LAMPIRAN

Lampiran 1. Surat Kesepakatan Bimbingan

SURAT KESEPAKATAN BIMBINGAN SKRIPSI

Kami yang bertanda tangan di bawah ini :

Pihak Pertama

Nama	: Koandres
NIM	: 20090123
Program Studi	: Sarjana Terapan Teknik Informatika

Pihak Kedua

Nama	: Muhammad Fikri Hidayattullah, S.T., M.Kom.
Status	: Dosen Tetap
NIDN	: 0623108801
Jabatan Fungsional	: Lektor
Pangkat/Golongan	: Penata Tk. I/III-d

Pada hari ini Kamis tanggal 14 Maret 2024 telah terjadi sebuah kesepakatan bahwa Pihak Kedua bersedia menjadi Pembimbing I/~~II~~ Skripsi Pihak Pertama dengan syarat Pihak Pertama berjanji akan menyelesaikan Skripsi dalam waktu maksimal 3 bulan (awal bulan Juni 2024). Jika syarat tersebut tidak terpenuhi, maka Pihak Kedua berhak untuk tidak melanjutkan proses bimbingan. Adapun waktu dan tempat pelaksanaan disepakati antar pihak.

Demikian kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Skripsi

Tegal, 14 Maret 2024

Pihak Pertama	Pihak Kedua
Koandres	Muhammad Fikri Hidayattullah, S.T., M.Kom.

Mengetahui
Ketua Program Studi Sarjana Terapan Teknik Informatika

Dyah Apriliani, S.T., M.Kom
NIPY. 09.015.225

SURAT KESEPAKATAN BIMBINGAN SKRIPSI

Kami yang bertanda tangan di bawah ini :

Pihak Pertama

Nama	: Koandres
NIM	: 20090123
Program Studi	: Sarjana Terapan Teknik Informatika

Pihak Kedua

Nama	: Sharfina Febbi Handayani, M.Kom
Status	: Dosen Tetap
NIDN	: 0617029201
Jabatan Fungsional	: Asisten Ahli
Pangkat/Golongan	: Penata Muda Tk I/IIIb

Pada hari ini Kamis tanggal 14 Maret 2024 telah terjadi sebuah kesepakatan bahwa Pihak Kedua bersedia menjadi Pembimbing I/II Skripsi Pihak Pertama dengan syarat Pihak Pertama bersedia bimbingan secara rutin untuk menyelesaikan Skripsi minimal 3 kali dalam sebulan, apabila dalam waktu 40 hari pertama setelah kesepakatan ini dibuat pihak pertama tidak konsisten untuk menyanggupi maka pihak kedua berhak membatalkan kesepakatan ini dan tidak melanjutkan untuk membimbing pihak pertama dengan tidak akan memberikan rekomendasi untuk mengikuti ujian skripsi. Adapun waktu dan tempat pelaksanaan disepakati antar pihak.

Demikian kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Skripsi

Tegal, 14 Maret 2024

Pihak Pertama	Pihak Kedua
Koandres	Sharfina Febbi Handayani, M.Kom

Mengetahui
Ketua Program Studi Sarjana Terapan Teknik Informatika

Dyah Apriliani, S.T., M.Kom
NIPY. 09.015.225

Lampiran 2. Surat Keterangan Penelitian

POLITEKNIK HARAPAN BERSAMA
the true vocational campus

Sarjana Terapan Teknik Informatika

Nomor : 42.03/TI.PHB/VII/2024
Lampiran : -
Hal : Permohonan Izin Observasi
Kepada Yth. : **Pengurus Komunitas Pendakian Tegal**
di Tegal

Dengan hormat, mahasiswa dengan identitas berikut ini:

nama : Koandres
NIM : 20090123
prodi : Sarjana Terapan Teknik Informatika

Bermaksud melakukan penelitian untuk keperluan Skripsi dengan judul "**VIRTUAL ASSISTANT BERBASIS MOBILE UNTUK MENGEDUKASI PENDAKI MENGGUNAKAN CHATBOT**". Kami memohon Bapak/Ibu memberikan izin kepada mahasiswa yang bersangkutan agar memperoleh data, keterangan, dan bahan yang diperlukan.

Demikian permohonan ini disampaikan, atas perhatian kami ucapkan terima kasih.

Tegal, 9 Juli 2024
Ka. Prodi S.Tr. Teknik Informatika,

Dyah Aprilliani, S.T., M.Kom
NIPY : 09.015.225

Jl. Mataram No. 9 Kota Tegal 52143, Jawa Tengah, Indonesia.
(0283)352000
informatika@poltektegal.ac.id
poltektegal.ac.id

Lampiran 3. Surat Pernyataan Pengajuan HKI

SURAT PERNYATAAN

Yang bertanda tangan di bawah ini, pemegang hak cipta:

1. Nama : Koandres
   Kewarganegaraan : Indonesia
   Alamat : Jl. Ababil Indah no.68 RT/RW 010/010, Kecamatan Tegal Selatan, Kota Tegal, Provinsi Jawa Tengah
2. Nama : Muhammad Fikri Hidayattullah, S.T., M.Kom.
   Kewarganegaraan : Indonesia
   Alamat : Jl. Glatik No.68, Kelurahan Randugunting, Kecamatan Tegal Selatan, Kota Tegal, Provinsi Jawa Tengah
3. Nama : Sharfina Febbi Handayani, S.Kom., M.Kom.
   Kewarganegaraan : Indonesia
   Alamat : Jl. Brayan 1 RT 02 RW 06 Desa Grobog Kulon Kecamatan Pangkah Kabupaten Tegal

Dengan ini menyatakan bahwa:

1. Karya Cipta yang saya mohonkan:
   Berupa : Program Komputer
   Berjudul : SiMun: *Virtual Assistant Software* Pendakian Gunung
   - Tidak meniru dan tidak sama secara esensial dengan Karya Cipta milik pihak lain atau obyek kekayaan intelektual lainnya sebagaimana dimaksud dalam Pasal 68 ayat (2);
   - Bukan merupakan Ekspresi Budaya Tradisional sebagaimana dimaksud dalam Pasal 38;
   - Bukan merupakan Ciptaan yang tidak diketahui penciptanya sebagaimana dimaksud dalam Pasal 39;
   - Bukan merupakan hasil karya yang tidak dilindungi Hak Cipta sebagaimana dimaksud dalam Pasal 41 dan 42;
   - Bukan merupakan Ciptaan seni lukis yang berupa logo atau tanda pembeda yang digunakan sebagai merek dalam perdagangan barang/jasa atau digunakan sebagai lambang organisasi, badan usaha, atau badan hukum sebagaimana dimaksud dalam Pasal 65 dan;
   - Bukan merupakan Ciptaan yang melanggar norma agama, norma susila, ketertiban umum, pertahanan dan keamanan negara atau melanggar peraturan perundang-undangan sebagaimana dimaksud dalam Pasal 74 ayat (1) huruf d Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.
2. Sebagai pemohon mempunyai kewajiban untuk menyimpan asli contoh ciptaan yang dimohonkan dan harus memberikan apabila dibutuhkan untuk kepentingan penyelesaian sengketa perdata maupun pidana sesuai dengan ketentuan perundang-undangan.

3. Karya Cipta yang saya mohonkan pada Angka 1 tersebut di atas tidak pernah dan tidak sedang dalam sengketa pidana dan/atau perdata di Pengadilan.

4. Dalam hal ketentuan sebagaimana dimaksud dalam Angka 1 dan Angka 3 tersebut di atas saya / kami langgar, maka saya / kami bersedia secara sukarela bahwa:
   a. permohonan karya cipta yang saya ajukan dianggap ditarik kembali; atau
   b. Karya Cipta yang telah terdaftar dalam Daftar Umum Ciptaan Direktorat Hak Cipta, Direktorat Jenderal Hak Kekayaan Intelektual, Kementerian Hukum Dan Hak Asasi Manusia R.I dihapuskan sesuai dengan ketentuan perundang-undangan yang berlaku.
   c. Dalam hal kepemilikan Hak Cipta yang dimohonkan secara elektronik sedang dalam berperkara dan/atau sedang dalam gugatan di Pengadilan maka status kepemilikan surat pencatatan elektronik tersebut ditangguhkan menunggu putusan Pengadilan yang berkekuatan hukum tetap.

Demikian Surat pernyataan ini saya/kami buat dengan sebenarnya dan untuk dipergunakan sebagaimana mestinya

Tegal, 28 Juni 2024

(Koandres)

(Muhammad Fikri Hidayattullah, S.T., M.Kom.)

(Sharfina Febbi Handayani, S.Kom., M.Kom.)

