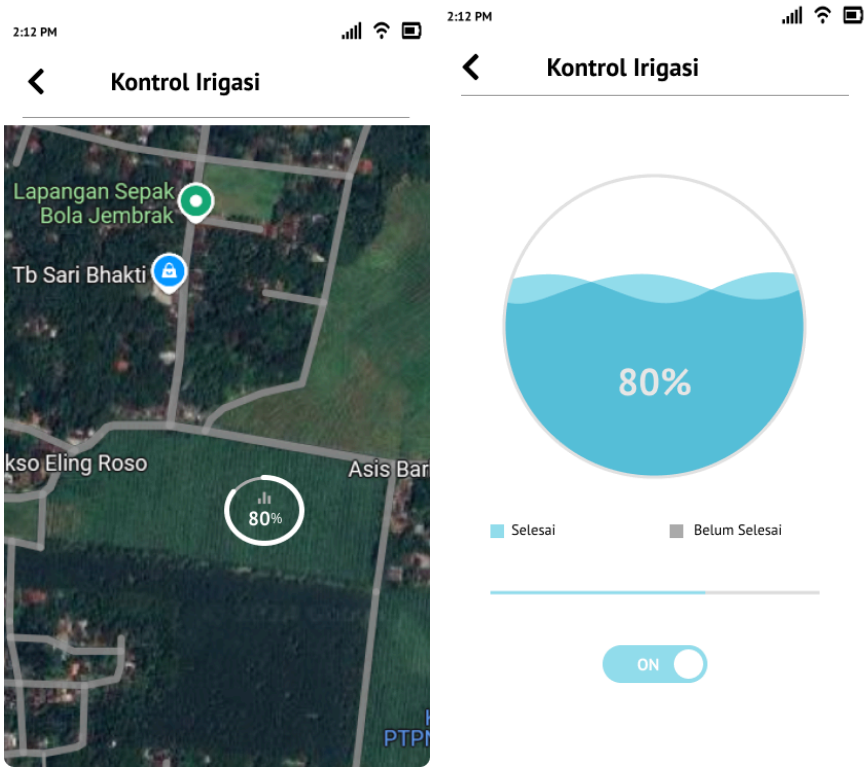
- **Laman Jadwal**



- **Laman Kondisi Tanah**



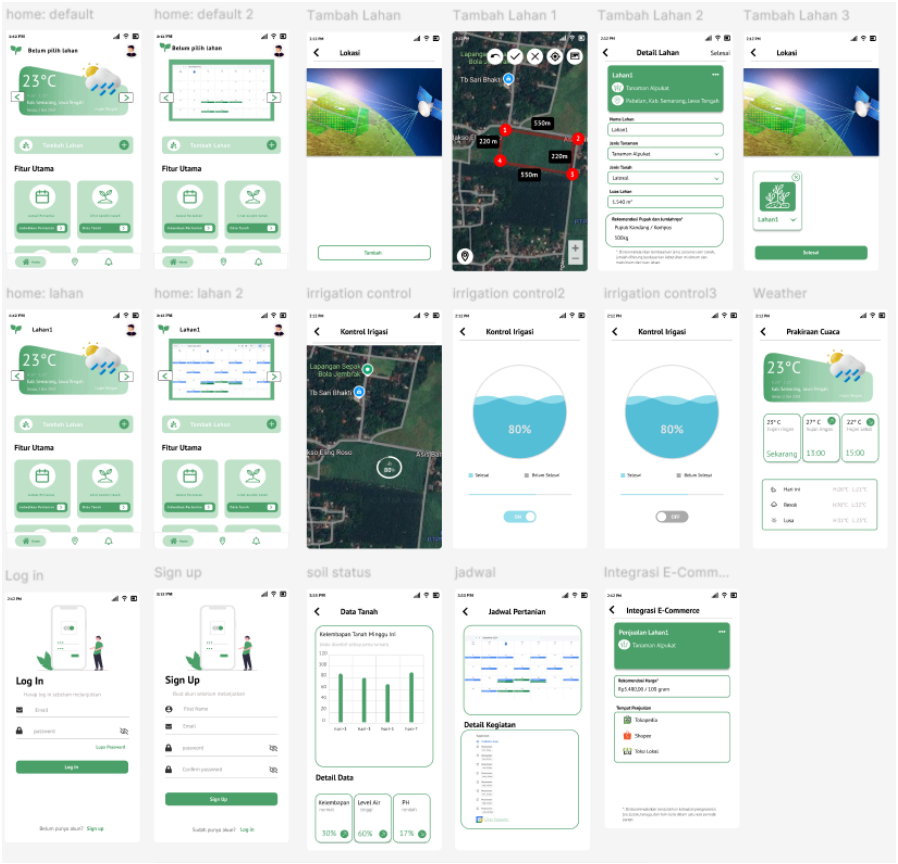
- **Laman Kontrol Irigasi**



- **Laman Integrasi E-commerce**

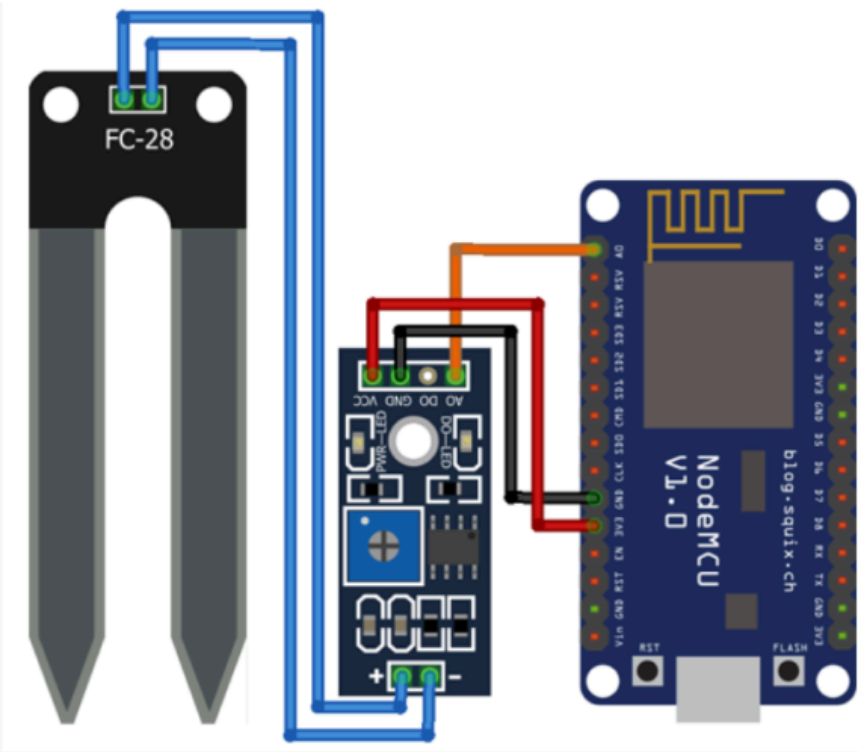
- **Laman Prakiraan Cuaca**

- **Seluruh Laman**



5.2 Implementasi IoT

- **Ilustrasi Implementai IoT**



- **Kode IoT**

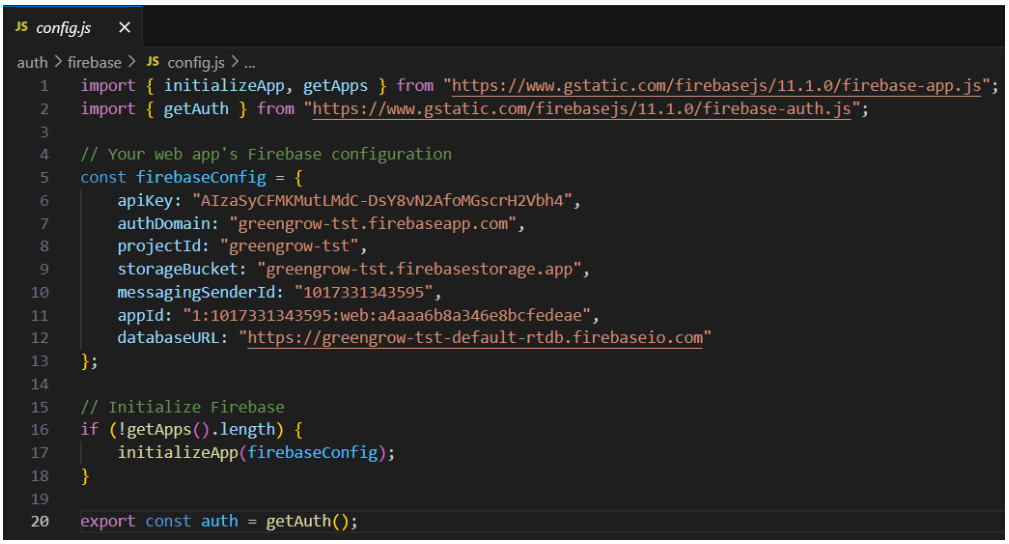
Untuk prototyping, kita menggunakan aplikasi BLYNK yang ada di android terlebih dahulu untuk melakukan visualisasi sensor kelembapan tanahnya.

