BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama menjalani masa magang di PDASRH Kementerian Kehutanan Republik Indonesia, ditempatkan pada Divisi Data dan Informasi (Datin) dan berperan sebagai *Mobile Application Developer*. Tugas utama adalah merancang dan mengembangkan aplikasi Penghijauan yang ditujukan untuk mempermudah proses pencatatan kegiatan penghijauan oleh masyarakat.

Dalam pelaksanaan tugas tersebut, berada di bawah bimbingan Bapak Nabil selaku pembimbing lapangan dari Divisi Datin. Beliau memberikan arahan, informasi kebutuhan sistem, serta evaluasi secara berkala terhadap progres pengembangan aplikasi. Koordinasi dan diskusi dilakukan secara aktif dengan pembimbing untuk memastikan bahwa aplikasi yang dikembangkan sesuai dengan kebutuhan.

3.2 Tugas yang Dilakukan

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1-3	Perkenalan, Analisis Kebutuhan, mulai mendesain *UI/UX* di *Figma*
4-6	Melanjutkan mendesain *UI/UX*, memperbaiki *Gradle*, mulai mengoding tampilan *front end*
7-10	Masih melakukan *front end*
11-13	Melakukan *front end* sekaligus *back end*, memperbaiki *crash* pada halaman *Plant a Tree*
14-16	Melakukan *back end* dan melakukan *debugging*
17-21	Melakukan *debugging*

3.3.1 Penjelasan Tugas

Aplikasi Penghijauan merupakan platform digital yang dikembangkan untuk mendukung program penanaman pohon dan konservasi lingkungan. Aplikasi ini dirancang untuk memfasilitasi pengguna dalam melakukan penanaman pohon, mencatat aktivitas penghijauan, dan mengakses informasi terkait persemaian di Indonesia.

Tabel 3.2. Penjelasan Tugas Singkat

Halaman/Fitur	**Fungsi**
Welcome Screen	Menampilkan tombol *sign in* dan *sign up* untuk masuk atau mendaftar.
Login/Register	Mengelola *autentikasi* pengguna menggunakan *Firebase Authentication*.
Home Page	Menyapa pengguna dengan pesan personal, misalnya "Halo, Nama Pengguna".
Plant A Tree	Formulir digital untuk mencatat jumlah pohon, titik koordinat, foto, tanggal, dan bulan penanaman, disimpan di *Firebase*.
History	Menampilkan riwayat penanaman pohon pengguna dari data *Firebase*.
Claim Certificate	Mengizinkan pengguna mengklaim sertifikat setelah menyelesaikan penanaman.
Nursery Location	Menyediakan informasi lokasi persemaian di Indonesia dengan fitur pencarian.
Profile	Menampilkan informasi pengguna, seperti nama dan foto profil.

3.3.2 Proses Pelaksanaan

Selama periode pelaksanaan magang dalam merancang aplikasi *mobile*, diperlukan beberapa perangkat lunak serta perangkat keras. Berikut adalah perangkat lunak yang digunakan dalam membuat aplikasi *mobile*:

1. *Android Studio*
2. *Firebase*

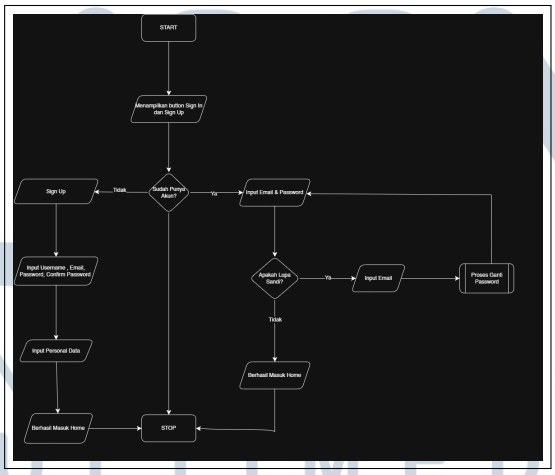
3. *Figma*
4. *Google Chrome*

Adapun perangkat keras yang digunakan dalam pelaksanaan kegiatan magang ini sebagai berikut:

1. *Processor Laptop*: AMD Ryzen 5 4600H with Radeon Graphics, 3.00 GHz
2. *RAM Laptop*: 8.00 GB (7.37 GB usable)
3. *Sistem Operasi Laptop*: 64-bit *operating system*, x64-based *processor*
4. *Hard disk Laptop*: 477 GB
5. *Oppo A53* (*Android 10*)

A Flowchart Aplikasi

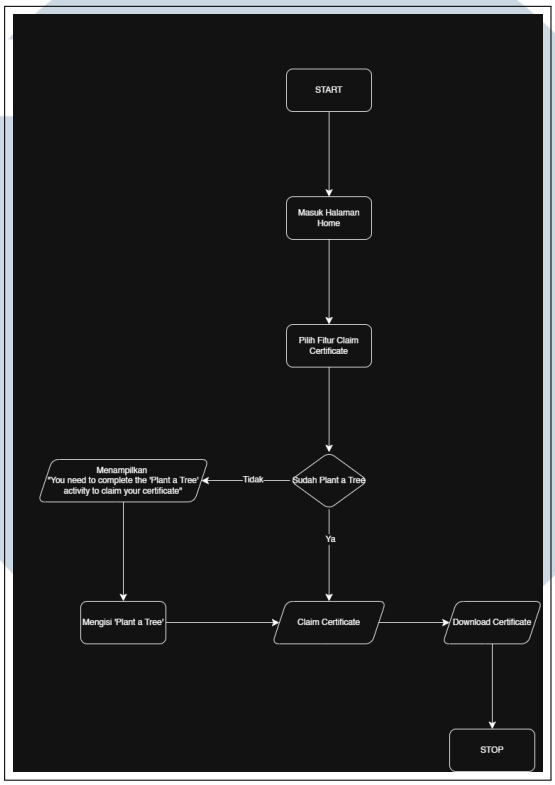
Pada bagian ini disajikan *flowchart* dari fitur-fitur utama dalam aplikasi. *Flowchart* digunakan untuk memvisualisasikan alur kerja aplikasi sehingga pengguna maupun pengembang dapat memahami proses yang terjadi di dalamnya secara lebih mudah.



Gambar 3.1. Antarmuka *Flowchart Sign In* dan *Sign Up*

Gambar 3.1 memperlihatkan alur proses masuk (*sign in*) dan pendaftaran (*sign up*) pengguna. Apabila pengguna belum memiliki akun, maka perlu melakukan pendaftaran terlebih dahulu. Setelah memiliki akun, pengguna dapat masuk ke dalam aplikasi menggunakan email dan kata sandi yang telah didaftarkan.

Setelah berhasil masuk, pengguna dapat mengakses berbagai fitur aplikasi, salah satunya adalah fitur untuk klaim sertifikat. *Flowchart* dari proses ini ditampilkan pada gambar berikut.



Gambar 3.2. Antarmuka *Flowchart Claim Certificate*

Gambar 3.2 memperlihatkan proses pengguna dalam mengklaim sertifikat. Sebelum dapat mengakses fitur klaim, pengguna harus terlebih dahulu memenuhi persyaratan tertentu, seperti telah melakukan aksi *plant a tree*. Jika syarat tersebut sudah dipenuhi, maka pengguna dapat melanjutkan proses klaim dan mengunduh sertifikat.

B Implementasi Database Firebase Firestore

Firebase Firestore digunakan sebagai basis data utama dalam pengembangan aplikasi ini. Firestore merupakan layanan *cloud-hosted NoSQL* yang menyimpan data dalam bentuk *collections* dan *documents*. Setiap dokumen dapat berisi pasangan *key-value*, termasuk objek bersarang, array, dan berbagai tipe data lainnya.

Pemilihan Firestore didasarkan pada kebutuhan aplikasi yang memerlukan pembaruan data secara *real-time* dan dapat disinkronkan antar perangkat. Dengan

struktur yang fleksibel dan skalabilitas tinggi, Firestore sangat cocok digunakan dalam pengembangan aplikasi mobile seperti yang dibuat dalam proyek ini.

B.1 Struktur Koleksi `trees`

Koleksi `trees` menyimpan informasi mengenai pohon yang telah ditanam oleh pengguna. Setiap dokumen dalam koleksi ini mewakili satu aktivitas penanaman pohon dan memiliki atribut-atribut sebagai berikut:

Tabel 3.3. Struktur Koleksi `trees` pada Firebase Firestore

Field	Tipe Data	Deskripsi
completed	Boolean	Menunjukkan status apakah penanaman telah selesai (`true` atau `false`).
locationCoordinates	String	Koordinat lokasi penanaman yang diinput secara manual oleh pengguna berdasarkan titik pada Google Maps, dalam format `latitude,longitude`.
photoBase64	String	Gambar pohon yang disimpan dalam format teks *base64*.
planterId	String	UID pengguna Firebase Authentication yang menanam pohon.
plantingDate	String	Tanggal pohon ditanam, disimpan dalam format `dd/mm/yyyy`.
timestamp	Number	Waktu penanaman dalam format *UNIX timestamp* (milidetik).
treeCount	Number	Jumlah pohon yang ditanam pada aktivitas tersebut.

B.2 Contoh Dokumen Koleksi `trees`

Berikut adalah contoh representasi data dalam koleksi `trees` yang disimpan di Firestore:

```
{
  "completed": true,
  "locationCoordinates": "-6.2067864,106.8002860",
  "photoBase64": "...",
  "planterId": "KphvNobtlThsjecwGPIBPItRk6I3",
```

```
  "plantingDate": "30/6/2025",
  "timestamp": 1751264305007,
  "treeCount": 6
}
```

Kode 3.1: Contoh Dokumen Koleksi `trees`

B.3 Struktur Koleksi `users`

Koleksi `users` menyimpan informasi terkait profil pengguna yang terdaftar pada aplikasi. Setiap dokumen dalam koleksi ini diidentifikasi berdasarkan UID pengguna dari Firebase Authentication dan menyimpan atribut-atribut berikut:

Tabel 3.4. Struktur Koleksi `users` pada Firebase Firestore

Field	Tipe Data	Deskripsi
birthDate	String	Tanggal lahir pengguna, disimpan dalam format `dd/mm/yyyy`.
email	String	Alamat surel pengguna yang digunakan saat pendaftaran.
fullname	String	Nama lengkap pengguna.
gender	String	Jenis kelamin pengguna, misalnya `Male` atau `Female`.
phoneNumber	String	Nomor telepon yang dimasukkan oleh pengguna.
username	String	Nama pengguna unik (*username*) yang digunakan di aplikasi.

B.4 Contoh Dokumen Koleksi `users`

Berikut adalah contoh data pengguna yang disimpan dalam dokumen koleksi `users`:

```
{
  "birthDate": "5/7/2025",
  "email": "cagyaaaa7524@gmail.com",
  "fullname": "neisya",
  "gender": "Female",
  "phoneNumber": "082112587717",
  "username": "neca"
```

```
}
```

Kode 3.2: Contoh Dokumen Koleksi `users`

B.5 Aturan Keamanan Firestore (*Security Rules*)

Untuk menjaga integritas dan privasi data pengguna, Firestore menyediakan fitur *security rules*. Aturan ini berfungsi membatasi akses data berdasarkan status autentikasi pengguna.

Aplikasi ini menggunakan aturan berikut:

- Pengguna hanya dapat membaca dan menulis data pada dokumen `users` miliknya sendiri (berdasarkan UID).
- Setiap pengguna yang telah terautentikasi dapat membaca dan menulis data pada koleksi `trees`.