Lampiran 4. Surat Pengalihan Hak Cipta

SURAT PENGALIHAN HAK CIPTA

Yang bertanda tangan di bawah ini, pemegang hak cipta:

1. Nama : Koandres
   Kewarganegaraan : Indonesia
   Alamat : Jl. Ababil Indah no.25 RT/RW 010/010, Kelurahan Randugunting, Kecamatan Tegal Selatan, Kota Tegal, Provinsi Jawa Tengah
2. Nama : Muhammad Fikri Hidayattullah, S.T., M.Kom.
   Kewarganegaraan : Indonesia
   Alamat : Jl. Glatik No.68, Kelurahan Randugunting, Kecamatan Tegal Selatan, Kota Tegal, Provinsi Jawa Tengah
3. Nama : Sharfina Febbi Handayani, S.Kom., M.Kom.
   Kewarganegaraan : Indonesia
   Alamat : Jl. Brayan 1 RT 02 RW 06 Desa Grobog Kulon Kecamatan Pangkah Kabupaten Tegal

Adalah **Pihak I** selaku pencipta selaku pencipta, dengan ini menyerahkan karya ciptaan saya kepada :

Nama : Pusat Penelitian dan Pengabdian Masyarakat (P3M)
Alamat : Jl. Mataram No. 9 Pesurungan Lor Kota Tegal

Adalah **Pihak II** selaku Pemegang Hak Cipta berupa Program Komputer dengan judul "SiMun: *Virtual Assistant Software* Pendakian Gunung". untuk didaftarkan di Direktorat Hak Cipta dan Desain Industri, Direktorat Jenderal Kekayaan Intelektual, Kementerian Hukum dan Hak Asasi Manusia Republik Indonesia.

Demikianlah surat pengalihan hak ini kami buat, agar dapat dipergunakan sebagaimana mestinya.

Tegal, 28 Juni 2024

Pemegang Hak Cipta
Ketua P3M

(Dr. Aldi Budi Riyanta, S.Si., M.T)

Pencipta

(Koandres)

(Muhammad Fikri Hidayattullah, S.T., M.Kom.)

(Sharfina Febbi Handayani, S.Kom., M.Kom.)

Lampiran 5. Syarat Pengajuan HKI

DAFTAR ISI

DAFTAR GAMBAR

LATAR BELAKANG

Mendaki gunung adalah kegiatan yang diminati oleh berbagai kalangan, namun juga membawa risiko tinggi. Persiapan yang matang sangat diperlukan, termasuk pengetahuan mendalam tentang medan, cuaca, peralatan yang tepat, serta tindakan yang harus diambil dalam berbagai situasi darurat. Sayangnya, banyak pendaki yang kurang memperoleh informasi yang memadai sebelum melakukan pendakian, yang dapat berakibat fatal seperti hipotermia, jatuh dari ketinggian, atau tersesat. Data menunjukkan bahwa kecelakaan di gunung tidak jarang terjadi, dengan contoh konkret seperti 104 kasus kecelakaan di Gunung Rinjani selama periode lima tahun.

Untuk mengatasi tantangan ini, pengembangan aplikasi mobile menjadi solusi yang efektif dalam era digital saat ini. Aplikasi ini tidak hanya menyediakan informasi tentang persiapan pendakian seperti pengetahuan dasar, peralatan yang diperlukan, dan informasi rute, tetapi juga menggunakan teknologi Natural Language Processing (NLP) untuk menyediakan chatbot yang dapat memberikan informasi secara instan dan relevan kepada pengguna. Diharapkan aplikasi ini dapat meningkatkan kesiapan dan edukasi para pendaki gunung sebelum mereka memulai petualangan mereka, serta mengurangi jumlah kecelakaan dan insiden yang terjadi di gunung-gunung Indonesia dalam jangka panjang.

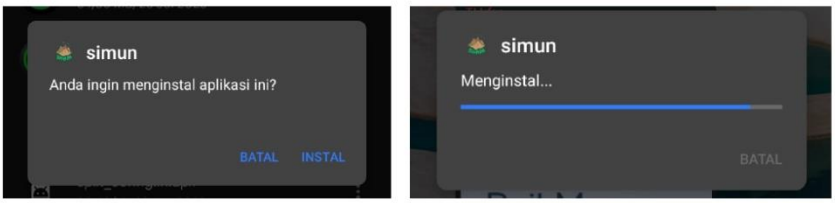
Gambar 1. Logo Aplikasi

TUJUAN DAN MANFAAT

- Tujuan dari pengembangan aplikasi ini adalah untuk meningkatkan kesadaran dan kesiapan pendaki gunung melalui penyediaan informasi yang mudah diakses dan terkini mengenai persiapan pendakian, menggunakan teknologi NLP untuk memfasilitasi interaksi yang efektif melalui chatbot, serta mengembangkan fitur tracking offline untuk meminimalisir risiko tersesat.
- Manfaatnya meliputi pengasahan keahlian penelitian dan analitis bagi penulis, kontribusi sosial dengan meningkatkan keselamatan di kalangan pendaki, referensi berharga bagi penelitian lanjutan di institusi, serta sumber informasi kredibel bagi masyarakat yang ingin meningkatkan keselamatan mereka dalam aktivitas mendaki gunung.

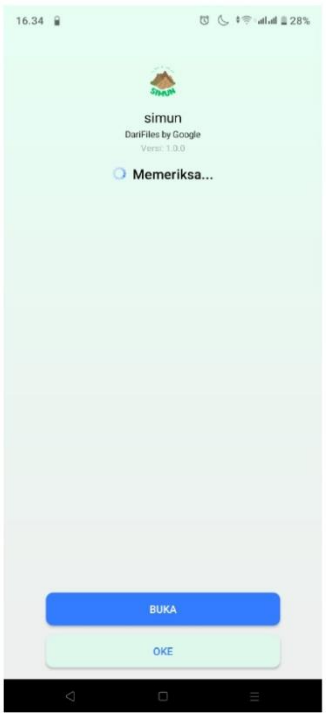
INSTALLASI APLIKASI

1. Buka aplikasi SiMun yang telah di download sebelumn, kemudian pilih instal, kemudian tunggu hingga proses installasi selesai.

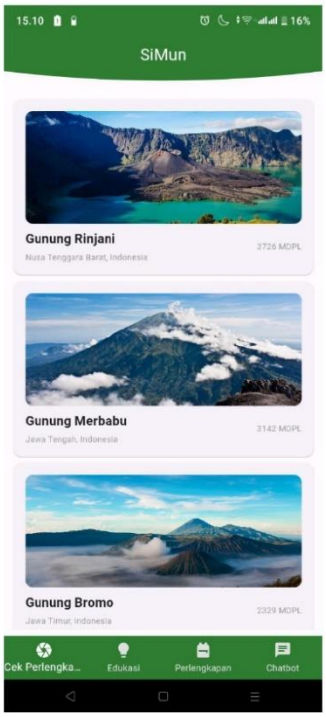


Gambar 2. Cara Installasi Aplikasi

2. Setelah proses installasi selesai, pilih "buka" untuk menjalankan aplikasi.



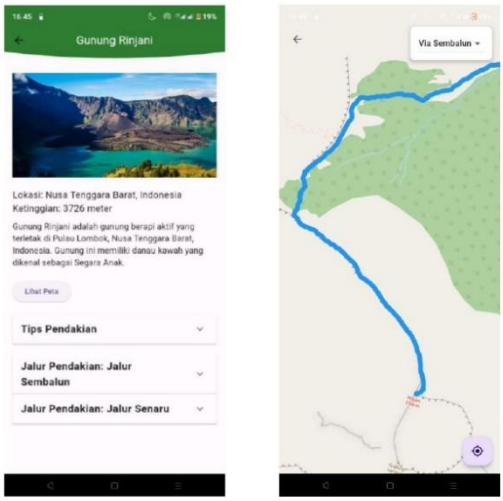
Gambar 3. Buka Aplikasi



Gambar 4. Dashboard

CARA PENGGUNAAN FITUR PADA APLIKASI

1. Pilih gunung yang akan di daki, kemudian terdapat beberapa fitur yang dapat mebantu pengguna salah satunya fitur “Peta” yang dapat digunakan secara offline untuk memandu pengguna.