```
#include <ESP8266WiFi.h> //library esp8266
#include <BlynkSimpleEsp8266.h> //library BLYNK

#define BLYNK_PRINT Serial

BlynkTimer timer;

char auth[] = "d8ff16e65ff044d18f9b6c3ad1eaa87b";
//cek email masing2 setelah setting di aplikasi BLYNK

char ssid[] = "Blok21no12"; //nama wifi
char pass[] = "Komponen08"; //password wifi

void myTimerEvent()
{
int adc = analogRead(A0); //PIN ADC di NODEMCU
float tegangan = adc * (5.0 / 1023.0);
Blynk.virtualWrite(V0,  adc); //Virtual wire "0"
```

```
untuk data pembacaan ADC
Blynk.virtualWrite(V1,  tegangan); //Virtual wire "1"
untuk data pembacaan ADC
}

void setup()
{
  Serial.begin(115200); //baud serial monitor
  Blynk.begin(auth, ssid, pass);
  timer.setInterval(1000L, myTimerEvent); //1000L = 1
detik
}

void loop()
{
 Blynk.run();
 timer.run();
}
```

5.3 Implementasi API

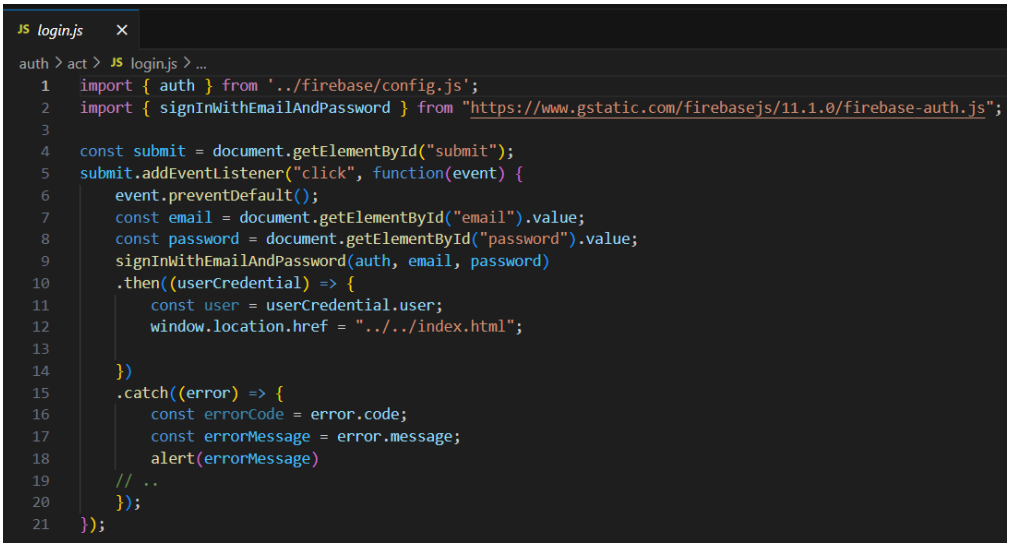
API yang digunakan adalah API Firebase.
File config.js (Untuk melakukan konfigurasi firebase)