Berikut adalah potongan kode *security rules* yang digunakan:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    // Akses hanya untuk pengguna yang sesuai UID-nya
    match /users/{userId} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }

    // Akses untuk semua pengguna yang telah login
    match /trees/{treeId} {
      allow read, write: if request.auth != null && request.auth.uid != null;
    }
  }
}
```

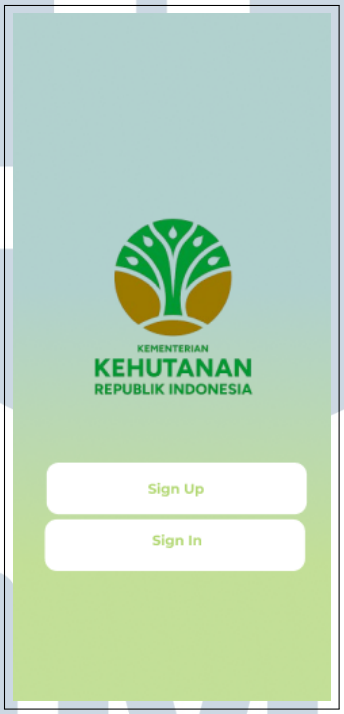
Kode 3.3: Aturan Keamanan Firebase Firestore

C Rancangan Antarmuka Aplikasi *Mobile*

Pada bagian ini dipaparkan rancangan antarmuka pengguna dari aplikasi *mobile* yang dikembangkan selama masa magang. Perancangan antarmuka

dilakukan dengan mempertimbangkan kemudahan penggunaan, tampilan yang intuitif, serta kesan profesional yang mencerminkan identitas Kementerian Kehutanan. Setiap halaman pada aplikasi memiliki peran dan fungsi yang saling terintegrasi untuk menunjang proses pendataan kegiatan penghijauan oleh masyarakat.

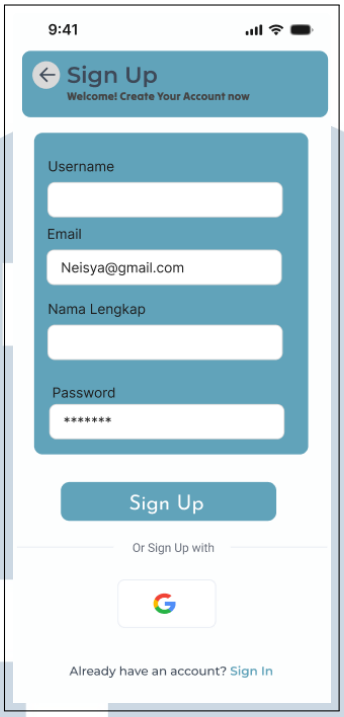
Salah satu halaman awal yang penting dalam aplikasi ini adalah tampilan awal atau layar sambutan pengguna. Tampilan ini menjadi titik masuk pertama sebelum pengguna melakukan proses autentikasi.



Gambar 3.3. Antarmuka *Welcome Screen*

Gambar 3.3 menampilkan tampilan *Welcome Screen* pada aplikasi. Halaman ini berfungsi sebagai layar pembuka yang menyambut pengguna sebelum masuk ke dalam sistem. Pada bagian tengah layar ditampilkan logo resmi Kementerian Kehutanan Republik Indonesia, yang memberikan kesan kredibel dan resmi terhadap aplikasi. Di bawah logo, terdapat dua tombol utama yaitu *Sign Up* dan *Sign In*. Kedua tombol ini mengarahkan pengguna ke proses registrasi akun baru maupun *login* bagi pengguna yang sudah memiliki akun.

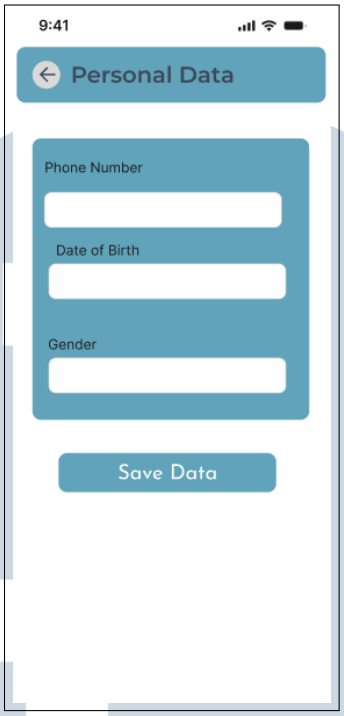
Setelah melewati halaman sambutan, pengguna diarahkan untuk melakukan registrasi akun apabila belum memiliki akun sebelumnya. Halaman registrasi ini memuat berbagai kolom isian yang dibutuhkan untuk membuat akun baru.



Gambar 3.4. Antarmuka *Sign Up*

Gambar 3.4 menampilkan antarmuka *Sign Up* yang digunakan oleh pengguna baru untuk membuat akun di aplikasi. Halaman ini dirancang dengan *form* isian yang terdiri dari *Username*, *Email*, *Nama Lengkap*, dan *Password*. Komponen *form* ditata secara vertikal dengan jarak yang proporsional agar mudah diisi dan dibaca. Selain itu, terdapat opsi *Sign Up with Google* sebagai alternatif metode pendaftaran yang lebih praktis. Navigasi menuju halaman *Sign In* juga tersedia di bagian bawah, memastikan pengguna dapat berpindah halaman dengan mudah. Desain ini menekankan kemudahan penggunaan dan aksesibilitas yang baik bagi pengguna baru.

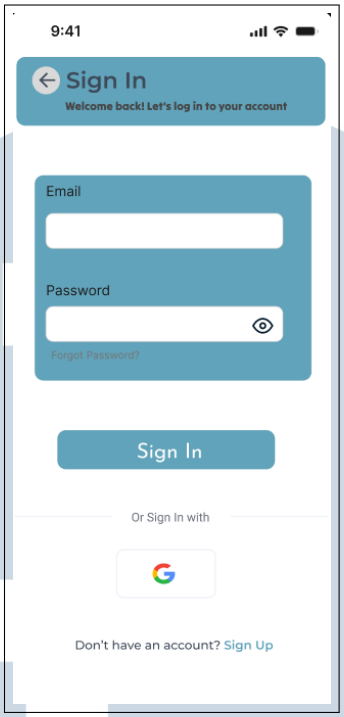
Berikutnya adalah tampilan halaman data pribadi yang muncul setelah pengguna berhasil mendaftar akun.



Gambar 3.5. Antarmuka *Personal Data*

Gambar 3.5 merupakan tampilan antarmuka *Personal Data* yang muncul setelah pengguna berhasil melakukan proses *Sign Up*. Pada halaman ini, pengguna diminta untuk melengkapi informasi pribadi seperti nomor telepon, tanggal lahir, dan jenis kelamin. Seluruh elemen isian dirancang secara vertikal dengan *layout* yang sederhana dan mudah diakses. Tombol *Save Data* diletakkan di bagian bawah untuk menyimpan data yang telah diisi.

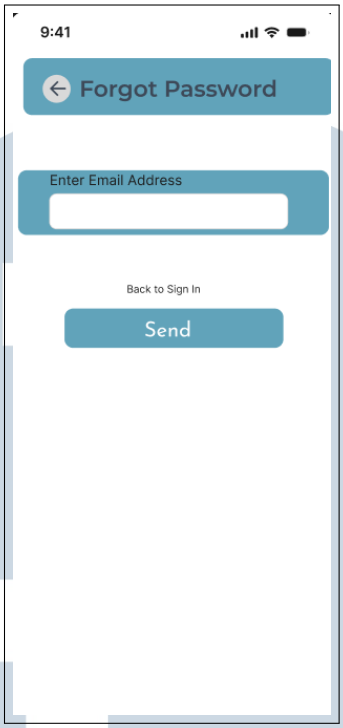
Selanjutnya terdapat halaman masuk bagi pengguna yang telah memiliki akun aplikasi.



Gambar 3.6. Antarmuka *Sign In*

Gambar 3.6 merupakan tampilan antarmuka *Sign In*, tampilan ini terdiri atas kolom isian *Email* dan *Password*, dilengkapi dengan ikon untuk menampilkan atau menyembunyikan kata sandi. Di bawah kolom *Password*, tersedia tautan *Forgot Password?* untuk memudahkan pengguna yang lupa sandi. Tombol Sign In disajikan dengan ukuran dan posisi yang jelas untuk mendorong aksi utama pengguna. Selain itu, terdapat opsi masuk menggunakan akun Google (Sign In with Google) yang memberikan alternatif metode *autentikasi*. Pada bagian bawah, juga disediakan tautan untuk mendaftar akun (Sign Up) bagi pengguna baru.

Jika pengguna mengalami lupa kata sandi, aplikasi menyediakan halaman pemulihan yang dapat diakses dengan mudah.



Gambar 3.7. Antarmuka *Forgot Password*

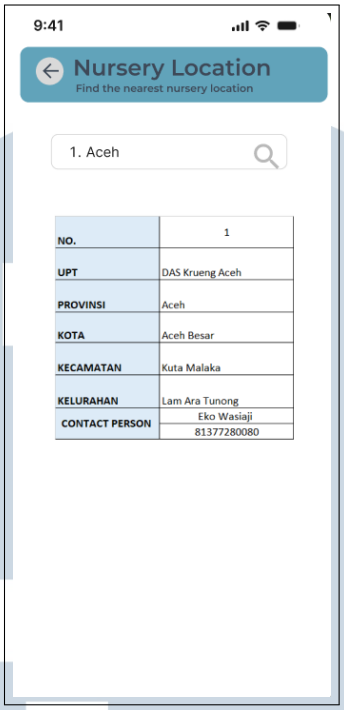
Gambar3.7 menunjukkan tampilan antarmuka *Forgot Password* yang ditampilkan saat pengguna lupa kata sandi. Desain halaman ini dibuat dengan sederhana dan fokus pada satu tujuan, yaitu memasukkan alamat *email* untuk proses pemulihan akun. Komponen utama terdiri dari kolom *input* alamat *email*, tombol aksi *Send*, serta tautan kecil untuk kembali ke halaman *Sign In*.

Halaman selanjutnya merupakan fitur utama dalam aplikasi, yaitu pencatatan aktivitas penanaman pohon oleh pengguna.

Gambar 3.8. Antarmuka *Plant a Tree*

Gambar 3.8 merupakan tampilan desain antarmuka *Plant a Tree* yang dirancang dengan susunan elemen yang sederhana dan terstruktur. Pada bagian atas terdapat judul halaman beserta ikon kembali untuk memudahkan navigasi. Di tengah layar terdapat ilustrasi pohon sebagai elemen visual pendukung. Formulir isian ditampilkan dalam satu kolom, mencakup input jumlah pohon, tanggal penanaman, unggahan foto, serta titik koordinat. Tombol *Save Data* diletakkan di bagian bawah sebagai aksi utama yang mudah dijangkau pengguna. Secara keseluruhan, tampilan ini dirancang untuk memberikan pengalaman pengguna yang jelas dan intuitif.

Selain fitur pelaporan penanaman pohon, aplikasi juga menyediakan informasi mengenai lokasi persemaian bibit agar pengguna dapat mengetahui tempat pengambilan bibit secara lebih mudah.



Gambar 3.9. Antarmuka *Nursery Location*

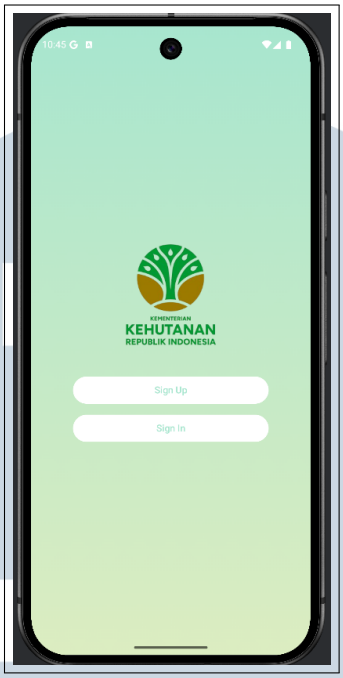
Gambar 3.7 menampilkan desain antarmuka *Nursery Location* dengan tata letak yang sederhana dan informatif. Informasi mengenai lokasi persemaian ditampilkan secara terstruktur, sehingga memudahkan pengguna untuk mengenali area persemaian melalui tampilan visual yang bersih dan mudah dipahami.

D Implementasi Aplikasi *Mobile*

Subbagian ini menjelaskan hasil implementasi dari antarmuka aplikasi *mobile* yang dikembangkan selama masa magang. Setiap tampilan dirancang dan diimplementasikan dengan memperhatikan prinsip kemudahan penggunaan, estetika visual, serta keselarasan dengan identitas Kementerian Kehutanan.

D.1 Halaman *Welcome Screen*

Halaman pertama yang akan dijelaskan adalah tampilan awal ketika pengguna membuka aplikasi.

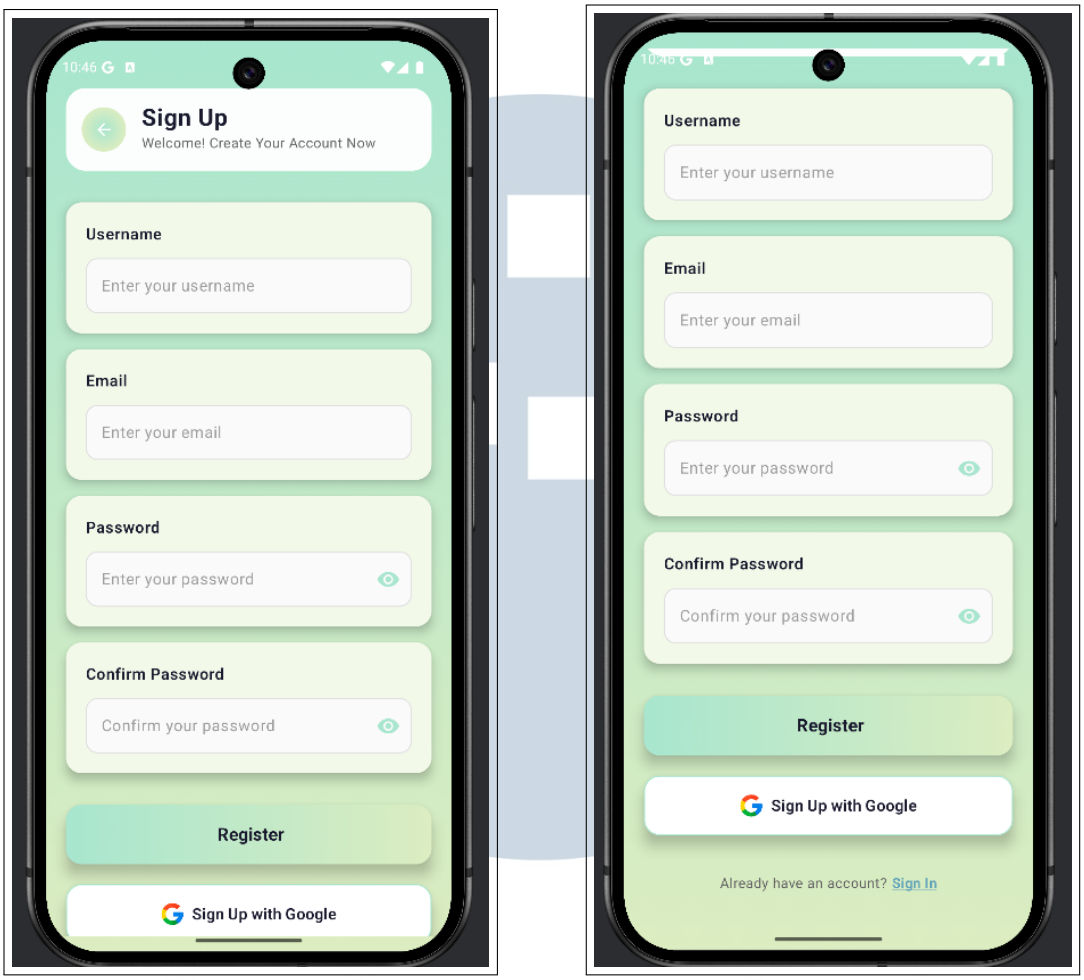


Gambar 3.10. Tampilan Implementasi Aplikasi: Halaman *Welcome Screen*

Gambar 3.10 menunjukkan tampilan *Welcome Screen* pada aplikasi, yang menyajikan dua tombol utama, yaitu *Sign In* dan *Sign Up*. Tampilan ini akan muncul ketika pengguna belum melakukan proses masuk (*Sign In*) ataupun pendaftaran akun (*Sign Up*), sehingga menjadi pintu awal sebelum pengguna dapat mengakses fitur-fitur utama dalam aplikasi.