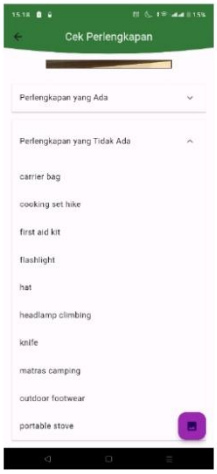
Gambar 5. Halaman Pendakian

2. Buka Fitur Cek Kelengkapan, kemudian pilih icon “galeri” untuk mendeteksi kelengkapan.

Gambar 6. Halaman Cek Kelengkapan

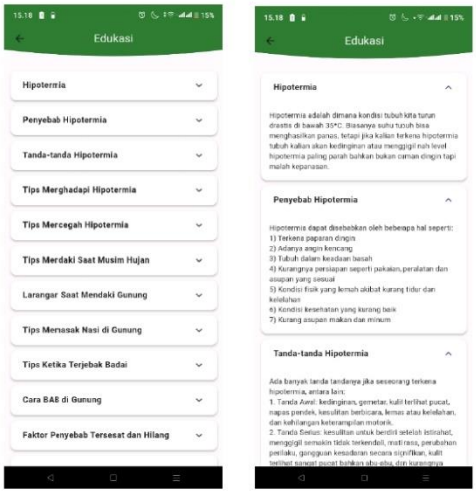
3. “Take a Picture” akan membuka kamera pada hanphone pengguna dan mengambil gambar sedangkan untuk “Select Image” akan mengakses foto dari galeri pengguna.
4. Setelah mengambil gambar, akan muncul perlengkapan yang sudah ada dan tidak ada

Gambar 7. Cek Perlengkapan yang Ada



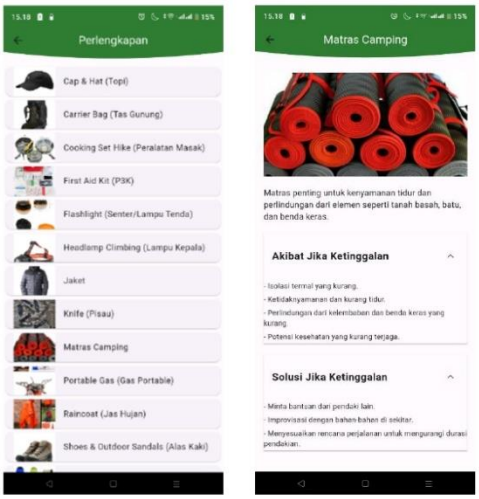
Gambar 8. Cek Perlengkapan yang Tidak Ada

5. Selanjutnya fitur “Edukasi” di mana pengguna dapat membaca pengetahuan tentang seputar pendakian.



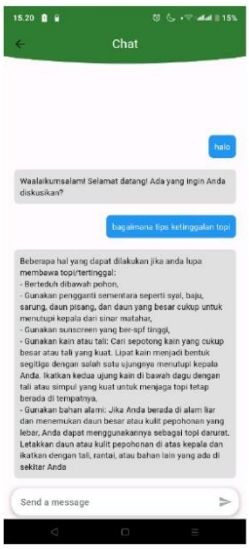
Gambar 9. Halaman Edukasi

6. Fitur ke 3 yaitu “Perlengkapan”, penggguna dapa melihat perlengkapan apa saja yang dibutuhkan pada saat melakukan pendakian.



Gambar 10. Halaman Perlengkapan

7. Fitur terakhir yaitu “Chatbot” di mana pengguna dapat berinteraksi menggunakan chat, kemudian sistem akan meresponnya dengan cepat.



Gambar 11. Halaman Chatbot

TECHNICAL DOCUMENT

SIAP MUNCAK

SIMUN

VIRTUAL ASSISTANT SOFTWARE PENDAKIAN GUNUNG

KOANDRES

MUHAMMAD FIKRI HIDAYATTULLAH, S.T., M.Kom

SHARFINA FEBBI HANDAYANI, S.Kom., M.Kom

DAFTAR ISI

DAFTAR GAMBAR

PENDAHULUAN

SiMun, aplikasi virtual assistant yang dirancang khusus untuk mendukung kegiatan mendaki gunung Anda. Mendaki gunung adalah aktivitas yang menawarkan banyak keindahan dan tantangan, namun juga membawa risiko yang tidak bisa diabaikan. Kesiapan dan pengetahuan yang memadai sangat penting untuk memastikan setiap pendakian berjalan aman dan menyenangkan.

Dalam beberapa tahun terakhir, semakin banyak orang dari berbagai kalangan yang tertarik untuk mendaki gunung. Sayangnya, banyak dari mereka yang melakukan pendakian tanpa persiapan yang memadai, yang sering kali berujung pada situasi darurat. Data menunjukkan bahwa di Gunung Rinjani saja, terdapat 104 kasus kecelakaan dalam periode lima tahun terakhir. Insiden seperti hipotermia, jatuh dari ketinggian, dan tersesat di jalur pendakian adalah beberapa contoh dari risiko yang dapat dihadapi oleh pendaki yang kurang terinformasi.

SiMun hadir sebagai solusi untuk mengatasi tantangan ini. Aplikasi ini dirancang untuk memberikan informasi dan dukungan yang komprehensif kepada para pendaki sebelum dan selama pendakian mereka. Dengan menggunakan teknologi terkini seperti Natural Language Processing (NLP), SiMun menyediakan chatbot yang mampu memberikan informasi secara instan dan relevan mengenai persiapan pendakian, cuaca, medan, dan peralatan yang dibutuhkan.

SPESIFIKASI TEKNIS

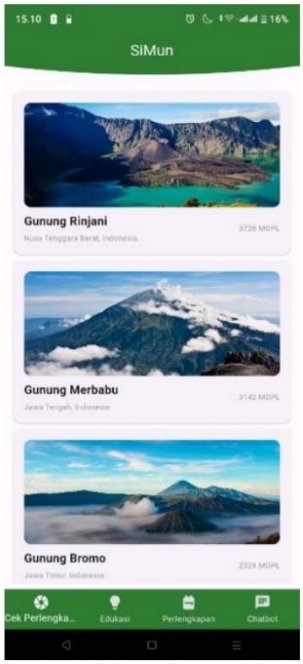
Spesifikasi Teknis meliputi:

1. Modul Pengguna
2. Source Code

Berikut uraian spesifikasi untuk pengembangan aplikasi:

1. Laptop RAM 8GB
2. Visual Studio Code
3. Google Colaboratory
4. Smartphone

Berikut Uraian Spesifikasi Modul



Gambar 1. Halaman Utama Aplikasi

Gambar disamping merupakan tampilan dari aplikasi SiMun yang dapat digunakan oleh pengguna. Aplikasi ini memiliki beberapa fitur seperti Informasi Gunung, Peta, Edukasi (Tips Mendaki), Informasi Perlengkapan, Cek Perlengkapan, dan Chatbot.

SOURCE CODE

Aplikasi SiMun dikembangkan sepenuhnya menggunakan Flutter, sebuah framework open-source yang dirancang oleh Google untuk pengembangan aplikasi mobile. Pendekatan ini memungkinkan pembuatan antarmuka yang dinamis dan efisien untuk platform Android dan iOS.

Tidak seperti aplikasi lain yang mungkin menggunakan framework back-end seperti Flask atau Django untuk pengelolaan logika dan data server-side, SiMun sepenuhnya berbasis Dart, bahasa pemrograman yang menjadi inti dari Flutter. Semua fungsi yang biasanya dikembangkan dalam bahasa lain dan diakses melalui API eksternal, diimplementasikan langsung dalam Dart. Ini mencakup semua operasi logika dan data, yang diintegrasikan secara langsung ke dalam aplikasi tanpa memerlukan lapisan tambahan untuk komunikasi dengan server.

Selain itu, SiMun tidak menggunakan database eksternal untuk penyimpanan data pengguna atau informasi lainnya. Sebagai gantinya, semua data yang diperlukan disimpan secara lokal dalam bentuk file JSON dan/atau teks (txt). Pendekatan ini memanfaatkan in-built resources dari aplikasi, memungkinkan akses data yang cepat dan efisien serta mengurangi ketergantungan pada server dan jaringan.