5.4 Implementasi Autentikasi

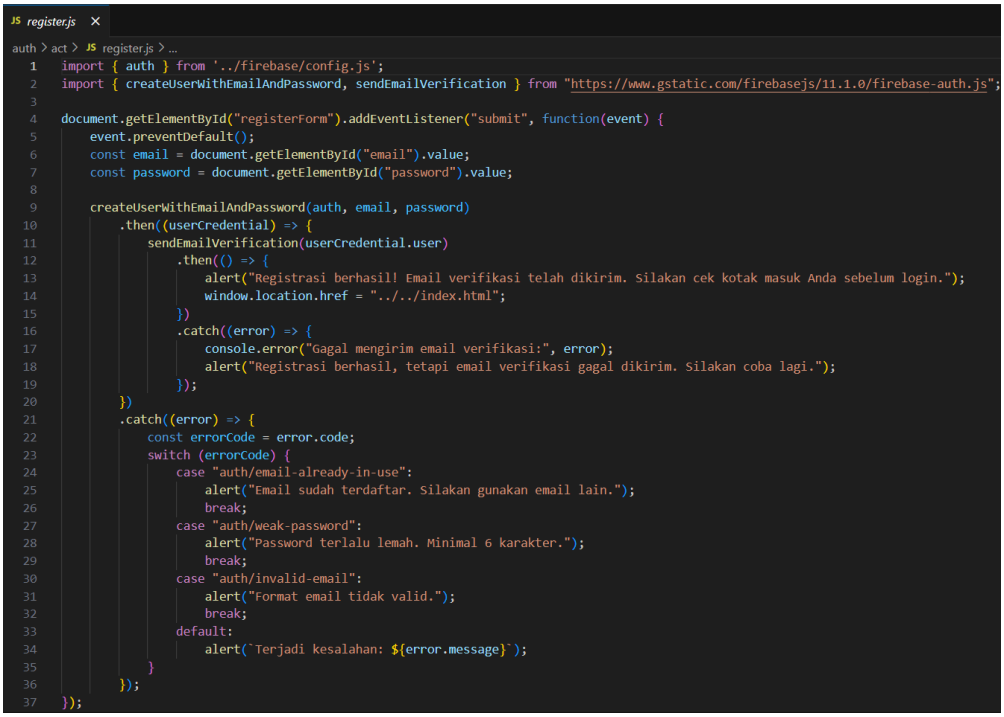
Autentikasi menggunakan API Firebase Authentication. Maka dari itu, dibuat:
Bukti penerapan:

- File login.js (Untuk memasukkan sesi login)



```
auth > act > JS login.js > ...
import { auth } from '../firebase/config.js';
import { signInWithEmailAndPassword } from "https://www.gstatic.com/firebasejs/11.1.0/firebase-auth.js";

const submit = document.getElementById("submit");
submit.addEventListener("click", function(event) {
    event.preventDefault();
    const email = document.getElementById("email").value;
    const password = document.getElementById("password").value;
    signInWithEmailAndPassword(auth, email, password)
    .then((userCredential) => {
        const user = userCredential.user;
        window.location.href = "../../index.html";

    })
    .catch((error) => {
        const errorCode = error.code;
        const errorMessage = error.message;
        alert(errorMessage)
    // ..
    });
});
```

- File register.js (Untuk menambahkan data user ke database)

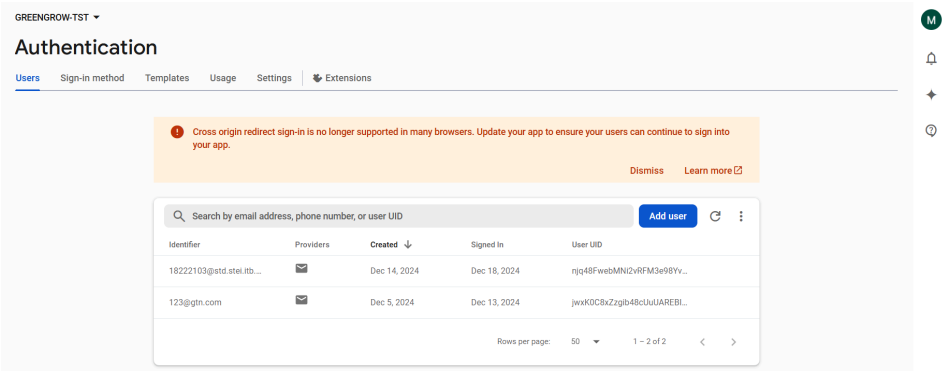


```
auth > act > JS register.js > ...
import { auth } from '../firebase/config.js';
import { createUserWithEmailAndPassword, sendEmailVerification } from "https://www.gstatic.com/firebasejs/11.1.0/firebase-auth.js";

document.getElementById("registerForm").addEventListener("submit", function(event) {
    event.preventDefault();
    const email = document.getElementById("email").value;
    const password = document.getElementById("password").value;

    createUserWithEmailAndPassword(auth, email, password)
        .then((userCredential) => {
            sendEmailVerification(userCredential.user)
                .then(() => {
                    alert("Registrasi berhasil! Email verifikasi telah dikirim. Silakan cek kotak masuk Anda sebelum login.");
                    window.location.href = "../../index.html";
                })
                .catch((error) => {
                    console.error("Gagal mengirim email verifikasi:", error);
                    alert("Registrasi berhasil, tetapi email verifikasi gagal dikirim. Silakan coba lagi.");
                });
        })
        .catch((error) => {
            const errorCode = error.code;
            switch (errorCode) {
                case "auth/email-already-in-use":
                    alert("Email sudah terdaftar. Silakan gunakan email lain.");
                    break;
                case "auth/weak-password":
                    alert("Password terlalu lemah. Minimal 6 karakter.");
                    break;
                case "auth/invalid-email":
                    alert("Format email tidak valid.");
                    break;
                default:
                    alert(`Terjadi kesalahan: ${error.message}`);
            }
        });
});
```