D.2 Halaman *Sign Up*

Setelah halaman awal, pengguna diarahkan ke halaman *Sign Up* jika belum memiliki akun. Halaman ini merupakan bagian penting untuk proses pendaftaran.



Gambar 3.11. Tampilan Implementasi Aplikasi: Halaman *Sign Up*

Gambar 3.11 merupakan tampilan halaman *Sign Up* yang telah diimplementasikan secara langsung ke dalam bentuk aplikasi *mobile*. Halaman ini digunakan oleh pengguna baru yang belum memiliki akun sebagai langkah awal bagi pengguna baru sebelum mengakses fitur utama aplikasi.

Halaman ini memuat sejumlah *field input* yang perlu dilengkapi oleh pengguna, yaitu:

- *Username*: untuk membuat identitas unik pengguna,
- *Email*: sebagai data kontak dan autentikasi,
- *Password* dan *Confirm Password*: sebagai verifikasi keamanan akun.

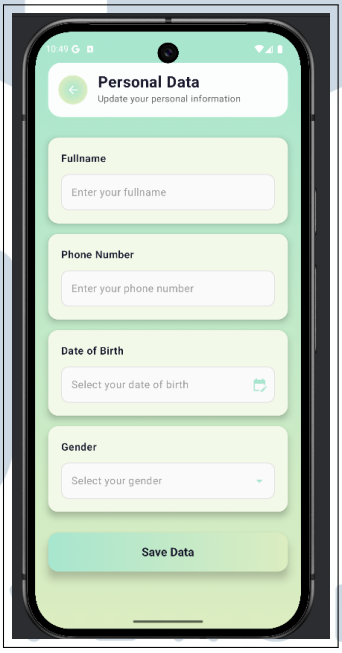
Selain itu, terdapat dua tombol, yaitu tombol *Register* untuk mendaftar secara manual, serta tombol *Sign Up with Google* sebagai alternatif pendaftaran yang lebih praktis dengan menggunakan akun Google.

Tampilan antarmuka ini menggunakan desain gradasi warna hijau ke biru muda sebagai latar belakang. Warna ini dipilih untuk mencerminkan konsep aplikasi yang berhubungan dengan lingkungan dan penghijauan, sehingga menciptakan kesan yang segar, alami, dan sesuai dengan identitas visual Kementerian Kehutanan.

Desain *UI* juga mengutamakan kesederhanaan dan keterbacaan, dengan elemen-elemen *form* yang tersusun secara rapi dan responsif. Implementasi ini menunjukkan bahwa halaman *Sign Up* tidak hanya berfungsi sebagai gerbang awal aplikasi, tetapi juga dirancang dengan mempertimbangkan pengalaman pengguna (*user experience*) yang baik.

D.3 Halaman *Personal Data*

Setelah proses pendaftaran, pengguna akan diarahkan ke halaman *Personal Data* untuk melengkapi informasi pribadi yang dibutuhkan oleh sistem.



Gambar 3.12. Tampilan Implementasi Aplikasi: Halaman *Personal Data*

Gambar 3.12 menunjukkan tampilan halaman *Personal Data* yang telah diimplementasikan dalam aplikasi. Halaman ini berfungsi untuk mengumpulkan data pribadi pengguna setelah proses pendaftaran selesai. Pengisian data ini menjadi bagian penting dalam sistem karena digunakan untuk keperluan identifikasi dan personalisasi layanan aplikasi.

Adapun komponen yang terdapat pada halaman ini meliputi:

- *Fullname*: untuk mencatat nama lengkap pengguna,
- *Phone Number*: sebagai informasi kontak yang valid,
- *Date of Birth*: digunakan untuk keperluan segmentasi data pengguna berdasarkan usia,
- *Gender*: sebagai data demografis tambahan.

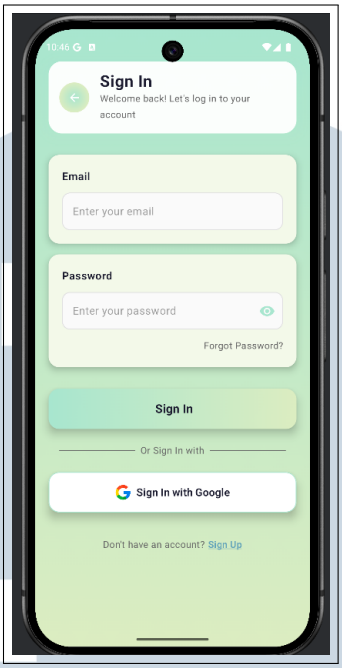
Pengguna diminta untuk melengkapi semua informasi tersebut, kemudian menekan tombol *Save Data* untuk menyimpan data ke sistem.

Dari sisi desain, halaman ini menggunakan warna latar belakang gradasi hijau ke biru muda yang seragam dengan halaman lainnya. Warna tersebut dipilih untuk mencerminkan identitas aplikasi yang mengusung tema lingkungan dan penghijauan. Selain itu, elemen *form* dirancang secara minimalis dengan fokus pada kemudahan *input* dan keterbacaan.

Halaman *Personal Data* ini bertujuan untuk memastikan bahwa aplikasi memiliki data pengguna yang lengkap dan valid.

D.4 Halaman *Sign In*

Bagi pengguna yang telah memiliki akun, aplikasi menyediakan halaman *Sign In* untuk masuk ke sistem.



Gambar 3.13. Tampilan Implementasi Aplikasi: Halaman *Sign In*

Gambar 3.13 menampilkan halaman *Sign In* yang merupakan salah satu bagian penting dalam implementasi aplikasi. Pada halaman ini, pengguna yang sudah terdaftar dapat mengakses aplikasi dengan memasukkan informasi akun yang dimiliki.

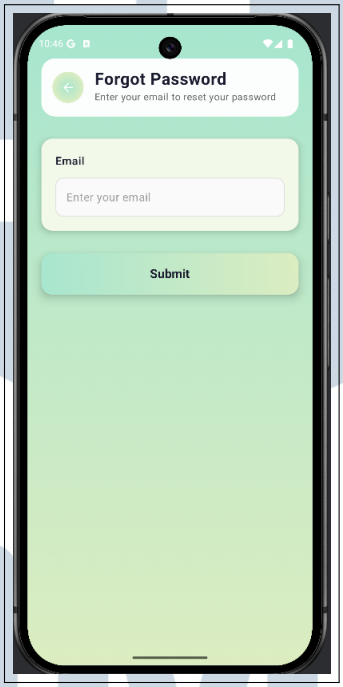
Pada tampilan ini terdapat beberapa komponen utama, yaitu:

- *Email Field*: untuk memasukkan alamat email yang telah digunakan saat proses pendaftaran.
- *Password Field*: untuk memasukkan kata sandi akun pengguna.
- Tombol *Sign In*: berfungsi untuk memproses data autentikasi dan mengarahkan pengguna ke halaman utama aplikasi.
- Tautan lupa kata sandi (*Forgot Password*): memberikan opsi pemulihan akun jika pengguna lupa kata sandinya.
- Tombol *Sign In with Google*: memberikan alternatif login yang lebih cepat melalui akun Google.
- Tautan *Sign Up*: ditampilkan bagi pengguna yang belum memiliki akun untuk diarahkan ke halaman pendaftaran.

Dari segi desain, halaman ini menggunakan gradasi warna hijau muda yang selaras dengan identitas visual aplikasi secara keseluruhan, yang merepresentasikan konsep ramah lingkungan dan penghijauan. Tata letak antarmuka dibuat bersih dan intuitif untuk memudahkan pengguna dalam proses login.

D.5 Halaman *Forgot Password*

Setelah pengguna mengakses halaman *Sign In*, tersedia opsi pemulihan kata sandi apabila pengguna lupa password.



Gambar 3.14. Tampilan Implementasi Aplikasi: Halaman *Forgot Password*

Gambar 3.14 memperlihatkan halaman *Forgot Password* yang merupakan bagian dari sistem pemulihan akun. Halaman ini dirancang untuk memudahkan pengguna yang lupa kata sandi, dengan menyediakan fitur pengiriman tautan reset melalui email.

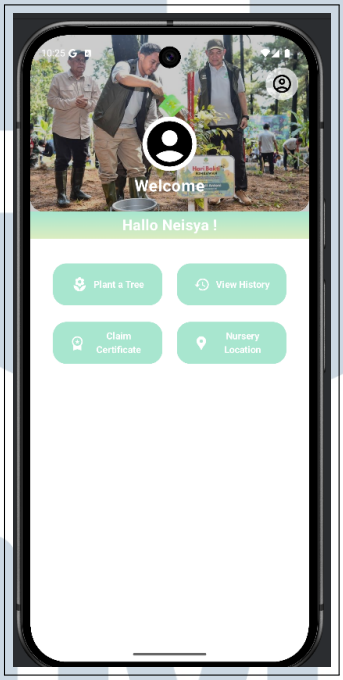
Tampilan antarmuka halaman ini terdiri dari komponen berikut:

- *Input Email*: pengguna diminta untuk memasukkan alamat email yang terdaftar pada sistem.
- Tombol *Submit*: tombol ini digunakan untuk mengirimkan permintaan reset password ke sistem. Setelah menekan tombol ini, sistem akan mengirimkan instruksi pemulihan ke email yang dimasukkan.

Dari sisi desain, halaman ini mengadopsi palet warna hijau gradasi muda yang konsisten dengan halaman lainnya, memberikan kesan natural dan tenang yang merepresentasikan tema aplikasi yang berkaitan dengan penghijauan dan lingkungan.

D.6 Halaman *Home*

Setelah berhasil login, pengguna diarahkan ke halaman utama, yaitu halaman *Home* yang menjadi pusat aktivitas dalam aplikasi.



Gambar 3.15. Tampilan Implementasi Aplikasi: Halaman *Home*

Gambar 3.15 menunjukkan tampilan halaman *Home* dari aplikasi *Penghijauan* yang telah berhasil diimplementasikan. Halaman ini merupakan pusat navigasi utama bagi pengguna setelah berhasil melakukan login.

Tampilan halaman ini terdiri dari beberapa elemen penting, antara lain:

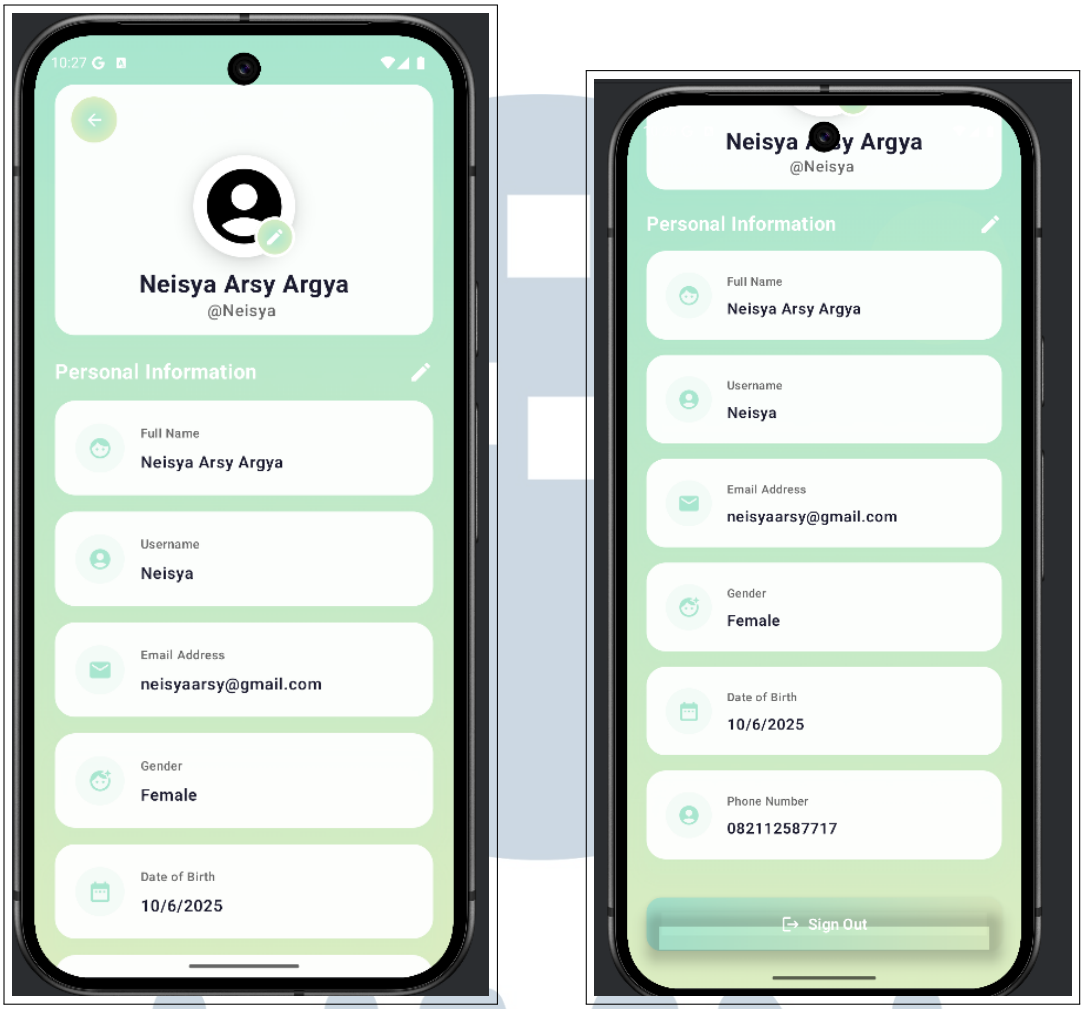
- Header dengan Foto Kegiatan Penghijauan: menampilkan suasana kegiatan penanaman pohon sebagai bentuk visualisasi semangat program penghijauan.
- Profil Pengguna: menampilkan ikon profil dan sapaan “Hallo [Nama Pengguna]” secara personal, dalam hal ini “Hallo Neisya!”, untuk meningkatkan interaksi yang bersifat personal.

- Navigasi Utama dalam Bentuk Tombol:
  - *Plant a Tree* — untuk mencatat aktivitas penanaman pohon.
  - *View History* — untuk melihat riwayat penanaman pohon yang telah dilakukan.
  - *Claim Certificate* — untuk mengklaim sertifikat partisipasi penghijauan.
  - *Nursery Location* — untuk melihat lokasi pembibitan pohon di Indonesia.