1. **Konfigurasi pubspec.yaml**

```
dependencies:
  flutter:
    sdk: flutter

  cupertino_icons: ^1.0.6
  tokenizer:
  tflite_flutter:
  stemmer:
  flutter_map:
  latlong2:
  image_picker:
  image:
  geolocator:
  flutter_map_tile_caching:
```

Gambar 2. Konfigurasi pubspec.yaml 1

```
flutter:
  uses-material-design: true
  assets:
    - assets/data/
    - assets/data/rute/
    - assets/images/
```

Gambar 3. Konfigurasi pubspec.yaml 2

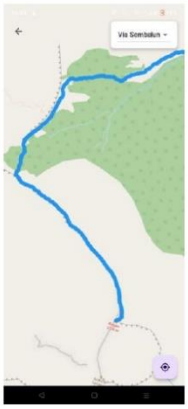
Pubspec.yaml adalah file konfigurasi utama dalam proyek Flutter yang mendefinisikan dependensi dan aset aplikasi. Dependensi termasuk Flutter SDK untuk pengembangan, cupertino_icons untuk ikon gaya iOS, tokenizer untuk tokenisasi teks, tflite_flutter untuk integrasi TensorFlow Lite, stemmer untuk pemrosesan bahasa, flutter_map untuk peta interaktif, latlong2 untuk koordinat geografis, image_picker untuk pilihan gambar, image untuk pemrosesan gambar, dan geolocator untuk akses lokasi GPS. Aset seperti direktori data umum, gambar, dan model machine learning juga dideklarasikan untuk pengelolaan yang efisien dalam aplikasi Flutter.

2. **Konfigurasi Peta**

```
Object? initErr;
try {
  await FMTCObjectBoxBackend().initialise();
} catch (err) {
  initErr = err;
  print(initErr);
}

await FMTCStore('mapStore').manage.create();
```

Gambar 4. Konfigurasi Peta

Dalam script tersebut, variabel initErr digunakan untuk menangkap dan menyimpan informasi kesalahan yang mungkin terjadi selama proses inisialisasi. Pada blok try, objek dari kelas FMTCObjectBoxBackend dibuat dan metode initialise() dipanggil secara asynchronous dengan await. Jika terjadi pengecualian, blok catch menangkap kesalahan tersebut, dengan pesan kesalahan dicetak ke konsol untuk debugging. Setelah blok try-catch, metode create() dipanggil pada objek FMTCStore('mapStore'), bagian dari proses setelah inisialisasi untuk melanjutkan konfigurasi atau penciptaan selanjutnya. Dengan menggunakan try-catch, aplikasi dapat mengelola kesalahan inisialisasi dengan baik sambil memastikan kelangsungan proses berikutnya sesuai rencana. Di bawah ini merupakan tampilan peta yang telah tersimpan cache nya menggunakan baris syntax di atas.



Gambar 5. Tampilan Halaman Peta

3. Informasi Gunung

```
String jsonString = await rootBundle.loadString('assets/data/mountains.json');

List<dynamic> jsonData = json.decode(jsonString);
List<Mountain> mountains =
    List<Mountain>.from(jsonData.map((e) => Mountain.fromJson(e)));
```

Gambar 6. Memuat Data Gunung

Baris kode tersebut digunakan untuk memuat dan mengurai data JSON dari file 'assets/data/mountains.json' dalam aplikasi Flutter. Pertama, await rootBundle.loadString('assets/data/mountains.json') digunakan untuk mengambil konten JSON dari file aset aplikasi. Data JSON yang dimuat diubah menjadi string menggunakan metode loadString dari rootBundle, yang mengelola akses ke file aset.

Selanjutnya, List<dynamic> jsonData = json.decode(jsonString); mengurai string JSON menjadi struktur data Dart menggunakan json.decode, sehingga menghasilkan objek Dart sesuai dengan struktur JSON yang didefinisikan.

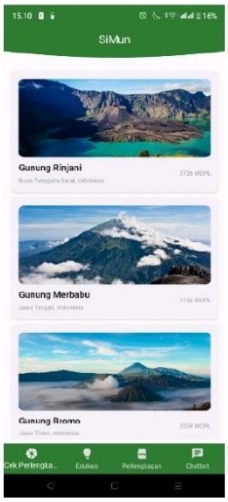
Terakhir, setiap objek dalam jsonData dikonversi menjadi objek Mountain menggunakan metode Mountain.fromJson(e), di mana fromJson adalah metode konversi yang mengubah objek JSON menjadi objek Mountain. Hasilnya adalah daftar Mountain yang berisi data dari file JSON yang dimuat. Adapun detail dari objek Mountain adalah sebagai berikut:

```dart
class Mountain {
  final String nama;
  final String lokasi;
  final int ketinggian;
  final String deskripsi;
  final List<Map<String, dynamic>> tipsPendakian;
  final List<Map<String, dynamic>> jalurPendakian;
  final String foto;
  final double latitude;
  final double longitude;
  final List<String> rute;

  Mountain({
    required this.nama,
    required this.lokasi,
    required this.ketinggian,
    required this.deskripsi,
    required this.tipsPendakian,
    required this.jalurPendakian,
    required this.foto,
    required this.latitude,
    required this.longitude,
    required this.rute,
  });

  factory Mountain.fromJson(Map<String, dynamic> json) {
    return Mountain(
      nama: json['nama'],
      lokasi: json['lokasi'],
      ketinggian: json['ketinggian'],
      deskripsi: json['deskripsi'],
      tipsPendakian: List<Map<String, dynamic>>.from(json['tipsPendakian']),
      jalurPendakian: List<Map<String, dynamic>>.from(json['jalurPendakian']),
      foto: json['foto'],
      latitude: json['latitude'],
      longitude: json['longitude'],
      rute: List<String>.from(json['rute']),
    ); // Mountain
  }
}
```

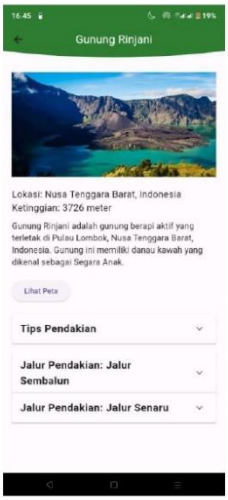
Gambar 7. Kelas Mountain

Kelas Mountain memiliki atribut seperti nama (String), lokasi (String), ketinggian (int), deskripsi (String), tipsPendakian dan jalurPendakian (List<Map<String, dynamic>>), foto (String), latitude dan longitude (double) untuk koordinat geografis, serta rute (List<String>) untuk daftar rute pendakian. Konstruktor utama Mountain membutuhkan semua atribut tersebut untuk membuat objek Mountain.

Metode fromJson adalah metode pabrik yang mengubah data JSON (dalam bentuk Map<String, dynamic>) menjadi objek Mountain. Metode ini menginisialisasi objek Mountain dengan nilai dari JSON untuk setiap atributnya. Di bawah ini merupakan tampilan dari Informasi Gunung yang dimuat menggunakan syntax yang telah dijelaskan di atas.



Gambar 8. Halaman Utama



Gambar 9. Halaman Detail Gunung

4. Edukasi

```
String educationDataString =
    await rootBundle.loadString('assets/data/educations.json');
Map<String, dynamic> educationDataJson = json.decode(educationDataString);
educations = educationDataJson.entries.map((entry) {
  return Education(title: entry.key, description: entry.value['deskripsi']);
}).toList();
```

Gambar 10. Memuat Data Edukasi

Fungsi loadEducationData() dalam aplikasi Flutter bertugas memuat data pendidikan dari file JSON di direktori aset. Pertama, menggunakan await rootBundle.loadString('assets/data/educations.json') untuk mendapatkan konten JSON dari file tersebut menjadi string educationDataString. Kemudian, string JSON diurai menjadi Map<String, dynamic> dengan json.decode(educationDataString), hasilnya disimpan dalam educationDataJson.

Selanjutnya, educationDataJson dipetakan untuk membuat objek Education dari setiap entri menggunakan educationDataJson.entries.map((entry) {...}). Di dalam pemetaan ini, setiap kunci digunakan sebagai title objek Education, sedangkan nilai dari kunci tersebut diambil sebagai description. Contohnya, objek Education dibuat dengan Education(title: entry.key, description: entry.value['deskripsi']).

Hasilnya adalah List<Education> yang disimpan dalam variabel educations. Setelah proses pengolahan selesai, setState(() {...}) dipanggil untuk

mengupdate state aplikasi, dimana isLoading diatur ke false. Ini menandakan bahwa proses memuat data telah selesai, dan aplikasi siap untuk menggunakan data pendidikan yang telah dimuat. Adapun detail dari objek Education adalah sebagai berikut:

```
class Education {
  final String title;
  final String description;

  Education({required this.title, required this.description});

  factory Education.fromJson(Map<String, dynamic> map) {
    return Education(
      title: map.keys.first,
      description: map.values.first['deskripsi'],
    );
  }
}
```

Gambar 11. Kelas Edukasi

Kelas Education merepresentasikan data pendidikan dengan properti title untuk judul dan description untuk deskripsi. Konstruktor Education menggunakan named parameter dengan kata kunci required untuk memastikan title dan description harus disediakan saat membuat objek Education.

Metode factory Education.fromJson(Map<String, dynamic> map) digunakan untuk membuat objek Education dari data dalam bentuk Map<String, dynamic>, biasanya dari representasi JSON. Metode fromJson mengambil judul dari map.keys.first dan deskripsi dari map.values.first['deskripsi'].

Kelas Education memungkinkan aplikasi untuk mengelola data pendidikan secara terstruktur dan menyediakan metode fromJson untuk konversi data JSON ke objek Education dengan mudah, digunakan dalam logika aplikasi selanjutnya. Di bawah ini merupakan tampilan dari halaman edukasi yang datanya telah dimuat menggunakan baris kode di atas.

Gambar 12. Halaman Edukasi

5. Informasi Perlengkapan

```
String jsonStringE =
    await rootBundle.loadString('assets/data/equipments.json');
Map<String, dynamic> jsonDataE = json.decode(jsonStringE);
jsonDataE.forEach((key, value) {
  equipments.add(Equipment.fromJson(key, value));
});

setState(() {
  isLoading = false;
});
```

Gambar 13. Memuat Data Perlengkapan

Fungsi fetchDataEquipment() dalam aplikasi Flutter bertugas untuk memuat dan mengurai data peralatan dari file JSON di direktori aset. Pertama, menggunakan await rootBundle.loadString('assets/data/equipments.json') untuk memuat konten JSON dari file tersebut menjadi string jsonStringE.

Selanjutnya, string JSON diurai menjadi Map<String, dynamic> dengan json.decode(jsonStringE), hasilnya disimpan dalam jsonDataE.

Kemudian, jsonDataE.forEach((key, value) { ... }) digunakan untuk mengiterasi melalui setiap entri dalam map jsonDataE. Di setiap iterasi, metode fromJson dipanggil pada kelas Equipment untuk membuat objek Equipment baru menggunakan key sebagai judul atau identifier, dan value sebagai data detail peralatan dari entri saat ini dalam map. Objek Equipment yang baru dibuat ditambahkan ke dalam daftar equipments.

Setelah proses pengolahan selesai, setState(() { isLoading = false; }) dipanggil untuk mengupdate state aplikasi, dimana isLoading diatur ke false. Ini menandakan bahwa proses memuat data peralatan telah selesai, dan aplikasi siap

untuk menggunakan data peralatan yang telah dimuat. Adapun detail dari kelas Equipment dapat dilihat pada gambar di bawah ini.

```dart
class Equipment {
  final String nama;
  final String deskripsi;
  final List<String> akibatKetinggalan;
  final List<String> solusiKetinggalan;
  final String foto;

  Equipment({
    required this.nama,
    required this.deskripsi,
    required this.akibatKetinggalan,
    required this.solusiKetinggalan,
    required this.foto,
  });

  factory Equipment.fromJson(String nama, Map<String, dynamic> json) {
    return Equipment(
      nama: nama,
      deskripsi: json['deskripsi'],
      akibatKetinggalan:
          List<String>.from(json['akibatKetinggalan'][0]['akibat']),
      solusiKetinggalan:
          List<String>.from(json['solusiKetinggalan'][0]['solusi']),
      foto: json['foto'],
    ); // Equipment
  }
}
```

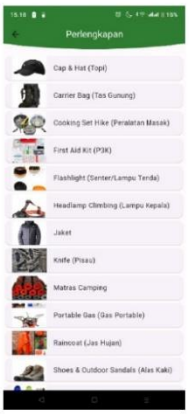
Gambar 14. Kelas Equipment

Kelas Equipment dalam aplikasi Dart atau Flutter digunakan untuk merepresentasikan data peralatan dengan properti seperti nama, deskripsi, akibat ketinggalan peralatan, solusi untuk mengatasi ketinggalan, dan foto peralatan.

Konstruktor Equipment menggunakan named parameter dengan kata kunci required untuk memastikan semua properti yang diperlukan harus disediakan saat membuat objek Equipment.

Metode factory Equipment.fromJson(String nama, Map<String, dynamic> json) digunakan sebagai pabrik untuk membuat objek Equipment dari data dalam bentuk Map<String, dynamic>, biasanya dari representasi JSON. Metode ini membutuhkan nama peralatan sebagai parameter tambahan karena kunci untuk nama tidak tersedia langsung dalam data JSON, tetapi diberikan sebagai input saat pembuatan objek.

Dalam proses pembuatan objek menggunakan fromJson, deskripsi diambil dari kunci 'deskripsi' dalam json, sedangkan akibatKetinggalan dan solusiKetinggalan diambil dari daftar yang mungkin ada dalam json untuk mendetailkan akibat dan solusi dari ketinggalan peralatan. Properti foto diambil langsung dari kunci 'foto' dalam json, yang merujuk ke lokasi atau referensi gambar peralatan. Di bawah ini merupakan tampilan dari halaman informasi perlengkapan yang datanya telah dimuat menggunakan syntax di atas.



Gambar 15. Halaman Perlengkapan

Gambar 16. Halaman Detail Perlengkapan

6. Cek Perlengkapan

```
Future<void> _loadModel() async {
  await _classifier.loadModel('assets/model/model_quantized.tflite');
  setState(() {
    _loading = false;
  });
}

Codeium: Refactor | Explain | Generate Function Comment | ✕ | Codiumate: Options | Test this method
Future<void> _loadCategories() async {
  final String text =
      await rootBundle.loadString('assets/model/labels_image.txt');
  setState(() {
    categories = text.split('\n').map((s) => s.trim()).toList();
    print('Categories Loaded: $categories');
  });
}
```

Gambar 17. Fungsi _loadModel dan _loadCategories

Metode _loadModel() bertanggung jawab memuat model machine learning yang telah dikompresi menggunakan TensorFlow Lite dari lokasi 'assets/model/model_quantized.tflite'. Setelah model terbeban, setState(() { _loading = false; }) dipanggil untuk mengubah variabel _loading menjadi false, menandakan bahwa proses memuat model telah selesai.

Metode _loadCategories() memuat kategori label terkait dengan model machine learning. Teks dari file 'assets/model/labels_image.txt' dimuat, dipisahkan berdasarkan baris, kemudian di-trim dan dikonversi menjadi daftar kategori. Setelah itu, setState() digunakan untuk mengupdate variabel categories dengan daftar kategori yang baru dimuat.

Kedua metode ini menggunakan setState() untuk memperbarui state aplikasi setelah proses memuat data selesai, sehingga aplikasi dapat merespons dengan benar terhadap status memuat dan ketersediaan data untuk operasi machine learning selanjutnya.

```
void _predict() async {
  if (_image == null) return;
  setState(() {
    _loading = true;
  });

  img.Image imageInput = img.decodeImage(_image!.readAsBytesSync())!;
  final List<Map<String, String>> pred =
      await _classifier.classify(imageInput);
  setState(() {
    _predictions = pred;
    _loading = false;
  });
  _groupCategories(pred);
}
```

Gambar 18. Fungsi _predict

Metode _predict() dalam aplikasi Flutter digunakan untuk melakukan prediksi berdasarkan gambar yang dipilih. Jika _image tidak null, proses prediksi dimulai dengan mengatur _loading menjadi true menggunakan setState(), menandakan dimulainya proses prediksi.

Gambar yang dipilih dari _image dibaca sebagai byte menggunakan _image!.readAsBytesSync() dan diuraikan menjadi objek img.Image dengan img.decodeImage(_image!.readAsBytesSync())!. Objek imageInput ini digunakan sebagai input untuk metode classify dari _classifier, yang bertanggung jawab atas klasifikasi menggunakan model machine learning.

Hasil prediksi dari _classifier.classify(imageInput) disimpan dalam variabel pred, yang berisi daftar map dengan format Map<String, String> yang merepresentasikan hasil klasifikasi. Setelah prediksi selesai, setState() dipanggil lagi untuk mengupdate state dengan mengatur _predictions menjadi pred dan _loading kembali ke false, menandakan bahwa proses prediksi telah selesai.

Terakhir, metode _groupCategories(pred) dipanggil untuk mengelompokkan kategori-kategori yang terkait dengan hasil prediksi tersebut, sesuai dengan logika aplikasi yang memerlukan pengelompokan atau penanganan khusus terhadap hasil prediksi.

```
void _groupCategories(List<Map<String, String>> predictions) {
  List<String> presentItems = [];
  List<String> absentItems =
      categories.toList();

  for (var prediction in predictions) {
    double probability = double.parse(prediction['probability']!);
    if (probability > 0.05) {
      int intentIndex =
          int.parse(prediction['intent']!);
      if (intentIndex >= 0 && intentIndex < categories.length) {
        String categoryName = categories[intentIndex];
        presentItems.add(categoryName);
        absentItems.remove(categoryName);
      }
    }
  }

  setState(() {
    _presentItems = presentItems;
    _absentItems = absentItems;
    print('Present Items: $_presentItems');
    print('Absent Items: $_absentItems');
  });
}
```

Gambar 19. Fungsi _groupCategories

Metode _groupCategories(List<Map<String, String>> predictions) dalam aplikasi Flutter digunakan untuk mengelompokkan kategori-kategori berdasarkan hasil prediksi yang diberikan. Pertama, metode ini menginisialisasi dua list: presentItems untuk kategori yang terdeteksi hadir dalam prediksi, dan absentItems yang diisi dengan seluruh kategori dari categories, menandakan kategori yang tidak terdeteksi dalam prediksi.

Selama iterasi melalui predictions, setiap prediction adalah sebuah map yang berisi informasi tentang kategori dan probabilitas dari hasil prediksi. Probabilitas dari prediksi diambil dengan mengonversi string 'probability' menjadi double. Jika probabilitas lebih besar dari 0.05, intent diambil dari string 'intent' dan digunakan untuk mengakses kategori yang sesuai dari categories. Kategori ini kemudian ditambahkan ke dalam presentItems dan dihapus dari absentItems.

Setelah selesai mengelompokkan kategori, setState() dipanggil untuk memperbarui state aplikasi. Pada pemanggilan setState(), _presentItems diperbarui dengan presentItems yang baru, dan _absentItems diperbarui dengan absentItems yang baru. Informasi ini juga dicetak ke konsol untuk tujuan debugging atau pelacakan. Adapun kelas Classifier dapat dilihat pada gambar dibawah ini.

```
class Classifier {
  late Interpreter _interpreter;

  // Load the TFLite model
  Future<void> loadModel(String modelPath) async {
    try {
      _interpreter = await Interpreter.fromAsset(modelPath);
      print('Model loaded successfully');
    } catch (e) {
      print('Failed to load model: $e');
    }
  }
```

Gambar 20. Kelas Classifier

Kelas Classifier dalam aplikasi Flutter mengklasifikasikan gambar menggunakan model TensorFlow Lite (TFLite), dengan menggunakan paket tflite_flutter untuk mengakses model TFLite dan paket image dari Dart untuk pemrosesan gambar. Metode loadModel(String modelPath) digunakan untuk memuat model TFLite dari path yang diberikan. Model dimuat secara asynchronous menggunakan Interpreter.fromAsset(modelPath) dan disimpan dalam variabel _interpreter. Jika pembebanan model berhasil, pesan "Model loaded successfully" dicetak ke konsol; jika tidak, pesan error ditampilkan.

```
Future<List<Map<String, String>>> classify(img.Image image) async {
  // Preprocess the image
  var input = _preprocessImage(image);

  // Output tensor should match the model's specification
  var output = List.filled(1 * 15, 0.0).reshape([1, 15]);

  // Run inference
  try {
    _interpreter.run(input, output);
    return _postProcess(output);
  } catch (e) {
    print('Error during inference: $e');
    return [];
  }
}
```

Gambar 21. Fungsi Classify

Metode classify(img.Image image) pada kelas Classifier digunakan untuk melakukan klasifikasi pada gambar input. Proses dimulai dengan preprocessing gambar menggunakan _preprocessImage(image) untuk mengubahnya menjadi tensor float32 yang sesuai dengan format yang dibutuhkan oleh model. Output dari model diinisialisasi sebagai list dengan nilai awal 0 dan direshape ke dimensi yang sesuai. Kemudian, _interpreter.run(input, output) digunakan untuk menjalankan inferensi pada model TensorFlow Lite (TFLite). Hasil inferensi tersebut kemudian diproses menggunakan _postProcess(output) untuk menghasilkan prediksi dalam bentuk daftar map yang berisi intent (kategori) dan probabilitas prediksi.

```dart
List<List<List<List<double>>>> _preprocessImage(img.Image image) {
  var resizedImage = img.copyResize(image, width: 256, height: 256);
  var input = List.generate(
    1,
    (_) => List.generate(
      256,
      (_) => List.generate(
        256,
        (_) => List.generate(3, (_) => 0.0),
      ), // List.generate
    ), // List.generate
  ); // List.generate

  for (int y = 0; y < 256; y++) {
    for (int x = 0; x < 256; x++) {
      var pixel = resizedImage.getPixel(x, y);

      // Set RGB components directly from pixel attributes
      input[0][y][x][0] = pixel.r / 255.0; // Red component
      input[0][y][x][1] = pixel.g / 255.0; // Green component
      input[0][y][x][2] = pixel.b / 255.0; // Blue component
    }
  }
  return input;
}
```

Gambar 22. Fungsi _preprocessImage

Metode _preprocessImage(img.Image image) digunakan untuk mengubah gambar menjadi format tensor float32 yang dibutuhkan oleh model. Gambar diubah ukurannya menjadi 256x256 piksel, dan setiap nilai RGB dari setiap piksel di-normalisasi menjadi rentang 0 hingga 1.

```dart
List<Map<String, String>> _postProcess(List<dynamic> output) {
  var results = <Map<String, String>>[];
  if (output.isNotEmpty) {
    var predictions =
        output[0].cast<double>();
    for (int i = 0; i < predictions.length; i++) {
      results.add({
        'intent': i.toString(),
        'probability': predictions[i].toStringAsFixed(5),
      });
    }
    results.sort((a, b) => double.parse(b['probability']!)
        .compareTo(double.parse(a['probability']!)));
  }
  return results;
}
```

Gambar 23. Fungsi _postProcess

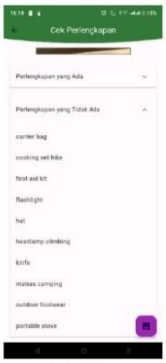
Metode _postProcess(List<dynamic> output) memproses output dari model. Output diasumsikan sebagai array dengan satu baris dan 15 kolom. Setiap nilai prediksi diurutkan berdasarkan probabilitasnya, dan hasilnya dikembalikan dalam bentuk daftar map dengan intent (indeks kategori) dan probabilitasnya yang diformat dengan lima angka di belakang koma.

Kelas Classifier ini digunakan untuk klasifikasi gambar menggunakan TensorFlow Lite (TFLite) dalam aplikasi Flutter. Ini meliputi proses memuat model, preprocessing gambar, inferensi, dan postprocessing hasil prediksi. Kelas ini dirancang untuk aplikasi yang memerlukan klasifikasi gambar real-time atau berbasis input pengguna, mengoptimalkan penggunaan TensorFlow Lite dan Dart

untuk operasi machine learning. Berikut merupakan tampilan dari halaman cek perlengkapan yang dijalankan menggunakan fungsi yang telah dibuat di atas.

Gambar 24. Halaman Cek Perlengkapan 1

Gambar 25. Halaman Cek Perlengkapan 2



Gambar 26. Halaman Cek Perlengkapan 3

7. Chatbot

```
void _addMessage(String text) async {
  setState(() {
    _messages.insert(0, Message(text: text, isUser: true));
    _controller.clear();
    if (!_isWaitingForResponse) {
      _isWaitingForResponse = true;
      // Mendapatkan response dari chatbot
      chatbotResponse(text).then((response) {
        setState(() {
          _messages.insert(0, Message(text: response, isUser: false));
          _isWaitingForResponse = false;
        });
      });
    }
  });
}
```

Gambar 27. Method _addMessage

Metode _addMessage(String text) dalam aplikasi Flutter digunakan untuk memfasilitasi interaksi pengguna dengan chatbot secara real-time melalui antarmuka obrolan. Saat pengguna mengirim pesan baru, aplikasi menggunakan setState() untuk memperbarui antarmuka dengan menambahkan pesan pengguna ke bagian atas daftar pesan. Setelah itu, controller teks input dibersihkan untuk menjaga kebersihan antarmuka. Jika tidak sedang menunggu respons sebelumnya, aplikasi mengubah status _isWaitingForResponse menjadi true, menandakan bahwa sedang menunggu respons dari chatbot. Selanjutnya, aplikasi memanggil fungsi chatbotResponse(text) untuk memperoleh respons chatbot berdasarkan pesan pengguna.

Ketika respons tersedia, pesan dari chatbot ditambahkan ke daftar pesan dengan menandai bahwa itu bukan pesan dari pengguna. Setelah proses selesai, status _isWaitingForResponse kembali diatur menjadi false, memungkinkan aplikasi untuk menerima pesan baru dari pengguna dan melanjutkan interaksi. Dengan demikian, metode _addMessage menghubungkan antara input pengguna dan proses komunikasi dengan chatbot dalam aplikasi Flutter, memastikan pengalaman pengguna yang responsif dan terhubung secara real-time.

```
Future<List<String>> tokenizeAndLemmatize(String sentence) async {
  final tokenizer = Tokenizer({',', ' '});
  final lemmatizer = PorterStemmer();
  final c = StreamController<String>();

  c.add(sentence);
  c.close();

  final tokens = await c.stream.transform(tokenizer.transformer).toList();
  final sentence_words = [for (var word in tokens) lemmatizer.stem(word)];

  return sentence_words;
}
```

Gambar 28. Fungsi tokenizeAndLemmatize

Metode tokenizeAndLemmatize(String sentence) mengambil sebuah kalimat sebagai input untuk melakukan tokenisasi dan lemmatisasi terhadap setiap kata di dalamnya. Tokenisasi dilakukan menggunakan Tokenizer dengan delimiter yang telah ditetapkan, dan setelah itu, kata-kata hasil tokenisasi dilemmatisasi menggunakan PorterStemmer. Hasilnya dikembalikan dalam bentuk daftar kata-kata yang telah diproses.

```
Future<List<List<int>>> bow(String sentence) async {
  final String wordsfile =
      await rootBundle.loadString('assets/model/texts.txt');
  final words = wordsfile.split('\n').map((label) => label.trim()).toList();
  // Tokenize and Lemmatize pattern
  final sentenceWords = await tokenizeAndLemmatize(sentence);
  // Bag of words - matrix of N words, vocabulary matrix
  final bag = List<int>.filled(words.length, 0);
  for (var s in sentenceWords) {
    for (var i = 0; i < words.length; i++) {
      final w = words[i];
      if (w == s) {
        // Assign 1 if current word is in the vocabulary position
        bag[i] = 1;
      }
    }
  }
  final reshapedBag = [bag];
  return Future.value(reshapedBag);
}
```

Gambar 29. Fungsi bow

Metode bow(String sentence) menerima sebuah kalimat sebagai input. Pertama, metode ini memuat daftar kata-kata dari file teks yang tersedia. Selanjutnya, menggunakan metode tokenizeAndLemmatize(sentence), kata-kata dalam kalimat diolah untuk tokenisasi dan lemmatisasi. Setelah itu, metode membentuk model Bag-of-Words (BoW) dengan menghasilkan vektor biner yang

menunjukkan kehadiran kata-kata dalam kumpulan kata-kata yang telah ditentukan sebelumnya.

```
Future<List<Map<String, String>>> predictClass(String sentence) async {
  final String data = await rootBundle.loadString('assets/model/labels.txt');
  final classes = data.split('\n').map((label) => label.trim()).toList();
  final _interpreter =
      await tfl.Interpreter.fromAsset('assets/model/model.tflite');

  final input = await bow(sentence);
  List<double> output1en = List<double>.filled(classes.length, 0);
  final output = [output1en];

  _interpreter.run(input, output);

  final double errorThreshold = 0.1;
  final List<Map<String, String>> returnList = [];

  for (var i = 0; i < output[0].length; i++) {
    final double probability = output[0][i];
    if (probability > errorThreshold) {
      returnList.add({
        "intent": classes[i],
        "probability": probability.toString(),
      });
    }
  }

  _interpreter.close();

  return returnList;
}
```

Gambar 30. Fungsi predictClass

Metode predictClass(String sentence) mengambil sebuah kalimat sebagai input. Pertama, metode memuat kelas-kelas dari file teks yang menyimpan label, dan selanjutnya memuat model TensorFlow Lite (TFLite) dari file yang tersedia. Metode bow(sentence) digunakan untuk mengubah input menjadi representasi dalam bentuk Bag-of-Words (BoW). Kemudian, model TFLite dijalankan menggunakan input BoW untuk menghasilkan output, yang berupa probabilitas dari setiap kelas. Hanya output yang melewati threshold kesalahan yang ditambahkan ke daftar respons yang akan dikirimkan kembali ke pengguna.

```
Future<String> getResponse(List<Map<String, dynamic>> intents) async {
  final jsonString = await rootBundle.loadString('assets/model/data.json');
  final intentsJson = json.decode(jsonString);
  final tag = intents[0]['intent'];
  final list_of_intents = intentsJson['intents'] as List<dynamic>;
  String result = '';

  for (var i in list_of_intents) {
    if (i['tag'] == tag) {
      final responses = i['responses'] as List<dynamic>;
      result = responses[Random().nextInt(responses.length)];
      break;
    }
  }

  return result;
}
```

Gambar 31. Fungsi getResponse

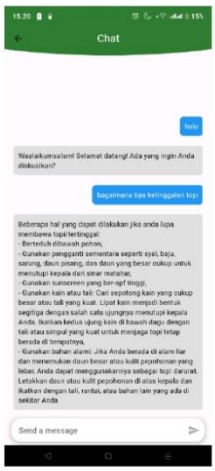
Metode getResponse(List<Map<String, dynamic>> intents) digunakan untuk memilih respons yang sesuai berdasarkan label atau intent yang diberikan dalam daftar intents yang dihasilkan oleh predictClass(String sentence). Ini

dilakukan dengan memuat daftar intents dari file JSON, mencocokkan intent dengan tag yang sesuai, dan memilih respons acak dari daftar respons yang tersedia.

```
Future<String> chatbotResponse(String msg) async {
  final ints = await predictClass(msg);
  print("ini ints: $ints");
  const double ERROR_THRESHOLD = 0.5;
  String res = '';

  if (ints.isNotEmpty &&
      double.parse(ints[0]['probability']!) >= ERROR_THRESHOLD) {
    res = await getResponse(ints);
    print("probability ${double.parse(ints[0]['probability']!)}");
  } else {
    res =
        "Maaf, saya tidak mengerti pertanyaan Anda. Bisakah Anda menjelaskan lebih lanjut?";
  }

  return res;
}
```

Gambar 32. Fungsi chatbotResponse

Metode chatbotResponse(String msg) adalah titik masuk utama untuk interaksi dengan chatbot. Ini memanggil predictClass(msg) untuk memperoleh prediksi kelas atau intent dari input pesan pengguna. Jika probabilitas prediksi melewati threshold kesalahan yang ditentukan, maka metode getResponse(ints) dipanggil untuk mendapatkan respons yang sesuai. Jika tidak, chatbot memberikan respons default yang meminta pengguna untuk memberikan informasi lebih lanjut. Di bawah ini merupakan tampilan dari halaman chatbot.



Gambar 33. Halaman Chatbot

Lampiran 6. Sertifikat HKI yang Terbit

REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC00202459822, 4 Juli 2024
Pencipta		
Nama	:	**Koandres, Muhammad Fikri Hidayattullah, S.T., M.Kom. dkk**
Alamat	:	Jl. Ababil Indah No.25 RT/RW 010/010, Kelurahan Randugunting, Kecamatan Tegal Selatan, Kota Tegal, Propinsi Jawa Tengah 52131, Tegal Selatan, Tegal, Jawa Tengah, 52131
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**Pusat Penelitian dan Pengabdian Masyarakat (P3M) Politeknik Harapan Bersama**
Alamat	:	Jalan Mataram No. 9, Pesurungan Lor, Kecamatan Margadana 52142 , Margadana, Tegal, Jawa Tengah 52142
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Program Komputer**
Judul Ciptaan	:	**SiMun: Virtual Assistant Software Pendakian Gunung**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	4 Juli 2024, di Tegal
Jangka waktu pelindungan	:	Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.
Nomor pencatatan	:	000635180

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n. MENTERI HUKUM DAN HAK ASASI MANUSIA
DIREKTUR JENDERAL KEKAYAAN INTELEKTUAL
u.b
Direktur Hak Cipta dan Desain Industri