- File session.js (Untuk mengambil data user yang sedang login)

```
session.js
auth > act > session.js > ...
1  import { auth } from '../firebase/config.js';
2  import { onAuthStateChanged } from "https://www.gstatic.com/firebasejs/11.1.0/firebase-auth.js";
3
4  const loggedOutState = document.getElementById('logged-out-state');
5  const loggedInState = document.getElementById('logged-in-state');
6
7  onAuthStateChanged(auth, (user) => {
8      if (user) {
9          console.log("User ID:", user.uid);
10         console.log("User Email:", user.email);
11         console.log("User Display Name:", user.displayName);
12         loggedOutState.classList.add('hidden');
13         loggedInState.classList.remove('hidden');
14     } else {
15         console.log("No user is signed in.");
16         loggedOutState.classList.remove('hidden');
17         loggedInState.classList.add('hidden');
18     }
19 });
20
21
```

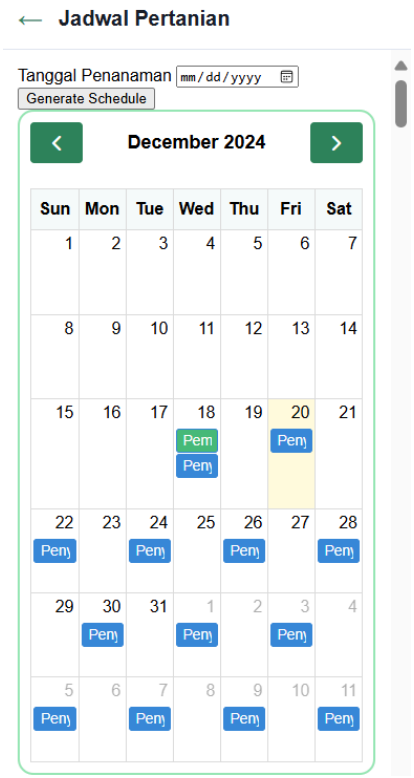
- Bukti pada Firebase

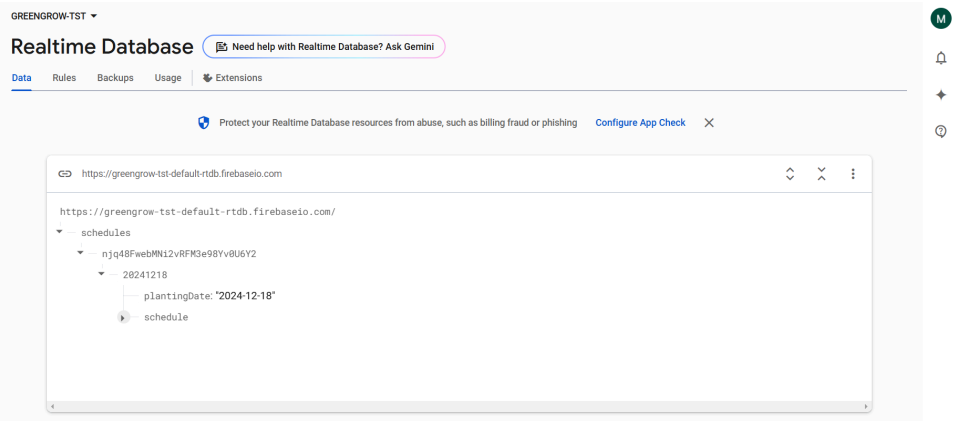


5.5 Implementasi Core Domain

Pada core domain yang sudah didefinisikan, akhirnya diputuskan untuk melakukan implementasi kalender untuk melakukan penjadwalan pertanian (dapat digunakan pada core domain Irrigation Control, Fertilization Management, dan Farm Scheduling). Digunakan REST untuk mengimplementasikan hal ini.