Dari segi tampilan, antarmuka ini mengusung kombinasi warna hijau muda dan putih yang menciptakan kesan alami, sejalan dengan tema kehutanan dan pelestarian lingkungan. Elemen seperti ikon dan tombol didesain secara minimalis namun tetap intuitif, sehingga mempermudah pengguna dalam mengakses fitur utama dengan cepat dan efisien. Halaman ini berfungsi sebagai pusat kendali interaktif yang tidak hanya menyajikan informasi, tetapi juga responsif terhadap kebutuhan pengguna dalam mendukung upaya pelestarian lingkungan secara digital.

D.7 Halaman *Profile*

Pengguna juga dapat mengakses halaman *Profile* untuk melihat dan mengelola informasi pribadi mereka.



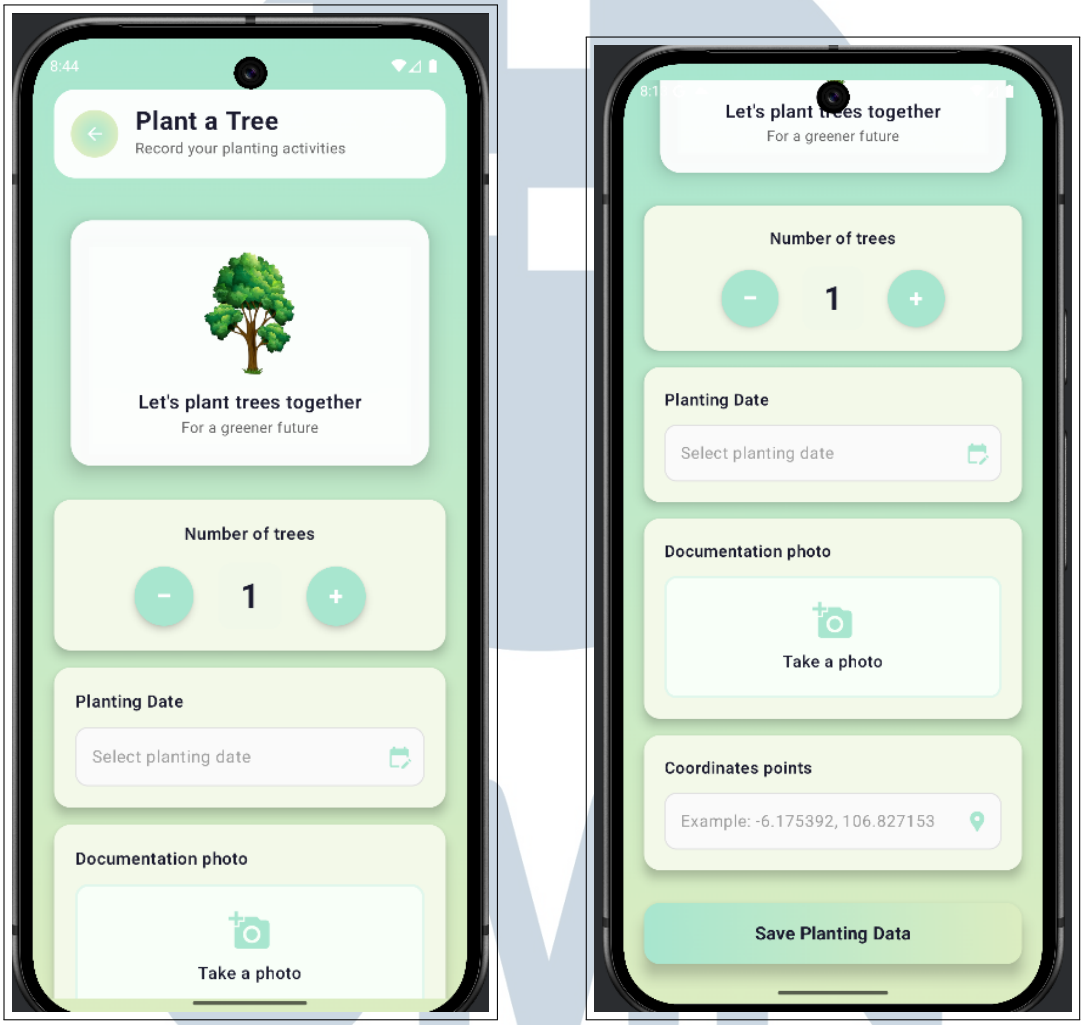
Gambar 3.16. Tampilan Implementasi Aplikasi: Halaman *Profile*

Gambar 3.16 menunjukkan implementasi dari halaman *Profile* dalam aplikasi *mobile*. Halaman ini menampilkan informasi pribadi pengguna yang telah disimpan sebelumnya, seperti nama lengkap, *username*, alamat *email*, jenis kelamin, tanggal lahir, dan nomor telepon. Terdapat ikon pensil di bagian atas yang berfungsi sebagai tombol untuk mengedit informasi.

Selain itu, pengguna juga dapat melakukan *Sign Out* dari akun melalui tombol yang tersedia di bagian bawah layar. Seluruh komponen didesain dengan tampilan yang bersih dan minimalis, menggunakan nuansa warna hijau pastel yang konsisten dengan halaman-halaman lainnya dalam aplikasi. Warna ini dipilih untuk merepresentasikan nilai-nilai ramah lingkungan sesuai dengan tujuan aplikasi, yaitu mendukung kegiatan *penghijauan*.

D.8 Halaman *Plant a Tree*

Fitur utama dalam aplikasi ini adalah halaman *Plant a Tree*, di mana pengguna mencatat kegiatan penanaman yang telah mereka lakukan.



Gambar 3.17. Tampilan Implementasi Aplikasi: Halaman *Plant a Tree*

Gambar 3.17 memperlihatkan tampilan halaman *Plant a Tree* pada aplikasi yang dikembangkan. Fitur ini merupakan komponen utama dari aplikasi karena berfungsi sebagai media input pengguna dalam mendata kegiatan penanaman pohon.

Pengguna dapat mengisi informasi penting terkait kegiatan penanaman, meliputi:

- Jumlah pohon yang ditanam

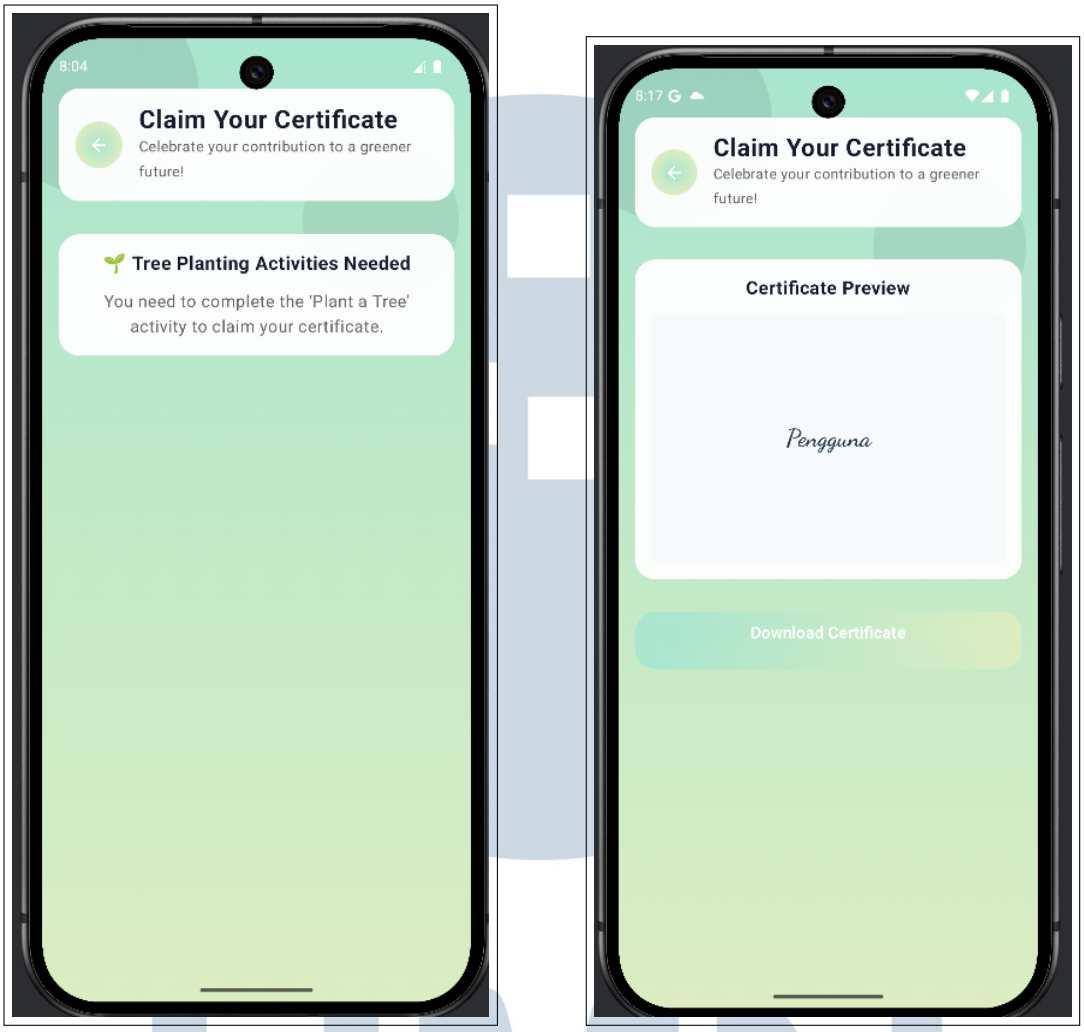
- Tanggal penanaman
- Foto dokumentasi penanaman
- Titik koordinat lokasi penanaman

Antarmuka dirancang agar intuitif dan mudah digunakan. Elemen-elemen input disusun secara vertikal dan jelas, dilengkapi dengan label yang informatif dan tombol aksi seperti *"Take a photo"* dan *"Save Planting Data"*. Tujuan dari fitur ini adalah untuk mendigitalisasi proses dokumentasi *penghijauan* agar data dapat tersimpan dengan baik, akurat, dan terdokumentasi secara *real-time*.

Warna latar belakang halaman menggunakan gradasi hijau muda yang memberikan kesan alami dan menenangkan, selaras dengan tema aplikasi yang berfokus pada pelestarian lingkungan.

D.9 Halaman *Claim Certificate*

Fitur selanjutnya yang tersedia dalam aplikasi adalah halaman *Claim Certificate*, yang berfungsi untuk memberikan apresiasi dalam bentuk sertifikat digital.



Gambar 3.18. Tampilan Implementasi Aplikasi: Halaman *Claim Certificate*

Gambar 3.18 menunjukkan tampilan halaman *Claim Your Certificate* pada aplikasi. Halaman ini bertujuan untuk memberikan apresiasi kepada pengguna yang telah berkontribusi dalam kegiatan penanaman pohon.

Fitur ini memungkinkan pengguna mengunduh sertifikat digital sebagai bentuk dokumentasi dan penghargaan atas aksi *penghijauan* yang telah dilakukan. Namun, sebelum dapat mengklaim sertifikat, pengguna diwajibkan menyelesaikan proses penanaman pohon terlebih dahulu.

Tampilan halaman dibagi menjadi dua kondisi:

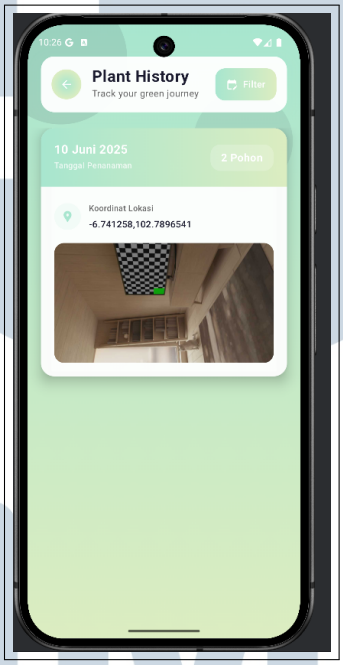
- Jika pengguna belum menyelesaikan penanaman, akan ditampilkan notifikasi berupa pesan informatif.
- Jika penanaman telah dilakukan, maka sistem akan menampilkan pratinjau

sertifikat beserta tombol *Download Certificate*.

Desain antarmuka tetap mempertahankan warna gradasi hijau muda yang natural dan serasi dengan tema pelestarian lingkungan. Elemen antarmuka pengguna dirancang sederhana dan komunikatif agar mudah dipahami pengguna.

D.10 Halaman *History*

Pengguna dapat memantau seluruh kegiatan penanaman pohon yang telah dilakukan melalui halaman *History*.



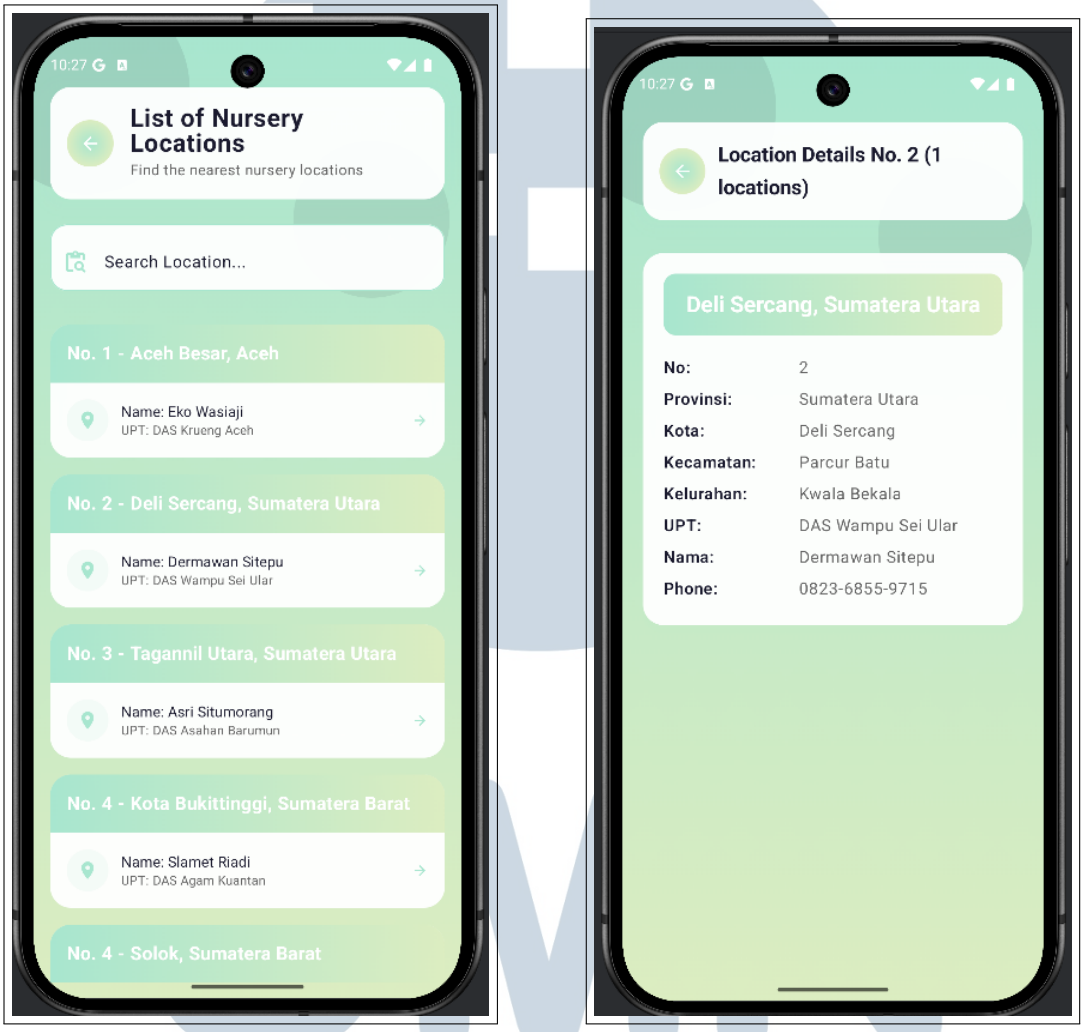
Gambar 3.19. Tampilan Implementasi Aplikasi: Halaman *History*

Gambar 3.19 menunjukkan tampilan halaman *Plant History* pada aplikasi. Halaman ini memungkinkan pengguna untuk melihat riwayat aktivitas penanaman pohon yang telah dilakukan, lengkap dengan informasi tanggal, jumlah pohon, koordinat lokasi, dan foto dokumentasi.

Penggunaan tampilan visual seperti foto dan data koordinat membantu menciptakan sistem *monitoring* yang transparan dan terintegrasi. Antarmuka dirancang dengan nuansa hijau yang lembut untuk memperkuat kesan ramah lingkungan dan selaras dengan tema aplikasi secara keseluruhan.

D.11 Halaman *Nursery Location*

Halaman *Nursery Location* menyediakan informasi penting mengenai lokasi-lokasi persemaian yang dapat diakses oleh pengguna.



Gambar 3.20. Tampilan Implementasi Aplikasi: Halaman *Nursery Location*

Gambar 3.20 memperlihatkan halaman *Nursery Location* pada aplikasi. Halaman ini menyajikan daftar lokasi persemaian beserta informasi lengkap dari masing-masing lokasi yang meliputi nama daerah, nama penanggung jawab, hingga nomor kontak. Informasi ditampilkan dalam dua tampilan terpisah, yaitu daftar lokasi dan halaman detail.

Fitur ini dirancang untuk memudahkan pengguna dalam menemukan lokasi persemaian terdekat sekaligus menyediakan akses langsung kepada informasi kontak yang relevan. Desain halaman dibuat konsisten dengan tampilan halaman

lain, dengan menggunakan gradasi hijau sebagai warna utama yang memperkuat identitas visual aplikasi berbasis lingkungan.

E Kode Aplikasi *Mobile*

Bagian ini menyajikan cuplikan kode program dari implementasi aplikasi *mobile* menggunakan Kotlin dan *Jetpack Compose*, tanpa menampilkan antarmuka visual. Cuplikan kode difokuskan pada logika fitur dan integrasi autentikasi menggunakan *Firebase*.

E.1 Halaman Welcome

Halaman *Welcome* merupakan halaman awal yang ditampilkan ketika pengguna membuka aplikasi. Tampilan ini dibangun menggunakan *Jetpack Compose*, sebuah toolkit modern dari Android untuk membangun antarmuka pengguna secara deklaratif [4].

Desain halaman ini memanfaatkan komponen `Box` dan `Column` untuk mengatur tata letak elemen secara terpusat dan vertikal. Latar belakang diberikan efek gradasi vertikal dari dua warna hijau muda, yaitu `Color(0xFFA8E6CF)` dan `Color(0xFFDCEDC1)`, untuk memberikan nuansa yang alami dan menyegarkan.

Salah satu elemen utama pada halaman ini adalah gambar logo yang dianimasikan secara terus-menerus. Animasi dilakukan dengan memanfaatkan fungsi `animateFloatAsState`, yang menghasilkan efek rotasi 360 derajat setiap dua detik. Efek ini membantu menciptakan tampilan visual yang dinamis dan menarik perhatian pengguna. Berikut ini cuplikan kode untuk animasi gambar:

```
val rotation by animateFloatAsState(
    targetValue = 360f,
    animationSpec = infiniteRepeatable(
        animation = tween(durationMillis = 2000, easing =
    LinearEasing),
        repeatMode = RepeatMode.Restart
    )
)
Image(
    painter = painterResource(id = R.drawable.kemenhut),
    modifier = Modifier.rotate(rotation)
)
```

Kode 3.4: Animasi Rotasi Gambar Logo

Selain itu, terdapat dua tombol utama yang berfungsi untuk navigasi, yaitu tombol `Sign Up` dan `Sign In`. Kedua tombol ini memiliki latar belakang putih dengan bentuk melengkung (rounded corners) dan teks berwarna hijau muda. Ketika tombol ditekan, fungsi navigasi akan dijalankan untuk berpindah ke halaman login atau registrasi. Berikut cuplikan kode untuk tombol navigasi:

```
Button(onClick = { onNavigateToRegister() }) {
    Text("Sign Up", color = Color(0xFFA8E6CF))
}
Button(onClick = { onNavigateToLogin() }) {
    Text("Sign In", color = Color(0xFFA8E6CF))
}
```

Kode 3.5: Navigasi Tombol Sign Up dan Sign In

Secara keseluruhan, halaman *Welcome* ini berfungsi sebagai pintu masuk utama aplikasi dan didesain agar ramah pengguna, intuitif, serta menampilkan identitas visual dari aplikasi secara konsisten.

E.2 Halaman Login

Halaman *Login* merupakan komponen penting dalam aplikasi yang memungkinkan pengguna untuk masuk menggunakan email/kata sandi atau akun Google. Dibangun dengan *Jetpack Compose*, serta terintegrasi dengan Firebase Authentication untuk proses login yang aman.

Tata letak halaman memanfaatkan `Box` dengan latar belakang gradasi vertikal dari `Color(0xFFA8E6CF)` ke `Color(0xFFDCEDC1)`, memberikan tampilan visual yang menarik. Konten diatur secara vertikal menggunakan `Column` yang dapat digulir dan terpusat.

Elemen utama dalam halaman ini mencakup:

- Kolom Input: Terdiri dari `OutlinedTextField` untuk email dan kata sandi. Fitur ikon pada kolom kata sandi memungkinkan pengguna mengatur visibilitasnya.

```
OutlinedTextField(
    value = password,
    onValueChange = { password = it },
    visualTransformation = if (isPasswordVisible)
    VisualTransformation.None
                          else PasswordVisualTransformation
    (),
```

```
    trailingIcon = {
        IconButton(onClick = { isPasswordVisible = !
    isPasswordVisible }) {
            Icon(
                painter = painterResource(
                    id = if (isPasswordVisible) R.drawable.
    ic_visibility_off
                        else R.drawable.ic_visibility
                ),
                contentDescription = "Toggle password
    visibility"
            )
        }
    }
)

```

Kode 3.6: Kolom Kata Sandi dengan Ikon Visibilitas

- Tombol Autentikasi:Tersedia `Button` untuk login menggunakan email/kata sandi dan `OutlinedButton` untuk login Google. Saat proses autentikasi berlangsung, indikator pemuatan ditampilkan.

```
Button(
    onClick = {
        if (email.isBlank() || password.isBlank()) {
            errorMessage = "Email and password cannot be left
    blank"
        } else {
            isLoading = true
            errorMessage = null
            auth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener { task ->
                    isLoading = false
                    if (task.isSuccessful) {
                        onLoginClick()
                    } else {
                        val exception = task.exception
                        errorMessage = when (exception) {
                            is
    FirebaseAuthInvalidUserException -> "Email not registered"
                            is
    FirebaseAuthInvalidCredentialsException -> "Incorrect
    password"
```

```
                                else -> "An error occurred during
    login. Incorrect password."
                            }
                        }
                    }
                }
            }
        ) {
            if (isLoading) {
                CircularProgressIndicator()
                Text("Logging in...")
            } else {
                Text("Sign In")
            }
        }
```

Kode 3.7: Logika Login Email/Kata Sandi

- Penanganan Kesalahan: Sebuah `Card` menampilkan pesan kesalahan jika input tidak valid atau terjadi kegagalan autentikasi.
- Tautan Navigasi: Tautan teks seperti ”Lupa Kata Sandi?” dan ”Daftar” mengarahkan pengguna ke halaman pemulihan akun atau registrasi.

Firebase Authentication menangani login menggunakan email/kata sandi serta Google Sign-In melalui `FirebaseAuth` dan `GoogleSignInClient`. Fungsi `handleGoogleSignInResult` mengelola hasil login Google dan memproses ID token untuk autentikasi:

```
private fun handleGoogleSignInResult(
    data: Intent,
    auth: FirebaseAuth,
    onLoginSuccess: () -> Unit,
    onErrorMessage: (String) -> Unit
) {
    try {
        val account = GoogleSignIn.getSignedInAccountFromIntent(
    data)
                        .getResult(ApiException::class.java)
        val idToken = account?.idToken ?: return onErrorMessage(
            "Google Sign-In failed: ID token is null"
        )
        val credential = GoogleAuthProvider.getCredential(idToken,
     null)
```

```
        auth.signInWithCredential(credential)
            .addOnCompleteListener { authTask ->
                if (authTask.isSuccessful) {
                    onLoginSuccess()
                } else {
                    onErrorMessage("Authentication failed: ${
    authTask.exception?.message}")
                }
            }
    } catch (e: ApiException) {
        onErrorMessage("Google Sign-In error: ${e.message}")
    }
}
```

Kode 3.8: Penanganan Login Google

`LoginScreen` dirancang modular dengan parameter lambda untuk menangani navigasi dan aksi, meningkatkan fleksibilitas dan reusabilitas kode. Selain itu, anotasi `@Preview` disediakan untuk membantu pengembangan dan pengujian antarmuka secara visual.

Halaman ini mengintegrasikan desain antarmuka yang intuitif dengan sistem autentikasi yang aman, memberikan pengalaman login yang lancar dan profesional kepada pengguna.

E.3 Halaman Forgot Password

Halaman *Forgot Password* memungkinkan pengguna untuk mengatur ulang kata sandi mereka dengan memasukkan alamat email. Pengguna memasukkan email melalui `OutlinedTextField`, dan proses pengiriman email pengaturan ulang dilakukan dengan tombol `Submit`. Pesan kesalahan atau keberhasilan ditampilkan dalam `Card`.

Fitur utama adalah pengiriman email pengaturan ulang kata sandi melalui Firebase Authentication. Berikut adalah cuplikan kode inti untuk pengiriman email:

```
Button(onClick = {
    if (email.isBlank()) {
        errorMessage = "Email cannot be empty"
    } else {
        isLoading = true
        errorMessage = null
        auth.sendPasswordResetEmail(email)
            .addOnCompleteListener { task ->
```

```
                isLoading = false
                if (task.isSuccessful) {
                    isSuccess = true
                    onPasswordResetSuccess()
                } else {
                    errorMessage = "Failed to send verification
    code. Check your email!"
                }
            }
    }
}) {
    Text("Submit")
}
```

Kode 3.9: Pengiriman Email Pengaturan Ulang Kata Sandi

Halaman ini juga menyediakan tombol kembali untuk navigasi ke halaman sebelumnya. Berikut adalah kode untuk tombol kembali:

```
Icon(
    painter = painterResource(id = R.drawable.ic_back),
    contentDescription = "Back",
    tint = Color.White,
    modifier = Modifier.size(20.dp)
)
```

Kode 3.10: Navigasi Tombol Kembali

Secara keseluruhan, halaman *Forgot Password* dirancang untuk memberikan pengalaman pengguna yang sederhana dan intuitif dalam mengatur ulang kata sandi, dengan integrasi Firebase Authentication untuk keamanan dan fungsionalitas pengiriman email.

E.4 Halaman Register

Halaman *Register* merupakan halaman untuk mendaftarkan pengguna baru ke dalam aplikasi. Halaman ini dibangun menggunakan *Jetpack Compose*.Tata letaknya mencakup kolom input untuk username, email, password, dan konfirmasi password, serta tombol untuk registrasi dan autentikasi dengan Google. Untuk memberikan pengalaman pengguna yang responsif, pesan error ditampilkan jika terjadi kesalahan selama proses registrasi.

Fitur utama halaman ini adalah registrasi pengguna melalui dua metode: email/password dan Google Sign-In. Untuk registrasi dengan email, aplikasi

memvalidasi input untuk memastikan semua kolom terisi dan password sesuai, kemudian menggunakan `FirebaseAuth.createUserWithEmailAndPassword` untuk membuat akun. Data pengguna seperti username dan email disimpan ke *Firestore*. Berikut adalah cuplikan kode inti untuk registrasi dengan email:

```
fun registerUser() {
    if (username.isBlank() || email.isBlank() || password.isBlank() || confirmPassword.isBlank()) {
        errorMessage = "All fields must be filled"
        return
    }
    if (password != confirmPassword) {
        errorMessage = "Passwords do not match"
        return
    }
    auth.createUserWithEmailAndPassword(email, password)
        .addOnSuccessListener { authResult ->
            val userId = authResult.user?.uid ?: return@addOnSuccessListener
            val userData = mapOf("username" to username, "email" to email)
            db.collection("users").document(userId).set(userData)
        }
}
```

Kode 3.11: Registrasi dengan Email dan Password

Untuk Google Sign-In, aplikasi menggunakan `GoogleSignInClient` untuk memulai proses autentikasi. Token ID dari Google dikirim ke *FirebaseAuth* untuk membuat akun, dan data pengguna disimpan ke Firestore. Berikut cuplikan kode untuk autentikasi Google:

```
val launcher = rememberLauncherForActivityResult(ActivityResultContracts.StartActivityForResult()) { result ->
    val task = GoogleSignIn.getSignedInAccountFromIntent(result.data)
    try {
        val account = task.getResult(ApiException::class.java)
        firebaseAuthWithGoogle(
            idToken = account.idToken!!,
            auth = auth,
            db = db,
            onSuccess = { isRegisterSuccess = true },
            onError = { errorMessage = it }
```

```
        )
    } catch (e: ApiException) {
        errorMessage = "Google sign-in failed"
    }
}
```

Kode 3.12: Autentikasi dengan Google Sign-In

Halaman ini juga menyediakan tombol navigasi ke halaman login untuk pengguna yang sudah memiliki akun. Berikut adalah kode untuk tombol navigasi:

```
TextButton(onClick = { onNavigateToLogin() }) {
    Text("Already have an account? Sign In")
}
```

Kode 3.13: Navigasi ke Halaman Login

Secara keseluruhan, halaman *Register* dirancang untuk memfasilitasi pendaftaran pengguna dengan antarmuka yang sederhana, fungsional, dan terintegrasi dengan sistem autentikasi Firebase, memastikan proses registrasi yang aman dan efisien.

E.5 Halaman Personal Data

Halaman *Personal Data* memungkinkan pengguna untuk mengisi dan memperbarui informasi pribadi setelah registrasi.Tata letak mencakup kolom input untuk nama lengkap, nomor telepon, tanggal lahir, dan jenis kelamin, serta tombol untuk menyimpan data. Pesan error ditampilkan dalam kartu berwarna merah muda jika ada kolom yang belum diisi.

Fitur utama adalah pengisian dan penyimpanan data pribadi pengguna. Kolom tanggal lahir menggunakan `DatePickerDialog` untuk memilih tanggal, sedangkan jenis kelamin menggunakan `DropdownMenu` dengan opsi ”Male”, ”Female”, dan ”Other”. Data yang diisi divalidasi untuk memastikan semua kolom terisi sebelum disimpan ke *Firestore*. Berikut adalah cuplikan kode inti untuk menyimpan data pengguna:

```
Button(onClick = {
    if (phoneNumber.isBlank() || birthDate.isBlank() || gender.
    isBlank()) {
        errorMessage = "All fields must be filled"
    } else {
        currentUser?.let { user ->
            val userData = mapOf(
```

```
                "phoneNumber" to phoneNumber,
                "fullname" to fullname,
                "birthDate" to birthDate,
                "gender" to gender
            )
            db.collection("users").document(user.uid).update(
    userData)
        }
    }
}) {
    Text("Save Data")
}
```

Kode 3.14: Penyimpanan Data Pribadi ke Firestore

Untuk memilih tanggal lahir, `DatePickerDialog` ditampilkan saat pengguna mengklik ikon kalender. Berikut adalah kode untuk pemilihan tanggal:

```
val datePickerDialog = DatePickerDialog(
    context,
    { _, year, month, day -> birthDate = "$day/${month + 1}/$year"
    },
    calendar.get(Calendar.YEAR),
    calendar.get(Calendar.MONTH),
    calendar.get(Calendar.DAY_OF_MONTH)
)
OutlinedTextField(
    value = birthDate,
    onValueChange = {},
    readOnly = true,
    trailingIcon = {
        IconButton(onClick = { datePickerDialog.show() }) {
            Icon(painterResource(id = R.drawable.ic_calendar),
    contentDescription = "Calendar")
        }
    }
)
```

Kode 3.15: Pemilihan Tanggal Lahir

Halaman ini juga menyediakan tombol kembali untuk navigasi ke halaman sebelumnya. Berikut adalah kode untuk tombol kembali:

```
Icon(
    painter = painterResource(id = R.drawable.ic_back),
    contentDescription = "Back",
```

```
    modifier = Modifier.clickable { onBackClick() }
)
```

Kode 3.16: Navigasi Tombol Kembali

Secara keseluruhan, halaman *Personal Data* dirancang untuk memudahkan pengguna mengisi informasi pribadi dengan Firestore untuk penyimpanan data yang aman.

E.6 Halaman Profile

Halaman *Profile* menampilkan informasi pribadi pengguna dan memungkinkan pengeditan data serta penggantian foto profil. Informasi pengguna seperti nama lengkap, username, email, jenis kelamin, tanggal lahir, dan nomor telepon ditampilkan dalam kartu profil. Pengguna dapat mengedit nama lengkap dan username melalui dialog, mengganti foto profil, atau keluar dari akun melalui tombol *Sign Out*.

Fitur utama adalah pengambilan dan pembaruan data pengguna dari *Firestore* serta pembaruan foto profil. Data pengguna diambil saat halaman dimuat menggunakan `Firestore.collection("users").document(userId).get()`. Berikut adalah cuplikan kode inti untuk pengambilan data pengguna:

```
LaunchedEffect(userId) {
    if (userId.isNotEmpty()) {
        firestore.collection("users").document(userId).get().
    addOnSuccessListener { document ->
            if (document.exists()) {
                username = document.getString("username") ?: ""
                fullname = document.getString("fullname") ?: ""
                email = document.getString("email") ?: ""
                gender = document.getString("gender") ?: ""
                birthDate = document.getString("birthDate") ?: ""
                phoneNumber = document.getString("phoneNumber") ?:
     ""
                profileImageUrl = document.getString("
    profileImageUrl")
            }
        }
    }
}
```

Kode 3.17: Pengambilan Data Pengguna dari Firestore

Pengguna dapat mengedit nama lengkap dan username melalui dialog yang menyimpan perubahan ke Firestore. Berikut adalah kode untuk pembaruan data profil:

```
EditProfileDialog(
    currentFullname = fullname,
    currentUsername = username,
    onDismissRequest = { showEditDialog = false },
    onSave = { newFullname, newUsername ->
        val updates = mapOf("fullname" to newFullname, "username" to newUsername)
        firestore.collection("users").document(userId).update(updates)
    }
)
```

Kode 3.18: Pembaruan Data Profil

Pengguna juga dapat mengganti foto profil menggunakan `ActivityResultContracts.GetContent`. Gambar disimpan ke penyimpanan internal dan path-nya diperbarui di Firestore. Berikut adalah kode untuk pembaruan foto profil:

```
val imagePickerLauncher = rememberLauncherForActivityResult(ActivityResultContracts.GetContent()) { uri ->
    uri?.let {
        val savedPath = copyUriToInternalStorage(context, it, "$userId.jpg")
        if (savedPath != null) {
            firestore.collection("users").document(userId)
                .update("profileImageUrl", savedPath)
        }
    }
}
```

Kode 3.19: Pembaruan Foto Profil

Tombol *Sign Out* menampilkan dialog konfirmasi sebelum pengguna keluar menggunakan `FirebaseAuth.signOut()`. Berikut adalah kode untuk proses logout:

```
TextButton(onClick = { dialogIsOpen.value = true }) {
    Text("Sign Out")
}
LogoutDialog(
```

```
    onDismissRequest = { dialogIsOpen.value = false },
    onConfirmation = {
        auth.signOut()
        onLogoutClick()
    },
    dialogTitle = "Confirm to Sign Out",
    dialogText = "Are you sure you want to sign out?",
    icon = painterResource(R.drawable.frame_99)
)
```

Kode 3.20: Proses Logout

Secara keseluruhan, halaman *Profile* dirancang untuk memberikan akses mudah ke informasi pengguna, mendukung pengeditan data, dan memastikan pengelolaan akun yang aman melalui integrasi dengan Firebase.

E.7 Halaman Plant a Tree

Halaman *Plant a Tree* memungkinkan pengguna untuk merekam aktivitas penanaman pohon, termasuk jumlah pohon, tanggal penanaman, koordinat lokasi, dan dokumentasi foto. Pengguna dapat memilih tanggal penanaman menggunakan `DatePickerDialog`, memasukkan koordinat lokasi secara manual, dan mengambil foto menggunakan kamera. Data disimpan ke Firestore dalam format Base64 untuk foto, bersama dengan informasi lainnya.

Fitur utama adalah penyimpanan data penanaman pohon ke Firestore. Data divalidasi untuk memastikan tanggal, koordinat, dan foto telah diisi, lalu disimpan sebagai dokumen dengan ID pengguna. Berikut adalah cuplikan kode inti untuk penyimpanan data:

```
Button(onClick = {
    if (plantingDate.isEmpty() || locationCoordinates.isEmpty() ||
     capturedImageUri == null) {
        return@Button
    }
    capturedImageUri?.let { uri ->
        val inputStream = context.contentResolver.openInputStream(
   uri)
        val bitmap = BitmapFactory.decodeStream(inputStream)
        val outputStream = ByteArrayOutputStream()
        bitmap.compress(Bitmap.CompressFormat.JPEG, 50,
   outputStream)
```

```
        val base64Image = Base64.encodeToString(outputStream.
    toByteArray(), Base64.DEFAULT)
        val treeData = hashMapOf(
            "treeCount" to treeCount,
            "plantingDate" to plantingDate,
            "locationCoordinates" to locationCoordinates,
            "photoBase64" to base64Image,
            "planterId" to userId,
            "completed" to true
        )
        firestore.collection("trees").document(userId).set(
    treeData)
    }
}) {
    Text("Simpan Data Penanaman")
}
```

Kode 3.21: Penyimpanan Data Penanaman ke Firestore

Pengambilan foto menggunakan *CameraX* dengan `ImageCapture` untuk menyimpan gambar ke file sementara. Berikut adalah kode untuk pengambilan foto:

```
CameraPreviewScreen(
    onImageCaptured = { uri -> capturedImageUri = uri },
    onClose = { showCamera = false }
)
```

Kode 3.22: Pengambilan Foto dengan CameraX

Tanggal penanaman dipilih melalui `DatePickerDialog`, yang ditampilkan saat pengguna mengklik ikon kalender. Berikut adalah kode untuk pemilihan tanggal:

```
val datePickerDialog = DatePickerDialog(
    context,
    { _, year, month, day -> plantingDate = "$day/${month + 1}/
    $year" },
    calendar.get(Calendar.YEAR),
    calendar.get(Calendar.MONTH),
    calendar.get(Calendar.DAY_OF_MONTH)
)
OutlinedTextField(
    value = plantingDate,
    onValueChange = {},
    readOnly = true,
    trailingIcon = {
```

```
        IconButton(onClick = { datePickerDialog.show() }) {
            Icon(painterResource(id = R.drawable.ic_calendar), "
    Calendar")
        }
    }
)
```

Kode 3.23: Pemilihan Tanggal Penanaman

Halaman ini juga menyediakan tombol kembali untuk navigasi ke halaman sebelumnya. Berikut adalah kode untuk tombol kembali:

```
Icon(
    imageVector = Icons.Default.ArrowBack,
    contentDescription = "Back",
    modifier = Modifier.clickable { navController.popBackStack() }
)
```

Kode 3.24: Navigasi Tombol Kembali

Secara keseluruhan, halaman *Plant a Tree* dirancang untuk memfasilitasi pencatatan aktivitas penanaman pohon.

E.8 Halaman Claim Certificate

Halaman *Claim Certificate* memungkinkan pengguna untuk mengklaim sertifikat digital setelah menyelesaikan aktivitas penanaman pohon. Pengguna dapat melihat pratinjau sertifikat dengan nama mereka dan mengunduhnya sebagai PDF jika telah menyelesaikan aktivitas penanaman pohon.

Fitur utama adalah pembuatan dan penyimpanan sertifikat dalam format PDF. Data pengguna (nama lengkap) dan status penanaman diambil dari Firestore. Sertifikat dibuat sebagai `Bitmap` menggunakan `Canvas` dan dikonversi ke PDF menggunakan `PdfDocument`. Berikut adalah cuplikan kode inti untuk pembuatan dan penyimpanan sertifikat:

```
Button(onClick = {
    captureCertificateToBitmap(context, fullname) { bitmap ->
        val uri = saveBitmapToPdf(context, bitmap, fullname)
    }
}) {
    Text("Download Certificate")
}
```

Kode 3.25: Pembuatan dan Penyimpanan Sertifikat PDF

Data pengguna dan status penanaman diambil dari Firestore saat halaman dimuat. Berikut adalah kode untuk pengambilan data:

```
LaunchedEffect(userId) {
    if (userId.isNotEmpty()) {
        val userDocument = firestore.collection("users").document(userId).get().await()
        fullname = userDocument.getString("fullname")?.takeIf { it.isNotEmpty() } ?: "Pengguna"
        val plantDocument = firestore.collection("trees").document(userId).get().await()
        hasPlantedTree = plantDocument.exists() && plantDocument.getBoolean("completed") == true
    }
}
```

Kode 3.26: Pengambilan Data dari Firestore

Halaman ini juga menyediakan tombol kembali untuk navigasi ke halaman sebelumnya. Berikut adalah kode untuk tombol kembali:

```
Icon(
    painter = painterResource(id = R.drawable.ic_back),
    contentDescription = "Back",
    modifier = Modifier.clickable { onBackClick() }
)
```

Kode 3.27: Navigasi Tombol Kembali

Secara keseluruhan, halaman *Claim Certificate* dirancang untuk memberikan pengalaman pengguna yang intuitif dalam mengklaim sertifikat, dengan integrasi Firestore untuk validasi dan fungsi pembuatan PDF untuk dokumentasi resmi.

E.9 Halaman History

Halaman *History* menampilkan riwayat aktivitas penanaman pohon pengguna, termasuk jumlah pohon, tanggal penanaman, koordinat lokasi, dan foto dokumentasi. Pengguna dapat memfilter riwayat berdasarkan bulan menggunakan dialog pemilih bulan. Riwayat penanaman ditampilkan dalam `LazyColumn` untuk efisiensi rendering, dengan setiap item menggunakan `Card` untuk menampilkan informasi dan foto dalam format Base64.

Fitur utama adalah pengambilan data penanaman pohon dari Firestore, dengan opsi filter berdasarkan bulan. Data diambil berdasarkan `planterId`

pengguna dan status `completed`. Berikut adalah cuplikan kode inti untuk pengambilan data:

```
LaunchedEffect(selectedDate, currentUserId) {
    val query = if (selectedDate != null) {
        val calendar = Calendar.getInstance().apply { time = selectedDate!! }
        val startOfMonth = calendar.apply { set(Calendar.DAY_OF_MONTH, 1) }.timeInMillis
        val endOfMonth = calendar.apply { set(Calendar.DAY_OF_MONTH, getActualMaximum(Calendar.DAY_OF_MONTH)) }.timeInMillis
        firestore.collection("trees")
            .whereEqualTo("planterId", currentUserId)
            .whereEqualTo("completed", true)
            .whereGreaterThanOrEqualTo("timestamp", startOfMonth)
            .whereLessThanOrEqualTo("timestamp", endOfMonth)
            .orderBy("timestamp", Query.Direction.DESCENDING)
    } else {
        firestore.collection("trees")
            .whereEqualTo("planterId", currentUserId)
            .whereEqualTo("completed", true)
            .orderBy("timestamp", Query.Direction.DESCENDING)
    }
    query.get().addOnSuccessListener { result ->
        treeItems = result.documents.mapNotNull { doc ->
            TreeItem(
                id = doc.id,
                treeCount = doc.getLong("treeCount")?.toInt() ?: 0,
                plantingDate = doc.getString("plantingDate") ?: "",
                locationCoordinates = doc.getString("locationCoordinates") ?: "",
                photoBase64 = doc.getString("photoBase64") ?: "",
                timestamp = doc.get("timestamp") as? Long ?: 0L,
                planterId = doc.getString("planterId") ?: ""
            )
        }
        isLoading = false
    }
}
```

Kode 3.28: Pengambilan Data dari Firestore

Fitur filter bulan ditampilkan melalui dialog dengan `LazyColumn` untuk memilih bulan. Berikut adalah kode untuk dialog pemilih bulan:

```
if (showMonthPicker) {
    Dialog(onDismissRequest = { showMonthPicker = false }) {
        Column {
            Text("Select Month")
            LazyColumn {
                items(months) { month ->
                    Text(
                        text = month,
                        modifier = Modifier
                            .fillMaxWidth()
                            .clickable {
                                val calendar = Calendar.
    getInstance()
                                calendar.set(Calendar.MONTH,
    months.indexOf(month))
                                selectedDate = calendar.time
                                showMonthPicker = false
                            }
                            .padding(16.dp)
                    )
                }
            }
        }
    }
}
```

Kode 3.29: Dialog Pemilih Bulan

Halaman ini juga menyediakan tombol kembali untuk navigasi ke halaman sebelumnya. Berikut adalah kode untuk tombol kembali:

```
Icon(
    painter = painterResource(id = R.drawable.ic_back),
    contentDescription = "Kembali",
    modifier = Modifier.clickable { navController.popBackStack() }
)
```

Kode 3.30: Navigasi Tombol Kembali

Secara keseluruhan, halaman *History* dirancang untuk memberikan akses mudah ke riwayat penanaman pohon pengguna, dengan filter bulan yang intuitif dan integrasi Firestore untuk data yang akurat.

E.10 Halaman Nursery List dan Nursery Detail

Halaman *Nursery List* menampilkan daftar lokasi pembibitan pohon yang dapat dicari berdasarkan provinsi, kota, kecamatan, desa/kelurahan, nama, kontak, atau UPT. Data lokasi bersumber dari `nurseryLocations`, sebuah daftar statis yang diimpor dari `data.NurseryLocation`. Halaman *Nursery Detail* menampilkan informasi lengkap untuk lokasi tertentu berdasarkan nomor identifikasi (`no`).

Pencarian dilakukan melalui `OutlinedTextField`, dan hasil ditampilkan dalam `LazyColumn` untuk efisiensi rendering. Setiap item lokasi menggunakan `Card` dengan animasi klik. Halaman *Nursery Detail* menampilkan detail dalam `Column` yang dapat digulir, dengan setiap lokasi ditampilkan dalam `Card`.

Fitur utama *Nursery List* adalah pencarian dan navigasi ke detail lokasi. Pencarian memfilter `nurseryLocations` berdasarkan kata kunci. Berikut adalah cuplikan kode inti untuk pencarian dan daftar lokasi:

```
var searchQuery by remember { mutableStateOf("") }
val filteredLocations = if (searchQuery.isEmpty()) {
    nurseryLocations.distinctBy { it.no to it.nama }
} else {
    nurseryLocations.filter {
        it.provinsi.contains(searchQuery, ignoreCase = true) ||
        it.kota.contains(searchQuery, ignoreCase = true) ||
        it.kecamatan.contains(searchQuery, ignoreCase = true) ||
        it.desaKel.contains(searchQuery, ignoreCase = true) ||
        it.nama.contains(searchQuery, ignoreCase = true) ||
        it.contactPerson.contains(searchQuery, ignoreCase = true)
    ||
        it.upt.contains(searchQuery, ignoreCase = true)
    }.distinctBy { it.no to it.nama }
}
LazyColumn {
    items(filteredLocations.sortedBy { it.no }) { location ->
        NurseryListItem(location = location) {
            navController.navigate("nursery_detail/${location.no
    }")
        }
    }
}
```

Kode 3.31: Pencarian dan Daftar Lokasi

Fitur utama *Nursery Detail* adalah menampilkan informasi lengkap lokasi

berdasarkan `no`. Data diambil dari `nurseryLocations` dengan filter. Berikut adalah kode inti untuk menampilkan detail:

```
val locations = nurseryLocations.filter { it.no == no }
if (locations.isEmpty()) {
    Text("Location not found for No: $no")
} else {
    Column {
        locations.forEach { location ->
            Card {
                Column {
                    Text("${location.kota}, ${location.provinsi}")
                    DetailRow(label = "No", value = location.no.
    toString())
                    DetailRow(label = "Provinsi", value = location
    .provinsi)
                    DetailRow(label = "Kota", value = location.
    kota)
                    DetailRow(label = "Kecamatan", value =
    location.kecamatan)
                    DetailRow(label = "Kelurahan", value =
    location.desaKel)
                    if (location.upt.isNotEmpty()) DetailRow(label
     = "UPT", value = location.upt)
                    if (location.nama.isNotEmpty()) DetailRow(
    label = "Nama", value = location.nama)
                    if (location.contactPerson.isNotEmpty())
    DetailRow(label = "Contact", value = location.contactPerson)
                }
            }
        }
    }
}
```

Kode 3.32: Menampilkan Detail Lokasi

Kedua halaman menyediakan tombol kembali untuk navigasi ke halaman sebelumnya. Berikut adalah kode untuk tombol kembali:

```
Icon(
    imageVector = Icons.Default.ArrowBack,
    contentDescription = "Back",
    modifier = Modifier.clickable { navController.popBackStack() }
)
```

Kode 3.33: Navigasi Tombol Kembali

Secara keseluruhan, halaman *Nursery List* dan *Nursery Detail* dirancang untuk memudahkan pengguna menemukan dan melihat detail lokasi pembibitan pohon.

3.4 Kendala dan Solusi yang Ditemukan

3.4.1 Kendala

Dalam proses pengembangan aplikasi *Penghijauan* yang dilakukan di lingkungan Kementerian Kehutanan, terdapat beberapa kendala yang dihadapi, khususnya terkait keterbatasan penggunaan layanan berbayar. Hal ini dikarenakan proyek yang dikembangkan merupakan bagian dari kegiatan magang di instansi pemerintahan, sehingga diusahakan untuk menggunakan layanan gratis (free-tier) tanpa biaya tambahan.

- Kendala penyimpanan dan penampilan gambar : Kendala utama yang dihadapi adalah dalam hal penyimpanan dan penampilan gambar hasil dokumentasi penanaman pohon oleh pengguna. Awalnya direncanakan menggunakan *Firebase Storage*, namun layanan ini memerlukan biaya tambahan setelah melewati batas penggunaan gratis (free quota).

- Kendala akses titik koordinat otomatis : Kendala lain terjadi pada fitur pencatatan titik koordinat lokasi penanaman pohon. Idealnya, aplikasi dapat secara otomatis mendeteksi lokasi geografis pengguna melalui layanan *Geolocation API*. Namun, sebagian besar layanan dengan akurasi tinggi memerlukan langganan atau biaya tertentu yang tidak sesuai dengan batasan anggaran.

- Kendala dalam pelaksanaan secara individu : Tantangan lain adalah seluruh proses perancangan dan pengembangan aplikasi dilakukan secara individu. Tidak adanya rekan diskusi atau pembagian tugas membuat semua aspek, mulai dari desain hingga pengujian aplikasi, harus ditangani sendiri. Hal ini menambah beban kerja dan memperlambat proses saat terjadi kendala teknis.

3.4.2 Solusi

Untuk mengatasi kendala-kendala yang ada, diterapkan beberapa solusi alternatif agar pengembangan aplikasi tetap berjalan sesuai tujuan.

- Solusi untuk penyimpanan dan penampilan gambar : Format gambar diubah menjadi *Base64* sebelum disimpan ke *Firebase Firestore*, yang masih termasuk dalam batas layanan gratis. *Base64* mengubah data gambar menjadi string teks yang bisa disimpan dalam dokumen Firestore, lalu didekodekan kembali menjadi gambar saat ditampilkan di aplikasi. Pendekatan ini memang kurang efisien, tetapi cukup efektif untuk kebutuhan ringan.
- Solusi untuk akses titik koordinat : Fitur pencatatan lokasi dilakukan secara manual, yaitu pengguna diminta untuk menginput titik koordinat (latitude dan longitude) secara mandiri. Solusi ini tetap memungkinkan pencatatan lokasi secara fungsional meskipun kurang praktis dan kurang akurat dibandingkan metode otomatis.
- Solusi untuk pelaksanaan individu : Dibuat *daily timeline* atau jadwal kerja harian untuk menjaga ritme kerja tetap terstruktur dan mencegah penundaan. Selain itu, pekerjaan didokumentasikan secara berkala agar memudahkan proses debugging serta membantu saat melakukan revisi atau pengembangan lebih lanjut.