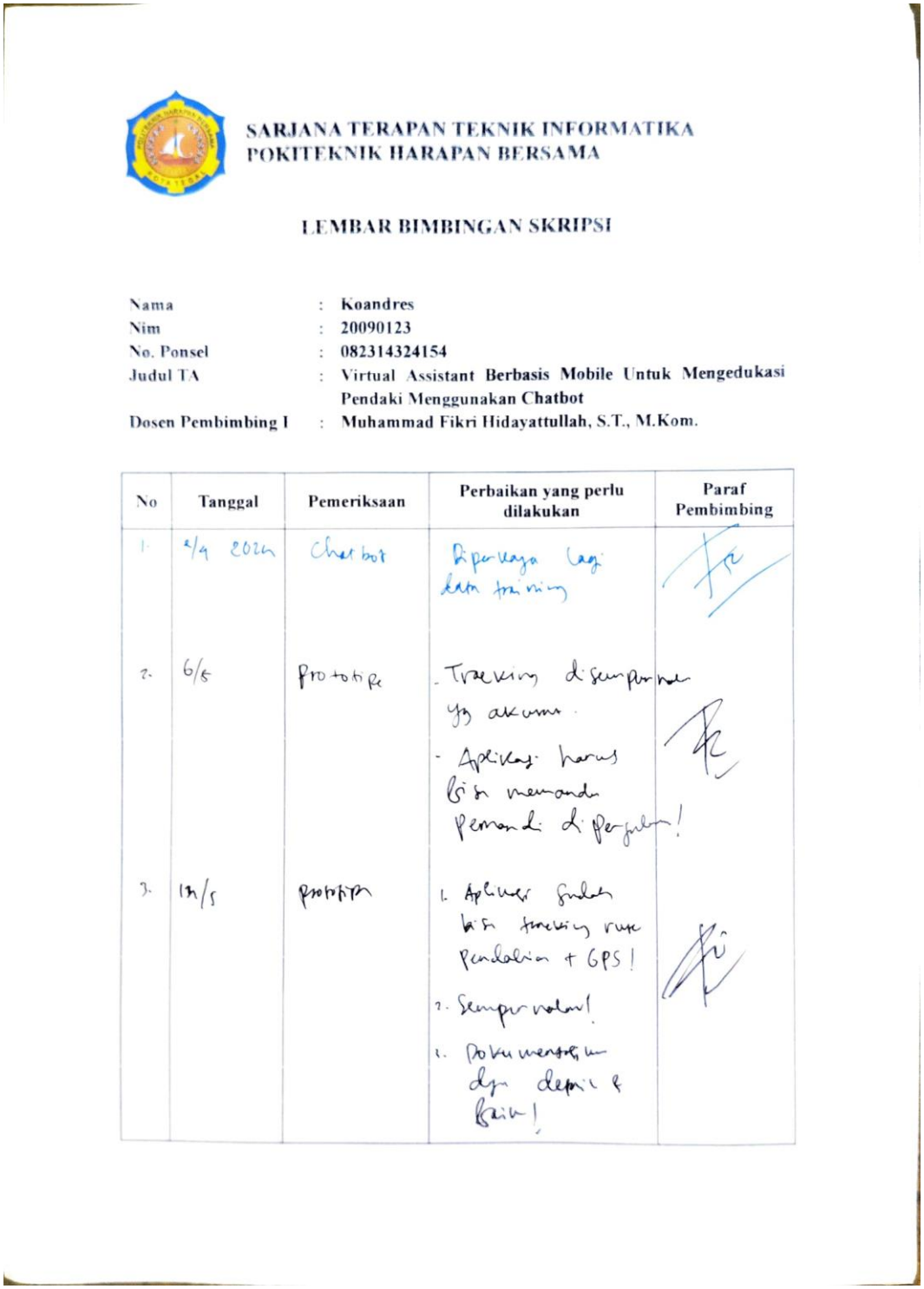
IGNATIUS M.T. SILALAHI
NIP. 196812301996031001

Disclaimer:
Dalam hal pemohon memberikan keterangan tidak sesuai dengan surat pernyataan, Menteri berwenang untuk mencabut surat pencatatan permohonan.

LAMPIRAN PENCIPTA

No	Nama	Alamat
1	Koandres	Jl. Ababil Indah No.25 RT/RW 010/010, Kelurahan Randugunting, Kecamatan Tegal Selatan, Kota Tegal, Propinsi Jawa Tengah 52131, Tegal Selatan, Tegal
2	Muhammad Fikri Hidayattullah, S.T., M.Kom.	Jl. Glatik No.68, Kelurahan Randugunting, Kecamatan Tegal Selatan, Kota Tegal. Propinsi Jawa Tengah 53131, Tegal Selatan, Tegal
3	Sharfina Febbi Handayani, S.Kom., M.Kom.	Jl. Brayan 1 RT 02 RW 06 Desa Grobog Kulon , Kecamatan Pangkah, Kabupaten Tegal, Propinsi Jawa Tengah 52471, Pangkah, Tegal

Lampiran 7. Lembar Bimbingan



SARJANA TERAPAN TEKNIK INFORMATIKA
POKITEKNIK HARAPAN BERSAMA

LEMBAR BIMBINGAN SKRIPSI

Nama	:	Koandres
Nim	:	20090123
No. Ponsel	:	082314324154
Judul TA	:	Virtual Assistant Berbasis Mobile Untuk Mengedukasi Pendaki Menggunakan Chatbot
Dosen Pembimbing I	:	Muhammad Fikri Hidayattullah, S.T., M.Kom.

No	Tanggal	Pemeriksaan	Perbaikan yang perlu dilakukan	Paraf Pembimbing
1.	2/4 2024	Chat bot	Diperluas lagi data training	
2.	6/5	Prototipe	- Tracking disempurnakan yg akurat - Aplikasi harus bisa memanda pemandi di perjalan!	
3.	13/5	Prototipe	1. Aplikasi sudah bisa tracking rute pendakian + GPS! 2. Sempurnakan! 3. Dokumentasikan dgn detail & baik!	

SARJANA TERAPAN TEKNIK INFORMATIKA
POKITEKNIK HARAPAN BERSAMA

No	Tanggal	Pemeriksaan	Perbaikan yang perlu dilakukan	Paraf Pembimbing
4.	26/6	Aplikasi	Lanjutkan pembuatan Manual book + dok. teknikal	
5.	27/6	Manual Book & Dokumen Teknikal	Tambahkan capture tampilan di setiap sintaks yg dijelaskan	
6.	28/6	Manual Book & Dok. Tek.	Silakan diajukan HKI-nya	
7.	8/7	Laporan	- Type - Verifikasi flowchart	
8.	9/7	Laporan	Siap	
9.	10/7	Lap	ACC. Siap disidangkan	

SARJANA TERAPAN TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA

No	Tanggal	Pemeriksaan	Perbaikan yang perlu dilakukan	Paraf Pembimbing

Tegal, 12 Juli 2024
Dosen Pembimbing I

Muhammad Fikri Hidayattullah, S.T., M.Kom.
NIPY. 09.016.307

SARJANA TERAPAN TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA

LEMBAR BIMBINGAN SKRIPSI

Nama : **Koandres**
Nim : **20090123**
No. Ponsel : **082314324154**
Judul TA : ***Virtual Assistant* Berbasis *Mobile* Untuk Mengedukasi Pendaki Menggunakan *Chatbot***
Dosen Pembimbing II : **Sharfina Febbi Handayani, S.Kom., M.Kom.**

No	Tanggal	Pemeriksaan	Perbaikan yang perlu dilakukan	Paraf Pembimbing
1.	21/03/2024	Rancangan Chatbot dan desain tampilan di mobile	- Fokus menyelesaikan model chatbot - Deployment chatbot ke mobile	Sharfina
2.	28/03/2024	Chatbot Pendakian	- Dilanjutkan dengan deployment tampilan mobilenya	Sharfina
3.	04/04/2024	Tampilan rincian informasi pada mobile	- Lanjutkan untuk melengkpai informasi gunung dengan referensi yang sesuai	Sharfina
4.	26/04/2024	Informasi gunung dan halaman peta yang sudah ditambahkan	- Fokus ke pengembangan peta yakni fitur tracking user	Sharfina
5.	03/05/2024	Integrasi chatbot ke aplikasi mobile	- menyelesaikan deployement model nya	Sharfina

SARJANA TERAPAN TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA

No	Tanggal	Pemeriksaan	Perbaikan yang perlu dilakukan	Paraf Pembimbing
6.	17/05/2024	Peta offline yang sudah dikembangkan	- Aplikasi sudah bisa masuk tahap pengajuan HKI - Untuk fitur cek perlengkapan jika masih belum bisa diintegrasikan bisa dimasukkan ke saran	
7.	20/06/2024	Fitur Cek Perelengkapan dan menu edukasi serta perlengkapan	- Lanjutkan untuk pembuatan manual book dan dokumen teknis	
8.	26/06/2024	Manual Book dan Dokumen Teknis	- manual book dan dokumen teknis sudah cukup - lanjut ke pengajuan HKI	
9.	01/07/2024	Laporan Bab 1	- Lanjutkan ke bab 2 dan 3	
10.	05/07/2024	Laporan Bab 2 dan 3	- Acc Laporan	

Tegal, 11 Juli 2024
Dosen Pembimbing II

Sharfina Febbi Handayani, S.Kom., M.Kom.
NIPY. 08.020.451