- Kode Implementasi
  Dapat dilihat di repository github.
  (https://github.com/MreffyH/TubesTST103/blob/main/page/act/jadwal.js)
- Bukti Implementasi (Pada implementasi web dan data di firebase)





5.6 Repository Kode

index.html
script.js
styles.css
page

- jadwal.html
- profile.html
- act
  - jadwal.js
  - profile.js
- style

        - jadwal.css
        - profile.css

auth

- login.html
- register.html
- firebase
    - config.js
- act
    - auth.js
    - login.js
    - logout.js
    - register.js
    - session.js
- style
    - auth.css

assets (Menyimpan gambar)

6. Evaluasi

Selama pengerjaan tugas besar ini terdapat banyak masalah yang dihadapi serta pelajaran yang bisa diambil. Berikut adalah pemaparan evaluasi pengerjaan tugas besar ini.

6.1. Masalah yang Dihadapi dan Solusi

Dalam pengerjaan tugas ini, terdapat beberapa masalah yang saya hadapi. Berikut merupakan beberapa masalah tersebut beserta solusi yang telah dilakukan.

Masalah	Solusi
Kurangnya pemahaman mengenai berbagai bahsa pemrograman, *framework,* dan *liblary* yang harus digunakan.	Belajar dari berbagai sumber di internet dan mencari pengetahuan lebih dari teman maupun kakak tingkat.
Kurangnya manajemen waktu karena tugas besar yang sangat banyak di semester ini.	Berusaha untuk lebih disiplin dalam melakukan kegiatan sesuai jadwal dan mebuat jadwal lebih terstruktur.
Pengetahuan dalam pengembangan aplikasi mobile yang masih sangat sedikit.	Membuat website dengan HTML, CSS, dan JS tetapi dengan tampilan seolah-olah seperti mobile.
Repository yang awalnya sangat berantakan sehingga membingungkan selama pengerjaan.	Merapikan repository dengan membuat folder dan membagi file sesuai fungsi dan jenis file.

Kesulitan dalam menentukan *tools* yang digunakan untuk merealisasikan tugas besar ini.	Mencari berbagai *tools* dan menentukan *tools* yang mudah diimplementasikan dalam waktu yang terbatas.

6.2. Pelajaran yang Dapat Diambil

Dalam pengerjaan tugas besar ini, terdapat beberapa hal baru yang telah saya pelajari.

- Tidak boleh menyerah dalam mencari segala solusi dalam menghadapi *error* dan berbagai kesulitan yang dihadapi.
- Mendapatkan pengetahuan dalam mengembangkan *website* (secara full stack) dari proses pembuatan UI/UX, *frontend, backend*, hingga integrasi API.
- Implementasi API secara baik dan lebih efisien.
- Mendapatkan pengetahuan lebih dalam mengenai sistem terintegrasi.

Lampiran

- **Business Capability Map:** https://www.canva.com/design/DAGXYXk94JU/RjoeqMFHXqHygZ9oMfe_zQ/edit
- **Model Sistem:** **https://drive.google.com/file/d/11drX6ujbG_8kIWnlMUM1I1T738lmqrZM/view?usp=sharing**
- **Deploy:** **https://tubes-tst-103.vercel.app/**
- **Github:** **https://github.com/MreffyH/TubesTST103**
- **Figma:** **https://www.figma.com/proto/DFdLRUc5u0oVz5oLL9MikL/GreenGrow-TST?node-id=108-2102&p=f&t=Wizhw9hnFYPck6GQ-1&scaling=min-zoom&content-scaling=fixed&page-id=103%3A243&starting-point-node-id=108%3A2102**
- **Presentasi:** **https://www.canva.com/design/DAGZ0OYsXP0/0OU0Zt8HJGuhNgZ7qwhMHA/edit?utm_content=DAGZ0OYsXP0&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton**